TI-Innovator™ Technology Guidebook

Per sapere di più sulle tecnologie TI, consultare la guida online all'indirizzo
education.ti.com/eguide

Informazioni Importanti

A meno che non sia indicato diversamente nella licenza fornita con un programma, Texas Instruments, relativamente ai programmi o ai materiali di riferimento, non rilascia alcuna garanzia, né esplicita né implicita, ivi comprese, a mero titolo esemplificativo, garanzie implicite di commerciabilità e idoneità per uno scopo particolare. Tali materiali, quindi, sono disponibili solo "così come forniti". In nessun caso Texas Instruments potrà essere ritenuta responsabile di danni speciali, collaterali, accidentali o conseguenti, collegati o riconducibili all'acquisto o all'utilizzo di tali materiali. A prescindere da qualunque tipo di azione legale eventualmente intrapresa, la responsabilità di Texas Instruments è limitata all'importo indicato nella licenza del programma. Texas Instruments, inoltre, non potrà essere ritenuta responsabile per eventuali reclami, di qualunque tipo, riguardanti l'utilizzo di tali materiali da parte di terzi.

Ulteriori informazioni con la TI-Innovator™ Technology eGuide

Parti di questo documento fanno riferimento alla TI-Innovator™ Technology eGuide per maggiori dettagli. La eGuide è un manuale basato sul web TI-Innovator™ contenente informazioni concernenti:

- Programmazione con la famiglia di calcolatrici grafiche TI CE e la tecnologia TI-Nspire™, inclusi programmi campione.
- Moduli I/O disponibili e rispettivi comandi.
- Componenti della basetta sperimentale disponibili e rispettivi comandi.
- La TI-array RGB e i relativi comandi.
- TI-Innovator™ Rover disponibile e rispettivi comandi.
- Collegamento per aggiornare il software TI-Innovator™ Sketch.
- Attività didattiche gratuite per TI-Innovator™ Hub.

Indice

TI-Innovator™ Hub Guida introduttiva

L’Hub TI-Innovator™ è il fiore all’occhiello della tecnologia TI-Innovator™, un kit di progettazione che estende la funzionalità delle calcolatrici grafiche Texas Instruments (TI) per rendere la programmazione e la progettazione accessibili agli studenti in aula.

Gli argomenti della guida introduttiva comprendono:

- Panoramica dell’Hub
- Contenuto della confezione
- Collegamento dell’Hub TI-Innovator™
- Aggiornamento del software dell’Hub
- Programmazione dell’Hub sulla calcolatrice grafica TI CE
- Programmazione dell’Hub nella tecnologia TI-Nspire™ CX
- Moduli I/O TI-Innovator™
- Kit della basetta sperimentale TI-Innovator™
- Utilizzo di una fonte di alimentazione ausiliaria
- Risoluzione dei problemi
- Precauzioni generali

Hub TI-Innovator™ Panoramica

La TI-Innovator™ Hub consente di utilizzare la calcolatrice grafica TI compatibile o il software per computer TI-Nspire™ CX per controllare i componenti, leggere i sensori e creare un’esperienza di apprendimento unica.

- È possibile comunicare con l’Hub mediante i comandi di programmazione TI Basic.
- Gli host che sono compatibili con l’ TI-Innovator™ Hub sono i seguenti:
  - Famiglia di calcolatrici grafiche TI CE (TI-83 Premium CE, TI-84 Plus CE e TI-84 Plus CE-T) con sistema operativo installato versione 5.3 o successiva. Può anche essere necessario installare o aggiornare l’app Hub che contiene il menu Hub.
  - Palmari TI Nspire™ CX o TI Nspire™ CX CAS con sistema operativo installato versione 4.5 o successiva
  - Software per computer TI Nspire™ versione 4.5 o successiva
- **TI-Innovator™ Hub.** Comunica con l’host, i componenti di bordo dell’Hub e i componenti esterni collegati. Inoltre fornisce energia ai componenti esterni.
- **TI-Innovator™ Componenti**. Questi componenti, venduti separatamente, comprendono sensori, motori e LED che si collegano all’Hub attraverso le porte I/O e i connettori della basetta sperimentale.

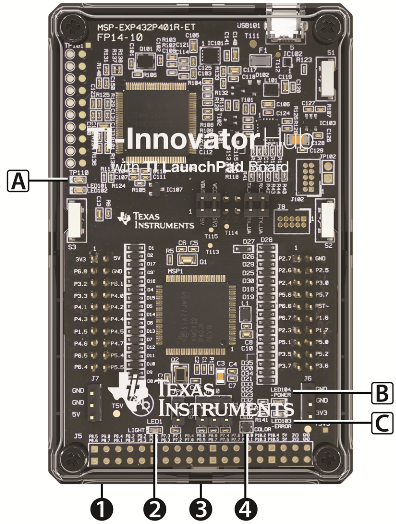
Ulteriori informazioni

Per un riepilogo delle precauzioni da prendere quando si usa l’Hub e i suoi componenti, consultare *Precauzioni generali* (pagina 27).

Per informazioni su accessori, moduli esterni e componenti della basetta sperimentale, visitare education.ti.com/go/innovator.

Contenuto della confezione

TI-Innovator™ Hub con componenti di bordo



❶ Un sensore di intensità luminosa nella parte inferiore dell’Hub può essere letto come "BRIGHTNESS" nelle stringhe di comando dell’Hub.

❷ Il LED rosso è indirizzabile come "LIGHT" nelle stringhe di comando dell’Hub.

❸ L’altoparlante (sul lato posteriore dell’Hub, non illustrato) è indirizzabile come "SOUND" nelle stringhe di comando Hub.

❹ Il LED rosso-verde-blu è indirizzabile come "COLOR" nelle stringhe di comando Hub.

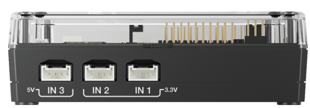
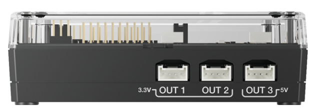
Sulla parte frontale dell’Hub sono inoltre visibili:

A LED verde di alimentazione ausiliaria

B LED verde di alimentazione

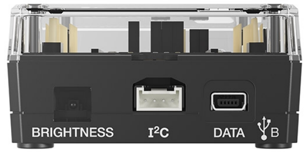
C LED rosso di errore

Porte integrate

Lato sinistro - Tre porte per la raccolta di dati o informazioni sullo stato dai moduli di input: • **IN 1** e **IN 2** forniscono una tensione di 3,3 V. • **IN 3** fornisce una tensione di 5 V.	
Lato destro - Tre porte per controllare i moduli di output: • **OUT 1** e **OUT 2** forniscono una tensione di 3,3 V. • **OUT 3** fornisce una tensione di 5 V.	

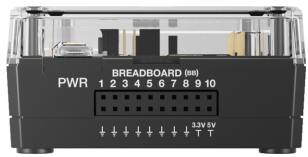
Parte inferiore - Sensore di intensità luminosa (descritto in precedenza) e due porte:

- **Porta I^{2}C** per collegare periferiche che utilizzano il protocollo di comunicazione I^{2}C.
- **DATI** Porta Mini-B utilizzata con il cavo appropriato per collegare una calcolatrice grafica compatibile o un computer per i dati e l'alimentazione.

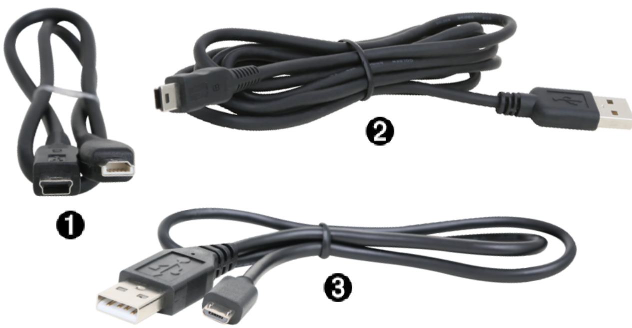


Parte superiore - Due connettori:

- Connettore USB-Micro (**PWR**) per l'alimentazione ausiliaria richiesta da alcuni componenti. Utilizzato anche per aggiornare il software interno dell'Hub.
- Connettore della basetta sperimentale con 20 pin contrassegnati per la comunicazione con i componenti connessi. Una basetta sperimentale e i cavi di accoppiamento sono in dotazione con il TI-Innovator™ Breadboard Pack, venduto separatamente.



Cavi USB



❶ USB Unit-to-Unit (Mini-A to Mini-B) - Collega l'Hub a una calcolatrice grafica TI CE o a un palmare TI-Nspire™ CX.

❷ USB Standard A to Mini-B - Collega l'Hub a un computer in cui viene eseguito il software TI-Nspire™ CX.

❸ USB Standard A to Micro - Collega la porta **PWR** dell'Hub a un alimentatore elettrico approvato da TI e richiesto da alcune periferiche.

Alimentazione ausiliaria

TI Wall Charger - Genera energia mediante l' TI-Innovator™ Hub per componenti come i motori, che richiedono una potenza aggiuntiva.

Il External Battery Pack opzionale può anche fornire alimentazione ausiliaria.

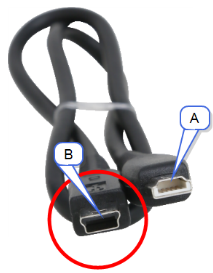
Nota: il LED di alimentazione ausiliaria sull'Hub emette una luce per indicare che l'Hub è alimentato.

Collegamento dell' TI-Innovator™ Hub

La TI-Innovator™ Hub si collega via cavo USB a un computer o a una calcolatrice grafica. Il collegamento consente all'Hub di essere alimentato e di scambiare dati con l'host.

Nota: alcune periferiche, come i motori, potrebbero richiedere un alimentatore ausiliario. Per ulteriori informazioni, vedere Utilizzo di una fonte di alimentazione ausiliaria (pagina 24).

Collegamento a una calcolatrice grafica



1. Identificare il connettore "**B**" sul USB Unit-to-Unit (Mini-A to Mini-B) . Su ogni estremità di questo cavo è stata impressa una lettera.
2. Inserire il connettore "**B**" nelle porta **DATI** nella parte inferiore dell' TI-Innovator™ Hub.
3. Inserire l'estremità libera del cavo (il connettore "**A**") nella porta USB della calcolatrice.

Hub collegato a
calcolatrice grafica TI CE

Hub collegato a
palmare TI-Nspire™ CX

4. Accendere la calcolatrice se ancora non è stata accesa.

   Il LED di alimentazione sull’Hub emette una luce verde per indicare che il dispositivo è alimentato.

Collegamento a un computer in cui viene eseguito il software TI-Nspire™ CX.

1. Identificare il connettore "**B**" sul USB Standard A to Mini-B cavo per Windows®/Mac®. Su ogni estremità di questo cavo è stata impressa una lettera.
2. Inserire il connettore "**B**" nelle porta **DATI** nella parte inferiore dell’ TI-Innovator™ Hub.

3. Inserire l’estremità libera del cavo (il connettore "**A**") in una porta USB del computer.

   Il LED di alimentazione sull’Hub emette una luce verde per indicare che il dispositivo è alimentato.

Aggiornamento del software dell’Hub

La TI-Innovator™ Hub contiene il software, TI-Innovator™ Sketch, che interpreta i comandi dell’Hub e comunica con i dispositivi di bordo e i moduli collegati. Uno strumento basato sul web consente di aggiornare lo Sketch. Le versioni aggiornate contengono le correzioni dei bug e assicurano che l’ TI-Innovator™ Hub possa comunicare con i componenti più recenti.

Per ottenere l’ultima versione del TI-Innovator™ Sketch, visitare il sito seguente:

https://education.ti.com/go/innovator

Domande relative al software Hub

Che cos'è il sketch TI-Innovator™?

Il "sketch" è il software installato su TI-Innovator™ Hub che comunica con la calcolatrice grafica, elabora i comandi e controlla i componenti esterni.

Occorre aggiornare il sketch su TI-Innovator™ Hub?

Per risultati ottimali, utilizzare sempre l'ultima versione del sketch TI-Innovator™. Per tenersi informati sugli aggiornamenti di TI-Innovator™ Hub, registrare il proprio prodotto all'indirizzo education.ti.com/register oppure controllare il sito Web TI-Innovator™ all'indirizzo education.ti.com/go/innovator.

Che cos'è l'ultima versione del sketch?

Per risultati ottimali, utilizzare sempre l'ultima versione del sketch TI-Innovator. La versione più recente del sketch è sempre disponibile all'indirizzo education.ti.com/go/innovator.

Perché occorre aggiornare il sketch?

Esistono due motivi per l'aggiornamento del sketch.

1. Ottenere la versione più recente di TI che potrebbe includere nuove funzionalità.
2. Ripristinare il sketch TI dopo aver caricato un sketch personalizzato. Questa opzione interessa solo gli utenti avanzati che utilizzano un sketch alternativo.

Com'è possibile caricare il sketch su TI-Innovator™ Hub?

È possibile aggiornare il sketch mediante il software di aggiornamento di TI-Innovator Hub. Questo software può essere scaricato gratuitamente dal sito Web TI.

È possibile aggiornare contemporaneamente più TI-Innovator Hub?

Il software di aggiornamento di TI-Innovator Hub consente di aggiornare un unico Hub alla volta. Tuttavia, l'applicazione è stata progettata per consentire l'aggiornamento di diversi Hub senza dover riavviare il software.

È possibile modificare il sketch in dotazione con TI-Innovator™ Hub in modo da aggiungere funzionalità e continuare comunque a utilizzare la calcolatrice TI? Il sketch è open source?

Il codice per il sketch caricato su TI-Innovator™ non è stato pubblicato per essere modificato o alterato da altri. Per mantenere la compatibilità tra TI-Innovator™ Hub e i prodotti della calcolatrice TI, utilizzare esclusivamente il sketch pubblicato ufficialmente per TI-Innovator™ Hub.

Programmazione dell'Hub sulla calcolatrice grafica TI CE

Nota: Queste istruzioni valgono per le calcolatrici grafiche TI CE. Per istruzioni analoghe applicabili alla tecnologia TI-Nspire™ CX, fare riferimento a Programmazione dell'Hub nella tecnologia TI-Nspire™ CX (page 13).

La TI-Innovator™ Hub risponde ai comandi di programmazione TI Basic quali **Send** e **Get**.

- **Send** - Invia le stringhe di comando all'Hub per controllare i dispositivi o richiedere informazioni.
- **Get** - Recupera le informazioni richieste dall'Hub.
- **eval** - Fornisce il risultato di un'espressione sotto forma di stringa di caratteri. Utile in modo particolare all'interno della stringa di comando dell'Hub nei comandi **Send**.
- **Wait** - Interrompe l'esecuzione del programma per un numero di secondi specificato.

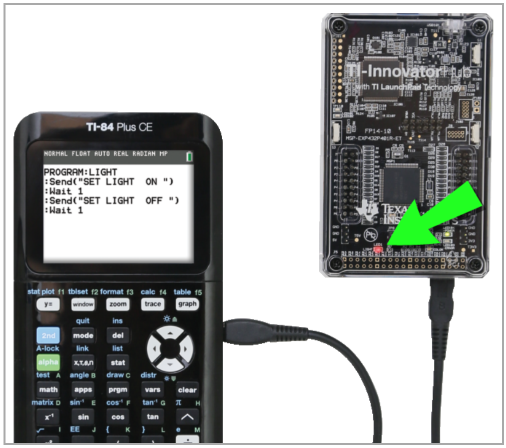
Esempi di codice: calcolatrice grafica TI CE

Azione desiderata	**Codice del programma**
Attivare il LED rosso integrato ("LIGHT").	`Send("SET LIGHT ON")`
Riprodurre un segnale acustico da 440 Hz nell'altoparlante integrato ("SOUND") per 2 secondi.	`Send("SET SOUND 440 TIME 2")`
Attivare l'elemento blu del LED rosso-verde-blu integrato ("COLOR") impostando una luminosità del 100%.	`Send("SET COLOR.BLUE 255")`
Leggere e visualizzare il valore corrente del sensore di luminosità integrato ("BRIGHTNESS"). L'intervallo è compreso tra 0% e 100%.	`Send("READ BRIGHTNESS")` `Get(A):Disp A`

Programma di esempio per far lampeggiare un LED di bordo

Il seguente programma della calcolatrice grafica TI CE utilizza i comandi **Send** e **Wait** per far lampeggiare il LED rosso integrato nell'Hub. I comandi sono contenuti in un loop "For...End" che ripete il ciclo intermittente ON/OFF per 10 iterazioni.

```
PRGM: BLINK
For(N,1,10)
Send("SET LIGHT ON")
Wait 1
Send("SET LIGHT OFF")
Wait 1
End
```



Come creare ed eseguire un programma

Nota: Questa è la versione breve delle istruzioni. Per istruzioni più dettagliate su come creare ed eseguire i programmi, fare riferimento a *Programmazione TI-Basic per la calcolatrice grafica TI CE*. Le istruzioni sono accessibili dalla TI-Innovator™ Technology eGuide (page ii).

Prima di cominciare

- Consultare i requisiti di sistema (page 2e aggiornare il sistema operativo della calcolatrice e l'app Hub, se necessario. L'aggiornamento può essere eseguito dal software TI Connect™ CE o da un'altra calcolatrice aggiornata.

  Nota: È possibile scaricare l'app TI-Innovator™ Hub dal sito Web TI all'indirizzo education.ti.com/latest.

Per creare un nuovo programma nella calcolatrice grafica TI CE:

1. Nella schermata iniziale, premere [prgm], selezionare **New** e premere [enter].
2. Digitare un nome per il programma, come "SOUNDTST" e premere [enter].

   Viene visualizzato l'editor di programma con un modello per il codice del programma.

3. Introdurre le righe di codice che compongono il programma.
   - È necessario utilizzare il menu Hub per immettere i comandi TI-Basic, quali **Send** e **Get**. (Premere [prgm] e selezionare **Hub**).
   - È possibile immettere il comando dell'Hub strinSetgs e parametri quali **"SET LIGHT ON"** utilizzando il menu o la tastiera. Se le stringhe vengono digitate, ricordarsi di distinguere tra maiuscole e minuscole.
   - Al termine di ogni riga, premere [enter]. Ogni nuova riga viene preceduta automaticamente da due punti (:).

- Utilizzare i tasti freccia per spostarsi all’interno di un programma. Premere [del] per eliminare o premere [2nd] [ins] per inserire.

Per chiudere l’editor di programma

▶ Premere [2nd] [quit] per tornare alla schermata iniziale.

Il programma può essere riaperto utilizzando il tasto[prgm].

Per eseguire il programma:

1. Verificare che l’ TI-Innovator™ Hub sia collegato alla calcolatrice.
2. Assicurarsi che tutti i moduli I/O o i componenti della basetta sperimentale siano collegati all’Hub.
3. Nella schermata iniziale, premere [prgm], selezionare il nome del programma dall’elenco visualizzato e premere [enter].

   Il nome del programma viene copiato nella schermata iniziale.
4. Premere di nuovo [enter] per eseguire il programma.

Per modificare un programma esistente:

1. Nella schermata iniziale, premere [prgm], selezionare **Edit**.
2. Selezionare il nome del programma dall’elenco visualizzato e premere [enter].

   Il programma viene aperto nell’editor di programmazione.

Utilizzo del menu Hub per compilare comandi

Il menu Hub è disponibile sulla calcolatrice grafica TI CE ogni volta che si sta creando o modificando un programma. Consente di risparmiare tempo nella creazione di comandi e di scrivere i comandi con l’ortografia e la sintassi corrette.

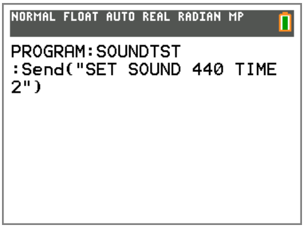
Nota: Per compilare un comando dal menu Hub, è necessario conoscere:

• Il nome univoco del componente che si sta indirizzando, come "SOUND" per il diffusore su scheda.

• I parametri del comando applicabili al componente, come la frequenza e la durata del segnale acustico. Alcuni parametri sono opzionali e potrebbe essere necessario conoscere l’intervallo di valori di un parametro.

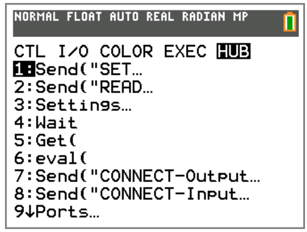
Esempi di utilizzo del menu Hub:

In questo esempio sulla calcolatrice grafica CE TI viene compilato il comando **Send ("SET SOUND 440 TIME 2")** per emettere un segnale acustico di 440 Hz per 2 secondi nell’altoparlante integrato.



1. Aprire (o creare) il programma che si utilizzerà per comunicare con l'Hub.
2. Posizionare il cursore nel punto in cui si desidera inserire il comando.
3. Premere [prgm] e selezionare **Hub**.

   Viene visualizzato il menu Hub.



4. Selezionare **Send "SET** e premere [enter], quindi selezionare **SOUND** e premere [enter].

5. Digitare **440** come frequenza sonora.

6. Nel menu Hub, selezionare **Impostazioni > TEMPO**.

7. Digitare **2** come valore per TEMPO.

8. Per completare il comando, digitare le virgolette di chiusura (premere [alpha] [+]) e premere [)].

9. Per tornare alla schermata iniziale e testare il comando, premere [2nd] [quit] e seguire le istruzioni precedenti per l'esecuzione di un programma.

Suggerimenti per la programmazione con la calcolatrice grafica TI CE

- Assicurarsi che il codice non contenga spazi non necessari che possono causare errori di sintassi. Tra questi figurano gli spazi ripetuti all'interno della riga e uno o più spazi al termine di una riga.
- Il codice copiato da una sorgente esterna potrebbe mostrare virgolette "curve" (“...”) laddove è necessario utilizzare le virgolette diritte ("..."). Per digitare le virgolette diritte, premere [alpha] e quindi [+].
- Per cancellare l'attuale riga di codice, premere [clear].
- Per digitare operatori relazionali quali $=$, $<$ e $\leq$, premere [2nd] [test].
- Per digitare uno spazio, premere [alpha] e quindi [0].

- Se il programma non risponde durante l'esecuzione, premere il tasto [on].
- **Nota**: se la sintassi di un comando non contiene una parentesi a sinistra aperta, come **"Wait "**, l'utilizzo di una coppia di parentesi in un argomento può essere interpretato come l'argomento intero e può causare un errore di sintassi imprevisto. Quando si immettono lunghe espressioni con parentesi, racchiudere l'intera espressione tra parentesi accoppiate per evitare errori di sintassi di questo tipo.

  Valido: Wait ((X+4)*5)
  Valido: Wait X+4*5
  Errore di sintassi: Wait (X+4)*5

Ulteriori informazioni

Per programmi di esempio e maggiori dettagli sulla programmazione dell' TI-Innovator™ Hub, vedere la TI-Innovator™ Technology eGuide (page ii).

Programmazione dell'Hub nella tecnologia TI-Nspire™ CX

Nota: Queste istruzioni si applicano alla tecnologia TI-Nspire™ CX. Per istruzioni analoghe applicabili alla calcolatrice grafica TI CE, fare riferimento a Programmazione dell'Hub sulla calcolatrice grafica TI CE (page 9).

La TI-Innovator™ Hub risponde ai comandi di programmazione TI Basic quali **Send** e **Get**.

- **Send** - Invia le stringhe di comando all'Hub per controllare i dispositivi o richiedere informazioni.
- **Get** e **GetStr** - Recuperano le informazioni richieste dall'Hub.
- **eval()** - Fornisce il risultato di un'espressione sotto forma di stringa di caratteri. Valido solo all'interno dei comandi **Send**, **Get** e **GetStr**.
- **Wait** - Interrompe l'esecuzione del programma per un numero di secondi specificato.

Esempi di codice: Tecnologia TI-Nspire™ CX

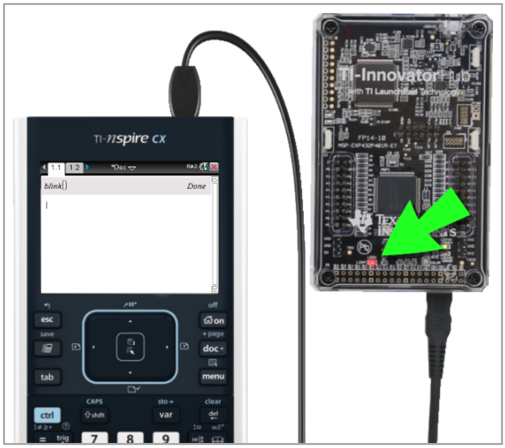
Azione desiderata	Codice del programma
Attivare il LED rosso integrato ("LIGHT").	`Send "SET LIGHT ON"`
Riprodurre un segnale acustico da 440 Hz nell'altoparlante integrato ("SOUND") per 2 secondi.	`Send "SET SOUND 440 TIME 2"`
Attivare l'elemento blu del LED rosso-verde-blu integrato ("COLOR") impostando una luminosità del 100%.	`Send "SET COLOR.BLUE 255"`
Leggere e visualizzare il valore corrente	`Send "READ BRIGHTNESS"`

Azione desiderata	Codice del programma
del sensore di luminosità integrato ("BRIGHTNESS"). L'intervallo è compreso tra 0% e 100%.	`Get a: Disp a`

Programma di esempio per far lampeggiare un LED di bordo

Il seguente programma TI-Nspire™ CX utilizza i comandi **Send** e **Wait** per far lampeggiare il LED rosso integrato nell'Hub. I comandi sono contenuti in un loop "For...EndFor" che ripete il ciclo intermittente ON/OFF per 10 iterazioni.

```
Define blink()=
Prgm
For n,1,10
  Send "SET LIGHT ON"
  Wait 1
  Send "SET LIGHT OFF"
  Wait 1
EndFor
EndPrgm
```



Come creare ed eseguire un programma

Nota: Questa è la versione breve delle istruzioni. Per istruzioni più dettagliate fare riferimento a *Editor di programmazione TI-Nspire™ CX*, accessibile dalla TI-Innovator™ Technology eGuide (page ii).

Prima di cominciare:

- Consultare i requisiti di sistema (page 2) e aggiornare il software se necessario.
  - Nei palmari TI-Nspire™ CX utilizzare il software del computer TI-Nspire™ per aggiornare il sistema operativo.
  - Nei computer in cui viene eseguito il software TI-Nspire™ CX utilizzare il menu della guida per aggiornare il software.

Per creare un nuovo programma in un documento TI-Nspire™ CX:

1. Sul palmare premere [doc▾] e selezionare **Inserisci > Editor di programma > Nuovo**. Dal software del computer, fare clic su **Inserisci > Editor di programma > Nuovo**.

2. Digitare un nome per il programma, come "soundtst", selezionare **Programma** come tipo e fare clic su **OK**.

   Viene visualizzato l’editor di programma con un modello per il codice del programma.

3. Tra le righe **Prgm** e **EndPrgm**, digitare le righe di codice che compongono il programma.
   - È possibile digitare i nomi dei comandi o inserirli dal menu dell’editor di programma.
   - Dopo aver digitato ogni riga, premere **Invio** per digitare altri codici.
   - Utilizzare i tasti freccia per spostarsi all’interno del programma.

Per memorizzare il programma:

È necessario memorizzare il programma prima di poterlo eseguire.

▶ Sul palmare premere [menu] e selezionare **Controlla sintassi e salva > Controlla sintassi e salva**.
Nel menu dell’editor di programma fare clic su **Controlla sintassi e salva > Controlla sintassi e salva**.

Per chiudere l’editor di programma

▶ Sul palmare premere [menu] e selezionare **Azioni > Chiudi**.
Nel menu dell’editor di programma fare clic su **Azioni > Chiudi**.

Se sono state apportate modifiche dopo la memorizzazione del programma, viene richiesto di controllare la sintassi e memorizzare.

Per eseguire il programma:

1. Verificare che l’ TI-Innovator™ Hub sia collegato al palmare o al computer.
2. Assicurarsi che tutti i moduli I/O o i componenti della basetta sperimentale siano collegati all’Hub.
3. Aprire il documento che contiene il programma.
4. In una pagina della calcolatrice immettere il nome del programma e le parentesi. Se il programma richiede argomenti, racchiuderli all’interno di parentesi, separati da virgole.

   Il programma viene eseguito.

Per modificare un programma esistente:

1. Se necessario, aprire il documento che contiene il programma.
2. Andare in una pagina della calcolatrice.
3. Sul palmare premere [menu] e selezionare **Funzioni e programmi > Editor di programma> Apri**.
   Nel menu della calcolatrice, fare clic su **Funzioni e programmi > Editor di programma> Apri**.

4. Selezionare il nome del programma dall’elenco visualizzato.

   Il programma viene visualizzato in una pagina dell’editor di programma.

Utilizzo del menu Hub per compilare comandi

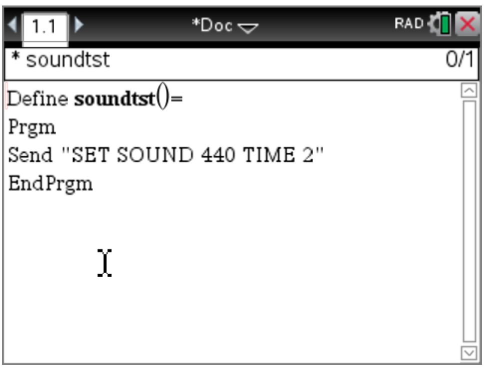
Il menu Hub è disponibile nella tecnologia TI-Nspire™ CX ogni volta che si sta creando o modificando un programma. Consente di risparmiare tempo nella creazione di comandi e di scrivere i comandi con l’ortografia e la sintassi corrette.

Nota: Per compilare un comando dal menu Hub, è necessario conoscere:

- Il nome univoco del componente che si sta indirizzando, come "SOUND" per il diffusore su scheda.
- I parametri del comando applicabili al componente, come la frequenza e la durata del segnale acustico. Alcuni parametri sono opzionali e potrebbe essere necessario conoscere l’intervallo di valori di un parametro.

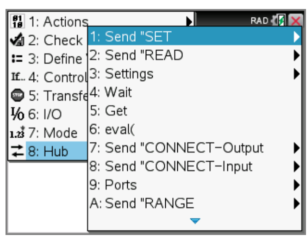
Esempi di utilizzo del menu Hub:

In questo esempio su TI-Nspire™ CX viene compilato il comando **Send ("SET SOUND 440 TIME 2")** per emettere un segnale acustico di 440 Hz per 2 secondi nell’altoparlante integrato.



1. Aprire (o creare) il programma che si utilizzerà per comunicare con l’Hub.
2. Posizionare il cursore nel punto in cui si desidera inserire il comando.
3. Sul palmare premere [menu] e selezionare **Hub**. Nel menu dell'editor di programma, selezionare **Hub**.

   Viene visualizzato il menu Hub.



4. Selezionare **Send "SET** e quindi selezionare **SOUND** per inserire la prima parte del comando.

```
Define soundtst()=
Prgm
Send "SET SOUND "
EndPrgm
```

5. Digitare **440** come valore della frequenza.

```
Send "SET SOUND 440"
```

6. Nel menu Hub, selezionare **Impostazioni > TEMPO**.

```
Send "SET SOUND 440 TIME "
```

7. Per completare il comando, digitare **2** come valore per TEMPO.

```
Define soundtst()=
Prgm
Send "SET SOUND 440 TIME 2"
EndPrgm
```

8. Per testare il comando, seguire le istruzioni precedentemente fornite per eseguire un programma.

Suggerimenti per la programmazione con la tecnologia TI-Nspire™ CX

- Il codice copiato da una sorgente esterna potrebbe contenere virgolette "curve" (“...”) laddove è necessario utilizzare le virgolette diritte ("..."). Per digitare le virgolette diritte, premere `ctrl` `×`.
- Per cancellare l’attuale riga di codici, premere `ctrl` `clear`.
- Per digitare operatori relazionali quali $=$, $<$ e $\leq$, premere `ctrl` `=`.
- Per digitare uno spazio, premere `␣`.
- Se il programma non risponde durante l’esecuzione:

  Palmare TI-Nspire™ CX: Tenere premuto il tasto `on` e premere `enter` più volte.
  Windows®: Tenere premuto il tasto **F12** e premere **Invio** più volte.
  Mac®: Tenere premuto il tasto **F5** e premere **Invio** più volte.

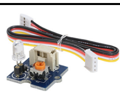
Ulteriori informazioni

Per programmi di esempio e maggiori dettagli sulla programmazione dell’ TI-Innovator™ Hub, vedere la TI-Innovator™ Technology eGuide (page ii).

TI-Innovator™ Moduli I/O

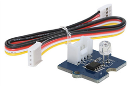
Questi moduli di input/output (venduti separatamente) comprendono i cavi per collegare i moduli all’ TI-Innovator™ Hub.

Modulo	Porte	Immagine	Codice di esempio per calcolatrice grafica TI CE

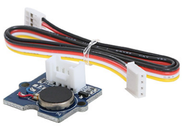
LED bianco *	OUT 1
	OUT 2
	OUT 3
			Per accendere il modulo LED bianco collegato a **OUT 1**:
			`Send("CONNECT LED 1 TO OUT 1")`
			`Send("SET LED 1 ON")`

Modulo	Porte	Immagine	Codice di esempio per calcolatrice grafica TI CE

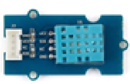
Servomotore **	OUT 3		Per ruotare l'albero del servomotore collegato a **OUT 3** in senso antiorario di 90°:
			`Send("CONNECT SERVO 1 TO OUT 3")`
			`Send("SET SERVO 1 TO -90")`
			Codice equivalente che utilizza una variabile con **eval()**:
			`angdeg:=-90`
			`Send("CONNECT SERVO 1 TO OUT 3")`
			`Send("SET SERVO 1 TO eval(angdeg)")`

Sensore di luce analogico	IN 1
	IN 2
	IN 3
			Per leggere e visualizzare il livello di luce nell'ambiente dal sensore collegato a **IN 2**:
			`Send("CONNECT LIGHTLEVEL 1 TO IN2")`
			`Send("READ LIGHTLEVEL 1")`
			`Get(L):Disp(L)`

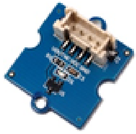
Ranger ultrasonico	IN 1
	IN 2
			Per leggere e visualizzare la distanza misurata dal ranger collegato a **IN 2**:
			`Send("CONNECT RANGER 1 TO IN2")`
			`Send("READ RANGER 1")`
			`Get(R):Disp(R)`

Motore a vibrazione	OUT 1
	OUT 2
	OUT 3
			Per accendere il motore a vibrazione collegato a **OUT 1**:
			`Send("CONNECT VIB.MOTOR 1 TO OUT 1")`
			`Send("SET VIB.MOTOR 1 TO ON")`

Temperatura Sensore	IN 1
	IN 2
	IN 3
			Leggere e visualizzare la temperatura ambiente dal sensore collegato a **IN 3**:
			`Send("CONNECT TEMPERATURE 3 TO IN3")`
			`Send("READ TEMPERATURE 3")`
			`Get(T):Disp(T)`

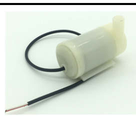
Sensore di temperatura e umidità	IN 1
	IN 2
	IN 3
			Collegare il sensore **DHT** alla porta **IN 2**
			`Send( "CONNECT DHT 1 TO IN2 ")`
			Leggere la temperatura dal sensore **DHT** collegato a **IN 2**:
			`Send( "READ DHT 1 TEMPERATURE")`
			`Get temperature`
			Leggere l'umidità dal sensore **DHT**:
			`Send "READ DHT 1 HUMIDITY"`
			`Get humidity`

Modulo	Porte	Immagine	Codice di esempio per calcolatrice grafica TI CE

Sensore Hall	IN 1
	IN 2
	IN 3
			Collegare il sensore con effetto Hall alla porta **IN3**:
			`Send "CONNECT ANALOG.IN 1 TO IN 3"`
			Leggere il valore del campo magnetico indicato dal sensore:
			`Send "READ ANALOG.IN 1"`
			`Get m`

Sensore di umidità	IN 1
	IN 2
	IN 3
			Collegare il sensore di umidità a **IN 1**:
			`Send "CONNECT MOISTURE 1 IN 1"`
			Configurare l'intervallo di misurazione in modo che sia compreso tra 0 e 100. L'intervallo è un indice e non ha unità.
			`Send "RANGE MOISTURE 1 0 100"`
			Leggere il sensore:
			`Send "READ MOISTURE 1"`
			`Get moisture`

MOSFET	OUT 1
	OUT 2
			Collegare il **MOSFET** alla porta **OUT 1**:
			`Send "CONNECT ANALOG.OUT 1 TO OUT 1"`
			Controllare la pompa/il motore collegato al 50% della velocità per 3 secondi:
			`Send "SET ANALOG.OUT 1 128 TIME 3"`

Pompa idraulica			Viene controllata mediante un modulo MOSFET.

*****Il modulo LED bianco deve essere assemblato.

******Il Servomotore ha bisogno di un alimentatore ausiliario e deve essere assemblato.

Nota: Per maggiori dettagli, consultare la TI-Innovator™ Technology eGuide (page ii).

Collegamento a un modulo I/O

Utilizzare il cavo I/O in dotazione con il modulo per collegarlo a una porta di ingresso o uscita dell’Hub.

1. Fare riferimento alla tabella sopra riportata per verificare quali porte I/O supportano il modulo da collegare.
2. Collegare entrambe le estremità del cavo I/O al connettore bianco del modulo.

3. Collegare l’estremità libera del cavo I/O alla porta dell’Hub che si desidera utilizzare.
4. Se il modulo ha bisogno di un alimentatore ausiliario, collegare l’alimentatore (page 24),

Programma di esempio per far lampeggiare un modulo LED

Il programma della calcolatrice grafica TI CE utilizza i comandi **Send** e **Wait** per far lampeggiare un modulo LED collegato a una porta I/O.

Nota: questo programma funziona correttamente solo se la calcolatrice è collegata all’Hub e il modulo LED è fisicamente collegato alla porta **OUT 1**.

```
PRGM: BLINKIO
Send("CONNECT LED 1 TO
OUT1")
For(N,1,10)
Send("SET LED 1 ON")
Wait 1
Send("SET LED 1 OFF")
Wait 1
End
Send("DISCONNECT LED 1")
```

Nota: se si sta utilizzando la tecnologia TI-Nspire™ CX, omettere le parentesi e sostituire **`End`** con **`EndFor`**.

La stringa di comando "CONNECT LED 1 TO OUT1" dell’Hub comunica all’Hub che il modulo LED è collegato alla porta **OUT 1** dell’Hub. Una volta inviato questo comando, il codice può indirizzare il LED come "LED 1." Il comando CONNECT è richiesto solo per i moduli I/O e i componenti della basetta sperimentale. Non è necessario con componenti di bordo quali l’altoparlante integrato.

Ulteriori informazioni

Per un riepilogo delle precauzioni da prendere quando si usano i moduli I/O, consultare *Precauzioni generali* (page 27).

Per programmi di esempio, un elenco di moduli I/O aggiuntivi e maggiori dettagli sulla programmazione dei moduli I/O, vedere la TI-Innovator™ Technology eGuide (page ii).

TI-Innovator™ Breadboard Pack

La basetta sperimentale e i suoi componenti (venduti separatamente) consentono di creare prototipi di circuiti stampati da collegare all’ TI-Innovator™ Hub tramite i pin del connettore della basetta sperimentale.

La basetta sperimentale è costituita dai seguenti componenti:

- Una basetta sperimentale e cavi di accoppiamento per la creazione di collegamenti elettrici.
- Componenti indirizzabili, come LED e sensori che rispondono ai comandi dell'Hub. Questi sono elencati nella tabella qui sotto.
- Componenti passivi, come resistori, condensatori e interruttori manuali che non sono direttamente indirizzabili dall'Hub ma che sono richiesti in molti prototipi di circuiti stampati.
- Un portabatterie per quattro batterie AA. Le batterie non sono incluse.

Componenti indirizzabili

Componente	Immagine	Usato con pin	Descrizione
LED rossi		BB 1-10	Diodo ad emissione luminosa che emette luce quando viene attraversato dalla corrente.
LED verdi		BB 1-10	Diodo ad emissione luminosa che emette luce quando viene attraversato dalla corrente.
LED RGB (rossi-verdi-blu)		BB 8-10	Diodo ad emissione luminosa con elementi rossi, verdi e blu regolabili indipendentemente. Può produrre un’ampia gamma di colori.
Termistore		BB 5,6,7 (input analogico necessario)	Resistore la cui resistenza cambia in base alla temperatura. Usato per misurazioni e controlli.
Display a 7 segmenti		BB 1-10	Array di LED predisposti per visualizzare i numeri e alcuni caratteri alfabetici. Ha anche un LED per il punto decimale.

Motore CC piccolo		BB 1-10 (utilizza il digitale per generare il segnale PWM nel software)	Motore che converte l'alimentazione a corrente continua in energia meccanica.
MOSFET di alimentazione TTL		BB 1-10	Transistore utilizzato per amplificare o commutare i segnali elettronici.
Sensore di temperatura analogico TI		BB 5,6,7 (input analogico necessario)	Sensore che segnala una tensione proporzionale alla temperatura ambiente in un intervallo compreso tra −55 °C e 130 °C.
Sensore di luce visibile		BB 5,6,7 (input analogico necessario)	Sensore che segnala il livello di luce nell'ambiente.
Trasmettitore a infrarossi LTE-302, punto giallo		BB 1-10 (output digitale)	LED a infrarossi con proiezione laterale della luce, progettato per essere accoppiato con il fototransistore LTR-301.
Ricevitore a infrarossi LTR-301, punto rosso		BB 1-10 (input digitale)	Fototransistore a infrarossi con proiezione laterale della luce, progettato per essere accoppiato con l'emettitore a infrarossi LTE-302.

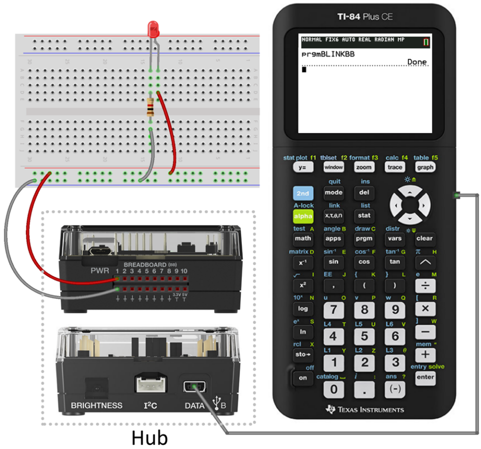
Codice di esempio per far lampeggiare il LED di una basetta sperimentale

Il seguente programma per calcolatrici grafiche TI CE utilizza i comandi **Send** e **Wait** per far lampeggiare un LED specifico su una basetta sperimentale.

Nota: questo programma funziona correttamente solo se la calcolatrice è collegata all'Hub e il LED è fisicamente collegato a **BB1** (pin 1 della basetta sperimentale) sull'Hub.

```
PRGM: BLINKBB
Send("CONNECT LED 1 TO BB1")
For(NS,1,10)
Send("SET LED 1 ON")
Wait 1
Send("SET LED 1 OFF")
Wait 1
End
Send("DISCONNECT LED 1")
```

Nota: se si sta utilizzando la tecnologia TI-Nspire™ CX, omettere le parentesi e sostituire **`End`** con **`EndFor`**.



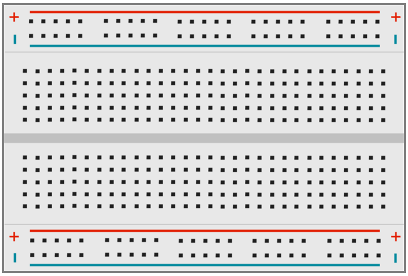
La stringa di comando dell’Hub "CONNECT LED 1 TO BB1" comunica all’Hub che un LED della basetta sperimentale è collegato al pin **1** sull’Hub. Una volta inviato questo comando, il codice può indirizzare il LED come "LED 1." Il comando CONNECT è richiesto solo per i moduli I/O e i componenti della basetta sperimentale. Non è necessario con componenti di bordo quali l’altoparlante integrato.

Caratteristiche di base di una basetta sperimentale

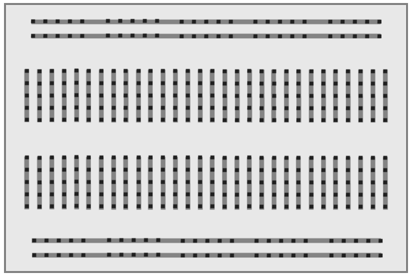
La basetta sperimentale semplifica il collegamento dei componenti elettronici di un progetto mediante l’inserimento dei connettori dei componenti e dei cavi di accoppiamento nei pin della basetta sperimentale.

I pin sono disposti in gruppi di 5. I 5 pin in ogni gruppo sono collegati elettricamente gli uni con gli altri sul retro della scheda. Per collegare insieme i connettori e i cavi, inserirli nei pin all’interno dello stesso gruppo.

- Le linee di contatto in alto e in basso sono indicate con strisce rosse (+) e blu (–). I gruppi in ogni linea sono collegati elettricamente lungo l’intera lunghezza della striscia.
- I restanti gruppi di 5 pin sulla scheda sono contrassegnati con numeri e lettere. Ogni gruppo è elettricamente isolato dagli altri.



Parte anteriore della scheda con linee di contatto e pin di collegamento



Interconnessioni sul retro della scheda (normalmente nascoste). I gruppi di 5 pin in ogni linea di contatto sono interconnessi. Tutti gli altri gruppi di 5 pin sono isolati.

Lo spazio vuoto al centro della basetta sperimentale agevola il collegamento di componenti elettronici forniti come pacchetti dual-inline.

Utilizzare cavi di accoppiamento tra l'Hub e la basetta sperimentale per alimentare i componenti della basetta e per controllarli o monitorarli tramite il codice del programma. L'Hub è provvisto di 20 pin contrassegnati, inclusi 10 pin di segnalazione, 8 pin di messa a terra, un pin di alimentazione a 3,3 V e un pin di alimentazione a 5,0 V.

Ulteriori informazioni

Per un riepilogo delle precauzioni da prendere quando si usa la basetta sperimentale e i suoi componenti, consultare *Precauzioni generali* (page 27).

Per programmi di esempio e maggiori dettagli sulla programmazione dei componenti della basetta sperimentale sull' TI-Innovator™ Hub, vedere la TI-Innovator™ Technology eGuide (page ii).

Utilizzo di una fonte di alimentazione Origine

Di norma, l' TI-Innovator™ Hub e i suoi componenti collegati vengono alimentati dal computer o dalla calcolatrice host, tramite il connettore **DATI**. Alcuni componenti, come il servomotore opzionale, richiedono più energia di quanta una calcolatrice possa fornirne in modo affidabile.

Il connettore **PWR** sull'Hub consente di collegare un alimentatore ausiliario. È possibile utilizzare il TI Wall Charger o il External Battery Pack.

TI Wall Charger (incluso nell’Hub)

- Si inserisce in una presa a muro.
- Non fa uso di batterie.

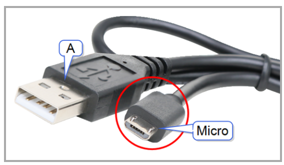
External Battery Pack (venduto separatamente)

- Ricaricabile.
- Ha un pulsante On/Off con una serie di LED che indicano per qualche istante il livello di carica della batteria, quando viene attivata.
- Si spegne quando è scollegato dall’Hub per circa 3 minuti.

Nota: per ricaricare il External Battery Pack, scollegarlo dall’Hub e quindi collegarlo al TI Wall Charger utilizzando il USB Standard A to Micro . Non utilizzare il External Battery Pack come fonte di alimentazione ausiliaria mentre è sotto carica.

Collegamento dell’alimentatore

1. Identificare il connettore Micro sul USB Standard A to Micro cavo di alimentazione ausiliario.
2. Inserire il connettore Micro nel connettore **PWR** nella parte superiore dell’Hub.



3. Inserire l’estremità libera del cavo (il connettore "**A**") nella porta USB dell’alimentatore.
4. Accendere l’alimentatore:
   - Se si utilizza il TI Wall Charger, inserirlo in una presa a muro.
   - Se si utilizza il External Battery Pack, premere il pulsante di accensione.

   Il LED di alimentazione ausiliaria sull’Hub emette una luce per indicare che l’Hub è alimentato.
5. Collegare l’ TI-Innovator™ Hub alla calcolatrice host utilizzando il USB Standard A to Mini-B .
6. Collegare il modulo I/O o un componente della basetta sperimentale all’Hub.

Risoluzione dei problemi di TI-Innovator™ Hub

Non vedo il LED verde quando collego l' TI-Innovator™ Hub.

- Assicurarsi di aver acceso la calcolatrice.
- Se si utilizza un cavo USB Unit-to-Unit (Mini-A to Mini-B) per collegare una calcolatrice, assicurarsi di aver connesso l'estremità "B" del cavo al connettore **DATI** nella parte inferiore dell'Hub. L'inversione di questo cavo impedisce all'Hub di ricevere energia elettrica.
- Assicurarsi che la calcolatrice o il computer soddisfi i requisiti di sistema (pagina 2).
- Verificare che l'estremità del cavo USB collegato alla calcolatrice sia inserito correttamente.

Come si spegne l'Hub?

1. Spegnere il computer o la calcolatrice host.
   - OPPURE -
   Scollegare il cavo USB.
2. Scollegare qualsiasi alimentatore ausiliario collegato alla porta **PWR** dell'Hub.

Perché il mio programma mi segnala un errore di sintassi?

- Se hai copiato il codice da una sorgente esterna o un editor di testo, il codice potrebbe contenere virgolette "curve" (“...”) laddove è necessario utilizzare le virgolette diritte ("..."). Occorrerà sostituire alcune o tutte le virgolette curve.
- Le regole di sintassi sono leggermente diverse tra la calcolatrice grafica TI CE e la tecnologia TI-Nspire™ CX. Potrebbe essere necessario modificare il codice creato in origine per una piattaforma, affinché possa funzionare su un'altra.
- Sulla calcolatrice grafica TI CE assicurarsi che non ci sia un carattere di spaziatura al termine di una riga di codice. Per trovare questi spazi finali in una riga, spostare il cursore sulla riga e premere [2nd] [▸]. Gli spazi adiacenti nel codice possono ugualmente causare un errore di sintassi.

Come faccio per arrestare un programma che non risponde?

- Calcolatrice grafica TI CE: Premere il tasto [on].
- Palmare TI-Nspire™ CX: Tenere premuto il tasto [⌂on] e premere [enter] più volte.
- Windows®: Tenere premuto il tasto **F12** e premere **Invio** più volte.
- Mac®: Tenere premuto il tasto **F5** e premere **Invio** più volte.

Perché si verifica un errore quando tento di aggiornare lo TI-Innovator™ Sketch?

- Per aggiornare lo sketch, assicurarsi di utilizzare il cavo USB Standard A to Micro , non il cavo USB Standard A to Mini-B . Collegare la micro estremità del cavo al connettore **PWR** nella parte superiore dell'Hub.
- Assicurarsi di utilizzare uno dei browser web richiesti per l'aggiornamento. Vedere Aggiornamento del software dell'Hub (pagina 7).

Ulteriori informazioni

Per ulteriori informazioni sulla risoluzione dei problemi, vedere la TI-Innovator™ Technology eGuide (pagina ii).

Precauzioni generali per TI-Innovator™ Hub

TI-Innovator™ Hub

- Non esporre il Hub a temperature superiori ai 60°C (140°F).
- Non smontare o danneggiare il Hub.
- Non concatenare insieme più Hubs attraverso porte I/O o il connettore della basetta sperimentale.
- Usare unicamente il cavo USB fornito con l'Hub.
- Usare unicamente gli alimentatori forniti da TI:
    - TI Wall Charger incluso nell' TI-Innovator™ Hub
    - opzionale External Battery Pack
    - Portabatterie per 4 batterie AA incluso nel TI-Innovator™ Breadboard Pack
- Assicurarsi che i componenti alimentati dall'Hub non superino il limite di potenza di 1 amp dell'Hub's.
- Evitare di utilizzare l'Hub per controllare la corrente elettrica alternata.

Connettore della basetta sperimentale sull'Hub

- Non inserire i connettori dei LED e altri componenti direttamente nel connettore della basetta sperimentale dell'Hub's. Assemblare i componenti sulla basetta sperimentale e utilizzare i cavi di accoppiamento forniti per collegare la basetta sperimentale all'Hub.
- Non collegare il pin fisso da 5 V sul connettore della basetta sperimentale dell'Hub's con nessun altro pin, specialmente i pin di messa a terra. Altrimenti si rischia di danneggiare l'Hub.
- Si sconsiglia di collegare la fila superiore di pin fissi (BB1-10) alla fila inferiore (pin di alimentazione e messa a terra).
- Nessun pin sul connettore della basetta sperimentale dell'Hub's può avere una corrente di caduta o sorgente superiore a 4 mA.

Basetta sperimentale

- Non collegare i connettori positivi e negativi di un alimentatore con lo stesso gruppo di 5 pin sulla basetta sperimentale. Altrimenti si rischia di danneggiare la basetta sperimentale e l'alimentatore.
- Rispettare la polarità corretta:
    - Quando si collega la basetta sperimentale all'Hub.
    - Quando si collegano componenti che sono sensibili alla polarità, come i LED e il MOSFET di alimentazione TTL.

Moduli I/O

- Utilizzare la porta input od output corretta in base alle necessità di ogni modulo.

- Motore a vibrazione: supportato su **OUT 1**, **OUT 2** e **OUT 3**.
- Servomotore: utilizzare solo **OUT 3**.
- LED bianco: supportato su **OUT 1**, **OUT 2** e **OUT 3**.
- Sensore di luce analogico: supportato su **IN 1**, **In 2** e **IN 3**.
- Ranger ultrasonico: supportato su **IN 1**, **IN 2**.

- Utilizzare una fonte di alimentazione ausiliaria per i moduli che richiedono più di 50 mA, come:
  - Motore a vibrazione
  - Servomotore
- Non trattenere l'albero del servomotore mentre ruota. Inoltre, non ruotare a mano il servo motore.
- LED bianco:
  - Non curvare i connettori ripetutamente; altrimenti si finirà per logorare i fili rischiando di romperli.
  - Il LED richiede la polarità corretta quando viene inserito nella sua presa. Per maggiori dettagli, fare riferimento alle istruzioni su come assemblare il LED nella TI-Innovator™ Technology eGuide (pagina ii).
  - Il LED richiede la polarità corretta quando viene inserito nella sua presa. Per maggiori dettagli, fare riferimento alle istruzioni su come assemblare il LED (pagina 326).
- Nessun modulo I/O può avere una corrente di caduta o sorgente superiore a 4 mA.

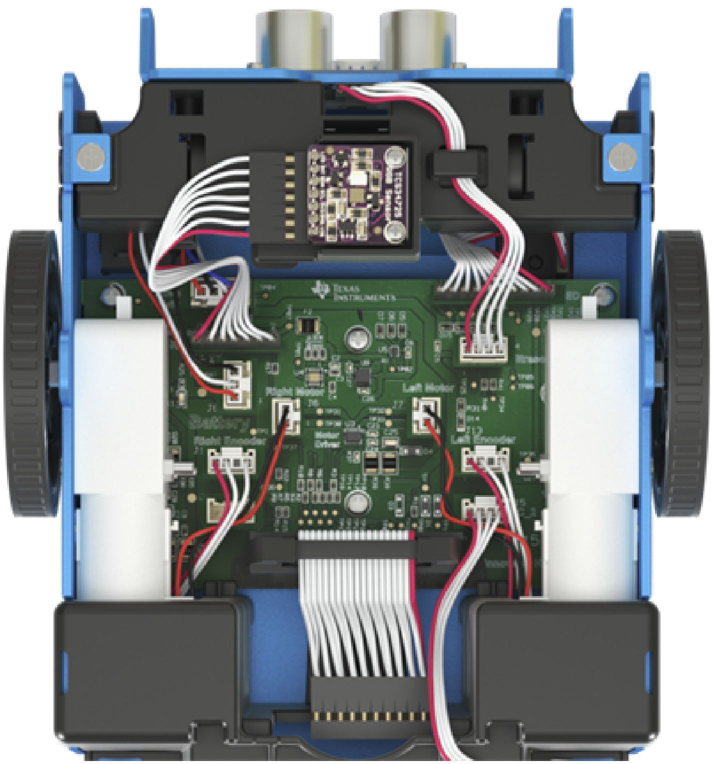
TI-Innovator™ Rover

- Non esporre il Rover a temperature superiori ai 60°C (140°F).
- Non smontare o danneggiare il Rover.
- Non appoggiare nulla che pesi più di 1 Kg o 2,2 lb sulla Rover piattaforma.
- Usare unicamente il cavo USB fornito con il TI-Innovator™ Hub.
- Usare unicamente i cavi a nastro forniti con il Rover.
- Usare unicamente il caricatore da muro fornito da TI incluso con l'Hub.
- Il Ultrasonic Ranger montato anteriormente rileverà gli oggetti a una distanza fino a 4 metri dal Rover. Per ottenere i risultati migliori assicurarsi che la superficie dell'oggetto sia più grande di una cartellina. Se usato per rilevare oggetti di piccole dimensioni, come una tazza, posizionare il Rover a una distanza fino a 1 metro dall'oggetto.
- Per ottenere i risultati migliori rimuovere la custodia della calcolatrice grafica.
- Per le prestazioni migliori usare Rover sul pavimento e non su tavoli. La caduta del Rover da un tavolo può danneggiarlo.
- Per le prestazioni migliori usare Rover su una superficie dura. Le ruote del Rover possono impigliarsi in un tappeto o agganciarlo e tirarlo.

- Non girare i fermagli di supporto sulla piattaforma per la calcolatrice senza averli prima sollevati. Potrebbero rompersi.
- Non usare il pennarello come una leva per tirare o spingere il Rover.
- Non svitare la cassa protettiva sul fondo del Rover. Gli encoder hanno spigoli appuntiti che non devono essere esposti.
- Non muovere Rover dopo l’esecuzione di un programma. Il giroscopio interno può cercare involontariamente di rimettere in carreggiata il Rover usando la posizione iniziale.
- Quando si inserisce il cavo a nastro della basetta sperimentale nel connettore della basetta sperimentale dell’Hub, è fondamentale che il cavo venga inserito correttamente. Assicurarsi che lo spinotto del filo rosso (scuro) sia inserito nel foro 5v sul connettore della basetta sperimentale dell’Hub's.

Attenzione: Se si rimuove o scollega uno qualsiasi dei cavi, fare riferimento a questa immagine per ripristinare collegamenti in maniera corretta.

Riferimento alla vista dal basso



TI-Innovator™ Hub Commands Version 1.4

Use the Hub menus to create or edit a program. They can save you time building commands and help you with correct command spelling and syntax.

When you see "**Code Sample**" in a command table, this "**Code Sample**" may be copied and pasted *as is* to send to your graphing calculator to use in your calculations.

Example:

Code Sample:	`Send("RV FORWARD")` `Send("RV FORWARD SPEED 0.2 M/S TIME 10")`

Note: To build a command from the Hub menu, you need to know:

- The unique name of the component that you are addressing, such as "SOUND" for the on-board speaker.
- The command parameters that apply to the component, such as sound frequency and duration. Some parameters are optional, and you might need to know the value range of a parameter.

Understanding Syntax

- Capitalized words are keywords
- Lower case words are placeholders for numbers
- Commands within brackets are optional parameters

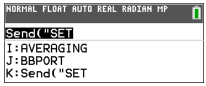
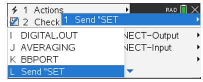
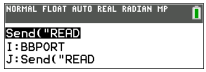
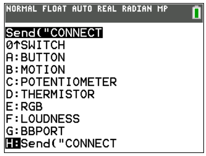
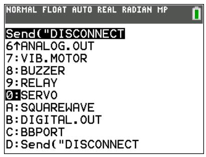
For example in: SET LIGHT ON [[BLINK|TOGGLE] frequency] [[TIME] seconds], "frequency" is entered as "**1**" and "seconds" is entered as "**10**".

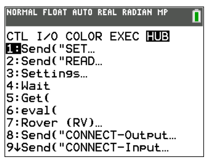
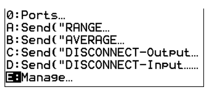
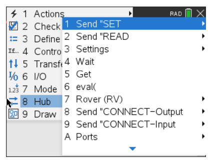
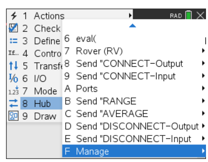
```
Send("SET LIGHT 1 BLINK 2 TIME 10")
```

NOTE: The commands listed below are for CE Calculators. If you are using TI-Nspire™ CX technology the parentheses are omitted. In addition, you will notice some other minor differences in the commands such as "**Endfor**" instead of "**End**" with the TI-Nspire™ CX technology. Screenshots are provided for reference. **Note:** Actual menus may vary slightly from provided images.

Last Menu Entry

Notice the last menu entries. These allow you to type in the name of the object instead of selecting it from the menu. These can also be used for sensors and peripherals that are not explicitly included in the menus. To use these, select the menu item to paste the beginning of the command. You then type in the name of the sensor or device you are using.

Last Menu Entry	CE Calculators	TI-Nspire™ CX
– Send("SET		
– Send("READ		
– Send("CONNECT		
– Send("DISCONNECT		

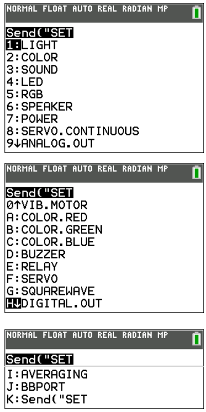
HUB Menus	CE Calculators	TI-Nspire™ CX
– Send("SET... – Send("READ... – Settings – Wait – Get(– eval(– Rover (RV)... – Send("CONNECT-Output... – Send("CONNECT-Input... – Ports... – Send("RANGE... – Send("AVERAGE... – Send("DISCONNECT-Output... – Send("DISCONNECT-Input... – Manage...	 	 

Send("SET…

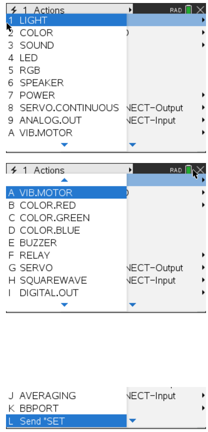
- SET
  - LIGHT
  - COLOR
  - SOUND
  - LED
  - RGB
  - SPEAKER
  - POWER
  - SERVO.CONTINOUS
  - DCMOTOR
  - ANALOG.OUT
  - VIB.MOTOR
  - COLOR.RED
  - COLOR.GREEN
  - COLOR.BLUE
  - BUZZER
  - RELAY
  - SERVO
  - SQUAREWAVE
  - DIGITAL.OUT
  - AVERAGING
  - BBPORT
  - Send("SET

Additional **Set** Commands

CE Calculators



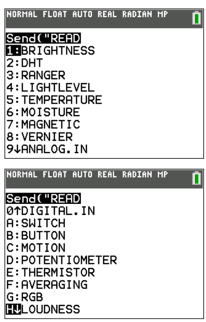
TI-Nspire™ CX



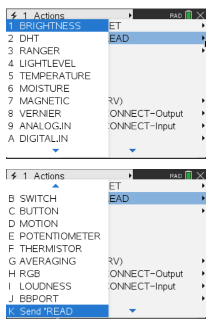
Send("READ…

- READ
  - BRIGHTNESS
  - DHT
  - RANGER
  - LIGHTLEVEL
  - TEMPERATURE
  - MOISTURE
  - MAGNETIC
  - VERNIER

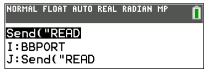
CE Calculators



TI-Nspire™ CX



– ANALOG.IN
– DIGITAL.IN
– SWITCH
– BUTTON
– MOTION
– POTENTIOMETER
– THERMISTOR
– AVERAGING
– RGB
– LOUDNESS
– BBPORT
– TIMER
– Send("READ

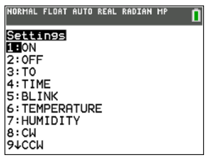


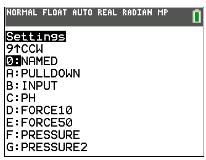
Additional **READ** Commands

Settings...

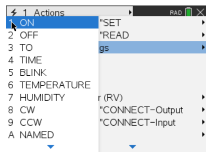
- Settings
  – ON
  – OFF
  – TO
  – TIME
  – BLINK
  – TEMPERATURE
  – HUMIDITY
  – CW
  – CCW
  – NAMED
  – PULLDOWN
  – INPUT
  – PH
  – FORCE10
  – FORCE50
  – PRESSURE
  – PRESSURE2

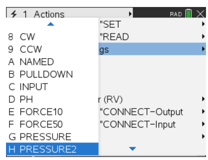
CE Calculators





TI-Nspire™ CX

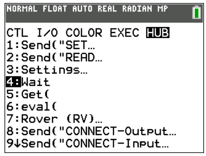




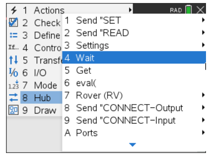
Wait

- Wait

CE Calculators



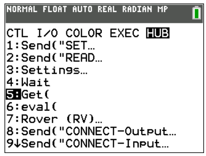
TI-Nspire™ CX



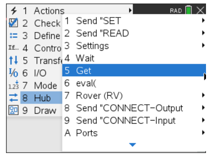
Get(

- Get(

CE Calculators



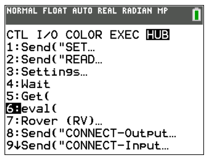
TI-Nspire™ CX



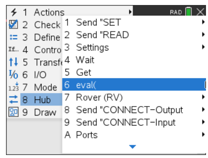
eval(

- eval(

CE Calculators



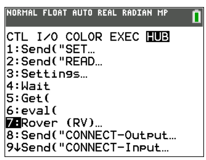
TI-Nspire™ CX



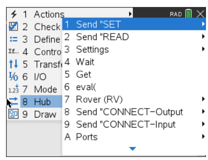
Rover (RV)…

- Drive RV…
- Read RV Sensors…
- RV Settings…
- Read RV Path…
- RV Color…
- RV Setup…
- RV Control…
- Send "CONNECT RV"
- Send "DISCONNECT RV"

CE Calculators



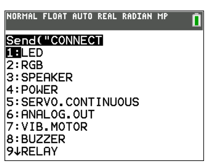
TI-Nspire™ CX



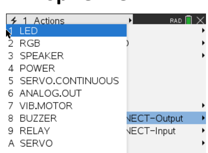
Send("CONNECT-Output…

- CONNECT-Output
  - LED
  - RGB
  - SPEAKER

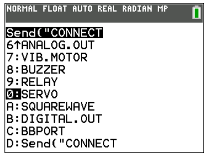
CE Calculators

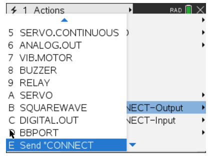


TI-Nspire™ CX



- POWER
- SERVO.CONTINUOUS
- DCMOTOR
- ANALOG.OUT
- VIB.MOTOR
- BUZZER
- RELAY
- SERVO
- SQUAREWAVE
- DIGITAL.OUT
- BBPORT
- Send("CONNECT



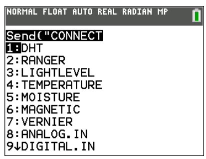


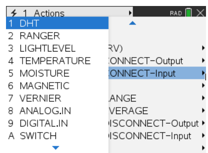
- LIGHT
- COLOR
- SOUND

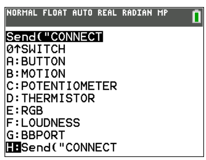
Send("CONNECT-Input...

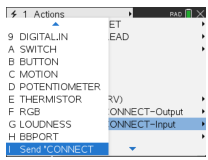
- CONNECT-Input
  - DHT
  - RANGER
  - LIGHTLEVEL
  - TEMPERATURE
  - MOISTURE
  - MAGNETIC
  - VERNIER
  - ANALOG.IN
  - DIGITAL.IN
  - SWITCH
  - BUTTON
  - MOTION
  - POTENTIOMETER
  - THERMISTOR
  - RGB
  - LOUDNESS
  - BBPORT

CE Calculators **TI-Nspire™ CX**







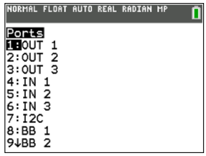


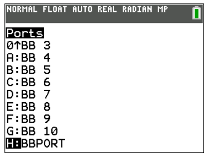
- Send("CONNECT
- BRIGHTNESS

Ports...

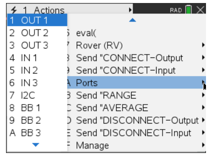
- Ports
  - OUT 1
  - OUT 2
  - OUT 3
  - IN 1
  - IN 2
  - IN: 3
  - I2C
  - BB 1
  - BB 2
  - BB 3
  - BB 4
  - BB 5
  - BB 6
  - BB 7
  - BB 8
  - BB 9
  - BB 10
  - BBPORT

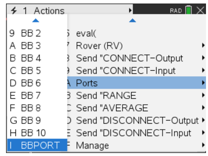
CE Calculators





TI-Nspire™ CX

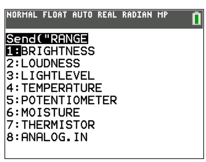




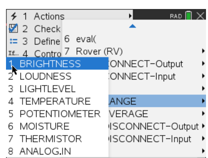
Send("RANGE...

- RANGE
  - BRIGHTNESS
  - LOUDNESS
  - LIGHTLEVEL
  - TEMPERATURE
  - POTENTIOMETER
  - MOISTURE
  - THERMISTOR
  - ANALOG.IN

CE Calculators



TI-Nspire™ CX

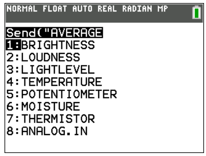


Send("AVERAGE...

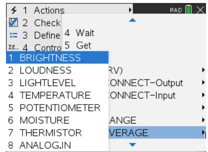
- AVERAGE
  - BRIGHTNESS
  - LOUDNESS
  - LIGHTLEVEL
  - TEMPERATURE
  - POTENTIOMETER
  - MOISTURE
  - THERMISTOR
  - ANALOG.IN

Additional **AVERAGE** Commands

CE Calculators



TI-Nspire™ CX

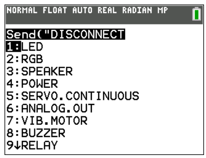


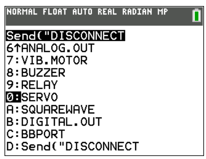
Send("DISCONNECT-Output...

- DISCONNECT-Output...
  - LED
  - RGB
  - SPEAKER
  - POWER
  - SERVO.CONTINUOUS
  - DCMOTOR
  - ANALOG.OUT
  - VIB.MOTOR
  - BUZZER
  - RELAY
  - SERVO
  - SQUAREWAVE
  - DIGITAL.OUT
  - BBPORT
  - Send("DISCONNECT

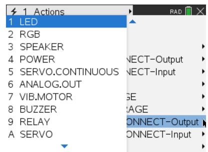
  - LIGHT
  - COLOR
  - SOUND

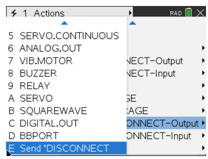
CE Calculators





TI-Nspire™ CX





Send("DISCONNECT-Input...

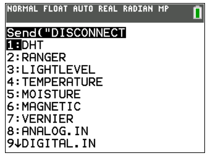
CE Calculators

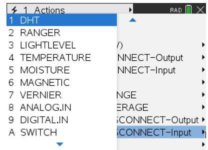
TI-Nspire™ CX

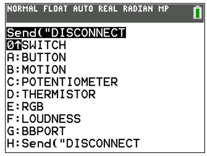
- DISCONNECT-Input...

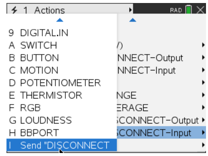
- DHT
- RANGER
- LIGHTLEVEL
- TEMPERATURE
- MOISTURE
- MAGNETIC
- VERNIER
- ANALOG.IN
- DIGITAL.IN
- SWITCH
- BUTTON
- MOTION
- POTENTIOMETER
- THERMISTOR
- RGB
- LOUDNESS
- BBPORT
- Send("DISCONNECT

- BRIGHTNESS





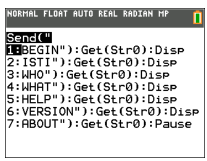




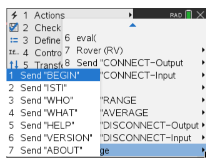
MANAGE

- MANAGE
  - BEGIN
  - ISTI
  - WHO
  - WHAT
  - HELP
  - VERSION
  - ABOUT

CE Calculators



TI-Nspire™ CX



Additional Supported Commands Not Found in the Hub Menu

- Additional **SET** Commands
  - FORMAT ERROR STRING/NUMBER
  - FORMAT ERROR NOTE/QUIET

- FLOW [TO] ON/OFF
- OUT1/2/3 [TO]

- Additional **READ** Commands
  - ANALOG.OUT
  - BUZZER
  - COLOR
    - RED
    - GREEN
    - BLUE
  - DCMOTOR i
  - DIGITAL.OUT i
  - FORMAT
  - FLOW
  - IN1/IN2/IN3
  - LAST ERROR
  - LED i
  - LIGHT
  - OUT1/2/3
  - PWR
  - RELAY i
  - RESOLUTION
  - RGB i
    - RED i
    - GREEN i
    - BLUE i
  - SERVO i
  - SERVO i CALIBRATION
  - SOUND
  - SPEAKER i
  - SQUAREWAVE i

- Additional **AVERAGE** Commands
  - PERIOD

- Additional **CALIBRATE** Commands
  - CALIBRATE

- SERVO i minimum maximum
- TEMPERATURE i c1 c2 c3 r
- THERMISTOR i c1 c2 c3 r

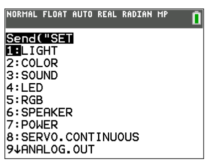
SET

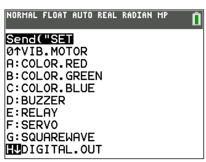
The **SET** command is used to generate outputs on pins or ports, or control output devices such as **LED**s, Servo motors, speaker tones, or other output operations. It is also used to control a variety of system settings. These include formatting of error information, and communications flow control. **SET** does NOT generate any response that requires reading. The success or failure of a **SET** command may be determined by sending a **READ LAST ERROR** command and obtaining the response to that command. The sensors, controls, and settings that **SET** can operate against are in the following table.

SET something'

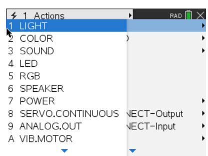
Command:	**SET**
Command Syntax:	**SET**
Code Sample:	
Range:	
Describe:	Used to set options, or output states, or provide information used to control an external actuator or output device, such as turning on a **RELAY**.
Result:	
Type or Addressable Component:	

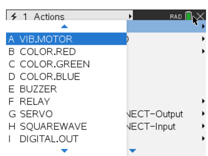
CE Calculators





TI-Nspire™ CX





LIGHT [TO] ON/OFF

Command:	LIGHT [TO] ON/OFF
Command Syntax:	**SET LIGHT ON [[BLINK|TOGGLE] frequency] [[TIME] seconds]** **SET LIGHT OFF** - same as **LED**, but for on-board red **LED**.
Range:	
Describe:	Provides control over the on-board digital **RED LED**. Set optional blink frequency and duration. **SET LIGHT ON [[BLINK|TOGGLE] frequency] [[TIME] seconds]** **SET LIGHT OFF**
Result:	Turns on LIGHT. Turns off LIGHT
Type or Addressable Component:	Control

COLOR [TO] r g b [[BLINK|TOGGLE] frequency] [[TIME] seconds]

Command:	COLOR [TO] r g b [[BLINK|TOGGLE] frequency] [[TIME] seconds]
Command Syntax:	**SET COLOR r g b [[BLINK|TOGGLE] frequency] [[TIME]seconds]** **SET COLOR.component x [[BLINK|TOGGLE] frequency] [[TIME]seconds]**
Range:	
Describe:	On-board **COLOR RGB LED** with sub-components **.RED**, **.GREEN**, **.BLUE**. Can have a blink frequency, and blink time for entire item, or for each component individually, as well as PWM levels given individually, or at one time.
Result:	Where r g b is r-value g-value b-value respectively, or operators from ON/OFF/UP/DOWN/STOP.
Type or Addressable Component:	Control

See Also:

COLOR.RED [TO] r [[BLINK|TOGGLE] frequency] [[TIME] seconds]

Command:	**COLOR.RED [TO] r [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Command Syntax:	**Send("SET COLOR.RED...")** **ON/OFF/UP/DOWN/STOP/0-255 (red element)** **[BLINK frequency] (in Hz)** **[TIME duration] (in secs)**
Range:	
Describe:	RED component of On-board **COLOR RGB LED**. Can have a blink frequency, and blink time for entire item, or for each component individually, as well as PWM levels given individually, or at one time.
Result:	Where r is red level, or operators from ON/OFF/UP/DOWN/STOP.
Type or Addressable Component:	Control

COLOR.GREEN [TO] g [[BLINK|TOGGLE] frequency] [[TIME] seconds]

Command:	**COLOR.GREEN [TO] g [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Command Syntax:	**SET COLOR.GREEN [TO] g [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Range:	
Describe:	GREEN component of On-board **COLOR RGB LED**. Can have a blink frequency, and blink time for entire item, or for each component individually, as well as PWM levels given individually, or at one time.
Result:	Where g is green level, or operators from ON/OFF/UP/DOWN/STOP.
Type or Addressable Component:	Control

COLOR.BLUE [TO] b [[BLINK|TOGGLE] frequency] [[TIME] seconds]

Command:	**COLOR.BLUE [TO] b [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Command Syntax:	**SET COLOR.BLUE [TO] b [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Range:	
Describe:	BLUE component of On-board **COLOR RGB LED**. Can have a blink frequency, and blink time for entire item, or for each component individually, as well as PWM levels given individually, or at one time.
Result:	Where b is blue level, or operators from ON/OFF/UP/DOWN/STOP.
Type or Addressable Component:	Control

SOUND [TO] frequency [[TIME] seconds]

Command:	**SOUND [TO] frequency [[TIME] seconds]**
Command Syntax:	**SET SOUND frequency [[TIME] seconds]**
Range:	
Describe:	**SOUND** is the on-board speaker and can generate a sound with a specified frequency. If not specified, sound will play for 1 second default. **SET SOUND frequency [[TIME] seconds]**
Result:	Play tone through on-board speaker.
Type or Addressable Component:	Control

SOUND OFF/0

Command:	SOUND OFF/0
Command Syntax:	**SET SOUND 0**
Range:	
Describe:	**SOUND** is the on-board speaker and can generate a sound with a specified frequency. If not specified, sound will play for 1 second default. **SET SOUND 0** – turns off sound on internal speaker immediately.
Result:	Stop playing sound.
Type or Addressable Component:	Control

LED i [TO] ON/OFF

Command:	LED i [TO] ON/OFF
Command Syntax:	**SET LED i ON/ OFF [[BLINK|TOGGLE] frequency] [[TIME] seconds]** – digital LED (on or off only)
Range:	
Describe:	Provides control over an external **LED** to set optional blink frequency and duration, as well as **PWM** capability if the associated pin connected to the **LED** supports it. **SET LED i ON [[BLINK|TOGGLE] frequency] [[TIME] seconds]** – digital LED (on or off only) **SET LED i OFF** – turns off LED (same as SET LED i 0).
Result:	Turns on LED. Turns off LED When connected to an Analog-PWM pin.
Type or Addressable Component:	Control

LED i [TO] 0-255

Command:	**LED i [TO] 0-255**
Command Syntax:	**SET LED i 0-255 [[BLINK|TOGGLE] frequency] [[TIME] seconds]** – analog LED (pwm duty cycyle)
Range:	
Describe:	Provides control over an external **LED** to set optional blink frequency and duration, as well as **PWM** capability if the associated pin connected to the **LED** supports it. **SET LED i 0-255 [[BLINK|TOGGLE] frequency] [[TIME] seconds]** – analog LED (pwm duty cycyle)
Result:	When connected to an Analog-PWM pin.
Type or Addressable Component:	Control

RGB

Command:	**CONNECT RGB**
Command Syntax:	**CONNECT RGB**
Range	n/a
Describe:	This command configures the sketch to use the TI-RGB Array. The array needs to be pre-connected through the BB port. An incorrect connection will result in an error indication.
Result:	The RGB array is now available for use in the program.
Type or Addressable Component:	Sensor TI-RGB Array Data Sheet

SPEAKER i [TO] frequency [[TIME] seconds]

Command:	SPEAKER i [TO] frequency [[TIME] seconds]
Command Syntax:	**SET SPEAKER i [TO] frequency [[TIME] seconds]**
Range:	
Describe:	Same as **SOUND** above, except sound is played on an external speaker attached to a digital output pin, available on any **IN/OUT** port, or the breadboard connector port. **Note**: On-board SOUND and external SPEAKER cannot be used concurrently.
Result:	Play tone with frequency given, optional duration in milliseconds, default = 1 second.
Type or Addressable Component:	Control

POWER

Command:	POWER i [TO] 0-100
Command Syntax:	**SET POWER 1 n** where **n** is the intensity of the output from 0 - 100 **SET POWER 1 50** – set the power to 50% of the maximum.
Range	0 – 100
Describe:	**POWER** is used to control the output power and it typically used with a **MOSFET** and a battery source. It can be used to control the output to devices like a motor or a pump.
Result:	Controls the output intensity of the device connected through the **MOSFET**.
Type or Addressable Component:	Control

SERVO i [TO] position

Command:	**SERVO i [TO] position**
Command Syntax:	**SET SERVO i [TO] position**.
Code Sample:	
Range:	
Describe:	Servo motor control interface. Servos can be either continuous or sweep style servos. Position = value from -90 to 90, ranged to -90 to 90) - used with **SWEEP SERVOS**
Result:	Sweep servos: position is a value from -90 to 90. Value 0 is same as specifying **ZERO**.
Type or Addressable Component:	Control

SERVO i [TO] STOP

Command:	**SERVO i [TO] STOP**
Command Syntax:	**SET SERVO i STOP**
Code Sample:	`Send("SET SERVO 1 STOP")`
Range:	
Describe:	Servo motor control interface. Servos can be either continuous or sweep style servos. **Note**: Sweep style servos will stop automatically at the end of the sweep. **SET SERVO i STOP** – stops motion on servo
Result:	Halt any continuous servo operation in progress. Turn **SERVO** Off
Type or	Control

Command:	SERVO i [TO] STOP
Addressable Component:	

SERVO i [TO] ZERO

Command:	SERVO i [TO] ZERO
Command Syntax:	**SET SERVO i ZERO/position**
Code Sample:	`Send("SET SERVO 1 ZERO")`
Range:	
Describe:	Set servo to zero position on sweep servo, or no motion on continuous servo.
Result:	Sweep servos: position is a value from -90 to 90. Value 0 is same as specifying **ZERO**.
Type or Addressable Component:	Control

SERVO i [TO] [CW/CCW] speed [[TIME] seconds]

Command:	SERVO i [TO] [CW/CCW] speed [[TIME] seconds]
Command Syntax:	**SET SERVO i CW/CCW speed [[TIME] seconds]**
Code Sample:	`Send("SET SERVO.CONTINUOUS 1 CW 100 TIME 3")` `Wait 3`
Range:	
Describe:	Speed from -100 to 100, **CW/CCW** optional, if speed <0, **CCW**, else **CW** unless **CW/CCW** keyword is specified,

Command:	SERVO i [TO] [CW/CCW] speed [[TIME] seconds]
	TIME optional, in seconds, default=1 second (for continuous servo operation) (**CW/CCW** required if TIME/seconds NOT specified.)
Result:	Continuous servo where direction of rotation is specified, along with speed, from 0 (no motion) to 100 (fastest). Optional time parameter used to specify how long the servo should rotate in seconds.
Type or Addressable Component:	Control

ANALOG.OUT i [TO]

Command:	ANALOG.OUT i [TO]
Command Syntax:	**SET ANALOG.OUT i 0-255 [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Range:	
Describe:	Software (or hardware, if available) generated pulse-width modulation output at 490 Hz with the specified duty cycle between 0 (off) and 255 (on). The PWM output can be toggled at a frequency from 0.1 to 20.0 Hz for a given duration. If no duration is given, the PWM continues until stopped or turned off. **SET ANALOG.OUT i 0-255 [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Result:	Generate pwm value (hw or sw) on analog output object.
Type or Addressable Component:	Control

ANALOG.OUT i OFF|STOP

Command:	ANALOG.OUT i OFF|STOP
Command Syntax:	**SET ANALOG.OUT i OFF** **SET ANALOG.OUT i STOP**
Range:	

Command:	**ANALOG.OUT** i OFF|STOP
Describe:	Software (or hardware, if available) generated pulse-width modulation output at 490 Hz with the specified duty cycle between 0 (off) and 255 (on). The PWM output can be toggled at a frequency from 0.1 to 20.0 Hz for a given duration. If no duration is given, the PWM continues until stopped or turned off. **SET ANALOG.OUT i OFF** **SET ANALOG.OUT i STOP**
Result:	Turn off pwm on associated pin, including blinking, etc.
Type or Addressable Component:	Control

VIB.MOTOR i [TO] PWM

Command:	**VIB.MOTOR i [TO] PWM**
Command Syntax:	**SET VIB.MOTOR i [TO] PWM**
Range:	**PWM** from 0 (none) and 255 (full on)
Describe:	Vibration motor control interface.
Result:	Vibrations : intensity is a value from 0 to 255.
Type or Addressable Component:	Control

VIB.MOTOR i [TO] OFF|STOP

Command:	**VIB.MOTOR i [TO] OFF|STOP**
Command Syntax:	**SET VIB.MOTOR i OFF|STOP**
Range:	
Describe:	Vibration motor control interface. **SET VIB.MOTOR i OFF|STOP** – stops motion on vibrations

Command:	**VIB.MOTOR i [TO] OFF|STOP**
Result:	Shut down vibration motor.
Type or Addressable Component:	Control

VIB.MOTOR i [TO] 0-255/UP/DOWN/ON/OFF [[BLINK|TOGGLE] freq] [[TIME] seconds]

Command:	**VIB.MOTOR i [TO] 0-255/UP/DOWN/ON/OFF [[BLINK|TOGGLE] freq] [[TIME] seconds]**
Command Syntax:	**SET VIB.MOTOR i 0-255/UP/DOWN/ON/OFF [[BLINK|TOGGLE] freq] [[TIME] seconds]**
Range:	**PWM** from 0 (none) and 255 (full on)
Describe:	Run vibration motor with numerous options
Result:	Run vibration motor with numerous options Optional time parameter used to specify how long the vibration should rotate in seconds.
Type or Addressable Component:	Control

RGB i [TO] r g b [[BLINK|TOGGLE] frequency] [[TIME] seconds]

Command:	**RGB i [TO] r g b [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Command Syntax:	**SET RGB i r g b [[BLINK|TOGGLE] frequency] [[TIME]seconds]**
Range:	
Describe:	External **RGB LED** controls, with same options as available for the on-board COLOR object. Individual color components can be addressed with the same index value i by name, **RED i**, **GREEN i**, **BLUE**

Command:	RGB i [TO] r g b [[BLINK|TOGGLE] frequency] [[TIME] seconds]
	i.
Result:	Where r g b is r-value g-value b-value respectively, or operators from ON/OFF/STOP.
Type or Addressable Component:	Control

RED i [TO] ON/OFF/UP/DOWN/value [[BLINK|TOGGLE] frequency] [[TIME] seconds]

Command:	RED i [TO] ON/OFF/UP/DOWN/value [[BLINK|TOGGLE] frequency] [[TIME] seconds]
Command Syntax:	**SET.RED i [TO] ON/OFF/UP/DOWN/value [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Range:	
Describe:	RED component of External RGB LED controls, with same options as available for the on-board COLOR object. Individual color components can be addressed with the same index value i by name, RED i, GREEN i, BLUE i.
Result:	
Type or Addressable Component:	Control

GREEN i [TO] ON/OFF/UP/DOWN/value [[BLINK|TOGGLE] frequency] [[TIME] seconds]

Command:	GREEN i [TO] ON/OFF/UP/DOWN/value [[BLINK|TOGGLE] frequency] [[TIME] seconds]
Command Syntax:	**SET.GREEN i [TO] ON/OFF/UP/DOWN/value [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Range:	
Describe:	GREEN component of External RGB LED controls, with same options as available for the on-board COLOR object. Individual color

Command:	GREEN i [TO] ON/OFF/UP/DOWN/value [[BLINK|TOGGLE] frequency] [[TIME] seconds]
	components can be addressed with the same index value i by name, RED i, GREEN i, BLUE i.
Result:	
Type or Addressable Component:	Control

BLUE i [TO] ON/OFF/UP/DOWN/value [[BLINK|TOGGLE] frequency] [[TIME] seconds]

Command:	BLUE i [TO] ON/OFF/UP/DOWN/value [[BLINK|TOGGLE] frequency] [[TIME] seconds]
Command Syntax:	**SET.BLUE i [TO] ON/OFF/UP/DOWN/value [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Range:	
Describe:	BLUE component of External RGB LED controls, with same options as available for the on-board COLOR object. Individual color components can be addressed with the same index value i by name, RED i, GREEN i, BLUE i.
Result:	
Type or Addressable Component:	Control

BUZZER i [TO] ON [TIME seconds]

Command:	BUZZER i [TO] ON [TIME seconds]
Command Syntax:	**SET BUZZER i ON [[TIME] seconds]**
Range:	
Describe:	Used to turn ON or OFF a tone on an active BUZZER for either 1 second default, or given length of time. **SET BUZZER i ON [[TIME] seconds]**

Command:	BUZZER i [TO] ON [TIME seconds]
Result:	Sound tone on **ACTIVE** buzzer for 1 second, or specified duration in seconds.
Type or Addressable Component:	Control

BUZZER i [TO] OFF

Command:	BUZZER i [TO] OFF
Command Syntax:	**SET BUZZER i OFF**
Range:	
Describe:	Used to turn ON or OFF a tone on an active BUZZER for either 1 second default, or given length of time. **SET BUZZER i OFF**
Result:	Turn off tone on active buzzer.
Type or Addressable Component:	Control

BUZZER i [TO] ON [TIME seconds]

Command:	BUZZER i [TO] ON [TIME seconds]
Command Syntax:	**SET BUZZER i ON [[TIME] seconds]**
Range:	
Describe:	Used to turn ON or OFF a tone on an active BUZZER for either 1 second default, or given length of time. **SET BUZZER i ON [[TIME] seconds]**
Result:	Sound tone on **ACTIVE** buzzer for 1 second, or specified duration in seconds.
Type or Addressable Component:	Control

BUZZER i [TO] OFF

Command:	**BUZZER i [TO] OFF**
Command Syntax:	**SET BUZZER i OFF**
Range:	
Describe:	Used to turn ON or OFF a tone on an active BUZZER for either 1 second default, or given length of time. **SET BUZZER i OFF**
Result:	Turn off tone on active buzzer.
Type or Addressable Component:	Control

RELAY i [TO] ON/OFF

Command:	**RELAY i [TO] On/Off**
Command Syntax:	**SET RELAY i ON/OFF /0/1 [[TIME] seconds]**.
Range:	Turns the specified **RELAY ON** or **OFF** for the given specified **TIME** in seconds.
Describe:	Control interface to an external RELAY control. **SET RELAY i ON/OFF/1/0 [[TIME] seconds]**
Result:	Turns RELAY on or off
Type or Addressable Component:	Control RELAY

SQUAREWAVE i [TO] frequency [duty [[TIME] seconds]]

Command:	SQUAREWAVE i [TO] frequency [duty [[TIME] seconds]]
Command Syntax:	**SET SQUAREWAVE i frequency [duty]**
Range:	
Describe:	**SQUAREWAVE** is used to generate a square wave form with a default duty cycle of 50% with frequencies from 0.1 Hz to 500 Hz. frequencies slower than 0.1 Hz are set to 0.1 Hz. frequencies above 500 Hz are set to 500 Hz. The optional duty cycle is a value from 1 to 99. **SET SQUAREWAVE i frequency [duty]**
Result:	Generate a digital squarewave from 1 to 500 hz at 1-99 duty cycle on up to 6 pins (i=1-4) duty=50% default, seconds=1.0 default.
Type or Addressable Component:	Control

SQUAREWAVE i OFF

Command:	SQUAREWAVE i OFF
Command Syntax:	**SET SQUAREWAVE i OFF** **frequency [duty]**
Range:	
Describe:	**SQUAREWAVE** is used to generate a square wave form with a default duty cycle of 50% with frequencies from 0.1 Hz to 500 Hz. frequencies slower than 0.1 Hz are set to 0.1 Hz. frequencies above 500 Hz are set to 500 Hz. The optional duty cycle is a value from 1 to 99. **SET SQUAREWAVE i OFF** – turn off squarewave generation
Result:	Stop generating squarewave output.
Type or Addressable Component:	Control

DIGITAL.OUT i [TO] ON/OFF/HIGH/LOW/[[BLINK|TOGGLE] frequency] [[TIME] seconds]

Command:	**DIGITAL.OUT i [TO] ON/OFF/HIGH/LOW/[[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Command Syntax:	**SET DIGITAL.OUT i [TO] ON/OFF/HIGH/LOW [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Range:	
Describe:	Used to generate output digital signal(s). **SET DIGITAL.OUT i ON/OFF/HIGH/LOW [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Result:	Digital.out operations.
Type or Addressable Component:	Control

DIGITAL.OUT i [TO] OUTPUT/CLOCK

Command:	**DIGITAL.OUT i [TO] OUTPUT/CLOCK**
Command Syntax:	**SET DIGITAL.OUT i [TO] OUTPUT/CLOCK**
Range:	
Describe:	Output or drive a clock pulse - digital.out other operations.
Result:	Output or drive a clock pulse - digital.out other operations.
Type or Addressable Component:	Control

DIGITAL.IN i [TO] INPUT/PULLUP/PULLDOWN

Command:	**DIGITAL.IN i [TO] INPUT/PULLUP/PULLDOWN**
Command Syntax:	**SET DIGITAL.IN i [TO] INPUT/PULLUP/PULLDOWN**

Command:	**DIGITAL.IN i [TO] INPUT/PULLUP/PULLDOWN**
Range:	
Describe:	Used for Pulldown and/or pullup control for digital.in operations.
Result:	Pulldown and pullup control for digital.in operations.
Type or Addressable Component:	Control

DIGITAL.OUT i [TO] ON/OFF/HIGH/LOW/[[BLINK|TOGGLE] frequency] [[TIME] seconds]

Command:	**DIGITAL.OUT i [TO] ON/OFF/HIGH/LOW/[[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Command Syntax:	**SET DIGITAL.OUT i [TO] ON/OFF/HIGH/LOW [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Range:	
Describe:	Used to generate output digital signal(s). **SET DIGITAL.OUT i ON/OFF/HIGH/LOW [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Result:	Digital.out operations.
Type or Addressable Component:	Control

DIGITAL.OUT i [TO] OUTPUT/CLOCK

Command:	**DIGITAL.OUT i [TO] OUTPUT/CLOCK**
Command Syntax:	**SET DIGITAL.OUT i [TO] OUTPUT/CLOCK**
Range:	
Describe:	Output or drive a clock pulse - digital.out other operations.
Result:	Output or drive a clock pulse - digital.out other operations.

Command:	DIGITAL.OUT i [TO] OUTPUT/CLOCK
Type or Addressable Component:	Control

DIGITAL.IN i [TO] INPUT/PULLUP/PULLDOWN

Command:	DIGITAL.IN i [TO] INPUT/PULLUP/PULLDOWN
Command Syntax:	**SET DIGITAL.IN i [TO] INPUT/PULLUP/PULLDOWN**
Range:	
Describe:	Used for Pulldown and/or pullup control for digital.in operations.
Result:	Pulldown and pullup control for digital.in operations.
Type or Addressable Component:	Control

DIGITAL.OUT i [TO] ON/OFF/HIGH/LOW/[[BLINK|TOGGLE] frequency] [[TIME] seconds]

Command:	DIGITAL.OUT i [TO] ON/OFF/HIGH/LOW/[[BLINK|TOGGLE] frequency] [[TIME] seconds]
Command Syntax:	**SET DIGITAL.OUT i [TO] ON/OFF/HIGH/LOW [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Range:	
Describe:	Used to generate output digital signal(s). **SET DIGITAL.OUT i ON/OFF/HIGH/LOW [[BLINK|TOGGLE] frequency] [[TIME] seconds]**
Result:	Digital.out operations.
Type or Addressable Component:	Control

DIGITAL.OUT i [TO] OUTPUT/CLOCK

Command:	**DIGITAL.OUT i [TO] OUTPUT/CLOCK**
Command Syntax:	**SET DIGITAL.OUT i [TO] OUTPUT/CLOCK**
Range:	
Describe:	Output or drive a clock pulse - digital.out other operations.
Result:	Output or drive a clock pulse - digital.out other operations.
Type or Addressable Component:	Control

DIGITAL.IN i [TO] INPUT/PULLUP/PULLDOWN

Command:	**DIGITAL.IN i [TO] INPUT/PULLUP/PULLDOWN**
Command Syntax:	**SET DIGITAL.IN i [TO] INPUT/PULLUP/PULLDOWN**
Range:	
Describe:	Used for Pulldown and/or pullup control for digital.in operations.
Result:	Pulldown and pullup control for digital.in operations.
Type or Addressable Component:	Control

AVERAGING [TO] n

Command:	**AVERAGING [TO] n** **Advanced user**
Command Syntax:	**AVERAGING.[TO] n**
Range:	
Describe:	Global setting for how many times we sample analog inputs when

Command:	AVERAGING [TO] n **Advanced user**
	obtaining a reading from a sensor using analog input **n** - (global default)
Result:	Sample analog inputs '**n**' times, averaging results (default is 3 unless changed; sets "global" averaging value.)
Type or Addressable Component:	Setting Default if not set with this command is 3
Note:	Global averaging value can be individually overridden by sensor by using the **AVERAGING** command on an item.

BBPORT

Command:	SET BBPORT [TO] nn [MASK value]
Command Syntax:	**SET BBPORT TO 100** **SET BBPORT TO 0X80**
Range	
Describe:	The **SET** operation on **BBPORT** is used to set the respective bits of the BB port to a 1 or 0 value based on the value given, the optional **MASK** (which is used to specify which pins are being used as digital outputs) and the internal connection mask specified in the **CONNECT BBPORT** operation.
Result:	
Type or Addressable Component:	Control

DCMOTOR i [TO] frequency [duty [[TIME] seconds]]

Command:	DCMOTOR i [TO] frequency [duty [[TIME] seconds]]
Command Syntax:	**SET DCMOTOR i frequency [duty]**
Range:	
Describe:	Generates a specific frequency and duty cycle digital pulse to a motor. **SET DCMOTOR i frequency [duty]**
Result:	Generate a digital pulse at given frequency from 1 to 500 hz at 1-99 % duty cycle; shares number-space with SQUAREWAVE. duty=50% default, seconds=1.0 default.
Type or Addressable Component:	Control

DCMOTOR i OFF

Command:	DCMOTOR i OFF
Command Syntax:	**SET DCMOTOR i OFF**

Command:	DCMOTOR i OFF
Range:	
Describe:	Generates a specific frequency and duty cycle digital pulse to a motor. **SET DCMOTOR i OFF**
Result:	Stop motor.
Type or Addressable Component:	Control

MAGNETIC

Command:	MAGNETIC i [TO] IN n
Command Syntax:	**CONNECT MAGNETIC 1 TO IN 1**
Range	
Describe:	The **MAGNETIC** sensor is used to detect the presence of a magnetic field. It uses the Hall effect. It is also known as a Hall effect sensor.
Result:	The **MAGNETIC** sensor is now available to use.
Type or Addressable Component:	Sensor

VERNIER

Command:	CONNECT VERNIER i TO IN n
Command Syntax:	**CONNECT VERNIER 1 TO IN 1 AS LIGHT** **CONNECT VERNIER 2 TO IN 2 AS ACCEL** **CONNECT VERNIER 1 TO IN 1 AS ENERGY**
Range	
Describe:	This command is used when a Vernier analog sensor is connected to the TI-Innovator™ Hub through the TI-SensorLink

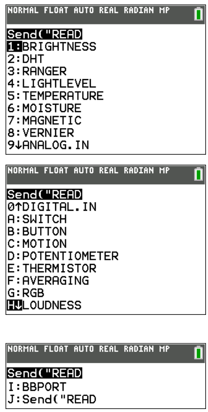
Command:	**CONNECT VERNIER i TO IN n**
	There is support for three additional Vernier analog sensors • LS-BTA • LGA-BTA • VES-BTA
Result:	
Type or Addressable Component:	Sensor

READ

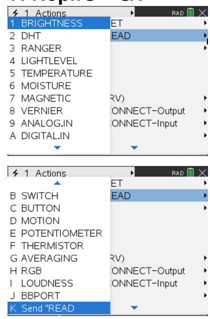
The **READ** command generates responses based on what is being requested.

Tells the Innovator to obtain data from the specified sensor, control, port, pin, or status information including the setup of the hub, such as flow control, error settings, etc. Must be followed by a Get() operation to receive the requested data.

CE Calculators



TI-Nspire™ CX



BRIGHTNESS

Command:	**BRIGHTNESS**
Command Syntax:	**READ BRIGHTNESS**
Range:	
Describe:	Returns the current internal reading from the on-board ambient light sensor. Note the optional keywords of **RANGE** and **AVERAGE** can be appended to the command to return the current **RANGE** setting for the **BRIGHTNESS** sensor if set or the current **AVERAGE** value applied when reading the ADC to obtain the reading. **READ BRIGHTNESS**
Result:	Read on-board light sensor level.
Type or	Control

Command:	BRIGHTNESS
Addressable Component:	

BRIGHTNESS AVERAGE

Command:	BRIGHTNESS AVERAGE **Advanced user**
Command Syntax:	**READ BRIGHTNESS.AVERAGE**
Range:	
Describe:	Returns the current internal reading from the on-board ambient light sensor. Note the optional keywords of **RANGE** and **AVERAGE** can be appended to the command to return the current **RANGE** setting for the **BRIGHTNESS** sensor if set or the current **AVERAGE** value applied when reading the ADC to obtain the reading. **READ BRIGHTNESS AVERAGE**
Result:	Read on-board light sensor level.
Type or Addressable Component:	Control

BRIGHTNESS RANGE

Command:	BRIGHTNESS RANGE **Advanced user**
Command Syntax:	**READ BRIGHTNESS.RANGE**
Range:	
Describe:	Returns the current internal reading from the on-board ambient light sensor. Note the optional keywords of **RANGE** and **AVERAGE** can be

Command:	**BRIGHTNESS RANGE** **Advanced user**
	appended to the command to return the current **RANGE** setting for the **BRIGHTNESS** sensor if set or the current **AVERAGE** value applied when reading the ADC to obtain the reading. **READ BRIGHTNESS RANGE**
Result:	Read on-board light sensor level.
Type or Addressable Component:	Control

DHT i

Command:	**DHT i**
Command Syntax:	**READ DHT i**
Range:	Temperature reading default is in Celsius Humidity reading from 0 to 100 %
Describe:	Returns a list consisting of the current temperature, humidity, type of sensor, and last cached read status. The temperature and humidity can be obtained by themselves by appending the TEMPERATURE or HUMIDITY keywords to the end of the command. The type of sensor is indicated by a 1 for a DHT11, and a 2 for DHT22 style sensors. The status values are: 1=OK, 2=Timeout, 3=Checksum/bad reading. **READ DHT i** – returns full cached information from last reading the DHT task obtained. **READ DHT i TEMPERATURE** – returns latest temperature reading. **READ DHT i HUMIDITY** – returns latest humidity reading.
Result:	Return list with current temperature in C, humidity in %, type (1=DHT11, 2=DHT22), and status (type/status only available in full list). Where the status = 1:OK, =2:Timeout, =3:Checksum.
Type or Addressable Component:	Sensor

DHT i TEMPERATURE

Command:	**DHT i TEMPERATURE**
Command Syntax:	**READ DHT i TEMPERATURE**
Range:	Temperature reading default is in Celsius Humidity reading from 0 to 100 %
Describe:	Returns a list consisting of the current temperature, humidity, type of sensor, and last cached read status. The temperature and humidity can be obtained by themselves by appending the TEMPERATURE or HUMIDITY keywords to the end of the command. The type of sensor is indicated by a 1 for a DHT11, and a 2 for DHT22 style sensors. The status values are: 1=OK, 2=Timeout, 3=Checksum/bad reading. **READ DHT i** – returns full cached information from last reading the DHT task obtained. **READ DHT i TEMPERATURE** – returns latest temperature reading. **READ DHT i HUMIDITY** – returns latest humidity reading.
Result:	Returns temperature component.
Type or Addressable Component:	Sensor

DHT i HUMIDITY

Command:	**DHT i HUMIDITY**
Command Syntax:	**READ DHT i HUMIDITY**
Range:	Temperature reading default is in Celsius Humidity reading from 0 to 100 %
Describe:	Returns a list consisting of the current temperature, humidity, type of sensor, and last cached read status. The temperature and humidity can be obtained by themselves by appending the TEMPERATURE or HUMIDITY keywords to the end of the command. The type of sensor is indicated by a 1 for a DHT11, and a 2 for DHT22

Command:	DHT i HUMIDITY
	style sensors. The status values are: 1=OK, 2=Timeout, 3=Checksum/bad reading. **READ DHT i** – returns full cached information from last reading the DHT task obtained. **READ DHT i TEMPERATURE** – returns latest temperature reading. **READ DHT i HUMIDITY** – returns latest humidity reading.
Result:	Returns humidity component.
Type or Addressable Component:	Sensor

RANGER i

Command:	RANGER i
Command Syntax:	**READ RANGER i**
Range:	
Describe:	Return the current distance measurement from the specified ultrasonic ranging device; distance in meters. If no measurement is made due to the distance being too far; a value of 0 will be returned. Valid measurements are in +meters.
Result:	Read distance in meters from distance sensor.
Type or Addressable Component:	Sensor

LIGHTLEVEL i

Command:	LIGHTLEVEL i
Command Syntax:	**READ LIGHTLEVEL i**
Range:	An integer value between 0 and 16383 (14 bit resolution)
Describe:	Returns the current **ADC** value from the specified external light sensor. External light sensors may be analog, or I2C (BH1750FVI I2C Light sensor). When an analog sensor is present, it is generally assumed to be a photodiode. Additionally, the light level sensor may have **AVERAGE** and/or **RANGE** values specified. These can be obtained by appending the **AVERAGE** or **RANGE** keywords to the **READ** command. **READ LIGHTLEVEL i** **READ LIGHTLEVEL i AVERAGE** **READ LIGHTLEVEL i RANGE**
Result:	Read analog value of light sensor (uses averaging), or I2C (value in **LUX** returned).
Type or Addressable Component:	Sensor

LIGHTLEVEL i AVERAGE

Command:	LIGHTLEVEL i AVERAGE **Advanced user**
Command Syntax:	**READ LIGHTLEVEL i AVERAGE**
Range:	An integer value between 0 and 16383 (14 bit resolution)
Describe:	Returns the current **ADC** value from the specified external light sensor. External light sensors may be analog, or I2C (BH1750FVI I2C Light sensor). When an analog sensor is present, it is generally assumed to be a photodiode. Additionally, the light level sensor may have **AVERAGE** and/or **RANGE** values specified. These can be obtained by appending the **AVERAGE** or **RANGE** keywords to the **READ** command. **READ LIGHTLEVEL i AVERAGE**

Command:	**LIGHTLEVEL i AVERAGE** **Advanced user**
Result:	Read analog value of light sensor (uses averaging), or I2C (value in **LUX** returned).
Type or Addressable Component:	Sensor

LIGHTLEVEL i RANGE

Command:	**LIGHTLEVEL i RANGE** **Advanced user**
Command Syntax:	**READ LIGHTLEVEL i RANGE**
Range:	An integer value between 0 and 16383 (14 bit resolution)
Describe:	Returns the current **ADC** value from the specified external light sensor. External light sensors may be analog, or I2C (BH1750FVI I2C Light sensor). When an analog sensor is present, it is generally assumed to be a photodiode. Additionally, the light level sensor may have **AVERAGE** and/or **RANGE** values specified. These can be obtained by appending the **AVERAGE** or **RANGE** keywords to the **READ** command. **READ LIGHTLEVEL i RANGE**
Result:	Read analog value of light sensor (uses averaging), or I2C (value in **LUX** returned).
Type or Addressable Component:	Sensor

TEMPERATURE i

Command:	**TEMPERATURE i**
Command	**READ TEMPERATURE i**

Command:	**TEMPERATURE i**
Syntax:	
Range:	Temperature reading default is in Celsius. The range depends on the specific temperature sensor being used. Humidity reading from 0 to 100 %
Describe:	Returns the current temperature reading from the associated temperature sensor. The temperature is given, by default, in Celsius. **READ TEMPERATURE i**
Result:	Return current temperature reading in Celsius.
Type or Addressable Component:	Sensor

TEMPERATURE i AVERAGE

Command:	**TEMPERATURE i AVERAGE** **Advanced user**
Command Syntax:	**READ TEMPERATURE i AVERAGE**
Range:	Temperature reading default is in Celsius. The range depends on the specific temperature sensor being used. Humidity reading from 0 to 100 %
Describe:	Returns the current temperature reading from the associated temperature sensor. The temperature is given, by default, in Celsius. **READ TEMPERATURE i AVERAGE**
Result:	Return current temperature reading in Celsius.
Type or Addressable Component:	Sensor

TEMPERATURE i CALIBRATION

Command:	TEMPERATURE i CALIBRATION **Advanced user**
Command Syntax:	**READ TEMPERATURE i CALIBRATION**
Range:	Temperature reading default is in Celsius. The range depends on the specific temperature sensor being used. Humidity reading from 0 to 100 %
Describe:	Returns the current temperature reading from the associated temperature sensor. The temperature is given, by default, in Celsius.
Result:	Returns list with current {c1,c2,c3,r} values used for connected analog temperature sensor.
Type or Addressable Component:	Sensor

MOISTURE i

Command:	MOISTURE i
Command Syntax:	**READ MOISTURE i**
Range:	An integer value between 0 and 16383 (14 bit resolution)
Describe:	Return the current analog level reported by the moisture sensor specified. Supports the **AVERAGE** and **RANGE** options. **READ MOISTURE i** **READ MOISTURE i AVERAGE** **READ MOISTURE i RANGE**
Result:	Read analog value of moisture sensor (uses averaging).
Type or Addressable Component:	Sensor

MOISTURE i AVERAGE

Command:	**MOISTURE i AVERAGE** **Advanced user**
Command Syntax:	**READ MOISTURE i AVERAGE**
Range:	
Describe:	Return the current analog level reported by the moisture sensor specified. Supports the **AVERAGE** and **RANGE** options. **READ MOISTURE i AVERAGE**
Result:	Read analog value of moisture sensor (uses averaging).
Type or Addressable Component:	Sensor

MOISTURE i RANGE

Command:	**MOISTURE i RANGE**
Command Syntax:	**READ MOISTURE i RANGE**
Range:	
Describe:	Return the current analog level reported by the moisture sensor specified. Supports the **AVERAGE** and **RANGE** options. **READ MOISTURE i RANGE**
Result:	Read analog value of moisture sensor (uses averaging).
Type or Addressable Component:	Sensor

MAGNETIC

Command:	MAGNETIC i
Command Syntax:	**READ MAGNETIC i**
Range	0 or 1 0 – no magnetic field is detected 1 – magnetic field is detected
Describe:	The MAGNETIC sensor is used to detect the presence of a magnetic field. It uses the Hall effect. It is also known as a Hall effect sensor.
Result:	
Type or Addressable Component:	Sensor

VERNIER

Command:	READ VERNIER i
Command Syntax:	**READ VERNIER 1**
Range	Depends on the specific Vernier analog sensor connected to the TI-SensorLink
Describe:	Reads the value from the sensor specified in the command.
Result:	
Type or Addressable Component:	Sensor

ANALOG.IN i

Command:	ANALOG.IN i
Command Syntax:	**READ.ANALOG.IN i**
Range:	

Command:	ANALOG.IN i
Describe:	Generic analog input sensor. **READ ANALOG.IN i** – will return the ADC reading on the analog input associated with the object.
Result:	Reads generic **ANALOG.IN** input object
Type or Addressable Component:	Sensor

ANALOG.IN i AVERAGE

Command:	ANALOG.IN i AVERAGE **Advanced user**
Command Syntax:	**READ.ANALOG.IN i AVERAGE**
Range:	
Describe:	**READ ANALOG IN i AVERAGE** – gets the current averaging value for the object.
Result:	Reads generic **ANALOG.IN** input object
Type or Addressable Component:	Sensor

ANALOG.IN i RANGE

Command:	ANALOG.IN i RANGE **Advanced user**
Command Syntax:	**READ.ANALOG.IN i RANGE**
Range:	
Describe:	**READ ANALOG IN i RANGE** – returns the upper and lower range values associated with the object if specified, or error otherwise
Result:	Reads generic **ANALOG.IN** input object
Type or	Sensor

Command:	ANALOG.IN i RANGE **Advanced user**
Addressable Component:	

ANALOG.OUT i

Command:	ANALOG.OUT i
Command Syntax:	**READ ANALOG.OUT i**
Range:	
Describe:	Returns current PWM duty cycle if the output is on, or 0 if not on.
Result:	Reads current PWM duty cycle on pin, 0 if none.
Type or Addressable Component:	Control

DIGITAL.IN i

Command:	DIGITAL.IN i
Command Syntax:	**READ DIGITAL.IN i**
Range:	
Describe:	Returns the current state of the digital pin connected to the DIGITAL object, or the cached state of the digital output value last SET to the object.
Result:	Return 0 (low), 1 (high).
Type or Addressable Component:	Control/Sensor

SWITCH i

Command:	**SWITCH i**
Command Syntax:	**READ SWITCH i**
Range:	
Describe:	Returns the current state of the associated switch. If the switch is connected, a value of 1 is returned. Not connected returns a value of 0. If the switch was connected since the last reading, but is no longer connected, a value of 2 is returned. **READ SWITCH i**
Result:	Returns state of switch (same status as **BUTTON** object, 0=not pressed, 1=pressed, 2=was pressed).
Type or Addressable Component:	Sensor

BUTTON i

Command:	**BUTTON i**
Command Syntax:	**READ BUTTON i**
Range:	
Describe:	Reads the current cached state of the button. A return value of 0 = *not pressed*, 1 = *currently pressed*, 2 = *was pressed* and released since the last reading. **READ BUTTON i**
Result:	Read state of button/switch n - 0=not pressed, 1=pressed, 2=was pressed.
Type or Addressable Component:	Sensor

MOTION i

Command:	MOTION i
Command Syntax:	**READ MOTION i**
Range:	
Describe:	Return the current **PIR Motion sensor** information. **PIR Motion sensors** are digital in nature, so are treated similar to a button in that the value returned indicates motion presence or not. **0**=*no motion detected*. **1**=*motion detected*. **2**=*motion was detected*.
Result:	Read state of **PIR Motion detector** - 0=no motion, 1=motion, 2=motion was detected but none now.
Type or Addressable Component:	Sensor

POTENTIOMETER i

Command:	POTENTIOMETER i
Command Syntax:	**READ POTENTIOMETER i**
Range:	
Describe:	Read analog value of the potentiometer (linear or rotary). The optional **AVERAGE** and **RANGE** keywords can be appended to the command to obtain the current average count, or mapped range being used, if present, for the given potentiometer. **READ POTENTIOMETER i** **READ POTENTIOMETER i RANGE** **READ POTENTIOMETER i AVERAGE**
Result:	Read analog value of rotary encoder / potentiometer (uses averaging).

Command:	POTENTIOMETER i
Type or Addressable Component:	Sensor

POTENTIOMETER i AVERAGE

Command:	POTENTIOMETER i AVERAGE **Advanced user**
Command Syntax:	**READ POTENTIOMETER i AVERAGE**
Range:	
Describe:	Read analog value of the potentiometer (linear or rotary). The optional **AVERAGE** and **RANGE** keywords can be appended to the command to obtain the current average count, or mapped range being used, if present, for the given potentiometer. **READ POTENTIOMETER i AVERAGE**
Result:	Read analog value of rotary encoder / potentiometer (uses averaging).
Type or Addressable Component:	Sensor

POTENTIOMETER i RANGE

Command:	POTENTIOMETER i RANGE **Advanced user**
Command Syntax:	**READ POTENTIOMETER i RANGE**
Range:	
Describe:	Read analog value of the potentiometer (linear or rotary). The optional **AVERAGE** and **RANGE** keywords can be appended to the command to obtain the current average count, or mapped range being used, if present, for the given potentiometer.

Command:	POTENTIOMETER i RANGE Advanced user
	READ POTENTIOMETER i RANGE
Result:	Read analog value of rotary encoder / potentiometer (uses averaging).
Type or Addressable Component:	Sensor

THERMISTOR i

Command:	THERMISTOR i
Command Syntax:	**READ THERMISTOR i**
Range:	
Describe:	Returns the current temperature reading from the associated thermistor sensor. Temperature is returned in Celsius.
Result:	Return current thermistor temperature in Celsius.
Type or Addressable Component:	Sensor

THERMISTOR i AVERAGE

Command:	THERMISTOR i AVERAGE Advanced user
Command Syntax:	**READ THERMISTOR i AVERAGE**
Range:	
Describe:	Returns the current temperature reading from the associated thermistor sensor. Temperature is returned in Celsius.

Command:	**THERMISTOR i AVERAGE** **Advanced user**
Result:	Return current thermistor temperature in Celsius.
Type or Addressable Component:	Sensor

THERMISTOR i CALIBRATION

Command:	**THERMISTOR i CALIBRATION** **Advanced user**
Command Syntax:	**READ THERMISTOR i CALIBRATION**
Range:	
Describe:	Returns the current temperature reading from the associated thermistor sensor. Temperature is returned in Celsius.
Result:	Returns list with current {c1,c2,c3,r} values used for connected thermistor.
Type or Addressable Component:	Sensor

AVERAGING

Command:	**AVERAGING** **Advanced user**
Command Syntax:	**READ AVERAGING**
Range:	
Describe:	Returns the current global setting for the analog averaging default value.

Command:	AVERAGING **Advanced user**
Result:	Return current oversampling/averaging count for sampling analog inputs (this is the GLOBAL default value currently in use).
Type or Addressable Component:	Setting

LOUDNESS i

Command:	LOUDNESS i
Command Syntax:	**READ LOUDNESS i**
Range:	
Describe:	Return the current analog level reported by the sound loudness level sensor specified. Supports the **AVERAGE** and **RANGE** options. **READ LOUDNESS i** **READ LOUDNESS i AVERAGE** **READ LOUDNESS i RANGE**
Result:	Return level of sound detected by sound sensor.
Type or Addressable Component:	Sensor

LOUDNESS i AVERAGE

Command:	LOUDNESS i **Advanced user**
Command Syntax:	**READ LOUDNESS i AVERAGE**
Range:	

Command:	LOUDNESS i **Advanced user**
Describe:	Return the current analog level reported by the sound loudness level sensor specified. Supports the **AVERAGE** and **RANGE** options. **READ LOUDNESS i AVERAGE**
Result:	Return level of sound detected by sound sensor.
Type or Addressable Component:	Sensor

LOUDNESS i RANGE

Command:	LOUDNESS i RANGE **Advanced user**
Command Syntax:	**READ LOUDNESS i.RANGE**
Range:	
Describe:	Return the current analog level reported by the sound loudness level sensor specified. Supports the **AVERAGE** and **RANGE** options. **READ LOUDNESS i** **READ LOUDNESS i AVERAGE** **READ LOUDNESS i RANGE**
Result:	Return level of sound detected by sound sensor.
Type or Addressable Component:	Sensor

BBPORT

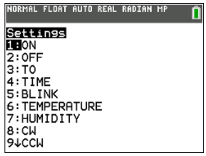
Command:	READ BBPORT
Command Syntax:	**READ BBPORT [MASK value]** **Get B**
Range	
Describe:	Reads the connected pins of the **BBPORT** object as inputs, switching pins from output state to input state. The default connection mask limits the pins that are used in this operation, as does the optional **MASK** value provided.
Result:	
Type or Addressable Component:	Sensor

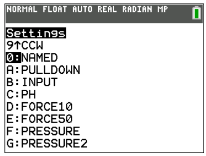
Settings

Settings menu contains operations to set the state of digital and analog pin operations such as the **LED** in the TI-Innovator™ Hub or a connected servo motor movement to states such as ON, OFF, CW (clockwise), and CCW (counterclockwise).

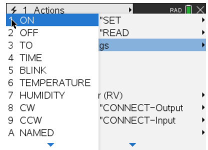
- 1: ON
- 2: OFF
- 3: TO
- 4: TIME
- 5: BLINK
- 6: TEMPERATURE
- 7: HUMIDITY
- 8: CW
- 9: CCW
- 0: NAMED
- A: PULLDOWN
- B: INPUT
- C: PH
- D: FORCE10
- E: FORCE50
- F: PRESSURE
- G: PRESSURE2

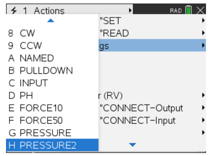
CE Calculators





TI-Nspire™ CX



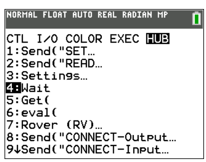


Wait

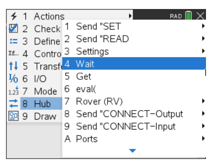
Wait suspends execution of a program for a given time. Maximum time is 100 seconds. During the wait time, the busy indicator is on in the top-right corner of the screen.

Wait may be used in TI-Innovator™ Hub programs to allow time for sensor or control communications prior to the program executing the next command line.

CE Calculators



TI-Nspire™ CX



Wait

Command:	**Wait**
Command Syntax:	Wait *timeInSeconds* Suspends execution for a period of *timeInSeconds* seconds.
Range	0 through 100
Describe:	**Wait** may be used in TI-Innovator™ Hub programs to allow time for sensor or control communications prior to the program executing the next command line. **Wait** is particularly useful in a program that needs a brief delay to allow requested data to become available. The argument *timeInSeconds* must be an expression that simplifies to a decimal value in the range 0 through 100. The command rounds this value up to the nearest 0.1 seconds. **Note**: You can use the **Wait** command within a user-defined program but not within a function.
Result:	**Wait** suspends execution of a program for a given time. Maximum time is 100 seconds. During the wait time, the busy indicator is on in the top-right corner of the screen.
Type or Addressable Component:	Not Applicable

Get(

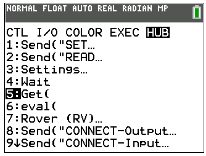
Get(Retrieves a value from a connected TI-Innovator™ Hub and stores the data to a variable on the receiving CE calculator.

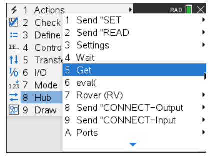
CE Calculators

Get(command definition is specific to the TI-8x calculator and the cable connection via DBus or USB. The CE calculator is USB connectivity only and here, **Get(** is designed for communication with the TI-Innovator™ Hub.

TI-Nspire™ CX

	CE Calculators	**TI-Nspire™ CX**





Get(

Command:	**Get(**
Command Syntax:	**CE Calculators**: **Get(***variable***)** **TI-Nspire™ CX** platform: **Get** [*promptString*,] *var*[, *statusVar*] **Get** [*promptString*,] *func*(*arg1*, *...argn*) [, *statusVar*]
Range	
Describe:	
Result:	Programming command: Retrieves a value from a connected TI-Innovator™ Hub and assigns the value to variable *var*. The value must be requested: • In advance, through a **Send "READ ..."** command. — or — • By embedding a **"READ ..."** request as the optional *promptString* argument. This method lets you use a single command to request the value and retrieve it. (**TI-Nspire™ CX** platform only). Implicit simplification takes place. For example, a received string of "123" is interpreted as a numeric value. **The information below applies only on the TI-Nspire CX platform**: To preserve the string, use **GetStr** instead of **Get**. If you include the optional argument *statusVar*, it is assigned a value based on the success of the operation. A value of zero means that no data was received. In the second syntax, the *func()* argument allows a program to store the received string as a function definition. This syntax operates as if the program executed the command: Define *func(arg1, ...argn) = received string*

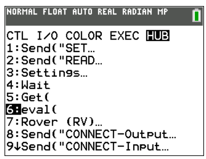
Command:	Get(
	The program can then use the defined function *func()*. **Note**: You can use the **Get** command within a user-defined program but not within a function.
Type or Addressable Component:	All input devices.

eval(

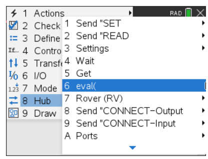
The software evaluates expression *Expr* and replaces the **eval()** statement with the result as a character string.

The argument *Expr* must simplify to a real number.

CE Calculators



TI-Nspire™ CX

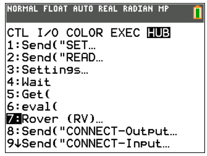
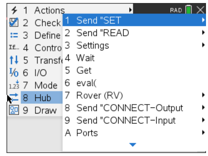


eval(

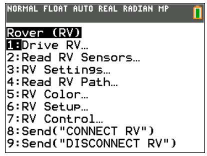
Command:	eval(
Command Syntax:	**eval(***Expr***)** ⇒ **string**
Range	
Describe:	The software evaluates expression *Expr* and replaces the **eval()** statement with the result as a character string. The argument *Expr* must simplify to a real number. **CE Calculators**: **eval()** can be used as a standalone command outside a TI-Innovator™ Hub command. **TI-Nspire™ CX** platform: **eval()** is valid only in the TI-Innovator™ Hub Command argument of programming commands **Get**, **GetStr**, and **Send**.

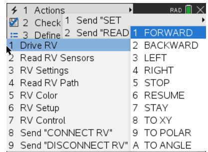
Command:	eval(
Result:	**CE Calculators**: For debugging purposes, using the command line Disp Ans immediately after a command line using Send(displays the complete string being sent. **TI-Nspire™ CX** platform: Although **eval()** does not display its result, you can view the resulting Hub command string after executing the command by inspecting any of the following special variables. *iostr.SendAns* *iostr.GetAns* *iostr.GetStrAns*
Type or Addressable Component:	Not Applicable

ROVER (RV) Menu

Rover (RV)…	CE Calcolatrici	TI-Nspire™ CX
		

- Drive RV…
- Read RV Sensors…
- RV Settings…
- Read RV Path…
- RV Color…
- RV Setup…
- RV Control…
- Send("CONNECT RV")
- Send("DISCONNECT RV")





Drive RV...

RV Drive Command Families

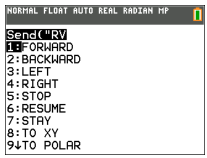
- Base Drive Commands (in the spirit of Turtle Graphics)
  - FORWARD, BACKWARD, RIGHT, LEFT, STOP, STAY
- Math Coordinate Drive Commands
  - Turn to Angle

Note: Drive commands have options for Speed, Time and Distance as appropriate

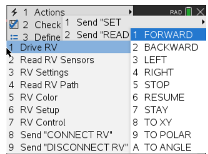
- See RV Settings for Machine-Level Control Commands
  - Set Left and Right Motor values for direction (CW/CCW) and level (0-255,Coast)
  - Read accumulated values for wheel encoder edges and gyro heading change.

- **Drive RV…**
  - Send("RV
    - FORWARD
    - BACKWARD
    - LEFT
    - RIGHT
    - STOP
    - RESUME
    - STAY
    - TO XY
    - TO POLAR
    - TO ANGLE

CE Calcolatrici



TI-Nspire™ CX



RV FORWARD

Command:	RV FORWARD
Command Syntax:	**RV FORWARD [[SPEED s] [DISTANCE d] [TIME t]]**

Code Samples:	`Send ("RV FORWARD 0.5 M")`
	`Send ("RV FORWARD SPEED 0.22 M/S TIME 10")`
	
	`[SET] RV FORWARD`
	`[SET] RV FORWARD [DISTANCE] d [M|UNIT|REV]`
	`[SET] RV FORWARD [DISTANCE] d [M|UNIT|REV] SPEED s.ss [M/S|[UNIT/S]|REV/S]`
	`[SET] RV FORWARD [DISTANCE] d [M|UNIT|REV] TIME t`
	`[SET] RV FORWARD SPEED s [M/S|UNIT/S|REV/S] [TIME t]`
	`[SET] RV FORWARD TIME t [SPEED s.ss [M/S|[UNIT/S]|REV/S]]`

Range:	N/A
Describe:	RV moves forward a given distance (default 0.75 m). Default distance if specified is in UNIT (grid units). Optional M=meters, UNIT=grid-unit, REV=wheel-revolution. Default speed is 0.20 m/sec, max is 0.23 m/sec, min is 0.14 m/sec. Speed may be given and specified in meters/second, unit/second, revolutions/second.
Result:	Action to make the RV move in a forward direction
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV BACKWARD

Command:	RV BACKWARD
Command Syntax:	**RV BACKWARD**

Code Sample:	`Send("RV BACKWARD 0.5 M")`
	`Send("RV BACKWARD SPEED 0.22 M/S TIME 10")`
	--- 	
	`[SET] RV BACKWARD`
	`[SET] RV BACKWARD [DISTANCE] d [M|UNIT|REV]`
	`[SET] RV BACKWARD [DISTANCE] d [M|UNIT|REV]`
	`      SPEED s.ss [M/S|[UNIT/S]|REV/S]`
	`[SET] RV BACKWARD [DISTANCE] d [M|UNIT|REV]`
	`      TIME t`
	`[SET] RV BACKWARD SPEED s.ss`
	`      [M/S|UNIT/S|REV/S] [TIME t]`
	`[SET] RV BACKWARD TIME t`
	`      [SPEED s.ss [M/S|UNIT/S|REV/S]]`

Range:	N/A
Describe:	RV moves backward a given distance (default 0.75 m). Default distance if specified is in UNIT (grid units). Optional M=meters, UNIT=grid-unit, REV=wheel-revolution. Default speed is 0.20 m/sec, max is 0.23 m/sec, min is 0.14 m/sec. Speed may be given and specified in meters/second, unit/second, revolutions/second.
Result:	Action to make the RV move in a backward direction.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV LEFT

Command:	**RV LEFT**
Command Syntax:	**RV LEFT**

Code Sample:	`Send "RV LEFT"`
	
	`[SET] RV LEFT [ddd [DEGREES]]`
	`[SET] RV LEFT [rrr RADIANS]`
	`[SET] RV LEFT [ggg GRADIANS]`

Range:	N/A
Describe:	Default turn is 90 degrees unless DEGREES, RADIANS, or GRADIANS keyword is present, and then the value is converted internally to degrees format from the specified units. Value given is ranged to a value between 0.0 and 360.0 degrees. The turn will be executed as a SPIN motion.
Result:	Turn Rover to the LEFT.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV RIGHT

Command:	**RV RIGHT**
Command Syntax:	**RV RIGHT**

Code Sample:	`Send "RV RIGHT"`
	
	`[SET] RV RIGHT [ddd [DEGREES]]`
	`[SET] RV RIGHT [rrr RADIANS]`
	`[SET] RV RIGHT [ggg GRADIANS]`

Range:	N/A
Describe:	Default turn is 90 degrees unless DEGREES, RADIANS, or GRADIANS keyword is present, and then the value is converted internally to degrees format from the specified units. Value given is ranged to a value between 0.0 and 360.0 degrees. The turn will be executed as a SPIN motion.

Command:	RV RIGHT
Result:	Turn Rover to the RIGHT.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV STOP

Command:	RV STOP
Command Syntax:	**RV STOP**

Code Sample:	`Send "RV STOP"`
	
	`[SET] RV STOP`
	
	`[SET] RV STOP CLEAR`

Range:	N/A
Describe:	The **RV** will stop any current movement immediately. That movement can be resumed from where it left off with a **RESUME** operation. Any movement commands will cause the queue to flush immediately, and begin the just-posted new movement operation
Result:	Stop processing Rover commands from the command queue, and leave pending operations in the queue. (immediate action). Queue can be resumed by **RESUME**.The **RV** will stop any current movement immediately. That movement can be resumed from where it left off with a **RESUME** operation. Any movement commands will cause the queue to flush immediately, and begin the just-posted new movement operation. Stop processing Rover commands from the command queue, and flush any pending operations left in the queue. (immediate action).
Type or Addressable Component:	Control **Note:** This Rover control command is executed immediately.

RV RESUME

Command:	RV RESUME
Command Syntax:	**RV RESUME**

Code Sample:	`Send "RV RESUME"`
	
	`[SET] RV RESUME`

Range:	N/A
Describe:	Enable processing of Rover commands from the command queue. (immediate action), or resume (see RV STAY) operation.
Result:	Resume operation.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV STAY

Command:	RV STAY
Command Syntax:	**RV STAY**

Code Sample:	`Send "RV STAY"`
	
	`[SET] RV STAY [[TIME] s.ss]`

Range:	N/A
Describe:	Tells RV to "stay" in place for an optionally specified amount of time in seconds. Default is 30.0 seconds.
Result:	RV stays in position.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV TO XY

Command:	RV TO XY
Command Syntax:	**RV TO XY** x-coordinate y-coordinate [[SPEED] s.ss [UNIT/S] |M/S|REV/S] [XYLINE]

Code Sample:	`Send "RV TO XY 1 1"`
	`Send "RV TO XY eval(X) eval(Y)"`
	`Send "RV TO XY 2 2 SPEED 0.23 M/S"`

Range:	-327 to +327 for X and Y coordinates
Describe:	This command controls the movement of Rover on a virtual grid. Default location at start of program execution is (0,0) with Rover facing the positive x-axis. The x and y coordinates match the current grid size (default: 0.1 M/grid unit). Grid size can be changed through "SET RV.GRID.M/UNIT" command The speed parameter is optional.
Result:	Moves Rover from current grid location to the specified grid location.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV TO POLAR

Command:	RV TO POLAR
Command Syntax:	**RV TO POLAR** R-coordinate Theta-coordinate [[DEGREES] |RADIANS|GRADS] [[SPEED] s.ss [UNIT/S]|M/S|REV/S] [XYLINE]

Code Sample:	`Send("RV TO POLAR 5 30") - r = 5 units, theta = 30 degrees`
	`Send("RV TO POLAR 5 2 RADIANS")`
	`Send("RV TO POLAR eval(sqrt(3^2+4^2)) eval (tan-1(4/3) DEGREES ")`

Range:	Theta-coordinate: -360 to +360 degrees R-coordinate: -327 to +327
Describe:	Moves the RV from its current position to the specified polar position relative to that position. The RV's X/Y position will be updated to reflect the new position. The "r" coordinate matches the current grid size (default: 0.1 M/grid

Command:	RV TO POLAR
	unit). Default location at start of program execution is (0,0) with Rover facing the positive x-axis. Default unit of theta is Degrees. The speed parameter is optional.
Result:	Moves Rover from current grid location to the specified grid location.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV TO ANGLE

Command:	RV TO ANGLE
Command Syntax:	**RV TO ANGLE**

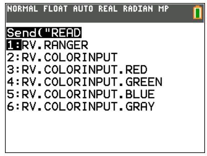
Code Sample:	`Send "RV TO ANGLE"`
	
	`[SET] RV TO ANGLE rr.rr [[DEGREES]|RADIANS|GRADIANS]`

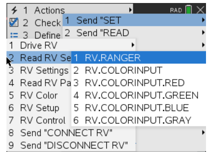
Range:	N/A
Describe:	
Result:	Spins the RV to the specified angle from current heading.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

READ RV Sensors...

SEND("Read Sensor Commands

- Reading of low level sensors for learning foundations of robotics.





- **Read RV Sensors…**
  - Send("READ
    - RV.RANGER
    - RV.COLORINPUT
    - RV.COLORINPUT.RED
    - RV.COLORINPUT.GREEN
    - RV.COLORINPUT.BLUE
    - RV.COLORINPUT.GRAY
- **RV.RANGER:** Returns value in Meters.
- **RV.COLORINPUT:** Reads color sensor that is built into the RV.

RV.RANGER

Command:	**RV.RANGER**	
Command Syntax:	**RV.RANGER**	
Code Sample:	`Send("READ RV.RANGER")` `Get(R)`	
	Connects the Rover Vehicle to the TI-Innovator™ Hub. This establishes connections with the motor driver, color sensor, gyroscope, ultrasonic ranger, and proximity sensors.	`CONNECT RV`
	Returns the current distance from the front of the RV to an obstacle. If there is no obstacle detected, a range of 10.00 meters is reported	`READ RV.RANGER` `Get(R)`

Command:	RV.RANGER
Range:	N/A
Describe:	The front-facing ultrasonic distance sensor. Returns measurements in meters. ~10.00 meters means no obstacle was detected.
Result:	Returns value in Meters.
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT

Command:	RV.COLORINPUT
Command Syntax:	**RV.COLORINPUT**
Code Sample:	`Send("READ RV.COLORINPUT")` `Get(C)`
Range:	1 thru 9
Describe:	Bottom-mounted color sensor detects the color of the surface. Can also detect gray-level scale of black (0) to white (255).
Result:	Returns current color sensor information. The return value is in the 1 – 9 range which maps to the colors below: **Color** — **Return value** Red — 1 Green — 2 Blue — 3 Cyan — 4 Magenta — 5 Yellow — 6 Black — 7 White — 8 Gray — 9
Type or	Sensor

Command:	RV.COLORINPUT
Addressable Component:	**Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT.RED

Command:	RV.COLORINPUT.RED
Command Syntax:	**RV.COLORINPUT.RED**
Code Sample:	`Send("READ RV.COLORINPUT.RED")` `Get(R)`
Range:	0 - 255
Describe:	Detect intensity of individual red components of surface. The results are in 0-255 range.
Result:	Returns current color sensor "red value".
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT.GREEN

Command:	RV.COLORINPUT.GREEN
Command Syntax:	**RV.COLORINPUT.GREEN**
Code Sample:	`Send("READ RV.COLORINPUT.GREEN")` `Get(G)`
Range:	0 - 255
Describe:	Detect intensity of individual green components of surface. The results are in 0-255 range.
Result:	Returns current color sensor "green" value.
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT.BLUE

Command:	RV.COLORINPUT.BLUE
Command Syntax:	**RV.COLORINPUT.BLUE**
Code Sample:	`Send("READ RV.COLORINPUT.BLUE")` `Get(B)`
Range:	0 - 255
Describe:	Detect intensity of individual blue components of surface. The results are in 0-255 range.
Result:	Returns current color sensor "blue" value.
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT.GRAY

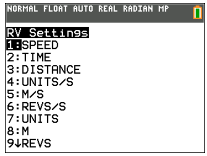
Command:	RV.COLORINPUT.GRAY
Command Syntax:	**RV.COLORINPUT.GRAY**
Code Sample:	`Send("READ RV.COLORINPUT.GRAY")` `Get(G)`
Range:	0 - 255
Describe:	Detect grayness of surface. The result will be in 0-255 range.
Result:	Returns an interpolated "grayscale" value based on 0.3*red + 0.59*green + 0.11*blue 0-black, 255 - white.
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

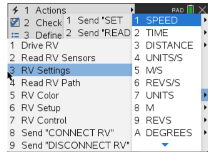
RV Settings...

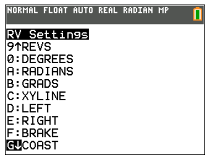
RV Settings Commands

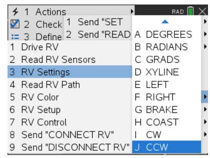
Settings menu for Rover contains other commands that support RV commands such as FORWARD or BACKWARD.

- **RV Settings...**
  - RV Settings
    - SPEED
    - TIME
    - DISTANCE
    - UNIT/S
    - M/S
    - REV/S
    - UNITS
    - M
    - REVS
    - DEGREES
    - RADIANS
    - GRADS
    - XYLINE
    - LEFT
    - RIGHT
    - BRAKE
    - COAST
    - CW
    - CCW









SPEED

Command:	**SPEED**
Command Syntax:	**SPEED**
Code Sample:	`SPEED`
Range:	N/A

Command:	SPEED
Describe:	Speed may be given (default is 0.20 m/sec, max is 0.23 m/sec, min is 0.14 m/sec) and specified in meters/second, unit/second, revolutions/second, or feet/second.
Result:	
Type or Addressable Component:	Setting

TIME

Command:	TIME
Command Syntax:	**TIME**
Code Sample:	`TIME`
Range:	N/A
Describe:	.
Result:	
Type or Addressable Component:	Setting

DISTANCE

Command:	DISTANCE
Command Syntax:	**DISTANCE**
Code Sample:	`DISTANCE`
Range:	N/A
Describe:	
Result:	
Type or Addressable Component:	Setting

UNIT/S

Command:	UNIT/S
Command Syntax:	**UNIT/S**
Code Sample:	`UNIT/S`
Range:	N/A
Describe:	
Result:	
Type or Addressable Component:	Setting

M/S

Command:	M/S
Command Syntax:	**M/S**
Code Sample:	`M/S`
Range:	N/A
Describe:	
Result:	
Type or Addressable Component:	Setting

REV/S

Command:	REV/S
Command Syntax:	**REV/S**
Code Sample:	`REV/S`

Command:	REV/S
Range:	N/A
Describe:	
Result:	
Type or Addressable Component:	Setting

UNITS

Command:	UNITS
Command Syntax:	**UNITS**
Code Sample:	`UNITS`
Range:	N/A
Describe:	
Result:	
Type or Addressable Component:	Setting

M

Command:	M
Command Syntax:	**M**
Code Sample:	`M`
Range:	N/A
Describe:	
Result:	

Command:	M
Type or Addressable Component:	Setting

REVS

Command:	REVS
Command Syntax:	**REVS**
Code Sample:	`REVS`
Range:	N/A
Describe:	Return list of wheel revolutions traveled.
Result:	
Type or Addressable Component:	Setting

DEGREES

Command:	DEGREES
Command Syntax:	**DEGREES**
Code Sample:	`DEGREES`
Range:	N/A
Describe:	
Result:	
Type or Addressable Component:	Setting

RADIANS

Command:	**RADIANS**
Command Syntax:	**RADIANS**
Code Sample:	`RADIANS`
Range:	N/A
Describe:	
Result:	
Type or Addressable Component:	Setting

GRADS

Command:	**GRADS**
Command Syntax:	**GRADS**
Code Sample:	`GRADS`
Range:	N/A
Describe:	
Result:	
Type or Addressable Component:	Setting

XYLINE

Command:	**XYLINE**
Command Syntax:	**XYLINE**
Code Sample:	`XYLINE`

Command:	XYLINE
Range:	N/A
Describe:	
Result:	
Type or Addressable Component:	Setting

LEFT

Command:	LEFT
Command Syntax:	**LEFT**
Code Sample:	`LEFT`
Range:	N/A
Describe:	
Result:	
Type or Addressable Component:	Setting

RIGHT

Command:	RIGHT
Command Syntax:	**RIGHT**
Code Sample:	`RIGHT`
Range:	N/A
Describe:	
Result:	

Command:	RIGHT
Type or Addressable Component:	Setting

BRAKE

Command:	BRAKE
Command Syntax:	**BRAKE**
Code Sample:	`BRAKE`
Range:	N/A
Describe:	
Result:	
Type or Addressable Component:	Setting

COAST

Command:	COAST
Command Syntax:	**COAST**
Code Sample:	`COAST`
Range:	N/A
Describe:	
Result:	
Type or Addressable Component:	Setting

CW

Command:	CW
Command Syntax:	**CW**
Code Sample:	`CW`
Range:	N/A
Describe:	CW.b (value is positive) = Wheel rotate clockwise, backward direction CW.f (value is positive) = Wheel rotates clockwise, forward direction CW.f (value is negative) = Wheel rotates clockwise, forward direction
Result:	
Type or Addressable Component:	Setting

CCW

Command:	CCW
Command Syntax:	**CCW**
Code Sample:	`CCW`
Range:	N/A
Describe:	CCW.f (value is positive) = Wheel rotates counter-clockwise, forward direction CCW.b (value is positive) = Wheel rotates counter-clockwise, backward direction CWW.b (value is negative) = Wheel rotates counter-clockwise, backward direction
Result:	
Type or Addressable Component:	Setting

Read RV Path…

Reading WAYPOINT and PATH

Tracking the RV's Path

In order to support analysis of the Rover during and after a run, the sketch will automatically measure the following information for each Drive command:

- X Coordinate on virtual grid
- Y Coordinate on virtual grid
- Time in seconds that the current command has been executing.
- Distance in coordinate units for the path segment.
- Heading in degrees (absolute terms measured Counter Clockwise with the X-axis as 0 degrees.
- Revolutions by the wheel in executing the current command
- Command number, tracks the number of commands executed, begins with 0.

The Path values will be stored in lists, starting with the segments associated with the earliest commands and going to the segments associated with the latest commands.

The drive command in progress, the **WAYPOINT**, will repeatedly update the last element in the Path lists as the Rover progresses toward the last waypoint.

When a drive command is completed a new waypoint is initiated and the dimension of the Path lists are incremented.

> **Note:** This implies that when all the drive commands in the queue are completed that another waypoint for the stopped state is automatically started. This is similar to the initial position where the RV is stationary and counting time.

Max number of waypoints: 80

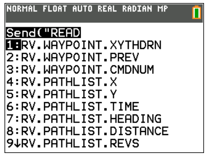
RV Position and Path

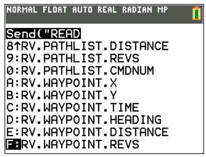
- Ability to read X,Y coordinate, Heading, Time and Distance for each drive command in execution.
- Will store path history in lists for plotting and analysis

Note: Coordinate grid scale can be set by the user, default is 10cm per unit. The user will have options to set the origin of the grid.

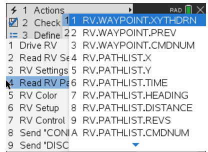
- **Read RV Path…**
  - Send("READ
    - RV.WAYPOINT.XYTHDRN
    - RV.WAYPOINT.PREV
    - RV.WAYPOINT.CMDNUM
    - RV.PATHLIST.X
    - RV.PATHLIST.Y
    - RV.PATHLIST.TIME
    - RV.PATHLIST.HEADING
    - RV.PATHLIST.DISTANCE
    - RV.PATHLIST.REVS
    - RV.PATHLIST.CMDNUM
    - RV.WAYPOINT.X
    - RV.WAYPOINT.Y
    - RV.WAYPOINT.TIME
    - RV.WAYPOINT.HEADING
    - RV.WAYPOINT.DISTANCE
    - RV.WAYPOINT.REVS

CE Calcolatrici





TI-Nspire™ CX



RV.WAYPOINT.XYTHDRN

Command:	RV.WAYPOINT.XYTHDRN
Command Syntax:	**RV.WAYPOINT.XYTHDRN**
Code Sample:	`Send("READ RV.WAYPOINT.XYTHDRN")`
Example:	Getting the distance traveled toward the current way-point from the last way-point
Code Sample:	`Send("READ RV.WAYPOINT.XYTHDRN")` Get(L_1) (L_1)(5)->D
Range:	N/A
Describe:	READ RV.WAYPOINT.XYTHDRN - read the x-coord, y-coord, time, heading, distance traveled, number of wheel revolutions, command number of the current waypoint. Returns a list with all these values as elements.
Result:	Return list of current way-point X, Y coordinates, Time, Heading, Distance, Revolutions, and command number.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.PREV

Command:	RV.WAYPOINT.PREV
Command Syntax:	**RV.WAYPOINT.PREV**
Code Sample:	`Send("READ RV.WAYPOINT.PREV")`
Example:	Getting the distance traveled during the previous way-point.
Code Sample:	`Send("READ RV.WAYPOINT.PREV")` Get(L_1) (L_1)(5)->D

Command:	RV.WAYPOINT.PREV
Range:	N/A
Describe:	READ RV.WAYPOINT.PREV - read the x-coord, y-coord, time, heading, distance traveled, number of wheel revolutions, command number of the previous waypoint. Returns a list with all these values as elements.
Result:	Return list of the previous way-point X, Y coordinates, time, heading, distance, revolutions, and command number.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.CMDNUM

Command:	RV.WAYPOINT.CMDNUM
Command Syntax:	**RV.WAYPOINT.CMDNUM**
Code Sample:	`Send("READ RV.WAYPOINT.CMDNUM")`
Example:	Program to determine if a drive command has completed without referring to a specific command number. **Note:** the **Wait** is intended to increase the probability of catching a difference in the Command Number.

Code Sample:	`Send("RV FORWARD 10")`
	`Send("READ RV.WAYPOINT.CMDNUM")`
	`Get(M)`
	`M->N`
	
	While M=N
	
	`Send("READ RV.WAYPOINT.CMDNUM")`
	`Get(N)`
	`End`
	
	Disp "Drive Command is completed"

Range:	N/A

Command:	RV.WAYPOINT.CMDNUM
Describe:	READ RV.WAYPOINT.CMDNUM - returns the last command number of the current waypoint.
Result:	Returns a value of 0 if the RV is currently "working" on a command and is either in motion, or running a STAY operation. This command will return a value of 1 when ALL queued operations are completed, nothing is remaining in the command queue, and the current operation has completed (and immediately after CONNECT RV).
Type or Addressable Component:	Returns Data

RV.PATHLIST.X

Command:	RV.PATHLIST.X
Command Syntax:	**RV.PATHLIST.X**
Code Samples:	`Send("READ RV.PATHLIST.X")`
Example:	Program to plot the RV path on the graph screen
Code Samples:	Plot1(xyLine, L_1, L_2,▫,BLUE) Send("READ RV.PATHLIST.X") Get(L1) Send("READ RV.PATHLIST.Y") Get(L2) DispGraph
Range:	N/A
Describe:	READ RV.PATHLIST.X - returns a list of X values from the beginning to and including the current Waypoint X value.
Result:	Return list of X coordinates traversed since last **RV.PATH CLEAR** or initial **CONNECT RV**.
Type or Addressable Component:	Returns Data

RV.PATHLIST.Y

Command:	RV.PATHLIST.Y
Command Syntax:	**RV.PATHLIST.Y**
Code Sample:	`Send("READ RV.PATHLIST.Y")`
Example:	Program to plot the RV path on the graph screen

Code Sample:	`Plot1(xyLine, L₁, L₂,▫,BLUE)`
	`Send("READ RV.PATHLIST.Y")`
	`Get(L1)`
	`Send("READ RV.PATHLIST.X")`
	`Get(L2)`
	`DispGraph`

Range:	N/A
Describe:	READ RV.PATHLIST.Y - returns a list of Y values from the beginning to and including the current Waypoint Y value.
Result:	Return list of Y coordinates traversed since last **RV.PATH CLEAR** or initial **CONNECT RV**.
Type or Addressable Component:	Returns Data

RV.PATHLIST.TIME

Command:	RV.PATHLIST.TIME
Command Syntax:	**RV.PATHLIST.TIME**
Code Sample:	`Send "READ RV.PATHLIST.TIME"`
Range:	N/A
Describe:	READ RV.PATHLIST.TIME - returns a list of the time in seconds from the beginning to and including the current Waypoint time value.

Command:	RV.PATHLIST.TIME
Result:	Return list of cumulative travel times for each successive way-point.
Type or Addressable Component:	Returns Data

RV.PATHLIST.HEADING

Command:	RV.PATHLIST.HEADING
Command Syntax:	**RV.PATHLIST.HEADING**
Code Sample:	`Send "READ RV.PATHLIST.HEADING"`
Range:	N/A
Describe:	READ RV.PATHLIST.HEADING - returns a list of the headings from the beginning to and including the current Waypoint heading value.
Result:	Return list of cumulative angular headings taken.
Type or Addressable Component:	Returns Data

RV.PATHLIST.DISTANCE

Command:	RV.PATHLIST.DISTANCE
Command Syntax:	**RV.PATHLIST.DISTANCE**
Example:	Getting the cumulative distance traveled since the beginning of a journey by the RV
Code Sample:	`Send "READ RV.PATHLIST.DISTANCE"` Get(L_1) sum(L_1)
Range:	N/A

Command:	RV.PATHLIST.DISTANCE
Describe:	READ RV.PATHLIST.DISTANCE - returns a list of the distances traveled from the beginning to and including the current Waypoint distance value.
Result:	Return list of cumulative distances traveled.
Type or Addressable Component:	Returns Data

RV.PATHLIST.REVS

Command:	RV.PATHLIST.REVS
Command Syntax:	**RV.PATHLIST.REVS**
Code Sample:	`Send "READ RV.PATHLIST.REVS"`
Range:	N/A
Describe:	READ RV.PATHLIST.REVS - returns a list of the number of revolutions traveled from the beginning to and including the current Waypoint revolutions value.
Result:	Return list of wheel revolutions traveled.
Type or Addressable Component:	Returns Data

RV.PATHLIST.CMDNUM

Command:	RV.PATHLIST.CMDNUM
Command Syntax:	**RV.PATHLIST.CMDNUM**
Code Sample:	`Send "READ RV.PATHLIST.CMDNUM"`
Range:	N/A
Describe:	READ RV.PATHLIST.CMDNUM - returns a list of command numbers

Command:	RV.PATHLIST.CMDNUM
	for the path
Result:	Return list of commands used to travel to the current way-point entry. 0 - Start of Way-points (if first action is a STAY, then no START is given, but a STAY will be shown instead.) 1 - Travel forward 2 - Travel backward 3 - Left spin motion 4 - Right spin motion 5 - Left turn motion 6 - Right turn motion 7 - Stay (no motion) the time the RV stays at the current position is given in the TIME list. 8 - RV is currently in motion on this way-point traversal.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.X

Command:	RV.WAYPOINT.X
Command Syntax:	**RV.WAYPOINT.X**
Code Samples:	`Send("READ RV.WAYPOINT.X")`
Range:	N/A
Describe:	READ RV.WAYPOINT.X - returns x coordinate of current waypoint.
Result:	Return current way-point X coordinate.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.Y

Command:	RV.WAYPOINT.Y
Command Syntax:	**RV.WAYPOINT.Y**
Code Samples:	`Send("READ RV.WAYPOINT.Y")`
Range:	N/A
Describe:	READ RV.WAYPOINT.Y - returns x coordinate of current waypoint.
Result:	Return current way-point Y coordinate.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.TIME

Command:	RV.WAYPOINT.TIME
Command Syntax:	**RV.WAYPOINT.TIME**
Code Sample:	`Send("READ RV.WAYPOINT.TIME")`
Range:	N/A
Describe:	READ RV.WAYPOINT.TIME - returns time spent traveling from previous to current waypoint
Result:	Return total cumulative way-point travel time value in seconds.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.HEADING

Command:	RV.WAYPOINT.HEADING
Command	**RV.WAYPOINT.HEADING**

Command:	RV.WAYPOINT.HEADING
Syntax:	
Code Sample:	`Send("READ RV.WAYPOINT.HEADING")`
Range:	N/A
Describe:	READ RV.WAYPOINT.HEADING - returns absolute heading of current waypoint
Result:	Return current absolute heading in degrees. (+h = counter-clockwise, -h = clockwise.)
Type or Addressable Component:	Returns Data

RV.WAYPOINT.DISTANCE

Command:	RV.WAYPOINT.DISTANCE
Command Syntax:	**RV.WAYPOINT.DISTANCE**
Code Sample:	`Send("READ RV.WAYPOINT.DISTANCE")`
Range:	N/A
Describe:	READ RV.WAYPOINT.DISTANCE - returns distance traveled between previous and current waypoint
Result:	Return cumulative total distance traveled in meters.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.REVS

Command:	RV.WAYPOINT.REVS
Command Syntax:	**RV.WAYPOINT.REVS**

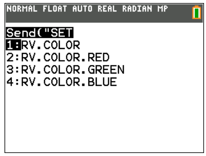
Command:	RV.WAYPOINT.REVS
Code Sample:	`Send("READ RV.WAYPOINT.REVS")`
Range:	N/A
Describe:	READ RV.WAYPOINT.REVS - returns number of revolutions needed to travel between previous and current waypoint
Result:	Return total revolutions of the wheels performed to travel the cumulative distance to the current way-point.
Type or Addressable Component:	Returns Data

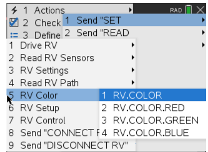
RV Color...

Send("SET Commands

RGB LED on Rover - This supports the same commands and parameters as the RGB LED on the TI-Innovator™ Hub.

- **RV Color...**
  - Send("SET
    - RV.COLOR
    - RV.COLOR.RED
    - RV.COLOR.GREEN
    - RV.COLOR.BLUE





RV.COLOR

Command:	**RV.COLOR**
Command Syntax:	**RV.COLOR**

Code Sample:	`Send "SET RV.COLOR`
	
	`[SET] RV.COLOR rr gg bb [[BLINK] b [[TIME] s.ss]]`

Range:	N/A
Describe:	Set the RGB color to be displayed on the Rover's RGB LED. Same syntax as for all RGB LED operations with COLOR, etc.
Result:	Return the current RGB color, as a three-element list, that is being displayed on the Rover's RGB LED
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV.COLOR.RED

Command:	**RV.COLOR.RED**
Command Syntax:	**RV.COLOR.RED**
Code	`Send "SET RV.COLOR.RED`

Command:	RV.COLOR.RED
Sample:	`[SET] RV.COLOR.RED rr [[BLINK] b [[TIME] s.ss]]`
Range:	N/A
Describe:	
Result:	Set the RED color to be displayed on the Rover's RGB LED.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV.COLOR.GREEN

Command:	RV.COLOR.GREEN
Command Syntax:	**RV.COLOR.GREEN**

Code Sample:	`Send "SET RV.COLOR.GREEN`
	
	`[SET] RV.COLOR.GREEN gg [[BLINK] b [[TIME] s.ss]]`

Range:	N/A
Describe:	
Result:	Set the GREEN color to be displayed on the Rover's RGB LED.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV.COLOR.BLUE

Command:	RV.COLOR.BLUE
Command Syntax:	**RV.COLOR.BLUE**

Command:	RV.COLOR.BLUE

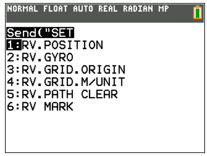
Code Sample:	`Send "SET RV.COLOR.BLUE`
	
	`[SET] RV.COLOR.BLUE bb [[BLINK] b [[TIME] s.ss]]`

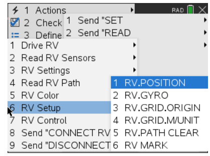
Range:	N/A
Describe:	
Result:	Set the BLUE color to be displayed on the Rover's RGB LED.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV Setup…

Send("SET Commands

- **RV Setup…**
  - Send("SET
    - RV.POSITION
    - RV.GYRO
    - RV.GRID.ORIGIN
    - RV.GRID.M/UNIT
    - RV.PATH CLEAR
    - RV MARK





RV.POSITION

Command:	**RV.POSITION**
Command Syntax:	**RV.POSITION**

Code Sample:	`Send "SET RV.POSITION"`
	
	`[SET] RV.POSITION xxx yyy`
	`[hhh [[DEGREES]|RADIANS|GRADIANS]]`

Range:	N/A
Describe:	Sets the coordinate position and optionally the heading of the Rover on the virtual grid.
Result:	Rover configuration is updated.
Type or Addressable Component:	Setting

RV.GYRO

Command:	**RV.GYRO**
Command Syntax:	**RV.GYRO**
Code Sample:	`Send "SET RV.GYRO"`

Command:	RV.GYRO
Range:	N/A
Describe:	Sets the on-board Gyroscope.
Result:	
Type or Addressable Component:	Control (for Gyroscope)

RV.GRID.ORIGIN

Command:	RV.GRID.ORIGIN
Command Syntax:	**RV.GRID.ORIGIN**

Code Sample:	`Send "SET RV.GRID.ORIGIN"`
	
	`[SET} RV.GRID.ORIGIN`

Range:	N/A
Describe:	Sets RV as being at current grid origin point of (0,0). The "heading" is set to 0.0 resulting in the current position of the RV now set to pointing down a virtual x-axis toward positive x values.
Result:	
Type or Addressable Component:	Setting

RV.GRID.M/UNIT

Command:	RV.GRID.M/UNIT
Command Syntax:	**RV.GRID.M/UNIT**

Code Sample:	`Send "SET RV.GRID.M/UNIT"`
	
	`[SET] RV.GRID.M/UNIT nnn`

Command:	RV.GRID.M/UNIT
Range:	N/A
Describe:	Set the size of a "grid unit" on the virtual grid. This setting is used by Rover when driving on the virtual grid. Default value is 0.1 (0.1M or 10 cm per grid unit). A value of 0.05 means 5 cm per unit grid. A value of 5 means 5M per grid unit. The maximum allowable value is 10.0 (for 10 meters per grid unit) and the lowest allowable value is 0.01 (for 1 cm per grid unit).
Result:	
Type or Addressable Component:	Setting

RV.PATH CLEAR

Command:	RV.PATH CLEAR
Command Syntax:	**RV.PATH CLEAR**

Code Sample:	`Send "SET RV.PATH CLEAR"`
	
	`[SET] RV.PATH CLEAR`

Range:	N/A
Describe:	Clears any pre-existing path / waypoint information. Recommended before doing a sequence of movement operations where waypoint / path-list information is desired.
Result:	
Type or Addressable Component:	Setting

RV MARK

Command:	RV MARK
Command Syntax:	**RV MARK**
Code	`Send "SET RV MARK"`

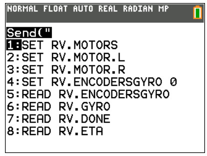
Command:	RV MARK
Sample:	`[SET] RV MARK [[TIME] s.ss]`
Range:	N/A
Describe:	Enable RV to make a "mark" with a pen at the specified time interval (default is 1 second if not specified). A time value of 0.0 turns OFF marking. Marking ONLY happens if the Rover is moving in a forward direction.
Result:	
Type or Addressable Component:	Setting (for Rover)

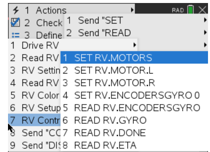
RV Control…

SEND(" Commands

Wheel commands and other commands relevant for learning foundations of the Rover vehicle.

- **RV Control …**
  - Send("
    - SET RV.MOTORS
    - SET RV.MOTOR.L
    - SET RV.MOTOR.R
    - SET RV.ENCODERSGYRO 0
    - READ RV.ENCODERSGYRO
    - READ RV.GYRO





SET RV.MOTORS

Command:	**SET RV.MOTORS**
Command Syntax:	**SET RV.MOTORS**

Code Sample:	`Send "SET RV.MOTORS"`
	
	`[SET] RV.MOTORS [LEFT][CW|CCW]`
	`<pwm value|BRAKE|COAST>`
	`[RIGHT][CW|CCW]`
	`<pwm value|BRAKE|COAST>`
	`[DISTANCE ddd [M|[UNITS]|REV|FT]]`
	`| [TIME s.ss]`

Range:	N/A
Describe:	Set left or right or both motor PWM values. Negative values imply **CCW** and Positive values imply **CW**. Left **CW**=backward motion. Left **CCW**=forward motion. Right CW=forward motion, Right **CCW**=backward motion. PWM values may be numeric from -255 to +255, or keywords "**COAST**" or "**BRAKE**". Value of 0 is stop (coast). Use of the **DISTANCE** option is only available if the **RV** is connected with all sensors. **CONNECT RV MOTORS** means no sensors are available to measure distance, so the **DISTANCE** option is an error in this instance.
Result:	Both the LEFT and RIGHT motor, managed as a single object for

Command:	SET RV.MOTORS
	direct control (advanced) use.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

SET RV.MOTOR.L

Command:	SET RV.MOTOR.L
Command Syntax:	**SET RV.MOTOR.L**

Code Sample:	`Send "SET RV.MOTOR.L"`
	`[SET] RV.MOTOR.L [CW|CCW] <+/-pwm value|BRAKE|COAST>`
	`[TIME s.ss]
		[DISTANCE ddd [[UNITS] |M|REV|FT]]`

Range:	N/A
Describe:	Set left motor direct PWM value. **CCW** = forward, **CW** = backward, pwm value negative = forward, positive = backward. **TIME** option available in all modes, **DISTANCE** option available only when **RV** is fully connected (not the **RV MOTORS** option).
Result:	Left wheel motor and control for direct control (advanced) use.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

SET RV.MOTOR.R

Command:	SET RV.MOTOR.R
Command Syntax:	**SET RV.MOTOR.R**

Code Sample:	`Send "SET RV.MOTOR.R"`
	
	`[SET] RV.MOTOR.R [CW|CCW] <+/-pwm value|BRAKE|COAST>`
	`[TIME s.ss]
		[DISTANCE ddd [[UNITS] |M|REV|FT]]`

Command:	SET RV.MOTOR.R
Range:	N/A
Describe:	Set right motor direct PWM value. **CW** = forward, **CCW** = backward, pwm value positive = forward, negative = backward. **TIME** option available in all modes, **DISTANCE** option available only when **RV** is fully connected (not the **RV MOTORS** option).
Result:	Right wheel motor and control for direct control (advanced) use.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

SET RV.ENCODERSGYRO 0

Command:	SET RV.ENCODERSGYRO 0
Command Syntax:	**SET RV.ENCODERSGYRO 0**
Code Sample:	`Send "SET RV.ENCODERSGYRO 0"`
Range:	N/A
Describe:	Reset the left and right encoder, coupled with the gyro and operating time information.
Result:	
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

READ RV.ENCODERSGYRO

Command:	READ RV.ENCODERSGYRO
Command Syntax:	**READ RV.ENCODERSGYRO**
Code Sample:	`Send "READ RV.ENCODERSGYRO"`

Command:	READ RV.ENCODERSGYRO
Range:	N/A
Describe:	The left and right encoder, coupled with the gyro and operating time information.
Result:	List of values of current left and right encoder, coupled with gyro and operating time information
Type or Addressable Component:	Control **Note:** This Rover READ command is executed immediately.

READ RV.GYRO

Command:	READ RV.GYRO
Command Syntax:	**READ RV.GYRO**

Code Sample:	`Send "READ RV.GYRO"`
	
	`READ RV.GYRO [[DEGREES]|RADIANS|GRADIANS]`

Range:	N/A
Describe:	The gyroscope is used to maintain the heading of Rover while it's in motion. It can also be used to measure the change in angle during turns. The gyroscope is ready to use after the **CONNECT RV** command is processed. The GYRO object shall be usable even when the RV is not in motion.
Result:	Returns current gyro sensor angular deviation from 0.0, reading partially drift-offset compensated.
Type or Addressable Component:	Control **Note:** This Rover READ command is executed immediately.

READ RV.DONE

Command:	READ RV.DONE
Command Syntax:	**READ RV.DONE**
Code Sample:	`Send("READ RV.DONE")`
Example:	**RV.DONE** as an alias for **RV.WAYPOINT.CMDNUM**

Code Sample:	`For n,1,16`
	`Send "RV FORWARD 0.1"`
	`Send "RV LEFT"`
	`EndFor`
	`@ Wait for Rover to finish driving`
	`Send "READ RV.DONE"`
	`Get d`
	`While d=0`
	`Send "READ RV.DONE"`
	`Get d`
	`Wait 0.1`
	`EndWhile`
	`Send "READ RV.PATHLIST"`
	`Get L`

Range:	N/A
Describe:	**RV.DONE** as an alias for **RV.WAYPOINT.CMDNUM** To improve usability a new state variable was created called **RV.DONE**. This is an alias of **RV.WAYPOINT.CMDNUM**.
Result:	
Type or Addressable Component:	Returns Data

See Also: RV.WAYPOINT.CMDNUM

READ RV.ETA

Command:	READ RV.ETA
Command Syntax:	**READ READ RV.ETA**
Code Sample:	`Send("READ RV.ETA")`
Example:	The code sample below returns the estimated time to drive to coordinate (4,4)

Code Sample:	`Send "RV TO XY 4 4"`
	`Send "READ RV.ETA"`
	`Get eta`
	`Disp eta`

	Note: This value will not be exact. It will depend on the surface for one, but it will be a close enough estimate for the expected applications. The value will be time in seconds with a minimum unit of 100 ms.
Example	If a different **READ** command is issued, the value of the variable is overwritten with the information that was requested.

Code Sample:	`Send "RV TO XY 3 4"`
	`Send "READ BRIGHTNESS"`
	`Get eta`

	Note: eta - will contain the value of the **BRIGHTNESS** sensor, not the **RV.ETA** variable
Range:	N/A
Describe:	Calculate the estimated time to complete each Rover command.
Result:	
Type or Addressable Component:	Returns Data

Sample program:

Set **RGB** to red while moving forward, green when turning.

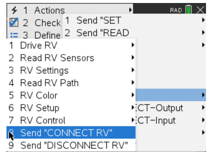
Code Sample:	`For n, 1, 4`
	`Send "RV FORWARD"`
	`Send "READ RV.ETA"`
	`Get eta`
	`Send "SET COLOR 255 0 0"`
	`Wait eta`
	`Send "RV LEFT"`
	`Send "READ RV.ETA"`
	`Get eta`
	`Send "SET COLOR 0 255 0"`
	`Wait eta`
	`EndFor`

Send "CONNECT RV"

SEND("CONNECT RV") Commands

CONNECT RV - initializes the hardware connections.

- Connects RV and inputs and outputs built into the RV.
- Resets the Path and the Grid Origin.
- Sets the units per meter to default value.
- **Send("CONNECT RV")**



CONNECT RV

Command:	**CONNECT RV**
Command Syntax:	CONNECT RV [MOTORS]
Code Sample:	`Send "CONNECT RV"` `Send "CONNECT RV MOTORS"`
Range:	N/A
Describe:	The "**CONNECT RV**" command configures the TI-Innovator™ Hub software to work with the TI-Innovator™ Rover. It establishes the connections to the various devices on the Rover – two motors, two encoders, one gyroscope, one RGB LED and one color sensor. It also clears the various counters and sensor values. The optional 'MOTORS' parameter configures only the motors and allows direct control of motors without the additional peripherals.
Result:	Connects the Rover Vehicle to the TI-Innovator™ Hub. This establishes connections with the motor driver, color sensor, gyroscope, ultrasonic ranger, and RGB LED. The Rover is now ready to be programmed
Type or Addressable Component:	All components of the Rover - two motors, two encoders, one gyroscope, one RGB LED and one color sensor.

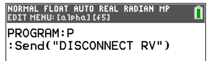
Send "DISCONNECT RV"

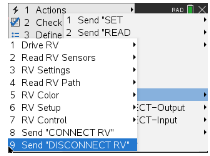
SEND("DISCONNECT RV") Commands

DISCONNECT RV - disconnects all the hardware peripherals from the Hub.

Format: Send("DISCONNECT RV")

- **Send("DISCONNECT RV")**





DISCONNECT RV

Command:	**DISCONNECT RV**
Command Syntax:	DISCONNECT RV

Code Sample:	`Send "DISCONNECT RV"`
	
	`DISCONNECT RV`

Range:	N/A
Describe:	The "**DISCONNECT RV**" command removes the logical connections between the TI-Innovator™ Hub and the TI-Innovator™ Rover. It also clears the counters and sensor values. It allows the use of the breadboard port of the TI-Innovator™ Hub with other devices.
Result:	The TI-Innovator™ Hub is now logically disconnected from the TI-Innovator™ Rover
Type or Addressable Component:	N/A

CONNECT-Output

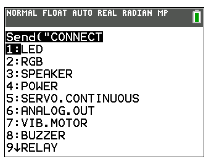
CONNECT associates a given control or sensor with a pin or port on the TI-Innovator. If the specified control or sensor is currently in use, an error will be generated. If the pin or port specified in the **CONNECT** command is currently in use, an error will be generated.

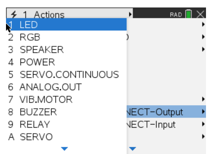
The **CONNECT** command does not generate an active response, but a variety of errors may occur during a connection attempt, such as pin-in-use, unsupported, invalid options, bad options, etc.

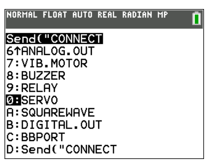
CONNECT 'something i' [TO] IN1/IN2/IN3/OUT1/OUT2/OUT3/BB1

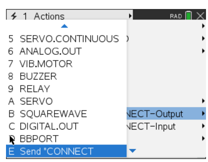
Command:	**CONNECT**
Command Syntax:	**CONNECT**
Range:	
Describe:	Associates a sensor or control with a given port or pin(s). Places the respective pin(s) in use
Result:	
Type or Addressable Component:	

CE Calculators **TI-Nspire™ CX**









LED i [TO] OUT n/BB n

Command:	**LED i [TO] OUT n/BB n**
Command Syntax:	**CONNECT LED i [TO] OUT n/BB n**

Command:	LED i [TO] OUT n/BB n
Range:	
Describe:	This object provides the ability to connect external **LED** objects. The **LED** object is either connected to a **PWM** function (if available, and the pin connecting to supports it), or a digital output pin which will be driven at 50% duty cycle; or the specified blink rate if one is specified in the **SET** operation. **CONNECT LED 1i [TO] BB3** **CONNECT LED 2i [TO] OUT1**
Result:	LED connected to specific port.
Type or Addressable Component:	Control

RGB i / COLOR [TO] BB r BB g BB b

Command:	RGB i / COLOR [TO] BB r BB g BB b
Command Syntax:	**CONNECT RGB i / COLOR [TO] BB r BB g BB b**
Range:	
Describe:	Connects an external **RGB LED** to three **PWM**-capable pins. If insufficient PWM pins are available for mapping to PWM function, an error will be given. To connect an external RGB, the on-board **RGB LED** should be **DISCONNECT**ed before the attempt to connect the external RGB is performed. **CONNECT RGB 1 [TO] BB8 BB9 BB10**
Result:	Digital pins supporting PWM.
Type or Addressable Component:	Control

SPEAKER i [TO] OUT n/BB n

Command:	SPEAKER i [TO] OUT n/BB n
Command Syntax:	**CONNECT SPEAKER i [TO] OUT n/BB n**

Command:	SPEAKER i [TO] OUT n/BB n
Range:	
Describe:	Connect an external speaker for sound generation. Requires a digital output pin. **CONNECT SPEAKER 1 [TO] OUT 1** **CONNECT SPEAKER i [TO] BB 3**
Result:	Connect a speaker to a digital output port or pin.
Type or Addressable Component:	Control

POWER

Command:	CONNECT POWER n [TO] OUT1/OUT2/OUT3
Command Syntax:	**CONNECT POWER n [TO] OUT1/OUT2/OUT3**
Range	
Describe:	Connects a **POWER** object to the specified analog output port. Default **PWM** value is zero.
Result:	The named **POWER** device can be used in the program after a CONNECT command.
Type or Addressable Component:	Control

SERVO.CONTINUOUS i [TO] BB 6

Command:	SERVO.CONTINUOUS i [TO] BB 6
Command Syntax:	**CONNECT SERVO.CONTINUOUS i [TO] BB 6**
Code Sample:	
Range:	

Command:	SERVO.CONTINUOUS i [TO] BB 6
Describe:	Used to connect either a normal sweep servo motor, or a continuous servo motor. External power must be provided before attempting to connect the servo. **CONNECT SERVO.CONTINUOUS i [TO] BB 6**
Result:	Servo motor with -90 to 90 degree movement.
Type or Addressable Component:	Control

ANALOG.OUT i [TO] OUT i/BB i

Command:	ANALOG.OUT i [TO] OUT n/BB n
Command Syntax:	**CONNECT ANALOG.OUT i [TO] OUT n/BB n**
Range:	
Describe:	Connect a generic “analog” output control to a pin/port that supports analog input. **ANALOG.OUT** shares number space with **DCMOTOR** and **SQUAREWAVE** objects. **CONNECT ANALOG.OUT i [TO] OUT 1** **CONNECT ANALOG.OUT i [TO] BB 4** **CONNECT ANALOG.OUT i [TO] BB 1**
Result:	Connect analog output to pin. If pin supports hardware pulse with modulation (**PWM**), the object uses. If the pin does not support hardware-generated **PWM**, the sketch will generate **PWM** in software at 490 Hz with the duty cycle specfic between 0 (none) and 255 (full on).
Type or Addressable Component:	Control

VIB.MOTOR

Command:	VIB.MOTOR i [TO] PWM
Command Syntax:	**SET VIB.MOTOR i [TO] PWM**

Command:	VIB.MOTOR i [TO] PWM
Range:	PWM from 0 (none) and 255 (full on)
Describe:	Vibration motor control interface.
Result:	Vibrations : intensity is a value from 0 to 255.
Type or Addressable Component:	Control

BUZZER i [TO] OUT n/BB n

Command:	BUZZER i [TO] OUT n/BB n
Command Syntax:	**CONNECT BUZZER i [TO] OUT n/BB n**
Range:	
Describe:	Connect an external active buzzer to an output digital pin. Active buzzers play a tone when their signal is set high/on, and stop the tone when the signal is dropped to ground. For piezo or passive buzzers, use the **SPEAKER** object type to allow generation of multiple tones. **CONNECT BUZZER i [TO] OUT1**
Result:	**ACTIVE** buzzers connect to a digital pin.
Type or Addressable Component:	Control

RELAY i [TO] OUT n/BB n

Command:	RELAY i [TO] OUT n/BB n
Command Syntax:	**CONNECT RELAY i [TO] OUT n/BB n**
Range:	
Describe:	With external power required, connect a relay module to a given control signal pin. Since the control is digital, as long as external power is present, any pin may be used.

Command:	RELAY i [TO] OUT n/BB n
	CONNECT RELAY 1 [TO] BB 3 **CONNECT RELAY 1 [TO] OUT 2**
Result:	Relays.
Type or Addressable Component:	Control

SERVO i [TO] OUT n

Command:	SERVO i [TO] OUT n
Command Syntax:	**CONNECT SERVO i [TO] OUT n**
Code Sample:	
Range:	
Describe:	Used to connect either a normal sweep servo motor, or a continuous servo motor. External power must be provided before attempting to connect the servo. **CONNECT SERVO 1 [TO] OUT 1**
Result:	Servo motor is connected to port.
Type or Addressable Component:	Control

SQUAREWAVE i [TO] OUT n/BB n

Command:	SQUAREWAVE i [TO] OUT n/BB n
Command Syntax:	**CONNECT SQUAREWAVE i [TO] OUT n/BB n**
Range:	
Describe:	Connect a software generated digital waveform generator object. These objects share the number-space with the **DCMOTOR** and **ANALOG.OUT** output objects. The associated pin is configured as a

Command:	SQUAREWAVE i [TO] OUT n/BB n
	digital output signal. **CONNECT SQUAREWAVE n [TO] BB 2**
Result:	Digital output squarewave from 1 to 500 hz.
Type or Addressable Component:	Control

DIGITAL.OUT i [TO] OUT n/BB n [[AS] OUTPUT]

Command:	DIGITAL.OUT i [TO] OUT n/BB n [[AS] OUTPUT]
Command Syntax:	**CONNECT DIGITAL.OUT i [TO] OUT n/BB n**
Range:	
Describe:	Connects a generic digital object to a specified pin or port. The connected pin is configured either as a digital output signal, default LOW, or a digital input signal, default INPUT with no pullup or pulldown enabled. The index number can refer to either an input or output. The index is shared by both items since a **DIGITAL** signal can be either an input or output. **CONNECT DIGITAL.OUT 1 [TO] OUT n/BB n**
Result:	Connect pin to digital object default output state, default **OUTPUT**, low.
Type or Addressable Component:	Control/Sensor

BBPORT

Command:	CONNECT BBPORT
Command Syntax:	**CONNECT BBPORT [MASK value]**
Range	
Describe:	When the optional **MASK** is not specified, this command connects all 10 BB pins to the **BBPORT** object as digital I/O pins. The optional **MASK** parameter may be used to selectively connect specific pins. The mask value may be specified in decimal, binary, or hexadecimal format. For example, 1023 or 0X3FF selects all 10 pins and is the default internal mask value used by the **BBPORT** object if a **MASK** is not specified. **Another example:** If only pins BB1 and BB2 are going to be used, a mask value of 3 or 0x03 will select on the two pins.
Result:	If not **MASK** is specified, the program can read/write to all pins of **BBPORT**. If a **MASK** is specified, the program can write to the specified pins.
Type or Addressable Component:	Sensor

DCMOTOR i [TO] OUT n/BB n

Command:	DCMOTOR i [TO] OUT n/BB n
Command Syntax:	**CONNECT DCMOTOR i [TO] OUT n/BB n**
Range:	
Describe:	Connect an external **DC Motor** object. This object requires the presence of power on the external power connector to allow operation. These objects share the number-space with the **SQUAREWAVE** output objects and **ANALOG.OUT** objects. The associated pin is configured as a digital output signal. **CONNECT DCMOTOR i [TO] OUT1**
Result:	Connect **DCMOTOR** to a digital output pin.
Type or Addressable Component:	Control

LIGHT

Command:	LIGHT
Command Syntax:	**CONNECT LIGHT**
Range:	
Describe:	This command is not needed for typical use since the on-board LIGHT (i.e. RED LED) is automatically connected. Re-connect a previously disconnected on-board RED LED. The LIGHT is always connected when the system is reset, or powered-on, or the BEGIN command is used to restore system state. No pin number is required. **CONNECT LIGHT**
Result:	Connects on-board digital LED (red) to known fixed pin. Digital only.
Type or Addressable Component:	Control

COLOR

Command:	COLOR
Command Syntax:	**CONNECT COLOR**
Range:	
Describe:	This command is not needed for typical use since the on-board COLOR LED is automatically connected. (Re-)connect the internal **RGB LED**. No pins are required for this command to operate as the internal pins are known. This sensor is automatically connected when the TI-Innovator is initially powered, and when the **BEGIN** command is used. When disconnected, two **PWM** signals are freed for external use by other pins. **CONNECT COLOR**
Result:	Connects on-board **RGB LED** to fixed pins on board. Uses 3 **PWM**s.
Type or Addressable Component:	Control

SOUND

Command:	**SOUND**
Command Syntax:	**CONNECT SOUND**
Range:	
Describe:	This command is not needed for typical use since the on-board object SOUND is automatically connected. Re-connect the on-board speaker for sound generation. No pin needed as it uses known, fixed pin for signal. **CONNECT SOUND**
Result:	Connects on-board speaker to fixed output digital pin.
Type or Addressable Component:	Control

CONNECT-Input

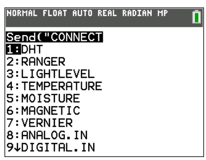
CONNECT associates a given control or sensor with a pin or port on the TI-Innovator. If the specified control or sensor is currently in use, an error will be generated. If the pin or port specified in the **CONNECT** command is currently in use, an error will be generated.

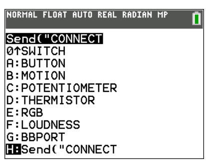
The **CONNECT** command does not generate an active response, but a variety of errors may occur during a connection attempt, such as pin-in-use, unsupported, invalid options, bad options, etc.

CONNECT 'something i' [TO] IN1/IN2/IN3/OUT1/OUT2/OUT3/BB1

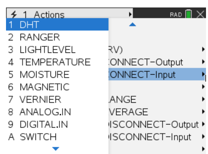
Command:	**CONNECT**
Command Syntax:	**CONNECT**
Range:	
Describe:	Associates a sensor or control with a given port or pin(s). Places the respective pin(s) in use
Result:	
Type or Addressable Component:	

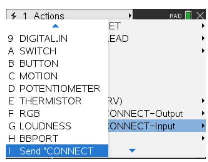
CE Calculators





TI-Nspire™ CX





DHT i [TO] IN n

Command:	**DHT i [TO] IN n**
Command Syntax:	**CONNECT DHT i [TO] IN n**

Command:	DHT i [TO] IN n
Range:	Temperature reading default is in Celsius Humidity reading from 0 to 100 %
Describe:	The **DHT** digital temperature humidity sensor can be connected via this object. The **DHT** can be either a **DHT11** or **DHT22** and is identified automatically when connected to the system via a digital signal line. **CONNECT DHT i [TO] IN1**
Result:	Digital humidity/temperature sensors (DHT11/DHT22, type is auto-detected).
Type or Addressable Component:	Sensor

RANGER i [TO] IN n

Command:	RANGER i [TO] IN n
Command Syntax:	**CONNECT RANGER i [TO] IN n**
Range:	
Describe:	Connect an external ultrasonic distance ranging module to an input port. **CONNECT RANGER 1i [TO] IN 1**
Result:	Ultrasonic ranging sensors with either individual trigger/echo pins, or same pin used for trigger/echo.
Type or Addressable Component:	Sensor

LIGHTLEVEL i [TO] IN n/BB n

Command:	LIGHTLEVEL i [TO] IN n/BB n
Command Syntax:	**CONNECT LIGHTLEVEL i [TO] IN n/BB n**
Range:	An integer value between 0 and 16383 (14 bit resolution)
Describe:	Connects an external light sensor. External light sensors can be

Command:	LIGHTLEVEL i [TO] IN n/BB n
	analog sensors. **CONNECT LIGHTLEVEL 1i [TO] IN1**
Result:	Analog light level sensors is connected to the specific port.
Type or Addressable Component:	Sensor

TEMPERATURE i [TO] IN n/BB n

Command:	TEMPERATURE i [TO] IN n/BB n
Command Syntax:	**CONNECT TEMPERATURE i [TO] IN n/BB n**
Range:	Temperature reading default is in Celsius. The range depends on the specific temperature sensor being used. Humidity reading from 0 to 100 %
Describe:	Connects a temperature sensor to the system using either of several connection methods. **Note:** The default temperature sensor is included in the Breadboard pack If the sensor is based on a thermistor and provides an analog output, it uses a single analog input pin. If the sensor is a DS18B20 digital temperature sensor, it uses a single bi-directional digital GPIO pin. The analog thermistor temperature sensors is by default, assumed to be a PTC thermistor. If the thermistor is an NTC style, an optional keyword can be added to the connect command sequence to change the style of the thermistor. The analog thermistor temperature sensor uses a specific set of thermistor constants, different than those used by the THERMISTOR object, to convert the reading into a temperature reading. The constants are used in the Steinhart-Hart model to convert the analog reading to temperature. <table><tr><th>Description</th><th>Value</th></tr><tr><td>C1</td><td>8.76741e-8</td></tr></table>

Command:	TEMPERATURE i [TO] IN n/BB n
	<table><tr><th>Description</th><th>Value</th></tr><tr><td>C2</td><td>2.34125e-4</td></tr><tr><td>C3</td><td>1.129148e-3</td></tr><tr><td>R1 – reference resistance</td><td>10000.0 ohms</td></tr></table> **CONNECT TEMPERATURE i [TO] IN 1** – thermistor sensor attached to analog input. **CONNECT TEMPERATURE i [TO] BB 1** – DS18B20 digital attached to digital pin. **CONNECT TEMPERATURE i [TO] I2 C** – LM75A attached to I2C port. **CONNECT TEMPERATURE i [TO] BB 5 NTC** – connect an analog temperature sensor to analog input and specifies an NTC style thermistor. **CONNECT TEMPERATURE i [TO] BB 6 PTC** – connect an analog temperature sensor to analog input and specifies a PTC style thermistor.
Result:	Analog temperature sensor.
Type or Addressable Component:	Sensor

MOISTURE i [TO] IN n/BB n

Command:	MOISTURE i [TO] IN n/BB n
Command Syntax:	**CONNECT MOISTURE i [TO] IN n/BB n**
Range:	An integer value between 0 and 16383 (14 bit resolution)
Describe:	Connect an analog moisture sensor to return relative moisture readings. **CONNECT MOISTURE 1i [TO] IN 1**
Result:	Analog moisture sensors.
Type or Addressable Component:	Sensor

MAGNETIC

Command:	**MAGNETIC i [TO] IN n**
Command Syntax:	**CONNECT MAGNETIC 1 TO IN 1**
Range	
Describe:	The **MAGNETIC** sensor is used to detect the presence of a magnetic field. It uses the Hall effect. It is also known as a Hall effect sensor.
Result:	The **MAGNETIC** sensor is now available to use.
Type or Addressable Component:	Sensor

VERNIER

Command:	**CONNECT VERNIER i TO IN n**
Command Syntax:	**CONNECT VERNIER 1 TO IN 1 AS LIGHT** **CONNECT VERNIER 2 TO IN 2 AS ACCEL** **CONNECT VERNIER 1 TO IN 1 AS ENERGY**
Range	
Describe:	This command is used when a Vernier analog sensor is connected to the TI-Innovator™ Hub through the TI-SensorLink There is support for three additional Vernier analog sensors • LS-BTA • LGA-BTA • VES-BTA
Result:	
Type or Addressable Component:	Sensor

ANALOG.IN i [TO] IN n/BB n

Command:	ANALOG.IN i [TO] IN n/BB n
Command Syntax:	**CONNECT ANALOG.IN i [TO] IN n/BB n**
Range:	
Describe:	Connect a generic "analog" input sensor to a pin/port that supports analog input. **CONNECT ANALOG.IN i [TO] IN 1** **CONNECT ANALOG.IN i [TO] BB 5**
Result:	Connect analog input to pin that supports that function (error if pin is not analog-input capable).
Type or Addressable Component:	Sensor

DIGITAL.IN i [TO] IN n/BB n [[AS] INPUT|PULLUP|PULLDOWN]

Command:	DIGITAL.IN i [TO] IN n/BB n [[AS] INPUT|PULLUP|PULLDOWN]
Command Syntax:	**CONNECT DIGITAL.IN i [TO] IN n/OUT n/BB n**
Range:	
Describe:	Connects a generic digital object to a specified pin or port. The connected pin is configured either as a digital output signal, default LOW, or a digital input signal, default INPUT with no pullup or pulldown enabled. The index number can refer to either an input or output. The index is shared by both items since a **DIGITAL** signal can be either an input or output. **CONNECT DIGITAL.IN 1 [TO] IN 1**
Result:	Connect pin to digital object default input state, default **INPUT**.
Type or Addressable Component:	Control/Sensor

SWITCH i [TO] IN n/BB n

Command:	**SWITCH i [TO] IN n/BB n**
Command Syntax:	**CONNECT SWITCH i [TO] IN n/BB n**
Range:	
Describe:	Connect an external switch to a digital input pin. The button task will monitor the state of the switch allowing reporting for the switch on, not on, and was on since last checked. The connected pin is set to a digital input state with its internal pulldown enabled. The other side of the switch is connected to a power supply (3.3v) pin (or 5v supply if using IN3 port). Switches share number space with Buttons. **CONNECT SWITCH 1 [TO] IN 1** **CONNECT SWITCH 2 [TO] BB 5**
Result:	Connect a switch object (similar to button, but connected to **Vcc** instead of **Gnd** when enabled.)
Type or Addressable Component:	Sensor

BUTTON i [TO] IN n/BB n

Command:	**BUTTON i [TO] IN n/BB n**
Command Syntax:	**CONNECT BUTTON i [TO] IN n/BB n**
Range:	
Describe:	Connect an external button to a digital input pin. The button task will monitor the state of the button allowing reporting for the button pressed, not pressed, and was pressed since last checked. The connected pin is set to a digital input state with its internal pullup enabled. The other side of the button is connected to a ground pin. Buttons share number space with Switches. **CONNECT BUTTON i [TO] IN1**
Result:	Digital button/switch/etc.
Type or Addressable Component:	Sensor

MOTION i [TO] IN n/BB n

Command:	MOTION i [TO] IN n/BB n
Command Syntax:	**CONNECT MOTION i [TO] IN n/BB n**
Range:	
Describe:	Connects a digital PIR (passive infrared) motion detection sensor to a digital input pin. This sensor is monitored the same as button objects for a three-state result. **CONNECT MOTION 1i [TO] IN 1**
Result:	Passive **I/R** motion detectors.
Type or Addressable Component:	Sensor

POTENTIOMETER i [TO] IN n/BB n

Command:	POTENTIOMETER i [TO] IN n/BB n
Command Syntax:	**CONNECT POTENTIOMETER i [TO] IN n/BB n**
Range:	
Describe:	Connect an external slide or rotary potentiometer to an analog input pin. **CONNECT POTENTIOMETER 1i [TO] IN 2** **CONNECT POTENTIOMETER 1 [TO] BB 2**
Result:	Rotary- potentiometer sensors.
Type or Addressable Component:	Sensor

THERMISTOR i [TO] IN n/BB n

Command:	THERMISTOR i [TO] IN n/BB n
Command Syntax:	**CONNECT THERMISTOR i [TO] IN n/BB n**
Range:	
Describe:	Connects a PTC thermistor to the system using a single analog input pin. The thermistor sensor uses the following values in the Steinhart-Hart model to convert the reading into a temperature. **Description** — **Value** C1 — 1.33342e-7 C2 — 2.22468e-4 C3 — 1.02119e-3 R1 – reference resistance — 15000.0 ohms **CONNECT THERMISTOR i [TO] IN 1** **CONNECT THERMISTOR i [TO] BB 5**
Result:	Analog thermistor.
Type or Addressable Component:	Sensor

RGB

Command:	CONNECT RGB
Command Syntax:	**CONNECT RGB**
Range	n/a
Describe:	This command configures the sketch to use the TI-RGB Array. The array needs to be pre-connected through the BB port. An incorrect connection will result in an error indication.
Result:	The RGB array is now available for use in the program.
Type or Addressable Component:	Sensor TI-RGB Array Data Sheet

LOUDNESS i [TO] IN n

Command:	LOUDNESS i [TO] IN n
Command Syntax:	**CONNECT LOUDNESS i [TO] IN n**
Range:	
Describe:	The **LOUDNESS** object measure sound intensity (loudness). **CONNECT LOUDNESS i1 [TO] IN2**
Result:	Analog sound level sensors.
Type or Addressable Component:	Sensor

BBPORT

Command:	CONNECT BBPORT
Command Syntax:	**CONNECT BBPORT [MASK value]**
Range	
Describe:	When the optional **MASK** is not specified, this command connects all 10 BB pins to the **BBPORT** object as digital I/O pins. The optional **MASK** parameter may be used to selectively connect specific pins. The mask value may be specified in decimal, binary, or hexadecimal format. For example, 1023 or 0X3FF selects all 10 pins and is the default internal mask value used by the **BBPORT** object if a **MASK** is not specified. **Another example:** If only pins BB1 and BB2 are going to be used, a mask value of 3 or 0x03 will select on the two pins.
Result:	If not **MASK** is specified, the program can read/write to all pins of **BBPORT**. If a **MASK** is specified, the program can write to the specified pins.
Type or Addressable Component:	Sensor

BRIGHTNESS

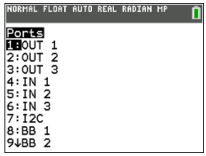
Command:	BRIGHTNESS
Command Syntax:	**CONNECT BRIGHTNESS**
Range:	
Describe:	This command is not needed for typical use since the on-board BRIGHTNESS sensor is automatically connected. (Re-)connect the internal analog ambient light sensor. No pin or port name is used with this internal object.
Result:	Connects on-board light sensor to known analog input pin.
Type or Addressable Component:	Sensor

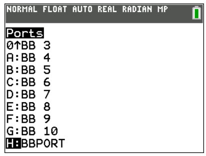
Ports

Settings menu contains operations to set the state of digital and analog pin operations such as the **LED** in the TI-Innovator™ Hub or a connected servo motor movement to states such as ON, OFF, CW (clockwise), and CCW (counterclockwise).

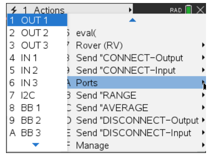
– 1: OUT 1
– 2: OUT 2
– 3: OUT 3
– 4: IN 1
– 5: IN 2
– 6: IN 3
– 7: I2C
– 8: BB 1
– 9: BB 2
– 0: BB 3
– A: BB 4
– B: BB 5
– C: BB 6
– D: BB 7
– E: BB 8
– F: BB 9
– G: BB 10
– H: BBPORT

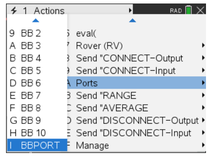
CE Calculators





TI-Nspire™ CX





See also: Breadboard Components and Usable Pins

RANGE

The **RANGE** command is used with several analog input sensors to re-map the internal ADC (Analog to Digital Converter) range of 0 to 16383 (14-bit ADC values) to a floating point range specified as the parameters to this command, along with the sensor to which the range is applied. The format for setting the range of a sensor is **RANGE sensor [i] minimum maximum**. To remove/reset to default the range from a given sensor, set the minimum and maximum value to zero. The minimum value must be less than the maximum value when setting a valid range.

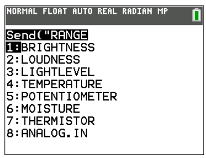
A sensors current range, if present, can be obtained by **READ sensor [i] RANGE**. A two-element list of numbers in the form *{ minimum, maximum }* will be returned.

Note: If no range has been applied to the sensor, an error will be returned if an attempt to read the sensor range is performed .

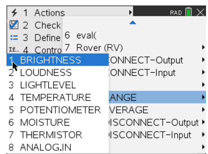
An individual sensors averaging value may be obtained by **READ sensor [i] RANGE**.

RANGE 'something' (for analog devices, maps ADC range from 0 to 16383 to the range specified, min < max, min, max any values.)

CE Calculators



TI-Nspire™ CX



BRIGHTNESS minimum maximum

Command:	**BRIGHTNESS minimum maximum** **Advanced user**
Command Syntax:	**RANGE BRIGHTNESS minimum maximum**
Range:	
Describe:	Changes/Sets the mapping of ADC input values from the ADC 0-16383 range to a user-selected range. The resulting sensor reading is mapped to this and a floating point result is returned. By default, the on-board BRIGHTNESS sensor is ranged to a 0-100 range. **RANGE BRIGHTNESS minimum maximum**
Result:	Set mapping for on-board brightness/light sensor.
Type or	Sensor

Command:	BRIGHTNESS minimum maximum **Advanced user**
Addressable Component:	

LOUDNESS i minimum maximum

Command:	LOUDNESS i minimum maximum **Advanced user**
Command Syntax:	**RANGE LOUDNESS i minimum maximum**
Range:	
Describe:	Changes/Sets the mapping of ADC input values from the ADC 0-16383 range to a user-selected range. The resulting sensor reading is mapped to this and a floating point result is returned. **RANGE LOUDNESS i minimum maximum**
Result:	Set mapping for sound-level analog sensor.
Type or Addressable Component:	Sensor

LIGHTLEVEL i minimum maximum

Command:	LIGHTLEVEL i minimum maximum **Advanced user**
Command Syntax:	**RANGE LIGHTLEVEL i minimum maximum**
Range:	An integer value between 0 and 16383 (14 bit resolution)
Describe:	Changes/Sets the mapping of ADC input values from the ADC 0-16383 range to a user-selected range. The resulting sensor reading is mapped to this and a floating point result is returned. **RANGE LIGHTLEVEL i minimum maximum**

Command:	LIGHTLEVEL i minimum maximum **Advanced user**
Result:	Set mapping for off-board light sensor (analog).
Type or Addressable Component:	Sensor

TEMPERATURE i minimum maximum

Command:	TEMPERATURE i minimum maximum **Advanced user**
Command Syntax:	**RANGE TEMPERATURE i minimum maximum**
Range:	
Describe:	. **RANGE TEMPERATURE i minimum maximum**
Result:	Set mapping for soil moisture analog sensor.
Type or Addressable Component:	Sensor

POTENTIOMETER i minimum maximum

Command:	POTENTIOMETER i minimum maximum **Advanced user**
Command Syntax:	**RANGE POTENTIOMETER i minimum maximum**
Range:	
Describe:	Changes/Sets the mapping of ADC input values from the ADC 0-16383 range to a user-selected range. The resulting sensor reading is mapped to this and a floating point result is returned. **RANGE**

Command:	POTENTIOMETER i minimum maximum **Advanced user**
	POTENTIOMETER i minimum maximum
Result:	Set mapping for rotary/linear potentiometers.
Type or Addressable Component:	Sensor

MOISTURE i minimum maximum

Command:	MOISTURE i minimum maximum **Advanced user**
Command Syntax:	**RANGE MOISTURE i minimum maximum**
Range:	An integer value between 0 and 16383 (14 bit resolution)
Describe:	Changes/Sets the mapping of ADC input values from the ADC 0-16383 range to a user-selected range. The resulting sensor reading is mapped to this and a floating point result is returned. **RANGE MOISTURE i minimum maximum**
Result:	Set mapping for soil moisture analog sensor.
Type or Addressable Component:	Sensor

THERMISTOR i minimum maximum

Command:	THERMISTOR i minimum maximum **Advanced user**
Command Syntax:	**RANGE THERMISTOR i minimum maximum**
Range:	

Command:	THERMISTOR i minimum maximum **Advanced user**
Describe:	. **RANGE THERMISTOR i minimum maximum**
Result:	Set mapping for xxxxxxxxxx.
Type or Addressable Component:	Sensor

ANALOG.IN i minimum maximum

Command:	ANALOG.IN i minimum maximum **Advanced user**
Command Syntax:	**RANGE ANALOG.IN i minimum maximum**
Range:	
Describe:	Changes/Sets the mapping of ADC input values from the ADC 0-16383 range to a user-selected range. The resulting sensor reading is mapped to this and a floating point result is returned. **RANGE ANALOG.IN i minimum maximum**
Result:	Set mapping for generic analog input objects.
Type or Addressable Component:	Sensor

AVERAGE

The **AVERAGE** command is used to set the number of ADC (Analog to Digital converter) samples taken to represent a single analog sensor reading. By default, the TI-Innovator™ Hub sets a global value of three (3) readings to be taken for a sensor measurement. This is done to reduce variation due to noise etc. This default is adjustable between 1 and 25 by the **SET AVERAGING n** command. The current default can be obtained by the **READ AVERAGING** command.

For individual sensors, the default can be changed after the **CONNECT** operation by using the **AVERAGE** command. The format is **AVERAGE sensor [i] value** where sensor is a sensor from the table below, **[i]** is the index, if needed to identify the specific sensor, and value is a number from 1 to 25.

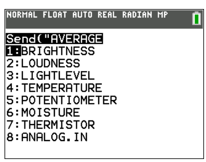
The sensor, when a sample is requested, will take value number of readings, 10 microseconds apart, summing the readings together and averaging them over the number of readings taken.

An individual sensors averaging value may be obtained by **READ sensor [i] AVERAGE**.

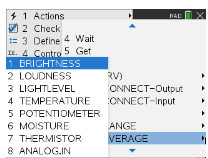
AVERAGE 'something' (for analog devices, sets the individual oversampling value for reading, from 1 to 25)

Command:	**AVERAGE**
Command Syntax:	**AVERAGE**
Describe:	Specifies the number of analog readings to take on a specific sensor to obtain a single reading of that sensor. Valid values are from 1 to 25 readings, taken 10 microseconds apart and averaged together. Sensors use the system default of 3 readings if not altered by changing the system global setting via a **SET AVERAGING** command.
Result:	
Type or Addressable Component:	

CE Calculators



TI-Nspire™ CX



BRIGHTNESS n

Command:	BRIGHTNESS n
Command Syntax:	**AVERAGE BRIGHTNESS n**
Range:	Where **n** ranges from 1 to 25
Describe:	Set the number of readings from the ADC to be used for the on-board light sensor.
Result:	Set oversampling for on-board brightness/light sensor.
Type or Addressable Component:	Sensor

LOUDNESS i n

Command:	LOUDNESS i n
Command Syntax:	**AVERAGE LOUDNESS i n**
Range:	– where **n** ranges from 1 to 25
Describe:	Set the number of readings from the ADC to be used with an external sound loudness sensor.
Result:	Set oversampling for sound-level analog sensor.
Type or Addressable Component:	Sensor

LIGHTLEVEL i n

Command:	LIGHTLEVEL i n
Command Syntax:	**AVERAGE LIGHTLEVEL i n**
Range:	– where **n** ranges from 1 to 25
Describe:	Set the number of readings from the ADC to be used for the

Command:	LIGHTLEVEL i n
	external light sensor connected to an analog input. Does not support I^2C light sensors.
Result:	Set oversampling for off-board light sensor (analog).
Type or Addressable Component:	Sensor

TEMPERATURE i n

Command:	TEMPERATURE i n
Command Syntax:	**AVERAGE TEMPERATURE i n**
Range:	Where **n** ranges from 1 to 25
Describe:	Set the number of readings from the ADC to be used for the external temperature sensor connected to an analog input. Does not support I^2C or digital temperature sensors.
Result:	When using an analog-style thermistor temperature sensor, oversample this many times.
Type or Addressable Component:	Sensor

POTENTIOMETER i n

Command:	POTENTIOMETER i n
Command Syntax:	**AVERAGE POTENTIOMETER i n**
Range:	Where **n** ranges from 1 to 25
Describe:	Set the number of readings from the ADC to be used with an external potentiometer, either a linear or rotary model.
Result:	Set oversampling for rotary/linear potentiometers.
Type or Addressable	Sensor

Command:	POTENTIOMETER i n
Component:	

MOISTURE i n

Command:	MOISTURE i n
Command Syntax:	**AVERAGE MOISTURE i n**
Range:	– where **n** ranges from 1 to 25
Describe:	Set the number of readings from the ADC to be used with an external moisture sensor.
Result:	Set oversampling for soil moisture analog sensor.
Type or Addressable Component:	Sensor

THERMISTOR i n

Command:	THERMISTOR i n
Command Syntax:	**AVERAGE THERMISTOR i n**
Range:	Where **n** ranges from 1 to 25
Describe:	Set the number of readings from the ADC to be used with an external thermistor connected to an analog input.
Result:	Set oversampling for thermistor device analog input.
Type or Addressable Component:	Sensor

ANALOG.IN i n

Command:	**ANALOG.IN i n**
Command Syntax:	**AVERAGE ANALOG.IN i n**
Range:	Where **n** ranges from 1 to 25
Describe:	Set the number of readings from the ADC to be used for the analog sensor attached to this generic analog item.
Result:	Sets oversampling count for generic analog input.
Type or Addressable Component:	Sensor

PERIOD n

Command:	**PERIOD n**
Command Syntax:	**PERIOD n**
Range:	
Describe:	The **AVERAGE** command is somewhat unique for **PERIOD** in that it specifies how many distinct periods are to be measured and averaged together to obtain the desired measurement. Up to 25 samples may be taken to obtain the period measurement for a given pin.
Result:	Set number of samples of frequency to take to be average together to generate period.
Type or Addressable Component:	Sensor

DISCONNECT-Output

DISCONNECT breaks the association between a specified control or sensor and the pin/port it is associated with. If the specified sensor or control is not currently connected to anything, an error is generated.

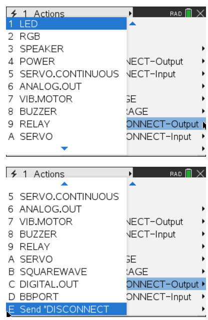
The **DISCONNECT** command does not generate an active response, other than possible error responses. Pins associated with an actively connected sensor, or control, are released from use and, in general, are set to a digital input state with no enabled pullup/pulldown.

DISCONNECT - disconnect something that has been connected, by index if needed.

Command:	**DISCONNECT-Output**
Command Syntax:	**DISCONNECT**
Range:	
Describe:	Removes the association of a sensor or control with a pin, or set of pins, if such association exists. Places the pin(s) back to an **OUTPUT** state.
Result:	.
Type or Addressable Component:	

CE Calculators

TI-Nspire™ CX



LED i

Command:	LED i
Command Syntax:	**DISCONNECT LED i**
Range:	
Describe:	Disconnect an external **LED** object from the system.
Result:	**LED i** is disconnected
Type or Addressable Component:	Control

RGB i

Command:	RGB i
Command Syntax:	**DISCONNECT RGB i**
Range:	
Describe:	Disconnect an external **RGB LED** from the system. These objects use three hardware **PWM** signals to properly operate, so in the initial product release, the on-board **COLOR** object must be disconnected to connect one of these objects.
Result:	Disconnect **RGB** and free up **PWM** outputs for use elsewhere.
Type or Addressable Component:	Control

SPEAKER i

Command:	SPEAKER i
Command Syntax:	**DISCONNECT SPEAKER i**

Command:	**SPEAKER i**
Range:	
Describe:	Disconnect an external speaker from its digital pin.
Result:	Disconnect a speaker from a digital output pin.
Type or Addressable Component:	Control

POWER

Command:	**DISCONNECT POWER i**
Command Syntax:	**DISCONNECT POWER 1**
Range	
Describe:	This command removes the name **POWER** device from the program.
Result:	The named **POWER** device cannot be used in the program after a **DISCONNECT** command.
Type or Addressable Component:	Control

SERVO.CONTINUOUS i

Command:	**SERVO CONTINUOUS i**
Command Syntax:	**DISCONNECT SERVO.CONTINUOUS i**
Code Sample:	
Range:	
Describe:	Disconnect a sweep or continuous **SERVO** motor from the digital pin associated with the motor.

Command:	SERVO CONTINUOUS i
Result:	Servo motor disconnected.
Type or Addressable Component:	Control

ANALOG.OUT i

Command:	ANALOG.OUT i
Command Syntax:	**DISCONNECT ANALOG.OUT i**
Range:	
Describe:	Disconnects the connected generic analog output device specified, freeing a hardware map-able **PWM** if it is in use with the object.
Result:	Disconnect generic analog **PWM** output from pin.
Type or Addressable Component:	Control

VIB.MOTOR

Command:	VIB.MOTOR i [TO] PWM
Command Syntax:	**SET VIB.MOTOR i [TO] PWM**
Range:	PWM from 0 (none) and 255 (full on)
Describe:	Vibration motor control interface.
Result:	Vibrations : intensity is a value from 0 to 255.
Type or Addressable Component:	Control

BUZZER i

Command:	BUZZER i
Command Syntax:	**DISCONNECT BUZZER i**
Range:	
Describe:	Disconnect an active buzzer from the system. Active buzzers play a tone when their signal is set high/on, and stop the tone when the signal is dropped to ground. **DISCONNECT BUZZER i**
Result:	**ACTIVE** buzzers disconnected from a digital pin.
Type or Addressable Component:	Control

RELAY i

Command:	RELAY i
Command Syntax:	**DISCONNECT RELAY i**

Command:	RELAY i
Range:	
Describe:	Disconnect a digital relay interface from the system.
Result:	Relay disconnected.
Type or Addressable Component:	Control

SERVO i

Command:	SERVO i
Command Syntax:	**DISCONNECT SERVO i**
Code Sample:	
Range:	
Describe:	Disconnect a sweep or continuous **SERVO** motor from the digital pin associated with the motor.
Result:	Servo motor disconnected.
Type or Addressable Component:	Control

SQUAREWAVE i

Command:	SQUAREWAVE i
Command Syntax:	**DISCONNECT SQUAREWAVE i**
Range:	
Describe:	Disconnect the software generated squarewave generator from an associated digital output pin. The pin reverts to digital input upon

Command:	**SQUAREWAVE i**
	disconnect.
Result:	Disconnect squarewave function from pin(s), stops squarewave generation.
Type or Addressable Component:	Control

DIGITAL.OUT i

Command:	**DIGITAL.OUT i**
Command Syntax:	**DISCONNECT DIGITAL.OUT i**
Range:	
Describe:	Disconnect a generic **DIGITAL** object. The associated pin is reverted to a digital **INPUT** pin with no enabled pullup or pulldown. The **DIGITAL** object number can be used to refer the same pin in either input, or output form...
Result:	Disconnect digital input object.
Type or Addressable Component:	Control/Sensor

BBPORT

Command:	DISCONNECT BBPORT
Command Syntax:	**DISCONNECT BBPORT**
Range	
Describe:	Disconnects all connected **BBPORT** object pins, and resets those pins to the default **INPUT** state, and unused/available for other use.
Result:	The **BBPORT** object is no longer available for use in the program.
Type or Addressable Component:	Control/Sensor

LIGHT

Command:	LIGHT
Command Syntax:	**DISCONNECT LIGHT**
Range:	
Describe:	Disconnect the on-board **RED LED** used for direct program control from the system.
Result:	On-board **LED** disconnected
Type or Addressable Component:	Control

COLOR

Command:	COLOR
Command Syntax:	**DISCONNECT COLOR**
Range:	
Describe:	Disconnects the on-board **RGB LED** item from use. This action (in the

Command:	COLOR
	initial release of the TI-Innovator™) frees three (3) hardware map-able **PWM** signals for use on other pins..
Result:	Disconnect on-board **RGB LED**.
Type or Addressable Component:	Control

SOUND

Command:	SOUND
Command Syntax:	**DISCONNECT SOUND**
Range:	
Describe:	Disconnect the on-board speaker from its digital pin.
Result:	Disconnects on-board speaker.
Type or Addressable Component:	Control

DCMOTOR i

Command:	DCMOTOR i
Command Syntax:	**DISCONNECT DCMOTOR i**
Range:	
Describe:	Disconnects a **DCMOTOR** object from the system. **DCMOTOR**, **ANALOG.OUT**, and **SQUAREWAVE** all share the same number space of items. **DCMOTOR** requires external power.
Result:	Disconnect **DCMOTOR** from pin.
Type or	Control

Command:	DCMOTOR i
Addressable Component:	

DISCONNECT-Input

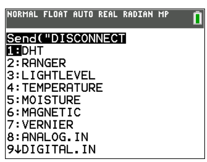
DISCONNECT breaks the association between a specified control or sensor and the pin/port it is associated with. If the specified sensor or control is not currently connected to anything, an error is generated.

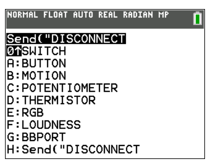
The **DISCONNECT** command does not generate an active response, other than possible error responses. Pins associated with an actively connected sensor, or control, are released from use and, in general, are set to a digital input state with no enabled pullup/pulldown.

DISCONNECT - disconnect something that has been connected, by index if needed.

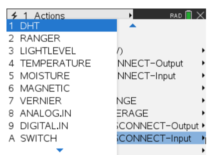
Command:	**DISCONNECT-Input...**
Command Syntax:	**DISCONNECT**
Range:	
Describe:	Removes the association of a sensor or control with a pin, or set of pins, if such association exists. Places the pin(s) back to an **INPUT** state.
Result:	.
Type or Addressable Component:	

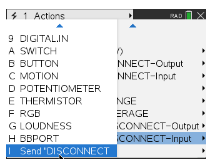
CE Calculators





TI-Nspire™ CX





DHT i

Command:	**DHT i**
Command	**DISCONNECT DHT i**

Command:	DHT i
Syntax:	
Range:	Temperature reading default is in Celsius Humidity reading from 0 to 100 %
Describe:	Disconnects the specified digital humidity **DHT** and temperature sensor from the system. This also removes that object from the period scan list of style sensors in the DHT task.
Result:	Digital humidity/temperature sensor(s) disconnected.
Type or Addressable Component:	Sensor

RANGER i

Command:	RANGER i
Command Syntax:	**DISCONNECT RANGER i**
Range:	
Describe:	Disconnect a digital ultrasonic ranging sensor from the two digital pins it uses.
Result:	Ultrasonic ranging sensor disconnected.
Type or Addressable Component:	Sensor

LIGHTLEVEL i

Command:	LIGHTLEVEL i
Command Syntax:	**DISCONNECT LIGHTLEVEL i**
Range:	
Describe:	Disconnect an external light sensor.

Command:	LIGHTLEVEL i
Result:	Light sensor disconnected.
Type or Addressable Component:	Sensor

TEMPERATURE i

Command:	TEMPERATURE i
Command Syntax:	**DISCONNECT TEMPERATURE i**
Range:	Temperature reading default is in Celsius. The range depends on the specific temperature sensor being used. Humidity reading from 0 to 100 %
Describe:	Disconnect a connected temperature sensor from the system. **TEMPERATURE** sensors can be either analog (thermistor-style). Disconnecting from the analog or digital reverts the associated pins to INPUT.
Result:	Disconnect temperature sensor.
Type or Addressable Component:	Sensor

MOISTURE i

Command:	MOISTURE i
Command Syntax:	**DISCONNECT MOISTURE i**
Range:	
Describe:	Disconnect an analog moisture sensor.
Result:	Disconnect analog moisture sensors
Type or Addressable Component:	Sensor

MAGNETIC

Command:	**DISCONNECT MAGNETIC i**
Command Syntax:	**DISCONNECT MAGNETIC 1**
Range	
Describe:	The **MAGNETIC** sensor is used to detect the presence of a magnetic field. It uses the Hall effect. It is also known as a Hall effect sensor. The **DISCONNECT** command removes the sensor from the program.
Result:	The name "**MAGNETIC 1**" is now disconnected from the sensor. It cannot be used in the program after a **DISCONNECT** command.
Type or Addressable Component:	Sensor

VERNIER

Command:	**DISCONNECT VERNIER i**
Command Syntax:	**DISCONNECT VERNIER 1**
Range	
Describe:	This command removes the named Vernier device from the program.
Result:	A Vernier analog sensor connected to the TI-Innovator™ Hub through a TI-SensorLink cannot be used in the program after a **DISCONNECT** command.
Type or Addressable Component:	Sensor

ANALOG.IN i

Command:	ANALOG.IN i
Command Syntax:	**DISCONNECT ANALOG.IN i**
Range:	
Describe:	Disconnects the connected generic analog input device specified.
Result:	Disconnect generic analog input from pin.
Type or Addressable Component:	Sensor

DIGITAL.IN i

Command:	DIGITAL.IN i
Command Syntax:	**DISCONNECT DIGITAL.IN i**
Range:	
Describe:	Disconnect a generic **DIGITAL** object. The associated pin is reverted to a digital **INPUT** pin with no enabled pullup or pulldown. The **DIGITAL** object number can be used to refer the same pin in either input, or output form.
Result:	Disconnect digital input object.
Type or Addressable Component:	Control/Sensor

SWITCH

Command:	SWITCH
Command Syntax:	**DISCONNECT SWITCH i**
Range:	
Describe:	Disconnect a switch from its digital pin. The pin reverts to INPUT

Command:	SWITCH
	state, and the switch is removed from the scanning sequence in the BUTTON task.
Result:	disconnect switch object from pin
Type or Addressable Component:	Sensor

BUTTON i

Command:	BUTTON i
Command Syntax:	**DISCONNECT BUTTON i**
Range:	
Describe:	Disconnects the specified button object from the system and removes it from the list of scanned buttons/switches in the **BUTTON** task.
Result:	Digital button/switch is disconnected.
Type or Addressable Component:	Sensor

MOTION i

Command:	MOTION i
Command Syntax:	**DISCONNECT MOTION i**
Range:	
Describe:	Disconnects a digital **PIR** (passive infrared) **MOTION** detector and removes the object from the scanning list in the **BUTTON** task.
Result:	Disconnect passive **I/R** motion detectors
Type or	Sensor

Command:	MOTION i
Addressable Component:	

POTENTIOMETER i

Command:	POTENTIOMETER i
Command Syntax:	**DISCONNECT POTENTIOMETER i**
Range:	
Describe:	Disconnect an analog variable resistor (**POTENTIOMETER**) from the system
Result:	Disconnect a rotary/linear potentiometer sensors
Type or Addressable Component:	Sensor

THERMISTOR i

Command:	THERMISTOR i
Command Syntax:	**DISCONNECT THERMISTOR i**
Range:	
Describe:	Disconnect an analog thermistor sensor from the associated pin.
Result:	disconnect analog thermistor
Type or Addressable Component:	Sensor

RGB

Command:	DISCONNECT RGB
Command Syntax:	**DISCONNECT RGB**
Range	
Describe:	The **DISCONNECT** command removes the TI-RGB Array from the program.
Result:	The TI-RGB Array cannot be used in the program after a **DISCONNECT** command.
Type or Addressable Component:	Sensor

LOUDNESS i

Command:	LOUDNESS i
Command Syntax:	**DISCONNECT LOUDNESS i**
Range:	
Describe:	Disconnect an analog sound intensity (**LOUDNESS**) sensor.
Result:	Analog sound level sensor disconnected
Type or Addressable Component:	Sensor

BBPORT

Command:	**DISCONNECT BBPORT**
Command Syntax:	**DISCONNECT BBPORT**
Range	
Describe:	Disconnects all connected **BBPORT** object pins, and resets those pins to the default **INPUT** state, and unused/available for other use.
Result:	The **BBPORT** object is no longer available for use in the program.
Type or Addressable Component:	Control/Sensor

BRIGHTNESS

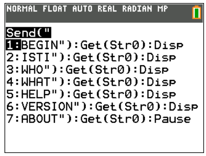
Command:	**BRIGHTNESS**
Command Syntax:	**DISCONNECT BRIGHTNESS**
Range:	
Describe:	Disconnects the internal connection to the on-board **BRIGHTNESS** (light sensor) object.
Result:	Disconnect on-board **LIGHT** sensor.
Type or Addressable Component:	Sensor

MANAGE

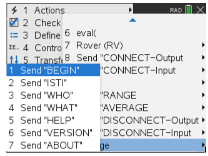
The **Manage** menu pastes a **Send(** command with the following management items.

Str0 is displayed on Home Screen with information if requested in the command.

CE Calculators



TI-Nspire™ CX



BEGIN

The **BEGIN** command disconnects all connected sensors and controls, re-initializes all sensor/control memory within the sketch, and resets the sensor average default value, error formatting, and flow control defaults. Additionally, all **IN***n* port pins, and the breadboard connector (**BB***n*) pins are set to the **INPUT** pin mode. All **OUT***n* port pins are set to the **INPUT** state, and allowed to float, including **OUT3** which will read as high due to a pullup resistor from the 5V supply on this pin.

When the entire process completes, a response of **READY** is sent to the host system. This response must be waited for by the host before any further operations are performed. Additional commands may be in the command queue to be executed, but will not be acted upon until this command completes.

BEGIN

Command:	**BEGIN**
Command Syntax:	**SEND("BEGIN"**
Describe:	Disassociates sensors from ports or pins, and resets all settings back to defaults. Disconnects any connected sensor objects and restores system to state as if **RESET** button pressed.
Result:	Responds with a "**READY**" when completed.
Type or Addressable Component:	Not Applicable

Note: The [:] is used to sequence command lines on one command line. The **Manage…** menu pastes a convenient set of commands to then display the information in **Str0** on the home screen.

ISTI

The **ISTI** command is used to synchronize communications with the sketch. The response to this command must be **TISTEM**. Responses may have a leading ***NUL*** (0) character on initial power-on of the Innovator hub. All responses from the Innovator hub will be followed with a *CR/LF* pair that may or may not be stripped by software layers in the host system prior to the response being received by the application layer on the host system.

ISTI

Command:	**ISTI**
Command Syntax:	**ISTI**
Describe:	Send "ISTI", and get response "TISTEM".
Result:	Handshake command used to determine presence of a supported “sketch” on the TI-Innovaotr™ Hub.
Type or Addressable Component:	

WHO

WHO is an identification command (similar to the **ISTI** handshake command below) that can be used to determine what product is present and running the sketch.

The correct response to **WHO** is "**TI INNOVATOR ON MSP432**" when this command is sent to the TI-Innovator Hub.

WHO

Command:	**WHO**
Command Syntax:	**WHO**
Describe:	Identification command to determine what product is running the sketch. Send ("WHO") Get Str0 Disp Str0
Result:	Identify the product - TI INNOVATOR ON MSP432.

Command:	WHO
Type or Addressable Component:	

WHAT

The **WHAT** command is an identification command. The response to **WHAT** for TI-Innovator is "**TI INNOVATOR HUB**".

WHAT

Command:	WHAT
Command Syntax:	**WHAT**
Describe:	Product name query. Identify the product - "**TI INNOVATOR HUB**" Send ("WHAT") Get Str0 Disp Str0
Result:	Identify the product.
Type or Addressable Component:	

HELP

HELP is used to obtain quick information about each of these commands. The **HELP command-name** is sent, and generates a string response with a one-line description of the given command.

HELP

Command:	HELP
Command Syntax:	**HELP**
Describe:	Provides per command quick help information. i.e. HELP SET, etc.
Result:	
Type or Addressable Component:	

VERSION

The **VERSION** command has a response that represents the current version of the sketch running on the TI-Innovator™ Hub.

The version will be of the *major.minor.patch.build* form in released products; for example, 1.0.0.

VERSION

Command:	**VERSION**
Command Syntax:	**VERSION**
Describe:	Returns version number (and possibly Accurev stream name from which sketch was built).
Result:	Report the version of the sketch in format *major.minor.patch.build*. Send ("VERSION") Get Str0 Disp Str0
Type or Addressable Component:	

ABOUT

The **ABOUT** command response is the product line name along with a copyright date and owner. The current response to this command is "**TI INNOVATOR (C)2015-2016 TEXAS INSTRUMENTS**".

ABOUT

Command:	**ABOUT**
Command Syntax:	**ABOUT**
Describe:	Product name and copyright information returned. Send ("ABOUT") Get Str0 Disp Str0
Result:	Returns copyright string. "**TI INNOVATOR (C)2015-2016 TEXAS INSTRUMENTS**"
Type or Addressable Component:	

Additional Supported Commands

The following sets of supported commands are not found in the Hub Menus.

Additional SET Commands

FORMAT ERROR STRING/NUMBER

Command:	FORMAT ERROR STRING/NUMBER **Advanced user**
Command Syntax:	**SET FORMAT ERROR STRING/NUMBER**
Range:	
Describe:	Used for setting error return format and optional audible tone on error. **SET FORMAT ERROR STRING/NUMBER** – returned error codes in string or numeric format.
Result:	Sets the format for the return of error information (numbers, or strings).
Type or Addressable Component:	Setting

FORMAT ERROR NOTE/QUIET

Command:	FORMAT ERROR NOTE/QUIET **Advanced user**
Command Syntax:	**SET FORMAT ERROR NOTE/QUIET**
Range:	
Describe:	Used for setting error return format and optional audible tone on error.

Command:	FORMAT ERROR NOTE/QUIET **Advanced user**
	SET FORMAT ERROR NOTE/QUIET – error display flash accompanied by speaker sound or no sound.
Result:	Enables tones, or disables tones in addition to the string/number reporting above.
Type or Addressable Component:	Setting

FLOW [TO] ON/OFF

Command:	FLOW [TO] ON/OFF **Advanced user**
Command Syntax:	**SET FLOW [TO] ON/OFF**
Range:	
Describe:	Enables (**ON**) or disables (**OFF**) the software flow control mechanism between the sketch and the communications hardware. **NOTE:** When the **SEGDISP** module is **CONNECT**ed, this setting determines whether or not the display module shows error information (flow control disabled), or command queue depth (flow control enabled).
Result:	Turn on xon/xoff flow control, or turn off (no flow control)
Type or Addressable Component:	Setting

OUT1/2/3 [TO]

Command:	OUT1/2/3 [TO]
Command Syntax:	**OUT1/2/3 [TO] ...** **SET OUTn 0-255** **SET OUTn HIGH/ON** **SET OUTn LOW/OFF**
Range:	Set analog PWM value on **OUT** port(s) of the TI-Innovator™ Hub
Describe:	Direct output of information to a given output port. These are PWM outputs on the TI-Innovator™ Hub. Set analog PWM value on TI-Innovator™ Hub **OUT** port(s). **SET OUTn 0-255** – 0=off, 255=on, anything else is a PWM signal @ 500 Hz with duty cycle high from 1 to 254, where that range provides a percentage of the high-time signal of the waveform. **SET OUTn HIGH/ON** – same as 255 **SET OUTn LOW/OFF** – same as 0
Result:	Set analog **PWM** value on **OUT** port(s) of the TI-Innovator™ Hub
Type or Addressable Component:	Port

BUZZER i

Command:	**BUZZER i**
Command Syntax:	**READ BUZZER i**
Range:	
Describe:	Returns the current state of the active buzzer specified; 0 = *silent*, 1 = *playing tone*.
Result:	Returns state of active buzzer, 0=silent, 1=on
Type or Addressable Component:	Control

COLOR

Command:	**COLOR**
Command Syntax:	**READ COLOR**
Range:	
Describe:	Read the current output state of the on-board **COLOR RGB LED** with sub-components **.RED**, **.GREEN**, **.BLUE**. When reading the entire item, a list of three values is returned, with values between 0 and 255 where 0=off, 255=full on, and values in between indicate **PWM** levels. **READ COLOR** – returns list of 3 values representing { red, green, blue } PWM levels **READ COLOR.RED** **READ COLOR.GREEN** **READ COLOR.BLUE** See Also: **RGB i**

Command:	COLOR
Result:	Returns list of 3 values representing { red, green, blue } **PWM** levels. Returns **RED/GREEN/BLUE** values for on-board **RGB** (color) **LED**.
Type or Addressable Component:	Control

COLOR.RED

Command:	COLOR RED
Command Syntax:	**READ COLOR.RED**
Range:	
Describe:	Read the current output state of the on-board **COLOR RGB LED** with sub-components **.RED**, **.GREEN**, **.BLUE**. When reading the entire item, a list of three values is returned, with values between 0 and 255 where 0=off, 255=full on, and values in between indicate **PWM** levels. **READ COLOR.RED**
Result:	Returns values representing {red} **PWM** levels. Returns **RED** values for on-board **RGB** (color) **LED**.
Type or Addressable Component:	Control

COLOR.GREEN

Command:	COLOR GREEN
Command Syntax:	**READ COLOR.GREEN**

Command:	COLOR GREEN
Range:	
Describe:	Read the current output state of the on-board **COLOR RGB LED** with sub-components **.RED**, **.GREEN**, **.BLUE**. When reading the entire item, a list of three values is returned, with values between 0 and 255 where 0=off, 255=full on, and values in between indicate **PWM** levels. **READ COLOR.GREEN**
Result:	Returns list of 3 values representing { red, green, blue } **PWM** levels. Returns **RED/GREEN/BLUE** values for on-board **RGB** (color) **LED**.
Type or Addressable Component:	Control

COLOR.BLUE

Command:	COLOR BLUE
Command Syntax:	**READ COLOR.BLUE**
Range:	
Describe:	Read the current output state of the on-board **COLOR RGB LED** with sub-components **.RED**, **.GREEN**, **.BLUE**. When reading the entire item, a list of three values is returned, with values between 0 and 255 where 0=off, 255=full on, and values in between indicate **PWM** levels. **READ COLOR.BLUE**
Result:	Returns list of 3 values representing { red, green, blue } **PWM** levels. Returns **RED/GREEN/BLUE** values for on-board **RGB** (color) **LED**.
Type or Addressable Component:	Control

DCMOTOR i

Command:	DCMOTOR i
Command Syntax:	**READ DCMOTOR i**
Range:	
Describe:	Motor that converts direct current electrical power into mechanical power.
Result:	Returns whether dcmotor is running (1) or stopped (0).
Type or Addressable Component:	Control

DIGITAL.OUT i

Command:	DIGITAL.OUT i
Command Syntax:	**READ DIGITAL.OUT i**
Range:	
Describe:	Returns the current state of the digital pin connected to the DIGITAL object, or the cached state of the digital output value last SET to the object.
Result:	Return 0 (output low), 1 (output high).
Type or Addressable Component:	Control/Sensor

FORMAT

Command:	FORMAT **Advanced user**
Command Syntax:	**READ FORMAT**
Range:	
Describe:	Return the current formatting flags for error reporting. The value returned is a byte value indicating various flags. Masking with values indicates what error reporting options are active. 1 = ERROR strings reported 2 = ERROR numbers reported +4 = ERROR TONE enabled, if not set, errors are reported silently.
Result:	Read error format (1=strings, 2=numbers, +4 to either: tones enabled).
Type or Addressable Component:	Setting

FLOW

Command:	FLOW **Advanced user**
Command Syntax:	**READ FLOW**
Range:	
Describe:	Returns the current flow control setting; 0=*disabled*, 1=*enabled*.
Result:	Read current flow control, 0 = none, 1 = xon/xoff
Type or Addressable Component:	Setting

IN1/IN2/IN3

Command:	**IN1/IN2/IN3**
Command Syntax:	**READ IN1** **READ IN2** **READ IN3**
Range:	
Describe:	Read the value present on the indicated port, and return that value to the host.
Result:	Read value of analog port on TI STEM board
Type or Addressable Component:	Port

LAST ERROR

Command:	**LAST ERROR**
Command Syntax:	**READ LAST ERROR**
Range:	
Describe:	Returns the last reported error from the last operation. Depending on the **FORMAT ERROR** setting, the response may be a **STRING** or a **NUMBER**.
Result:	Return last encountered error, resets automatically to 0, no error.
Type or Addressable Component:	Setting

LED i

Command:	**LED i**
Command Syntax:	**READ LED i**
Range:	
Describe:	Read the current state of the specified **LED**. If the **LED** is digital, a 0 or 1 is returned indicating the **LED** is off or on. If the **LED** is connected to a **PWM** output, a value from 0 to 255 will be returned, indicating the current **PWM** level where 0 is off, 255 is full on, and values in between indicate the current **PWM** setting.
Result:	Get state of **LED**, 0 or 1 if digital, 0-255 if **PWM** on analog.
Type or Addressable Component:	Control

LIGHT

Command:	**LIGHT**
Command Syntax:	**READ LIGHT**
Range:	
Describe:	Returns the state of the on-board **RED LED** (digital only). A value of 0 is off, and 1 is on.
Result:	Get current state of on-board red **LED** (0=off, 1=on).
Type or Addressable Component:	Control

OUT1/2/3

Command:	OUT1/2/3
Command Syntax:	**READ OUT1** **READ OUT2** **READ OUT3**
Range:	
Describe:	Read value of current port as input (may be a digital read since these do not support analog-input. **READ OUT1/OUT2/OUT3**
Result:	Read value of analog port on **TI STEM** board.
Type or Addressable Component:	Port

PWR

Command:	PWR
Command Syntax:	**READ PWR**
Range:	
Describe:	Returns the current state of presence of external power connected to the **PWR** port. The **PWR** port is read, and a status value of 0 (not present) or 1 (present) is returned, based on whether or not external power is available. **READ PWR**
Result:	Returns state of external power presence on **PWR** port (0=not present, 1=ext pwr present).
Type or Addressable Component:	Status

RELAY i

Command:	RELAY i
Command Syntax:	**READ RELAY i**
Range:	
Describe:	Return the current state of the specified relay. 0 = OFF, 1 = ON.
Result:	Read state of relay - 0=not active 1=active.
Type or Addressable Component:	Control

RESOLUTION

Command:	RESOLUTION
Command Syntax:	**READ RESOLUTION**
Range:	
Describe:	Returns the bit resolution used by the system for ADC readings.
Result:	Returns ADC resolution in use, in bits (default is 14).
Type or Addressable Component:	Setting

RGB i

Command:	RGB i
Command	**READ RGB i**

Command:	RGB i
Syntax:	
Range:	
Describe:	Same as the **COLOR** object referenced above, and has sub-objects named **RED**, **GREEN**, and **BLUE**. This command returns the current **PWM** level that the specified object is using. **READ RGB i** – returns a 3 element list, consisting of the { red, green, blue } color level. **READ RED i** – returns just the current red-component level. **READ GREEN i** **READ BLUE i**
Result:	Get state of **RGB LED**, {r,g,b} list values
Type or Addressable Component:	Control

RED i

Command:	RED i
Command Syntax:	**READ RED i**
Range:	
Describe:	Same as the **COLOR** object referenced above, and has sub-objects named **RED**, **GREEN**, and **BLUE**. This command returns the current PWM level that the specified object is using. **READ RGB i** – returns a 3 element list, consisting of the { red, green, blue } color level. **READ RED i** – returns just the current red-component level.
Result:	Get state of **RGB RED** component.
Type or Addressable Component:	Control

GREEN i

Command:	**GREEN i**
Command Syntax:	**READ GREEN i**
Range:	
Describe:	Same as the **COLOR** object referenced above, and has sub-objects named **RED**, **GREEN**, and **BLUE**. This command returns the current PWM level that the specified object is using. **READ RGB i** – returns a 3 element list, consisting of the { red, green, blue } color level. **READ GREEN i** – returns just the current green-component level.
Result:	Get state of **RGB GREEN** component.
Type or Addressable Component:	Control

BLUE i

Command:	**BLUE i**
Command Syntax:	**READ BLUE i**
Range:	
Describe:	Same as the **COLOR** object referenced above, and has sub-objects named **RED**, **GREEN**, and **BLUE**. This command returns the current PWM level that the specified object is using. **READ RGB i** – returns a 3 element list, consisting of the { red, green, blue } color level. **READ BLUE i** – returns just the current blue-component level
Result:	Get state of **RGB BLUE** component.
Type or Addressable Component:	Control

SERVO i

Command:	SERVO i
Command Syntax:	**READ SERVO i**
Range:	
Describe:	Returns the current position of a sweep servo in the range -90 to 90, OR the current speed of rotation of a continuous servo motor. Additionally, the current "calibration" setting for the servo which consists of a 2-element list representing the lower and upper microsecond pulse widths corresponding to the sweep/rotation ranges may be read. **READ SERVO i** – get current sweep position or rotation speed/direction. **READ SERVO i CALIBRATION** – get current microsecond range for sweep or rotation.
Result:	Return current servo position in degrees from -90 to +90.
Type or Addressable Component:	Control

SERVO i CALIBRATION

Command:	SERVO i CALIBRATION Advanced user
Command Syntax:	**READ SERVO i CALIBRATION**
Range:	
Describe:	Returns the current position of a sweep servo in the range -90 to 90, OR the current speed of rotation of a continuous servo motor. Additionally, the current "calibration" setting for the servo which consists of a 2-element list representing the lower and upper microsecond pulse widths corresponding to the sweep/rotation ranges may be read. **READ SERVO i CALIBRATION** – get current microsecond range for sweep or rotation.

Command:	SERVO i CALIBRATION Advanced user
Result:	Return current servo position in degrees from -90 to +90.
Type or Addressable Component:	Control

SOUND

Command:	SOUND
Command Syntax:	**READ SOUND**
Range:	
Describe:	Returns a value indicating whether sound is currently being played (1) or not (0) through the on-board speaker.
Result:	Return whether on-board speaker is playing a tone (1) or is silent(0).
Type or Addressable Component:	Control

SPEAKER i

Command:	SPEAKER i
Command Syntax:	**READ SPEAKER i**
Range:	
Describe:	Returns a value indicating whether sound is currently being played (1) or not (0) through an external speaker.

Command:	SPEAKER i
Result:	Return whether speaker is playing a tone (1) or silent (0).
Type or Addressable Component:	Control

SQUAREWAVE i

Command:	SQUAREWAVE i
Command Syntax:	**READ SQUAREWAVE i**
Range:	
Describe:	Returns a 0 the current squarewave object is not active. A value of 1 is returned if the object is actively generating an output.
Result:	Returns whether squarewave is active (1) or not active (0).
Type or Addressable Component:	Control

PERIOD n

Command:	PERIOD n
Command Syntax:	**PERIOD n**
Range:	
Describe:	The **AVERAGE** command is somewhat unique for **PERIOD** in that it specifies how many distinct periods are to be measured and averaged together to obtain the desired measurement. Up to 25 samples may be taken to obtain the period measurement for a given pin.
Result:	Set number of samples of frequency to take to be average together to generate period.
Type or Addressable Component:	Sensor

CALIBRATE

CALIBRATE is used to set various sensor and control values that do not otherwise fit within a means of setting any other way. For thermistors and temperature sensors that use an analog input port, it can be used to adjust the coefficients of the Steinhart-Hart equation used to map thermistor readings to temperature values. For servo motors, it is used to adjust the PWM pulse width within the range for a servo motor, where the zero position is set at 1500 microseconds. It is also used to set the calibration frequency for the DDS signal generator module (default is 24MHz).

For sensors supporting calibration, the value(s) may be obtained by **READ sensor [i] CALIBRATION**.

SERVO i / SERVO.CONTINUOUS i

Command:	**SERVO i /SERVO.CONTINUOUS i minimum maximum** **Advanced user**
Command Syntax:	**CALIBRATE SERVO i minimum maximum**
Code Sample:	
Range:	
Describe:	Servos operate by using pulse modulation where the high pulse width determines both direction of servo operation and possibly the speed of operation. The time between pulses is generally 20 milliseconds and is not adjustable by this command. The pulse width generally varies around a mid-point of 1.5 milliseconds (1500 microseconds). Pulse widths less than 1.5 milliseconds cause servo operation in one direction, while pulse widths greater than 1.5 milliseconds cause operation in the opposite direction. The **CALIBRATE** command for **SERVO** allows programmable changes to the minimum and maximum pulse widths. Parameters are pulse width times in microseconds. Current defaults are minimum 600 and maximum 2400 microseconds.

Command:	SERVO i /SERVO.CONTINUOUS i minimum maximum Advanced user
Result:	Set minimum and maximum pulse width for servo motor, values in microseconds, default 600 and 2400.
Type or Addressable Component:	Control

TEMPERATURE i C1 C2 C3 R1

Command:	TEMPERATURE i C1 C2 C3 R1 Advanced user
Command Syntax:	**CALIBRATE TEMPERATURE i C1 C2 C3 R1**
Range:	
Describe:	The **CALIBRATE** command for analog temperature sensors allows changing the default Steinhart-Hart equation coefficients to match those of the thermistor element in the sensor being used. The default values are: C1: 8.76741e-8 C2: 2.34125e-4 C3: 1.129148e-3 R1: 10000.0 (reference resistor value = 10kΩ)
Result:	When using an analog-style thermistor temperature sensor.
Type or Addressable Component:	Sensor

THERMISTOR i C1 C2 C3 R1

Command:	**THERMISTOR i C1 C2 C3 R1** **Advanced user**
Command Syntax:	**CALIBRATE THERMISTOR i C1 C2 C3 R1**
Range:	
Describe:	The **CALIBRATE** command for analog thermistors allows changing the default Steinhart-Hart equation coefficients to match those of the thermistor element in the sensor being used. The default values are: C1: 1.33342e-7 C2: 2.22468e-4 C3: 1.02119e-3 R1: 15000.0 (reference resistor value = 15kΩ)
Result:	Where c1/c2/c3 are float constants for the Steinhart-Hart equation. ... that models the thermistor, and r is resistance for the reference. ... resistor used to create a voltage divider with the thermistor.
Type or Addressable Component:	Sensor

TI-Innovator™ Hub Data Sheets

The TI-Innovator™ Hub Data Sheets include the following; a product name and number, a brief description, a product image, specifications, on-board components function, and Hub commands with simple code samples.

Topic Links

- TI-Innovator™ Hub Data Sheet
    - TI-Innovator™ Hub Ports and Breadboard Usable Pins
- TI-Innovator™ Hub On-Board Component Data Sheets
    - On-Board RGB LED Data Sheet
    - On-Board Red LED Data Sheet
    - On-Board Speaker Data Sheet
    - On-Board Light Brightness Sensor Data Sheet
    - On-Board - Auxiliary Power Indicator Data Sheet
    - On-Board Green LED - Power Indicator Data Sheet
    - On-Board Red LED - Error Indicator Data Sheet
- USB Mini A to Mini B Cable Data Sheet
- USB Standard A to Mini B Cable Data Sheet
- USB Standard A to Micro B Cable Data Sheet
- TI Wall Charger Data Sheet
- External Battery Data Sheet

TI-Innovator™ Hub Data Sheet

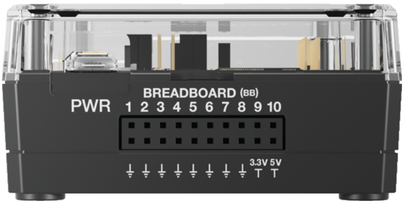
Title	**TI-Innovator™ Hub**
TI Item Name	STEM/BK/B
Quantity	1
Included in	TI-Innovator™ Hub
Description	Use the TI-Innovator™ Hub with your compatible TI graphing calculator or TI-Nspire™ software to control components, read sensors, and create powerful learning experiences.
Category	Hub
Hub Connection	Not Applicable
Assembly Instructions	Not Applicable
Precautions	Do not expose the Hub to temperatures above 140˚F (60˚C). Do not disassemble or mistreat the Hub. Do not chain together multiple Hubs through the I/O ports or the Breadboard Connector. Use only the USB cables provided with the Hub. Use only the TI provided power supplies: • TI Wall Charger included with the TI-Innovator™ Hub • Optional External Battery 4-AA battery holder included in the TI-Innovator™ Breadboard Pack Ensure that the components receiving power from the Hub do not exceed the Hub's 1-amp power limit. Avoid using the Hub to control AC electricity. **See also:** TI-Innovator™ Hub Ports and Breadboard Usable Pins

Title	**TI-Innovator™ Hub**
Specifications	See the TI-Innovator™ Hub specifications section of education.ti.com/go/innovator.

TI-Innovator™ Hub Ports and Breadboard Usable Pins

Breadboard Connector Characteristics

Different pins on the breadboard connector have different capabilities.



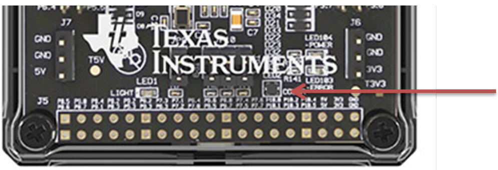
Pin	Digital I/O	Pulse Width Modulation (PWM)	ANALOG IN
BB1	Y		
BB2	Y		
BB3	Y		
BB4	Y	Y	
BB5	Y		Y
BB6	Y		Y
BB7	Y		Y
BB8	Y	Y	
BB9	Y	Y	
BB10	Y	Y	

TI-Innovator™ Hub On-Board Component Data Sheets

Topic Links

- On-Board RGB LED Data Sheet
- On-Board Red LED Data Sheet
- On-Board Speaker Data Sheet
- On-Board Light Brightness Sensor Data Sheet
- On-Board - Auxiliary Power Indicator Data Sheet
- On-Board Green LED - Power Indicator Data Sheet
- On-Board Red LED - Error Indicator Data Sheet

On-Board RGB LED Data Sheet



On-Board RGB LED (LED2)

Title	On-Board RGB LED
TI Item Name	Built into the Hub
Quantity	1
Included in	TI-Innovator™ Hub
Description	Built-in light-emitting diode (LED) that is capable of emitting a variety of colors when current passes through it.
Category	LEDs and Displays
Hub Connection	on-board Hub
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

HUB Commands	
Sketch Object	COLOR
Command Syntax	Send("SET COLOR …")

	ON/OFF/0-255 (red element) ON/OFF/0-255 (green element) ON/OFF/0-255 (blue element) [BLINK frequency] (in Hz) [TIME duration] (in seconds)

Code Sample:

Desired Action	Code Sample
Turn ON Red and Green elements of tri-color LED	`Send("SET COLOR ON ON OFF")`
Set Red to full intensity, Green to half intensity, Blue to off	`Send("SET COLOR 255 128 0")`
Set Red to full intensity, Green to half intensity, Blue to off for 10 seconds	`Send("SET COLOR 255 128 0 TIME 10")`
Set Red to full intensity, Green to half intensity, Blue to off and blink them at 2 Hz (2 times a second) for 10 seconds	`Send("SET COLOR 255 128 0 BLINK 2 TIME 10")`
Turn OFF the Red element	`Send("SET COLOR.RED 0")`
Turn ON the Green element at half intensity and blink it at 2 Hz for 10 seconds	`Send("SET COLOR.GREEN 128 BLINK 2 TIME 10")`

On-Board Red LED Data Sheet

On-Board RED LED
(LED1)

Title	On-Board Red LED
TI Item Name	Built into the Hub
Quantity	1
Included in	TI-Innovator™ Hub
Description	Built-in light-emitting diode (LED) that emits a red light when current passes through it.
Category	LEDs and Displays
Hub Connection	on-board Hub
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

HUB Commands	
Sketch Object	LIGHT
Command Syntax	Send("SET LIGHT …") ON/OFF [BLINK frequency] [TIME duration] (in seconds)

Code Sample:

Desired Action	Code Sample
Turn LED ON	`Send("SET LIGHT ON")`
Turn LED OFF	`Send("SET LIGHT OFF")`
Turn LED ON for 10 seconds	`Send("SET LIGHT ON TIME 10"`
Turn LED ON, blink it at 2 Hz for 10 seconds	`Send("SET LIGHT ON BLINK 2 TIME 10")`

See Also: Red LED - Error Indicator

On-Board Speaker Data Sheet

Speaker (at back of Hub) is addressable as "SOUND" in Hub command strings.

Title	**On-Board Speaker**
TI Item Name	Built into the Hub
Quantity	1
Included in	TI-Innovator™ Hub
Description	Built-in speaker located at the back of the Hub. It converts electrical current into sound you can hear.
Category	Sound Output
Hub Connection	on-board Hub
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

HUB Commands	
Sketch Object	SOUND
Command Syntax	Send("SET SOUND ...") Frequency in Hz or Note as C1, CS1, D2, ... [TIME duration in seconds]

Code Sample:

Desired Action	**Code Sample**
Play tone at 261.23 Hz	`Send("SET SOUND 261.23")`
Evaluate the expression 2^8 (= 256) and play that tone	`Send("SET SOUND eval (2^8)")`

Desired Action	Code Sample
Evaluate the expression 2^8 (= 256) and play that tone for .25 seconds	`Send("SET SOUND eval (2^8) TIME .25")`
Evaluate the expression 2^9 (= 512) and play that tone for 0.25 seconds (result of evaluating 1/4)	`Send("SET SOUND eval (2^9) TIME eval(1/4)")`
Turn speaker off	`Send("SET SOUND OFF")`

On-Board Light Brightness Sensor Data Sheet

Light Brightness Sensor

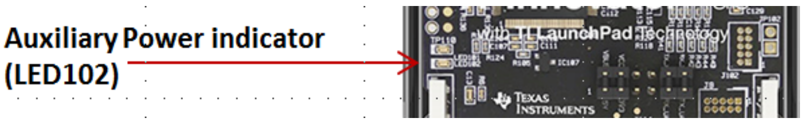
Title	**On-Board Light Brightness Sensor**
TI Item Name	Built into the Hub
Quantity	1
Included in	TI-Innovator™ Hub
Description	Built-in light brightness sensor located at the bottom of the Hub. The sensor detects light intensity.
Category	Environmental Sensors
Hub Connection	on-board Hub
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

HUB Commands	
Sketch Object	BRIGHTNESS
Command Syntax	Send("READ BRIGHTNESS")

Code Sample:

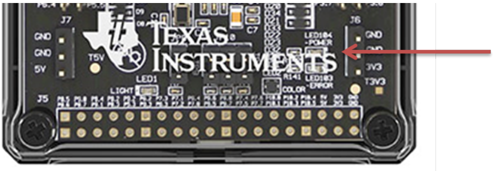
Desired Action	Code Sample
Read the built-in light brightness sensor	`Send("READ BRIGHTNESS")` `Get(B)`

On-Board - Auxiliary Power Indicator Data Sheet



Title	Auxiliary Power Indicator (LED102)
TI Item Name	Built into the Hub
Quantity	1
Included in	TI-Innovator™ Hub
Description	Indicates a Auxiliary power connection.
Category	LEDs and Displays
Hub Connection	on-board
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

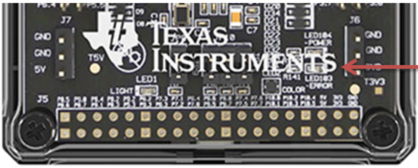
On-Board Green LED - Power Indicator Data Sheet



Green LED – Power Indicator (LED104)

Title	**Green LED - Power Indicator**
TI Item Name	Built into the Hub
Quantity	1
Included in	TI-Innovator™ Hub
Description	Indicates a USB connection on the DATA port.
Category	LEDs and Displays
Hub Connection	on-board
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

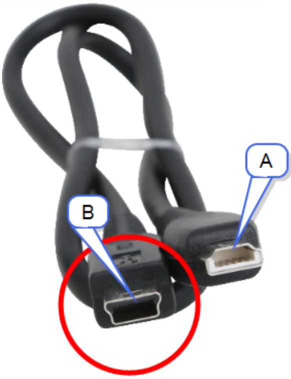
On-Board Red LED - Error Indicator Data Sheet



Title	**Red LED - Error Indicator**
TI Item Name	Built into the Hub
Quantity	1
Included in	TI-Innovator™ Hub
Description	Indicates an error in the sketch command.
Category	LEDs and Displays
Hub Connection	on-board Hub
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

See Also: On-Board Red LED

USB Mini A to Mini B Cable Data Sheet



Title	**USB Mini A to Mini B Cable**
TI Item Name	XX/CA/USB15/A
Quantity	1
Included in	TI-Innovator™ Hub
Description	Connects the Hub to a TI-CE Graphing Calculator or a TI-Nspire™ CX Handheld..
Category	Accessories
Hub Connection	Not Applicable
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

USB Standard A to Mini B Cable Data Sheet

Title	**USB Standard A to Mini B Cable**
TI Item Name	STEM/CA/USB20/A
Quantity	1
Included in	TI-Innovator™ Hub
Description	Connects the Hub to a computer running TI-Nspire™ CX Software.
Category	Accessories
Hub Connection	"B" connector to the USB Mini-B port
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

USB Standard A to Micro B Cable Data Sheet

Title	**USB Standard A to Micro B Cable**
TI Item Name	XX/CA/USB60/C
Quantity	1
Included in	TI-Innovator™ Hub
Description	Connects the Hub to a TI approved power source used with peripherals that require the 5V output port.
Category	Accessories
Hub Connection	"B" connector to the USB Mini-B port
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

TI Wall Charger Data Sheet

Title	**TI Wall Charger**
TI Item Name	XX/AD/9212USB/A
Quantity	1
Included in	TI-Innovator™ Hub
Description	Wall charger that supplies power through the TI-Innovator™ Hub for connected modules that require additional power.
Category	Accessories
Hub Connection	Micro connector of the USB Standard A to Micro B Cable to the PWR connector
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

External Battery Data Sheet

Title	**External Battery**
TI Item Name	STEMBT/A
Quantity	1
Included in	External Battery Pack
Description	External battery that supplies power through the TI-Innovator™ Hub for connected modules that require additional power.
Category	Accessories
Hub Connection	Micro connector of the USB Standard A to Micro B Cable to the PWR connector.
Assembly Instructions	Connect to PWR port on TI-Innovator™ Hub
Precautions	Not Applicable
Specifications	Not Applicable

Guida alla configurazione del TI-Innovator™ Rover

TI-Innovator™ Rover è un veicolo robotico programmabile a due ruote che funziona con l'Hub TI-Innovator™ Hub con scheda TI LaunchPad™. L'utente comunica con l'Hub TI-Innovator™ e controlla il Rover tramite i comandi di programmazione TI Basic. I componenti integrati comprendono due motori, un sensore per i colori, un ranger ultrasonico, un giroscopio e LED RGB.

Gli argomenti della guida introduttiva comprendono:

- Panoramica su TI-Innovator™ Rover
- Contenuto della confezione
- Requisiti per la configurazione di TI-Innovator™ Rover
- Preparazione del TI-Innovator™ Rover
- Collegamento del TI-Innovator™ Rover
- Esplorazione del TI-Innovator™ Rover assemblato
- Precauzioni generali

Panoramica su TI-Innovator™ Rover

TI-Innovator™ Rover è un veicolo robotico programmabile a due ruote che funziona con l'Hub TI-Innovator™ con scheda TI LaunchPad™. L'utente comunica con l'hub e controlla il Rover tramite i programmi TI Basic su uno di questi prodotti TI:

- Famiglia di calcolatrici grafiche TI CE (TI-83 Premium CE, TI-84 Plus CE e TI-84 Plus CE-T) con sistema operativo installato versione 5.3 o successiva. Può anche essere necessario installare o aggiornare l'app Hub che contiene il menu Hub.
- Palmari TI-Nspire™ CX o TI-Nspire™ CX CAS con sistema operativo installato versione 4.5 o successiva
- Software per computer TI-Nspire™ versione 4.5 o successiva

Seguire le istruzioni del presente manuale per configurare il TI-Innovator™ Rover con la calcolatrice grafica TI CE o il palmare TI-Nspire™ CX.

Ulteriori informazioni

Consultare la guida elettronica TI-Innovator™ Technology eGuide per ulteriori dettagli.

La eGuide è un manuale basato sul web contenente informazioni concernenti TI-Innovator™, compreso:

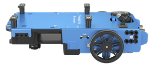
- Programmazione con la famiglia di calcolatrici grafiche TI CE e la tecnologia TI-Nspire™, inclusi programmi campione.
- Moduli I/O disponibili e rispettivi comandi.
- Componenti della basetta sperimentale disponibili e rispettivi comandi.
- TI-Innovator™ Rover e rispettivi comandi.
- Collegamento per aggiornare il software TI-Innovator™ Sketch.
- Attività didattiche gratuite per Hub e Rover.

Per accedere alla eGuide, visitare https://education.ti.com/go/eguide/hub/EN.

Per un riepilogo delle precauzioni da prendere quando si usa il Rover e i suoi componenti, consultare *Precauzioni generali* (page 250).

Requisiti per la configurazione di TI-Innovator™ Rover

Per configurare il TI-Innovator™ Rover con l’Hub TI-Innovator™ e una calcolatrice grafica saranno necessari gli articoli riportati di seguito.

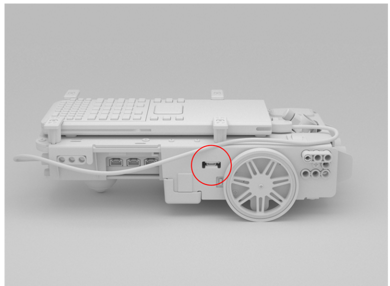
Componente	Immagine	Descrizione
TI-Innovator™ Rover		Un veicolo robotico programmabile a due ruote che funziona con l’hub.
Cavo a nastro della basetta sperimentale		Collega il Rover al connettore della basetta sperimentale dell’hub.
Cavo I^{2}C		Collega il Rover alla porta I^{2}C sull’hub.
Hub TI-Innovator™ con scheda TI LaunchPad™		Controlla il Rover per mezzo di comandi di programmazione TI Basic.
USB Unit-to-Unit (Mini-A to Mini-B) Cavo		Incluso nel Hub. Collega il Hub a una calcolatrice grafica TI CE o a un palmare TI-Nspire™ CX.
USB Standard A to Micro Cavo		Incluso nell’hub. Collega la porta **PWR** del Rover a un alimentatore elettrico approvato da TI.
Calcolatrice grafica TI CE oppure palmare TI-Nspire™ CX		Esegue i programmi TI Basic per inviare i comandi all’hub.
TI Wall Charger		Incluso nell’hub. Alimentatore elettrico per caricare il Rover.

Preparazione del TI-Innovator™ Rover

Seguire i seguenti passaggi per caricare completamente il TI-Innovator™ Rover.

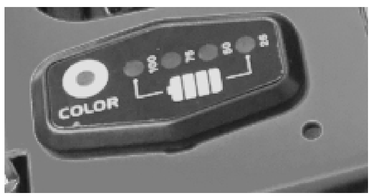
1. Identificare il connettore Micro sul USB Standard A to Micro cavo.

2. Inserire il connettore Micro nella porta **PWR** sul lato del Rover.



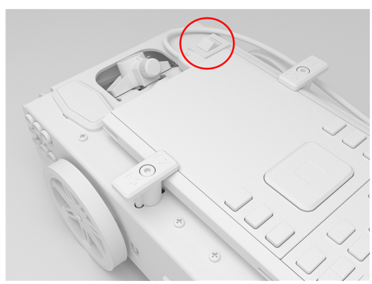
3. Inserire l’estremità libera del cavo (il connettore "**A**") nella porta USB del computer o TI Wall Charger.

   Nota: La spia verde dell’indicatore di carica della batteria è accesa in modo fisso quando la batteria è completamente carica.



Assicurarsi che il TI-Innovator™ Rover sia **SPENTO** prima di collegarlo all’Hub TI-Innovator™.

- Spostare l’interruttore **On/Off (I/O)** nella posizione di spegnimento **Off (O)**.



Collegamento del TI-Innovator™ Rover

Ci sono due set di operazioni di collegamento per usare il TI-Innovator™ Rover.

- Per prima cosa, collegare il Rover all’Hub TI-Innovator™, usando i due cavi a nastro forniti.
- Poi, collegare l’hub a una calcolatrice grafica, usando il USB Unit-to-Unit (Mini-A to Mini-B) cavo incluso con l’hub.

Collegamento del TI-Innovator™ Rover all’Hub TI-Innovator™

1. Inserire il **cavo a nastro della basetta sperimentale** nel **connettore della basetta sperimentale** sull’hub.

   Nota: È fondamentale inserire il cavo correttamente. Assicurarsi che lo spinotto del filo rosso (scuro) sia inserito nel foro 5v sul **connettore della basetta sperimentale** dell’hub.

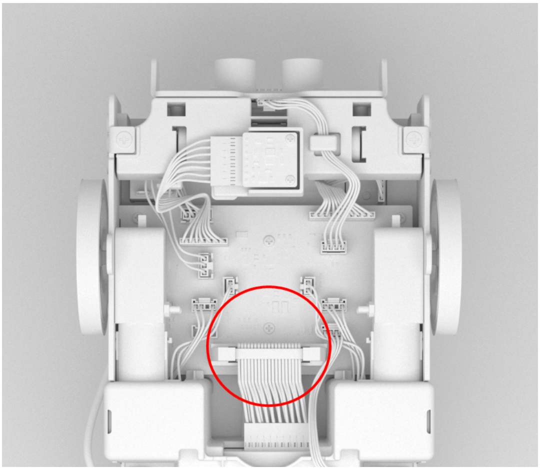
2. Guidare attentamente il cavo a nastro attaccato attraverso l’apertura sul lato posteriore del Rover.
3. Durante il passaggio del cavo, far scorrere l’hub in posizione usando le **rotaie di guida**.

   Quando l’hub è inserito correttamente si sentirà un click.

4. Aprire i due nottolini sul **connettore del cavo della scheda con i circuiti integrati del Rover**.

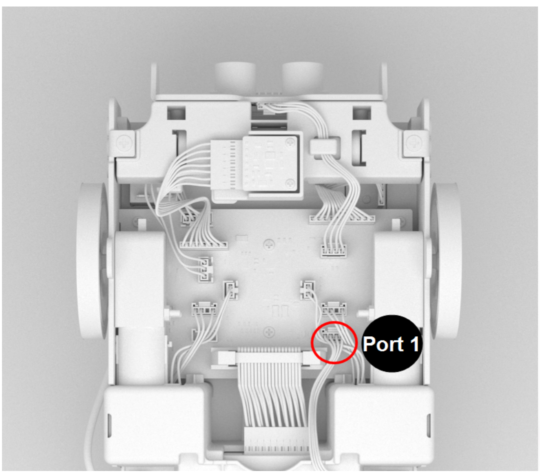
5. Allineare l’incavo nel cavo a nastro con la scanalatura sul connettore della scheda con i circuiti integrati.

6. Inserire il cavo a nastro e chiudere i nottolini.

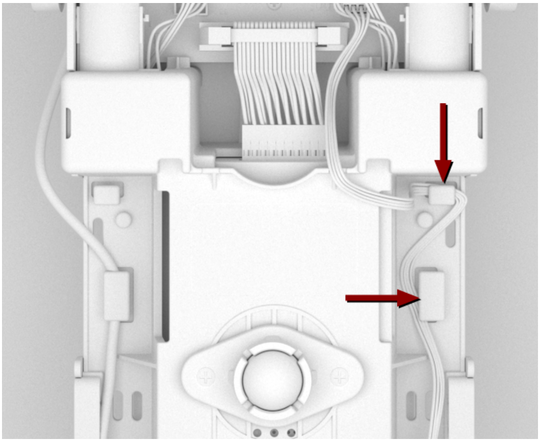


7. Inserire un’estremità del cavo **I²C** nella scheda con i circuiti integrati del Rover.

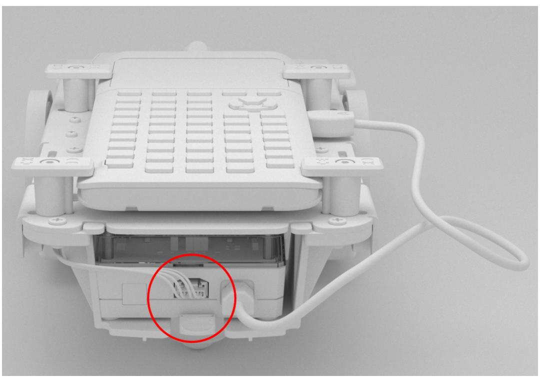
   Nota: Esistono due porte **I²C possibili**. Usare la **porta 1**.



8. Inserire il cavo **I²C** lasco nelle rotaie laterali.



9. Allineare la linguetta sul cavo **I²C** alla parte superiore della porta **I²C**.
10. Inserire l'estremità libera del connettore del cavo **I²C** nella porta **I²C** sul lato posteriore dell'hub.

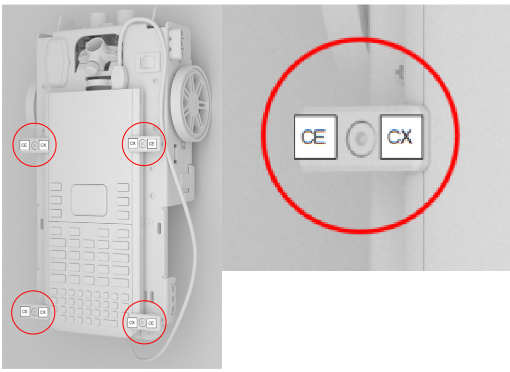


Collegamento dell’Hub TI-Innovator™ a una calcolatrice grafica

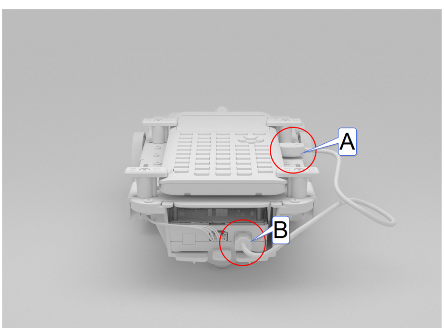
1. Girare il Rover in modo che il lato destro sia rivolto verso l’alto.
2. Sollevare e girare i **fermagli per il supporto della calcolatrice** in modo che siano paralleli al lato del Rover.
3. Posizionare la calcolatrice grafica TI CE o il palmare TI-Nspire™ CX sulla piattaforma con lo schermo rivolto verso il **supporto pennarello**.
4. Girare i fermagli in modo che l’etichetta CE o CX sia posizionata verso l’interno in modo da corrispondere alla calcolatrice grafica.

   I fermagli scatteranno in posizione quando sono posizionati correttamente.

   Attenzione: Non girare i **fermagli per il supporto della calcolatrice** senza averli prima sollevati. Potrebbero rompersi.



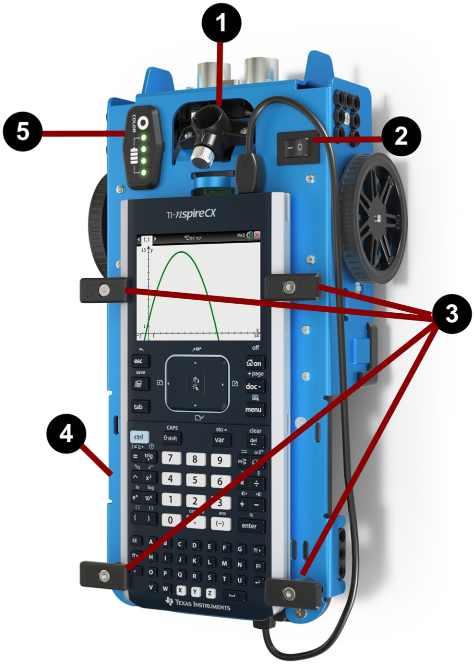
5. Identificare il connettore "**B**" sul cavo **USB di collegamento tra unità (da Mini-A a Mini-B)**. Su ogni estremità di questo cavo è stata impressa una lettera.
6. Inserire il connettore "**B**" nella porta **DATI** sull’hub.
7. Inserire l’estremità libera del cavo (il connettore "**A**") nella porta USB della calcolatrice grafica.



Esplorazione del TI-Innovator™ Rover assemblato

Esplorare tutti i lati del TI-Innovator™ Rover assemblato con l'Hub TI-Innovator™ e collegato alla calcolatrice grafica TI CE o al palmare TI-Nspire™ CX.

Lato superiore del Rover



❶ **Supporto pennarello** - Supporta un pennarello per tracciare delle linee.

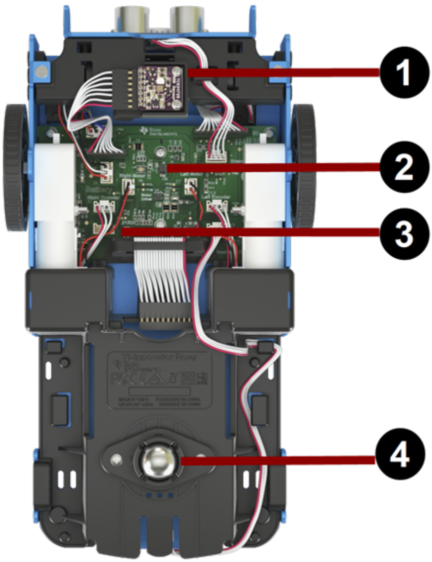
❷ **Interruttore ON/OFF (I/O)** - **ACCENDE (–)** o **SPEGNE (O) il Rover**.

❸ **Fermagli per il supporto della calcolatrice** - Fissa una calcolatrice grafica alla piattaforma per calcolatrice.

❹ **Piattaforma per calcolatrice** - Supporta una calcolatrice grafica TI CE o un palmare TI-Nspire™ CX.

❺ **Pannello a LED (LED RGB/indicatore di carica della batteria)** - Visualizza un feedback programmabile per mezzo di **LED rosso-verde-blu (Red-Green-Blue, RGB)** e il livello di carica della batteria.

Lato inferiore del Rover



1 **Sensore per i colori** - Il sensore per i colori montato sul fondo rileva il colore della superficie. Può anche rilevare una scala di grigi da nero (0) a bianco (255).

2 **Giroscopio** - Misura e mantiene l’orientamento.

3 **Porta di espansione I^{2}C**.

4 **Ruota sferica** - Realizza un movimento agevole sulle superfici dure.
Nota: Non consigliato per l’uso sui tappeti.

Attenzione: Se si rimuove o scollega uno qualsiasi dei cavi, fare riferimento a questa immagine per ripristinare collegamenti in maniera corretta.

Lato anteriore del Rover

Ranger ultrasonico - Misura le distanze rispetto a eventuali ostacoli.

Lato posteriore del Rover

Rotaie di guida - Permette all’hub di scorrere facilmente nel Rover e collegare la scheda con i circuiti integrati del Rover.

Nota: Con l’Hub TI-Innovator™ inserito, accedere a un sensore e a due porte.

- **Sensore di intensità luminosa** - Legge la "LUMINOSITA" nelle stringhe di comando dell’hub.
- **Porta I^2C** - Usa un cavo I^2C per collegare l’hub alla scheda con i circuiti integrati del Rover.
- **DATI** Porta mini-B - Usa il cavo USB di collegamento tra unità (da mini-A a mini-B) per collegare l’hub a una calcolatrice grafica.

Lato destro del Rover

Accesso al Rover:

- **Porta PWR** - Usa un cavo elettrico ausiliario USB standard A a Micro per caricare la batteria ricaricabile del Rover.
- **Montaggi anteriori e posteriori** - Per aggiungere delle strutture al Rover usando blocchetti di connessione di plastica.

Nota: Con l’hub inserito, accedere tre porte per controllare i moduli di output.

- **OUT 1** e **OUT 2** forniscono una potenza di 3,3 V.
- **OUT 3** fornisce una potenza di 5 V.

Lato sinistro del Rover

Accesso al Rover:

- **Montaggi anteriori e posteriori** - Per aggiungere delle strutture al Rover usando blocchetti di connessione di plastica.

Nota: Con l’hub inserito, accedere tre porte per raccogliere i dati o lo stato dai moduli di input.

- **IN 1** e **IN 2** forniscono una potenza di 3,3 V.

- **IN 3** fornisce una potenza di 5 V.

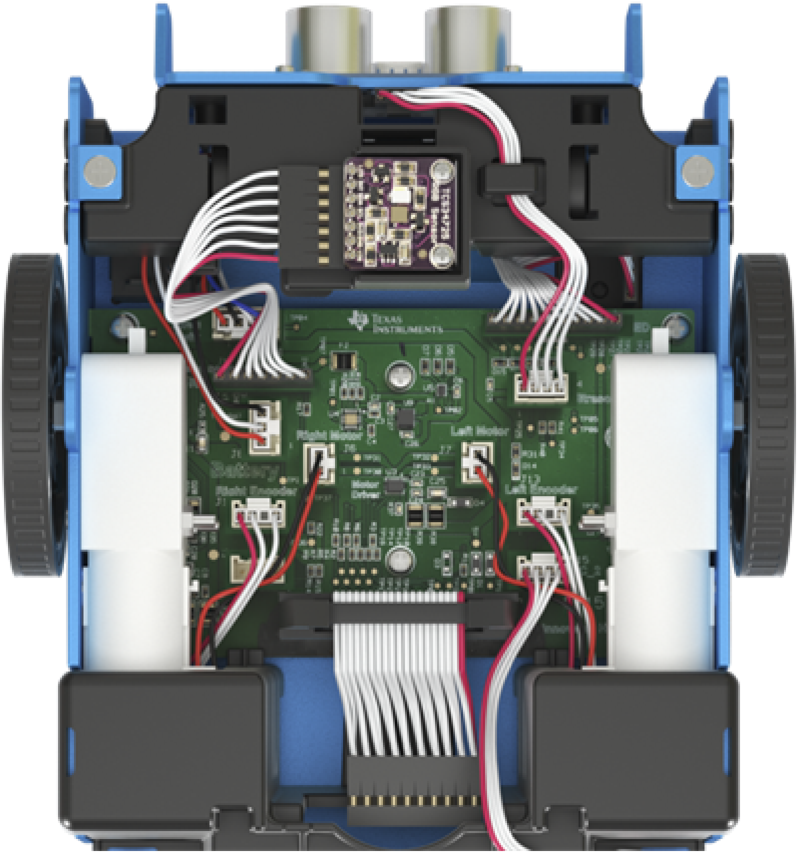
Precauzioni generali

TI-Innovator™ Rover

- Non esporre il Rover a temperature superiori ai 60°C (140°F).
- Non smontare o danneggiare il Rover.
- Non appoggiare nulla che pesi più di 1 Kg o 2,2 lb sulla Rover piattaforma.
- Usare unicamente il cavo USB fornito con il TI-Innovator™ Hub.
- Usare unicamente i cavi a nastro forniti con il Rover.
- Usare unicamente il caricatore da muro fornito da TI incluso con l’Hub.
- Il Ultrasonic Ranger montato anteriormente rileverà gli oggetti a una distanza fino a 4 metri dal Rover. Per ottenere i risultati migliori assicurarsi che la superficie dell’oggetto sia più grande di una cartellina. Se usato per rilevare oggetti di piccole dimensioni, come una tazza, posizionare il Rover a una distanza fino a 1 metro dall’oggetto.
- Per ottenere i risultati migliori rimuovere la custodia della calcolatrice grafica.
- Per le prestazioni migliori usare Rover sul pavimento e non su tavoli. La caduta del Rover da un tavolo può danneggiarlo.
- Per le prestazioni migliori usare Rover su una superficie dura. Le ruote del Rover possono impigliarsi in un tappeto o agganciarlo e tirarlo.
- Non girare i fermagli di supporto sulla piattaforma per la calcolatrice senza averli prima sollevati. Potrebbero rompersi.
- Non usare il pennarello come una leva per tirare o spingere il Rover.
- Non svitare la cassa protettiva sul fondo del Rover. Gli encoder hanno spigoli appuntiti che non devono essere esposti.
- Non muovere Rover dopo l’esecuzione di un programma. Il giroscopio interno può cercare involontariamente di rimettere in carreggiata il Rover usando la posizione iniziale.
- Quando si inserisce il cavo a nastro della basetta sperimentale nel connettore della basetta sperimentale dell’Hub, è fondamentale che il cavo venga inserito correttamente. Assicurarsi che lo spinotto del filo rosso (scuro) sia inserito nel foro 5v sul connettore della basetta sperimentale dell’Hub's.

Attenzione: Se si rimuove o scollega uno qualsiasi dei cavi, fare riferimento a questa immagine per ripristinare collegamenti in maniera corretta.

Riferimento alla vista dal basso



TI-Innovator™ Rover Commands Version 1.4

Prerequisite: Use the Send "Connect RV" Command First

The "**CONNECT RV**" command needs to be used first when using the Rover. The "**CONNECT RV**" command configures the TI-Innovator™ Hub software to work with the TI-Innovator™ Rover.

It establishes the connections to the various devices on the Rover – two motors, two encoders, one gyroscope, one RGB LED and one color sensor. It also clears the various counters and sensor values. The optional 'MOTORS' parameter configures only the motors and allows direct control of motors without the additional peripherals.

CONNECT RV - initializes the hardware connections.

- Connects RV and inputs and outputs built into the RV.
- Resets the Path and the Grid Origin.
- Sets the units per meter to default value of 10. Default Grid unit = 10cm.

Named RV Subsystems

The RV object contains several subsystems that are directly addressed by name. These subsystems consist of the wheels, and sensors that let the Rover sense the world.

The subsystems are listed by name in the following table.

Subsystem Name	Description of Subsystem
RV	The RV object as a whole.
RV.COLOR	The tri-color RGB LED on the top surface of the Rover can be controlled through user programs to display any color combination.
RV.COLORINPUT	The color sensor is on the bottom of the Rover and is used to detect the color of the surface.
RV.RANGER	The front-facing ultrasonic distance sensor. Returns measurements in meters. ~10.00 meters means no obstacle was detected.
RV.ENCODERGYRO	The rotary encoders – one on each motor – measure the distance traveled by the Rover. The left and right encoder, coupled with the gyroscope and operating time information.
RV.GYRO	The gyroscope is used to maintain the heading of Rover while it's in motion. It can also be used to measure the change in angle during turns.
RV.MOTOR.L	Left wheel motor and control for direct control (advanced) use.
RV.MOTOR.R	Right wheel motor and control for direct control

Subsystem Name	Description of Subsystem
	(advanced) use.
RV.MOTORS	Both the LEFT and RIGHT motor, managed as a single object for direct control (advanced) use.

Rover Command Categories

The Rover commands fall into two categories:

1. Queued execution: All of the Rover motion commands – FORWARD, BACKWARD, LEFT, RIGHT, ANGLE – are queued on the TI-Innovator Hub. They may execute at a future time.
2. Immediate execution: Other commands – like the ones to read the sensors or set the RGB LED on the Rover – are executed immediately.

This means that certain statements in your program will execute before statements that appear earlier in the program especially if the latter commands are part of the queued family.

For example, in the program below, the RGB LED will turn RED before the Rover stops moving:

`Send "SET RV.COLOR 255 0 255"` – immediately executed

`Send "RV FORWARD 5"` – queued command

`Send "RV LEFT 45"` – queued command

`Send "RV RIGHT 90"` – queued command

`Send "SET RV.COLOR 255 0 0"` – immediately executed

Example:

To change color after a "FORWARD" movement, use "TIME" parameter with "WAIT".

```
Send "RV FORWARD TIME 5"

WAIT 5

Send "SET RV.COLOR 255 0 255"
```

RV Commands, Code Samples, and Syntax

The following examples show how various commands for the RV are used. Anywhere a **SET** command is used, the **SET** may be left off (optional use).

Code Samples

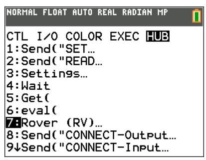
When you see "**Code Sample**" in a command table, this "**Code Sample**" may be copied and pasted *as is* to send to your graphing calculator to use in your calculations.

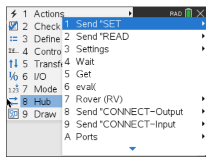
Example:

Code Sample:	`Send ("RV FORWARD 5")` `Send ("RV FORWARD SPEED 0.2 M/S TIME 10")`

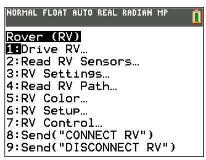
TI-Innovator™ Rover Menu

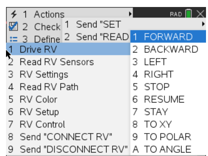
Rover (RV)…	**CE Calcolatrici**	**TI-Nspire™ CX**





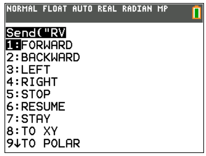
- Drive RV…
- Read RV Sensors…
- RV Settings…
- Read RV Path…
- RV Color…
- RV Setup…
- RV Control…
- Send("CONNECT RV")
- Send("DISCONNECT RV")



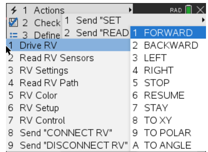


- **Drive RV…**
  - Send("RV
    - FORWARD
    - BACKWARD
    - LEFT
    - RIGHT
    - STOP
    - RESUME
    - STAY
    - TO XY
    - TO POLAR
    - TO ANGLE

CE Calcolatrici

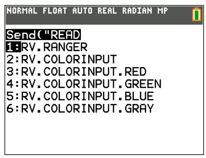


TI-Nspire™ CX

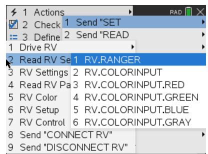


- **Read RV Sensors…**
  - Send"READ"
    - RV.RANGER
    - RV.COLORINPUT
    - RV.COLORINPUT.RED
    - RV.COLORINPUT.GREEN
    - RV.COLORINPUT.BLUE
    - RV.COLORINPUT.GRAY

CE Calcolatrici

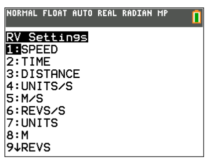


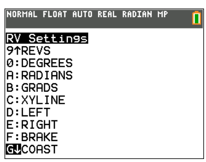
TI-Nspire™ CX



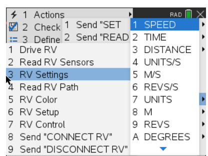
- **RV Settings…**
  - RV Settings
    - SPEED
    - TIME
    - DISTANCE
    - UNIT/S
    - M/S
    - REV/S
    - UNITS
    - M
    - REVS
    - DEGREES
    - RADIANS

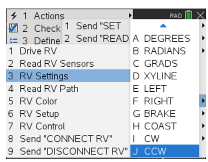
CE Calcolatrici





TI-Nspire™ CX

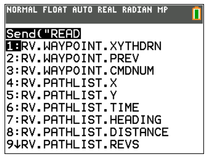


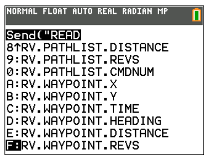


- GRADS
- XYLINE
- LEFT
- RIGHT
- BRAKE
- COAST
- CW
- CCW

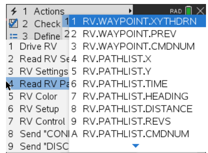
- **Read RV Path…**
  - Send "READ
    - RV.WAYPOINT.XYTHDRN
    - RV.WAYPOINT.PREV
    - RV.WAYPOINT.CMDNUM
    - RV.PATHLIST.X
    - RV.PATHLIST.Y
    - RV.PATHLIST.TIME
    - RV.PATHLIST.HEADING
    - RV.PATHLIST.DISTANCE
    - RV.PATHLIST.REVS
    - RV.PATHLIST.CMDNUM
    - RV.WAYPOINT.X
    - RV.WAYPOINT.Y
    - RV.WAYPOINT.TIME
    - RV.WAYPOINT.HEADING
    - RV.WAYPOINT.DISTANCE
    - RV.WAYPOINT.REVS

CE Calcolatrici



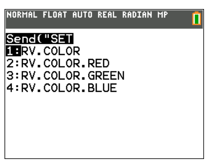


TI-Nspire™ CX

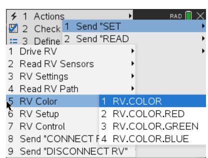


- **RV Color…**
  - Send "SET
    - RV.COLOR
    - RV.COLOR.RED
    - RV.COLOR.GREEN
    - RV.COLOR.BLUE

CE Calcolatrici

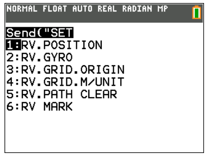


TI-Nspire™ CX

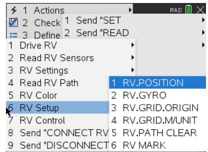


- **RV Setup…**
  - Send "SET
    - RV.POSITION
    - RV.GYRO
    - RV.GRID.ORIGIN
    - RV.GRID.M/UNIT
    - RV.PATH CLEAR
    - RV MARK

CE Calcolatrici

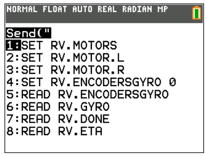


TI-Nspire™ CX

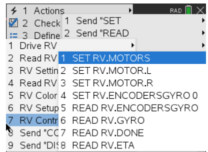


- **RV Control…**
  - Send "
    - SET RV.MOTORS
    - SET RV.MOTOR.L
    - SET RV.MOTOR.R
    - SET RV.ENCODERSGYRO 0
    - READ RV.ENCODERSGYRO
    - READ RV.GYRO
    - READ RV.DONE
    - READ RV.ETA

CE Calcolatrici

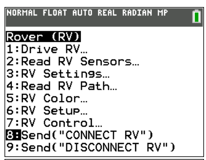


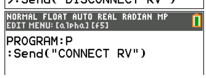
TI-Nspire™ CX



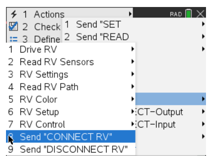
- **Send "CONNECT RV"**
  - Send "CONNECT RV"
    - CONNECT RV

CE Calcolatrici



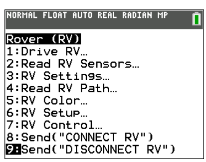


TI-Nspire™ CX

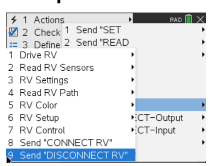


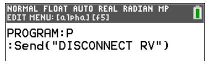
- **Send "DISCONNECT RV"**
  - Send "DISCONNECT RV"
    - DISCONNECT RV

CE Calcolatrici



TI-Nspire™ CX




NORMAL FLOAT AUTO REAL RADIAN MP
EDIT MENU: [alpha] [f5]
PROGRAM:P
:Send("DISCONNECT RV")

Drive RV...

RV Drive Command Families

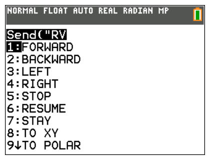
- Base Drive Commands (in the spirit of Turtle Graphics)
  - FORWARD, BACKWARD, RIGHT, LEFT, STOP, STAY
- Math Coordinate Drive Commands
  - Turn to Angle

Note: Drive commands have options for Speed, Time and Distance as appropriate

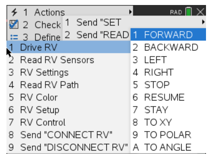
- See RV Settings for Machine-Level Control Commands
  - Set Left and Right Motor values for direction (CW/CCW) and level (0-255,Coast)
  - Read accumulated values for wheel encoder edges and gyro heading change.

- **Drive RV…**
  - Send("RV
    - FORWARD
    - BACKWARD
    - LEFT
    - RIGHT
    - STOP
    - RESUME
    - STAY
    - TO XY
    - TO POLAR
    - TO ANGLE

CE Calcolatrici



TI-Nspire™ CX



RV FORWARD

Command:	**RV FORWARD**
Command Syntax:	**RV FORWARD [[SPEED s] [DISTANCE d] [TIME t]]**

Code Samples:	`Send ("RV FORWARD 0.5 M")`
	`Send ("RV FORWARD SPEED 0.22 M/S TIME 10")`
	--- 	
	`[SET] RV FORWARD`
	`[SET] RV FORWARD [DISTANCE] d [M|UNIT|REV]`
	`[SET] RV FORWARD [DISTANCE] d [M|UNIT|REV]`
	`      SPEED s.ss [M/S|[UNIT/S]|REV/S]`
	`[SET] RV FORWARD [DISTANCE] d [M|UNIT|REV]`
	`      TIME t`
	`[SET] RV FORWARD SPEED s`
	`[M/S|UNIT/S|REV/S]`
	`      [TIME t]`
	`[SET] RV FORWARD TIME t [SPEED s.ss`
	`      [M/S|[UNIT/S]|REV/S]]`

Range:	N/A
Describe:	RV moves forward a given distance (default 0.75 m). Default distance if specified is in UNIT (grid units). Optional M=meters, UNIT=grid-unit, REV=wheel-revolution. Default speed is 0.20 m/sec, max is 0.23 m/sec, min is 0.14 m/sec. Speed may be given and specified in meters/second, unit/second, revolutions/second.
Result:	Action to make the RV move in a forward direction
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV BACKWARD

Command:	RV BACKWARD
Command Syntax:	**RV BACKWARD**

Code Sample:	`Send("RV BACKWARD 0.5 M")`
	`Send("RV BACKWARD SPEED 0.22 M/S TIME 10")`
	
	`[SET] RV BACKWARD`
	`[SET] RV BACKWARD [DISTANCE] d [M|UNIT|REV]`
	`[SET] RV BACKWARD [DISTANCE] d [M|UNIT|REV] SPEED s.ss [M/S|[UNIT/S]|REV/S]`
	`[SET] RV BACKWARD [DISTANCE] d [M|UNIT|REV] TIME t`
	`[SET] RV BACKWARD SPEED s.ss [M/S|UNIT/S|REV/S] [TIME t]`
	`[SET] RV BACKWARD TIME t [SPEED s.ss [M/S|UNIT/S|REV/S]]`

Range:	N/A
Describe:	RV moves backward a given distance (default 0.75 m). Default distance if specified is in UNIT (grid units). Optional M=meters, UNIT=grid-unit, REV=wheel-revolution. Default speed is 0.20 m/sec, max is 0.23 m/sec, min is 0.14 m/sec. Speed may be given and specified in meters/second, unit/second, revolutions/second.
Result:	Action to make the RV move in a backward direction.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV LEFT

Command:	RV LEFT
Command Syntax:	**RV LEFT**

Code Sample:	`Send "RV LEFT"`
	
	`[SET] RV LEFT [ddd [DEGREES]]`
	`[SET] RV LEFT [rrr RADIANS]`
	`[SET] RV LEFT [ggg GRADIANS]`

Range:	N/A
Describe:	Default turn is 90 degrees unless DEGREES, RADIANS, or GRADIANS keyword is present, and then the value is converted internally to degrees format from the specified units. Value given is ranged to a value between 0.0 and 360.0 degrees. The turn will be executed as a SPIN motion.
Result:	Turn Rover to the LEFT.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV RIGHT

Command:	RV RIGHT
Command Syntax:	**RV RIGHT**

Code Sample:	`Send "RV RIGHT"`
	
	`[SET] RV RIGHT [ddd [DEGREES]]`
	`[SET] RV RIGHT [rrr RADIANS]`
	`[SET] RV RIGHT [ggg GRADIANS]`

Range:	N/A
Describe:	Default turn is 90 degrees unless DEGREES, RADIANS, or GRADIANS keyword is present, and then the value is converted internally to degrees format from the specified units. Value given is ranged to a value between 0.0 and 360.0 degrees. The turn will be executed as a SPIN motion.

Command:	RV RIGHT
Result:	Turn Rover to the RIGHT.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV STOP

Command:	RV STOP
Command Syntax:	**RV STOP**

Code Sample:	`Send "RV STOP"`
	
	`[SET] RV STOP`
	
	`[SET] RV STOP CLEAR`

Range:	N/A
Describe:	The **RV** will stop any current movement immediately. That movement can be resumed from where it left off with a **RESUME** operation. Any movement commands will cause the queue to flush immediately, and begin the just-posted new movement operation
Result:	Stop processing Rover commands from the command queue, and leave pending operations in the queue. (immediate action). Queue can be resumed by **RESUME**.The **RV** will stop any current movement immediately. That movement can be resumed from where it left off with a **RESUME** operation. Any movement commands will cause the queue to flush immediately, and begin the just-posted new movement operation. Stop processing Rover commands from the command queue, and flush any pending operations left in the queue. (immediate action).
Type or Addressable Component:	Control **Note:** This Rover control command is executed immediately.

RV RESUME

Command:	RV RESUME
Command Syntax:	**RV RESUME**

Code Sample:	`Send "RV RESUME"`
	
	`[SET] RV RESUME`

Range:	N/A
Describe:	Enable processing of Rover commands from the command queue. (immediate action), or resume (see RV STAY) operation.
Result:	Resume operation.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV STAY

Command:	RV STAY
Command Syntax:	**RV STAY**

Code Sample:	`Send "RV STAY"`
	
	`[SET] RV STAY [[TIME] s.ss]`

Range:	N/A
Describe:	Tells RV to "stay" in place for an optionally specified amount of time in seconds. Default is 30.0 seconds.
Result:	RV stays in position.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV TO XY

Command:	**RV TO XY**
Command Syntax:	**RV TO XY** x-coordinate y-coordinate [[SPEED] s.ss [UNIT/S] |M/S|REV/S] [XYLINE]

Code Sample:	`Send "RV TO XY 1 1"`
	`Send "RV TO XY eval(X) eval(Y)"`
	`Send "RV TO XY 2 2 SPEED 0.23 M/S"`

Range:	-327 to +327 for X and Y coordinates
Describe:	This command controls the movement of Rover on a virtual grid. Default location at start of program execution is (0,0) with Rover facing the positive x-axis. The x and y coordinates match the current grid size (default: 0.1 M/grid unit). Grid size can be changed through “SET RV.GRID.M/UNIT“ command The speed parameter is optional.
Result:	Moves Rover from current grid location to the specified grid location.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV TO POLAR

Command:	**RV TO POLAR**
Command Syntax:	**RV TO POLAR** R-coordinate Theta-coordinate [[DEGREES] |RADIANS|GRADS] [[SPEED] s.ss [UNIT/S]|M/S|REV/S] [XYLINE]

Code Sample:	`Send("RV TO POLAR 5 30") - r = 5 units, theta = 30 degrees`
	`Send("RV TO POLAR 5 2 RADIANS")`
	`Send("RV TO POLAR eval(sqrt(3^2+4^2)) eval (tan-1(4/3) DEGREES ")`

Range:	Theta-coordinate: -360 to +360 degrees R-coordinate: -327 to +327
Describe:	Moves the RV from its current position to the specified polar position relative to that position. The RV's X/Y position will be updated to reflect the new position. The “r” coordinate matches the current grid size (default: 0.1 M/grid

Command:	RV TO POLAR
	unit). Default location at start of program execution is (0,0) with Rover facing the positive x-axis. Default unit of theta is Degrees. The speed parameter is optional.
Result:	Moves Rover from current grid location to the specified grid location.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV TO ANGLE

Command:	RV TO ANGLE
Command Syntax:	**RV TO ANGLE**

Code Sample:	`Send "RV TO ANGLE"`
	
	`[SET] RV TO ANGLE rr.rr [[DEGREES]|RADIANS|GRADIANS]`

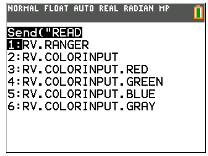
Range:	N/A
Describe:	
Result:	Spins the RV to the specified angle from current heading.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

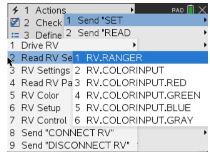
READ RV Sensors…

SEND("Read Sensor Commands

- Reading of low level sensors for learning foundations of robotics.

- **Read RV Sensors…**
  - Send("READ
    - RV.RANGER
    - RV.COLORINPUT
    - RV.COLORINPUT.RED
    - RV.COLORINPUT.GREEN
    - RV.COLORINPUT.BLUE
    - RV.COLORINPUT.GRAY





- **RV.RANGER:** Returns value in Meters.
- **RV.COLORINPUT:** Reads color sensor that is built into the RV.

RV.RANGER

<table>
<tr><th>Command:</th><th>RV.RANGER</th></tr>
<tr><td>Command Syntax:</td><td>RV.RANGER</td></tr>
<tr><td>Code Sample:</td><td>Send("READ RV.RANGER")
Get(R)
<table>
<tr><td>Connects the Rover Vehicle to the TI-Innovator™ Hub. This establishes connections with the motor driver, color sensor, gyroscope, ultrasonic ranger, and proximity sensors.</td><td>CONNECT RV</td></tr>
<tr><td>Returns the current distance from the front of the RV to an obstacle. If there is no obstacle detected, a range of 10.00 meters is reported</td><td>READ RV.RANGER
Get(R)</td></tr>
</table>
</td></tr>
</table>

Command:	RV.RANGER
Range:	N/A
Describe:	The front-facing ultrasonic distance sensor. Returns measurements in meters. ~10.00 meters means no obstacle was detected.
Result:	Returns value in Meters.
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT

Command:	RV.COLORINPUT
Command Syntax:	**RV.COLORINPUT**
Code Sample:	`Send("READ RV.COLORINPUT")` `Get(C)`
Range:	1 thru 9
Describe:	Bottom-mounted color sensor detects the color of the surface. Can also detect gray-level scale of black (0) to white (255).
Result:	Returns current color sensor information. The return value is in the 1 – 9 range which maps to the colors below: **Color** — **Return value** Red — 1 Green — 2 Blue — 3 Cyan — 4 Magenta — 5 Yellow — 6 Black — 7 White — 8 Gray — 9
Type or	Sensor

Command:	RV.COLORINPUT
Addressable Component:	**Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT.RED

Command:	RV.COLORINPUT.RED
Command Syntax:	**RV.COLORINPUT.RED**
Code Sample:	`Send("READ RV.COLORINPUT.RED")` `Get(R)`
Range:	0 - 255
Describe:	Detect intensity of individual red components of surface. The results are in 0-255 range.
Result:	Returns current color sensor "red value".
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT.GREEN

Command:	RV.COLORINPUT.GREEN
Command Syntax:	**RV.COLORINPUT.GREEN**
Code Sample:	`Send("READ RV.COLORINPUT.GREEN")` `Get(G)`
Range:	0 - 255
Describe:	Detect intensity of individual green components of surface. The results are in 0-255 range.
Result:	Returns current color sensor "green" value.
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT.BLUE

Command:	RV.COLORINPUT.BLUE
Command Syntax:	**RV.COLORINPUT.BLUE**
Code Sample:	`Send("READ RV.COLORINPUT.BLUE")` `Get(B)`
Range:	0 - 255
Describe:	Detect intensity of individual blue components of surface. The results are in 0-255 range.
Result:	Returns current color sensor "blue" value.
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

RV.COLORINPUT.GRAY

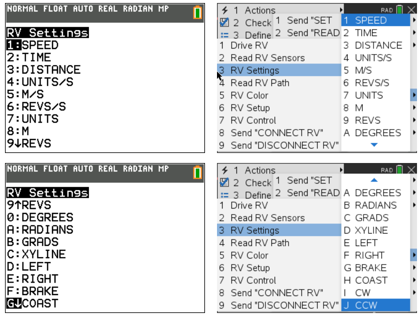
Command:	RV.COLORINPUT.GRAY
Command Syntax:	**RV.COLORINPUT.GRAY**
Code Sample:	`Send("READ RV.COLORINPUT.GRAY")` `Get(G)`
Range:	0 - 255
Describe:	Detect grayness of surface. The result will be in 0-255 range.
Result:	Returns an interpolated "grayscale" value based on 0.3*red + 0.59*green + 0.11*blue 0-black, 255 - white.
Type or Addressable Component:	Sensor **Note:** This Rover sensor command is executed immediately.

RV Settings...

RV Settings Commands

Settings menu for Rover contains other commands that support RV commands such as FORWARD or BACKWARD.

- **RV Settings...**
  - RV Settings
    - SPEED
    - TIME
    - DISTANCE
    - UNIT/S
    - M/S
    - REV/S
    - UNITS
    - M
    - REVS
    - DEGREES
    - RADIANS
    - GRADS
    - XYLINE
    - LEFT
    - RIGHT
    - BRAKE
    - COAST
    - CW
    - CCW



Read RV Path…

Reading WAYPOINT and PATH

Tracking the RV's Path

In order to support analysis of the Rover during and after a run, the sketch will automatically measure the following information for each Drive command:

- X Coordinate on virtual grid
- Y Coordinate on virtual grid
- Time in seconds that the current command has been executing.
- Distance in coordinate units for the path segment.
- Heading in degrees (absolute terms measured Counter Clockwise with the X-axis as 0 degrees.
- Revolutions by the wheel in executing the current command
- Command number, tracks the number of commands executed, begins with 0.

The Path values will be stored in lists, starting with the segments associated with the earliest commands and going to the segments associated with the latest commands.

The drive command in progress, the **WAYPOINT**, will repeatedly update the last element in the Path lists as the Rover progresses toward the last waypoint.

When a drive command is completed a new waypoint is initiated and the dimension of the Path lists are incremented.

> **Note:** This implies that when all the drive commands in the queue are completed that another waypoint for the stopped state is automatically started. This is similar to the initial position where the RV is stationary and counting time.

Max number of waypoints: 80

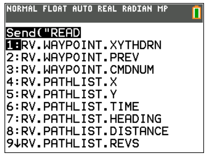
RV Position and Path

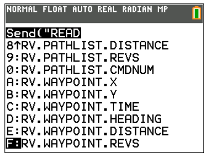
- Ability to read X,Y coordinate, Heading, Time and Distance for each drive command in execution.
- Will store path history in lists for plotting and analysis

Note: Coordinate grid scale can be set by the user, default is 10cm per unit. The user will have options to set the origin of the grid.

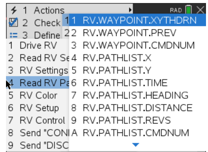
- **Read RV Path…**
  - Send("READ
    - RV.WAYPOINT.XYTHDRN
    - RV.WAYPOINT.PREV
    - RV.WAYPOINT.CMDNUM
    - RV.PATHLIST.X
    - RV.PATHLIST.Y
    - RV.PATHLIST.TIME
    - RV.PATHLIST.HEADING
    - RV.PATHLIST.DISTANCE
    - RV.PATHLIST.REVS
    - RV.PATHLIST.CMDNUM
    - RV.WAYPOINT.X
    - RV.WAYPOINT.Y
    - RV.WAYPOINT.TIME
    - RV.WAYPOINT.HEADING
    - RV.WAYPOINT.DISTANCE
    - RV.WAYPOINT.REVS

CE Calcolatrici





TI-Nspire™ CX



RV.WAYPOINT.XYTHDRN

Command:	RV.WAYPOINT.XYTHDRN
Command Syntax:	**RV.WAYPOINT.XYTHDRN**
Code Sample:	`Send("READ RV.WAYPOINT.XYTHDRN")`
Example:	Getting the distance traveled toward the current way-point from the last way-point
Code Sample:	`Send("READ RV.WAYPOINT.XYTHDRN")` Get(L_1) (L_1)(5)->D
Range:	N/A
Describe:	READ RV.WAYPOINT.XYTHDRN - read the x-coord, y-coord, time, heading, distance traveled, number of wheel revolutions, command number of the current waypoint. Returns a list with all these values as elements.
Result:	Return list of current way-point X, Y coordinates, Time, Heading, Distance, Revolutions, and command number.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.PREV

Command:	RV.WAYPOINT.PREV
Command Syntax:	**RV.WAYPOINT.PREV**
Code Sample:	`Send("READ RV.WAYPOINT.PREV")`
Example:	Getting the distance traveled during the previous way-point.
Code Sample:	`Send("READ RV.WAYPOINT.PREV")` Get(L_1) (L_1)(5)->D

Command:	RV.WAYPOINT.PREV
Range:	N/A
Describe:	READ RV.WAYPOINT.PREV - read the x-coord, y-coord, time, heading, distance traveled, number of wheel revolutions, command number of the previous waypoint. Returns a list with all these values as elements.
Result:	Return list of the previous way-point X, Y coordinates, time, heading, distance, revolutions, and command number.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.CMDNUM

Command:	RV.WAYPOINT.CMDNUM
Command Syntax:	**RV.WAYPOINT.CMDNUM**
Code Sample:	`Send("READ RV.WAYPOINT.CMDNUM")`
Example:	Program to determine if a drive command has completed without referring to a specific command number. **Note:** the **Wait** is intended to increase the probability of catching a difference in the Command Number.

Code Sample:	`Send("RV FORWARD 10")`
	`Send("READ RV.WAYPOINT.CMDNUM")`
	`Get(M)`
	`M->N`
	
	While M=N
	
	`Send("READ RV.WAYPOINT.CMDNUM")`
	`Get(N)`
	`End`
	
	Disp "Drive Command is completed"

Range:	N/A

Command:	**RV.WAYPOINT.CMDNUM**
Describe:	READ RV.WAYPOINT.CMDNUM - returns the last command number of the current waypoint.
Result:	Returns a value of 0 if the RV is currently "working" on a command and is either in motion, or running a STAY operation. This command will return a value of 1 when ALL queued operations are completed, nothing is remaining in the command queue, and the current operation has completed (and immediately after CONNECT RV).
Type or Addressable Component:	Returns Data

RV.PATHLIST.X

Command:	**RV.PATHLIST.X**
Command Syntax:	**RV.PATHLIST.X**
Code Samples:	`Send("READ RV.PATHLIST.X")`
Example:	Program to plot the RV path on the graph screen

Code Samples:	`Plot1(xyLine, L1, L2,▫,BLUE)`
	`Send("READ RV.PATHLIST.X")`
	`Get(L1)`
	`Send("READ RV.PATHLIST.Y")`
	`Get(L2)`
	`DispGraph`

Range:	N/A
Describe:	READ RV.PATHLIST.X - returns a list of X values from the beginning to and including the current Waypoint X value.
Result:	Return list of X coordinates traversed since last **RV.PATH CLEAR** or initial **CONNECT RV**.
Type or Addressable Component:	Returns Data

RV.PATHLIST.Y

Command:	RV.PATHLIST.Y
Command Syntax:	**RV.PATHLIST.Y**
Code Sample:	`Send("READ RV.PATHLIST.Y")`
Example:	Program to plot the RV path on the graph screen

Code Sample:	`Plot1(xyLine, L`$_1$`, L`$_2$`,▫,BLUE)`
	`Send("READ RV.PATHLIST.Y")`
	`Get(L1)`
	`Send("READ RV.PATHLIST.X")`
	`Get(L2)`
	`DispGraph`

Range:	N/A
Describe:	READ RV.PATHLIST.Y - returns a list of Y values from the beginning to and including the current Waypoint Y value.
Result:	Return list of Y coordinates traversed since last **RV.PATH CLEAR** or initial **CONNECT RV**.
Type or Addressable Component:	Returns Data

RV.PATHLIST.TIME

Command:	RV.PATHLIST.TIME
Command Syntax:	**RV.PATHLIST.TIME**
Code Sample:	`Send "READ RV.PATHLIST.TIME"`
Range:	N/A
Describe:	READ RV.PATHLIST.TIME - returns a list of the time in seconds from the beginning to and including the current Waypoint time value.

Command:	RV.PATHLIST.TIME
Result:	Return list of cumulative travel times for each successive way-point.
Type or Addressable Component:	Returns Data

RV.PATHLIST.HEADING

Command:	RV.PATHLIST.HEADING
Command Syntax:	**RV.PATHLIST.HEADING**
Code Sample:	`Send "READ RV.PATHLIST.HEADING"`
Range:	N/A
Describe:	READ RV.PATHLIST.HEADING - returns a list of the headings from the beginning to and including the current Waypoint heading value.
Result:	Return list of cumulative angular headings taken.
Type or Addressable Component:	Returns Data

RV.PATHLIST.DISTANCE

Command:	RV.PATHLIST.DISTANCE
Command Syntax:	**RV.PATHLIST.DISTANCE**
Example:	Getting the cumulative distance traveled since the beginning of a journey by the RV
Code Sample:	`Send "READ RV.PATHLIST.DISTANCE"` Get(L_1) sum(L_1)
Range:	N/A

Command:	RV.PATHLIST.DISTANCE
Describe:	READ RV.PATHLIST.DISTANCE - returns a list of the distances traveled from the beginning to and including the current Waypoint distance value.
Result:	Return list of cumulative distances traveled.
Type or Addressable Component:	Returns Data

RV.PATHLIST.REVS

Command:	RV.PATHLIST.REVS
Command Syntax:	**RV.PATHLIST.REVS**
Code Sample:	`Send "READ RV.PATHLIST.REVS"`
Range:	N/A
Describe:	READ RV.PATHLIST.REVS - returns a list of the number of revolutions traveled from the beginning to and including the current Waypoint revolutions value.
Result:	Return list of wheel revolutions traveled.
Type or Addressable Component:	Returns Data

RV.PATHLIST.CMDNUM

Command:	RV.PATHLIST.CMDNUM
Command Syntax:	**RV.PATHLIST.CMDNUM**
Code Sample:	`Send "READ RV.PATHLIST.CMDNUM"`
Range:	N/A
Describe:	READ RV.PATHLIST.CMDNUM - returns a list of command numbers

Command:	RV.PATHLIST.CMDNUM
	for the path
Result:	Return list of commands used to travel to the current way-point entry. 0 - Start of Way-points (if first action is a STAY, then no START is given, but a STAY will be shown instead.) 1 - Travel forward 2 - Travel backward 3 - Left spin motion 4 - Right spin motion 5 - Left turn motion 6 - Right turn motion 7 - Stay (no motion) the time the RV stays at the current position is given in the TIME list. 8 - RV is currently in motion on this way-point traversal.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.X

Command:	RV.WAYPOINT.X
Command Syntax:	**RV.WAYPOINT.X**
Code Samples:	`Send("READ RV.WAYPOINT.X")`
Range:	N/A
Describe:	READ RV.WAYPOINT.X - returns x coordinate of current waypoint.
Result:	Return current way-point X coordinate.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.Y

Command:	RV.WAYPOINT.Y
Command Syntax:	**RV.WAYPOINT.Y**
Code Samples:	`Send("READ RV.WAYPOINT.Y")`
Range:	N/A
Describe:	READ RV.WAYPOINT.Y - returns x coordinate of current waypoint.
Result:	Return current way-point Y coordinate.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.TIME

Command:	RV.WAYPOINT.TIME
Command Syntax:	**RV.WAYPOINT.TIME**
Code Sample:	`Send("READ RV.WAYPOINT.TIME")`
Range:	N/A
Describe:	READ RV.WAYPOINT.TIME - returns time spent traveling from previous to current waypoint
Result:	Return total cumulative way-point travel time value in seconds.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.HEADING

Command:	RV.WAYPOINT.HEADING
Command	**RV.WAYPOINT.HEADING**

Command:	RV.WAYPOINT.HEADING
Syntax:	
Code Sample:	`Send("READ RV.WAYPOINT.HEADING")`
Range:	N/A
Describe:	READ RV.WAYPOINT.HEADING - returns absolute heading of current waypoint
Result:	Return current absolute heading in degrees. (+h = counter-clockwise, -h = clockwise.)
Type or Addressable Component:	Returns Data

RV.WAYPOINT.DISTANCE

Command:	RV.WAYPOINT.DISTANCE
Command Syntax:	**RV.WAYPOINT.DISTANCE**
Code Sample:	`Send("READ RV.WAYPOINT.DISTANCE")`
Range:	N/A
Describe:	READ RV.WAYPOINT.DISTANCE - returns distance traveled between previous and current waypoint
Result:	Return cumulative total distance traveled in meters.
Type or Addressable Component:	Returns Data

RV.WAYPOINT.REVS

Command:	RV.WAYPOINT.REVS
Command Syntax:	**RV.WAYPOINT.REVS**

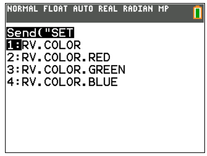
Command:	RV.WAYPOINT.REVS
Code Sample:	`Send("READ RV.WAYPOINT.REVS")`
Range:	N/A
Describe:	READ RV.WAYPOINT.REVS - returns number of revolutions needed to travel between previous and current waypoint
Result:	Return total revolutions of the wheels performed to travel the cumulative distance to the current way-point.
Type or Addressable Component:	Returns Data

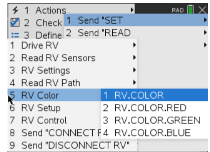
RV Color…

Send("SET Commands

RGB LED on Rover - This supports the same commands and parameters as the RGB LED on the TI-Innovator™ Hub.

- **RV Color…**
  - Send("SET
    - RV.COLOR
    - RV.COLOR.RED
    - RV.COLOR.GREEN
    - RV.COLOR.BLUE





RV.COLOR

Command:	**RV.COLOR**
Command Syntax:	**RV.COLOR**

Code Sample:	`Send "SET RV.COLOR`
	
	`[SET] RV.COLOR rr gg bb [[BLINK] b [[TIME] s.ss]]`

Range:	N/A
Describe:	Set the RGB color to be displayed on the Rover's RGB LED. Same syntax as for all RGB LED operations with COLOR, etc.
Result:	Return the current RGB color, as a three-element list, that is being displayed on the Rover's RGB LED
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV.COLOR.RED

Command:	**RV.COLOR.RED**
Command Syntax:	**RV.COLOR.RED**
Code	`Send "SET RV.COLOR.RED`

Command:	RV.COLOR.RED
Sample:	`[SET] RV.COLOR.RED rr [[BLINK] b [[TIME] s.ss]]`
Range:	N/A
Describe:	
Result:	Set the RED color to be displayed on the Rover's RGB LED.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV.COLOR.GREEN

Command:	RV.COLOR.GREEN
Command Syntax:	**RV.COLOR.GREEN**

Code Sample:	`Send "SET RV.COLOR.GREEN`
	
	`[SET] RV.COLOR.GREEN gg [[BLINK] b [[TIME] s.ss]]`

Range:	N/A
Describe:	
Result:	Set the GREEN color to be displayed on the Rover's RGB LED.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV.COLOR.BLUE

Command:	RV.COLOR.BLUE
Command Syntax:	**RV.COLOR.BLUE**

Command:	RV.COLOR.BLUE

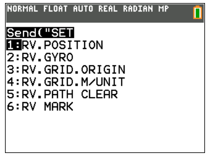
Code Sample:	`Send "SET RV.COLOR.BLUE`
	
	`[SET] RV.COLOR.BLUE bb [[BLINK] b [[TIME] s.ss]]`

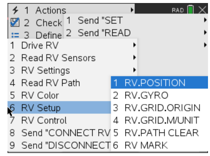
Range:	N/A
Describe:	
Result:	Set the BLUE color to be displayed on the Rover's RGB LED.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

RV Setup…

Send("SET Commands

- **RV Setup…**
  - Send("SET
    - RV.POSITION
    - RV.GYRO
    - RV.GRID.ORIGIN
    - RV.GRID.M/UNIT
    - RV.PATH CLEAR
    - RV MARK





RV.POSITION

Command:	**RV.POSITION**
Command Syntax:	**RV.POSITION**

Code Sample:	`Send "SET RV.POSITION"`
	
	`[SET] RV.POSITION xxx yyy`
	`[hhh [[DEGREES]|RADIANS|GRADIANS]]`

Range:	N/A
Describe:	Sets the coordinate position and optionally the heading of the Rover on the virtual grid.
Result:	Rover configuration is updated.
Type or Addressable Component:	Setting

RV.GYRO

Command:	**RV.GYRO**
Command Syntax:	**RV.GYRO**
Code Sample:	`Send "SET RV.GYRO"`

Command:	RV.GYRO
Range:	N/A
Describe:	Sets the on-board Gyroscope.
Result:	
Type or Addressable Component:	Control (for Gyroscope)

RV.GRID.ORIGIN

Command:	RV.GRID.ORIGIN
Command Syntax:	**RV.GRID.ORIGIN**

Code Sample:	`Send "SET RV.GRID.ORIGIN"`
	
	`[SET} RV.GRID.ORIGIN`

Range:	N/A
Describe:	Sets RV as being at current grid origin point of (0,0). The "heading" is set to 0.0 resulting in the current position of the RV now set to pointing down a virtual x-axis toward positive x values.
Result:	
Type or Addressable Component:	Setting

RV.GRID.M/UNIT

Command:	RV.GRID.M/UNIT
Command Syntax:	**RV.GRID.M/UNIT**

Code Sample:	`Send "SET RV.GRID.M/UNIT"`
	
	`[SET] RV.GRID.M/UNIT nnn`

Command:	RV.GRID.M/UNIT
Range:	N/A
Describe:	Set the size of a "grid unit" on the virtual grid. This setting is used by Rover when driving on the virtual grid. Default value is 0.1 (0.1M or 10 cm per grid unit). A value of 0.05 means 5 cm per unit grid. A value of 5 means 5M per grid unit. The maximum allowable value is 10.0 (for 10 meters per grid unit) and the lowest allowable value is 0.01 (for 1 cm per grid unit).
Result:	
Type or Addressable Component:	Setting

RV.PATH CLEAR

Command:	RV.PATH CLEAR
Command Syntax:	**RV.PATH CLEAR**

Code Sample:	`Send "SET RV.PATH CLEAR"`
	
	`[SET] RV.PATH CLEAR`

Range:	N/A
Describe:	Clears any pre-existing path / waypoint information. Recommended before doing a sequence of movement operations where waypoint / path-list information is desired.
Result:	
Type or Addressable Component:	Setting

RV MARK

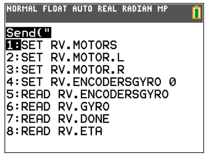
Command:	RV MARK
Command Syntax:	**RV MARK**
Code	`Send "SET RV MARK"`

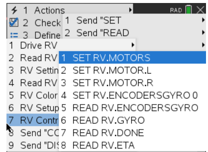
Command:	RV MARK
Sample:	`[SET] RV MARK [[TIME] s.ss]`
Range:	N/A
Describe:	Enable RV to make a "mark" with a pen at the specified time interval (default is 1 second if not specified). A time value of 0.0 turns OFF marking. Marking ONLY happens if the Rover is moving in a forward direction.
Result:	
Type or Addressable Component:	Setting (for Rover)

SEND(" Commands

Wheel commands and other commands relevant for learning foundations of the Rover vehicle.

- **RV Control …**
  - Send("
    - SET RV.MOTORS
    - SET RV.MOTOR.L
    - SET RV.MOTOR.R
    - SET RV.ENCODERSGYRO 0
    - READ RV.ENCODERSGYRO
    - READ RV.GYRO





SET RV.MOTORS

<table>
<tr><th>Command:</th><th>SET RV.MOTORS</th></tr>
<tr><td>Command Syntax:</td><td>SET RV.MOTORS</td></tr>
<tr><td>Code Sample:</td><td>Send "SET RV.MOTORS"

[SET] RV.MOTORS [LEFT][CW|CCW]
<pwm value|BRAKE|COAST>
[RIGHT][CW|CCW]
<pwm value|BRAKE|COAST>
[DISTANCE ddd [M|[UNITS]|REV|FT]]
| [TIME s.ss]</td></tr>
<tr><td>Range:</td><td>N/A</td></tr>
<tr><td>Describe:</td><td>Set left or right or both motor PWM values. Negative values imply CCW and Positive values imply CW. Left CW=backward motion. Left CCW=forward motion. Right CW=forward motion, Right CCW=backward motion. PWM values may be numeric from -255 to +255, or keywords "COAST" or "BRAKE". Value of 0 is stop (coast).
Use of the DISTANCE option is only available if the RV is connected with all sensors. CONNECT RV MOTORS means no sensors are available to measure distance, so the DISTANCE option is an error in this instance.</td></tr>
<tr><td>Result:</td><td>Both the LEFT and RIGHT motor, managed as a single object for</td></tr>
</table>

Command:	SET RV.MOTORS
	direct control (advanced) use.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

SET RV.MOTOR.L

Command:	SET RV.MOTOR.L
Command Syntax:	**SET RV.MOTOR.L**

Code Sample:	`Send "SET RV.MOTOR.L"`
	`[SET] RV.MOTOR.L [CW|CCW] <+/-pwm value|BRAKE|COAST>`
	`[TIME s.ss]
		[DISTANCE ddd [[UNITS]|M|REV|FT]]`

Range:	N/A
Describe:	Set left motor direct PWM value. **CCW** = forward, **CW** = backward, pwm value negative = forward, positive = backward. **TIME** option available in all modes, **DISTANCE** option available only when **RV** is fully connected (not the **RV MOTORS** option).
Result:	Left wheel motor and control for direct control (advanced) use.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

SET RV.MOTOR.R

Command:	SET RV.MOTOR.R
Command Syntax:	**SET RV.MOTOR.R**

Code Sample:	`Send "SET RV.MOTOR.R"`
	
	`[SET] RV.MOTOR.R [CW|CCW] <+/-pwm value|BRAKE|COAST>`
	`[TIME s.ss]
		[DISTANCE ddd [[UNITS]|M|REV|FT]]`

Command:	SET RV.MOTOR.R
Range:	N/A
Describe:	Set right motor direct PWM value. **CW** = forward, **CCW** = backward, pwm value positive = forward, negative = backward. **TIME** option available in all modes, **DISTANCE** option available only when **RV** is fully connected (not the **RV MOTORS** option).
Result:	Right wheel motor and control for direct control (advanced) use.
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

SET RV.ENCODERSGYRO 0

Command:	SET RV.ENCODERSGYRO 0
Command Syntax:	**SET RV.ENCODERSGYRO 0**
Code Sample:	`Send "SET RV.ENCODERSGYRO 0"`
Range:	N/A
Describe:	Reset the left and right encoder, coupled with the gyro and operating time information.
Result:	
Type or Addressable Component:	Control **Note:** This Rover control command is sent and executed in a queue.

READ RV.ENCODERSGYRO

Command:	READ RV.ENCODERSGYRO
Command Syntax:	**READ RV.ENCODERSGYRO**
Code Sample:	`Send "READ RV.ENCODERSGYRO"`

Command:	READ RV.ENCODERSGYRO
Range:	N/A
Describe:	The left and right encoder, coupled with the gyro and operating time information.
Result:	List of values of current left and right encoder, coupled with gyro and operating time information
Type or Addressable Component:	Control **Note:** This Rover READ command is executed immediately.

READ RV.GYRO

Command:	READ RV.GYRO
Command Syntax:	**READ RV.GYRO**

Code Sample:	`Send "READ RV.GYRO"`
	
	`READ RV.GYRO [[DEGREES]|RADIANS|GRADIANS]`

Range:	N/A
Describe:	The gyroscope is used to maintain the heading of Rover while it's in motion. It can also be used to measure the change in angle during turns. The gyroscope is ready to use after the **CONNECT RV** command is processed. The GYRO object shall be usable even when the RV is not in motion.
Result:	Returns current gyro sensor angular deviation from 0.0, reading partially drift-offset compensated.
Type or Addressable Component:	Control **Note:** This Rover READ command is executed immediately.

READ RV.DONE

Command:	READ RV.DONE
Command Syntax:	**READ RV.DONE**
Code Sample:	`Send("READ RV.DONE")`
Example:	**RV.DONE** as an alias for **RV.WAYPOINT.CMDNUM**

Code Sample:	`For n,1,16`
	`Send "RV FORWARD 0.1"`
	`Send "RV LEFT"`
	`EndFor`
	`@ Wait for Rover to finish driving`
	`Send "READ RV.DONE"`
	`Get d`
	`While d=0`
	`Send "READ RV.DONE"`
	`Get d`
	`Wait 0.1`
	`EndWhile`
	`Send "READ RV.PATHLIST"`
	`Get L`

Range:	N/A
Describe:	**RV.DONE** as an alias for **RV.WAYPOINT.CMDNUM** To improve usability a new state variable was created called **RV.DONE**. This is an alias of **RV.WAYPOINT.CMDNUM**.
Result:	
Type or Addressable Component:	Returns Data

See Also: RV.WAYPOINT.CMDNUM

READ RV.ETA

Command:	READ RV.ETA
Command Syntax:	**READ READ RV.ETA**
Code Sample:	`Send("READ RV.ETA")`
Example:	The code sample below returns the estimated time to drive to coordinate (4,4)

Code Sample:	`Send "RV TO XY 4 4"`
	`Send "READ RV.ETA"`
	`Get eta`
	`Disp eta`

	Note: This value will not be exact. It will depend on the surface for one, but it will be a close enough estimate for the expected applications. The value will be time in seconds with a minimum unit of 100 ms.
Example	If a different **READ** command is issued, the value of the variable is overwritten with the information that was requested.

Code Sample:	`Send "RV TO XY 3 4"`
	`Send "READ BRIGHTNESS"`
	`Get eta`

	Note: eta - will contain the value of the **BRIGHTNESS** sensor, not the **RV.ETA** variable
Range:	N/A
Describe:	Calculate the estimated time to complete each Rover command.
Result:	
Type or Addressable Component:	Returns Data

Sample program:

Set **RGB** to red while moving forward, green when turning.

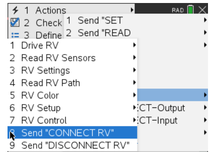
Code Sample:	`For n, 1, 4`
	`Send "RV FORWARD"`
	`Send "READ RV.ETA"`
	`Get eta`
	`Send "SET COLOR 255 0 0"`
	`Wait eta`
	`Send "RV LEFT"`
	`Send "READ RV.ETA"`
	`Get eta`
	`Send "SET COLOR 0 255 0"`
	`Wait eta`
	`EndFor`

Send "CONNECT RV"

SEND("CONNECT RV") Commands

CONNECT RV - initializes the hardware connections.

- Connects RV and inputs and outputs built into the RV.
- Resets the Path and the Grid Origin.
- Sets the units per meter to default value.
- **Send("CONNECT RV")**



CONNECT RV

Command:	**CONNECT RV**
Command Syntax:	CONNECT RV [MOTORS]
Code Sample:	`Send "CONNECT RV"` `Send "CONNECT RV MOTORS"`
Range:	N/A
Describe:	The "**CONNECT RV**" command configures the TI-Innovator™ Hub software to work with the TI-Innovator™ Rover. It establishes the connections to the various devices on the Rover – two motors, two encoders, one gyroscope, one RGB LED and one color sensor. It also clears the various counters and sensor values. The optional 'MOTORS' parameter configures only the motors and allows direct control of motors without the additional peripherals.
Result:	Connects the Rover Vehicle to the TI-Innovator™ Hub. This establishes connections with the motor driver, color sensor, gyroscope, ultrasonic ranger, and RGB LED. The Rover is now ready to be programmed
Type or Addressable Component:	All components of the Rover - two motors, two encoders, one gyroscope, one RGB LED and one color sensor.

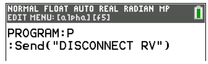
Send "DISCONNECT RV"

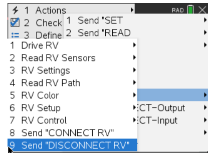
SEND("DISCONNECT RV") Commands

DISCONNECT RV - disconnects all the hardware peripherals from the Hub.

Format: Send("DISCONNECT RV")

- **Send("DISCONNECT RV")**





DISCONNECT RV

Command:	**DISCONNECT RV**
Command Syntax:	DISCONNECT RV

Code Sample:	`Send "DISCONNECT RV"`
	
	`DISCONNECT RV`

Range:	N/A
Describe:	The "**DISCONNECT RV**" command removes the logical connections between the TI-Innovator™ Hub and the TI-Innovator™ Rover. It also clears the counters and sensor values. It allows the use of the breadboard port of the TI-Innovator™ Hub with other devices.
Result:	The TI-Innovator™ Hub is now logically disconnected from the TI-Innovator™ Rover
Type or Addressable Component:	N/A

TI-Innovator™ Rover – Programmable Component Data Sheets

The TI-Innovator™ Rover Programmable Component Data Sheets include the following; a product name or number, a brief description, a product image, specifications, how the component connects to the TI-Innovator™ Hub, and Rover commands with simple code samples.

Device

Device	Category
Rover (RV)	Accessory

Sensors

Sensors	Category
Rotary Encoders	Motion and Distance Sensor
Gyroscope	Motion and Distance Sensor
Ultrasonic Ranger	Motion and Distance Sensor
Color Sensor	Environmental Sensor
On-Board Light Brightness Sensor (On Hub)	Environmental Sensor

Controllable Devices

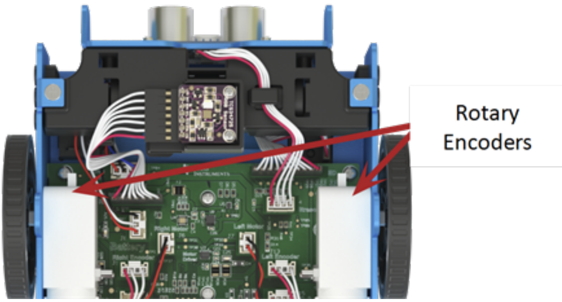
Controllable Devices	Category
Electric Motors	Motors
RGB (Red-Green-Blue) LED	LEDs and Displays
On-Board Speaker (on Hub)	Sound Output

TI-Innovator™ Rover

Title	**TI-Innovator™ Rover Data Sheet**
TI Item Name	TI-Innovator™ Rover
Quantity	1
Included in	TI-Innovator™ Rover
Description	**TI-Innovator™ Rover** is a two-wheeled programmable robotic vehicle which works with the TI-Innovator™ Hub with TI LaunchPad™ Board.
Category	Accessory
Hub Connection	**See:** Connecting TI-Innovator™ Rover
Assembly Instructions	**See:** Exploring the Assembled TI-Innovator™ Rover
Precautions	**See:** General Precautions
Specifications	**See:** TI-Innovator™ Rover Setup Requirements

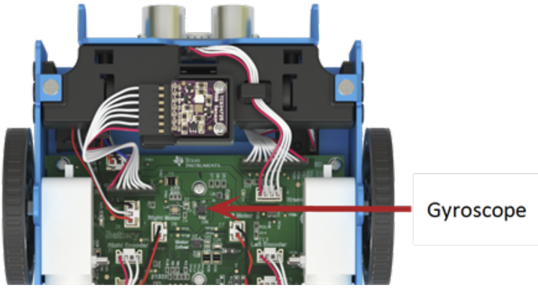
Rover Commands	
Sketch Object	RV
Command Syntax	
Code Sample:	<table><tr><th>Desired Action</th><th>Code Sample</th></tr><tr><td>Configure Hub for additional commands such as: RV Forward 2 RV Left</td><td><code>Send "CONNECT RV"</code></td></tr></table>

TI-Innovator™ Rover On-Board Rotary Encoders Data Sheet



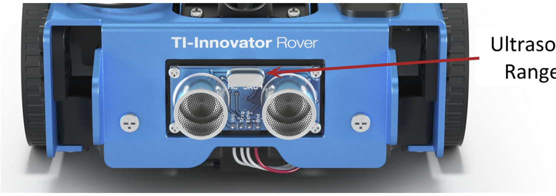
Title	**TI-Innovator™ Rover Rotary Encoders**
TI Item Name	Built into the TI-Innovator™ Rover
Quantity	2 - 1 for each wheel
Included in	TI-Innovator™ Rover
Description	Calculates linear distance by detecting how many rotations the wheels make as the Rover moves. Assists in balancing and aligning the wheels.
Category	Motion and Distance Sensors
Hub Connection	on-board Rover
Assembly Instructions	Not Applicable
Precautions	Do not unscrew the case enclosure. Encoder has sharp edges that should not be exposed.
Specifications	Not Applicable

TI-Innovator™ Rover On-Board Gyroscope Data Sheet



Title	TI-Innovator™ Rover Gyroscope
TI Item Name	Built into the TI-Innovator™ Rover
Quantity	1
Included in	TI-Innovator™ Rover
Description	Calculates angular displacement and heading as it maintains orientation.
Category	Motion and Distance Sensors
Hub Connection	on-board Rover
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

TI-Innovator™ Rover On-Board Ultrasonic Ranger Data Sheet



Title	Ultrasonic Ranger
TI Item Name	Built into the Rover
Quantity	1
Included in	TI-Innovator™ Rover
Description	Non-contact measurement module that reads distance from obstacle in meters.
Category	Motion and Distance Sensors
Hub Connection	On-board the Rover
Assembly Instructions	Not Applicable
Precautions	
Specifications	Measures distances up to 4 m

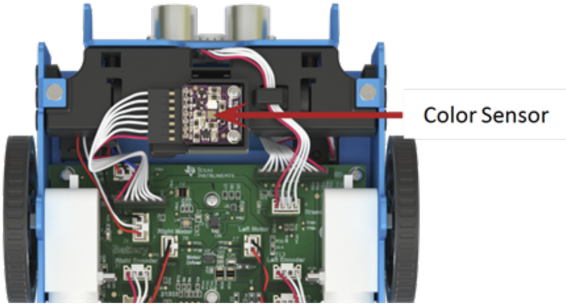
Rover Commands	
Sketch Object	RV.RANGER
Command Syntax	Send("READ RV.RANGER")

Code Sample:

Desired Action	Code Sample
Connects the Rover to the TI-Innovator Hub. This establishes connections with the motor driver, color sensor, gyroscope, ultrasonic ranger, and proximity sensors.	`CONNECT RV`
Returns the current distance from the	`READ RV.RANGER` `Get (R)`

	Desired Action	Code Sample
	front of the Rover to an obstacle. If there is no obstacle detected, a range of 10.00 meters is reported	

TI-Innovator™ Rover On-Board Color Sensor Data Sheet



Title	**TI-Innovator™ Rover Color Sensor**
TI Item Name	Built into the TI-Innovator™ Rover
Quantity	1
Included in	TI-Innovator™ Rover
Description	Bottom-mounted color sensor detects the color of the surface. Can also detect gray-level scale of black (0) to white (255). Measures surface color. Used to identify colors and execute Rover Hub commands based on color.
Category	Environmental Sensors
Hub Connection	on-board Rover
Assembly Instructions	Not Applicable
Precautions	Do not unplug the cable. If it becomes unattached see proper positioning as shown above.
Specifications	Not Applicable

Rover Commands	
Sketch Object	RV.COLORINPUT RV.COLORINPUT.RED RV.COLORINPUT.GREEN RV.COLORINPUT.BLUE RV.COLORINPUT.GRAY
Command Syntax	

Code Sample:		
	Desired Action	**Code Sample**
		`Send "READ RV.COLORINPUT.RED"` `Get (C)`

On-Board Light Brightness Sensor Data Sheet

Light Brightness Sensor

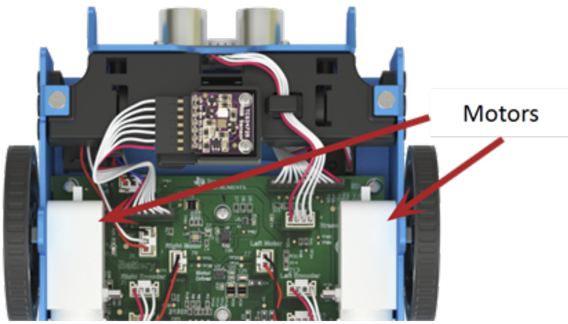
Title	**On-Board Light Brightness Sensor**
TI Item Name	Built into the Hub
Quantity	1
Included in	TI-Innovator™ Hub
Description	Built-in light brightness sensor located at the bottom of the Hub. The sensor detects light intensity.
Category	Environmental Sensors
Hub Connection	on-board Hub
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

HUB Commands	
Sketch Object	BRIGHTNESS
Command Syntax	Send("READ BRIGHTNESS")

Code Sample:

Desired Action	**Code Sample**
Read the built-in light brightness sensor	`Send("READ BRIGHTNESS")` `Get(B)`

TI-Innovator™ Rover On-Board Electric Motors Data Sheet



Title	**TI-Innovator™ Rover Motors**
TI Item Name	Built into the TI-Innovator™ Rover
Quantity	2 – 1 on each wheel with electric motor and rotary encoder to track rotations.
Included in	TI-Innovator™ Rover
Description	Motors that can be programmed to move the wheels independently and at variable speeds.
Category	Motors
Hub Connection	on-board Rover
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

Rover Commands	**`Send "SET RV.MOTORS`**
Sketch Object	RV.MOTORS
Command Syntax	

Code Sample:

Desired Action	Code Sample
Direct control of	`Send "SET RV.MOTORS"`

Rover Commands	Send "SET RV.MOTORS

Desired Action	Code Sample
motors.	[SET] RV.MOTORS [LEFT] [CW|CCW] <pwm value|BRAKE|COAST> [RIGHT][CW|CCW] <pwm value|BRAKE|COAST> [DISTANCE ddd [M|[UNITS]|REV|FT]] | [TIME s.ss]

TI-Innovator™ Rover On-Board RGB (Red-Green-Blue) LED Data Sheet

Title	**TI-Innovator™ Rover RGB (Red-Green-Blue) LED**
TI Item Name	Built into the TI-Innovator™ Rover
Quantity	1
Included in	TI-Innovator™ Rover
Description	Light-emitting diode with independently adjustable red, green and blue elements. Can produce a wide variety of colors.
Category	LEDs and Displays
Hub Connection	on-board Rover
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

Rover Commands	
Sketch Object	RV.COLOR
Command Syntax	

Code Sample:

Desired Action	Code Sample
Configure LED **Note:** RV.COLOR supports the same functions	`Send("SET RV.COLOR 255 0 255")`

	Desired Action	Code Sample
	as the Hub COLOR object	

On-Board Speaker Data Sheet

Speaker (at back of Hub) is addressable as "SOUND" in Hub command strings.

Title	**On-Board Speaker**
TI Item Name	Built into the Hub
Quantity	1
Included in	TI-Innovator™ Hub
Description	Built-in speaker located at the back of the Hub. It converts electrical current into sound you can hear.
Category	Sound Output
Hub Connection	on-board Hub
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Not Applicable

HUB Commands	
Sketch Object	SOUND
Command Syntax	Send("SET SOUND ...") Frequency in Hz or Note as C1, CS1, D2, ... [TIME duration in seconds]

Code Sample:

Desired Action	**Code Sample**
Play tone at 261.23 Hz	`Send("SET SOUND 261.23")`
Evaluate the expression 2^8 (= 256) and play that tone	`Send("SET SOUND eval (2^8)")`

Desired Action	Code Sample
Evaluate the expression 2^8 (= 256) and play that tone for .25 seconds	`Send("SET SOUND eval (2^8) TIME .25")`
Evaluate the expression 2^9 (= 512) and play that tone for 0.25 seconds (result of evaluating 1/4)	`Send("SET SOUND eval (2^9) TIME eval(1/4)")`
Turn speaker off	`Send("SET SOUND OFF")`

I/O Modules Data Sheets

The TI-Innovator™ I/O Module Data Sheets include the following; a product name and number, a brief description, a product image, specifications, how the component connects to the TI-Innovator™ Hub, and Hub commands with simple code samples.

Topic Links

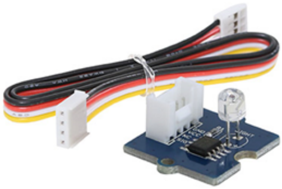
- Analog Light Sensor Data Sheet
- Vibration Motor Data Sheet
- White LED Data Sheet
- Servo Motor Data Sheet
- Ultrasonic Ranger Data Sheet

Environmental Sensors

Topic Links

- Analog Light Sensor Data Sheet
- Moisture Sensor Data Sheet
- Temperature Sensor
- Temperature & Humidity Sensor Data Sheet
- Water Pump Data Sheet

Analog Light Sensor Data Sheet



Title	Analog Light Sensor
TI Item Name	STEMKT/AC/A
Quantity	1
Included in	TI-Innovator™ I/O Module Pack
Description	Sensor that detects the light intensity of the environment.
Category	Environmental Sensors
Hub Connection	4-Pin Cable to any of these ports: IN 1, IN 2, IN 3
Assembly Instructions	Not applicable
Precautions	Light sensor leads may break if bent repeatedly
Specifications	Maximum voltage: 150, Maximum power: 100, Environmental Temp: -30~+70, Spectrum Peak Value: 540

HUB Commands	
Sketch Object	LIGHTLEVEL
Command Syntax	Send("READ LIGHTLEVEL n")

Code Sample:

Desired Action	Code Sample
Configure the program to use LIGHTLEVEL on port **IN 1**	`Send("CONNECT LIGHTLEVEL 1 TO IN 1")`
Read the light sensor	`Send("READ LIGHTLEVEL 1")` `Get(L)`

Moisture Sensor Data Sheet

Title	**Moisture Sensor**
TI Item Name	STEMKT/AC/MM/A
Description	Detects the moisture of soil, and measures the dampness around the sensor. It can be used to decide if the plants in a garden need watering. **See Also:** Grove - Moisture Sensor (SKU 101020008) https://www.seeedstudio.com/depot/Grove-Moisture-Sensor-p-955.html
Category	Environmental Sensors
Hub Connection	4-Pin Cable to any of these ports: IN 1, IN 2, IN 3
Assembly Instructions	
Precautions	This sensor is not hardened against contamination or prolonged exposure to water and may be prone to electrolytic corrosion across the probes. This effect will be mitigated using the 3.3V of IN 1 and IN 2.
Specifications	Operating voltage: 3.3~5V, Operating current: 35mA, Sensor Output Value in dry soil: 0~ 300, Sensor Output Value in humid soil: 300~700, Sensor Output Value in water: 700 ~ 950, PCB size: 2.0cm X 6.0cm, Operating voltage: 3.3~5V, Operating current: 35mA, Sensor Output Value in dry soil: 0~ 300, Sensor Output Value in humid soil: 300~700 These output values are nonsensical. They may be for a 10 bit ADC.

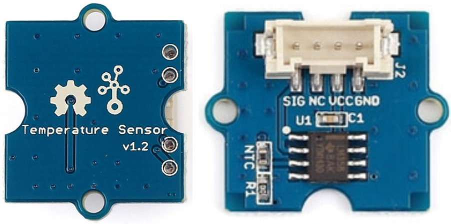
HUB Commands	
Sketch Object	MOISTURE
Command Syntax	
Code Sample:	

Desired Action	**Code Sample**
Connect moisture sensor to **IN 1**	`Send "CONNECT MOISTURE 1 IN 1"`

HUB Commands

Desired Action	Code Sample
Configure the measurement range to be between 0 and 100. The range is an index and has no units.	`Send "RANGE MOISTURE 1 0 100"`
Read the sensor	`Send "READ MOISTURE 1"` `Get moisture`

Temperature Sensor Data Sheet



Title	**Temperature Sensor**
TI Item Name	STEMKT/AC/F
Description	Uses a thermistor to measure ambient temperature. The resistance of thermistor changes based on ambient temperature. This resistance value alters the output of a voltage divider which is measured by the TI-Innovator™ Hub and converted to a temperature value in centigrade degrees. The operating range is -40 to 125°C , with an accuracy of 1.5°C . This sensor is not waterproof and cannot be submerged. **See Also:** Grove Temperature Sensor (SKU 101020015) https://www.seeedstudio.com/Grove-Temperature-Sensor-p-774.html
Category	Environmental Sensors
Hub Connection	4-Pin Cable to any of these ports: IN 1, IN 2, IN 3
Assembly Instructions	
Precautions	
Specifications	Operating voltage: 3.3~5V Thermistor Zero power resistance: 100 KΩ Resistance Tolerance: ±1% Thermistor: NCP18WF104F03RC (NTC) Nominal B-Constant: 4250 ~ 4299K Operating temperature range: -40°C to 125°C Accuracy: ± 1.5°C

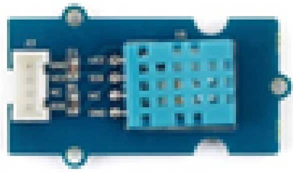
HUB Commands	
Sketch Object	TEMPERATURE
Command Syntax	

HUB Commands

Code Sample:

Desired Action	**Code Sample**
Connected the temperature sensor to **IN 1** port	`Send "CONNECT TEMPERATURE 1 TO IN 1"`
Read the temperature value from the sensor in centigrade degrees	`Send "READ TEMPERATURE 1"` `Get t`

Temperature & Humidity Sensor Data Sheet

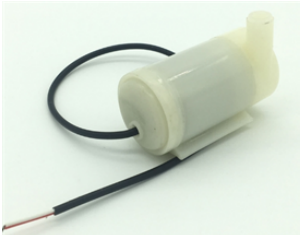


Title	**Temperature & Humidity Sensor**
TI Item Name	STEMKT/AC/HT/A
Description	Temperature & Humidity Sensor that measures percent relative humidity and the temperature in centigrade degrees. **See Also:** Grove - Temperature & Humidity Sensor (SKU 101020011) https://www.seeedstudio.com/depot/Grove-TempHumi-Sensor-p-745.html
Category	Environmental Sensors
Hub Connection	4-Pin Cable to any of these ports: IN 1, IN 2, IN 3
Assembly Instructions	
Precautions	
Specifications	Input Voltage: 3.3V & 5V Measuring Current: 1.3 - 2.1 mA Measuring Humidity Range: 20% - 90% RH Measuring Temperature Range: 0 - 50 ℃ degrees centigrade

HUB Commands	
Sketch Object	DHT
Command Syntax	Sensor may not report correct readings during initial warm up period.
Code Sample:	(see table below)

Desired Action	**Code Sample**
Connect the **DHT** sensor to port **IN 2**	`Send "CONNECT DHT 1 TO IN 2 "`
Read the temperature from the **DHT** sensor	`Send "READ DHT 1 TEMPERATURE"` `Get temperature`
Read the humidity from the **DHT** sensor	`Send "READ DHT 1 HUMIDITY"` `Get humidity`

Water Pump Data Sheet



Title	**Water Pump**
TI Item Name	STEMKT/AC/WP/A
Description	The water pump is used in projects that use flowing water for irrigation etc. **See Also:** Water Pump - JT-DC3W
Category	Environmental Sensors
Hub Connection	Connects to TI-Innovator™ Hub through a MOSFET module
Assembly Instructions	
Precautions	
Specifications	Submersible Pump Plastic tubing: 18" length Wires: 18" length

HUB Commands	**See MOSFET commands**
Sketch Object	N/A It is controlled through a MOSFET module.
Command Syntax	
Code Sample:	

Desired Action	**Code Sample**

LEDs and Displays

xxx

Blue LED

Green LED

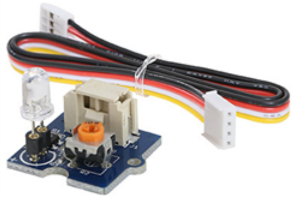
Multi Color Flash LED (5mm)

Purple LED (3mm)

Red LED

Variable Color LED

White LED Data Sheet



Title	**White LED**
TI Item Name	STEMKT/AC/C
Quantity	1
Included in	TI-Innovator™ I/O Module Pack
Description	White LED module that bends to any position.
Category	LEDs and Displays
Hub Connection	4-Pin Cable to any of these ports: OUT 1, OUT 2, OUT 3
Assembly Instructions	Insert LED into socket - longer leg (lead) is positive (anode). If both leads are of equal length, the lead adjacent to the flat edge on LED casing is the negative (cathode) lead.
Precautions	Do not bend the leads repeatedly; this will weaken the wires and may cause them to break.
Specifications	Operating voltage: 3.3v/5v, Emitting Color: White

HUB Commands	
Sketch Object	LED
Command Syntax	Send("SET LED 1 TO ON/OFF [[BLINK|TOGGLE] frequency] [[TIME] seconds]")
Code Sample:	

Desired Action	**Code Sample**
Configure the program to use LED on port **OUT 1**	`Send("CONNECT LED 1 TO OUT 1")`
Turn LED ON	`Send("SET LED 1 ON")`
Turn LED OFF	`Send("SET LED 1 OFF")`

Desired Action	Code Sample
Turn external LED ON for 5 seconds	`Send("SET LED 1 TO ON TIME 5")`
Turn external LED ON and blink it at 2 Hz (2 times a second) for 5 seconds	`Send("SET LED 1 TO ON BLINK 2 TIME 5")`

Motion and Distance Sensors

Topic Links

- Magnetic field (Hall Effect) Sensor Data Sheet
- Ultrasonic Ranger Data Sheet

Magnetic field (Hall Effect) Sensor Data Sheet

Title	**Hall Sensor**
TI Item Name	STEMKT/AC/HS/A
Description	Measures the magnetic field around the sensor using the Hall Effect. The sensor reports a low value in the presence of a magnetic field and a high value in the absence of one. It can be used to detect when a magnet is close to the sensor. **See Also:** Grove - Hall Sensor (SKU 101020046) https://www.seeedstudio.com/depot/Grove-Hall-Sensor-p-965.html
Category	Motion and Distance Sensors
Hub Connection	4-Pin Cable to any of these ports: IN 1, IN 2, IN 3
Assembly Instructions	
Precautions	
Specifications	Dimensions: 130mm x 90mm x 9.5mm Weight: G.W 6g

HUB Commands	
Sketch Object	ANALOG.IN
Command Syntax	

Code Sample:

Desired Action	Code Sample
Connect the Hall effect sensor to **IN3** port	`Send "CONNECT ANALOG.IN 1 TO IN 3"`

HUB Commands

Desired Action	Code Sample
Read the value of the magnetic field reported by the sensor	`Send "READ ANALOG.IN 1"` `Get m`

Ultrasonic Ranger Data Sheet

Title	Ultrasonic Ranger
TI Item Name	STEMKT/AC/E
Quantity	1
Included in	TI-Innovator™ Ultrasonic Ranger Module
Description	Non-contact measurement module that reads distance from sensor in meters.
Category	Motion and Distance Sensors
Hub Connection	4-Pin Cable to any of these ports: IN 1, IN 2
Assembly Instructions	Not Applicable
Precautions	Sold separately, not included in the I/O Modules Pack
Specifications	Operating voltage: 3.3~5V, Operating current: 15mA, Ultrasonic frequency: 42kHz, Measuring range: 3-400m, Resolution: 1m, Output: PWM

HUB Commands	
Sketch Object	RANGER
Command Syntax	Send("READ RANGER n") Get(R)
Code Sample:	

Desired Action	Code Sample
Configure the program to use RANGER on port **IN 1**	`Send("CONNECT RANGER 1 TO IN 1")`
Read the Ultrasonic Ranger	`Send("READ RANGER 1")` `Get(R)`

Motors

xxx

Servo

Servo Motor Data Sheet

There are two types of Servo motors, continuous and sweep. More about SERVO Motors

Title	**Servo Motor**
TI Item Name	STEMKT/AC/D
Quantity	1
Included in	TI-Innovator™ I/O Module Pack
Description	360 degree, continuous rotation servo motor with gearing and feedback system; used in driving mechanism of robots.
Category	Motors
Hub Connection	4-Pin Cable to only this port: OUT 3
Assembly Instructions	Mount a gear to the top of the Servo Motor using one of the provided screws.
Precautions	Use an Auxiliary Power Source. Do not hold the Servo Motor's shaft while it is rotating. Also, do not rotate the Servo Motor by hand.
Specifications	Operating Speed: 110RPM (4.8V), 130RPM (6V); Stall Torque: 1.3kg.cm/18.09oz.in (4.8V), 1.5kg.cm/20.86oz.in (6V); Operating Voltage: 4.8V~6V

HUB Commands	
Sketch Object	SERVO
Command Syntax	Send("SET SERVO n TO [CW/CCW] speed [[TIME] seconds] -- speed from -100 to 100, CW/CCW (Clockwise/Counterclosewise) optional, if speed <0, CCW, else CW unless CW/CCW keyword is specified, TIME optional, in seconds, default=1 second (for continuous servo operation) (CW/CCW required if TIME/seconds NOT specified.)

Code Samples:

Desired Action for the SERVO.CONTINOUS	**Code Sample**
Configure the program to use **SERVO** on port **OUT 3**	`Send("CONNECT SERVO 1 TO OUT 3")`
Set **SERVO** to turn Counterclockwise (CCW) at full (100%) speed for 2 seconds	`Send("SET SERVO 1 CCW 100 2")`
Set **SERVO** to turn Clockwise (CW) at half (50%) speed for 1 second (default time if not specified)	`Send("SET SERVO 1 CW 50")`

Turn **SERVO** Off	`Send("SET SERVO 1 ZERO")`
	or
	`Send("SET SERVO 1 STOP")`

More about SERVO Motors

There are two types of Servo motors, **continuous** and **sweep**.

- The motor in the I/O pack is a continuous motor. This motor rotates either clockwise (positive direction) or counter clockwise (negative direction) for a specified amount of time.
- Sweep servo motors turn only 90 degrees in each direction.

It is necessary to connect servo motors to the OUT 3 port. This port provides the 5 volts needed for these motors.

- You will also need to plug an external battery into the PWR micro USB port.
- You will also give a number, in this case 1, as part of identifying to an external device plugged into a port. (**Note:** In some cases you may have multiple devices of the same type connected to the TI-Innovator (e.g. temperature probes))

Code Sample:	`Send("CONNECT SERVO.CONTINUOUS 1 TO OUT 3 ")`

The first argument sets the speed and direction of the motor. You can use -100 to 100. Negative values are in the counter clockwise direction. Positive values are in the clockwise direction. Zero stops the motor.

The second argument is the amount of time in seconds to run the motor in the specified direction and speed. If there is no argument for time, then the motor will turn for one second.

The Wait command is used in the case to delay execution of commands on the calculator until the motor action is complete. This optional command is useful in delaying commands that might replace the current motor command before the desired end time.

Code Sample:

```
Send("SET SERVO.CONTINUOUS 1 ¬50 TIME 2")
Wait 2
```

You also have the options of using explicit CW and CCW settings with values for speed of 0 to 100.

Code Sample:

```
Send("SET SERVO.CONTINUOUS 1 CW 100 TIME
3")
Wait 3
```

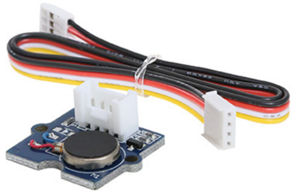
This example prompts for the inputs to the **SERVO.CONTINUOUS** command.

The example also uses a While loop with getKey as a way for the user to control when the command to stop the motor is executed.

Code Sample:

```
ClrHome
Disp "S:SPEED AND DIRECTION"
Disp "ENTER VALUES ¬100 TO 100 FOR S"
Disp "T:TIME IN SECONDS"
Prompt S,T
Send("SET SERVO.CONTINUOUS 1 eval(S) TIME
eval(T)")
Disp "PRESS 1 TO STOP THE MOTOR AND END THE
PROGRAM"
0→K
While K≠92
getKey→K
End
Send("SET SERVO.CONTINUOUS 1 0”)
```

Vibration Motor Data Sheet



Title	Vibration Motor
TI Item Name	STEMKT/AC/B
Quantity	1
Included in	TI-Innovator™ I/O Module Pack
Description	Coin type motor that vibrates when the input logic is HIGH.
Category	Motors
Hub Connection	4-Pin Cable to any of these ports: OUT 1, OUT 2, OUT 3
Assembly Instructions	Not Applicable
Precautions	Use an Auxiliary Power Source
Specifications	Operate Voltage: 3.0V to 5.5V, Control Mode: Logic Level (When Logic HIGH, the motor is ON. When LOW, the motor is OFF.), Rated speed: 9000 rpm

HUB Commands	
Sketch Object	VIB.MOTOR
Command Syntax	Send("SET VIB.MOTOR 1 TO pwm") - pwm from 0 to 255

Code Sample:

Desired Action	Code Sample
Configure the program to use ANALOG.OUT on port **OUT 1**	`Send("CONNECT VIB.MOTOR 1 TO OUT 1")`
Turn OFF vibration motor	`Send("SET VIB.MOTOR 1 TO 0")`
Turn ON vibration	`Send("SET VIB.MOTOR 1`

HUB Commands

Desired Action	Code Sample
motor at full power	`TO 255")`
Turn ON vibration motor at half power	`Send("SET VIB.MOTOR 1 TO 128")`

Power and Signal Sensors

Topic Links

- MOSFET

MOSFET Data Sheet

Title	**MOSFET**
TI Item Name	STEMKT/AC/MOSFET/A
Description	Enables you to control higher power projects with the TI-Innovator™ Hub. It is typically used to control DC motors and pumps. It cannot be used with AC power supplies. The **MOSFET** allows the connection of an external power source like AA batteries to power the motor and enables the TI-Innovator™ Hub to control the speed of the motor. This allows the TI-Innovator™ Hub to control a high power device without directly providing power to the device. The power supply positive lead is connected to the (+) screw terminal and the negative lead to the (-) screw terminal. The device positive lead is connected to the OUT screw terminal and the device negative is connected to the GND screw terminal. **See Also:** Grove - MOSFET (SKU 103020008) https://www.seeedstudio.com/Grove-MOSFET-p-1594.html
Category	
Hub Connection	Working voltage: 5V, Vin: 5 ~ 15V MOSFET Model: CJQ4435
Assembly Instructions	
Precautions	Indirect pin support.
Specifications	

HUB Commands	
Sketch Object	
Command Syntax	The **MOSFET** may be connected to OUT 1, OUT 2, or OUT 3. However, the device will not be shut off completely when using OUT 3. It is recommended to avoid using OUT 3.
Code Sample:	

Desired Action	Code Sample
Connect the **MOSFET** to the **OUT 1** port	`Send "CONNECT ANALOG.OUT 1 TO OUT 1"`
Control the connected motor/pump at 50% speed for 3 seconds	`Send "SET ANALOG.OUT 1 128 TIME 3"`

TI-Innovator™ Breadboard Data Sheets

The TI-Innovator™ Breadboard Data Sheets include the following; a product name and number, a brief description, a product image, specifications, how the component connects to the TI-Innovator™ Hub, and simple code samples.

Topic Links

- Breadboard Components and Usable Pins
- Environmental Sensors
- LEDs and Displays
- Motors
- Power and Signal Control
- Passive Components

Breadboard Components and Usable Pins

Here is a list of all the components in our breadboard pack and the usable pins for each component.

Component	Use with pins
1 Breadboard	N/A
10 Male/Female Breadboard Jumper Cables	N/A
40 Male/Male Breadboard Jumper Cables	N/A
5 Green LED	BB 1-10
10 Red LED	BB 1-10
2 RGB (Red-Green-Blue) LED	BB 8-10
10 Resistor 100 Ohm	N/A
10 Resistor 1K Ohm	N/A
10 Resistor 10K Ohm	N/A
10 Resistor 100K Ohm	N/A
10 Resistor 10M Ohm	N/A
1 Diode	BB 1-10
1 Thermistor	BB 5,6,7 (analog input required)
1 SPDT Slide Switch	BB 1-10
1 8 Position SIP DIP Switch	BB 1-10 (digital input)
1 8 100 Ohm Resistor SIP	N/A
1 Potentiometer with Knob	BB 5,6,7
1 Capacitor 100µF	N/A
1 Capacitor 10µF	N/A
1 Capacitor 1µF	N/A
1 7-Segment Display	BB 1-10
1 Small DC Motor	BB 1-10 (uses digital to generate software PWM)
2 TTL Power MOSFET	BB 1-10
1 TI Analog Temperature Sensor	BB 5,6,7 (analog input required)
1 Visible Light Sensor	BB 5,6,7 (analog input required)
1 4-AA Battery Holder	N/A

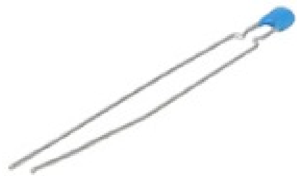
Component	Use with pins
1 Infrared Receiver	BB 1-10 (digital input)
1 Infrared Transmitter	BB 1-10 (digital output)

Environmental Sensors

Topic Links

- Thermistor Data Sheet
- TI Analog Temperature Sensor Data Sheet
- Visible Light Sensor Data Sheet

Thermistor Data Sheet



Title	**Thermistor**
TI Item Name	STEMEE/AC/THERM/A
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	Resistor whose resistance changes based on temperature. Used for measurement and control.
Category	Environmental Sensors
Hub Connection	breadboard circuit
Assembly Instructions	No polarity
Precautions	Not Applicable
Specifications	Resistance in Ohms @ 25°C: 10k, Resistance Tolerance: ±1%, B Value Tolerance: ±1%, Operating Temperature: -40°C ~ 125°C, Power – Max: 7.5mW

HUB Commands	
Sketch Object	THERMISTOR
Command Syntax	Send("READ THERMISTOR n")

Code Sample:

Desired Action	Code Sample
Configure the program to use THERMISTOR on pin BB 1	`Send("CONNECT THERMISTOR 1 TO BB 1")`
Read the thermistor	`Send("READ THERMISTOR 1")` `Get(T):Disp T`

TI Analog Temperature Sensor Data Sheet

Title	**TI Analog Temperature Sensor**
TI Item Name	STEMEE/AC/TEMPSN/A
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	Sensor that reports a voltage proportional to the ambient temperature within a range of −55°C to 130°C.
Category	Environmental Sensors
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Accuracy at +30°C ±2.5 °C (max)’ Accuracy at +130°C & −55°C ±3.5 to ±3.8 °C (max)’ Power Supply Voltage Range +2.4V to +5.5V’ Current Drain 10 µA (max), Nonlinearity ±0.4 % (typ), Output Impedance 160 Ω (max), Load Regulation0µA < IL< +16 µA **See**: Detailed Technical documentation.

HUB Commands	
Sketch Object	TEMPERATURE
Command Syntax	Send("READ TEMPERATURE n")

Code Sample:

Desired Action	**Code Sample**
Configure the program to use TEMPERATURE on pin BB 1	`Send("CONNECT TEMPERATURE 1 TO BB 1")`
Read the temperature sensor	`Send("READ TEMPERATURE 1")` `Get (T):Disp T`

Visible Light Sensor Data Sheet

Title	Visible Light Sensor
TI Item Name	STEMEE/AC/LHTSEN/A
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	Sensor that reports the level of ambient light.
Category	Environmental Sensors
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	

HUB Commands	
Sketch Object	LIGHTLEVEL or ANALOG.IN
Command Syntax	Send("READ LIGHTLEVEL n")

Code Sample:

Desired Action	Code Sample
Configure the program to use LIGHT LEVEL on pin BB 4	`Send("CONNECT LIGHTLEVEL 1 TO BB 4")`
Read the light sensor	`Send("READ LIGHTLEVEL 1")` `Get(L):Disp L`

LEDs and Displays

Topic Links

- Green LED Data Sheet
- RGB (Red-Green-Blue) LED Data Sheet
- Red LED Data Sheet
- Diode Data Sheet
- 7-segment Display Data Sheet
- Infrared Receiver Data Sheet
- Infrared Transmitter Data Sheet

Green LED Data Sheet

Title	Green LED
TI Item Name	STEMEE/AC/LED/A
Quantity	5
Included in	TI-Innovator™ Breadboard Pack
Description	Light-emitting diode that emits green light when current passes through it.
Category	LEDs and Displays
Hub Connection	breadboard circuit
Assembly Instructions	Longer leg (lead) is positive (anode). If both leads are of equal length, the lead adjacent to the flat edge on LED casing is the negative (cathode) lead.
Precautions	Do not insert the leads of LEDs directly into the Hub's Breadboard Connector. Assemble the components on the breadboard and use the provided jumper cables to connect the breadboard to the Hub.
Specifications	Voltage - Forward (Vf) (Typ): 2.1V, Current – Test: 10mA, Viewing Angle: 36°, Mounting Type: Through Hole.

HUB Commands	
Sketch Object	LED or DIGITAL.OUT
Command Syntax	Send("SET LED i [TO] 0-255 [[BLINK|TOGGLE] frequency] [[TIME] seconds]")
Code Sample:	

Desired Action	Code Sample

	`Send("SET LED 1 TO ON")`
	`Send("SET LED 1 TO OFF")`
	`Send("SET LED 1 TO ON TIME 5")`

HUB Commands		
	Desired Action	**Code Sample**

		`Send("SET DIGITAL.OUT 1 TO ON")`
		`Send("SET DIGITAL.OUT 1 TO OFF")`
		`Send("SET DIGITAL.OUT 1 TO ON TIME 5")`

RGB (Red-Green-Blue) LED Data Sheet

Title	**RGB (Red-Green-Blue) LED**
TI Item Name	STEMEE/AC/LED/B
Quantity	2
Included in	TI-Innovator™ Breadboard Pack
Description	Light-emitting diode with independently adjustable red, green and blue elements. Can produce a wide variety of colors..
Category	LEDs and Displays
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Do not insert the leads of LEDs directly into the Hub’s Breadboard Connector. Assemble the components on the breadboard and use the provided jumper cables to connect the breadboard to the Hub.
Specifications	Not Applicable

HUB Commands	
Sketch Object	RGB
Command Syntax	Send("SET RGB 1 TO r g b") - r = red value, g = green value, b = blue value Send("SET RGB 1 TO r g b [[BLINK|TOGGLE] frequency] [[TIME] seconds]")
Code Sample:	

Desired Action	**Code Sample**

Configure LED	`Send("SET RGB 1 ON ON OFF")`
	`Send("SET RG 1 255 128 0")`
	`Send("SET RGB 1 255 128 0 TIME 10")`

	Desired Action	Code Sample
		Send("SET RGB 1 255 128 0 BLINK 20 TIME 10") Send("SET RED 1 0") Send("SET GREEN 1 128 BLINK 2 TIME 10")

Red LED Data Sheet

Title	**Red LED**
TI Item Name	STEMEE/AC/LED/C
Quantity	10
Included in	TI-Innovator™ Breadboard Pack
Description	Light-emitting diode that emits red light when current passes through it.
Category	LEDs and Displays
Hub Connection	breadboard circuit
Assembly Instructions	Longer leg (lead) is positive (anode). If both leads are of equal length, the lead adjacent to the flat edge on LED casing is the negative (cathode) lead.
Precautions	Do not insert the leads of LEDs directly into the Hub's Breadboard Connector. Assemble the components on the breadboard and use the provided jumper cables to connect the breadboard to the Hub.
Specifications	Voltage - Forward (Vf) (Typ): 2V, Current – Test: 10mA, Viewing Angle: 60°, Mounting Type: Through Hole

HUB Commands	
Sketch Object	LED or DIGITAL.OUT
Command Syntax	Send("SET LED n …") ON/OFF [BLINK frequency] [TIME duration]

Code Sample:

Desired Action	Code Sample

Configure LED	`Send("SET LED 1 TO ON")`
	`Send("SET LED 1 TO OFF")`
	`Send("SET LED 1 TO BLINK 2 TIME 5")`
	`Send("SET LED 1 TO ON`

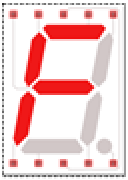
Desired Action	Code Sample
	`TIME 5")`

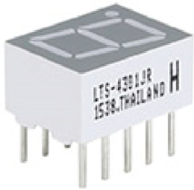
	`Send("SET DIGITAL.OUT 1 TO ON")`
	`Send("SET DIGITAL.OUT 1 TO OFF")`
	`Send("SET DIGITAL.OUT 1 TO BLINK 2 TIME 5")`
	`Send("SET DIGITAL.OUT 1 TO ON TIME 5")`

Diode Data Sheet

Title	**Diode**
TI Item Name	STEMEE/AC/DIO/A
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	Component that allows an electric current to pass in one direction, while blocking current in the opposite direction.
Category	LEDs and Displays
Hub Connection	breadboard circuit
Assembly Instructions	Lead near grey band is cathode (negative pin)
Precautions	Not Applicable
Specifications	Voltage - DC Reverse (Vr) (Max): 100V, Current - Average Rectified (Io): 200mA, Voltage - Forward (Vf) (Max) @ If: 1V @ 10mA, Speed: Small Signal =< 200mA (Io), Any Speed, Current - Reverse Leakage @ Vr: 5µA @ 75V, Capacitance @ Vr, F: 4pF @ 0V, 1MHz, Operating Temperature – Junction: -65°C ~ 175°C

7-segment Display Data Sheet





Title	**7-segment Display**
TI Item Name	STEMEE/AC/DISP/A
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	Array of LEDs arranged to display numbers and some alphabetic characters. Also has an LED for a decimal point.
Category	LEDs and Displays
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	20mA max per segment, Vf:2V

HUB Commands	
Sketch Object	DIGITAL.OUT
Command Syntax	Send("SET DIGITAL.OUT n ON") - n = 1 to 7
Code Sample:	

Desired Action	**Code Sample**

Configure the program to use 7 DIGITAL.OUT on pins BB 1 - 7	**For (N, 1, 7)**
	`Send("CONNECT DIGITAL.OUT eval(N) TO BB eval(N)")`
	`Send("SET DIGITAL.OUT eval(N) ON")`
	`End`

Infrared Receiver Data Sheet

Title	**Infrared Receiver**
TI Item Name	STEMEE/AC/REC/A
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	Side emitting Infrared LED, designed to be paired with the LTR-301 Photo-Transistor.
Category	LEDs and Displays
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Power Dissipation: 100mW, Peak Formard Current: 3A with 300 x 1µs pulses per second, Continuous Forward Current: 50 mA, Reverse Voltage: 5V, Forward Voltage: 1.2V, Operating Temperature Range: -55°C - 100°C, Peak Wavelength: 940 nM, Viewing Angle: 40°

HUB Commands	
Sketch Object	DIGITAL.IN
Command Syntax	Send("READ DIGITAL.IN n")

Code Sample:

Desired Action	Code Sample
	`Send("CONNECT DIGITAL.IN 1 TO BB 2")`
	`Send("READ DIGITAL.IN 1")` `Get(D):Disp D`

Infrared Transmitter Data Sheet

Title	**Infrared Transmitter**
TI Item Name	STEMEE/AC/TRANS/A
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	Side sensing Infrared photo transistor, designed to be paired with the LTE-301 Infrared Emitter.
Category	LEDs and Displays
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Power Dissipation: 100mW, Collector-Emitter Voltage: 30V, Emitter-Collector Voltage: 5V, Operating Temperature: -40°C to 85°C, Storage Temperature: -55°C to 100°

HUB Commands	
Sketch Object	DIGITAL.OUT
Command Syntax	Send("SET DIGITAL.OUT n ON")
Code Sample:	

Desired Action	**Code Sample**
	`Send("CONNECT DIGITAL.OUT 1 TO BB 5")` `Send("SET DIGITAL.OUT 1 ON")`

Motors

Small DC Motor Data Sheet

Title	**Small DC Motor**
TI Item Name	STEMEE/AC/MOTOR/A
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	Motor that converts direct current electrical power into mechanical power.
Category	Motors
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Nominal Voltage: 4.7V, Operating Voltage: 2.0-5.5V, No Load Speed: 19900 r/min, No Load Current: 0.11A, At Maximum Efficiency of Torque: 0.14mN.m (1.4g.cm), At Maximum Efficiency of Output: 0.23W, Stall Torque: 0.7mN.m(7.1g.cm), Stall Current: 0.42A

HUB Commands	
Sketch Object	DCMOTOR
Command Syntax	Send("SET DCMOTOR n TO frequency [duty [TIME] seconds]") frequency - 1 to 500Hz duty - 1 to 99% duty cycle (default: 50%) seconds = 1s default

HUB Commands		
Code Sample:	**Desired Action**	**Code Sample**
		`Send("SET DCMOTOR 1 TO 50 TIME 5")`

Power and Signal Control

Topic Links

- SPDT Slide Switch Data Sheet
- 8 Position DIP Switch Data Sheet
- 8 100 Ohm Resistor SIP Package Data Sheet
- TTL Power MOSFET Data Sheet

SPDT Slide Switch Data Sheet

Title	**SPDT Slide Switch**
TI Item Name	STEMEE/AC/SWIT/A
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	Single pole, double throw switch. Slide the switch knob back and forth to open and close contacts.
Category	Power and Signal Control
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	30V, 200mA

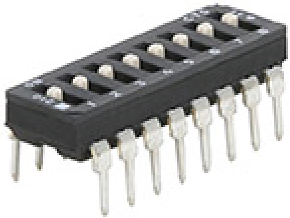
HUB Commands	
Sketch Object	SWITCH
Command Syntax	Send("READ SWITCH n")

Code Sample:

Desired Action	Code Sample

Configure the program to use SWITCH on port BB 1	`Send("CONNECT SWITCH 1 TO BB 1")`
	`Send("READ SWITCH 1")`
	`Get(T):Disp T`

8 Position DIP Switch Data Sheet



Title	**8 Position DIP Switch**
TI Item Name	STEMEE/AC/SWIT/B
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	Set of 8 slide switches used to customize the behavior of the circuit components for specific situations.
Category	Power and Signal Control
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	'0.100", 100mA, 20VDC

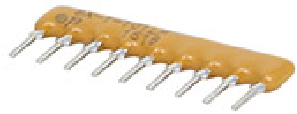
HUB Commands	
Sketch Object	DIGITAL.IN
Command Syntax	Send("READ DIGITAL.IN n") - n = 1 to 8 **or** Send("READ SWITCH n") - n = 1 to 8

Code Sample:

Desired Action	Code Sample

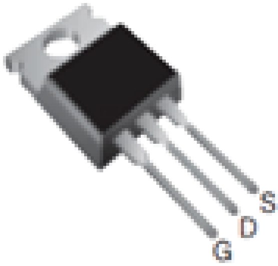
Configure the program to use 8 SWITCHs on pins BB 1 - 8	**For (N, 1, 8)**
	`Send("CONNECT SWITCH eval(N) TO BB eval(N) ")`
	`Send("READ SWITCH eval (N)")`
	`Get(S):Disp S`
	`End`

8 100 Ohm Resistor SIP Package Data Sheet



Title	**8 100 Ohm resistor SIP Package**
TI Item Name	STEMEE/AC/RES/E
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	8 100 Ohm resistor SIP package for use with the 8 Position DIP Switch.
Category	Power and Signal Control
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	Bussed array

TTL Power MOSFET Data Sheet



Title	**TTL Power MOSFET**
TI Item Name	STEMEE/AC/MOSFET/A
Quantity	2
Included in	TI-Innovator™ Breadboard Pack
Description	Transistor used for amplifying or switching electronic signals.
Category	Power and Signal Control
Hub Connection	breadboard circuit
Assembly Instructions	Connect the G-GATE to the BB pin of the TI-Innovator™ Hub, the D-DRAIN to the load being controlled (e.g., DC motor) and the S-SINK to ground.
Precautions	If the metal plate on the MOSFET becomes hot during use, disconnect the battery immediately and re-check all connections.
Specifications	supports 100A

HUB Commands	
Sketch Object	RELAY **or** ANALOG.OUT
Command Syntax	Send("SET RELAY n TO ON/OFF [[TIME] seconds]") **or** Send("SET ANALOG.OUT n TO 0-255/ON/OFF [[BLINK] frequency] [[TIME] seconds]")
Code Sample:	**Note**: a MOSFET can either be used as an ON/OFF control (RELAY) or for finer control (ANALOG.OUT)

Desired Action	Code Sample
	Send("CONNECT RELAY 1 TO BB 7") Send("SET RELAY 1 ON")
	Send("CONNECT ANALOG.OUT 1 TO BB 7") Send("SET ANALOG.OUT 1 127")

Passive Components

Topic Links

- Accessories
- Breadboard
- Capacitors
- Resistors

Accessories

40-Pack Male to Male Breadboard Jumper Cable Data Sheet

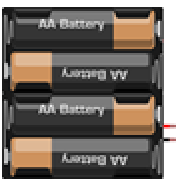
Title	**40-Pack Male to Male Breadboard Jumper Cables**
TI Item Name	STEMEE/AC/CABKT/A
Quantity	40
Included in	TI-Innovator™ Breadboard Pack
Description	Male to Male jumper cables for connecting components on the breadboard.
Category	Accessories
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Cable lead may break if bent repeatedly
Specifications	Male to Male Pack of 40, 20cm

10-Pack Male to Female Breadboard Jumper Cable Data Sheet

Title	**10-Pack Male to Female Breadboard Jumper Cables**
TI Item Name	STEMEE/AC/CABKT/B
Quantity	10
Included in	Innovator™ Breadboard Pack

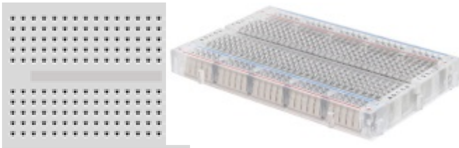
Title	10-Pack Male to Female Breadboard Jumper Cables
Description	Male to female jumper cables for connecting components on the breadboard.
Category	Accessories
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Cable lead may break if bent repeatedly
Specifications	Male to Female Pack of 10, 20cm

4-AA Battery Holder Data Sheet



Title	4-AA Battery Holder
TI Item Name	STEMEE/AC/BATHLD/A
Quantity	1
Included in	Innovator™ Breadboard Pack
Description	4-AA battery holder with tined solid leads for easy breadboard insertion.
Category	Accessories
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	BHC-341-1A with lead wires 150mm, Strip & Tin: 5mm+/-1mm, UL1007, AWG 26

Breadboard Data Sheet



Title	Breadboard
TI Item Name	STEMEE/AC/BRDBD/A
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	Platform for connecting the electronic components of a project by inserting component leads and jumper cables into pins.
Category	Breadboard
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Do not connect the positive and negative leads of a power source to the same group of 5 pins on the breadboard. Doing so could damage the breadboard and the power source. Observe the correct polarity: When connecting the breadboard to the Hub. When connecting components that are sensitive to polarity, such as LEDs and the TTL Power MOSFET. **See also:** TI-Innovator™ Hub Ports and Breadboard Usable Pins
Specifications	45.7x35.6x9.4mm, 170 tie-point, POM plastic (150℃), Round Hole, with screwsx2pcs

Capacitors

Capacitor 100µF Data Sheet

Title	**Capacitor 100µF**
TI Item Name	STEMEE/AC/CAP/A
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	Capacitor that temporarily stores an electric charge of up to 100µF.
Category	Capacitors
Hub Connection	breadboard circuit
Assembly Instructions	Longer leg (lead) is positive (anode). If both leads are of equal length, the lead adjacent to the colored strip on the casing is the negative (cathode) lead.
Precautions	Not Applicable
Specifications	Capacitance: 100µF, Tolerance: ±20%, Voltage Rating: 16V

Capacitor 10µF Data Sheet

Title	**Capacitor 10µF**
TI Item Name	STEMEE/AC/CAP/B
Quantity	1
Included in	Innovator™ Breadboard Pack
Description	Capacitor that temporarily stores an electric charge of up to 10µF.
Category	Capacitors

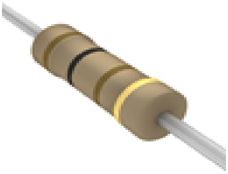
Title	Capacitor 10µF
Hub Connection	breadboard circuit
Assembly Instructions	Longer leg (lead) is positive (anode). If both leads are of equal length, the lead adjacent to the colored strip on the casing is the negative (cathode) lead.
Precautions	Not Applicable
Specifications	Capacitance: 10µF, Tolerance: ±20%, Voltage Rating: 16V

Capacitor 1µF Data Sheet

Title	Capacitor 1µF
TI Item Name	STEMEE/AC/CAP/C
Quantity	1
Included in	Innovator™ Breadboard Pack
Description	Capacitor that temporarily stores an electric charge of up to 1µF.
Category	Capacitors
Hub Connection	breadboard circuit
Assembly Instructions	Longer leg (lead) is positive (anode). If both leads are of equal length, the lead adjacent to the colored strip on the casing is the negative (cathode) lead.
Precautions	Not Applicable
Specifications	Capacitance: 1µF, Tolerance: ±20%, Voltage Rating: 16V

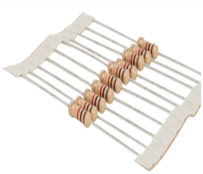
Resistors

Resistor 100 Ohm Data Sheet



Title	**Resistor 100 Ohm**
TI Item Name	STEMEE/AC/RES/A
Quantity	10
Included in	TI-Innovator™ Breadboard Pack
Description	Resistor that provides 100 Ohms of resistance in a circuit. Color Code Value: brown, black, brown.
Category	Resistors
Hub Connection	breadboard circuit
Assembly Instructions	No polarity
Precautions	Not Applicable
Specifications	'Resistance (Ohms): 100, Tolerance: ±5%, Power (Watts): 0.5W, 1/2W, Temperature Coefficient: 0/ -400ppm/°C, Operating Temperature: -55°C ~ 155°C

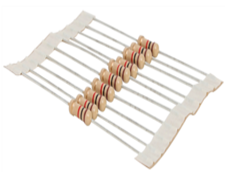
Resistor 1K Ohm Data Sheet

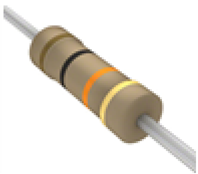


Title	**Resistor 1K Ohm**
TI Item Name	STEMEE/AC/RES/B

Title	**Resistor 1K Ohm**
Quantity	10
Included in	TI-Innovator™ Breadboard Pack
Description	Resistor that provides 1K Ohms of resistance in a circuit. Color Code Value: brown, black, red.
Category	Resistors
Hub Connection	breadboard circuit
Assembly Instructions	No polarity
Precautions	Not Applicable
Specifications	'Resistance (Ohms): 1K, Tolerance: ±5%, Power (Watts): 0.5W, 1/2W, Temperature Coefficient: 0/ -400ppm/°C, Operating Temperature: -55°C ~ 155°C

Resistor 10K Ohm Data Sheet

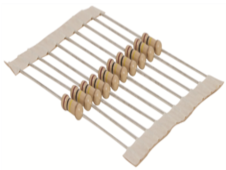
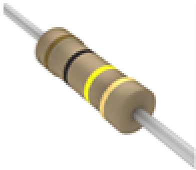




Title	**Resistor 10K Ohm**
TI Item Name	STEMEE/AC/RES/C
Quantity	10
Included in	TI-Innovator™ Breadboard Pack
Description	Resistor that provides 10K Ohms of resistance in a circuit. Color Code Value: brown, black, orange.
Category	Resistors
Hub Connection	breadboard circuit
Assembly Instructions	No polarity

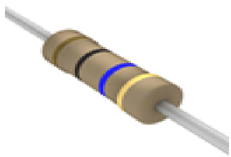
Title	Resistor 10K Ohm
Precautions	Not Applicable
Specifications	'Resistance (Ohms): 10K, Tolerance: ±5%, Power (Watts): 0.5W, 1/2W, Temperature Coefficient: 0/ -400ppm/°C, Operating Temperature: -55°C ~ 155°C

Resistor 100K Ohm Data Sheet

Title	Resistor 100K Ohm
TI Item Name	STEMEE/AC/RES/D
Quantity	10
Included in	TI-Innovator™ Breadboard Pack
Description	Resistor that provides 100K Ohms of resistance in a circuit. Color Code Value: brown, black, yellow.
Category	Resistors
Hub Connection	breadboard circuit
Assembly Instructions	No polarity
Precautions	Not Applicable
Specifications	'Resistance (Ohms): 100K, Tolerance: ±5%, Power (Watts): 0.5W, 1/2W, Temperature Coefficient: 0/ -400ppm/°C, Operating Temperature: -55°C ~ 155°C

Resistor 10M Ohm Data Sheet



Title	**Resistor 10M Ohm**
TI Item Name	STEMEE/AC/RES/F
Quantity	10
Included in	TI-Innovator™ Breadboard Pack
Description	Resistor that provides 10M Ohms of resistance in a circuit. Color Code Value: brown, black, blue.
Category	Resistors
Hub Connection	breadboard circuit
Assembly Instructions	No polarity
Precautions	Not Applicable
Specifications	'Resistance (Ohms): 10M, Tolerance: ±5%, Power (Watts): 0.5W, 1/2W, Temperature Coefficient: 0/ -400ppm/°C, Operating Temperature: -55°C ~ 155°C

Potentiometer with Knob Data Sheet

Title	**Potentiometer with Knob**
TI Item Name	STEMEE/AC/POTEN/A
Quantity	1
Included in	TI-Innovator™ Breadboard Pack
Description	Variable resistor with knob used to change the resistance

Title	**Potentiometer with Knob**
	in a circuit.
Category	Resistors
Hub Connection	breadboard circuit
Assembly Instructions	Not Applicable
Precautions	Not Applicable
Specifications	1 Turn, 10K

HUB Commands	
Sketch Object	POTENTIOMETER
Command Syntax	Send("READ POTENTIOMETER n")
Code Sample:	

Desired Action	Code Sample
Read potentiometer	`Send("READ POTENTIOMETER 1")` `Get(P):Disp P`

Domande frequenti

Questa sezione include alcune delle domande frequenti che abbiamo ricevuto a proposito della
tecnologia TI-Innovator™. Se la domanda inviata non è presente, inviare un feedback al team eGuide all'indirizzo hubeguide@list.ti.com

Collegamenti argomento

- Informazioni sulla compatibilità del prodotto
- Informazioni su TI LaunchPad™
- Informazioni generali sulle attività
- Informazioni generali sull'alimentazione di TI-Innovator™ Hub
    - Informazioni sulla batteria esterna per TI-Innovator™ Hub
    - Informazioni sulla batteria del Rover

Informazioni sulla compatibilità del prodotto

Quali prodotti TI sono compatibili con TI-Innovator™ Hub?

TI-Innovator™ Hub è compatibile con i seguenti prodotti TI. Per risultati ottimali, utilizzare sempre l'ultima versione del sketch TI-Innovator e dei prodotti compatibili.

- calcolatrice grafica TI CE
- Palmare TI-Nspire™ CX
- Palmare TI-Nspire™ CX CAS
- Software per computer TI-Nspire™ CX (Student, Teacher e TI-Nspire™ CX Navigator™)

Quale linguaggio di programmazione è compatibile con TI-Innovator™ Hub?

TI-Innovator™ Hub può essere programmato utilizzando il linguaggio di programmazione **TI BASIC** sia sulle calcolatrici grafiche TI CE sia sulle calcolatrici TI-Nspire™ CX. Questo linguaggio di programmazione viene utilizzato in diverse calcolatrici grafiche TI CE e si basa sul linguaggio di programmazione BASIC (Beginner's All-purpose Symbolic Instruction Code). BASIC è una serie di linguaggi di programmazione generici di alto livello la cui filosofia di progettazione si basa sulla loro facilità d'uso.

Inoltre, con la tecnologia TI-Nspire™ CX è possibile utilizzare la **programmazione LUA**, un linguaggio di scripting rapido e potente.

Vedere anche: Hub Programmazione dell'Hub sulla calcolatrice grafica TI CE per ulteriori.

Vedere anche: Hub Programmazione dell'Hub nella tecnologia TI-Nspire™ CX per ulteriori.

Quali sensori, attuatori e simili è possibile collegare a TI-Innovator™ Hub?

TI-Innovator™ Hub presenta due tipi di connettori:

- Connettore universale a 4 pin compatibile con una serie di moduli.
- Connettore della basetta sperimentale che può essere collegato a una basetta sperimentale per eseguire il prototipo dei progetti.

Per facilitare l'utilizzo di questi strumenti, sono disponibili comodi kit che contengono tutti i componenti necessari per il completamento delle attività. Per ulteriori dettagli, vedere le sezioni relative al modulo di I/O e alla basetta sperimentale.

È possibile utilizzare la basetta TI-Nspire™ Lab con i sensori Vernier™ mentre si utilizza TI-Innovator™ Hub?

Sì, la basetta TI-Nspire™ Lab può essere utilizzata contemporaneamente con TI-Innovator™ Hub sul palmare TI-Nspire™ CX o sul software TI-Nspire™ CX. Per poter utilizzare contemporaneamente TI-Innovator™ Hub e la basetta TI-Nspire™ Lab, occorre eseguire l'accesso mediante uno script LUA.

È possibile collegare direttamente i sensori Vernier™ a TI-Innovator™ Hub?

Le porte di TI-Innovator™ Hub non sono direttamente compatibili con i sensori Vernier™. È possibile collegare i sensori Vernier™ a una basetta TI-Nspire™ Lab. Per poter utilizzare contemporaneamente TI-Innovator™ Hub e la basetta TI-Nspire™ Lab, occorre eseguire l'accesso mediante uno script LUA.

È possibile utilizzare il sistema TI-Nspire™ CX Navigator™ quando si usa TI-Innovator™ Hub?

Sì, gli studenti possono collegare il loro palmare TI-Nspire™ CX al sistema TI-Nspire™ CX Navigator™ mentre utilizzano TI-Innovator™ Hub. L'insegnante può utilizzare la funzionalità TI-Nspire™ CX Navigator™, tra cui Presentatore dal vivo, Cattura schermo, Test rapido, ecc. mentre gli studenti usano TI-Innovator™ Hub.

Il software TI Connect™ CE o TI-SmartView™ CE è in grado di comunicare con TI-Innovator™ Hub?

TI-Innovator™ Hub non può comunicare direttamente con il software TI Connect™ CE o TI-SmartView™ CE. Tuttavia, è possibile utilizzare il software TI Connect™ CE per scrivere programmi da usare con TI-Innovator™ Hub. Il software TI-SmartView™ CE è un programma ideale per mostrare la procedura di programmazione agli studenti.

Informazioni su TI LaunchPad™

Che cos'è un kit di sviluppo TI LaunchPad™?

I kit TI LaunchPad sono una serie di kit di sviluppo per microcontrollori (anche chiamati schede di valutazione) prodotti da Texas Instruments. È possibile trovare ulteriori dettagli sull'ecosistema TI LaunchPad all'indirizzo https://www.ti.com/ww/en/launchpad/about.html.

Che tipo di kit TI LaunchPad™ viene utilizzato in TI-Innovator™ Hub?

TI-Innovator™ Hub è alimentato da un kit TI LaunchPad MSP432P401. Per ulteriori informazioni su LaunchPad MSP432P401, visitare il sito https://www.ti.com/ww/en/launchpad/launchpads-msp430-msp-exp432p401r.html#tabs.

È possibile utilizzare TI-Innovator™ Hub come kit di sviluppo TI LaunchPad™?

Anche se TI-Innovator™ Hub può essere utilizzato come scheda TI LaunchPad™, TI-Innovator™ Hub è stato specificamente progettato per essere utilizzato da studenti che stanno imparando a programmare, costruire ed esplorare utilizzando componenti elettronici. Per ulteriori informazioni su TI LaunchPad, visitare il sito https://www.ti.com/ww/en/launchpad/about.html.

Quali sono le risorse disponibili per TI LaunchPad?

Gli utenti interessati all'ecosistema TI LaunchPad possono visitare il sito https://www.ti.com/ww/en/launchpad/about.html per ulteriori risorse.

Come vengono utilizzati i kit di sviluppo/ le schede tecniche dagli ingegneri nel mondo reale?

Gli ingegneri si servono di schede di valutazione, come ad esempio le schede TI LaunchPad™, per eseguire il prototipo dei loro progetti e verificare la sostenibilità di un chip specifico all'interno del loro progetto. Queste schede permettono agli ingegneri di sperimentare approcci diversi prima di finalizzare il loro progetto. Le schede consentono inoltre agli ingegneri di misurare altri aspetti dei loro progetti, come ad esempio il consumo energetico e la velocità delle operazioni.

Queste schede di valutazione vengono inoltre utilizzate nelle università per imparare il funzionamento dei microcontrollori, la programmazione e l'interfacciamento con i sensori.

Informazioni generali sulle attività

What activities are available for the TI-Innovator™ Hub?

Con TI-Innovator™ Hub è possibile svolgere diverse attività. In collaborazione con gli insegnanti, abbiamo creato attività relative ai seguenti argomenti:

10 minuti di codice per TI-Innovator™ Hub: lo scopo è coinvolgere gli studenti in brevi attività che li aiutino a capire meglio i concetti matematici, la logica di programmazione e le competenze di codifica. Durante le attività, si utilizzano l'RGB integrato, il LED, il diffusore e il sensore di intensità luminosa di TI-Innovator™ Hub. Le attività sono disponibili per la serie di calcolatrici grafiche TI CE e per la tecnologia TI-Nspire™ CX.

10 minuti di codice per TI-Innovator™ Rover: lo scopo è continuare a imparare la programmazione con TI-Innovator™ Rover. Ideata per approfondire la conoscenza della programmazione di TI-Innovator™ Hub e per scrivere programmi per il controllo di TI-Innovator™ Rover. Questa attività consente di imparare i comandi di spostamento del Rover e di utilizzare il ranger e il sensore per i colori integrati. Le attività saranno disponibili per la serie di calcolatrici grafiche TI CE e per la tecnologia TI-Nspire™ CX.

"Conversazioni" durante la lezione di matematica e scienza per TI-Innovator™ Rover: programmi pronti all'uso per la serie di calcolatrici grafiche TI CE e per la tecnologia TI-Nspire™ CX. Questi programmi includono una guida all'uso per l'insegnante che contiene suggerimenti sull'implementazione di TI-Innovator™ Rover nei programmi in dotazione al fine di esplorare i concetti presentati durante la lezione di matematica e/o scienze.

Scienze attraverso la progettazione: lezioni interattive e ricche di contenuti per gli studenti delle scuole medie che studiano biologia e fisica. Si serve di componenti forniti nel pacchetto di moduli di I/O TI-Innovator™. Le attività sono disponibili per la tecnologia TI-Nspire™ CX.

Tragitto per i progetti STEM: progettazione, costruzione, collaudo, perfezionamento. Queste attività sequenziali sono indirizzate agli studenti delle scuole medie e superiori che stanno studiando i principi tecnici e hanno lo scopo di fornire loro le conoscenze e le competenze di base richieste per sintetizzare progetti STEM nuovi e unici. Queste attività richiedono i componenti in dotazione con il Kit della basetta sperimentale TI-Innovator™. Le attività sono disponibili per la serie di calcolatrici grafiche TI CE e per la tecnologia TI-Nspire™ CX.

Dov'è possibile posso scaricare le attività per TI-Innovator™ Hub?

Per scaricare le attività da utilizzare con TI-Innovator™ Hub, visitare il sito Web education.ti.com e selezionare la scheda delle attività nella parte superiore di ogni pagina. Di seguito sono riportati i collegamenti diretti a ogni serie di attività:

- 10 minuti di codice con TI-Innovator™ Hub: education.ti.com/ticodes
- 10 minuti di codice con TI-Innovator™ Rover: education.ti.com/ticodes
- "Conversazioni" durante la lezione di matematica e scienza per TI-Innovator™ Rover:
- Scienze attraverso la progettazione: https://education.ti.com/en/tisciencenspired/us/stem
- Tragitto per i progetti STEM: **da definire**

Quando saranno disponibili queste attività?

Le attività per TI-Innovator™ Hub sono già disponibili. Le attività per TI-Innovator™ Rover saranno disponibili a partire dall'autunno del 2017.

Informazioni generali sull'alimentazione di TI-Innovator™ Hub

How is the TI-Innovator™ Hub alimentato?

TI-Innovator™ Hub è alimentato dalle batterie della calcolatrice grafica TI CE o del palmare TI-Nspire™ CX. Per alcune attività che vengono svolte su dispositivi ad alto consumo come i servomotori, potrebbe essere necessario utilizzare una fonte di alimentazione supplementare, come un adattatore da parete TI o una batteria esterna.

How does the TI-Innovator™ Hub influenza la durata della batteria della calcolatrice grafica TI CE o di TI-Nspire™ CX?

TI-Innovator™ Hub esercita un impatto minimo sulla batteria della calcolatrice grafica TI CE o delle calcolatrici grafiche TI-Nspire™ CX.

Quando è necessario utilizzare l'alimentazione esterna?

Quando si utilizzano le porte di input e output:

> Alcuni moduli di I/O richiedono alimentazione esterna poiché si servono di porte da 5 V (OUT3 o IN3) su TI-Innovator™ Hub. Per i dettagli, vedere la sezione sui moduli di I/O.

Quando si utilizza il connettore della basetta sperimentale:

> Se il circuito è alimentato dall'output da 5 V del connettore della basetta sperimentale, sarà necessaria un'alimentazione esterna.

Quali sono le opzioni di alimentazione esterna disponibili?

Per ottenere un'alimentazione supplementare, è possibile utilizzare l'adattatore da parete TI o la batteria esterna. L'adattatore da parete TI è fornito in dotazione con TI-Innovator™ Hub ed è lo stesso caricatore da parete fornito con la calcolatrice grafica TI CE e le calcolatrici TI-Nspire™ CX. La batteria esterna è venduta separatamente come accessorio per TI-Innovator™ Hub.

Can I use a different battery/power supply with the TI-Innovator™ Hub?

Per garantire un funzionamento sicuro, si consiglia di utilizzare esclusivamente la batteria e la fonte di alimentazione fornite da TI.

Informazioni sulla batteria esterna per TI-Innovator™ Hub

Che cos'è la batteria esterna?

La batteria esterna fornisce alimentazione supplementare per quei componenti che richiedono ulteriore potenza rispetto a quella fornita dalla calcolatrice grafica TI.

Questa batteria (n. modello MP-3000) è stata selezionata per soddisfare le esigenze di alimentazione dei componenti di TI-Innovator™.

Come si utilizza la batteria esterna con TI-Innovator™ Hub?

Utilizzando il cavo USB da Standard-A a Micro-B fornito in dotazione con TI-Innovator™ Hub, collegare la batteria esterna alla porta USB PWR su TI-Innovator™ Hub. La batteria esterna è dotata di interruttore On/Off che deve essere acceso per poter alimentare TI-Innovator™ Hub.

Quanto dura la batteria completamente carica?

La durata della batteria dipenderà dai componenti collegati a TI-Innovator™ Hub. Ad esempio, con la batteria esterna, il modulo del servomotore utilizzato per le attività Scienza attraverso la progettazione può durare 8 ore ininterrotte. Altri componenti potrebbero durare più a lungo oppure esaurire la batteria più rapidamente.

Qual è la durata media della batteria?

Con il passare del tempo, le batterie agli ioni di litio perdono la loro capacità. Se conservate in modo corretto e utilizzate normalmente, le batterie possono durare circa tre anni.

Come si esegue la ricarica della batteria?

La batteria esterna può essere ricaricata con l'adattatore da parete TI (in dotazione con TI-Innovator™ Hub) o collegando il cavo USB fornito con TI-Innovator Hub alla porta USB del computer.

Com'è possibile verificare il livello di ricarica della batteria?

Quando si accende la batteria esterna, gli indicatori LED che si trovano sulla batteria esterna indicheranno il livello di carica approssimativo (25%, 50%, 75% e 100%). I LED si spengono automaticamente dopo 10 secondi.

È possibile utilizzare la batteria esterna con altri prodotti?

La batteria esterna è stata testata specificatamente per essere utilizzata con TI-Innovator™ Hub.

Informazioni sulla batteria del Rover

Quanto dura la batteria completamente carica?

La batteria durerà 8 ore di guida ininterrotta. Durante l'uso, si prendono generalmente frequenti pause per la programmazione. In questo caso, una ricarica completa dovrebbe durare diversi giorni.

Qual è la durata media della batteria?

Con il passare del tempo, le batterie agli ioni di litio perdono la loro capacità. Se conservate in modo corretto e utilizzate normalmente, le batterie possono durare circa 3 anni.

Come si esegue la ricarica della batteria?

Collegare un cavo micro-USB alla porta PWR sul lato anteriore destro del Rover. L'altra estremità del cavo può essere collegata a un PC o a un caricatore da parete TI.

Com'è possibile verificare il livello di ricarica della batteria?

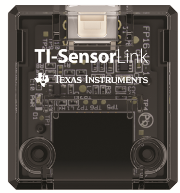
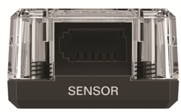
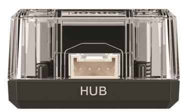
I LED con i quattro livelli di carica della batteria mostrano la capacità della batteria. Quando tutti e quattro i LED emettono una luce verde fissa, significa la batteria del Rover è completamente carica.

Adattatore TI-SensorLink

Che cos'è l'adattatore TI-SensorLink?

L'adattatore TI-SensorLink è un accessorio che consente a TI-Innovator™ Hub di utilizzare i sensori analogici Vernier con l'Hub. TI-SensorLink amplia i campi di applicazione dei progetti STEM collegando sensori Vernier specifici a TI-SensorLink e in seguito a TI-Innovator™ Hub.

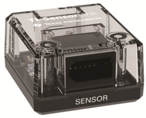
Attenzione: Se si rimuove o scollega uno qualsiasi dei cavi, fare riferimento a questa immagine per ripristinare collegamenti in maniera corretta.

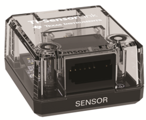
TI-SensorLink - Design industriale e marcature	
Vista dall'alto dell'adattatore TI-SensorLink	
Vista anteriore - Porta per il collegamento di rilevatori e sensori	
Vista posteriore - Porta per il collegamento dell'Hub	
Vista inferiore - Identificazione dell'etichetta	

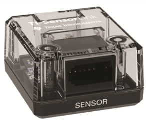
Sensori analogici Vernier supportati

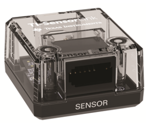
Con TI-SensorLink, vengono ufficialmente supportati questi quattro sensori analogici Vernier.

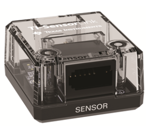
Modulo	Porte	Immagine	Codice di esempio per TI-SensorLink

Sensore di temperatura in acciaio inossidabile	TI-SensorLink
	
			Collegamento a:
			`Send "CONNECT VERNIER 1 TO IN1 AS TEMPERATURE"`
			`Send "READ VERNIER 1"`
			`Get T`

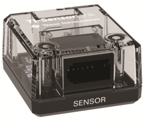
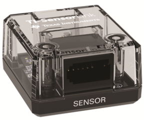
pH-metro	TI-SensorLink
	
			Collegamento a:
			`Send "CONNECT VERNIER 2 TO IN2 AS PH"`
			`Send "READ VERNIER 2"`
			`Get P`

Sensore di pressione del gas	TI-SensorLink
	
			Collegamento a:
			`Send "CONNECT VERNIER 1 TO IN1 AS PRESSURE"`
			`Send "READ VERNIER 1"`
			`Get P`

Sensore di forza	TI-SensorLink
	
			Collegamento a:
			`Send "CONNECT VERNIER 2 TO IN2 AS FORCE"`
			or
			`Send "CONNECT VERNIER 2 TO IN2 AS FORCE50"`
			
			`Send "READ VERNIER 2"`
			`Get F`

Accelerometro low-g	TI-SensorLink
	
			Collegamento a:
			`Send "CONNECT VERNIER 1 TO IN 1 AS ACCEL"`
			`Send "READ VERNIER 1"`

Rivelatore di luce	TI-SensorLink		Collegamento a:
			`Send "CONNECT VERNIER 1 TO IN 1 AS LIGHT"`
			`Send "READ VERNIER 1"`

Modulo	**Porte**	**Immagine**	**Codice di esempio per TI-SensorLink**
			
Sensore di energia Vernier	TI-SensorLink 		Collegamento a: `Send "CONNECT VERNIER 1 TO IN 1 AS ENERGY"` `Send "READ VERNIER 1"`

Requisiti per l'adattatore Vernier:

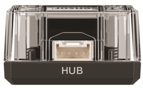
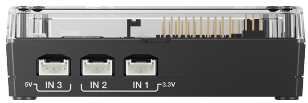
Hardware:

- Adattatore TI-SensorLink aggiuntivo su TI-Innovator™ Hub
- Supporta un unico sensore analogico Vernier
- Funziona su tutte le tre porte IN dell'Hub
  - L'utilizzo con la porta I2C o le porte OUT **NON** è supportato e il sketch indicherà un errore
- Vengono supportati i seguenti sensori:
  - Sensore di temperatura in acciaio inossidabile
  - pH-metro
  - Sensore di pressione del gas
  - Sensore di forza
  - Accelerometro low-g
  - Rivelatore di luce
  - Sensore di energia Vernier

Collegamento dell'adattatore TI-SensorLink

Eseguire questa serie di passaggi rispettandone l'ordine per collegare e utilizzare l'adattatore TI-SensorLink.

Collegamento dell'adattatore TI-SensorLink a TI-Innovator™ Hub

Adattatore TI-SensorLink	**Cavo in dotazione**	**Hub TI-Innovator™**
		

PASSAGGI

1. Collegare un'estremità del cavo in dotazione alla porta di TI-SensorLink etichettata HUB.
2. Collegare l'altra estremità del cavo in dotazione alla porta dell'Hub etichettata IN1.

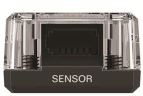
   Nota: è anche possibile inserire il cavo in IN2 o IN3.

Collegamento di TI-Innovator™ Hub a una calcolatrice grafica

La TI-Innovator™ Hub si collega via cavo USB a un computer o a una calcolatrice grafica. Il collegamento consente all'Hub di essere alimentato e di scambiare dati con l'host.

Vedere i dettagli completi (page 5).

Collegamento dell'adattatore TI-SensorLink a un sensore Vernier

Adattatore TI-SensorLink	**Sensore Vernier**
	

Collegare TI-SensorLink a uno dei quattro sensori analogici Vernier supportati utilizzando il connettore collegato del sensore analogico.

PASSAGGI

1. Collegare il sensore Vernier a TI-SensorLink (in questo esempio viene utilizzato il rilevatore di temperatura in acciaio inossidabile).
2. Nella calcolatrice grafica collegata, inserire il codice seguente:

```
Send "CONNECT VERNIER 1 TO IN1 AS TEMPERATURE"

Send "READ VERNIER 1"

Get T
```

Nota: I nuovi comandi e le nuove parole chiave devono essere digitate OPPURE copiate da un programma esistente. Tenere presente che eventuali errori tipografici nelle parole chiave attiverà un'indicazione di errore nel sketch.

Vedere gli esempi di codice per:

- Sensore di forza
- Sensore di pressione del gas
- pH-metro
- Sensore di temperatura in acciaio inossidabile

Precauzioni per l'adattatore TI-SensorLink e i sensori Vernier

Adattatore TI-SensorLink

- TI-SensorLink **non** è una soluzione per la raccolta dei dati. Per la raccolta e l'analisi dei dati, si consiglia di utilizzare i rilevatori collegati USB o la basetta Lab.
- I comandi dell'Hub per TI-SensorLink con i sensori analogici Verner **non** sono attualmente inclusi nell'app Hub (serie CE) o nel menu Hub (TI-Nspire™ CX).
- I nuovi comandi e le nuove parole chiave devono essere digitate OPPURE copiate da un programma esistente. Tenere presente che eventuali errori tipografici nelle parole chiave attiverà un'indicazione di errore nel sketch.

Sensori Vernier

- Sensore di pressione del gas: il sensore di pressione del gas è un elemento sensibile che viene danneggiato se entra in contatto diretto con un liquido.
- Sensore del pH: collocare l'elettrodo in una soluzione tampone con pH 4 o pH 7. Non deve mai essere conservato in acqua distillata. Se l'elettrodo viene involontariamente conservato a secco per un breve periodo di tempo, immergerne la punta nella soluzione di conservazione di KCl/tampone con pH 4 per almeno 8 ore prima di utilizzarlo.
- Rilevatore di temperatura in acciaio inossidabile -
  - Attorcigliamento del cavo. Talvolta gli studenti possono attorcigliare o schiacciare il cavo nelle vicinanze dell'impugnatura del sensore. Con il tempo, i cavi potrebbero allentarsi interrompendo il funzionamento del sensore.

- Surriscaldamento del sensore. Quanto utilizzato nei laboratori di chimica, talvolta gli studenti possono collocare il sensore su una piastra calda causando la vera e propria "cottura" dell'unità.
- L'unità non è resistente all'acqua. L'acqua può penetrare all'interno del manico del sensore danneggiandone i componenti elettronici. Durante la raccolta dei dati, immergere in acqua solo la parte del sensore in acciaio inossidabile.

Fogli dati dell'adattatore TI-SensorLink e dee sensore Vernier

I fogli dati dell'adattatore TI-SensorLink e del sensore Vernier includono le seguenti informazioni: il nome e il numero del prodotto, una breve descrizione, un'immagine del prodotto, le specifiche, il tipo di collegamento del componente a TI-Innovator™ Hub e i comandi dell'Hub con esempi di codice semplici.

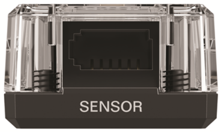
Collegamenti argomento

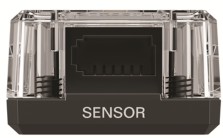
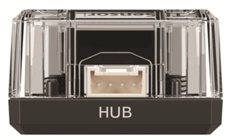
- Foglio dati dell'adattatore TI-SensorLink
- **Fogli dati del sensore Vernier**
  - Foglio dati del rilevatore di temperatura in acciaio inossidabile
  - Foglio dati del sensore pH
  - Foglio dati del sensore di forza
  - Foglio dati del sensore di pressione del gas
  - Foglio dati dell’accelerometro low-g
  - Foglio dati del sensore di luce
  - Foglio dati del sensore di energia Vernier

Nota:

- TI-SensorLink **non** è una soluzione per la raccolta dei dati. Per la raccolta e l'analisi dei dati, si consiglia di utilizzare i rilevatori collegati USB o la basetta Lab.
- I comandi dell'Hub per TI-SensorLink con i sensori analogici Verner **non** sono attualmente inclusi nell'app Hub (serie CE) o nel menu Hub (TI-Nspire™ CX).
- I nuovi comandi e le nuove parole chiave devono essere digitate OPPURE copiate da un programma esistente. Tenere presente che eventuali errori tipografici nelle parole chiave attiverà un'indicazione di errore nel sketch.

Foglio dati dell'adattatore TI-SensorLink



Titolo	**Adattatore TI-SensorLink**
Nome elemento TI	STEMKT/AC/SL/A
Incluso in	Adattatore TI-SensorLink
Quantità	1
Descrizione	Accessorio che consente a TI-Innovator™ Hub di utilizzare i sensori analogici Vernier con l'Hub **Nota:** Non è una soluzione per la raccolta dei dati – Per la raccolta e l’analisi dei dati, si consiglia di utilizzare i rilevatori collegati USB o la basetta Lab
Categoria	Adattatore
Hub Collegamento	 
Istruzioni di montaggio	N/D
Precauzioni	.
Specifiche tecniche	

Foglio dati del rilevatore di temperatura in acciaio inossidabile

Titolo	**Rilevatore di temperatura in acciaio inossidabile Vernier**
Nome elemento TI	n/d
Codice ordine Vernier	TMP-BTA
Incluso in	Sensore di temperatura in acciaio inossidabile
Quantità	1
Descrizione	Il rilevatore di temperatura in acciaio inossidabile è un sensore di temperatura generico e resistente che può essere utilizzato in liquidi organici, soluzioni saline, acidi e basi. Questo strumento può essere utilizzato come un termometro per esperimenti di chimica, fisica, biologia, scienze della terra e scienze ambientali. **Vedere anche:** Manuale dell'utente
Categoria	Sensore ambientale
Hub Collegamento	Adattatore TI-SensorLink per TI-Innovator™ Hub
Istruzioni di montaggio	N/D
Precauzioni	1. Attorcigliamento del cavo. Talvolta gli studenti possono attorcigliare o schiacciare il cavo nelle vicinanze dell'impugnatura del sensore. Con il tempo, i cavi potrebbero allentarsi interrompendo il funzionamento del sensore. 2. Surriscaldamento del sensore. Quanto utilizzato nei laboratori di chimica, talvolta gli studenti possono collocare il sensore su una piastra calda causando la vera e propria "cottura" dell'unità. 3. L'unità non è resistente all'acqua. L'acqua può penetrare all'interno del manico del sensore

Titolo	Rilevatore di temperatura in acciaio inossidabile Vernier
	danneggiandone i componenti elettronici. Durante la raccolta dei dati, immergere in acqua solo la parte del sensore in acciaio inossidabile.
Specifiche tecniche	Intervallo di temperatura: da -40 a 135 °C Temperatura massima tollerata dal sensore senza danni: 150 °C Risoluzione tipica: • 0,17 °C (da -40 a 0 °C) • 0,03 °C (da 0 a 40 °C) • 0,01 °C (da 40 a 100 °C) • 0,25 °C (da 100 a 135 °C) **Vedere anche:** per le specifiche complete, vedere qui.

HUB Comandi	
Oggetto del disegno	VERNIER
Sintassi dei comandi	
Codice Esempio:	

Azione desiderata	Esempio di codice

Leggere le temperature dal sensore Vernier collegato	`Send "CONNECT VERNIER 1 TO IN1 AS TEMPERATURE"`
	`Send "READ VERNIER 1"`
	`Get T`

Foglio dati del sensore pH

Titolo	**Sensore di pH Vernier**
Nome elemento TI	n/d
Codice ordine Vernier	PH-BTA
Incluso in	pH-metro
Quantità	1
Descrizione	Utilizzare il sensore di pH esattamente come un ph-metro tradizionale con il vantaggio di poter disporre di raccolta dei dati, grafica e analisi dei dati automatiche. **Vedere anche:** Manuale dell'utente
Categoria	Sensori ambientali
Hub Collegamento	Adattatore TI-SensorLink per TI-Innovator™ Hub
Istruzioni di montaggio	N/D
Precauzioni	Collocare l'elettrodo in una soluzione tampone a pH 4 o pH 7. Non deve mai essere conservato in acqua distillata. Se l'elettrodo viene involontariamente conservato a secco per un breve periodo di tempo, immergerne la punta nella soluzione di conservazione di KCl/tampone con pH 4 per almeno 8 ore prima di utilizzarlo.
Specifiche tecniche	• Tipo: sigillato, riempito con gel, corpo epossidico, Ag/AgCl • Tempo di risposta: 90% della lettura finale in 1 secondo • Intervallo di temperatura: da 5 a 80 °C (letture non compensate) • Intervallo: pH da 0 a 14 • Precisione: unità a +/- 0,2 pH • pH isopotenziale: pH 7 (punto in cui la temperatura non produce effetti)

Titolo	Sensore di pH Vernier
	• Valori di calibrazione predefiniti: pendenza: -3,838, intercetta: 13,720 • Diametro dell'albero: Densità ottica 12 mm **Vedere anche:** per le specifiche complete, vedere qui.

HUB Comandi	
Oggetto del disegno	VERNIER
Sintassi dei comandi	
Codice Esempio:	(see table below)

Azione desiderata	Esempio di codice

Leggere il valore del pH dal sensore Vernier collegato	`Send "CONNECT VERNIER 2 TO IN2 AS PH"`
	`Send "READ VERNIER 2"`
	`Get P`

Foglio dati del sensore di pressione del gas

Titolo	**Sensore di pressione del gas Vernier**
Nome elemento TI	n/d
Codice ordine Vernier	GPS-BTA
Incluso in	Sensore di pressione del gas
Quantità	1
Descrizione	Serve a monitorare le variazioni di pressione in un gas. L'intervallo è sufficientemente ampio da consentire la legge di Boyle, ma è anche sufficientemente sensibile da consentire l'esecuzione di esperimenti vapore-pressione o pressione-temperatura. Gli insegnanti di biologia possono utilizzare il sensore di pressione del gas per monitorare la traspirazione o la respirazione in un ambiente chiuso. **Vedere anche:** Manuale dell'utente
Categoria	Sensore ambientale
Hub Collegamento	Adattatore TI-SensorLink per TI-Innovator™ Hub
Istruzioni di montaggio	N/D
Precauzioni	L'elemento sensibile del sensore di pressione del gas viene danneggiato se entra in contatto diretto con un liquido.
Specifiche tecniche	• Intervallo di pressione: da 0 a 210 kPa (da 0 a 2,1 atm o da 0 a 1600 mm Hg) • Precisione: ±4 kPa • Pressione massima tollerata dal sensore senza danni permanenti: 4 atm • Elemento sensibile: Honeywell SSCMRNN030PAAA5 **Nota**: sono disponibili due varianti del sensore di pressione del gas.

Titolo	**Sensore di pressione del gas Vernier**
	La versione 1.3 del disegno per TI-Innovator™ Hub include le costanti di calibrazione per una delle due varianti. I programmi di riferimento mostrano come utilizzare il comando CALIBRATE per usare l'altro tipo di sensore di pressione del gas. **Vedere anche:** per le specifiche complete, vedere qui.

HUB Comandi	
Oggetto del disegno	VERNIER
Sintassi dei comandi	
Codice Esempio:	<table><tr><th>Azione desiderata</th><th>Esempio di codice</th></tr><tr><td>Leggere il valore della pressione del gas dal sensore Vernier collegato</td><td><code>Send "CONNECT VERNIER 1 TO IN1 AS PRESSURE"</code> <code>Send "READ VERNIER 1"</code> <code>Get P</code></td></tr></table>

Novità di Sketch v 1.4

È disponibile un’ulteriore variante del sensore di pressione del gas Vernier con costanti di calibrazione differenti.

Nuova parola chiave: **PRESSURE2**

Le costanti di calibrazione sono: 51.71 -25.86

Codice Esempio:	`Send "CONNECT VERNIER 1 TO IN 1 AS PRESSURE2"`
	`Send "READ VERNIER 1"`
	`Get P`

Foglio dati del sensore di forza

Titolo	Sensore di forza Vernier
Nome elemento TI	n/d
Codice ordine Vernier	DFS-BTA
Incluso in	Sensore di forza Vernier
Quantità	1
Descrizione	Sensore generico per la misurazione delle forze di compressione e di trazione. Due intervalli consentono di misurare forze da 0,01 newton fino a 50 newton. **Vedere anche:** Manuale dell'utente
Categoria	Sensore ambientale
Hub Collegamento	Adattatore TI-SensorLink per TI-Innovator™ Hub
Istruzioni di montaggio	Progettato per essere montato su un supporto circolare, un carrello, una barra o un tavolo delle forze in diversi modi. Utilizzare un'asta da 13 mm estendendola attraverso il foro sul sensore di forza. Serrare la vite ad alette in dotazione.
Precauzioni	N/D
Specifiche tecniche	Risoluzione intervallo ± 10 N: 0,01 N Risoluzione intervallo ± 50 N: 0,05 N **Nota**: su questo sensore è presente un interruttore di misurazione: – ± 10 N – ± 50 N **Vedere anche:** per le specifiche complete, vedere qui.

<table>
<tr><th colspan="2">HUB Comandi</th></tr>
<tr><td>Oggetto del disegno</td><td>VERNIER</td></tr>
<tr><td>Sintassi dei comandi</td><td></td></tr>
<tr><td>Codice Esempio:</td><td>
<table>
<tr><th>Azione desiderata</th><th>Esempio di codice</th></tr>
<tr><td>Leggere il valore della forza dal sensore Vernier collegato nella configurazione 10 N</td><td><code>Send "CONNECT VERNIER 2 TO IN2 AS FORCE"</code>
<code>Send "READ VERNIER 2"</code>
<code>Get F</code></td></tr>
<tr><td>Leggere il valore della forza dal sensore Vernier collegato nella configurazione 50 N
(Tenere presente che il comando CONNECT include FORCE50)</td><td><code>Send "CONNECT VERNIER 2 TO IN2 AS FORCE50"</code>
<code>Send "READ VERNIER 2"</code>
<code>Get F</code></td></tr>
</table>
</td></tr>
</table>

Foglio dati dell'accelerometro low-g

(Codice ordine - LGS-BTA)

Titolo	**Accelerometro low-g**
Nome elemento TI	n/d
Codice ordine Vernier	LGA-BTA
Incluso in	Accelerometro low-g
Quantità	1
Descrizione	L'accelerometro low-g può essere utilizzato per una vasta gamma di esperimenti e dimostrazioni, sia all'interno del laboratorio che fuori. **Vedere anche:**Manuale dell'utente
Categoria	Sensore ambientale
Hub Collegamento	Adattatore TI-SensorLink per TI-Innovator™ Hub
Istruzioni di montaggio	N/D
Precauzioni	
Specifiche tecniche	**Vedere:** per le specifiche complete, vedere qui.

Foglio dati del sensore di luce

(Codice ordine - LS-BTA)

Titolo	**Rivelatore di luce**
Nome elemento TI	n/d
Codice ordine Vernier	LS-BTA
Incluso in	Rivelatore di luce
Quantità	1
Descrizione	Il sensore di luce può essere utilizzato per effettuare misurazioni dell'intensità della luce in una vasta gamma di situazioni. **Vedere anche:**Manuale dell'utente
Categoria	Sensore ambientale
Hub Collegamento	Adattatore TI-SensorLink per TI-Innovator™ Hub
Istruzioni di montaggio	N/D
Precauzioni	Il sensore di luce è sufficientemente sensibile per rilevare lo sfarfallio delle lampade fluorescenti fino a 60 o 120 Hz, il quale può interferire negli esperimenti con le luci. Se si pensa che si stia verificando questa interferenza, provare quanto indicato di seguito: • Eliminare prima tutte le fonti di luce artificiali (tranne le torce alimentate a batteria) e provare nuovamente l'esperimento. • Quindi, provare il sensore di luce nella posizione in cui si intende utilizzarlo. Impostare il campionamento su 1.000 punti/secondo per 0,1 secondi. Se il problema è lo sfarfallio, si osserverà una drastica variazione nell'intensità della luce con un periodo pari a 60 o 120 Hz (50 o 100 Hz in Paesi fuori dall'America del Nord). • Se lo sfarfallio è un problema, impostare la velocità

Titolo	Rivelatore di luce
	di campionamento su un numero che non sia un fattore di 60. Ad esempio, utilizzare 30, 20 o 10 campioni è un’opzione peggiore rispetto all’utilizzare 17, 23 e 27 campioni.
Specifiche tecniche	**Vedere:** per le specifiche complete, vedere qui. Valori di calibrazione predefiniti: 0-600 lux pendenza: 154 lux/V intercetta: 0 lux 0-6000 lux pendenza: 1692 lux/V intercetta: 0 lux 0-150000 lux pendenza: 38424 lux/V intercetta: 0 lux

Foglio dati del sensore di energia Vernier

(Codice ordine - VES-BTA)

Titolo	**Sensore di energia Vernier**
Nome elemento TI	n/d
Codice ordine Vernier	VES-BTA
Incluso in	Sensore di energia
Quantità	1
Descrizione	Il sensore di energia Vernier consente agli studenti di misurare facilmente la corrente e la tensione. I terminali della sorgente si collegano alle fonti di output dell’energia come le turbine eoliche modello o i pannelli solari, mentre i terminali di carico si collegano ai carichi come LED, pompe d’acqua, resistori o carichi variabili. **Vedere anche:**Manuale dell’utente
Categoria	Sensore ambientale
Hub Collegamento	Adattatore TI-SensorLink per TI-Innovator™ Hub
Istruzioni di montaggio	N/D
Precauzioni	
Specifiche tecniche	**Vedere:** per le specifiche complete, vedere qui.

TI-RGB Array

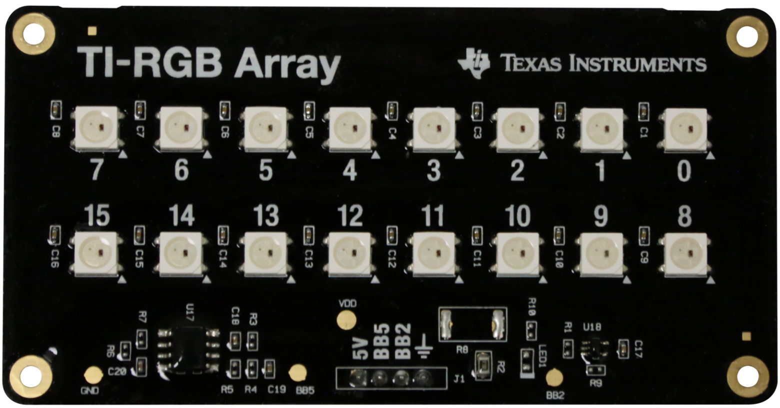
Che cos'è il TI-RGB Array?

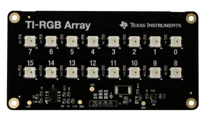
TI-RGB Array è un accessorio all'Hub TI-Innovator™.

TI-RGB Array ha 16 LED RGB programmabili.

Più applicazioni

- Serra intelligente
- Contatore binario
- Progetti STEAM
- Lezioni di coding



TI-RGB Array – Design e marchi industriali	
Vista dall'alto del TI-RGB Array.	
Vista inferiore - Identificazione dell'etichetta	

Requisiti per il TI-RGB Array:

Hardware:

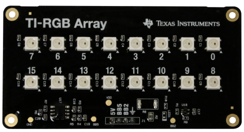
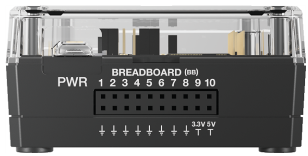
Estensione TI-RGB Array all’Hub TI-Innovator™

Usa Hub Sketch v1.4 o successiva

Collegare il TI-RGB Array

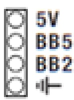
Esegui questa serie di passaggi rispettandone l'ordine per collegare e utilizzare il TI-RGB Array.

Collega il TI-RGB Array all’Hub TI-Innovator™

TI-RGB Array	**Cavo in dotazione**	**Hub TI-Innovator™**
		

PASSAGGI

1. Collega un'estremità del cavo in dotazione alla porta etichettata del TI-RGB Array.



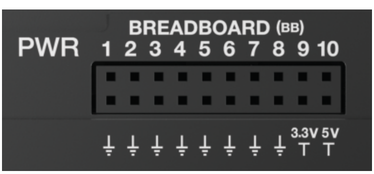
2. Collega i fili corrispondenti ai pin utilizzabili sull’Hub etichettato:

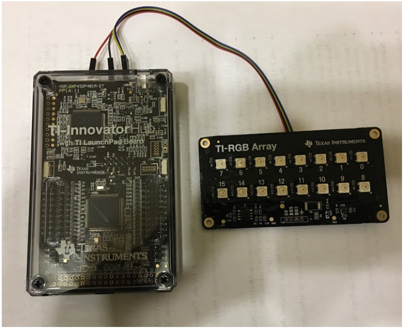
   Rosso: `5V` - alimentazione

   Blu : `BB5` - uscita analogica

   Giallo: `BB2` - segnale SPI

   Nero: ⏚ GND - messa a terra





Collegamento di TI-Innovator™ Hub a una calcolatrice grafica

La TI-Innovator™ Hub si collega via cavo USB a un computer o a una calcolatrice grafica. Il collegamento consente all’Hub di essere alimentato e di scambiare dati con l’host.

Vedere i dettagli completi (page 5).

Comandi TI-RGB Array

Prerequisito: Usa prima il comando invia "Connect RGB"

Il comando "**CONNECT RGB**" deve essere usato per primo quando si utilizza il TI-RGB Array. Il comando "**CONNECT RGB**" configura il software dell'Hub TI-Innovator™ Hub per il funzionamento con il TI-RGB Array.

Stabilisce la connessione ai vari slot binari dei led sul TI-RGB Array, da 0 fino a 15 LED RGB. Riazzera anche i vari contatori e valori dei sensori.

Per comandi agguntivi visita: education.ti.com/eguide

Esempio di codice

CONNECT RGB

Comando:	CONNECT RGB
Sintassi dei comandi:	CONNECT RGB
Codice Esempio:	`Invia "CONNECT RGB"`
Intervallo:	N/D
Descrivi:	Il comando "**CONNECT RGB**" configura il software dell'Hub TI-Innovator™ Hub per il funzionamento con il TI-RGB Array.
Risultato:	Collega il TI-RGB Array all'Hub TI-Innovator™. Il TI-RGB Array è pronto per essere programmato
Digita o Componenti indirizzabili:	Tutti i componenti del TI-RGB Array.

Comando:	CONNECT RGB AS LAMP
Sintassi del comando:	**CONNECT RGB AS LAMP**
Esempio di codice:	`Send "CONNECT RGB AS LAMP"`
Intervallo:	N/D
Descrizione:	Questo comando consente di attivare la modalità "luminosità elevata" del TI-RGB Array ammesso che una fonte di alimentazione esterna, ad esempio la batteria USB, sia collegata alla porta **PWR**.

Comando:	CONNECT RGB AS LAMP
	Nota: il comando “AS LAMP” deve essere digitato.
Risultato:	Il TI-RGB Array è ora configurato in modalità luminosità elevata. Se l’alimentazione esterna non è collegata, il comando “**AS LAMP**” non produrrà alcun effetto, ossia la luminosità rimarrà al livello predefinito. Tenere inoltre presente che un errore verrà indicato con un segnale acustico.
Tipo o componente indirizzabile:	Tutti i componenti del TI-RGB Array. **Vedere anche:** Nuovi comandi da utilizzare con il TI-RGB Array

SET RGB

Comando:	SET RGB n r g b
Comando Sintassi:	SET RGB n r g b SET RGB eval(n) r g b
Codice Esempio:	`Invia "SET RGB 1 255 0 255"`
Intervallo:	0-15 per 'n', 0-255 per r,g,b
Descrivi:	Il comando SET RGB controlla la luminosità e il colore di ciascun LED RGB nel TI-RGB Array
Risultato:	Il LED specifico si illumina con il colore specificato
Digita o Indirizzabile Componente:	Tutti i componenti del TI-RGB Array **Vedere anche:** Nuovi comandi da utilizzare con il TI-RGB Array **Vedere anche:** SET RGB ALL

SET RGB ALL

Comando:	SET RGB ALL r g b
Comando Sintassi:	**SET RGB ALL r g b**
Codice Esempio:	`SET RGB ALL 255 0 255`
	`SET RGB ALL 255 0 0`

Comando:	SET RGB ALL r g b
	`SET RGB ALL eval(R) eval(G) eval(B)`
	`SET RGB ALL 0 0 0`
Intervallo:	
Descrivi:	Per controllare tutti i LED con un singolo comando utilizzare: SET RGB ALL r g b
Risultato:	Controllo di tutti i LED con un singolo comando
Digita o Indirizzabile Componente:	Tutti i componenti del TI-RGB Array

READ RGB

Comando:	READ RGB
Comando Sintassi:	Invia "READ RGB"
Codice Esempio:	`Invia "READ RGB"` `Ottieni c`
Intervallo:	0-15 per 'n', 0-255 per r,g,b
Descrivi:	Dà il valore di quanto consumato attualmente in mA dal TI-RGB Array
Risultato:	
Digita o Indirizzabile Componente:	Tutti i componenti del TI-RGB Array

Precauzioni generali

TI-RGB Array

- Non esporre il TI-RGB Array a temperature superiori ai 60˚C (140˚F).
- Usare unicamente il cavo a nastro fornito con il TI-RGB Array.
- Quando inserisci il cavo a nastro nei TI-RGB Array connettori, assicurati che il pin con filo rosso (scuro) sia inserito nel foro 5v.

- Usa il TI-RGB Array a una distanza non inferiore ai 20 cm dai tuoi occhi.
- Riposa periodicament gli occhi concentrandoti su un oggetto ad almeno 1,5 m di distanza.

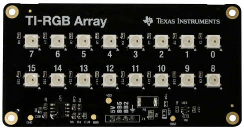
Scheda dati TI-RGB Array

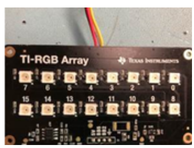
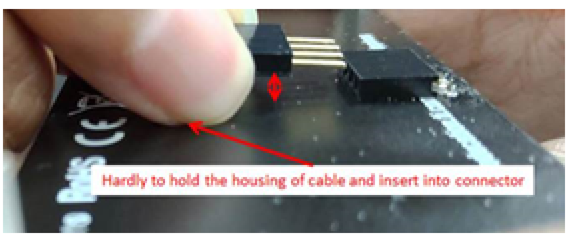
La scheda dati del TI-RGB Array include le seguenti informazioni: il nome e il numero del prodotto, una breve descrizione, un'immagine del prodotto, le specifiche, come il componente si collega all'Hub TI-Innovator™ e i comandi dell'Hub con esempi di codice semplici.

Collegamenti argomento

- Scheda dati TI-RGB Array
- Cavo basetta per TI-RGB Array

Scheda dati TI-RGB Array

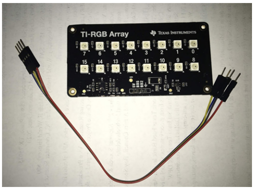


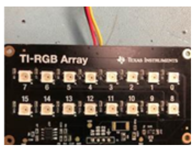
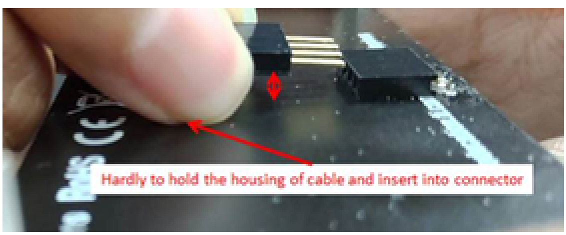
Titolo	**TI-RGB Array**
Nome elemento TI	STEMRGB/BK/A
Incluso in	TI-RGB Array
Quantità	1
Descrizione	Accessorio per l’Hub TI-Innovator™. • 16 LED RGB programmati individualmente • Cavi M-M collegano l’Array alla porta basetta dell’Hub – Rosso: 5V – alimentazione – Blu: BB5 – uscita analogica – Giallo: BB2 – segnale SPI – Nero: GND – messa a terra • L’Hub misura il consumo attuale dei LED
Categoria	Accessorio
Hub Collegamento	 
Istruzioni di montaggio	N/D 
Precauzioni	**Vedere:** Precauzioni generali TI-RGB Array
Specifiche tecniche	**Vedere:** TI-RGB Array

HUB Comandi	
Oggetto del disegno	RGB Array
Sintassi dei comandi	`Invia "COLLEGA RGB"`
Codice Esempio:	

Azione desiderata	Esempio di codice
Collega il TI-RGB Array all'Hub TI-Innovator™. Il TI-RGB Array è pronto per essere programmato	`Invia "COLLEGA RGB"`

Scheda dati cavo basetta per TI-RGB Array



Titolo	**Cavo basetta per il TI-RGB Array**
Nome elemento TI	STEMRGB/CA/A
Incluso in	TI-RGB Array
Quantità	1
Descrizione	• Cavi M-M collegano l’Array alla porta basetta dell’Hub – Rosso: 5V – alimentazione – Blu: BB5 – uscita analogica – Giallo: BB2 – segnale SPI – Nero: GND – messa a terra
Categoria	Accessorio
Hub Collegamento	 
Istruzioni di montaggio	N/D 
Precauzioni	**Vedere:** Precauzioni generali TI-RGB Array
Specifiche tecniche	**Vedere:** TI-RGB Array

Risoluzione dei problemi

Questa sezione descrive alcuni problemi che si possono riscontrare e offre suggerimenti sulla loro risoluzione.

Se è necessaria ulteriore assistenza, contattare TI-Cares.

Risoluzione dei problemi di TI-Innovator™ Hub

Cosa si può fare se la calcolatrice grafica TI CE o il palmare TI-Nspire™ CX non riconosce TI-Innovator™ Hub? Quando si collega la calcolatrice grafica TI CE o il palmare TI-Nspire™ CX a TI-Innovator™ Hub non viene visualizzato alcun LED verde.

- Assicurarsi di aver acceso la calcolatrice.
- Se per il collegamento a una calcolatrice viene utilizzato un cavo USB di collegamento tra unità (da Mini-A a Mini-B), verificare di collegare l'estremità "B" del cavo alla porta "DATA B" nella parte inferiore dell'Hub. Se le estremità del cavo vengono invertite, l'Hub non riceve corrente.
- Assicurarsi che il sistema operativo della calcolatrice sia aggiornato con la versione più recente.
- Verificare che l'estremità del cavo USB collegato alla calcolatrice sia inserito correttamente.
- Scollegare il cavo USB da TI-Innovator™ Hub, attendere 3 secondi e ricollegare il cavo USB.

Cosa si può fare se il software per computer TI-Nspire™ CX non riconosce TI-Innovator™ Hub?

- Assicurarsi che il software TI-Nspire™ CX sia aggiornato con la versione più recente. Con la versione più recente, viene installato un driver che consente al computer di riconoscere TI-Innovator™ Hub.
- Assicurarsi che TI-Innovator™ Hub sia collegato mediante la porta "DATA B" sulla porta TI-Innovator™.
- Scollegare il cavo USB da TI-Innovator™ Hub, attendere 3 secondi e ricollegare il cavo USB.
- Se non si utilizza il cavo USB fornito in dotazione con TI-Innovator™ Hub, è possibile che il cavo USB sia solo un cavo di alimentazione e non un cavo per dati e alimentazione. Provare con un cavo USB diverso.

Come si spegne l'Hub?

- Spegnere il computer o la calcolatrice host.

  \- OPPURE -

- Scollegare il cavo USB.

Che cosa indica il lampeggiamento del LED di errore e l'emissione di un segnale acustico da parte del diffusore?

Se il LED di errore lampeggia e il diffusore emette un segnale acustico, significa che si è verificato un errore durante l'invio dei comandi a TI-Innovator™ Hub. Rivedere i comandi di esempio per i componenti di bordo, dei moduli di I/O e della basetta sperimentale per informazioni su come modificare il programma.

Perché sul computer viene installato un driver per Silicon Labs CP210x quando si collega TI-Innovator™ Hub?

TI-Innovator™ Hub utilizza il chip Silicon Labs per l'interfaccia USB. Il driver consente al software del computer desktop di comunicare con l'hub. Questa installazione viene eseguita la prima volta che si collega TI-Innovator™ Hub al computer.

Risoluzione dei problemi dei componenti dell'Hub di bordo

Il programma in dotazione non funziona quando utilizzato unitamente al componente di bordo. Com'è possibile verificare che il componente di bordo non sia danneggiato?

- Scaricare il programma di test ed eseguirlo per testare il componente di bordo.
- Assicurarsi che i valori utilizzati dal programma rientrino negli intervalli supportati dai componenti di bordo.
  - RGB: intervallo da 0 a 255 per il livello di intensità
  - Diffusore: intervallo da 40 a 4.000 Hz

Perché l'RGB di bordo si spegne ogni volta che il diffusore emette un suono? Perché non è possibile controllare l'RGB di bordo durante la riproduzione del suono?

Non è possibile utilizzare contemporaneamente i comandi COLOR/RGB e SOUND/SPEAKER. I programmi degli utenti devono attendere il completamento del comando SOUND/SPEAKER prima di inviare il comando COLOR/RGB a TI-Innovator™ Hub.

Perché il sensore di intensità luminosa di bordo restituisce letture diverse anche se la sorgente luminosa non è cambiata? Le letture dell'intensità luminosa alternano i valori massimi e i valori minimi quando dovrebbero invece restituire un valore costante.

Le sorgenti luminose dei LED producono uno sfarfallio a velocità elevate. Anche se l'occhio umano non può rilevare questo sfarfallio, il sensore di intensità luminosa lo registra e riporta i rispettivi valori nella lettura.

Risoluzione dei problemi di TI-Innovator™ Rover

Il Rover in dotazione non funziona correttamente. Perché?

- Verificare che sia caricato.
- Assicurarsi che sia acceso.
- Assicurarsi che tutti i cavi siano collegati.
  - Assicurarsi che il cavo della basetta sperimentale sia configurato correttamente (il cavo rosso deve essere collocato sul lato corretto).
  - Assicurarsi che tutti i pin della basetta sperimentale siano dritti.
- Assicurarsi di disporre del sketch più recente.
- Assicurarsi di disporre del sistema operativo più recente.
- Eseguire il programma di test.
- Non collocare altri oggetti, ad eccezione della calcolatrice, sulla parte superiore del Rover.

Il Rover in dotazione non si sposta o non si sposta correttamente. Perché?

- Se si utilizza il portapenne, assicurarsi che la penna non sia inserita così lontano da sollevare il Rover.
- Pulire le rotelle.
- Per migliori ottimali, utilizzare una superficie liscia e piatta.
- Verificare che l'orientamento corrisponda alle direzioni previste dal programma.

Il Rover non ha disegnato la forma prevista. Perché?

- Il Rover non è uno strumento di sketch di precisione. Con forme specifiche non è possibile garantire un livello di precisione.
- Al momento della rotazione, il Rover potrebbe subire una variazione di +/- 0,5 gradi. Più grande è il numero di segmenti (o rotazioni), maggiore sarà la variazione composta.
- Le superfici migliori per il Rover sono quelle piatte e lisce (non tappeti o piastrelle).

Qual è il numero di segmenti o rotazioni consigliato per disegnare la forma prevista?

Sono disponibili due metodi per il sketch di forme (o funzioni) con il Rover.

Questi due metodi presentano livelli di precisione diversi e possono restituire risultati diversi anche per la stessa forma generica (ad esempio, un ottagono).

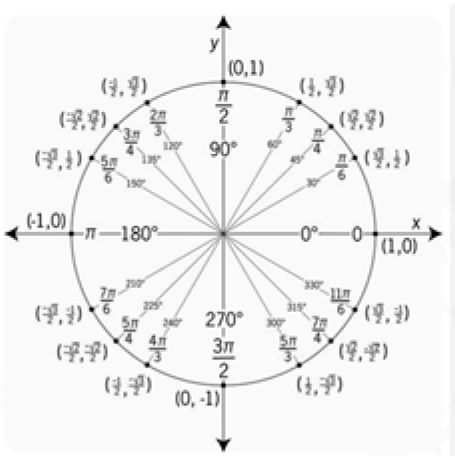
Metodo 1 utilizzando i comandi FORWARD/BACKWARD/LEFT/RIGHT, il Rover si sposterà in base alla distanza e all'angolo specificati. Il movimento angolare potrebbe non essere preciso e varia a seconda della superficie e della presenza del pennarello.

Metodo 2: utilizzando i comandi "TO XY", "TO POLAR", il Rover si sposterà su coordinate specifiche con rotazioni più precise. Dopo diversi segmenti, si accumuleranno comunque piccoli errori anche con l'uso di questi comandi. Per funzioni e forme che utilizzano una griglia di grandi dimensioni e/o più di 18 segmenti, il sketch potrebbe non corrispondere alla forma prevista.

Il Rover in dotazione esegue un numero maggiore o minore di rotazioni rispetto a quello previsto. Perché?

Sono disponibili due comandi diversi per la rotazione del Rover.

- Comandi RV LEFT/RV RIGHT: questi comandi comunicheranno al Rover di ruotare in base all'angolo specificato in relazione alla posizione attuale del Rover.
- RV TO ANGLE: questo comando eseguirà lo spostamento in base all'angolo specificato sulla circonferenza unitaria.



Esempi:

RV LEFT 30

RV LEFT 45

Questi comandi eseguiranno la rotazione del Rover in base a un angolo di 75 gradi.

Per confronto

RV TO ANGLE 30

RV TO ANGLE 45

Questi comandi eseguiranno la rotazione del Rover in base a un angolo di 45 gradi.

Assicurarsi che il programma utilizzi il comando di rotazione che corrisponde allo spostamento previsto per il Rover.

Come unità predefinita, questi comandi utilizzano i gradi anche se l'impostazione della calcolatrice è radianti.

È possibile specificare i comandi RADIANS o GRADS (gradi centesimali) per le rotazioni del Rover dal menu "Hub -> Rover (RV) -> RV Settings (Impostazioni RV)".

Il Rover non si sposta in base alla distanza prevista. Perché?

Il Rover utilizza un'unità predefinita di 10 cm.

Ciò significa che il comando RV FORWARD 1 sposta in avanti il Rover di 10 cm.

È l'equivalente dei comandi "RV FORWARD 1 UNITS" e "RV FORWARD 0.1 M".

Per spostare il Rover in base a distanze specifiche, è possibile utilizzare l'impostazione M per specificare metri.

Il pennarello non risulta stabile all'interno dell'apposito supporto. Perché?

Il supporto è adatto per pennarelli sottili standard o pennarelli cancellabili a secco. Il supporto è stato progettato in modo che sia la forza di gravità a mantenere in posizione il pennarello. La punta del pennarello rimane nella posizione corretta anche se l'altra estremità del pennarello potrebbe spostarsi.

Verso quale direzione punta il Rover quando si avvia un programma?

La posizione predefinita del Rover si trova sull'origine di un punto del diagramma cartesiano sull'asse x positivo.

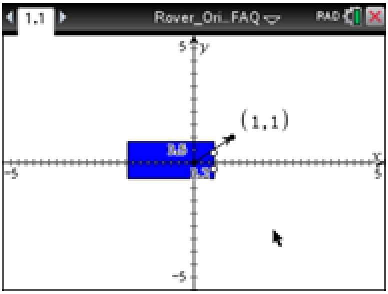
L'orientamento originale è: posizione (0,0); direzione 0 gradi (est - puntando verso l'asse x positivo).

TO XY esegue prima la rotazione in base all'angolo corretto, quindi esegue lo spostamento in linea retta verso il punto.

Esempio:

TO XY 1 1 esegue una rotazione di 45 gradi a sinistra, quindi uno spostamento di unità sqrt(2) (a 10 cm/unità = 14,14 cm).

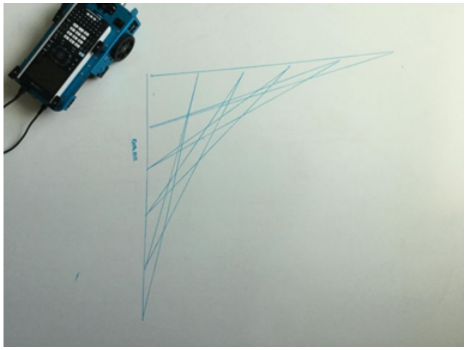
Vedere anche Rover>Configurazione>SET RV.POSITION.


1.1
Rover_Ori...FAQ
(1,1)

Con che comandi XY o Polar si potrebbe iniziare?

Table 1: Esempio 1:

```
Send "CONNECT RV"
Send "RV TO XY 0 0"
Send "RV TO XY 5 0"
Send "RV TO XY 0 0"
Send "RV TO XY 0 5"
Send "RV TO XY 0 0"
Send "RV TO XY 1 0"
Send "RV TO XY 0 5"
Send "RV TO XY 0 4"
Send "RV TO XY 2 0"
Send "RV TO XY 3 0"
Send "RV TO XY 0 3"
Send "RV TO XY 0 2"
Send "RV TO XY 4 0"
Send "RV TO XY 5 0"
Send "RV TO XY 0 1"
```



Perché il programma Rover viene eseguito in modo irregolare?

I comandi del Rover rientrano in due categorie:

1. Esecuzione in coda: tutti i comandi per lo spostamento del Rover, FORWARD, BACKWARD, LEFT, RIGHT, ANGLE, vengono eseguiti in coda su TI-Innovator Hub. Potrebbero essere eseguiti in un secondo momento.
2. Esecuzione immediata: altri comandi, come quelli per la lettura dei sensori o l'impostazione del LED RGB sul Rover, vengono eseguiti immediatamente.

Ciò significa che alcune istruzioni del programma verranno eseguite prima delle istruzioni il cui ordine è precedente all'interno del programma, soprattutto se gli ultimi comandi fanno parte della famiglia in coda.

Ad esempio, nel programma di seguito, il LED RGB si illuminerà di ROSSO prima che il Rover si arresti:

`Send "SET RV.COLOR 255 0 255"` - esecuzione immediata

`Send "RV FORWARD 5"` - comando in coda

`Send "RV LEFT 45"` - comando in coda

`Send "RV RIGHT 90"` - comando in coda

`Send "SET RV.COLOR 255 0 0"` - esecuzione immediata

Perché il Rover continua a funzionare anche se sulla calcolatrice viene indicato "Done"?

Ciò si può verificare se i comandi si trovano in coda per un'esecuzione successiva. La calcolatrice riporta "Done" perché i programmi hanno terminato di inviare tutti i comandi a TI-Innovator Hub. L'Hub eseguirà i comandi per il controllo del Rover anche se il programma della calcolatrice è stato completato.

Quando si collega il Rover, la batteria non indica alcuna carica. Perché?

Anche se generalmente lo stato di carica della batteria viene visualizzato immediatamente, può essere necessario un minuto prima che venga visualizzato.

Il Rover ha smesso di funzionare e non è possibile accenderlo. Cosa si può fare?

Caricare il Rover per un paio di minuti e attendere che venga visualizzato lo stato della batteria.

Il Rover è stato spento, ma il programma è ancora in esecuzione e alcuni elementi del Rover sono ancora in funzione. Perché?

Per disattivare completamente il Rover, SPEGNERE l'interruttore di alimentazione e scollegare il cavo USB dalla calcolatrice grafica.

Perché il Rover non si sposta in linea retta?

Ciò può accadere se i due motori non presentano la calibrazione interna simile. Siamo a conoscenza del problema e stiamo lavorando per trovare una soluzione che renderemo disponibile in un aggiornamento del sketch dell'Hub.

La calcolatrice grafica non si adatta al Rover.

Assicurarsi che le linguette siano orientate nella direzione corretta. Le linguette contrassegnate con "CE" e "CX" devono essere utilizzate rispettivamente per la serie di calcolatrici TI84Plus CE e la serie TI-Nspire CX.

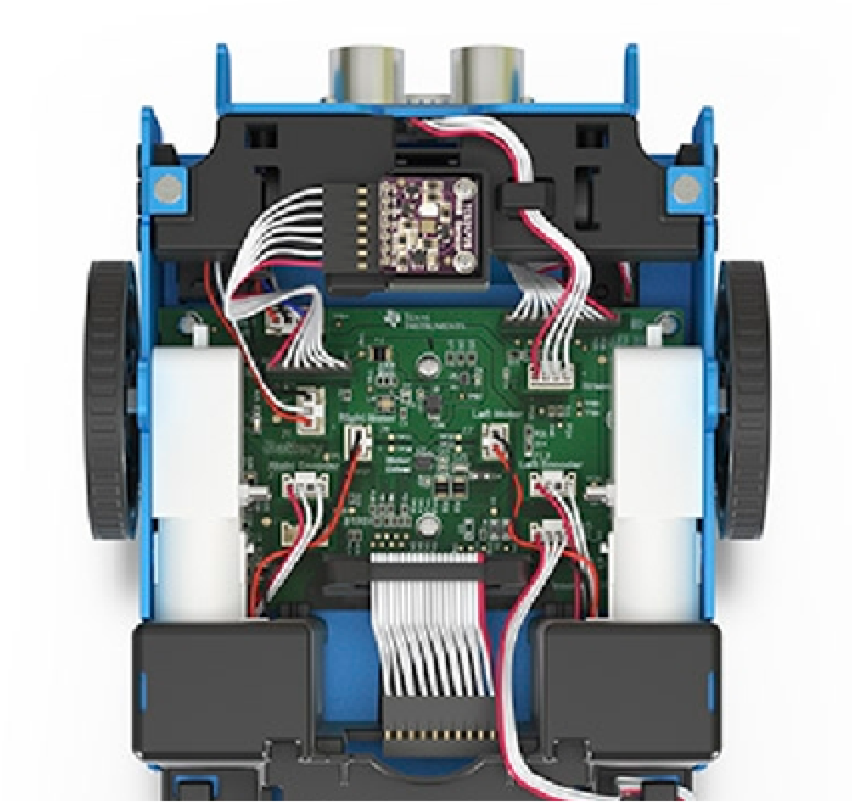
Il sensore integrato non funziona. Non restituisce i risultati previsti o non restituisce alcun dato.

Fare riferimento ai programmi di test.

Verificare che non vi siano collegamenti allentati.

Cosa si può fare se uno degli studenti ha scollegato tutti i cavi?

Fare riferimento allo schema elettrico di seguito.



Il Rover in dotazione si sposta in modo più lento o ruota in modo irregolare?

- Verificare che la rotella non sia sporca di detriti.
- Utilizzare aria compressa per pulire eventuale sporcizia.
- È possibile che le gomme si siano staccate. Verificare che siano posizionate correttamente sulla rotella.
- Si consiglia di posizionare lo strumento su una superficie liscia e piatta.

Le gomme si sono staccate.

- Verificare che siano posizionate correttamente sulla rotella.

I pin della basetta sperimentale sembrano piegati. È ancora possibile utilizzarli?

Riallineare i pin in base alla configurazione originale prima di collegarli all'Hub.

A cosa servono i comandi TO XY e TO POLAR? Sembra che non producano alcun effetto sul Rover.

Questi comandi verranno implementati in una versione futura del sketch dell'Hub.

Qual è il modo migliore per iniziare? Quali sono le operazioni che è possibile svolgere con il Rover?

- Demo: i programmi per svolgere attività. Effettuare una guida di prova.
- Programmi di test: provare con un componente alla volta. Verificare che funzionino correttamente.

Risoluzione dei problemi del modulo di I/O

Cosa si può fare se il modulo di I/O del LED bianco non funziona?

La seguente procedura di risoluzione dei problemi consente di determinare la presenza di un problema nel modulo di I/O del LED bianco.

- Assicurarsi che il LED sia inserito correttamente nella presa.
  - Inserire il LED nella presa tenendo conto che la gamba più lunga (connettore) è positiva (anodo). Se entrambi i connettori sono della stessa lunghezza, il connettore adiacente al bordo piatto sull'alloggiamento del LED corrisponde al connettore negativo (catodo).
- Scaricare il programma di test ed eseguirlo per testare il componente del modulo del LED bianco.
- Assicurarsi che il modulo di I/O sia stato collegato alla porta corretta richiesta dal programma.

Cosa si può fare se il modulo di I/O del sensore di luce analogico non funziona?

La seguente procedura di risoluzione dei problemi consente di determinare la presenza di un problema nel modulo di I/O del sensore di luce analogico.

- Scaricare il programma di test ed eseguirlo per testare il componente del modulo di I/O del sensore di luce analogico.
- Assicurarsi che il modulo di I/O sia stato collegato alla porta corretta richiesta dal programma.

Cosa si può fare se il modulo di I/O del motore a vibrazione non funziona?

La seguente procedura di risoluzione dei problemi consente di determinare la presenza di un problema nel modulo di I/O del motore a vibrazione.

- Scaricare il programma di test ed eseguirlo per testare il componente del modulo di I/O del motore a vibrazione.
- Assicurarsi che il modulo di I/O sia stato collegato alla porta corretta richiesta dal programma.

Cosa si può fare se il modulo di I/O del servomotore non funziona?

La seguente procedura di risoluzione dei problemi consente di determinare la presenza di un problema nel modulo di I/O del servomotore.

- Scaricare il programma di test ed eseguirlo per testare il componente del modulo di I/O del servomotore.

- Assicurarsi che il modulo di I/O sia stato collegato a OUT3 e che il programma in uso stia chiamando OUT3.
- Per disporre di un'alimentazione esterna, il servomotore richiede TI-Innovator™ Hub. Il connettore **PWR** sull'Hub consente di collegare un alimentatore ausiliario. È possibile utilizzare il caricatore da parete TI o la batteria esterna. Viene richiesta l'alimentazione esterna se si utilizza TI-Innovator™ Hub con una calcolatrice grafica o il software per computer TI-Nspire™ CX.
- Con il passare del tempo, potrebbe essere necessario ricalibrare il servomotore. Procedura di ricalibrazione:
  - Collegare l'alimentazione esterna a TI- Innovator HUB.
  - Collegare il servomotore a OUT3.
  - Inviare il comando "CONNECT SERVO 1 TO OUT3".
  - Inviare il comando "SET SERVO 1 CW 0 TIME 100" (imposta la velocità su 0, se necessario è possibile aumentare il valore temporale).
  - Se il servomotore non si sposta, significa che è già stato calibrato. Se il servomotore si sposta, utilizzare un cacciavite per spostare il potenziometro sul retro del motore finché non si arresta.

Cosa si può fare se il modulo di I/O del ranger ultrasonico non funziona?

La seguente procedura di risoluzione dei problemi consente di determinare la presenza di un problema nel modulo di I/O del ranger ultrasonico.

- Scaricare il programma di test ed eseguirlo per testare il componente del modulo di I/O del ranger ultrasonico.
- Assicurarsi che il modulo di I/O sia stato collegato alla porta corretta richiesta dal programma.

Perché il modulo di I/O del sensore di intensità luminosa di bordo e del sensore di luce analogico restituiscono letture leggermente diverse?

A seconda della sua posizione, la lettura del sensore integrato in TI-Innovator™ Hub può risultare leggermente diversa rispetto a quella del sensore di luce analogico.

Risoluzione dei problemi di TI-SensorLink

- TI-SensorLink **non** è una soluzione per la raccolta dei dati. Per la raccolta e l'analisi dei dati, si consiglia di utilizzare i rilevatori collegati USB o la basetta Lab.
- I comandi dell'Hub per TI-SensorLink con i sensori analogici Verner **non** sono attualmente inclusi nell'app Hub (serie CE) o nel menu Hub (TI-Nspire™ CX).
- I nuovi comandi e le nuove parole chiave devono essere digitate OPPURE copiate da un programma esistente. Tenere presente che eventuali errori tipografici nelle parole chiave attiverà un'indicazione di errore nel sketch.

Risoluzione dei problemi relativi alla programmazione con TI-Basic

Perché il mio programma mi segnala un errore di sintassi?

- Se hai copiato il codice da una sorgente esterna o un editor di testo, il codice potrebbe contenere virgolette "curve" (“...”) laddove è necessario utilizzare le virgolette diritte ("..."). Occorrerà sostituire alcune o tutte le virgolette curve.
- Le regole di sintassi della calcolatrice grafica TI CE e della tecnologia TI-Nspire™ CX sono leggermente diverse. Potrebbe essere necessario modificare il codice creato in origine per una piattaforma, affinché possa funzionare su un’altra.
- Sulla calcolatrice grafica TI CE assicurarsi che non ci sia un carattere di spaziatura al termine di una riga di codice. Per trovare questi spazi finali in una riga, spostare il cursore sulla riga e premere [2nd], quindi il tasto freccia destra. Gli spazi adiacenti nel codice possono ugualmente causare un errore di sintassi.

Come si può arrestare un programma che non risponde?

- Calcolatrice grafica TI CE: premere il tasto ON.
- Palmare TI-Nspire™ CX: tenere premuto il tasto Home/ON, quindi premere più volte INVIO.
- Windows®: Tenere premuto il tasto F12 e premere Invio più volte.
- Mac®: Tenere premuto il tasto F5 e premere Invio più volte.

TI-SmartView CE non mostra i comandi dell'Hub nel menu di programmazione.

Assicurarsi di utilizzare la versione più recente del software TI-SmartView CE, ovvero la versione 5.2. Questa versione installa l'app Hub che include i comandi di programmazione per TI-Innovator™ Hub.

Perché il software TI Connect™ CE non mostra i comandi dell'Hub?

I comandi di TI-Innovator™ Hub sono stati aggiunti al software TI Connect™ CE. Aggiornare il software con la versione più recente.

Il programma non restituisce alcun errore di sintassi, tuttavia il LED di errore segnale un errore.

Il LED di errore lampeggia quando è presente un errore nella struttura dei comandi e quando il sketch non è in grado di elaborare i comandi. Rivedere i comandi di esempio per i componenti di bordo, dei moduli di I/O e della basetta sperimentale per informazioni su come modificare il programma.

Risoluzione dei problemi di TI-Innovator™ Sketch

Perché si verifica un errore quando si tenta di aggiornare TI-Innovator™ Sketch?

- Per l'aggiornamento del sketch, assicurarsi di utilizzare un cavo USB da Standard A a Micro e non un cavo USB da Standard A a Mini-B. Collegare la micro estremità del cavo al connettore PWR nella parte superiore dell'Hub.

TI-Innovator™ Hub sembra acceso, tuttavia non è in grado di comunicare con lo strumento di aggiornamento.

- Potrebbe trattarsi di un problema relativo al cavo. Alcuni cavi USB sono adatti solo all'alimentazione e non alla trasmissione di dati.
- Assicurarsi di utilizzare il cavo fornito in dotazione con TI-Innovator™ Hub.

Sono necessari privilegi di amministratore sul computer in uso per l'aggiornamento del sketch?

Sì.

Risoluzione dei problemi della batteria esterna

Sembra che la batteria esterna non stia fornendo corrente a TI-Innovator™ Hub.

- Premere il pulsante On/Off per assicurarsi che la batteria sia accesa. La batteria esterna si spegne entro 3 minuti se non viene collegata a TI-Innovator™ Hub.
- Verificare che la batteria esterna sia caricata. Premere il pulsante On/Off. Se le spie del LED non si accendono, significa che la batteria esterna deve essere caricata.

Precauzioni generali per la tecnologia TI-Innovator™

In questa sezione vengono descritte le precauzioni generali suggerite per tutta la tecnologia TI-Innovator.

Se è necessaria ulteriore assistenza, contattare TI-Cares.

TI-Innovator™ Hub

- Non esporre il Hub a temperature superiori ai 60°C (140°F).
- Non smontare o danneggiare il Hub.
- Non concatenare insieme più Hubs attraverso porte I/O o il connettore della basetta sperimentale.
- Usare unicamente il cavo USB fornito con l'Hub.
- Usare unicamente gli alimentatori forniti da TI:
  - TI Wall Charger incluso nell' TI-Innovator™ Hub
  - opzionale External Battery Pack
  - Portabatterie per 4 batterie AA incluso nel TI-Innovator™ Breadboard Pack
- Assicurarsi che i componenti alimentati dall'Hub non superino il limite di potenza di 1 amp dell'Hub's.
- Evitare di utilizzare l'Hub per controllare la corrente elettrica alternata.

Connettore della basetta sperimentale sull'Hub

- Non inserire i connettori dei LED e altri componenti direttamente nel connettore della basetta sperimentale dell'Hub's. Assemblare i componenti sulla basetta sperimentale e utilizzare i cavi di accoppiamento forniti per collegare la basetta sperimentale all'Hub.
- Non collegare il pin fisso da 5 V sul connettore della basetta sperimentale dell'Hub's con nessun altro pin, specialmente i pin di messa a terra. Altrimenti si rischia di danneggiare l'Hub.
- Si sconsiglia di collegare la fila superiore di pin fissi (BB1-10) alla fila inferiore (pin di alimentazione e messa a terra).
- Nessun pin sul connettore della basetta sperimentale dell'Hub's può avere una corrente di caduta o sorgente superiore a 4 mA.

TI-Innovator™ Rover

- Non esporre il Rover a temperature superiori ai 60°C (140°F).
- Non smontare o danneggiare il Rover.
- Non appoggiare nulla che pesi più di 1 Kg o 2,2 lb sulla Rover piattaforma.
- Usare unicamente il cavo USB fornito con l'TI-Innovator™ Hub.
- Usare unicamente i cavi a nastro forniti con il Rover.

- Usare unicamente il caricatore da muro fornito da TI incluso con l’Hub.
- Il Ultrasonic Ranger montato anteriormente rileverà gli oggetti a una distanza massima di 4 metri dal Rover. Per ottenere i risultati migliori assicurarsi che la superficie dell’oggetto sia più grande di una cartellina. Se usato per rilevare oggetti di piccole dimensioni, come una tazza, posizionare il Rover a una distanza fino a 1 metro dall’oggetto.
- Per ottenere i risultati migliori rimuovere la custodia della calcolatrice grafica.
- Per le prestazioni migliori usare Rover sul pavimento e non su tavoli. La caduta del Rover da un tavolo può danneggiarlo.
- Per le prestazioni migliori usare Rover su una superficie dura. Le ruote del Rover possono impigliarsi in un tappeto o agganciarlo e tirarlo.
- Non girare i fermagli di supporto sulla piattaforma per la calcolatrice senza averli prima sollevati. Potrebbero rompersi.
- Non usare il pennarello come una leva per tirare o spingere il Rover.
- Non svitare la cassa protettiva sul fondo del Rover. Gli encoder hanno spigoli appuntiti che non devono essere esposti.
- Non muovere il Rover dopo l’esecuzione di un programma. Il giroscopio interno può cercare involontariamente di rimettere in carreggiata il Rover usando la posizione iniziale.
- Quando si inserisce il cavo a nastro della basetta sperimentale nel Hub connettore della basetta sperimentale, è fondamentale che il cavo venga inserito correttamente. Assicurarsi che lo spinotto del filo rosso (scuro) sia inserito nel foro 5v sul connettore della basetta sperimentale dell'Hub's.

Precauzioni per i moduli I/O

- Utilizzare la porta input od output corretta in base alle necessità di ogni modulo.
  - Motore a vibrazione: supportato su **OUT 1**, **OUT 2** e **OUT 3**.
  - Servomotore: utilizzare solo **OUT 3**.
  - LED bianco: supportato su **OUT 1**, **OUT 2** e **OUT 3**.
  - Sensore di luce analogico: supportato su **IN 1**, **In 2** e **IN 3**.
  - Ranger ultrasonico: supportato su **IN 1**, **IN 2**.
- Utilizzare una fonte di alimentazione ausiliaria per i moduli che richiedono più di 50 mA, come:
  - Motore a vibrazione
  - Servomotore
- Non trattenere l’albero del servomotore mentre ruota. Inoltre, non ruotare a mano il servo motore.
- LED bianco:
  - Non curvare i connettori ripetutamente; altrimenti si finirà per logorare i fili rischiando di romperli.

- Il LED richiede la polarità corretta quando viene inserito nella sua presa. Per maggiori dettagli, fare riferimento alle istruzioni su come assemblare il LED nella TI-Innovator™ Technology eGuide (page ii).
- Il LED richiede la polarità corretta quando viene inserito nella sua presa. Per maggiori dettagli, fare riferimento alle istruzioni su come assemblare il LED (page 326).

- Nessun modulo I/O può avere una corrente di caduta o sorgente superiore a 4 mA.

Precauzioni per la basetta sperimentale

- Non collegare i connettori positivi e negativi di un alimentatore con lo stesso gruppo di 5 pin sulla basetta sperimentale. Altrimenti si rischia di danneggiare la basetta sperimentale e l'alimentatore.
- Rispettare la polarità corretta:
  - Quando si collega la basetta sperimentale all'Hub.
  - Quando si collegano componenti che sono sensibili alla polarità, come i LED e il MOSFET di alimentazione TTL.

Precauzioni per l'adattatore TI-SensorLink e i sensori Vernier

TI-SensorLink

- TI-SensorLink **non** è una soluzione per la raccolta dei dati. Per la raccolta e l'analisi dei dati, si consiglia di utilizzare i rilevatori collegati USB o la basetta Lab.
- I comandi dell'Hub per TI-SensorLink con i sensori analogici Verner **non** sono attualmente inclusi nell'app Hub (serie CE) o nel menu Hub (TI-Nspire™ CX).
- I nuovi comandi e le nuove parole chiave devono essere digitate OPPURE copiate da un programma esistente. Tenere presente che eventuali errori tipografici nelle parole chiave attiverà un'indicazione di errore nel sketch.

Sensori Vernier

- Sensore di pressione del gas: il sensore di pressione del gas è un elemento sensibile che viene danneggiato se entra in contatto diretto con un liquido.
- Sensore del pH: collocare l'elettrodo in una soluzione tampone con pH 4 o pH 7. Non deve mai essere conservato in acqua distillata. Se l'elettrodo viene involontariamente conservato a secco per un breve periodo di tempo, immergerne la punta nella soluzione di conservazione di KCl/tampone con pH 4 per almeno 8 ore prima di utilizzarlo.
- Rilevatore di temperatura in acciaio inossidabile -
  - Attorcigliamento del cavo. Talvolta gli studenti possono attorcigliare o schiacciare il cavo nelle vicinanze dell'impugnatura del sensore. Con il tempo, i cavi potrebbero allentarsi interrompendo il funzionamento del sensore.
  - Surriscaldamento del sensore. Quanto utilizzato nei laboratori di chimica, talvolta gli studenti possono collocare il sensore su una piastra calda causando la vera e propria "cottura" dell'unità.

- L'unità non è resistente all'acqua. L'acqua può penetrare all'interno del manico del sensore danneggiandone i componenti elettronici. Durante la raccolta dei dati, immergere in acqua solo la parte del sensore in acciaio inossidabile.

Domande frequenti

Questa sezione include alcune delle domande frequenti che abbiamo ricevuto a proposito della
tecnologia TI-Innovator™. Se la domanda inviata non è presente, inviare un feedback al team eGuide all'indirizzo hubeguide@list.ti.com

Collegamenti argomento

- Informazioni sulla compatibilità del prodotto
- Informazioni su TI LaunchPad™
- Informazioni generali sulle attività
- Informazioni generali sull'alimentazione di TI-Innovator™ Hub
    - Informazioni sulla batteria esterna per TI-Innovator™ Hub
    - Informazioni sulla batteria del Rover

Informazioni sulla compatibilità del prodotto

Quali prodotti TI sono compatibili con TI-Innovator™ Hub?

TI-Innovator™ Hub è compatibile con i seguenti prodotti TI. Per risultati ottimali, utilizzare sempre l'ultima versione del sketch TI-Innovator e dei prodotti compatibili.

- calcolatrice grafica TI CE
- Palmare TI-Nspire™ CX
- Palmare TI-Nspire™ CX CAS
- Software per computer TI-Nspire™ CX (Student, Teacher e TI-Nspire™ CX Navigator™)

Quale linguaggio di programmazione è compatibile con TI-Innovator™ Hub?

TI-Innovator™ Hub può essere programmato utilizzando il linguaggio di programmazione **TI BASIC** sia sulle calcolatrici grafiche TI CE sia sulle calcolatrici TI-Nspire™ CX. Questo linguaggio di programmazione viene utilizzato in diverse calcolatrici grafiche TI CE e si basa sul linguaggio di programmazione BASIC (Beginner's All-purpose Symbolic Instruction Code). BASIC è una serie di linguaggi di programmazione generici di alto livello la cui filosofia di progettazione si basa sulla loro facilità d'uso.

Inoltre, con la tecnologia TI-Nspire™ CX è possibile utilizzare la **programmazione LUA**, un linguaggio di scripting rapido e potente.

Vedere anche: Hub Programmazione dell'Hub sulla calcolatrice grafica TI CE per ulteriori.

Vedere anche: Hub Programmazione dell'Hub nella tecnologia TI-Nspire™ CX per ulteriori.

Quali sensori, attuatori e simili è possibile collegare a TI-Innovator™ Hub?

TI-Innovator™ Hub presenta due tipi di connettori:

- Connettore universale a 4 pin compatibile con una serie di moduli.
- Connettore della basetta sperimentale che può essere collegato a una basetta sperimentale per eseguire il prototipo dei progetti.

Per facilitare l'utilizzo di questi strumenti, sono disponibili comodi kit che contengono tutti i componenti necessari per il completamento delle attività. Per ulteriori dettagli, vedere le sezioni relative al modulo di I/O e alla basetta sperimentale.

È possibile utilizzare la basetta TI-Nspire™ Lab con i sensori Vernier™ mentre si utilizza TI-Innovator™ Hub?

Sì, la basetta TI-Nspire™ Lab può essere utilizzata contemporaneamente con TI-Innovator™ Hub sul palmare TI-Nspire™ CX o sul software TI-Nspire™ CX. Per poter utilizzare contemporaneamente TI-Innovator™ Hub e la basetta TI-Nspire™ Lab, occorre eseguire l'accesso mediante uno script LUA.

È possibile collegare direttamente i sensori Vernier™ a TI-Innovator™ Hub?

Le porte di TI-Innovator™ Hub non sono direttamente compatibili con i sensori Vernier™. È possibile collegare i sensori Vernier™ a una basetta TI-Nspire™ Lab. Per poter utilizzare contemporaneamente TI-Innovator™ Hub e la basetta TI-Nspire™ Lab, occorre eseguire l'accesso mediante uno script LUA.

È possibile utilizzare il sistema TI-Nspire™ CX Navigator™ quando si usa TI-Innovator™ Hub?

Sì, gli studenti possono collegare il loro palmare TI-Nspire™ CX al sistema TI-Nspire™ CX Navigator™ mentre utilizzano TI-Innovator™ Hub. L'insegnante può utilizzare la funzionalità TI-Nspire™ CX Navigator™, tra cui Presentatore dal vivo, Cattura schermo, Test rapido, ecc. mentre gli studenti usano TI-Innovator™ Hub.

Il software TI Connect™ CE o TI-SmartView™ CE è in grado di comunicare con TI-Innovator™ Hub?

TI-Innovator™ Hub non può comunicare direttamente con il software TI Connect™ CE o TI-SmartView™ CE. Tuttavia, è possibile utilizzare il software TI Connect™ CE per scrivere programmi da usare con TI-Innovator™ Hub. Il software TI-SmartView™ CE è un programma ideale per mostrare la procedura di programmazione agli studenti.

Informazioni su TI LaunchPad™

Che cos'è un kit di sviluppo TI LaunchPad™?

I kit TI LaunchPad sono una serie di kit di sviluppo per microcontrollori (anche chiamati schede di valutazione) prodotti da Texas Instruments. È possibile trovare ulteriori dettagli sull'ecosistema TI LaunchPad all'indirizzo https://www.ti.com/ww/en/launchpad/about.html.

Che tipo di kit TI LaunchPad™ viene utilizzato in TI-Innovator™ Hub?

TI-Innovator™ Hub è alimentato da un kit TI LaunchPad MSP432P401. Per ulteriori informazioni su LaunchPad MSP432P401, visitare il sito https://www.ti.com/ww/en/launchpad/launchpads-msp430-msp-exp432p401r.html#tabs.

È possibile utilizzare TI-Innovator™ Hub come kit di sviluppo TI LaunchPad™?

Anche se TI-Innovator™ Hub può essere utilizzato come scheda TI LaunchPad™, TI-Innovator™ Hub è stato specificamente progettato per essere utilizzato da studenti che stanno imparando a programmare, costruire ed esplorare utilizzando componenti elettronici. Per ulteriori informazioni su TI LaunchPad, visitare il sito https://www.ti.com/ww/en/launchpad/about.html.

Quali sono le risorse disponibili per TI LaunchPad?

Gli utenti interessati all'ecosistema TI LaunchPad possono visitare il sito https://www.ti.com/ww/en/launchpad/about.html per ulteriori risorse.

Come vengono utilizzati i kit di sviluppo/ le schede tecniche dagli ingegneri nel mondo reale?

Gli ingegneri si servono di schede di valutazione, come ad esempio le schede TI LaunchPad™, per eseguire il prototipo dei loro progetti e verificare la sostenibilità di un chip specifico all'interno del loro progetto. Queste schede permettono agli ingegneri di sperimentare approcci diversi prima di finalizzare il loro progetto. Le schede consentono inoltre agli ingegneri di misurare altri aspetti dei loro progetti, come ad esempio il consumo energetico e la velocità delle operazioni.

Queste schede di valutazione vengono inoltre utilizzate nelle università per imparare il funzionamento dei microcontrollori, la programmazione e l'interfacciamento con i sensori.

Informazioni generali sulle attività

What activities are available for the TI-Innovator™ Hub?

Con TI-Innovator™ Hub è possibile svolgere diverse attività. In collaborazione con gli insegnanti, abbiamo creato attività relative ai seguenti argomenti:

10 minuti di codice per TI-Innovator™ Hub: lo scopo è coinvolgere gli studenti in brevi attività che li aiutino a capire meglio i concetti matematici, la logica di programmazione e le competenze di codifica. Durante le attività, si utilizzano l'RGB integrato, il LED, il diffusore e il sensore di intensità luminosa di TI-Innovator™ Hub. Le attività sono disponibili per la serie di calcolatrici grafiche TI CE e per la tecnologia TI-Nspire™ CX.

10 minuti di codice per TI-Innovator™ Rover: lo scopo è continuare a imparare la programmazione con TI-Innovator™ Rover. Ideata per approfondire la conoscenza della programmazione di TI-Innovator™ Hub e per scrivere programmi per il controllo di TI-Innovator™ Rover. Questa attività consente di imparare i comandi di spostamento del Rover e di utilizzare il ranger e il sensore per i colori integrati. Le attività saranno disponibili per la serie di calcolatrici grafiche TI CE e per la tecnologia TI-Nspire™ CX.

"Conversazioni" durante la lezione di matematica e scienza per TI-Innovator™ Rover: programmi pronti all'uso per la serie di calcolatrici grafiche TI CE e per la tecnologia TI-Nspire™ CX. Questi programmi includono una guida all'uso per l'insegnante che contiene suggerimenti sull'implementazione di TI-Innovator™ Rover nei programmi in dotazione al fine di esplorare i concetti presentati durante la lezione di matematica e/o scienze.

Scienze attraverso la progettazione: lezioni interattive e ricche di contenuti per gli studenti delle scuole medie che studiano biologia e fisica. Si serve di componenti forniti nel pacchetto di moduli di I/O TI-Innovator™. Le attività sono disponibili per la tecnologia TI-Nspire™ CX.

Tragitto per i progetti STEM: progettazione, costruzione, collaudo, perfezionamento. Queste attività sequenziali sono indirizzate agli studenti delle scuole medie e superiori che stanno studiando i principi tecnici e hanno lo scopo di fornire loro le conoscenze e le competenze di base richieste per sintetizzare progetti STEM nuovi e unici. Queste attività richiedono i componenti in dotazione con il Kit della basetta sperimentale TI-Innovator™. Le attività sono disponibili per la serie di calcolatrici grafiche TI CE e per la tecnologia TI-Nspire™ CX.

Dov'è possibile posso scaricare le attività per TI-Innovator™ Hub?

Per scaricare le attività da utilizzare con TI-Innovator™ Hub, visitare il sito Web education.ti.com e selezionare la scheda delle attività nella parte superiore di ogni pagina. Di seguito sono riportati i collegamenti diretti a ogni serie di attività:

- 10 minuti di codice con TI-Innovator™ Hub: education.ti.com/ticodes
- 10 minuti di codice con TI-Innovator™ Rover: education.ti.com/ticodes
- "Conversazioni" durante la lezione di matematica e scienza per TI-Innovator™ Rover:
- Scienze attraverso la progettazione: https://education.ti.com/en/tisciencenspired/us/stem
- Tragitto per i progetti STEM: **da definire**

Quando saranno disponibili queste attività?

Le attività per TI-Innovator™ Hub sono già disponibili. Le attività per TI-Innovator™ Rover saranno disponibili a partire dall'autunno del 2017.

Informazioni generali sull'alimentazione di TI-Innovator™ Hub

How is the TI-Innovator™ Hub alimentato?

TI-Innovator™ Hub è alimentato dalle batterie della calcolatrice grafica TI CE o del palmare TI-Nspire™ CX. Per alcune attività che vengono svolte su dispositivi ad alto consumo come i servomotori, potrebbe essere necessario utilizzare una fonte di alimentazione supplementare, come un adattatore da parete TI o una batteria esterna.

How does the TI-Innovator™ Hub influenza la durata della batteria della calcolatrice grafica TI CE o di TI-Nspire™ CX?

TI-Innovator™ Hub esercita un impatto minimo sulla batteria della calcolatrice grafica TI CE o delle calcolatrici grafiche TI-Nspire™ CX.

Quando è necessario utilizzare l'alimentazione esterna?

Quando si utilizzano le porte di input e output:

> Alcuni moduli di I/O richiedono alimentazione esterna poiché si servono di porte da 5 V (OUT3 o IN3) su TI-Innovator™ Hub. Per i dettagli, vedere la sezione sui moduli di I/O.

Quando si utilizza il connettore della basetta sperimentale:

> Se il circuito è alimentato dall'output da 5 V del connettore della basetta sperimentale, sarà necessaria un'alimentazione esterna.

Quali sono le opzioni di alimentazione esterna disponibili?

Per ottenere un'alimentazione supplementare, è possibile utilizzare l'adattatore da parete TI o la batteria esterna. L'adattatore da parete TI è fornito in dotazione con TI-Innovator™ Hub ed è lo stesso caricatore da parete fornito con la calcolatrice grafica TI CE e le calcolatrici TI-Nspire™ CX. La batteria esterna è venduta separatamente come accessorio per TI-Innovator™ Hub.

Can I use a different battery/power supply with the TI-Innovator™ Hub?

Per garantire un funzionamento sicuro, si consiglia di utilizzare esclusivamente la batteria e la fonte di alimentazione fornite da TI.

Informazioni sulla batteria esterna per TI-Innovator™ Hub

Che cos'è la batteria esterna?

La batteria esterna fornisce alimentazione supplementare per quei componenti che richiedono ulteriore potenza rispetto a quella fornita dalla calcolatrice grafica TI.

Questa batteria (n. modello MP-3000) è stata selezionata per soddisfare le esigenze di alimentazione dei componenti di TI-Innovator™.

Come si utilizza la batteria esterna con TI-Innovator™ Hub?

Utilizzando il cavo USB da Standard-A a Micro-B fornito in dotazione con TI-Innovator™ Hub, collegare la batteria esterna alla porta USB PWR su TI-Innovator™ Hub. La batteria esterna è dotata di interruttore On/Off che deve essere acceso per poter alimentare TI-Innovator™ Hub.

Quanto dura la batteria completamente carica?

La durata della batteria dipenderà dai componenti collegati a TI-Innovator™ Hub. Ad esempio, con la batteria esterna, il modulo del servomotore utilizzato per le attività Scienza attraverso la progettazione può durare 8 ore ininterrotte. Altri componenti potrebbero durare più a lungo oppure esaurire la batteria più rapidamente.

Qual è la durata media della batteria?

Con il passare del tempo, le batterie agli ioni di litio perdono la loro capacità. Se conservate in modo corretto e utilizzate normalmente, le batterie possono durare circa tre anni.

Come si esegue la ricarica della batteria?

La batteria esterna può essere ricaricata con l'adattatore da parete TI (in dotazione con TI-Innovator™ Hub) o collegando il cavo USB fornito con TI-Innovator Hub alla porta USB del computer.

Com'è possibile verificare il livello di ricarica della batteria?

Quando si accende la batteria esterna, gli indicatori LED che si trovano sulla batteria esterna indicheranno il livello di carica approssimativo (25%, 50%, 75% e 100%). I LED si spengono automaticamente dopo 10 secondi.

È possibile utilizzare la batteria esterna con altri prodotti?

La batteria esterna è stata testata specificatamente per essere utilizzata con TI-Innovator™ Hub.

Informazioni sulla batteria del Rover

Quanto dura la batteria completamente carica?

La batteria durerà 8 ore di guida ininterrotta. Durante l'uso, si prendono generalmente frequenti pause per la programmazione. In questo caso, una ricarica completa dovrebbe durare diversi giorni.

Qual è la durata media della batteria?

Con il passare del tempo, le batterie agli ioni di litio perdono la loro capacità. Se conservate in modo corretto e utilizzate normalmente, le batterie possono durare circa 3 anni.

Come si esegue la ricarica della batteria?

Collegare un cavo micro-USB alla porta PWR sul lato anteriore destro del Rover. L'altra estremità del cavo può essere collegata a un PC o a un caricatore da parete TI.

Com'è possibile verificare il livello di ricarica della batteria?

I LED con i quattro livelli di carica della batteria mostrano la capacità della batteria. Quando tutti e quattro i LED emettono una luce verde fissa, significa la batteria del Rover è completamente carica.

Informazioni Generali

Guida online

education.ti.com/eguide

Selezionare il proprio Paese per maggiori informazioni sul prodotto.

Contattare l'assistenza TI

education.ti.com/ti-cares

Selezionare il proprio Paese per assistenza tecnica e altre risorse.

Informazioni su servizi e garanzia

education.ti.com/warranty

Selezionare il proprio Paese per informazioni sulla durata e sui termini della garanzia o sull'assistenza ai prodotti.

Garanzia limitata. La presente garanzia non pregiudica i diritti spettanti per legge.