

Questo documento è pensato per i più giovani, motivo per cui lo stile è diverso dal resto del sito.

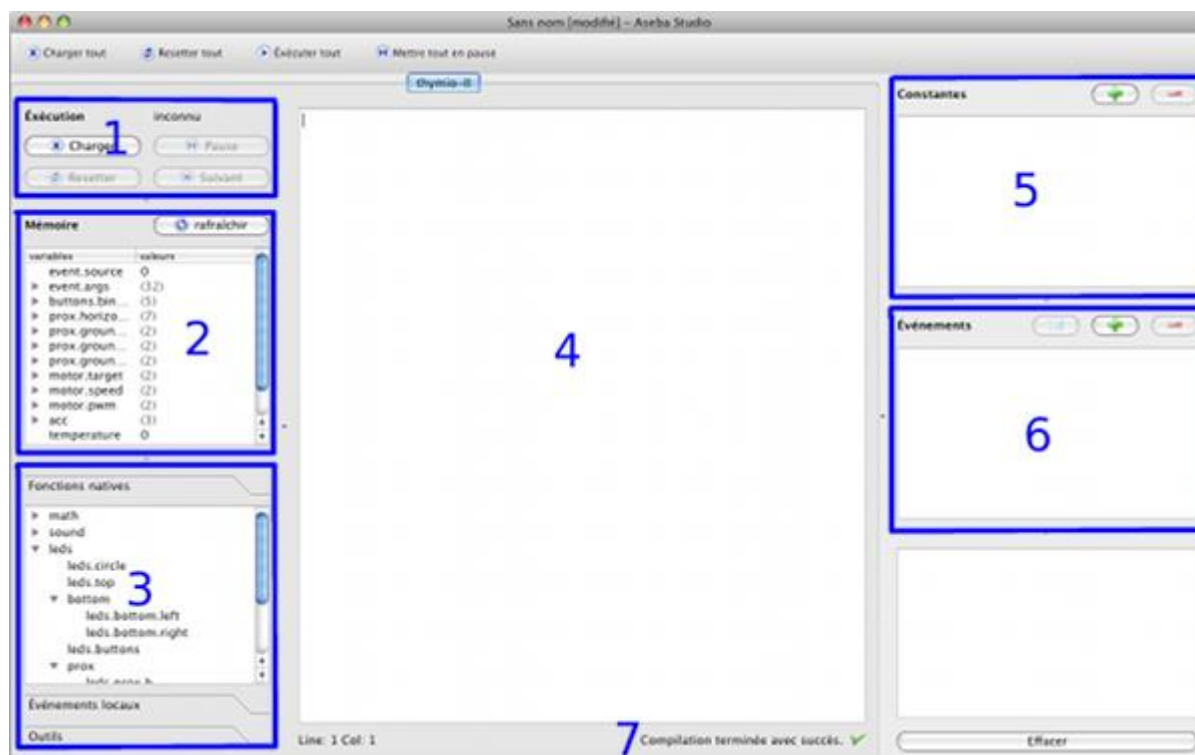
Questa guida vi condurrà attraverso una serie di esercizi con il Thymio II. Passo dopo passo imparerete come programmare Thymio, per controllare i LED, i motori e così via. Alla fine avrete definito un nuovo comportamento, o modalità, del vostro robot tutto da soli!

Ci sono diverse parti di questa guida e ognuna si concentra su un aspetto diverso.

Conoscere il robot e ASEBA

Iniziate prendendo possesso del robot. È possibile accenderlo premendo il pulsante centrale. Poi si può scegliere uno dei [comportamenti](#) predefiniti premendo un pulsante freccia: Thymio cambia colore. È possibile avviare questa modalità premendo il pulsante centrale. Nel resto di questo esercizio si creerà il proprio comportamento, con l'aggiunta di una nuova scelta per i colori.

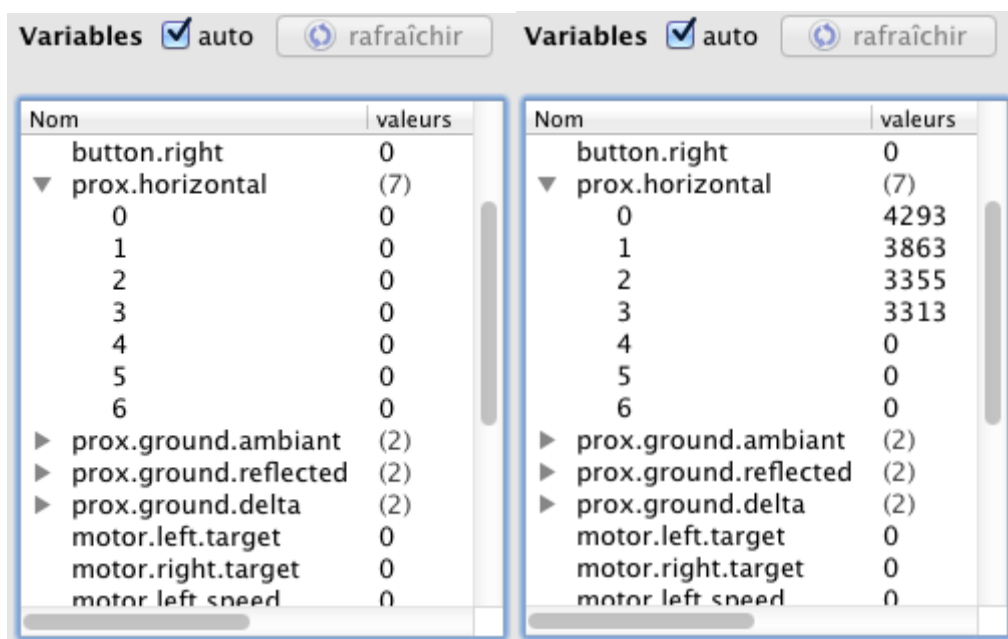
Collegare il robot al computer con il cavo USB e avviare ASEBA Studio. ASEBA Studio assomiglia a questo



1. Questi pulsanti vengono usati per caricare ed eseguire il codice (istruzioni) che è stato scritto.
2. Questa finestra mostra una panoramica dello stato delle diverse parti del robot: sensori, attuatori, variabili.
3. Qui vedi le funzioni e gli eventi disponibili che possono essere utilizzati nel codice.
4. Nel centro scriveremo il nostro *codice*. Questo è ciò che dirà al robot come comportarsi.
5. Qui le costanti possono essere create o verificate.
6. Qui possono essere aggiunti gli eventi.
7. Questa linea ci dice se le nostre istruzioni sono complete e scritte correttamente.

I sensori

È possibile utilizzare ASEBA Studio per guardare i valori del robot di *sensori* nel menu a sinistra. Un sensore è utilizzato per misurare qualcosa di fisico (reale), come la luce, un suono o una accelerazione, e convertirlo in un valore numerico che il robot può comprendere e utilizzare. Ad esempio i *sensori di distanza* nella parte anteriore del robot consentono al robot di misurare la distanza di oggetti vicini. È il valore **** prox.horizontal **** nella finestra **Variabili ****. **Se l'opzione **auto** è selezionata e si mette la mano davanti al robot, è possibile vedere il valore cambiare.



Nom	valeurs
button.right	0
▼ prox.horizontal	(7)
0	0
1	0
2	0
3	0
4	0
5	0
6	0
▶ prox.ground.ambient	(2)
▶ prox.ground.reflected	(2)
▶ prox.ground.delta	(2)
motor.left.target	0
motor.right.target	0
motor left speed	0

Nom	valeurs
button.right	0
▼ prox.horizontal	(7)
0	4293
1	3863
2	3355
3	3313
4	0
5	0
6	0
▶ prox.ground.ambient	(2)
▶ prox.ground.reflected	(2)
▶ prox.ground.delta	(2)
motor.left.target	0
motor.right.target	0
motor left speed	0

Nella prima foto si può vedere che i sensori non vedono nulla (0). Nella seconda, con una mano davanti, i valori sono elevati.

Che cosa è il codice?

Il codice è una serie di *istruzioni* per il robot, che *eseguirà* e che ne determinano il comportamento. Il codice deve essere scritto con particolari parole chiave che il robot può comprendere; non si può semplicemente parlargli in inglese o in italiano.

Un trucco per scrivere codice più rapidamente

In ASEBA Studio, è possibile trascinare le parole chiave dei comandi dai menu laterali nella finestra principale. Per creare blocchi di codice simile, è possibile copiare e incollare (Ctrl-C, Ctrl-V in Windows e Linux, o Comando-C, Comando-V in Mac) i blocchi.

Commenti

Ciò che è scritto nel codice viene interpretato dal robot come istruzioni. Volendo scrivere qualche nota per se stessi, per spiegare a cosa servono le istruzioni, è possibile introdurre dei *commenti*. Il

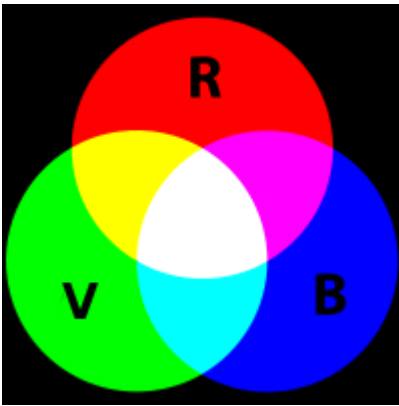
segno # indica che il testo seguente in quella linea non è un'istruzione per il robot. Vedremo un esempio più avanti.

In questa parte programmiamo Thymio in modo che cambi colore quando vengono premuti i tasti.

I LED

Un *LED* ([light emitting diode](#) = diodo emettitore di luce) è un dispositivo utilizzato per generare luce colorata sul robot. Un LED ha un colore che viene fissato durante la fabbricazione. Per esempio, ci sono LED rossi all'estremità anteriore del robot, verdi sulla parte superiore, ecc.

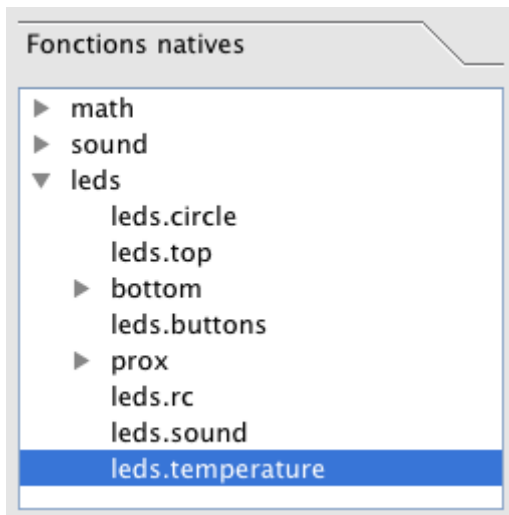
Abbiamo visto che il robot può cambiare colore. Come si fa? In realtà, ci sono tre diversi LED molto vicini tra loro, uno rosso, uno verde e uno blu. Questi tre colori sono chiamati colori primari, perché mescolandoli tra loro possono essere creati tutti gli altri colori. Ad esempio, se blu e rosso sono accesi insieme, dà magenta. Questi LED raggruppati sono chiamati *RGB LED* (Red=rosso, Green=verde, Blue=blu).



Accendere i LED

Per accendere una luce sul robot, ci accingiamo a *programmare*, vale a dire che scriveremo le istruzioni per il robot.

Se si fa clic su **Funzioni native** -> **led** potete vedere una lista di *Funzioni* (una funzione è una delle possibili istruzioni per il robot). Con queste funzioni possiamo accendere i diversi LED. Per esempio **leds.top** illumina il LED RGB superiore.

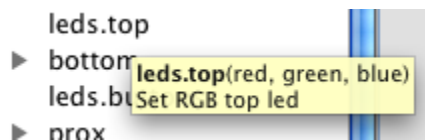


Nella finestra principale, digitare:

```
call leds.top(0,0,32)
```

call viene utilizzato per chiamare una funzione; **leds.top** accende il LED RGB sulla parte superiore del robot.

Se si lascia il cursore sopra il nome della funzione, potrete vedere apparire le istruzioni che dicono come usarla. Quando si chiama una funzione, è possibile fornire informazioni alla funzione tra le parentesi (). Queste sono chiamati *argomenti*.



Nel nostro caso ci sono tre argomenti: i valori per le tre parti del LED RGB: rosso, verde e blu. Il valore 0 indica spento (nessun indicatore), mentre 32 significa completamente acceso (luminoso). Valori intermedi danno una luce fioca.

Ora cliccate su **Carica**, quindi **Esegui**. Il robot esegue l'istruzione. Si può vedere che il LED superiore si illumina blu.

Se si modifica il codice in:

```
call leds.top(32,0,0)
```

Il LED sarà rosso, similmente:

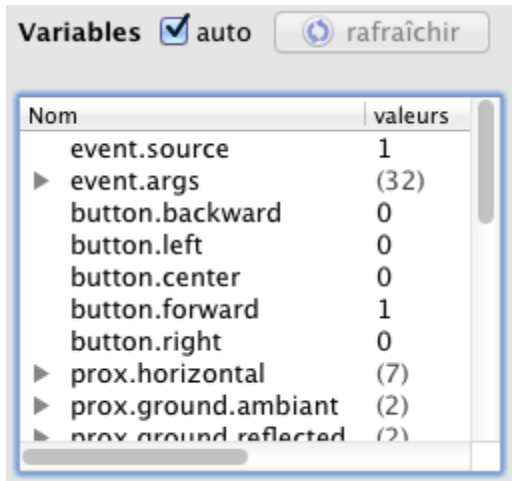
```
call leds.top(0,32,32)
```

dà una luce azzurra LED (green + blue). Si può provare a cambiare i valori per vedere quali colori vengono prodotti.

Condizioni

Al momento, non appena il pulsante 'Esegui' viene cliccato, il robot esegue il codice (accende il LED). Potremmo voler collegare questa azione a un pulsante, per esempio, in modo che il LED si accenda quando viene toccato il pulsante.

Per rilevare pressione di un pulsante, è possibile controllare la variabile nella finestra **Variabili: *button.name***



Qui si può vedere che il pulsante "avanti" viene toccato.

In questo caso utilizzeremo la "condizione// **if ... then ... else ... end**. L'idea è di essere in grado di dire al robot: **if** (se) il bottone è premuto, **then** (allora) accendi il LED. In Aseba, questo si scrive nella maniera seguente:

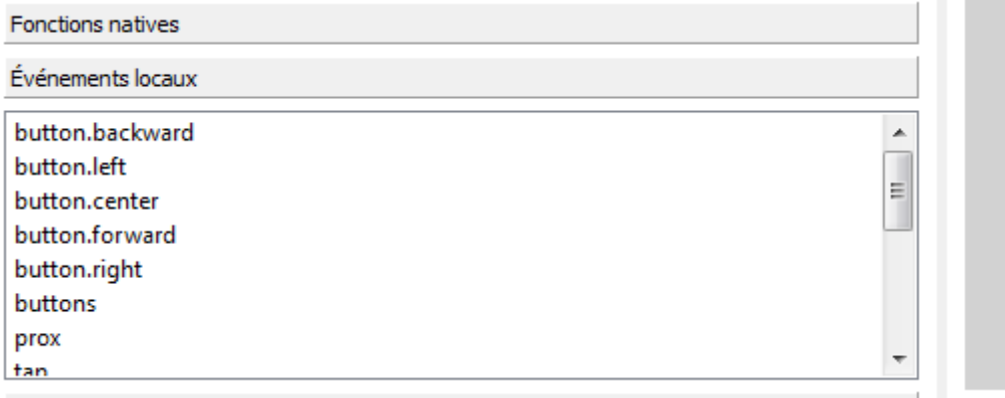
Questo codice specifica che quando viene premuto il tasto avanti (**button.forward**), il LED superiore diventa blu, altrimenti è spento. Guardate il testo che segue il simbolo #, che è in italiano, non in linguaggio *ASEBA*. Questi sono i commenti, note per noi, non per il robot, quindi li mettiamo dopo il carattere #.

Questo codice fa ciò che vorremmo che facesse?

No. Quando si prova questo codice, se si tocca e si rilascia il pulsante il robot non cambia colore. Vediamo che funziona solo una volta, all'inizio del programma. Se vogliamo che la luce si accenda ogni volta che si preme il pulsante, dobbiamo verificare lo stato del pulsante regolarmente. Per ottenere questo useremo un *evento*.

Eventi

Un evento è un istante, determinato da un certo insieme di condizioni, in cui un codice può essere eseguito. Un evento può verificarsi più volte! Ad esempio, se si guarda nella casella in basso a sinistra della ASEBA Studio, c'è un elenco degli **eventi locali**.



L'evento **buttons** corrisponde ad ogni volta che viene verificato se è stato premuto un tasto. Allo stesso modo, **buttons.forward** corrisponde a "ogni volta che viene verificato se è stato premuto il pulsante avanti". Quindi possiamo aggiungere una sola linea di fronte al nostro codice precedente:

```

1. # ogni volta che i bottoni vengono verificati
2. onevent button.forward
3. if button.forward==1 then      # se il bottone in avanti viene toccato
4.     call leds.top(0,0,32)      # accendi il LED di colore blu
5. else
6.     call leds.top(0,0,0)       # altrimenti spegnilo
7. end

```

Se testiamo questo codice (**Load, Run**), vedremo che il led del robot si accende blu ogni volta che tocchiamo il tasto frontale, e si spegne quando smettiamo. Ci sono molti altri eventi disponibili: **motor** per i motori, **acc** per l'accelerometro ecc

Ora ci accingiamo a far muovere robot. Il robot cambia direzione a seconda che si preme un tasto.

Motori

Per i motori ci sono 3 variabili differenti in memoria. Useremo solo i primi due per iniziare. La variabile **motor.side.target** viene utilizzata per specificare la velocità che vogliamo che i motori abbiano. La variabile **motore.laterale.speed** indica la velocità attuale.

```

▷ prox.ground.delta      (2)
motor.left.target       100
motor.right.target      -100
motor.left.speed         108
motor.right.speed       -110
motor.left.pwm           -204
motor.right.pwm          178
▷ acc                    (3)

```

Possiamo vedere che la velocità reale (**motor.side.speed**) differisce leggermente dalla velocità obiettivo (**motor.Lateral.target**)

```
1. Ora possiamo# ogni volta che si verifica se i bottoni sono stati premuti
2. onevent buttons
3.
4. if button.forward==1 then
5.     motor.left.target=200
6.     motor.right.target=200
7. end
8.
9. if button.center==1 then
10.     motor.left.target=0
11.     motor.right.target=0
12. end
13.
14. if button.backward==1 then
15.     motor.left.target=-200
16.     motor.right.target=-200
17. end
18.
19. if button.left==1 then
20.     motor.left.target=-200
21.     motor.right.target=200
22. end
23.
24. if button.right==1 then
25.     motor.left.target=200
26.     motor.right.target=-200
27. end
```

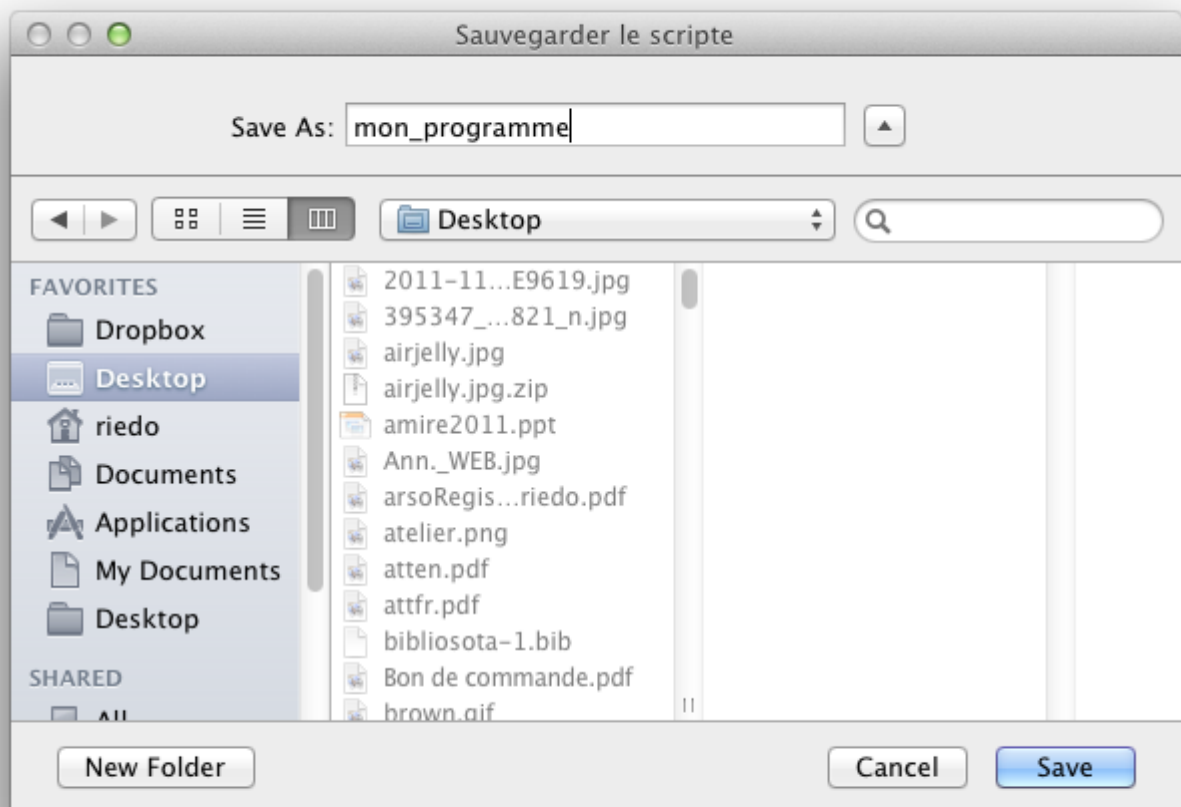
scrivere il codice che attiva i motori a seconda dei pulsanti che vengono premuti.

Ora si può provare a guidare il robot con i pulsanti (non dimenticare di **Caricare** ed **Eseguire**).

E se si desidera disconnettere il robot?

Testare il robot con il cavo collegato non è molto pratico. Che cosa possiamo fare?

Se si scollega il robot dal computer, ASEBA Studio tenterà di riconnettersi, in modo che quando si ricollega il robot sia possibile continuare la programmazione. Tuttavia, se si chiude ASEBA Studio, il programma sarà perso. Quindi vi consigliamo di salvare il codice prima di scollegare il vostro robot. Per farlo, andare al menu **file** e fare clic su **Salva**. Una finestra di dialogo simile a questa apparirà:



Dopo questa operazione il nuovo comportamento del robot viene conservato su un file. Se succede qualcosa con ASEBA Studio mentre si gioca con il robot, è possibile riavviare ASEBA Studio e ricaricare il file di programma che è appena stato salvato.

Utilizzare i sensori

Ora useremo i sensori di prossimità per evitare che il robot cada fuori dal tavolo. Ci sono nove sensori di prossimità su ogni Thymio II: 5 nella parte anteriore, 2 sotto e 2 sul retro. Quelli sotto possono essere utilizzati per fermarsi al bordo del tavolo.

Sensore di prossimità

Un sensore di prossimità può misurare la distanza degli oggetti nelle vicinanze. Per fare questo si utilizzano due diverse componenti: un emettitore a infrarossi e un ricevitore. L'emettitore irradia luce infrarossa (invisibile a noi) e il ricevitore misura quanta di questa luce viene riflessa e torna indietro. Se un oggetto è vicino, una gran parte della luce viene riflessa da esso e ritorna al robot. Se l'oggetto è più lontano, solo una piccola parte della luce ritorna. In questo modo è possibile risalire alla distanza dagli oggetti.

Nel nostro caso i sensori inferiori verranno utilizzati per riconoscere il bordo del tavolo. Mettete il robot sul tavolo e osservate i valori della variabile **prox.ground.delta** in Aseba Studio nell'area variabili (avendo cliccato **aggiorna** o selezionato **auto**).

```
▼ prox.ground.delta (2)
  0 574
  1 631
  ...
```

Ora fatelo di nuovo tenendo il robot in aria. Potete vedere che il valore mostrato cambia molto.

```
▼ prox.ground.delta (2)
  0 7
  1 8
  ...
```

Quando il robot è sul tavolo, i valori misurati sono intorno a 600 (molta luce ritorna). Quando è sollevato, i valori sono molto più piccoli (non molta luce ritorna). Quindi dovremo aggiungere al fondo del nostro codice questa condizione:

```
1. onevent prox
2. # se un sensore misura meno di 300 non sta vedendo il tavolo
3. if prox.ground.delta[0]<300 or prox.ground.delta[1]<300 then
4.     # ferma i motori
5.     motor.left.target=0
6.     motor.right.target=0
7.     # illumina di rosso il robot per segnalare una sosta di emergenza
8.     call leds.top(32, 0, 0)
9. else
10.    # se non siamo più al bordo del tavolo, spegnere la luce rossa
```

```
11.     call leds.top(0, 0, 0)i
12. end
```

Se il valore di uno dei due sensori è troppo basso (tavolo non rilevato), fermiamo i motori. Ora si può vedere che il robot si fermerà al bordo del tavolo.

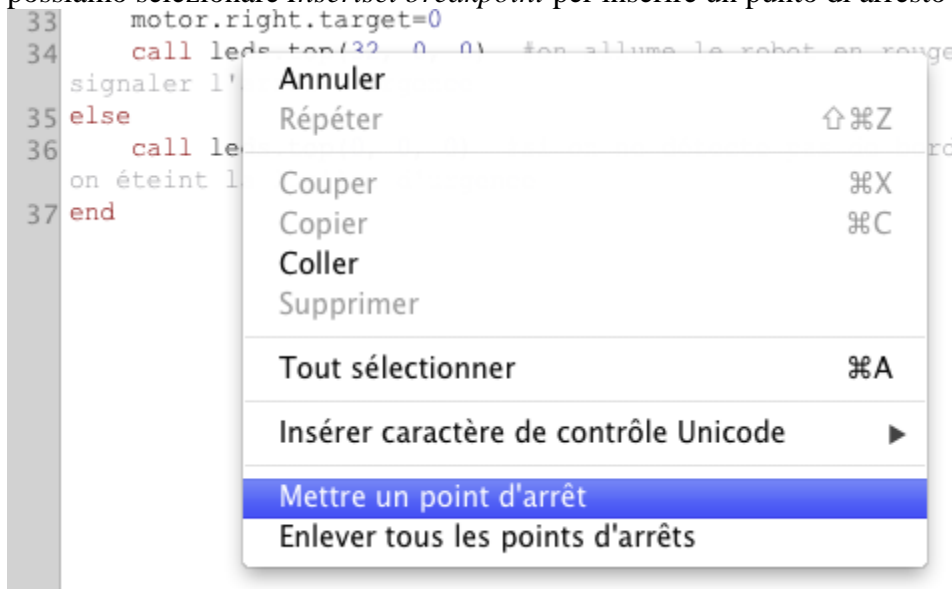
Il debugger

Può essere che durante il test del robot, vogliamo vedere più in dettaglio ciò che sta accadendo. Questo ci può aiutare a correggere gli errori nel nostro codice o per eseguire le istruzioni passo per passo. Nel nostro caso esamineremo cosa succede quando il robot raggiunge il bordo del tavolo.

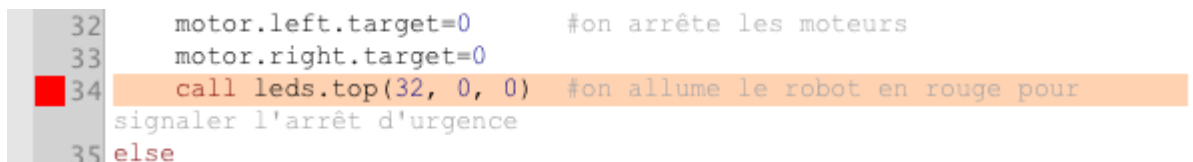
In ASEBA Studio, se facciamo tasto destro del mouse sulla linea:

```
call leds.top(32, 0, 0)
```

possiamo selezionare *Inserisci breakpoint* per inserire un punto di arresto nel codice



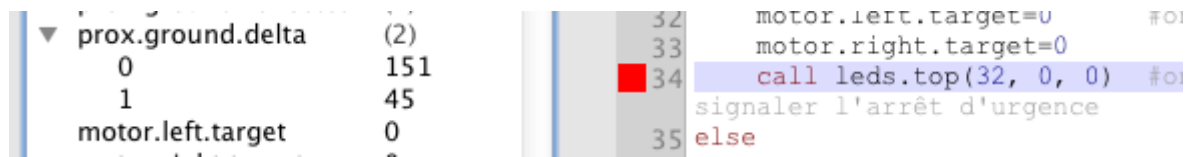
Il punto di arresto è indicato da un quadrato rosso a sinistra.



Ciò significa che quando si arriva a questo punto, il robot attenderà indefinitamente, fino a quando non diciamo di continuare.

Proviamo. Per prima cosa dobbiamo togliere il segno di spunta nella casella **auto** se vogliamo rilevare i valori al punto di interruzione. Una volta che il punto di interruzione è impostato, possiamo far muovere il robot al bordo del tavolo. Si ferma; possiamo osservare la situazione in

questo momento, per esempio noi vediamo qui che i sensori inferiori hanno misurato i valori 151 e 45 (non molta luce, quindi non viene rilevato il tavolo).



The image shows two side-by-side screenshots from a code editor. The left screenshot displays a list of variables: 'prox.ground.delta' with a value of 151, and 'motor.left.target' with a value of 0. The right screenshot shows a code snippet with a red square highlighting a line: 'call leds.top(32, 0, 0)'. The code includes comments in Italian: 'motor.left.target=0', 'motor.right.target=0', 'signaler l'arrêt d'urgence', and 'else'.

Appena clicchiamo di nuovo su *Esegui* il robot continua (si accende di rosso) fino al successivo punto di arresto. E' possibile rimuovere un punto di arresto facendo click con il tasto destro del mouse sulla riga del punto di interruzione, e quindi selezionando la voce di menu *Azzerà breakpoint*.

In questa parte, ci accingiamo a programmare Thymio modo che reagisca agli urti ed emetta un suono.

L'accelerometro

Come può essere rilevato un urto? Quale sensore può essere utilizzato per questo?

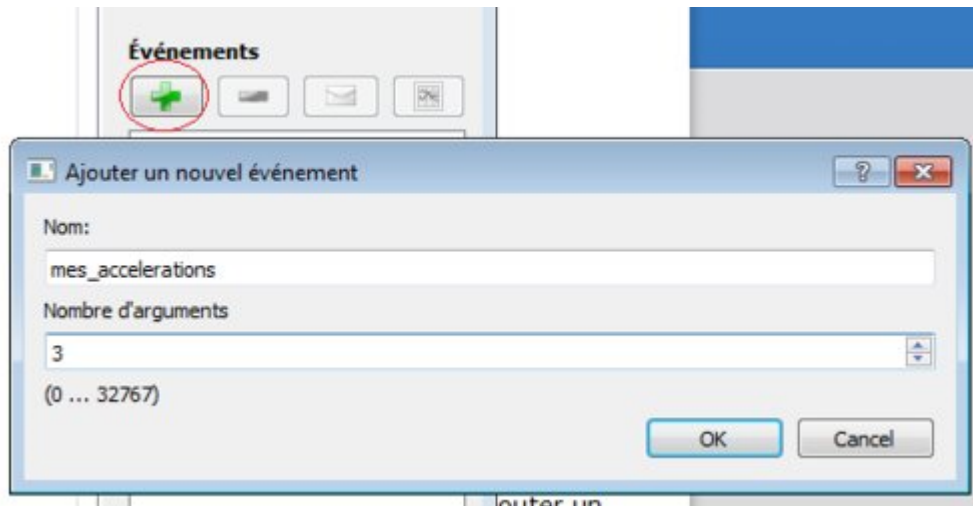
Rilevare urti o brusche variazioni di accelerazione è molto comune e utile. Ecco alcuni esempi:

- In una macchina, lo shock causato da una collisione può essere rilevato e utilizzato per gonfiare l'airbag.
- In un certo numero di telefoni cellulari, quando l'utente tocca con il dito una o più volte questo viene rilevato.
- In un computer portatile, il rilevamento di una caduta libera può essere utilizzato per proteggere il disco rigido in modo che non venga danneggiato dall'urto.

Questo tipo di evento viene rilevato utilizzando l'accelerometro. Come si può indovinare dal nome, l'accelerometro misura le accelerazioni, come la gravità. Si possono quindi anche misurare una caduta libera e gli urti.

Osservare l'attività dell'accelerometro

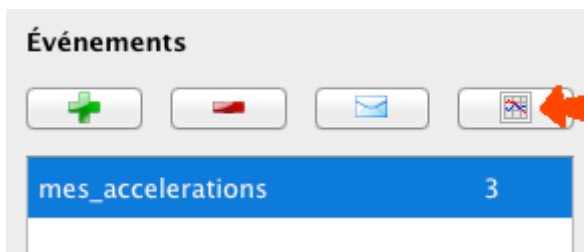
ASEBA Studio può fare grafici dei valori delle variabili. Per questo, dobbiamo creare il nostro *evento*. Se vogliamo osservare i tre assi dell'accelerometro (l'accelerazione viene misurata in tre dimensioni), abbiamo bisogno di creare un evento con 3 argomenti. Per fare questo, fare clic sul + **** per aggiungere un evento, quindi dare un nome (ad esempio ****my_accelerations**) e specificare il numero di argomenti (3).**



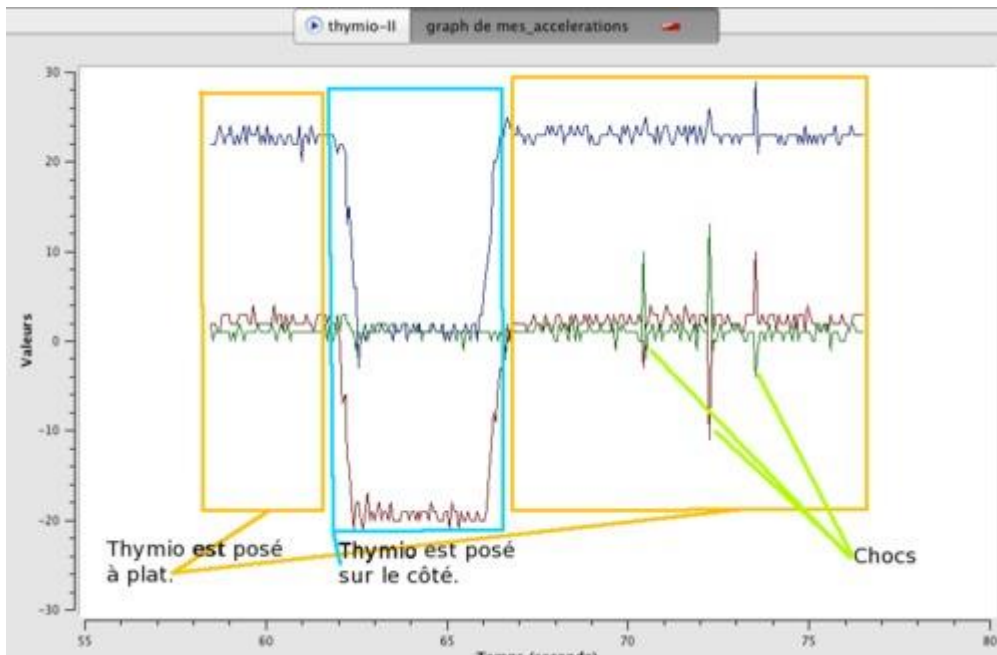
Ora, nel codice, possiamo emettere il valore di accelerazione (variabile **acc** nella finestra **variabili**, che dispone di 3 elementi), dall'evento **acc** al nostro nuovo evento **my_accelerations** che può trasmettere 3 argomenti:

1. `onevent acc`
2. `emit my_accelerations acc`

Quindi, nella finestra **Eventi** possiamo selezionare **my_accelerations** e cliccare sul bottone dei grafici.



Si apre una nuova finestra con un grafico delle 3 accelerazioni. Vi possiamo vedere cosa succede se il robot è ruotato o se viene urtato.



Qui vediamo i 3 assi in tre diversi colori. Quando il robot appoggia sulle sue ruote, la gravità viene rilevato sull'asse z (blu). Quando viene messo su un lato, l'accelerazione viene rilevato su un altro asse (quello rosso). Gli urti causano i picchi nei grafici.

Utilizzando l'accelerometro

L'accelerometro può essere utilizzato in diversi modi. Come abbiamo visto, la variabile **acc** fornisce i valori delle accelerazioni lungo i tre assi, e l'evento **acc** avviene ogni volta che le accelerazioni vengono misurate.

In aggiunta vi è un evento **tacco** (tap) che si verifica quando il robot viene toccato o rileva un colpo di qualche tipo. Questi eventi possono essere utilizzati per fare un suono quando il robot viene toccato.

1. `onevent tap`
2. `call sound.system(4) # questa funzione suona il suono 4 di sistema`

Complimenti!

avete raggiunto la fine di questa prima esercitazione e avete creato un nuovo comportamento per Thymio! Ora è possibile salvare il codice se si desidera conservarlo, chiudere ASEBA Studio e scollegare il vostro robot per provarlo, o continuare a fare la propria versione di comportamento del vostro robot.

Si può cercare di scoprire da soli come fare le seguenti operazioni:

- Muovere in avanti ed evitare gli ostacoli
- Rilevare una caduta libera
- Cambia colore a seconda su quale lato è appoggiato il vostro robot

Se si desidera aggiungere altre diverse reazioni al comportamento di Thymio, è possibile aggiungere altri pezzi di codice che utilizzano diversi sensori o LED. Fate i vostri esperimenti e buon divertimento!

Cosa leggere dopo?

Potresti essere interessato a leggere:

- [Continua la scoperta Thymio e programmazione attraverso progetti](#)
- [Un elenco completo delle modalità di accesso Thymio caratteristiche attraverso variabili, eventi e funzioni](#)
- [Descrizione delle funzioni native della libreria standard](#)
- [Descrizione del linguaggio Aseba](#)
- [Una descrizione dei concetti di base utilizzati in Aseba](#)
- [Documentazione dell'ambiente di sviluppo integrato](#)