Primi passi nella Robotica

con il

Robot Thymio-II

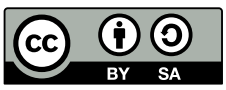
e

L'Ambiente di programmazione Aseba/VPL

Moti Ben-Ari e altri

vedere `authors.txt` per dettagli

Versione 1.3~pre3 per Aseba 1.3.1



Indice

Prefazione

Che cosa è un Robot?

Stai guidando la bicicletta e improvvisamente vedi che la strada inizia ad andare in salita. Pedali più veloce per fornire più potenza alle ruote in modo che la bicicletta non rallenti. Quando raggiungi la cima della collina e cominci ad andare in discesa, stringi la leva del freno. Questo fa sì che due pezzi di gomma (i pattini) vengano premuti contro la ruota e la bicicletta rallenti. Quando guidi una bicicletta, i tuoi occhi sono i *sensori* che percepiscono ciò che succede nel mondo. Quando questi sensori—i tuoi occhi—rilevano un *evento* come una curva della strada, esegui un'*azione*, come spostare il manubrio a sinistra o a destra.

In una macchina, ci sono sensori che *misurano* ciò che sta accadendo nel mondo. Il tachimetro misura quanto velocemente sta andando la vettura; se si rileva una velocità superiore al limite, si potrebbe dire al conducente che sta andando troppo veloce. In risposta, egli può eseguire un'azione, come ad esempio premere sul pedale del freno per rallentare la macchina. L'indicatore del livello carburante misura quanto carburante rimane in macchina; se si vede che il livello è troppo basso, si può dire al conducente di trovare un distributore di benzina. In risposta, egli può eseguire un'azione: sollevare la leva dell'indicatore di direzione per indicare una svolta a destra e girare il volante per andare nella stazione di servizio.

Il pilota della bicicletta e il conducente della vettura ricevono i dati dai sensori, decidono quali azioni intraprendere e fanno in modo che vengano eseguite. Un *robot* è un sistema in cui questo processo—riceve i dati, decidere un'azione, eseguire l'azione—è svolta da un elaboratore, di solito senza la partecipazione di un essere umano.

Il robot Thymio-II e l'ambiente Aseba VPL

Il Thymio II è un piccolo robot destinato a fini didattici (Figura 1.1). Il robot include sensori in grado di misurare la luce, il suono e la distanza, ed è in grado di rilevare quando i pulsanti vengono toccati e quando il corpo del robot viene urtato. L'azione più importante che si può effettuare è quella di spostarsi utilizzando due ruote, ciascuna alimentata da un proprio motore. Altre azioni includono la generazione di suoni e accendere e spegnere luci.

Nel resto di questo documento, il robot Thymio II sarà chiamato semplicemente Thymio. Si farà sempre riferimento alla versione II del robot.

ASEBA è un ambiente di programmazione per i piccoli robot mobili come Thymio. VPL è un componente di ASEBA di *programmazione visuale* che è stato progettato

per programmare Thymio in modo semplice attraverso blocchi di eventi e di azioni. Questo tutorial presuppone che il programma ASEBA sia stato installato sul vostro computer; se non è così, andate su https://aseba.wikidot.com/it:downloadinstall, selezionare il sistema operativo, scaricate il programma ed installatelo.

Schede di Riferimento

Troverete utile stampare le schede di riferimento, che sono nello stesso file zip con questo documento o disponibili online[1]:

- Una singola pagina che riassume eventi e azioni (online).
- Una doppia pagina che può essere piegata in tre per creare una comoda carta da tenere a portata di mano. Riassume l'interfaccia del VPL, i blocchi eventi e azioni, e include esempi di programmi (online)

Sommario dei capitoli

Ecco una panoramica dei capitoli di questo tutorial. Per ogni capitolo, diamo l'argomento principale coperto, nonché le liste dei blocchi di eventi e azioni che vengono introdotti. È possibile utilizzare la panoramica per decidere quali capitoli usare e in quale ordine, seguendo le linee guida riportate sotto.

Capitolo 1

Argomenti: Il Robot Thymio, ambiente di programmazione.

Eventi: Bottoni **Azioni**: Colori della parte superiore


Capitolo 2

Argomenti: Coppie evento-azione.

Eventi: Bottoni **Azioni**: Colori della parte superiore, colori della parte inferiore.



Capitolo 3

Argomenti: Muoversi, sentire.

Eventi: Bottoni, sensori del terreno **Azioni**: Motori

[1] Il link può essere trovato sulla pagina *Programmare Thymio II*: https://aseba.wikidot.com/it:thymioprogram.

Capitolo 4

Argomenti: Controllo a retroazione (feedback), velocità dei motori.

Eventi: Sensori frontali

Azioni: Motori

Capitolo 5

Argomenti: Seguire una linea.

Eventi: Sensori del terreno

Azioni: Motori

Capitolo 6

Argomenti: Suoni, urti.

Eventi: Urto, Rumore intenso

Azioni: Musica, colori della parte superiore, colori della parte inferiore

Capitolo 7

Argomenti: Timer.

Eventi: Timer scaduto

Azioni: Predisponi timer

Capitolo 8

Argomenti: Stati.

Eventi: Stato associato ad un evento

Azioni: Cambio stato

Capitolo 9

Argomenti: Contare, aritmetica binaria.

Eventi: Stato associato ad un evento

Azioni: Cambio stato (avanzato)

Capitolo 10

Argomenti: ambiente Aseba studio.

Linee Guida

I Capitoli 1–2 sono un'introduzione essenziale al robot, all'ambiente e al principale costrutto di programmazione—la coppia evento-azione.

I Capitoli 3–5 presentano gli eventi, le azioni e gli algoritmi per la costruzione di robot mobili autonomi e dovrebbero essere al centro di ogni attività che preveda l'utilizzo del sistema.

Il Capitolo 6 descrive le caratteristiche del robot che possono essere divertenti da usare, ma non sono essenziali. È possibile saltare il capitolo o lo si può utilizzare subito dopo i capitoli introduttivi per un'introduzione più dolce alla robotica.

Il Capitolo 7 mostra come utilizzare eventi a tempo. Questo è un argomento importante, ma è indipendente dagli altri e può essere saltato se il tempo manca.

Il Capitolo 8 utilizza funzioni disponibili nella modalità avanzata del VPL. Le macchine a stati sono un costrutto fondamentale utilizzato in robotica e questo materiale deve essere utilizzato in attività con studenti che siano sufficientemente maturi per capire. Il capitolo 9 è un capitolo opzionale che mostra come utilizzare gli stati per implementare conti aritmetici.

Il Capitolo 10 punta al passo successivo: l'utilizzo dell'ambiente testuale Studio, che offre significativamente più supporto per lo sviluppo di robot di quanto non faccia l'ambiente VPL.

Capitolo 1

Il tuo primo Progetto di Robotica

Conoscere il tuo Thymio

La Figura 1.1 mostra la parte anteriore e superiore del Thymio II. Sulla parte superiore è possibile vedere il pulsante centrale circolare (A) e quattro tasti direzionali (B). Dietro i pulsanti, la luce verde (C) indica quanta carica rimane nella batteria. Sul retro ci sono le luci superiori (D), che sono state impostate su rosso in questa immagine. Ci sono luci simili nella parte inferiore (impostate in verde nella Figura 3.2). I piccoli rettangoli neri (E) sono sensori che imparerai a conoscere nel Capitolo 4. È possibile ignorare le piccole luci rosse per ora.

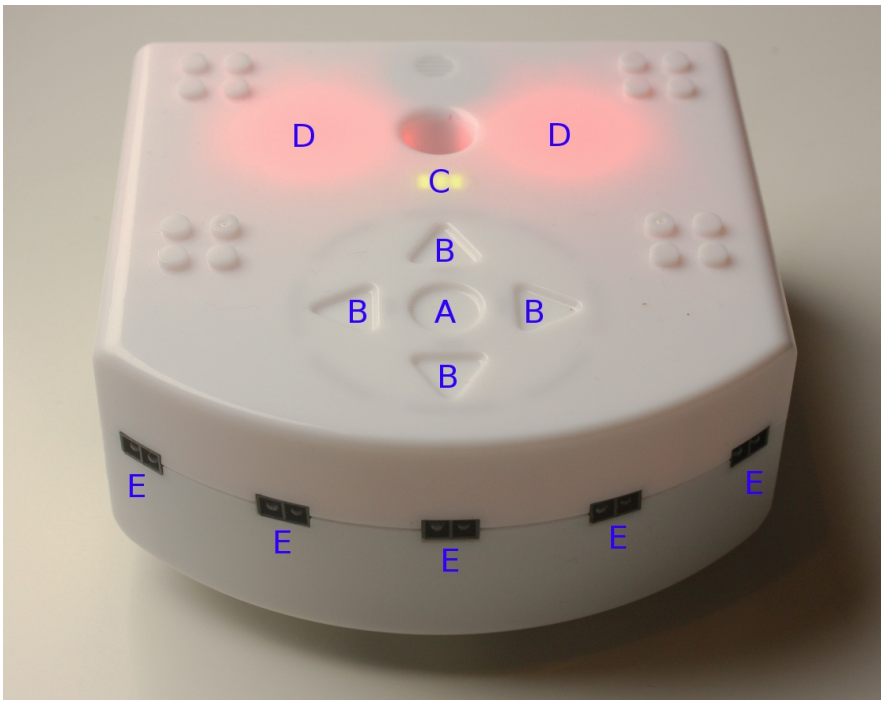


Figura 1.1: Il robot Thymio visto da davanti

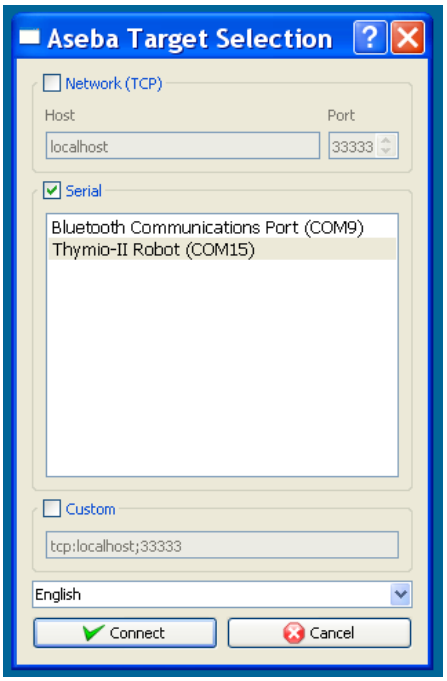


Figura 1.2: Connessione a Thymio, attraverso la porta seriale (USB)

Connettere il robot ed eseguire il VPL

Collega il vostro robot Thymio al computer con un cavo USB; il robot suonerà una sequenza di toni. Se il robot è spento, accenderlo toccando il tasto centrale per cinque secondi. Fai partire l'esecuzione di VPL facendo doppio clic sull'icona .

Informazione importante

Quando nel testo appare una piccola icona, nel margine viene anche visualizzata un'immagine più grande, per maggior chiarezza.

VPL solitamente è in grado di connettersi automaticamente al tuo robot. In caso contrario, sarà visualizzata la finestra mostrata in Figura 1.2. Seleziona la casella accanto a **Seriale**, clicca su **Thymio Robot** sotto di essa, seleziona una lingua, quindi fai clic su **Connetti**. A seconda della configurazione del computer e il sistema operativo che si sta utilizzando, ci possono essere diverse voci in questa tabella e i dati di **Thymio** possono essere diversi da quanto mostrato in figura.

Trucco

E 'anche possibile accedere a VPL da ASEBA Studio, l'ambiente di programmazione basato su testo, attraverso il plugin VPL che si trova nella zona degli *Strumenti* in basso a sinistra dello schermo.

L'interfaccia utente del VPL

L'interfaccia utente del VPL è mostrata sotto. Ci sono sei aree nell'interfaccia:

1. Una barra degli strumenti con le icone per l'apertura, il salvataggio, l'esecuzione di un programma, ecc...
2. Una zona programma in cui vengono costruiti i programmi per il controllo del robot.
3. Indica se il programma che si sta creando è ben formato oppure no.
4. Una colonna con blocchi di eventi disponibili per costruire il tuo programma.
5. Una colonna con blocchi di azione a disposizione per costruire il tuo programma.
6. La traduzione in testo del programma (vedere in fondo a questa pagina).

I blocchi di evento e azione disponibili saranno descritti nel corso di questo documento.

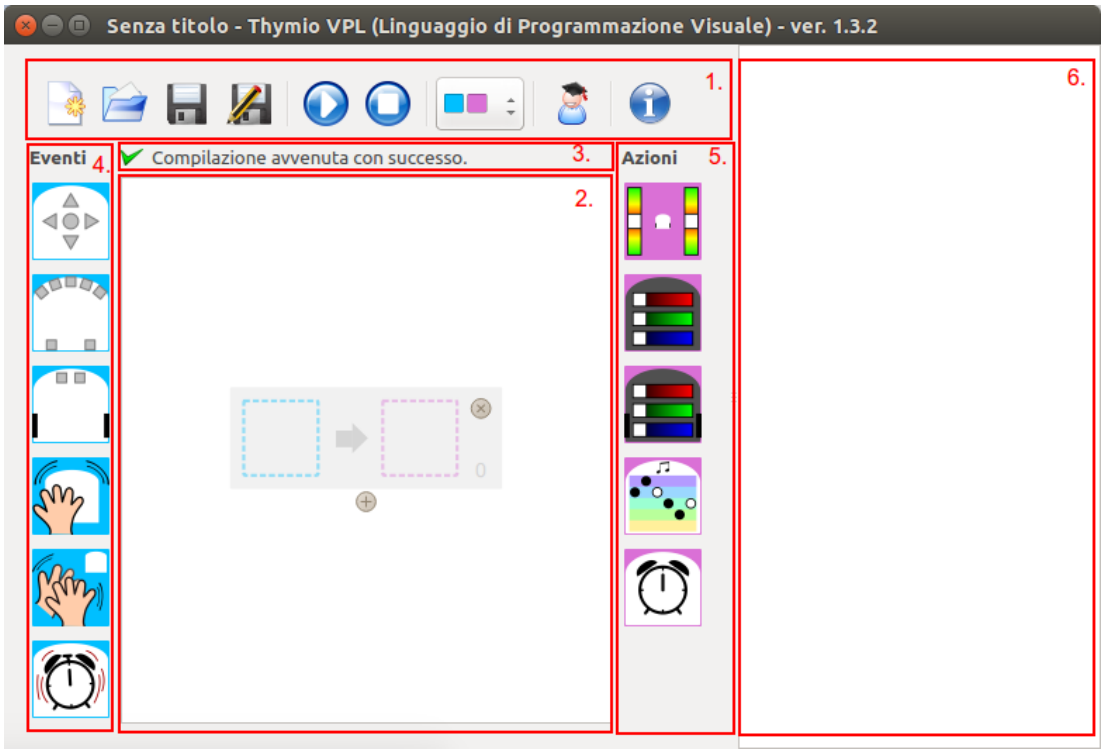


Figura 1.3: La schermata del VPL

> **Per andare oltre**
>
> Quando si costruisce un programma utilizzando VPL, il programma di testo che verrà caricato nel robot appare nella parte destra della finestra. Se sei curioso e vuoi capire questo linguaggio, è possibile leggere il tutorial della modalità testuale (`http://aseba.wikidot.com/it:thymiotutoriel`).

Scrivere un programma

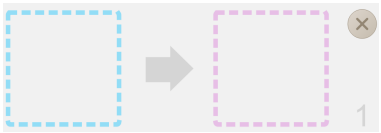
Quando si avvia VPL, viene visualizzata un'area vuota del programma; se, dopo aver costruito un pezzo di programma, si desidera cancellare il contenuto dell'area di programma, fare clic su  .

Un programma in VPL è costituito da uno o più *coppie evento-azione*, ciascuna fatta da un blocco di evento e da un blocco di azione. Ad esempio, la coppia:

fa in modo che la luce superiore del robot sia rossa quando si tocca il pulsante frontale sul robot.

> **Informazione importante**
> Il significato di una coppia evento-azione:
> **Quando si verifica l'evento, il robot esegue l'azione.**

Cerchiamo di costruire una coppia evento-azione. Nella zona interessata dal programma si vede un contorno di una coppia:



Sulla sinistra il quadrato azzurro è lo spazio per l'evento; sulla destra il quadrato rosa è lo spazio per l'azione. Per portare un blocco da lato all'area di programma (zone 4 e 5 della Figura 1.3), è possibile fare clic su di esso o trascinarlo nella casella corrispondente tenendo premuto il tasto sinistro del mouse e rilasciandolo quando il blocco è al suo posto.

Inizia portando il blocco per l'evento Bottoni  nel quadrato blu. Ora, porta il blocco di azione Colore Superiore  nel quadrato rosa. Hai costruito una coppia evento-azione!

Ora dobbiamo modificare l'evento e l'azione per fare quello che vogliamo. Per l'evento, fare clic sul pulsante anteriore (il triangolo in alto); si colorerà di rosso:

Questo specifica che **l'evento si verifica quando il pulsante anteriore del Thymio viene toccato**.

Il blocco di azione Colore Superiore contiene tre barre con i colori primari rosso, verde e blu; all'estremità sinistra di ogni barra vi è un quadrato bianco. La barra del colore con il suo quadrato bianco si chiama *cursore*. Trascina il quadrato bianco a destra e poi di nuovo a sinistra, e vedrai i cambiamenti del colore di sfondo del blocco. Mescolando

questi tre colori primari rosso, verde e blu si possono creare moltissimi altri colori secondari. Ora sposta il cursore del rosso fino a quando il quadrato è all'estrema destra, e sposta i cursori verdi e blu fino a quando sono all'estrema sinistra. Il colore del blocco sarà tutto rosso senza né blu senza né verde:

Salvare il programma

Prima di eseguire il programma, salvalo sul tuo computer. Fare clic sull'icona nella barra degli strumenti. Ti verrà chiesto di dare un nome al programma; scegli un nome che ti aiuterà a ricordare ciò che fa il programma, ad esempio **display-red**. Scegli la posizione in cui si desidera salvare il programma, sul desktop, per esempio, e fai clic su Salva.

Eseguire il programma

Per eseguire il programma, clicca su nella barra degli strumenti. Tocca il pulsante frontale sul robot; la luce sulla parte superiore del robot deve diventare di colore rosso.

> **Congratulazioni!**
> Hai creato ed eseguito il tuo primo programma. Il suo comportamento è:
> **Quando si tocca il pulsante avanti il Thymio diventa rosso.**

Spegnere il robot

Quando hai finito di lavorare con il robot Thymio, è possibile spegnerlo toccando il pulsante centrale per quattro secondi fino a quando si sente una sequenza di toni discendenti. La batteria del robot continuerà la carica fintanto che il robot è collegato a un computer. La luce rossa accanto al connettore del cavo USB significa che il robot è in carica; diventa blu quando la carica è completata (Figura 1.4). È possibile scollegare il cavo quando non si utilizza il robot.

> **Trucco**
> È possibile ricaricare il robot più velocemente usando un caricabatterie del cellulare con un connettore micro-USB.

Se il cavo USB si scollegasse durante la programmazione, VPL attenderà che il collegamento venga effettuato di nuovo. Controllare entrambe le estremità del cavo,

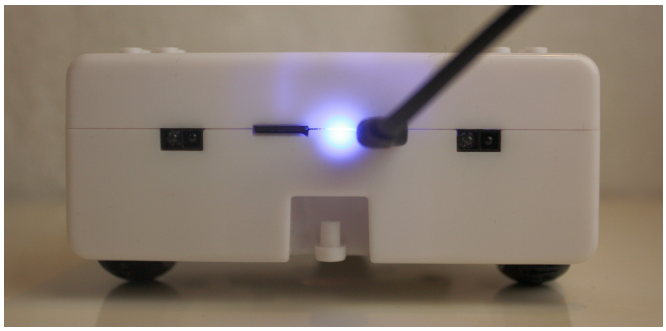


Figura 1.4: Il retro di Thymio che mostra il cavo USB e la luce di ricarica in corso

ricollegare e verificare se VPL funziona. Se hai un problema, si può sempre chiudere VPL, ricollegare il robot e aprire di nuovo VPL.

Modificare un programma

- Per eliminare una coppia evento-azione, fai clic su in alto a destra della coppia.

- Per aggiungere una coppia evento-azione, fai clic su presente sotto ogni coppia.
- Per spostare una coppia evento-azione in un'altra posizione nel programma, trascinala e rilasciala nella posizione desiderata.

Aprire un programma esistente

Supponiamo di aver salvato il programma e spento il robot e il computer, ma in seguito si desidera continuare a lavorare sul programma. Collega il robot ed manda in esecuzione VPL come descritto in precedenza. Fai clic sull'icona e seleziona il programma che desideri aprire, per esempio, **display-red**. Le coppie evento-azione del programma verranno visualizzate nell'area programma, e puoi continuare a modificarlo.

Altre operazioni nell'interfaccia VPL

Nella barra degli strumenti, troverai altre funzionalità:

- **Salva come** : Fai clic su questa icona per salvare il programma corrente con un *nome diverso*. Usalo quando hai un programma che funziona e vuoi provare qualcosa di nuovo senza modificare il programma esistente.

- **Stop** : Questo interrompe il programma in esecuzione e porta la velocità dei motori a zero. Usalo quando il programma ordina al robot di muoversi, ma non include una coppia evento-azione che possa arrestare il motore.

- **Cambio schema colori** : È possibile selezionare una diversa coppia di colori per lo sfondo dei blocchi di evento e di azione.

- **Modalità avanzata** : La modalità avanzata consente l'uso di variabili di stato come descritto in Capitolo 8.

- **Aiuto** : Visualizza la documentazione VPL nel browser (è necessaria una connessione a Internet). Questa documentazione si trova a questo link: `https://aseba.wikidot.com/it:thymiovpl`.

Capitolo 2

Cambiare colore

Colorare

Obiettivo di questo capitolo è creare un programma che visualizzi due diversi colori nella parte superiore del robot Thymio quando i pulsanti anteriore e posteriore vengono toccati, e altri due colori da visualizzare nella parte inferiore del robot quando vengono toccati i pulsanti destro e sinistro.

file di programma **colors.aesl**

Abbiamo bisogno di quattro coppie di eventi-azione. Ci sono quattro eventi—toccare i quattro pulsanti—e un'azione Colore è associata a ogni evento. Nota la differenza tra i blocchi di azione  e  . Il primo blocco cambia colore visualizzato sulla parte superiore del robot, mentre il secondo cambia il colore sulla parte inferiore del robot. Il blocco per la parte inferiore ha due segni neri che rappresentano le ruote.

Questo programma è mostrato nella Figura 2.1(a)

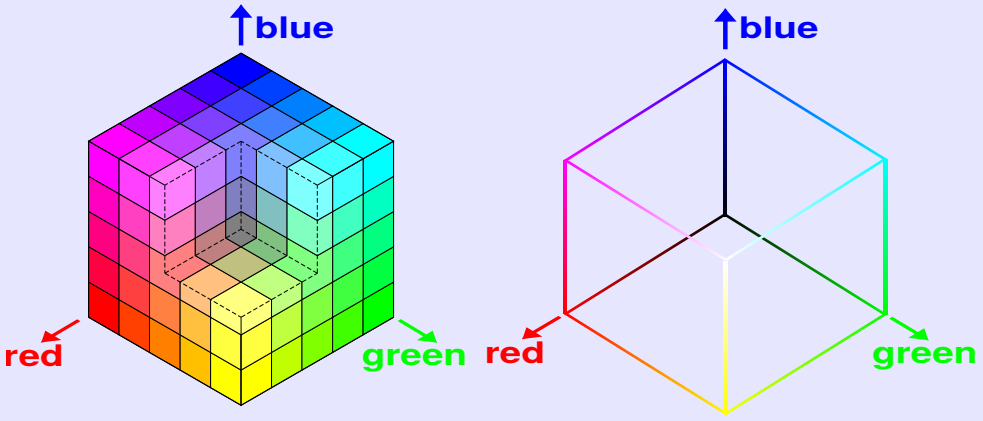
Quali colori vengono visualizzati? Nelle prime tre azioni, il cursore per un colore viene spostato sul bordo destro e visualizzato, mentre i cursori per gli altri due colori vengono spostati verso il bordo sinistro e quindi non sono mescolati. Pertanto, queste azioni mostrano i colori puri rosso, blu e verde, rispettivamente. L'azione associata al tasto sinistro mescola rosso e verde dando giallo. Si può vedere che lo sfondo dell'azione cambia colore a seconda della posizioni dei cursori; Lo sfondo mostra di quale colore il vostro Thymio si illuminerà.

Esegui il programma (icona ) e controlla che toccando i tasti cambi il colore del robot. La Figura 1.1 mostra il Thymio di colore rosso nella parte superiore e la Figura 3.2 mostra il colore verde sulla parte inferiore.

Esercizio 2.1

Sperimenta con i cursori per vedere quali colori vengono visualizzati.

Informazione



Miscelando rosso, verde e blu puoi fare qualsiasi altro colore!

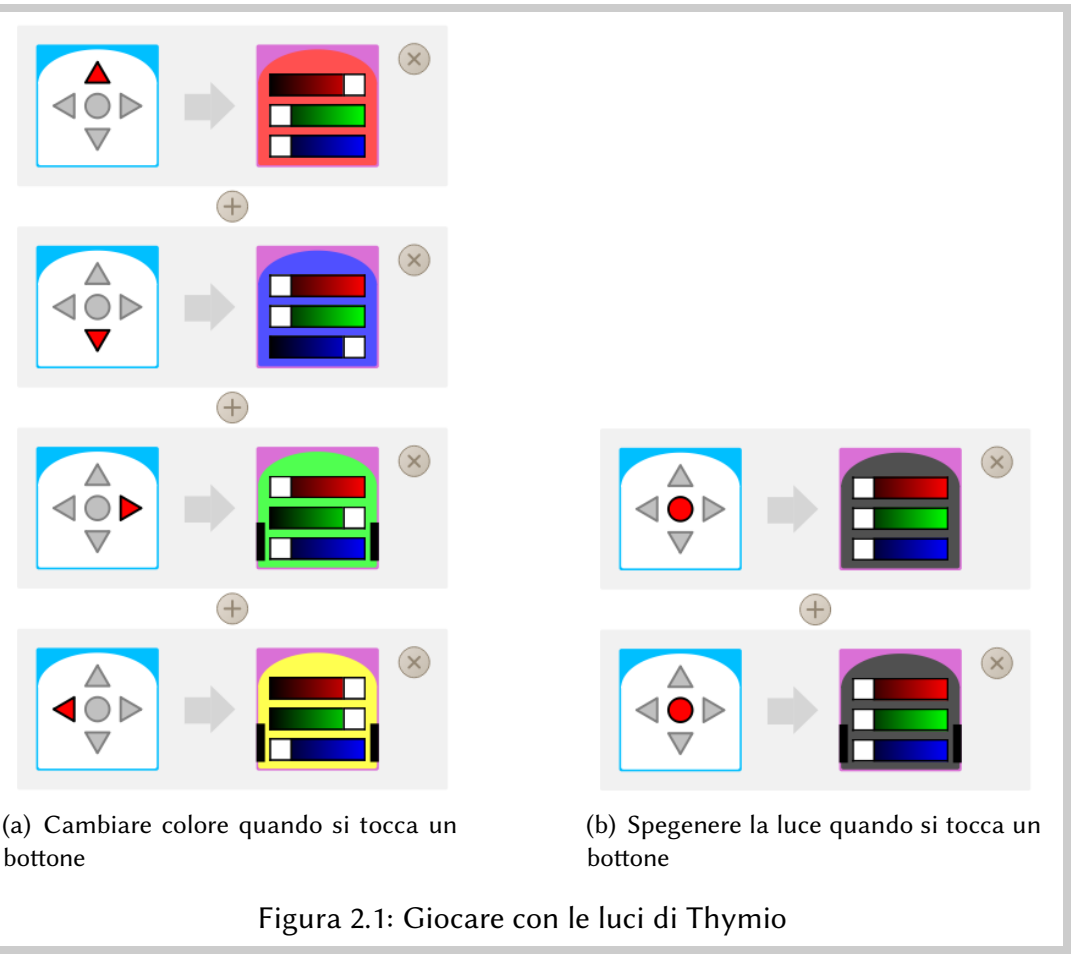
Spegnere la luce

Cerchiamo di modificare il programma in modo che le luci vengano spente quando si tocca il pulsante centrale. Abbiamo bisogno di due coppie di eventi-azione, uno per spegnere la luce superiore e l'altra per spegnere la luce inferiore. Spostando i cursori nel blocco di azione colore a sinistra, come nella Figura 2.1(b), senza colori vengono visualizzati e la spia si spegne. L'*evento è lo stesso in entrambe le coppie*—toccare il pulsante centrale—ma le *azioni sono diverse*—spegnere la luce superiore o inferiore.

Non dimenticare di cliccare sull'icona per eseguire il programma. In futuro, non ripeteremo più di fare clic su questa icona per eseguire un programma.

Coppie evento-azione multiple

- Quando si esegue un programma, tutte le coppie evento-azione del programma sono considerate.
- E'possibile che per diverse coppie vi sia lo stesso evento purchè siano diversi blocchi azione associati.
- Se l'evento e il blocco di azione sono identici in più coppie, VPL mostrerà un messaggio di errore (zona 3 nella Figura 1.3). Non sarai in grado di eseguire il programma fino a quando ci sono errori.



(a) Cambiare colore quando si tocca un bottone

(b) Spegenere la luce quando si tocca un bottone

Figura 2.1: Giocare con le luci di Thymio

Capitolo 3

Mettersi in movimento

Muoversi avanti e indietro

Il robot Thymio ha due motori, uno collegato a ciascuna ruota. I motori possono ruotare in avanti e indietro, facendo in modo che il robot si muova avanti e indietro in linea retta o curva. Cominciamo con un semplice progetto per conoscere i motori.

Il blocco azione Motori  mostra una piccola immagine del robot nel centro insieme a due cursori. I cursori controllano la velocità dei motori, uno per il motore sinistro e uno per il motore destro. Quando il quadrato bianco è centrato nel cursore, il corrispondente motore è spento. È possibile trascinare un quadrato verso l'alto per aumentare la velocità in avanti e verso il basso per aumentare la velocità indietro. Scriviamo un programma per spostare il robot in avanti quando il pulsante anteriore viene toccato e indietro quando il pulsante indietro viene toccato.

File di programma **moving.aesl**

Abbiamo bisogno di due coppie evento-azione (Figura 3.1). Trascina e rilascia i blocchi degli eventi e delle azioni e imposta i cursori allineati per il motore sinistro e destro, a metà corsa verso l'alto per andare avanti e a metà strada verso il basso per andare indietro.

Esegui il programma e tocca i pulsanti per far andare il robot avanti e indietro.

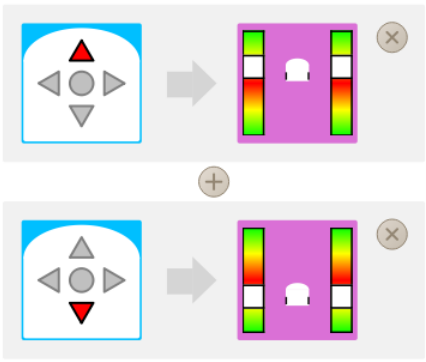


Figura 3.1: Muoversi avanti o indietro

Fermare il robot

Aiuto! Non riesco a fermare i motori del robot!

Fai clic sull'icona per fermare il robot.

Risolviamo questo problema aggiungendo una coppia evento-azione:

Questa fermerà i motori quando si tocca il pulsante centrale. Quando si trascina il blocco di azione motori nella zona interessata dal programma, è già impostato con i cursori al centro corrispondente a spegnere i motori.

Non cadere dal tavolo

Se il robot si muove sul pavimento, nel peggiore dei casi, potrebbe colpire un muro o scollegare il suo cavo, ma se si posiziona il robot su un tavolo, potrebbe cadere fuori, urtare e rompersi! Cerchiamo di fare in modo che il robot si fermi quando si raggiunge la fine di un tavolo.

> **Attenzione!**
> Ogni volta che il robot si muove su un tavolo, stai pronto a prenderlo nel caso in cui dovesse cadere dal bordo.

Gira il Thymio sulla sua schiena. Vedrai sulla parte anteriore due piccoli rettangoli neri con elementi ottici all'interno (Figura 3.2). Questi sono i *sensori del terreno*. Essi mandano un impulso di luce infrarossa e misurano la quantità di luce che viene riflessa. Su un tavolo di colore chiaro, c'è molta luce riflessa, però quando la parte anteriore del robot supera l'estremità del tavolo, ci sarà molta meno luce riflessa. Quando viene rilevata questa situazione vogliamo che il robot si fermi.

> **Trucco**
> Utilizza un tavolo colorato con un colore chiaro, però non utilizzare un tavolo di vetro, in quanto è probabile che, non riflettendo la luce, Thymio creda di non essere su un tavolo!

Trascina il blocco evento sensori terreno nel programma. Ci sono due piccoli quadrati nella parte superiore dell'icona. Cliccando i quadrati il colore cambia dal grigio al bianco al rosso ed infine di nuovo al grigio. Per questo blocco, i significati di questi colori sono:

- **Grigio**: Il sensore non è considerato.

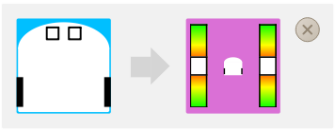
Figura 3.2: La parte inferiore del Thymio con i due sensori del terreno

- **Rosso**: l'evento si verifica se c'è molta luce riflessa.
- **Bianco**: l'evento si verifica se c'è poca luce riflessa.

Informazione

I colori grigio, rosso e bianco usati in un blocco sono arbitrari e se ne sarebbero potuti scegliere altri.

Per fare in modo che il robot si fermi al bordo del tavolo quando c'è poca luce riflessa, fai clic sui quadrati fino a che non sono bianchi e crea la seguente coppia evento-azione:



Posiziona il robot in prossimità del bordo del tavolo, di fronte al bordo e tocca il pulsante frontale. Il robot deve andare avanti e fermarsi prima di cadere fuori dal tavolo.

Esercizio 3.1

Esperimenta con la velocità del robot. Alla massima velocità, il robot è ancora in grado di fermarsi e non cadere fuori dal tavolo? Se no, a partire da che velocità il robot inizia a cadere? Si può evitare che il robot cada quando sta andando indietro?

Trucco

Quando ho eseguito il programma, il robot *è* caduto. Il motivo è stato che la mia bella scrivania ha un bordo arrotondato; prima che il robot rilevasse un basso livello di luce, non era già più stabile e si è ribaltato. La soluzione è stata quella di mettere una striscia di nastro nero vicino al bordo della scrivania.

Capitolo 4

Un robot da compagnia

I robot che costruiamo in questo capitolo sono chiamati *robot autonomi*. Essi mostrano un comportamento indipendente che è normalmente associato con esseri viventi come cani e gatti. Il comportamento si ottiene tramite il *feedback* (o retroazione): il robot sente che si verifica qualcosa nel mondo e modifica il suo comportamento di conseguenza.

Il robot ti obbedisce

In primo luogo programmeremo il robot per obbedire. Normalmente, il robot rimarrà fermo in un posto senza muoversi; quando rileverà la mano di fronte ad esso, si sposterà verso la vostra mano.

Ci sono cinque sensori orizzontali di distanza sul lato anteriore del robot Thymio e due sul retro del robot. Sono simili a quelli sotto il Thymio che abbiamo usato nel Capitolo 3. Portate la vostra mano lentamente verso i sensori; quando si avvicina, una luce rossa apparirà attorno ai sensori che rilevano la mano (Figura 4.1).

Il blocco  viene utilizzato per rilevare se qualcosa è vicino al sensore oppure no. In entrambi i casi fa in modo che un evento si verifichi. I piccoli quadrati grigi (cinque sulla parte anteriore e due nella parte posteriore) vengono utilizzati per specificare quando si verifica un evento. Cliccando su un quadrato esso cambia da grigio a bianco al rosso e ritorna al grigio. Per questo blocco, il significato di questi colori è:

- **Grigio**: Il valore del sensore non influenza il programma..
- **Rosso**: Un evento si verifica quando il sensore rileva un oggetto vicino ad esso.

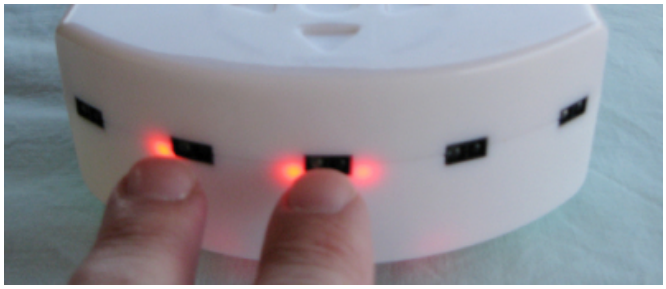


Figura 4.1: La parte anteriore del Thymio. Due sensori rilevano le dita.

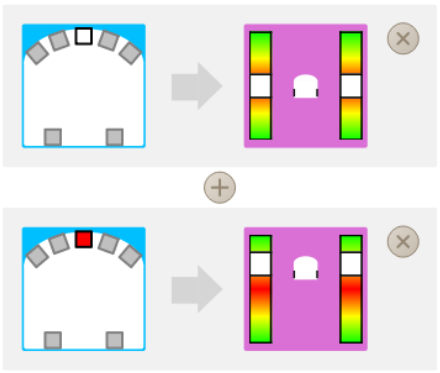


Figura 4.2: Muoversi verso la vostra mano

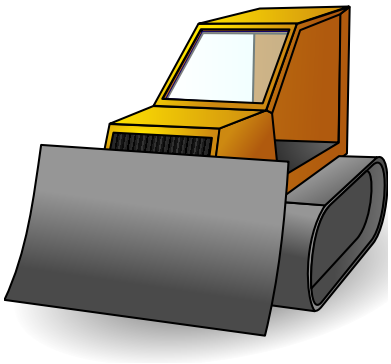


Figura 4.3: Un bulldozer con cingoli

- **Bianco**: Un evento si verifica quando il sensore *non* rileva un oggetto vicino ad esso.

> **sensori del terreno e sensori orizzontali**
>
> Fai attenzione a non confondere il comportamento dei sensori orizzontali con il comportamento dei sensori a terra.
>
> - Per i sensori orizzontali, il quadrato bianco indica che un evento si verifica se non vi è *nulla nelle vicinanze*, mentre il quadrato rosso specifica che un evento si verifica se vi è *qualcosa nelle vicinanze*.
> - Per i sensori di terra, il quadrato bianco indica che un evento si verificherà se *poca luce viene riflessa dalla superficie*, mentre il quadrato rosso indica che un evento si verifichi *se molta luce viene riflessa dalla superficie*.
>
> Il principio fisico di questi due tipi di sensori è simile, ma a causa della loro collocazione differente il loro comportamento è diverso.

Per implementare il comportamento, abbiamo bisogno di due coppie evento-azione indicate in Figura 4.2. Nella prima coppia, il sensore anteriore centrale è bianco e l'azione associata è che i motori sono spenti. Pertanto, quando il robot non vede nulla, non si muoverà e si fermerà se fosse stato in movimento. Nella seconda coppia, il sensore anteriore centrale è rosso e cursori del blocco motore sono trascinati verso l'alto. Pertanto, quando portate la mano vicino alla parte anteriore del robot, si verifica un evento che induce entrambi i motori a funzionare piuttosto velocemente e il robot a muoversi in avanti.

Guidare il robot Thymio

Il robot Thymio non dispone di un volante come una macchina o di un manubrio come una bicicletta. Così come può girare? Il robot usa una *guida differenziale*, che è comunemente utilizzata da veicoli cingolati come il bulldozer (Figura 4.3). Invece di girare un manubrio in una direzione desiderata, i cingoli o le ruote di sinistra e destra

sono mossi da motori individuali a *diverse* velocità. Se il cingolo di destra si muove più velocemente di quello di sinistra, il veicolo gira a sinistra, e se il cingolo di sinistra si muove più velocemente di quello di destra, il veicolo gira a destra.

In VPL è possibile implementare la guida differenziale sul robot Thymio impostando i cursori sinistro e destro di un blocco di azione Motori, e quindi le velocità delle ruote, a diversi valori. Maggiore è la differenza tra le due velocità, più stretta è la curva. Per realizzare una grande differenza di velocità, si può far girare una ruota in avanti e una indietro. In questo caso, se una ruota avanza ad una certa velocità, mentre l'altra gira all'indietro alla *stessa* velocità, il robot Thymio gira sul posto. Ad esempio, nel blocco di azione motori  , il cursore sinistro ha fissato una velocità veloce indietro, mentre il cursore a destra ha fissato una velocità in avanti veloce. Il risultato è che il robot girerà a sinistra, come indicato dalla piccola immagine del robot.

Esperimento con una coppia evento-azione come ad esempio:

Imposta i cursori sinistro e destro, esegui il programma e tocca il pulsante centrale; per fermare il robot clicca su  . Ora è possibile spostare i cursori e riprovare.

> **Trucco**
>
> L'icona di Thymio del blocco azione motori mostra al centro un'animazione del movimento del robot quando si spostano i cursori.

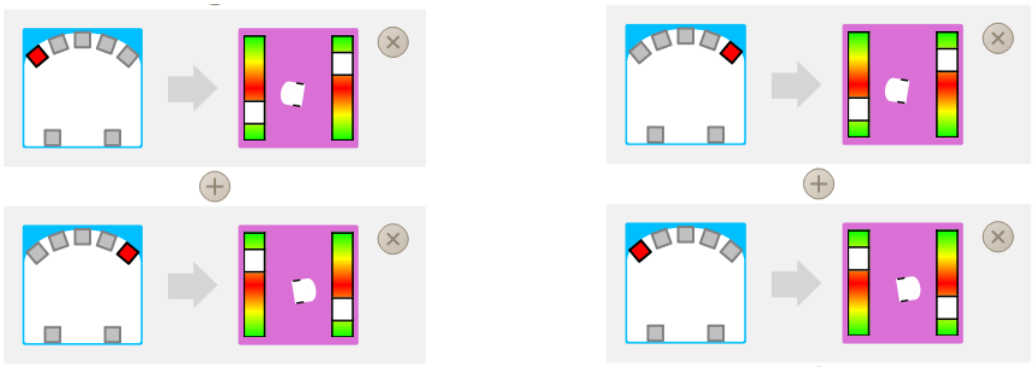
Al robot piaci

Un vero e proprio animale domestico ti segue in giro. Per fare in modo che il robot segua la mano, aggiungi due ulteriori coppie evento-azione: se il robot rileva un oggetto davanti al suo sensore più a sinistra, gira a sinistra, mentre se rileva un oggetto di fronte al suo sensore più a destra, gira a destra.

File di programma **likes.aesl**

Il programma per il robot che ti segue è costituito da due coppie di eventi-azione, come mostrato nella Figura 4.4(a). Sperimenta con i cursori su ogni blocco di azione motori.

> **Esercizio 4.1**
>
> Modifica il comportamento del robot da compagnia in modo che inizi a muoversi in avanti quando il programma viene eseguito e si fermi quando rileva il bordo di un tavolo (o una striscia di nastro adesivo nero).



(a) Al robot piacete

(b) Al robot non piacete

Figura 4.4: Programmi per il robot da compagnia

Come spiegato nel Capitolo 3, molta luce sarà riflessa da una superficie bianca, mentre pochissima luce viene riflessa da una superficie nera. Dovrai sperimentare con il blocco sensore orizzontale per determinare quando cliccare su un quadrato bianco e quando su un quadrato rosso, a seconda del piano del tavolo o dove si posiziona il robot.

Esercizio 4.2

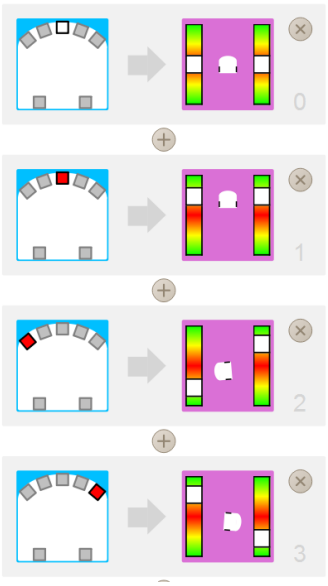
Che cosa succede se si cambia l'ordine delle coppie evento-azione che hai usato nell'esercizio precedente?

Al robot non piaci

A volte il tuo animale domestico può essere di cattivo umore e si allontana dalla tua mano. Scrivi un programma che riproduce questo comportamento nel robot.

File di programma **does-not-like.aesl**

Apri il programma per l'animale domestico che ti ama e scambia l'associazione degli eventi con le azioni. Il rilevamento di un ostacolo dal sensore di sinistra farà in modo che il robot giri a destra, mentre il rilevamento di un ostacolo dal sensore di destra farà sì che il robot svolti a sinistra, come mostrato nella Figura 4.4(b).



```
onevent prox # ogni 10 secondi
    if prox.horizontal[2] < 400 then
        motor.left.target = 0
        motor.right.target = 0
    end
    if prox.horizontal[2] > 500 then
        motor.left.target = 300
        motor.right.target = 300
    end
    if prox.horizontal[0] > 500 then
        motor.left.target = -300
        motor.right.target = 300
    end
    if prox.horizontal[4] > 500 then
        motor.left.target = 300
        motor.right.target = -300
    end
```

Figura 4.5: Un programma VPL e il corrispondente programma testuale.

Esercizio 4.3

Esperimento con i sensori. I sensori anteriori orizzontali sono numerati 0, 1, 2, 3, 4 dalla sinistra del robot alla sua destra. I sensori posteriori sono numerati 5 per il sinistro e 6 per quella di destra. Invece di utilizzare i sensori 0 e 4 come prima:

- Usa i sensori 1 e 3 per girare il robot a destra e a sinistra, rispettivamente.
- Usa entrambi i sensori 0 e 1 per girare il robot a sinistra e entrambi i sensori 3 e 4 per girare il robot a destra.
- Aggiungi coppie evento-azione per i sensori posteriori 5 e 6.

Impostare i cursori con precisione (avanzato)

È difficile impostate i cursori con precisione in modo che, per esempio, entrambi i motori funzionino alla stessa velocità. Osservando la traduzione delle coppie evento-azione in un programma di testo è possibile migliorare la precisione. La Figura 4.5 mostra il programma in cui il robot ti segue in giro insieme con la traduzione del testo a destra della finestra VPL. Questo testo viene modificato automaticamente quando si modificano le coppie evento-azione.

La linea `onevent prox` significa: ogni volta che l'evento di campionamento dei sensori di prossimità orizzontale (abbreviato *prox*) ha luogo (avviene 10 volte al secondo), le linee che seguono verranno eseguite.

Quando l'evento avviene, Thymio controlla i valori dei sensori utilizzando una condizione del tipo `if` … `then` … `end`. Si inizia testando il sensore 2 (centrale an-

teriore), come si vede da `prox.horizontal[2]`. Se questo valore è inferiore a 400, allora Thymio imposta le velocità del motore destro e sinistro a 0 con i comandi `motor.left.target = 0` e `motor.right.target = 0`. Ogni blocco `if` … `then` … `end` verifica un sensore specifico ed esegue o meno l'azione associata, in funzione del risultato del test. Quindi corrisponde ad una coppia evento-azione:

0. test se non c'è qualcosa di fronte; se questo è vero, Thymio si ferma.
1. test se qualcosa è di fronte; se questo è vero, Thymio va avanti.
2. test se c'è qualcosa a sinistra; se questo è vero, Thymio gira a sinistra.
3. test se c'è qualcosa a destra; se questo è vero, Thymio gira a destra.

Infine, una volta che il Thymio ha letto tutti questi sensori, attende il prossimo evento `prox` e ricomincia questi test, indefinitamente.

Per scrivere programmi in modalità testo, usare l'ambiente AsebaStudio (Capitolo 10).

Trucco

Spostando i cursori sui blocchi azione motori, vedrai che la velocità dei motori (`motor.X.target`) cambia a passi di 50 nella gamma da -500 a 500. Spostando con attenzione i cursori, è possibile impostare le velocità per uno qualsiasi di questi valori.

Capitolo 5

Il robot trova la strada da solo

Un'escursione in montagna è una semplice attività: prendi un paio di scarpe da trekking e segui un percorso. Per un robot, seguire un percorso a terra può anche essere molto utile. Considera un magazzino con carrelli robotizzati che portano gli oggetti in una zona centrale di dispacciamento. Ci sono linee tracciate sul pavimento del magazzino e il robot riceve istruzioni di seguire alcune linee fino a raggiungere lo scomparto di archiviazione dell'oggetto desiderato. Per farlo, il robot deve vedere queste righe. Scriviamo ora un programma che faccia in modo che il robot segua una linea sul pavimento.

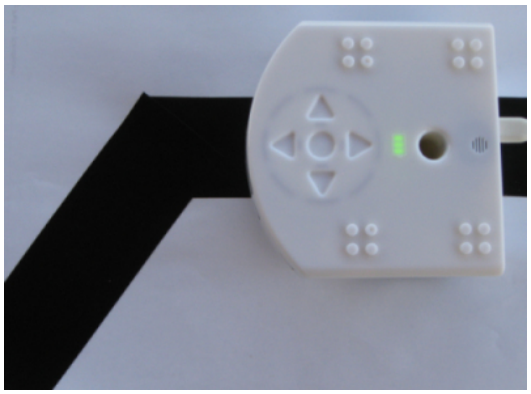
File di programma **follow-line.aesl**

Il compito di seguire una linea mette in evidenza tutta l'incertezza di costruire robot nel mondo reale: Il robot deve affrontare l'incertezza delle sue percezioni e delle sue azioni. Per esempio, la linea potrebbe non essere perfettamente rettilinea, la polvere può oscurare parte della linea, o la sporcizia può fare in modo che una ruota si muova più lentamente rispetto all'altra. Per seguire una linea, il robot deve utilizzare un *controller* che decide quanta potenza applicare per ogni motore a seconda dei dati ricevuti dai sensori.

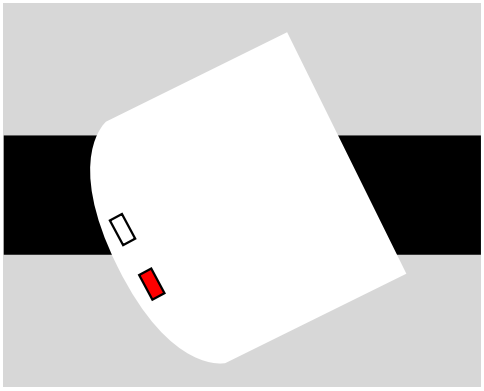
La linea e il robot

Per seguire una linea, usiamo i sensori di terra (Capitolo 3). Ricorda che questi funzionano con l'invio di luce infrarossa (invisibile all'occhio umano) e misurano quanto viene riflessa indietro. Se il pavimento è di colore chiaro, il sensore rileverà molta luce riflessa e si verificherà l'evento  . Abbiamo bisogno di una linea che causi l'evento poca luce riflessa  . Questo è facile da fare stampando una linea nera, dipingendo o mettendo nastro isolante nero sul pavimento (Figura 5.1(a)). La linea deve essere sufficientemente ampia in modo che entrambi i sensori di terra leggano il nero quando il robot segue correttamente la linea. Una larghezza di 5 centimetri è sufficiente per fare in modo che il robot segua la linea, anche se ci sono piccole deviazioni.

In primo luogo, facciamo andare avanti Thymio ogni volta che *entrambi* i sensori rilevano una superficie scura (il robot è sulla linea) e lo facciamo fermare ogni volta che *entrambi* i sensori rilevano una superficie chiara (il robot non è sulla linea). Le coppie evento-azione sono riportate nella Figura 5.2(a).

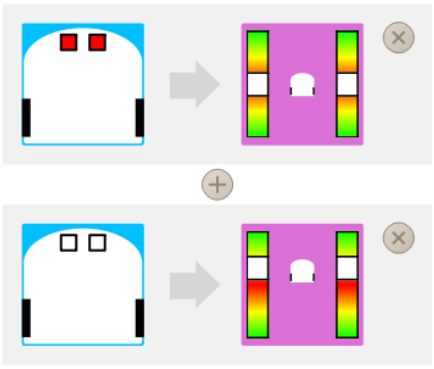


(a) Thymio segue una striscia di nastro

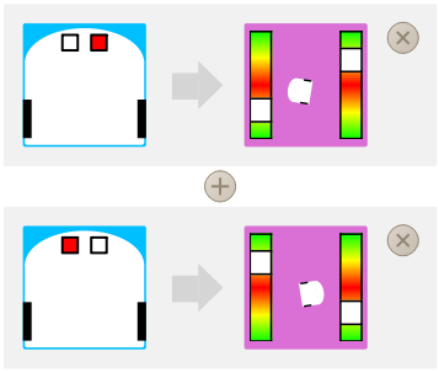


(b) Il sensore sinistro è fuori dal nastro e il sensore destro è sul nastro

Figura 5.1: Thymio su un nastro nero



(a) Avvia e ferma il robot



(b) Correggi le deviazioni

Figura 5.2: Un programma per seguire una linea

Trucco

Assicurati di utilizzare un cavo USB che sia abbastanza lungo (diciamo, due metri), in modo che Thymio possa restare collegato al computer anche quando si sposta. È possibile trovare le prolunghe in qualsiasi negozio di computer.

Il tuo primo controller

Il passo successivo è quello di programmare il controller che segue la linea::

- Se il robot si muove fuori dal nastro a *sinistra*, come in Figura 5.1(b), il sensore di *sinistra* rileverà il pavimento mentre il sensore di *destra* sta ancora rilevando

il nastro; in tal caso il robot deve ruotare leggermente a *destra*.

- Se il robot si muove fuori dal nastro a *destra*, il sensore di *destra* rileverà il pavimento mentre il sensore di *sinistra* sta ancora rilevando il nastro; in tal caso il robot deve ruotare leggermente a *sinistra*.

Sono necessarie due coppie evento-azione, come mostrato nella Figura 5.2(b).

Fissare i parametri

E' facile vedere che se il robot scappa dal bordo sinistro del nastro, si deve girare a destra, come nella Figura 5.1(b). La vera domanda è quanto stretta deve essere la curva? Se la curva è troppo dolce, *anche* il sensore di destra potrebbe uscire dal nastro prima che il robot torni indietro; se la curva è troppo aggressiva, potrebbe far uscire il robot dall'altro lato del nastro. In ogni caso, curve aggressive possono essere pericolose per il robot e tutto ciò che sta portando.

In questo programma, è possibile impostare le velocità dei motori di destra e di sinistra in ogni blocco azione motori. Avrai bisogno di fare un po' di prove con questi valori fino a quando il robot segua in modo *affidabile* la linea. Affidabile qui significa che il robot riesce a seguire con successo la linea più volte. E' necessario eseguire diversi test per assicurarsi che il programma funzioni, mettendo il robot sulla linea in una posizione leggermente diversa e che punti in una direzione leggermente diversa.

Ci sono molti modi per configurare i blocchi di azione motori. La velocità di avanzamento del robot sulla linea è un parametro importante. Se è troppo veloce, il robot può scappare dalla linea prima che le azioni di correzione possano influenzare la sua direzione. Tuttavia, se la velocità è troppo lenta, nessuno comprerà il tuo robot da utilizzare in un magazzino.

Se il Thymio esce dalla linea, che cosa dovrebbe fare? Se fa una brusca virata (un motore va avanti e l'altro indietro), il robot ritorna rapidamente sulla linea, però i suoi movimenti saranno molto a scatti. D'altra parte, se il robot fa una curva dolce (un motore va leggermente più veloce rispetto all'altro), il robot si sposta uniformemente ma può perdere la linea. Dovrai sperimentare per trovare buoni compromessi.

Esercizio 5.1

Il robot si arresta quando entrambi i sensori del terreno rilevano che sono fuori dal nastro. Modifica il programma in modo che il robot faccia una dolce svolta a sinistra nel tentativo di trovare nuovamente il nastro. Provalo su un nastro con una svolta a sinistra come quello mostrato nella Figura 5.1(a). Prova ad aumentare la velocità in avanti del robot. Cosa accade quando il robot arriva alla fine del nastro?

Esercizio 5.2

Modifica il programma dall'esercizio precedente in modo che il robot giri a destra quando viene perso il nastro. Che succede?

Sarebbe bello se potessimo *ricordare* quale sensore è stato l'ultimo a perdere il contatto con il nastro in modo da far sterzare il robot nella direzione giusta per ritrovare il nastro. Nel Capitolo 8 impareremo come Thymio possa ricordare le informazioni.

Esercizio 5.3

Sperimenta con diverse disposizioni delle linee di nastri:

- curve ampie;
- curve strette;
- linee a zig zag;
- linee più ampie;
- linee più strette.

Organizza delle gare di corsa con i tuoi amici: quale robot segue con successo la maggior parte delle linee? Per ogni linea, quale robot la segue nel tempo più breve?

Esercizio 5.4

Discuti su che effetto le seguenti modifiche di Thymio potrebbero avere sulla capacità del robot di seguire una linea:

- gli eventi di rilevamento dei sensori del terreno si verificano più spesso o meno spesso di 10 volte al secondo;
- i sensori sono più lontani o più vicini;
- ci sono più di due sensori di terra nella parte inferiore del robot.

Capitolo 6

Sonagli e fischietti

Prendiamoci qualche momento di svago dai compiti complessi come seguire una linea e divertiamoci un po' con il nostro robot Thymio. In questo capitolo ti mostriamo come il Thymio possa riprodurre musica, rispondere a un suono o reagire quando viene toccato.

Suonare musica

Il robot Thymio contiene un sintetizzatore di suoni e si può programmare per riprodurre melodie semplici utilizzando il blocco di azione musica:

File di programma **bells.aesl**

Non diventerai un nuovo Beethoven—si può suonare solo una sequenza di note alla volta, su cinque toni e due lunghezze differenti—ma si può comporre una melodia che farà distinguere il vostro robot da tutti gli altri. La Figura 6.1 mostra due coppie evento-azione che rispondono con una melodia quando il pulsante anteriore o posteriore vengono toccati. C'è una melodia diversa associato ad ogni evento.

I piccoli cerchi sono le sei note. Una nota nera è una nota breve e una nota bianca è una nota lunga; per passare da una durata all'altra, clicca sul cerchio. Ci sono cinque

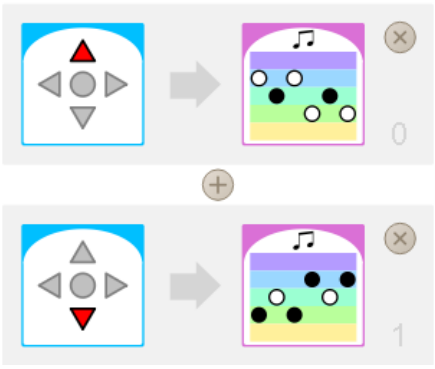


Figura 6.1: Esegui una melodia

righi orizzontali colorati, che rappresentano cinque toni. Per spostare un cerchio su un rigo, fai clic sul *rigo* sopra o sotto il cerchio. Non cercare di trascinare e rilasciare una nota; non funzionerà.

Esercizio 6.1

Scrivi un programma che vi permetterà di inviare un messaggio in codice Morse. Le lettere in codice Morse sono codificate in sequenze di toni lunghi (*linee*) e toni brevi (*punti*). Ad esempio, la lettera *V* è codificato da tre punti seguiti da una linea.

Controlla il tuo robot con i suoni

Il Thymio ha un microfono. L'evento si verificherà quando il microfono avverte un forte rumore, per esempio un applauso con le tue mani. la seguente coppia evento-azione accende le luci della parte inferiore quando battete le mani:

Informazioni

In un ambiente rumoroso potresti non essere in grado di utilizzare questo evento, perché il livello sonoro sarà sempre alto causando ripetuti eventi.

Esercizio 6.2

Scrivi un programma che fa muovere il robot quando batti le mani e lo fa fermare quando si tocca un tasto.

Quindi scrivi un programma che fa il contrario: si avvia quando si tocca un tasto e si ferma quando batti le mani.

Ottimo lavoro, robot!

Gli animali non sempre fanno quello che chiediamo loro di fare. A volte hanno bisogno di una pacca sulla testa per incoraggiarli. Si può fare lo stesso con il tuo robot. Il Thymio contiene un sensore di urto che provoca l'evento in risposta ad un rapido tocco sulla parte superiore del robot. Per esempio, la seguente coppia evento-azione fa sì che le luci superiori si accendano quando si tocca la parte superiore del robot:

Costruisci un programma da questa coppia evento-azione e la seguente coppia che accende le luci inferiori quando batti le mani:

File di programma bu whistles.aesl

Riesci ad attivare solo la parte superiore delle luci? Questo è difficile da fare: un colpetto provoca un suono che può essere abbastanza forte da far accendere pure le luci inferiori. Con un po' di pratica sono riuscito a toccare il robot abbastanza delicatamente in modo che il suono emesso dal colpetto non è stato considerato un evento.

Esercizio 6.3

Scrivi un programma che faccia andare avanti il robot fino a quando non colpisce una parete.

Assicurati che il robot si **muova lentamente** in modo che non si danneggi.

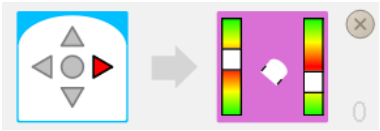
Capitolo 7

Un tempo per amare

Nel Capitolo 4 abbiamo programmato un robot da compagnia a cui piaciamo o non piaciamo. Consideriamo un comportamento più avanzato: un animale timido che non può decidersi se gli piaciamo o no. Inizialmente, l'animale si girerà verso la nostra mano cercando di raggiungerla, ma poi si terrà a distanza. Dopo un po' cambierà idea e tornerà indietro in direzione della nostra mano.

File di programma **shy.aesl**

Il comportamento del robot è il seguente. Quando il tasto destro viene toccato il robot gira a destra. Quando rileva la tua mano, gira a sinistra, ma dopo un po' si rammarica per la sua decisione e gira indietro. Noi sappiamo come costruire coppie evento-azione per la curva iniziale:



e per allontanarsi quando viene rilevata la mano:

Il comportamento di tornare indietro dopo un po' può essere suddiviso in due parti:

- *Quando* il robot inizia ad allontanarsi → *avviare un timer* per due secondi.
- *Quando* il timer scende a zero → *girare* a destra.

Abbiamo bisogno di una nuova *azione* per la prima parte e un nuovo *evento* per la seconda parte.

L'azione è quella di impostare un *timer*, che è come una sveglia ⏰. Normalmente, si imposta una sveglia per un tempo assoluto, ma quando ho impostato la sveglia nel mio smartphone per un tempo assoluto come le ore 7:00, mi dice il tempo relativo: allarme impostato tra 11 ore e 23 minuti da oggi. Il blocco del timer funziona allo stesso modo: si imposta il timer per un certo numero di secondi da quando si verifica l'evento e poi l'azione accade. Il timer può essere impostato per un massimo di quattro secondi. Fai clic su qualsiasi punto all'interno del cerchio nero che rappresenta il quadrante dell'orologio (ma non sul cerchio nero stesso). Ci sarà una breve animazione e poi la quantità di tempo fino a quando l'allarme scatta sarà mostrata di colore blu.

La coppia evento-azione per questa prima parte del comportamento è:

Il timer è impostato per due secondi. Quando l'evento che rileva la presenza della mano si verifica, ci saranno due azioni: ruotare il robot di fianco e l'impostazione del timer.

La seconda parte del comportamento utilizza un evento che si verifica quando l'allarme si spegne, cioè, quando la quantità di tempo impostato sul timer scende a zero. il blocco dell'evento mostra una sveglia che squilla.

Ecco la coppia evento-azione per fare girare il robot di nuovo a destra quando il timer scende:

Esercizio 7.1

Scrivi un programma che faccia muovere il robot in avanti alla massima velocità per tre secondi quando il pulsante in avanti viene toccato; quindi corra all'indietro. Aggiungi una coppia evento-azione per arrestare il robot toccando il pulsante centrale.

Capitolo 8

Stati: non fare sempre le stesse cose (Avanzato)

Un programma in VPL è una lista di coppie evento-azione. Tutti gli eventi sono controllati periodicamente e vengono adottate le azioni appropriate quando gli eventi si verificano. Questo limita i programmi che possiamo creare; per fare cose più complesse abbiamo bisogno di un modo per specificare che alcune coppie evento-azione sono attive in un determinato momento, mentre altre non lo sono. Per esempio, nel Capitolo 5, quando il robot abbandona il nastro, volevamo che svoltasse a sinistra o destra per cercare il nastro a seconda del lato del nastro da cui si era usciti.

La gestione degli stati è supportata nella modalità *avanzata* di VPL. Clicca su prima di iniziare a lavorare sui seguenti progetti.

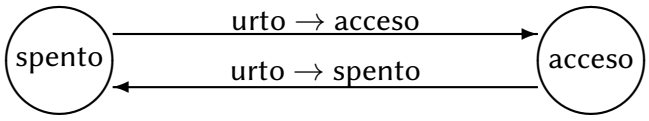
Toc, toc

In molti programmi, abbiamo utilizzato un pulsante per avviare il comportamento del robot e un altro per fermarlo. Considera, però, l'interruttore di accensione sul mio computer: Lo stesso interruttore viene usato per accendere il computer e per spegnerlo; l'interruttore *ricorda* se è in stato **acceso** o in stato **spento**. L'interruttore include una piccola luce verde che indica il suo stato attuale.

Scrivi un programma che accende le luci del robot quando viene urtato e le spegne quando viene urtato nuovamente.

File di programma **tap-on-off.aesl**

E' utile mostrare il comportamento richiesto con un *diagramma a stati*:



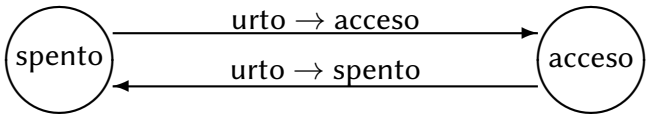
Nel diagramma ci sono due stati indicati da cerchi marcati con i nomi degli stati **spento** e **acceso**. Dallo stato **spento** il robot può andare allo stato **acceso** e tornare indietro, ma solo seguendo le istruzioni sulle frecce. Le istruzioni descrivono quando una transizione da uno stato all'altro può accadere e cosa succede quando si verifica:

- *Quando* il robot è nello stato **spento** ***e*** avviene l'evento *urto* → *accende* la luce ***e*** va nello stato **acceso**.

- *Quando* il robot è nello stato **acceso** ***e*** avviene l'evento *urto* → *spegne* la luce ***e*** va nello stato **spento**.

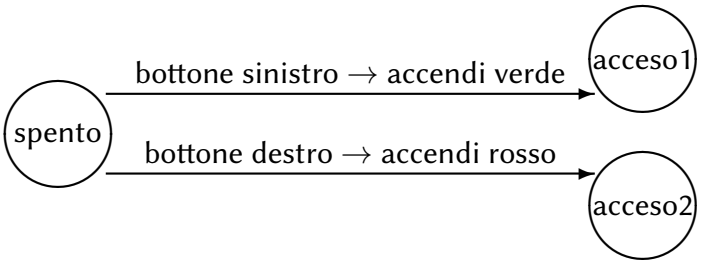
La parola evidenziata "***e***" prima della freccia → significa che ci sono *due condizioni* che devono verificarsi perchè la transizione avvenga: (a) Il robot deve essere in un certo stato e (b) l'evento deve accadere. Quando entrambe le condizioni sono vere avviene la transizione, che fa eseguire sia il cambio di stato che l'azione scritta dopo la freccia →.

E' importante comprendere che le due parti della condizione sono in dipendenti. Nel diagramma a stati (ripetuto qui):



l'evento *urto* (tap) appare due volte, ma l'azione provocata dal verificarsi dell'evento *dipende* dallo stato in cui il robot si trova.

Similmente, in un singolo stato, differenti eventi possono provocare differenti azioni e transizioni a nuovi stati differenti. Nel diagramma seguente:

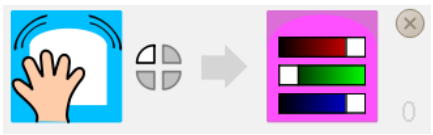


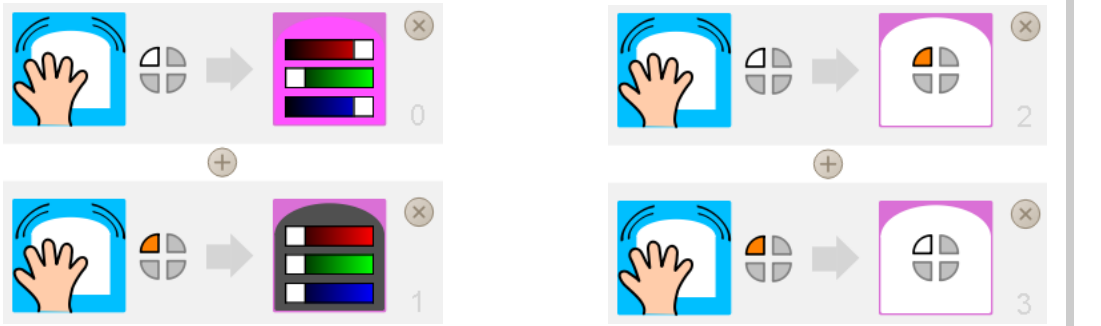
toccare il pulsante sinistro nello stato **spento** fa sì che si accenda la luce verde e si passi nello stato **acceso1**, mentre toccare il pulsante destro *nello stesso stato* causa una diverso azione, si accende la luce rossa e avviene un cambiamento di stato diverso, verso **acceso2**.

Realizzare un diagramma a stati con le coppie evento-azione

La Figura 8.1 mostra la realizzazione del comportamento descritto dalla macchina a stati, con coppie evento-azione.

Nella prima coppia evento-azione, l'evento è composto dal blocco urto con l'indicazione dello stato :





(a) Urtare per accendere e spegnere

(b) Urtare per cambiare di stato

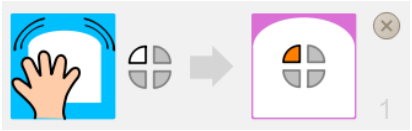
Figura 8.1: L'urto ha risultati differenti in dipendenza dallo stato

Uno stato è indicato da quattro quarti di un cerchio, ciascuno dei quali può essere sia acceso (arancione) o spento (bianco). In questo programma, useremo il quarto anteriore sinistro per indicare se la luce superiore del robot è spento o acceso. In questa coppia, questo quarto è di colore bianco, il che significa che la luce del robot è spenta. Pertanto, il significato della questa coppia è: se il robot è urtato e la luce è spenta, accenderla.

Analogamente, la seconda coppia evento-azione significa: se il robot è urtato e la luce è accesa, spegnerla:

Se si guarda di nuovo il diagramma a stati, si vedrà che solo metà del lavoro è fatto. Infatti, quando si accende o si spegne la luce, dobbiamo anche cambiare lo stato del robot da **spento** a **acceso** o da **acceso** a **spento**. Per questo abbiamo creato due ulteriori coppie evento-azione con il blocco di azione *stato* , come mostrato in Figura 8.1(b).

Il significato del primo è: *quando* il robot è urtato *e* lo stato è **spento**, modificare lo stato di **a**:



Allo stesso modo, il significato della seconda è: *quando* il robot viene urtato *e* lo stato è **acceso**, cambiare lo stato a **spento**:

Facendo riferimento al programma completo con quattro coppie evento-azione in Figura 8.1, vediamo che ogni evento provoca sia un'azione sulla luce che un cambia-

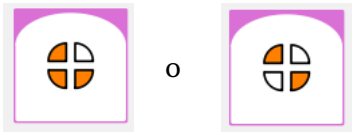
mento dello stato del robot. Sia l'azione che il cambiamento di stato dipendono dallo stato in cui è il robot si trova, chiamato *stato corrente*.

In quanti stati può essere il robot?

Lo stato è indicato da un cerchio diviso in quattro elementi (o quarti). In un blocco evento o in un blocco azione ogni elemento dello stato può essere:

- **Bianco**: l'elemento è *spento*;
- **Arancione** : l'elemento è *acceso*;
- **Grigio** : l'elemento viene ignorato.

Ad esempio, in , gli elementi anteriore sinistro e posteriore destro sono accesi, l'anteriore destro è spento e quello posteriore sinistro non viene preso in considerazione, il che significa che se è associato a un blocco evento, l'evento si può verificare se lo stato è impostato indifferentemente su:



o

Dal momento che ciascuno dei quattro elementi può essere acceso o spento, ci sono 2 $\times$ 2 $\times$ 2 $\times$ 2 = 16 possibili stati:

> **(spento, spento, spento, spento), (spento, spento, spento, acceso), (spento, spento, acceso, spento),**
> ...
> **(acceso, acceso, spento, acceso), (acceso, acceso, acceso, spento), (acceso, acceso, acceso, acceso)**.

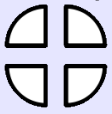
La Figura 8.2(a) enumera graficamente tutti questi stati.

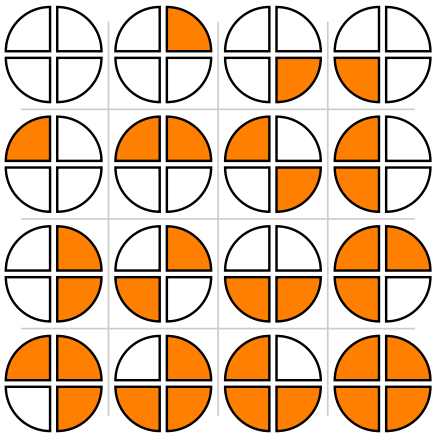
Informazione importante

Lo stato corrente del robot viene visualizzato nel cerchio di LED gialli sulla parte superiore del robot. La Figura 8.2(b) mostra il robot nello stato **(acceso, acceso, acceso, acceso)**.

Informazione

Quando viene eseguito un programma, lo stato iniziale è **(spento, spento, spento, spento)**:





(a) Tutti i possibili stati di Thymio

(b) Il cerchio di LED mostra lo stato

Figura 8.2: Gli stati di Thymio e la loro rappresentazione

Trucco

Se non utilizzi tutti i possibili 16 stati, ma solo 2 o 4, per esempio, sei libero di decidere quali elementi usare per rappresentare il tuo stato. Inoltre, se si hanno due cose diverse che si desidera codificare, e ciascuna di esse ha due valori possibili, è possibile utilizzare due quarti in modo indipendente. Ecco perché la capacità di *ignorare* un quarto è molto utile!

Prendi il topo

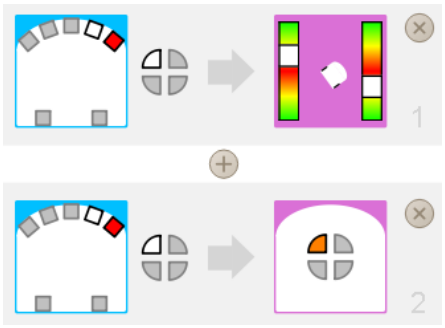
Scrivi un programma per implementare il comportamento di un gatto alla ricerca di un topo: Quando si tocca il pulsante centrale, il robot gira in senso antiorario (da destra a sinistra), alla ricerca di un topo. Se il robot rileva un topo con il suo sensore più a destra, gira in senso orario (da sinistra a destra) fino a quando il topo viene rilevato dal suo sensore centrale, a quel punto si ferma (Figura 8.3(a)).

File di programma **mouse.aesl**

Il seguente diagramma a stati descrive il comportamento del robot:

1. Quando si tocca il pulsante centrale, il robot entra nello stato **cerca a sinistra** e si sposta da destra a sinistra.
2. Quando il robot è in stato **cerca a sinistra** e rileva il topo nel sensore più a destra, si sposta nello stato **cerca a destra** e si sposta da sinistra a destra.

(a) Il gatto ha trovato il topo



(b) Cerca con il sensore più a destra

Figura 8.3: Il gatto robot cerca il topo

3. Quando il robot è nello stato **cerca a destra** e rileva il topo con il sensore centrale, va nello stato **trovato** e si ferma.

Il punto importante da notare è che quando il topo viene rilevato dal sensore di centro, il robot si ferma *solo* se è nello stato **cerca a destra**. Altrimenti (se il topo viene rilevato dal sensore centrale quando il robot è in stato **cerca a sinistra**), non accade nulla.

Vediamo ora come implementare questo comportamento. Rappresentiamo lo stato del robot con l'elemento in alto a sinistra dell'indicatore di stato. Abbiamo scelto bianco per lo stato **cerca a sinistra** e arancione per lo stato **cerca a destra**. Poiché il programma termina quando viene rilevato il topo in stato **cerca a destra** non abbiamo bisogno di rappresentare esplicitamente lo stato **trovato**. Inizialmente, tutti gli elementi dello stato sono **spenti** (bianchi).

La seguente coppia evento-azione implementa il comportamento del punto 1:

Le coppie evento-azione che implementano il punto 2 sono mostrate in Figura 8.3(b). Entrambe utilizzano lo stesso evento, ma le azioni sono diverse. La prima modifica la direzione del movimento del robot e la seconda cambia il suo stato da **cerca a sinistra** a **cerca a destra**.

Il piccolo quadrato accanto al sensore più a destra è impostato su bianco in modo che l'evento si verifichi solo quando solamente il sensore più a destra rileva il topo.

Il punto 3 è implementato dalla seguente coppia evento-azione:

che fa sì che il robot si fermi quando il topo si trova direttamente davanti al sensore centrale. Come previsto dal punto 3 questo avverrà solo durante la ricerca sulla destra: quando il robot è in stato **cerca a destra** come indicato dall'elemento in arancione.

Trucco

Dovrai fare degli esperimeni con la distanza del topo dal robot. Se è troppo vicino al robot, i sensori su entrambi i lati del sensore centrale rilevano anche loro il topo, mentre l'evento richiede che essi *non* lo rilevino.

Esercizio 8.1

Scrivi un programma che fa danzare il robot: si gira a sinistra sul posto per due secondi e poi si gira a destra sul posto per tre secondi. Questi movimenti vengono ripetuti all'infinito.

Esercizio 8.2(Difficile)

Modifica il programma per seguire una linea del Capitolo 5 in modo che il robot giri a sinistra quando lascia il lato destro della linea e giri a destra quando lascia il lato sinistro della linea.

Capitolo 9

Contare (Avanzato)

In questo capitolo mostreremo come gli stati del robot possono essere usati per contare i numeri e anche effettuare semplici calcoli aritmetici.

Il disegno e l'implementazione dei progetti di questo capitolo non saranno presentati in dettaglio. Assumiamo che tu abbia abbastanza esperienza ora per svilupparli da te. Il codice sorgente dei programmi funzionanti è incluso nell'archivio, ma non consultarlo a meno che tu non abbia veramente difficoltà a risolvere un problema.

Questi progetti usano l'evento *clap* (battito di mani) per cambiare gli stati e il comportamento di base del cerchio di LED per mostrare lo stato:

> **Informazione importante**
> Lo stato corrente del robot è mostrato nel cerchi di LED della parte superiore del robot. La Figura 8.2(b) mostra il robot nello stato **(on, on, on, on)**.

Sentiti libero di modificare qualsiasi di questi comportamenti.

Pari e Dispari

> **Programma**
> Scegli uno degli elementi dello stato del robot. Sarà **spento** (bianco) se il numero di battiti di mani sarà pari, e **acceso** (arancione) se il numero di appalusi è dispari. Toccando il bottone centrale si ripristina a zero cioè pari. (poichè zero è un numero pari per definizione).

File di programma **count-to-two.aesl**

Queso metodo di conteggio dimostra il concetto di *aritmetica modulo 2*. Contiamo a partire da 0 a 1 e poi di nuovo a 0. Il termine *modulo* è simile al termine *resto*: se ci sono stati 7 battiti di mani, dividendo 7 per 2 da 3 e resto 1. Teniamo solo il resto 1.

In aritmetica modulo 2, 0 e 1 sono spesso chiamati pari e dispari, rispettivamente.

Un altro termine per lo stesso concetto è *aritmetica ciclica*. Invece di contare da 0 a 1 e poi da 1 a 2, noi *cicliamo* dall'inizio: 0, 1, 0, 1, ...

Questi concetti sono molto familiari perché sono usati negli orologi. Minuti e secondi vengono contati modulo 60 e le ore e sono contati modulo 12 o 24. Pertanto, il secondo dopo 59 non è 60; invece, si cicla e si inizia di nuovo il conteggio da 0. Allo stesso modo,

l'ora dopo le 23 non è 24, ma 0. Se sono le 23:00 e siamo d'accordo di incontrarci dopo 3 ore, l'orario fissato per la riunione è 26 modulo 24, che equivale alle 2:00 di mattina del giorno dopo.

Contare in unario

> Modifica il programma per contare modulo 4. Ci sono quattro possibili resti 0, 1, 2, 3. Scegli tre elementi dello stato, uno per rappresentare ciascuno dei valori 1, 2 e 3; il valore 0 verrà rappresentato impostando ogni elemento dello stato a **spento**.

Questo metodo di rappresentazione dei numeri è chiamato *rappresentazione unaria* poiché ogni elemento dello stato rappresenta un numero diverso. Spesso usiamo la rappresentazione unaria per tenere traccia del conteggio di alcuni oggetti; per esempio, 𝍸 | rappresenta 6.

File di programma **count-to-four.aesl**

Esercizio 9.1

Fino a che numero possiamo contare con Thymio usando la rappresentazione unaria?

Contare in binario

Noi abbiamo molta familiarità con la *rappresentazione in base* dei numeri, in particolare con la rappresentazione in base 10 (decimale). Il numero 256 nella rappresentazione in base 10 non rappresenta tre diversi oggetti (2, 5, 6). In realtà il 6 rappresenta il numero di 1 (unità), il 5 rappresenta il numero di $10\times1=10$ (decine), e il 2 rappresenta il numero di $10\times10\times1=100$ (centinaia). La somma di questi fattori dà il numero duecentocinquantasei. Utilizzando la rappresentazione in base 10 possiamo scrivere numeri molto grandi in una rappresentazione compatta. Inoltre, l'aritmetica sui grandi numeri è relativamente facile con i metodi che abbiamo imparato a scuola.

Noi usiamo la rappresentazione in base 10 perché abbiamo 10 dita quindi per noi è più facile imparare ad usare questa rappresentazione. I computer invece hanno due "dita"(**spento** e **acceso**) così si usa aritmetica in base 2 per fare i calcoli. L'aritmetica in base 2 sembra strana all'inizio: utilizziamo i simboli familiari 0 e 1 usati anche in base 10, ma le regole per il conteggio sono di ciclare a 2 anziché ciclare a 10:

$$0, 1, 10, 11, 100, 101, 110, 111, 1000, \ldots$$

Dato un numero in base 2, ad esempio 1101, si calcola il suo valore da destra a sinistra come nella rappresentazione base 10. La cifra più a destra rappresenta il numero di uno,

la cifra successiva rappresenta il numero $1\times2=2$, la terza cifra rappresenta il numero $1\times2\times2=4$, e la cifra più a sinistra rappresenta il numero $1\times2\times2\times2=8$. Pertanto, 1101 rappresenta 1 + 0 + 4 + 8, che è tredici, rappresentato in base 10 come 13.

Programma
Modifica il programma per il conteggio modulo 4 per usare la rappresentazione binaria.

File di programma **count-to-four-binary.aesl**

Abbiamo solo bisogno di due elementi dello stato per rappresentare i numeri 0–3 in base 2. Fai in modo che l'elemento in alto a destra rappresenti il numero di 1, **spento** (bianco) per zero e **acceso** (arancione) per uno, e che l'elemento superiore sinistro rappresenti il numero di 2. Ad esempio,  rappresenta il numero 1 e  rappresenta il numero 2. Se i due elementi sono entrambi di colore bianco, lo stato rappresenta 0, e se entrambi gli elementi sono di colore arancione, lo stato rappresenta il 3.

Per contare sono necessarie quattro transizioni $0 \to 1$, $1 \to 2$, $2 \to 3$, $3 \to 0$, quindi sono necessarie quattro coppie evento-azione, oltre ad una coppia per ripristinare il programma quando si tocca il pulsante centrale.

Trucco

I due elementi inferiori dello stato non vengono utilizzati, quindi sono lasciati in grigio e vengono ignorati dal programma.

Esercizio 9.2

Estendi il programma in modo che conti modulo 8. L'elemento dello stato in basso a sinistra rappresenterà il numero di 4.

Esercizio 9.3

Fino a che numero possiamo contare con Thymio utilizzando la rappresentazione binaria?

Aggiungere e sottrarre

Scrivere il programma per contare fino a 8 è piuttosto noioso perchè devi programmare 8 coppie evento-azione, una per ciascuna transizione da n a $n + 1$ (modulo 8).Certamente non è così che si conta, invece, normalmente utilizziamo metodi per eseguire le addizioni sommando le cifre in ciascun posto e facendo il riporto verso la cella successiva. Nella rappresentazione in base 10:

$$\begin{array}{r} 387 \\ +426 \\ \hline 813 \end{array}$$

e similmente nella rappresentazione in base 2:

$$\begin{array}{r} 0011 \\ +1011 \\ \hline 1110 \end{array}$$

Quindi aggiungendo 1 a 1, invece di 2, otteniamo 10. Lo zero è scritto nella stessa colonna e riportiamo un 1 nella successiva colonna a sinistra. L'esempio mostra la somma di 3 (=0011) e di 11 (=1011) per ottenere 14 (=1110).

Programma
Scrivi un programma che inizia con una rappresentazione di 0. Ciascun battito di mani aggiunge 1 al numero. L'addizione è modulo 16, quindi aggiungendo 1 a 15 il risultato sarà 0.

Linee guida:

- L'elemento in basso a destra verrà utilizzato per rappresentare il numero 8.
- Se l'elemento in alto a destra che rappresenta il numero 1 mostra 0 (bianco), semplicemente cambiarlo a 1 (arancione). Fate questo indipendentemente da ciò che gli altri elementi mostrano.
- Se l'elemento in alto a destra che rappresenta il numero di 1 è a 1 (arancione), modificarlo a 0 (bianco) e poi riportare 1. Ci saranno tre coppie evento-azione, a seconda della posizione del *prossimo* elemento che è a 0 (bianco).
- Se tutti gli elementi sono a 1 (arancione), è rappresentato il valore 15. Aggiungendo 1 a 15 modulo 16 si ottiene 0, rappresentato da tutti gli elementi che sono a 0 (bianchi).

File di programma **addition.aesl**

Esercizio 9.4

Modifica il programma in modo che parta dal valore 15 e sottragga uno per ciascun battito di mani, e quindi ciclicamante torni a 15.

Esercizio 9.5

Metti una sequenza di corti segmenti di nastro nero su una superfice chiara (o nastro bianco su una superfice scura). Scrivi un programma che faccia andare in avanti il Thymio e fermare quando viene riconosciuto il quarto nastro.

L'esercizio non è facile: le strisce devo essere sufficientemente ampie in modo che il robot le riconosca, ma non così ampie da far avvenire più di un evento per striscia. Dovrai anche fare esperimenti con la velocità del robot.

Capitolo 10

Ed ora?

Questo tutorial ha introdotto il robot Thymio e l'ambiente Aseba/VPL. L'ambiente VPL con la sua semplice programmazione visuale è stato pensato per i principianti. Per sviluppare programmi più avanzati per il robot dovrai imparare ad usare l'ambiente Aseba Studio (Figura 10.1).

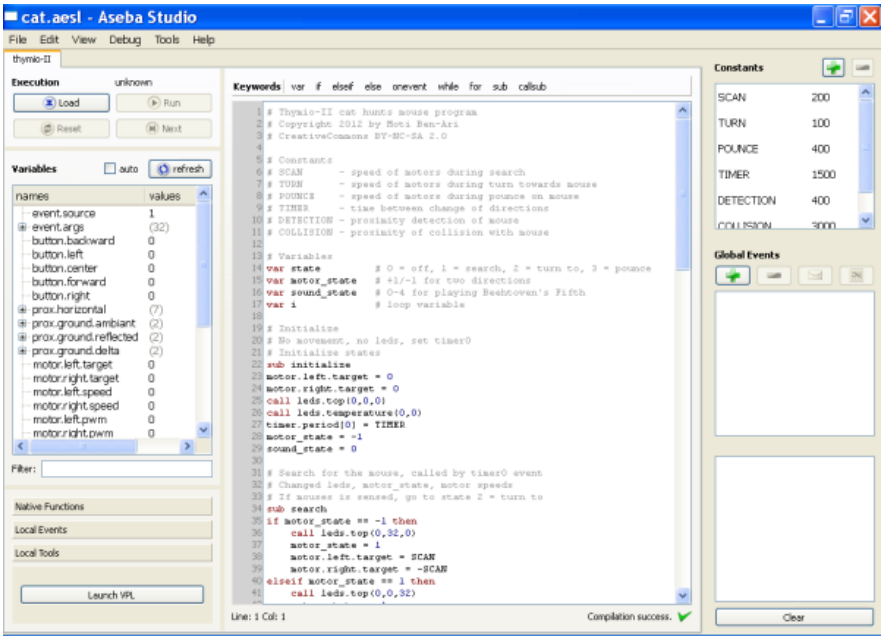


Figura 10.1: l'ambiente Aseba Studio

La Programmazione in Aseba Studio è sempre basata sui concetti di eventi ed azioni. poichè i programmi VPL sono tradotti in programmi testuali, tutto quello che hai imparato in questo tutorial è disponibile anche in Studio dove sono disponibili molte altre funzionalità di programmazione:

- Puoi controllare esattamente quando un evento causa un'azione, in dipendenza, ad esempio, dall'ammontare di luce riflessa misurata da un sensore del terreno o della distanza da un sensore orizzontale.
- Puoi specificare che una singola azione è composta da diverse operazioni differenti: controllare i motori, cambiare lo stato, definire soglie, accendere e spegnere luci, ecc...

- Hai la flessibilità di un linguaggio di programmazione completo con variabili, espressioni, direttive di controllo.

Aseba Studio vi dà accesso a funzionalità del robot Thymio che non sono disponibili in VPL:

- Puoi controllare tutte le luci come ad esempio il cerchio di luci che contorna i bottoni.
- Hai più flessibilità nel sintetizzare suoni.
- C'è un sensore di temperatura.
- Invece di rilevare semplicemente gli urti, gli accellerometri possono sentire la forza di gravità e ogni variazione di velocità nelle tre dimensioni.
- Si può anche usare un telecomando per impartire comandi al robot.

Quando stai lavorando con Aseba Studio puoi aprire il VPL cliccando sul bottone **Lancia VPL** nel tab *Strumenti* nella parte a sinistra in basso della finestra. Puoi importare programmi VPL direttamente in Aseba Studio semplicemente aprendo il file.

Per usare Aseba Studio, parti dalla pagina *Programmare Thymio II* al link:
`https://aseba.wikidot.com/it:thymioprogram`
e segui il link *Ambiente di programmazione testuale*.

puoi trovare tanti progetti interessanti con questo link:
`https://aseba.wikidot.com/it:thymioexamples`.

Divertiti e impara tante cose!
Grazie per aver letto questo tutorial!