Hands on NXT

Andrea Albertini

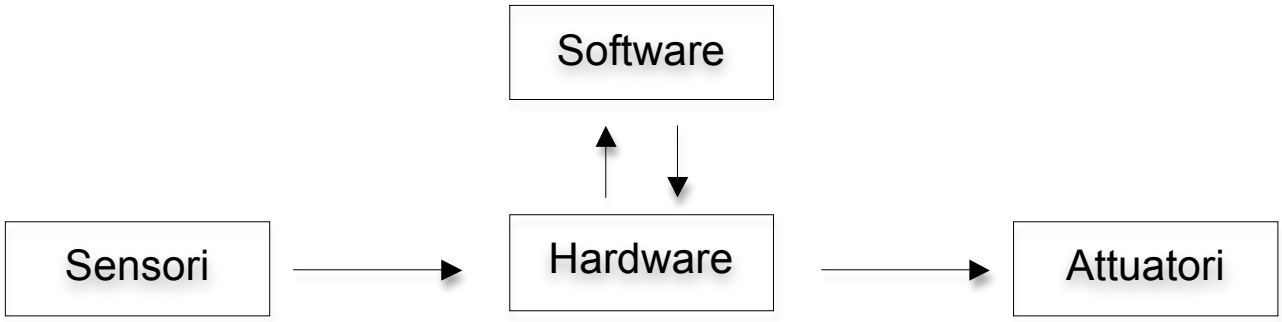
Un robot per apprendimenti interdisciplinari
Introduzione a Lego Mindstorms NXT

http://www.fll-si.ch
Ver. 1.7

Sommario

Introduzione

Un robot è un agente elettro-meccanico che è in grado di interagire con l'ambiente circostante e di svolgere dei compiti specifici in maniera autonoma. Generalmente è governato da un programma o da un circuito elettronico. L'interazione dei robot con l'ambiente avviene tramite sensori e attuatori.



Hardware

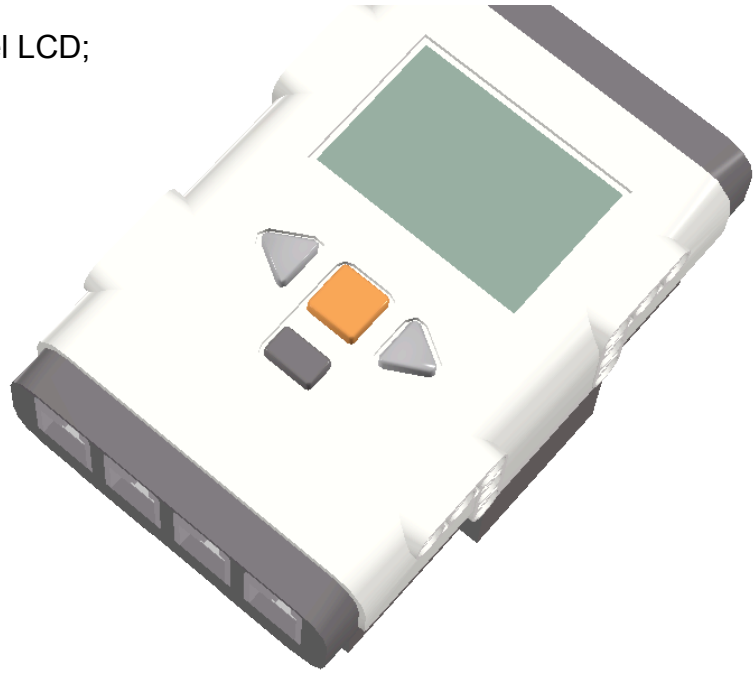
Con il termine hardware si intendono le componenti fisiche del robot, comprese tutte le parti meccaniche come il telaio, i motori e altri attuatori, i sensori, gli apparati di alimentazione e le schede di controllo con il processore e la memoria. Nel caso dei robot Mindstorm, l'hardware è costituito dal blocchetto NXT e da tutti i pezzi contenuti nella scatola.

NXT

Il componente principale del kit Mindstorm è un piccolo computer a forma di cubetto Lego chiamato NXT. L'NXT è munito di un display grafico e di quattro tasti che permettono di interagire con i menu del firmware e dispone di quattro porte di ingresso e due di uscita cui possono essere connessi sensori e attuatori. L'NXT può anche comunicare con altri dispositivi (ad esempio PC, cellulari e palmari o altri NXT) grazie ad una interfaccia Bluetooth e una porta USB.

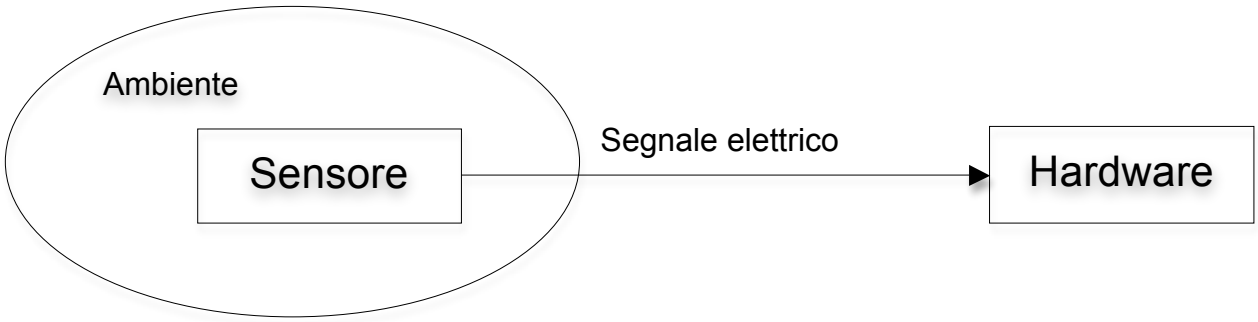
Caratteristiche tecniche dell'NXT:

- Processore principale ARM 32-bit 48 MHz (256 KB flash, 64 KB RAM);
- microcontrollore 8-bit ATmega48 @ 4 MHz (4 KB flash memory, 512 Bytes RAM);
- interfaccia Bluetooth;
- display grafico 100×64 pixel LCD;
- altoparlante;
- 4 porte di ingresso;
- 3 porte di uscita;
- interfaccia USB;



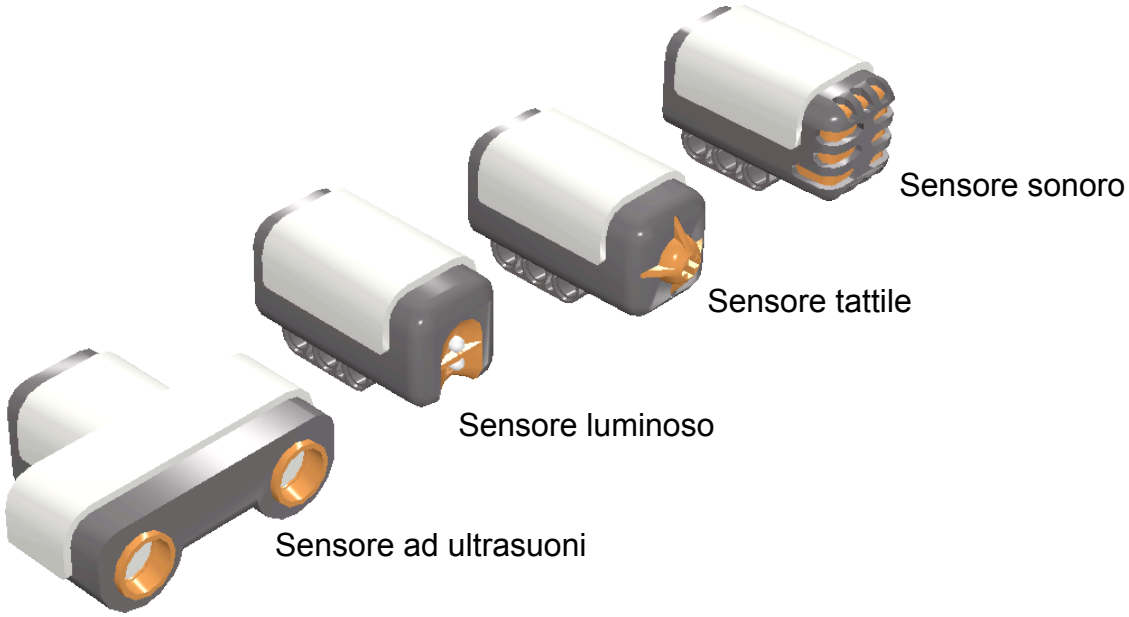
Sensori

Per acquisire dei dati provenienti dall'ambiente, i robot possiedono dei dispositivi che permettono di convertire le grandezze provenienti dal mondo esterno in segnali elettrici comprensibili dalla macchina. Questi elementi, che occupano il primo posto nella catena di acquisizione, sono i sensori.



Nel kit Mindstorm si trovano diversi tipi di sensori:

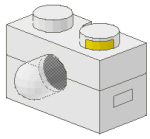
- 1 sensore ad ultrasuoni per misurare movimento e distanza;
- 1 sensore luminoso per misurare intensità luminosa e, solo nel NXT 2.0, colore;
- 2 sensori tattili;
- 1 sensore sonoro per misurare intensità e tonalità sonora.

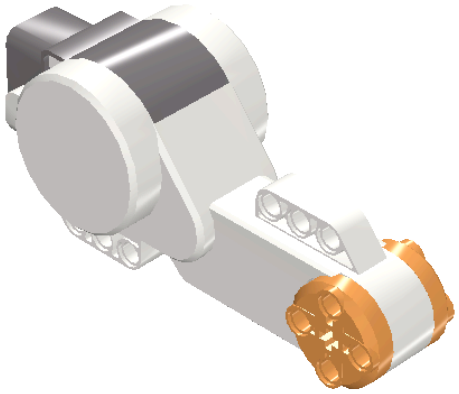


Attuatori

Per modificare la realtà circostante i robot utilizzano degli apparati detti attuatori. Gli attuatori ricevono dei comandi trasmessi dal robot in forma di segnali elettrici e li trasmutano in azioni fisiche.
Il kit NXT comprende:

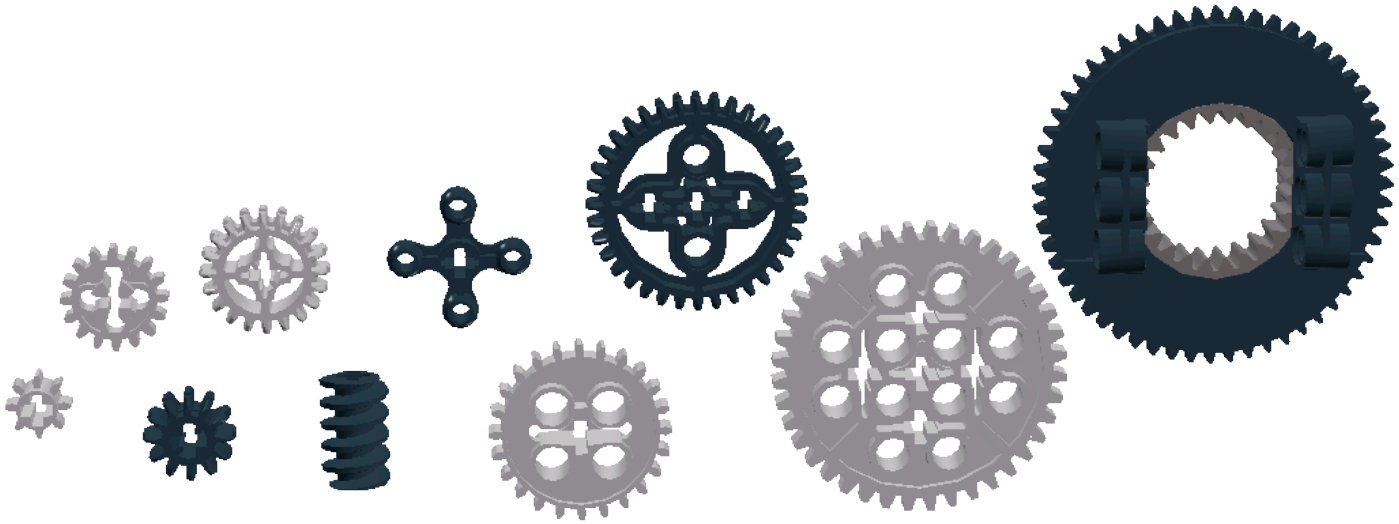
- 3 motori muniti di sensore di rotazione;
- 3 lampadine (solo nel kit EDU).





Pezzi tecnici

Il kit NXT contiene numerosi pezzi meccanici come ingranaggi, pulegge, assi, elementi costruttivi.



Software

Le azioni del robot sono governate da un programma. Cambiando la programmazione del robot se ne può cambiare il comportamento. In questo modo la stessa macchina può svolgere compiti differenti semplicemente modificandone la programmazione.

I robot Mindstorm possono essere programmati in un linguaggio grafico semplice e intuitivo derivato da LabView e chiamato NXT-G. Oltre a NXT-G esistono altri prodotti che permettono di programmare l'NXT in svariati altri linguaggi di programmazione; ecco qualche esempio:

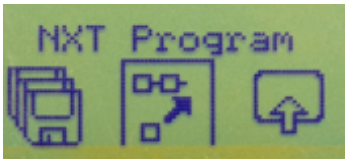
- Ada,
- Next Byte Codes (NBC),
- Not eXactly C (NXC),
- Robot C,
- NXTGCC,
- leJOS NXJ (Java),
- MATLAB e Simulink.

In questo corso viene trattata esclusivamente la programmazione con NXT-G.

Lego Firmware

Il *Lego Firmware* installato costituisce il sistema operativo e l'interfaccia utente del NXT. L'utente interagisce con il firmware tramite il menu che compare sullo schermo dell'NXT al momento dell'accensione:

My Files:
Permette di gestire file e cartelle presenti nella memoria dell'NXT e di eseguire i programmi.



NXT Program:
Per programmare semplicemente senza collegare l'NXT al computer.

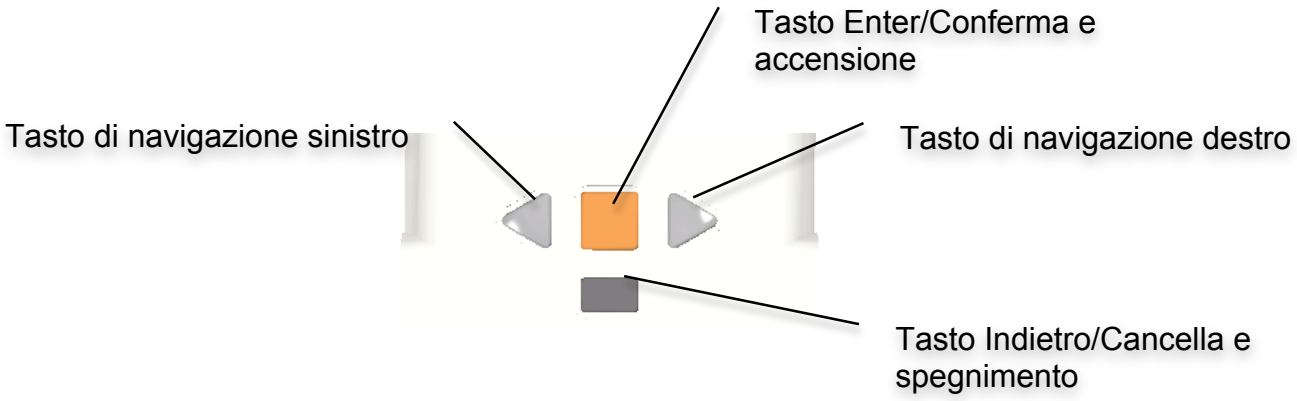
View:
Per vedere il valore letto dai sensori collegati all'NXT.

Bluetooth:
Per attivare e disattivare l'interfaccia Bluetooth e modificarne le impostazioni.

Settings:
Permette di modificare il volume dei suoni e il timer dello spegnimento automatico. Inoltre consente di visualizzare la versione del firmware e di cancellare file dalla memoria.

Try Me:
Contiene una serie di semplici programmi che permettono di sperimentare il funzionamento di tutti i sensori inclusi nel kit.

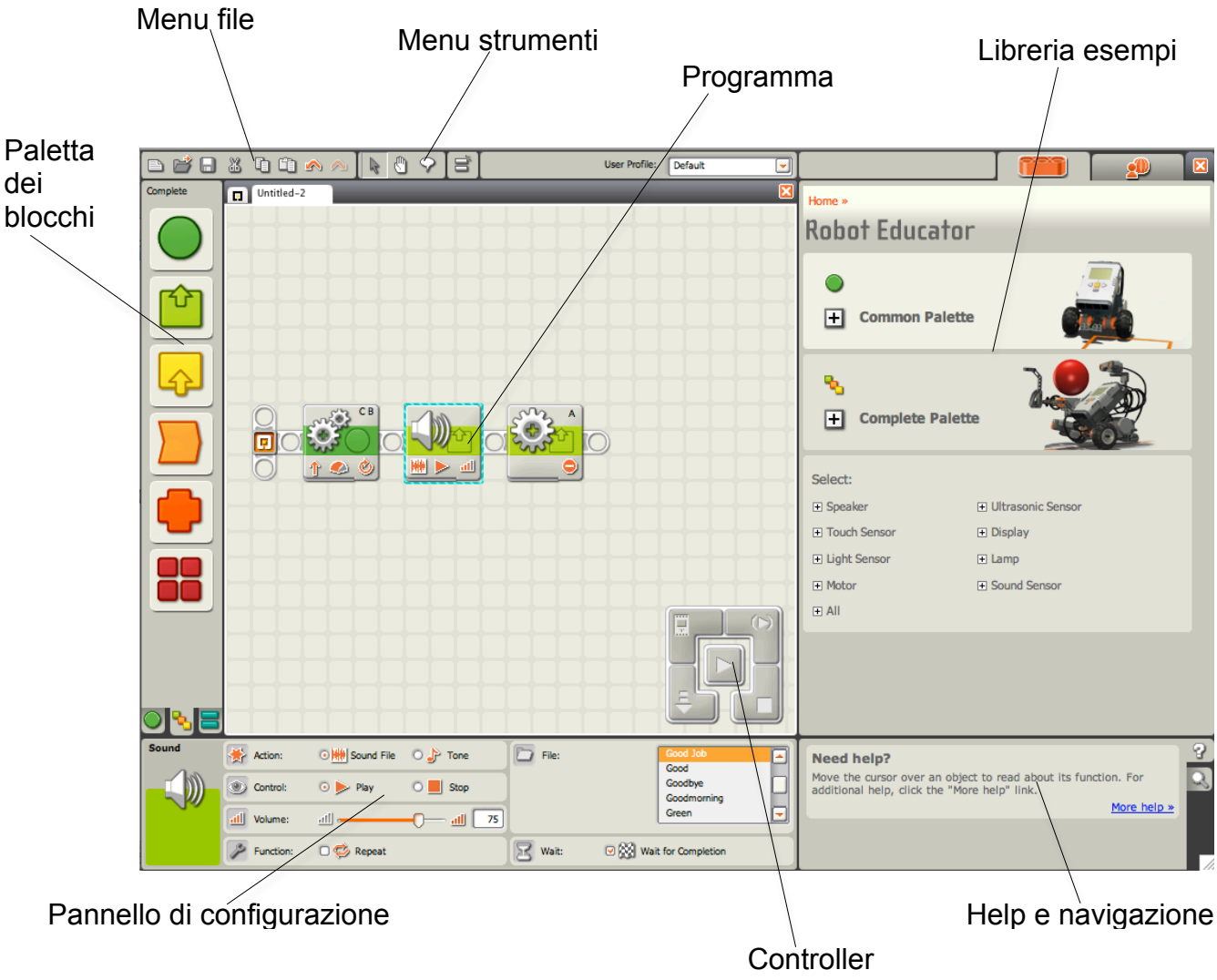
La navigazione all'interno dei menu del firmware avviene tramite i tasti dell'NXT:



Ambiente di sviluppo

La programmazione con NXT-G avviene in modo grafico all'interno di un ambiente di sviluppo che permette anche di svolgere tutte le attività di gestione dell'NXT.

Interfaccia



Menu file

Questo menu permette di gestire i file e di svolgere le funzionalità di base:


Nuovo file:
serve a creare un nuovo programma vuoto.


Apri:
serve ad aprire programmi già esistenti.


Salva:
salva il programma corrente.


Taglia:
taglia i blocchi selezionati.


Copia:
copia i blocchi selezionati.


Incolla:
incolla i blocchi copiati o tagliati.


Undo:
annulla l'ultima modifica.


Redo:
ripete l'ultima modifica.

Menu strumenti

Il menu strumenti contiene i tool seguenti:


Strumento pointer:
serve a selezionare e manipolare i blocchi.


Strumento pan:
serve a muoversi all'interno del programma (vedi anche Navigation).


Strumento commento:
serve a scrivere dei commenti testuali nel programma.


Strumento MyBlock:
Serve a creare dei blocchi personalizzati.

Palette dei blocchi

Le palette dei blocchi contengono tutti i blocchi disponibili per la programmazione raggruppati in tre sezioni:


Common palette:
contiene i blocchi di uso più comune.
Complete palette:
contiene tutti i blocchi disponibili.
Custom palette:
contiene tutti i blocchi personalizzati scaricati da internet o creati dall'utente.

Pannello di configurazione

Il pannello di configurazione contiene i parametri dei blocchi come ad esempio la potenza dei motori o le soglie dei sensori e permette di modificarli. I dettagli di configurazione dei blocchi saranno trattati in seguito.

Robot educator

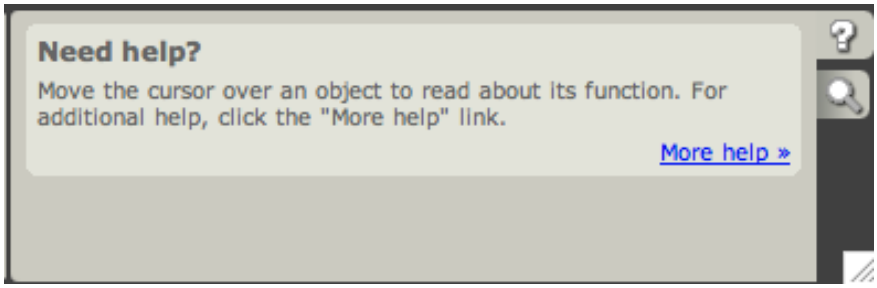
Questo pannello contiene la libreria di esempi di programmazione in NXT-G. La libreria è suddivisa in due sezioni contenenti una quarantina di esempi che spiegano il funzionamento di ogni blocco presente nella common e nella complete palette. Ogni esempio é strutturato in tre parti:

- **Challenge brief:** una breve spiegazione dell'esempio.
- **Building guide:** spiega come costruire il robot dell'esempio.
- **Programming guide:** spiega come programmare il robot d'esempio.

Inoltre è possibile filtrare gli esempi in base al tipo di blocco interessato.

Help e navigazione

Nell'angolo inferiore destro dell'ambiente di sviluppo si trova il riquadro contenente help e navigazione:



Help:
Durante la programmazione in questo riquadro viene visualizzato il riassunto della guida relativa al blocco selezionato e un link alla guida completa.

Navigazione:
Rappresenta una panoramica del programma corrente e permette di muoversi da una zona all'altra del codice. Questa funzionalità risulta particolarmente utile quando si lavora su programmi troppo grandi per essere visualizzati nell'area di programmazione.

Il controller permette di caricare i programmi (o parti di essi) su un NXT connesso al computer (con USB o Bluetooth) e di eseguirli. Tramite il controller è anche possibile modificare le impostazioni dell'NXT.

Download and run:
Compila il programma corrente, lo carica sull'NXT e lo esegue.

Download:
Compila il programma corrente e lo carica sull'NXT.

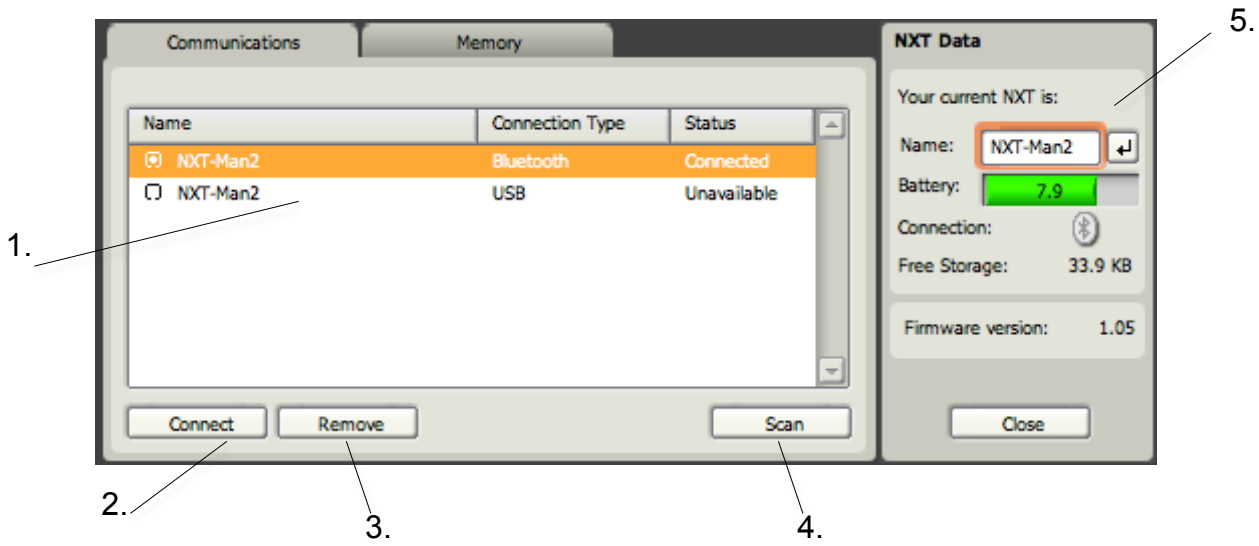
Stop:
Arresta l'esecuzione del programma attivo.

Download and run selected:
Compila la parte di programma selezionata , la carica sull'NXT e la esegue.

NXT window:
Permette di gestire le impostazioni di comunicazione e la memoria dell'NXT.

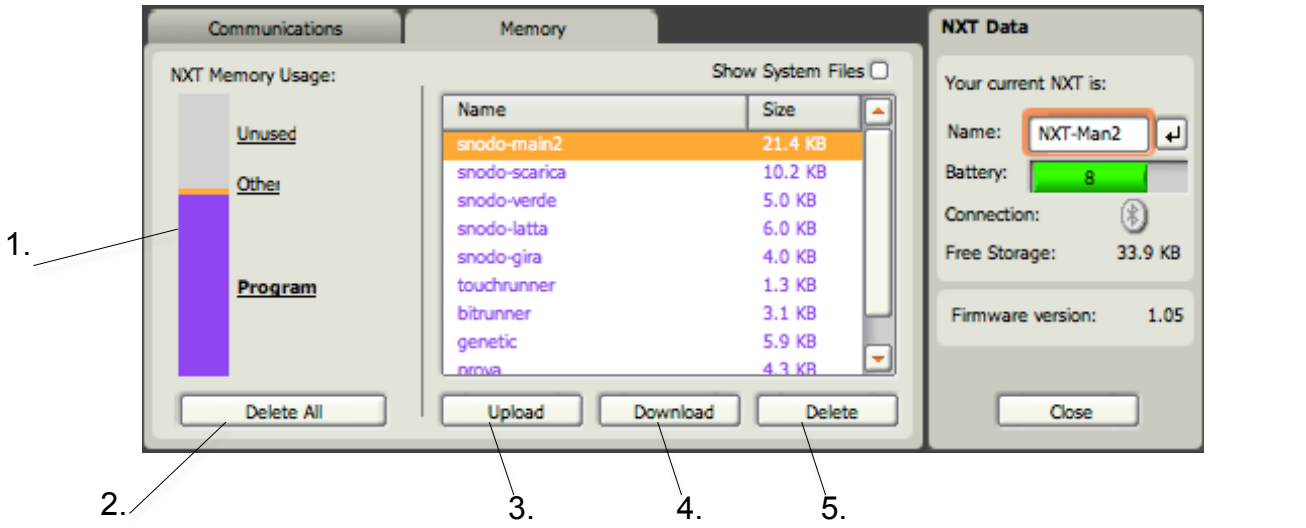
NXT window

Questa finestra è divisa in due pannelli. Il primo è dedicato alla comunicazione con l'NXT:



1. **Lista NXT disponibili:** in questo elenco vengono elencati gli tutti NXT disponibili per la connessione tramite USB e Bluetooth.
2. **Tasto Connect:** serve per connettere l'NXT selezionato nella lista.
3. **Remove:** rimuove dalla lista l'NXT selezionato.
4. **Scan:** effettua una scansione per trovare nuovi NXT disponibili per la connessione.
5. **Pannello dati NXT:** contiene le informazioni concernenti l'NXT connesso.

La seconda tabella della NXT window contiene le informazioni relative all'utilizzo della memoria dell'NXT connesso:



1. **Occupazione memoria:** rappresenta la quantità di memoria occupata dell'NXT.
2. **Delete All:** cancella tutti file presenti sul'NXT.
3. **Upload:** permette di caricare un file nell'NXT.
4. **Download:** scarica il file selezionato dall'NXT al computer.
5. **Delete:** cancella dalla memoria dell'NXT il file selezionato.

Programmazione grafica NXT-G

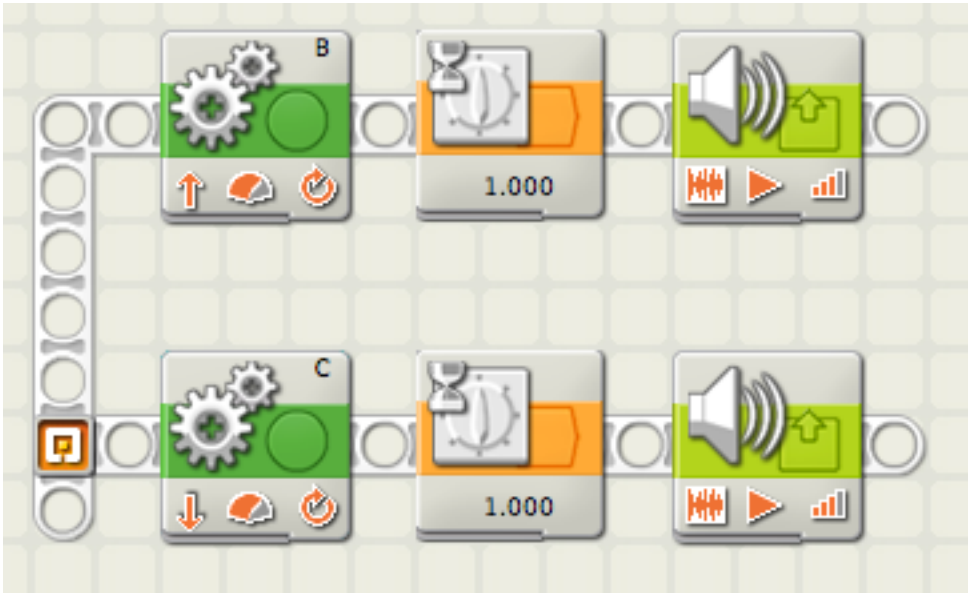
Concetti di base

Start-point e sequenze

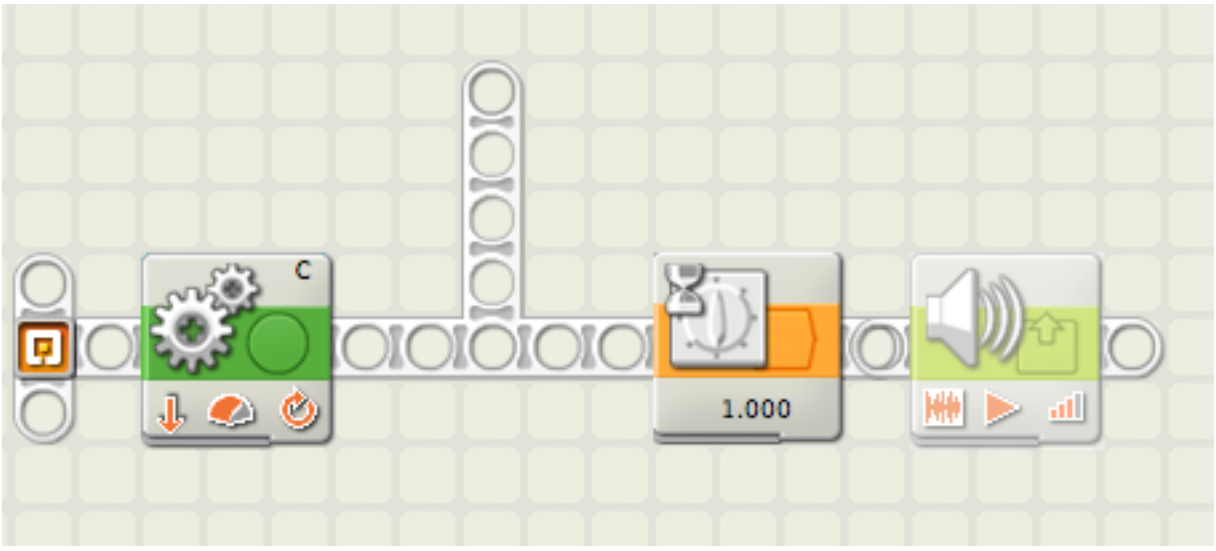
Il punto di inizio di ogni programma NXT-G è costituito dallo start-point.

Tutti i blocchi connessi allo start-point costituiscono una sequenza di istruzioni e verranno scaricati sull'NXT ed eseguiti quando verrà premuto il tasto Run del Controller.

È possibile creare più sequenze collegate allo start-point che verranno eseguite in parallelo:

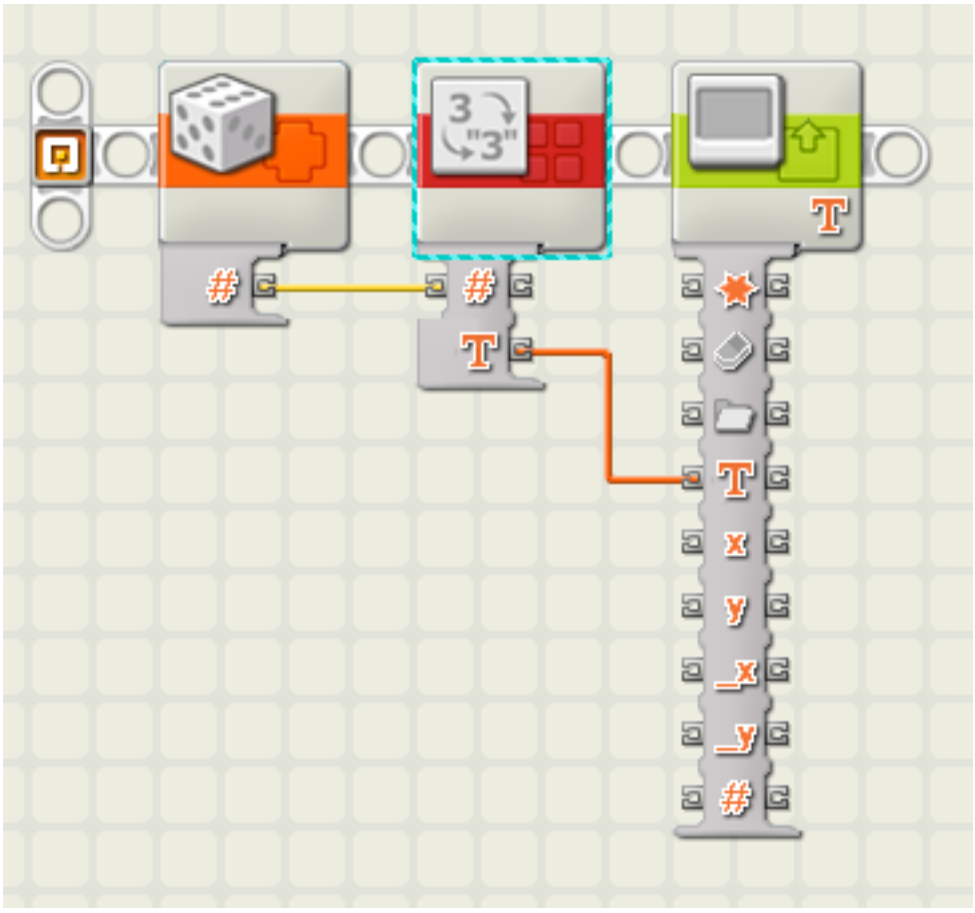


Si possono anche creare sequenze parallele a partire da qualsiasi punto del flusso:



Data wire

I *data wire* (fili per i dati) servono a trasferire informazioni da un blocco all'altro. Sebbene molti blocchi non necessitino di *data wire* per funzionare, in alcuni casi possono essere indispensabili per realizzare alcune funzioni. Ad esempio il numero casuale generato dal blocco *Random* può essere trasmesso ad altri blocchi tramite un data *wire*.



Blocchi NXT-G

La *complete palette* contiene tutti i blocchi disponibili per la programmazione; quelli della *common palette* e altri blocchi che forniscono funzionalità supplementari. I blocchi della *complete palette* sono organizzati nel seguente modo:

Common:
La common palette raggruppa i blocchi di uso più frequente che sono sufficienti per la realizzazione di programmi semplici.

Action:
Contiene tutti i blocchi dedicati alla gestione delle uscite (motori, lampadine, suono, display e invio di messaggi bluetooth).

Sensor:
Contiene tutti i blocchi dedicati alla gestione degli ingressi (sensori, bottoni, timer e ricezioni di messaggi bluetooth).

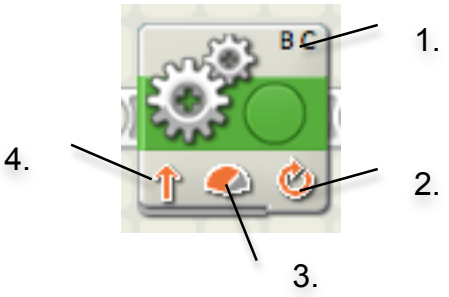
Flow:
Contiene i blocchi utilizzati per la gestione di flusso dei programmi (wait, loop, switch, stop).

Data:
Contiene blocchi che implementano funzioni di elaborazione di dati e variabili (operazioni logiche, operazioni aritmetiche, confronto di valori, range, generazione di numeri casuali, accesso alle variabili).

Advanced:
Contiene blocchi dedicati a funzionalità avanzate (concatenazione di stringhe, conversione di stringhe in numeri, inibizione del timer di spegnimento, accesso a file, calibrazione di sensori e reset di motori).

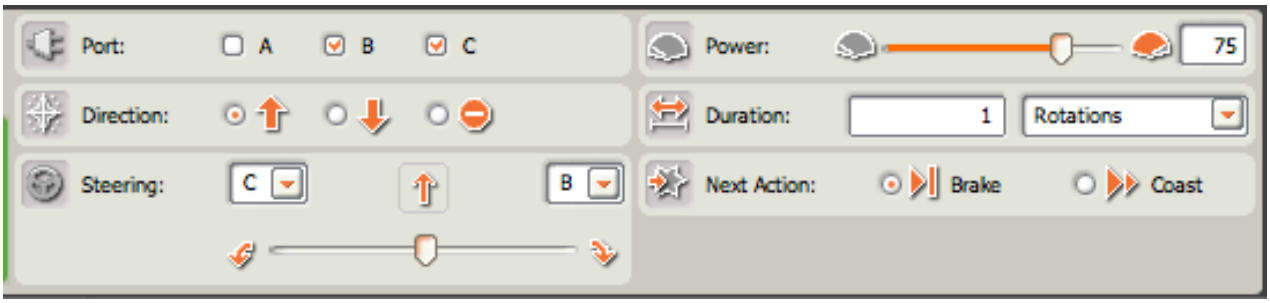
Move

Il blocco Move permette di far muovere il robot in avanti, indietro o tracciando una curva.



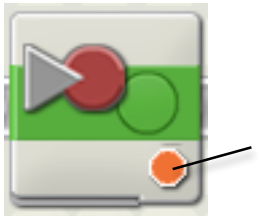
1. Porte controllate.
2. Rotazione.
3. Potenza.
4. Direzione.

Il pannello di configurazione del blocco Move:



- **Port:** le porte controllate dal blocchetto.
- **Direction:** la direzione del movimento del robot.
- **Steering:** la curvatura della traiettoria del robot.
- **Power:** la potenza dei motori.
- **Duration:** la durata del movimento, comprende quattro impostazioni possibili:
  - **Unlimited:** durata del movimento indefinita, i motori gireranno fino a quando non verranno fermati.
  - **Degrees:** durata espressa in gradi, i motori effettuano una rotazione pari ai gradi specificati e poi si fermano.
  - **Rotations:** durata espressa in rotazioni, i motori girano effettuano le rotazioni specificate e poi si fermano.
  - **Seconds:** durata espressa in secondi, i motori girano per il tempo specificato e poi si fermano.
- **Next Action:** l'azione da compiere alla fine della rotazione:
  - **Brake:** alla fine della rotazione i motori si bloccano.
  - **Coast:** alla fine della rotazione i motori rallentano gradualmente fino a fermarsi.

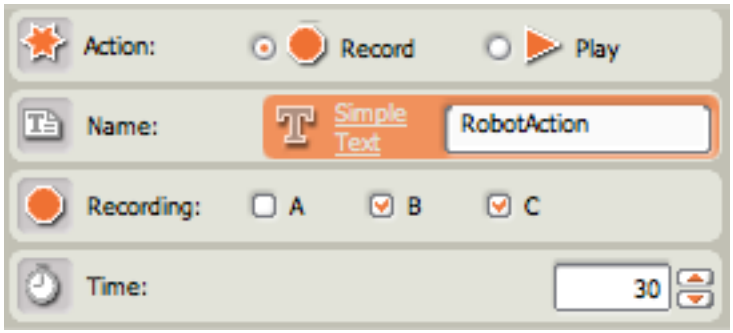
Questo blocco permette di registrare e di riprodurre le azioni che vengono compiute sul robot. In fase di registrazione memorizza le rotazioni impresse sui motori e in fase di riproduzione le riproduce azionando i motori.



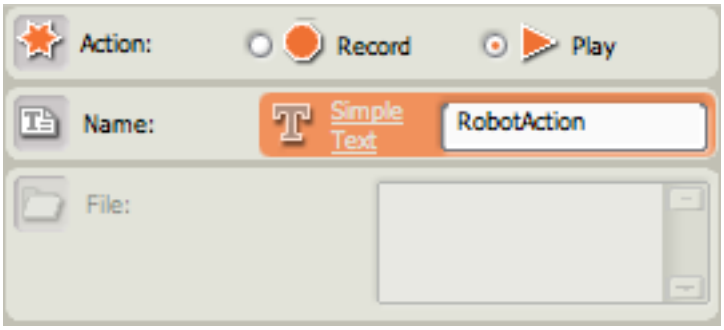
1. Funzione attiva (Play o Record).

Il pannello di configurazione del blocco Record/Play:

- **Action:** specifica se attivare la funzione *Record* o *Play*.
- **Name:** il nome dell'azione da memorizzare.
- **Recording:** le porte di uscita da registrare.
- **Time:** la durata della registrazione.

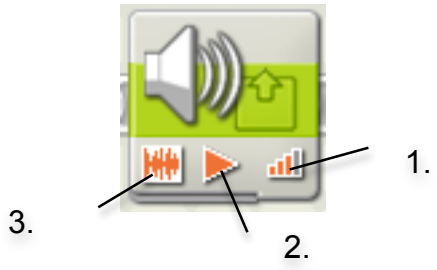


Se viene selezionata la funzione Play il pannello di configurazione si mostra nel modo seguente:



- **Name:** il nome dell'azione da riprodurre.
- **File:** lista delle registrazioni già presenti sull'NXT.

Il blocco sound serve a far produrre dei suoni all'NXT. Può riprodurre file audio o note musicali.



1. Volume.
2. Funzione (Play o Stop).
3. Modo (Sound file o Tone).

Il pannello di configurazione del blocco Sound in modalità *Sound File* si presenta nel modo seguente:

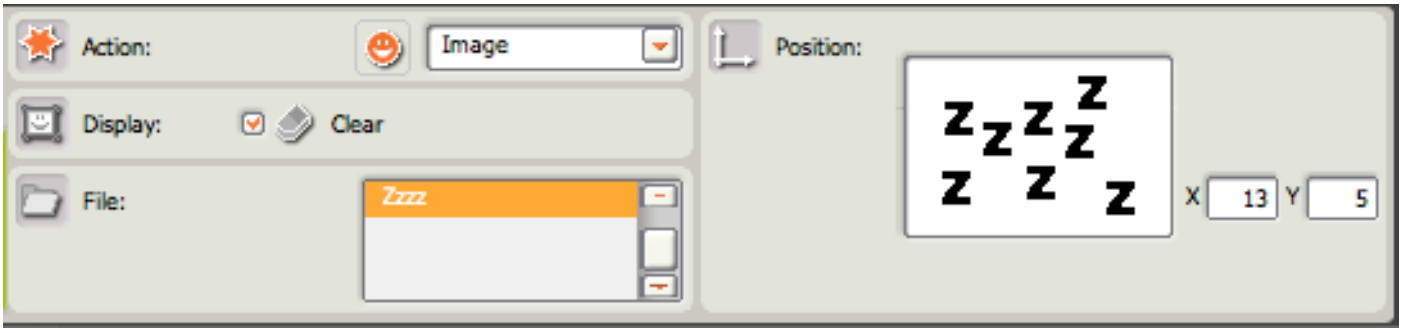
- **Action:** specifica se attivare la funzione *Sound File* o *Tone*.
- **Control:** *Play* fa partire la riproduzione di un suono e *Stop* la fa fermare.
- **Volume:** il volume di riproduzione del suono.
- **Function:** se l'opzione *Repeat* è attivata il suono viene ripetuto a ciclo continuo.
- **File:** il file sonoro da riprodurre.
- **Wait:** se l'opzione *Wait for complention* è selezionata le istruzioni successive al blocco Sound verranno eseguite solo alla fine della riproduzione del suono.

In modalità *Tone* il pannello di configurazione permette di selezionare la tonalità e la durata della nota da suonare:

Il blocco display serve a rappresentare testo, immagini o forme geometriche sul display dell'NXT.

1. Modo (Image, Text, Drawing, Reset).

In modalità *Image* il blocco Display permette di disegnare una immagine sullo schermo dell'NXT. Il pannello di configurazione:



- **Action:** specifica se attivare la funzione *Image*, *Text*, *Drawing* o *Reset*.
- **Display:** *Clear* provoca la cancellazione dello schermo.
- **File:** l'immagine da visualizzare.
- **Position:** le coordinate dell'immagine e un anteprima dell'immagine.

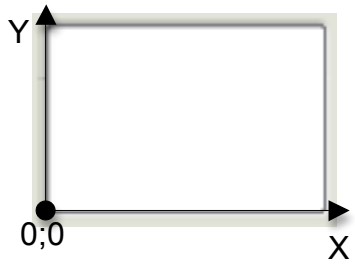
In modalità *Text* lo schermo dell'NXT è suddiviso in 8 righe di testo:

- **Action:** specifica se attivare la funzione *Image*, *Text*, *Drawing* o *Reset*.
- **Display:** *Clear* provoca la cancellazione dello schermo.
- **Text:** il testo da visualizzare.
- **Position:** la posizione del testo e un'anteprima. La posizione può essere espressa in coordinate o tramite il numero di riga.

La modalità *Drawing* disegna punti, righe e cerchi sullo schermo:

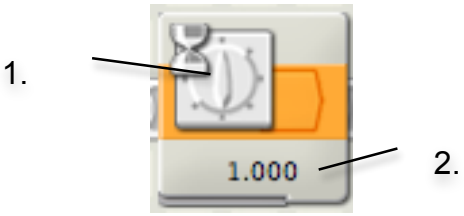
- **Action:** specifica se attivare la funzione *Image*, *Text*, *Drawing* o *Reset*.
- **Display:** *Clear* provoca la cancellazione dello schermo.
- **Type:** la forma da disegnare:
  - **Point:** disegna un punto (un pixel).
  - **Line:** disegna una linea.
  - **Circle:** disegna un cerchio.
- **Position:** le coordinate e gli eventuali parametri della forma:
  - **Point:** le coordinate del punto.
  - **Line:** le coordinate del punto di partenza e del punto di arrivo della linea.
  - **Circle:** le coordinate del centro e il raggio del cerchio.

Le coordinate di immagini, testo e forme sono espresse in pixel e sono riferite al sistema di coordinate del display:



La modalità *Reset* del blocco *Display* riporta lo schermo dell'NXT nello stato di default (icona Mindstorm).

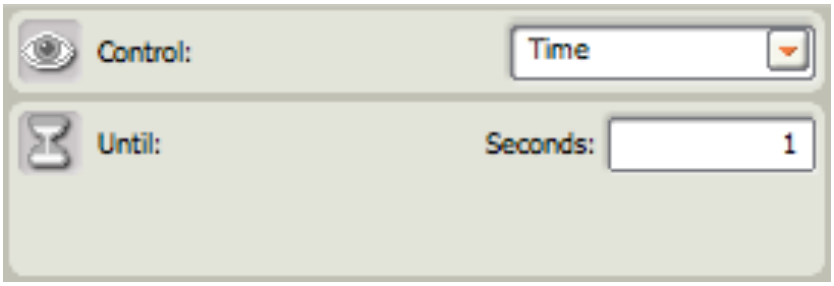
Il blocco *Wait* sospende l'esecuzione del programma fino a quando si verifica una condizione determinata. Questo blocco può essere utilizzato in due modi: il modo *Time* e il modo *Sensor*. Nella modalità *Time* l'esecuzione del programma viene sospesa per un tempo specificato:



1. Modalità *Time.*
2. Tempo di attesa espresso in secondi.

In modalità *Time* il pannello di configurazione permette solo di specificare il tempo di attesa:

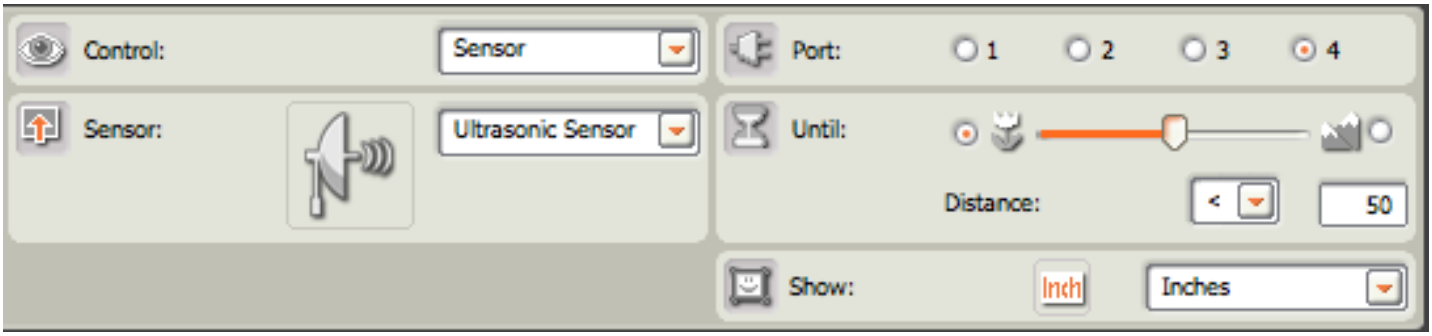
- **Control:** specifica se attivare la modalità *Time*, *Sensor*.
- **Until:** il tempo di attesa espresso in secondi.



In modalità *Sensor* il blocco *Wait* fa si che il robot aspetti l'avverarsi di una determinata condizione nell'ambiente. La sospensione del programma dura fino a quando un sensore non assume un valore specificato.

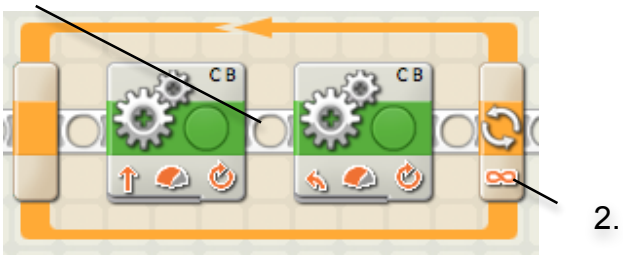
1. Tipo di sensore.
2. Porta di ingresso.
3. Condizione.

Il pannello di configurazione permette di definire quale sensore monitorare e quale condizione determina la fine dell'attesa:



- **Control:** specifica se attivare la modalità *Time* o *Sensor*.
- **Sensor:** il tipo di sensore da utilizzare:
  - **Light Sensor:** aspetta che il sensore di luce raggiunga un valore espresso in %.
  - **NXT Buttons:** aspetta che uno dei tasti dell'NXT venga premuto, rilasciato o cliccato.
  - **Receive Message:** aspetta che un messaggio tramite l'interfaccia Bluetooth.
  - **Rotation Sensor:** aspetta che un motore compia una rotazione.
  - **Sound Sensor:** aspetta che il sensore sonoro presenti un determinato valore.
  - **Timer:** attende che uno dei tre timer del NXT raggiunga un determinato valore.
  - **Touch Sensor:** aspetta che un sensore tattile venga premuto, rilasciato o cliccato.
  - **Ultrasonic Sensor:** attende che il sensore ad ultrasuoni misuri una determinata distanza.
  - **Light* Sensor:** Utilizza un sensore di luce del kit RCX (il nonno dell'NXT). Aspetta che un sensore di luce raggiunga un valore espresso in %.
  - **Rotation* Sensor**: Utilizza un sensore di rotazione del kit RCX. Attende che il sensore compia la rotazione specificata.
  - **Temperature* Sensor**: Utilizza un sensore di temperatura RCX. Attende il raggiungimento di un temperatura determinata.
  - **Touch* Sensor**: aspetta che un sensore tattile RCX venga premuto, rilasciato o cliccato.
- **Port:** la porta di ingresso.
- **Until:** la condizione di fine dell'attesa.

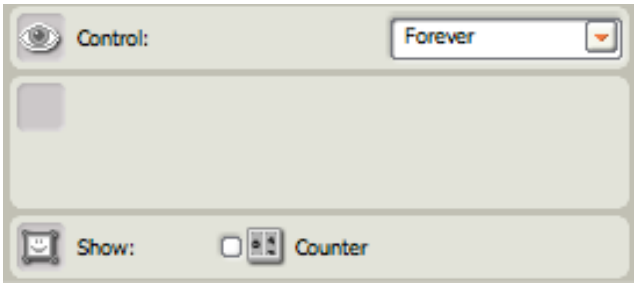
Il blocco *Loop* permette di realizzare la ripetizione di una sequenza di istruzioni all'infinito o fino all'avverarsi di una condizione determinata dal programmatore. Nel secondo caso la durata del ciclo può essere dettata da un tempo, un numero di ripetizioni, un segnale logico o lo stato di un sensore.



1. Sequenza di istruzioni da ripetere.
2. Condizione.

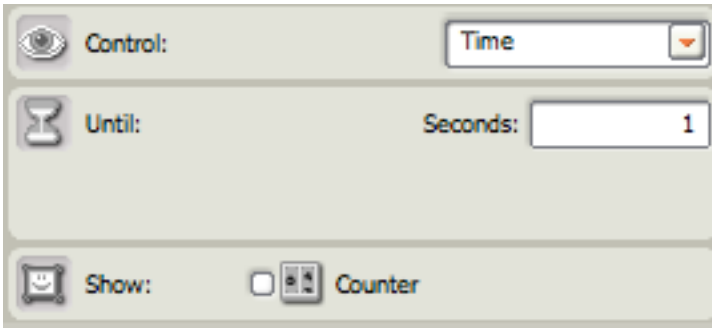
Il pannello di controllo di un blocco *Loop* cambia in funzione del tipo di condizione selezionata dal programmatore tra *Forever*, *Sensor*, *Time*, *Count* e *Logic*.

La modalità *Forever*, caratterizzata dal simbolo matematico dell'infinito, produce una ripetizione infinita del codice inserito nel ciclo. In questo caso le istruzioni poste dopo il *Loop* non verranno mai eseguite e l'iterazione del codice continuerà fino allo spegnimento del robot, all'interruzione manuale dell'esecuzione del programma o allo scoccare del timer di spegnimento automatico:

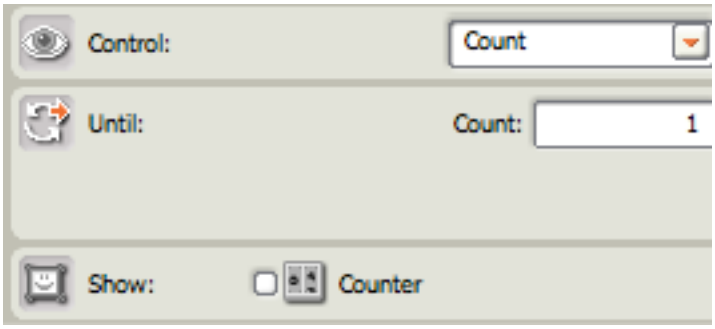


Nella modalità *Sensor* l'iterazione del codice avviene fino a quando non si presenta la condizione imposta dal programmatore. Quando ciò avviene, il ciclo termina e le istruzioni seguenti il *Loop* vengono eseguite. In questa modalità, il pannello di controllo del *Loop* è molto simile a quello descritto per il blocco *Wait*; permette infatti al programmatore di definire quale sensore monitorare e quali valori di soglia applicare alla condizione di permanenza nel ciclo:

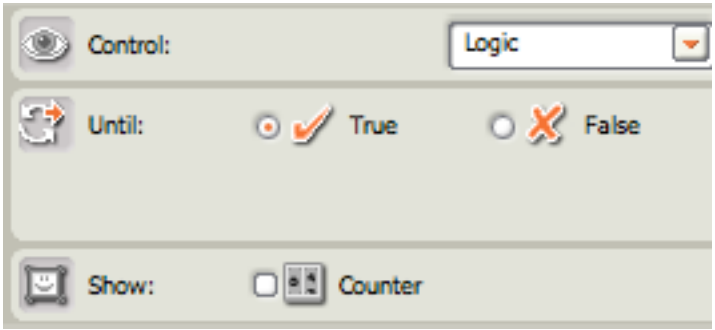
La modalità *Time*, come nel caso del blocco *Wait*, prevede la ripetizione di una sequenza di istruzioni fino allo scadere di un tempo determinato dal programmatore.



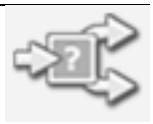
Nella modalità *Count* la ripetizione avviene un numero di volte determinato dal programmatore.



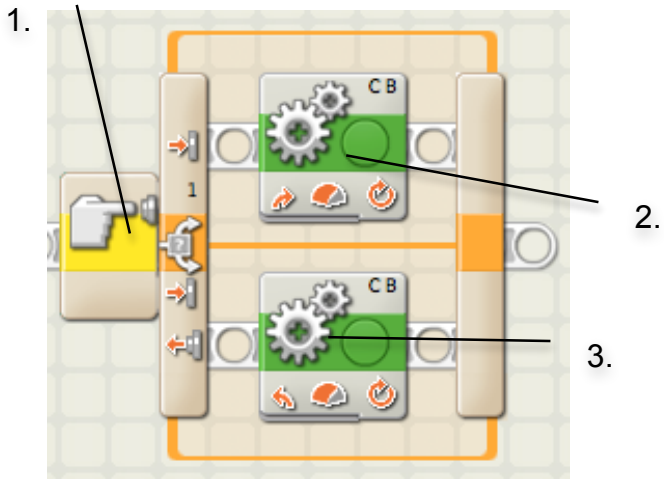
La modalità *Logic* permette la ripetizione del codice fino all'avverarsi di una condizione logica (vero, falso) fornita tramite un *data wire* connesso all'ingresso dati del ciclo.



In tutte le modalità di funzionamento di questo blocco, il numero di cicli effettuato dal programma è accessibile tramite il *data wire Loop Count* che appare attivando l'opzione *Show counter*.



Il blocco *Switch* permette di realizzare una struttura di controllo fondamentale in programmazione: la selezione; grazie ad esso si può esprimere il concetto di scelta all'interno dei programmi. In questo blocco, il programmatore introduce due sequenze di istruzioni. In fase di esecuzione del programma, il robot deciderà quale delle due sequenze eseguire, in funzione della condizione imposta dal programmatore.



1. Condizione della selezione.
2. Codice da eseguire se la condizione è vera.
3. Codice da eseguire se la condizione è falsa.

Il pannello di controllo di un blocco *Switch* prevede due tipi di condizione: *Sensor* e *Value*; nella modalità *Sensor* permette di definire quale tipo di sensore monitorare e su quale porta di ingresso, oltre al valore di soglia. In questo tipo di utilizzo il pannello di controllo è analogo a quello dei blocchi *Wait* e *Loop*:

Nella modalità *Value* la condizione della selezione è determinata da un *data wire* collegato al blocco *Switch* che fornisce al blocco il criterio di scelta. Il dato fornito dal data wire può essere di tipo logico, di testo o numerico.

Blocchi Action

I blocchi del gruppo *Action* permettono di controllare i dispositivi di uscita del NXT (motori, display, altoparlante e Bluetooth).

Motor:
Comanda un singolo motore.

Sound:
Riproduce suoni.

Display:
Comanda il display.

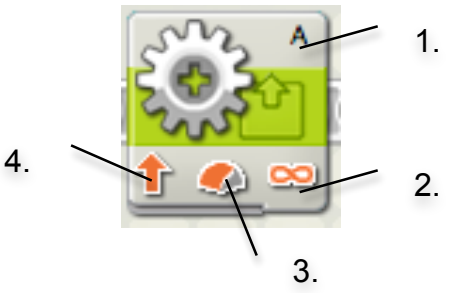
Send Message:
Invia un messaggio tramite Bluetooth.

Motor*:
Controlla un motore di tipo RCX.

Lamp*:
Controlla una lampadina.

Nota: Alcuni dei blocchi Action sono presenti anche nella common palette e sono quindi già stati trattati in precedenza.

Il blocco *Motor* permette di comandare un singolo motore.

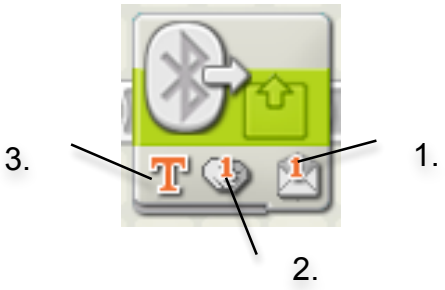


1. Porta controllata.
2. Rotazione.
3. Potenza.
4. Direzione.

Il pannello di configurazione del blocco *Motor*:

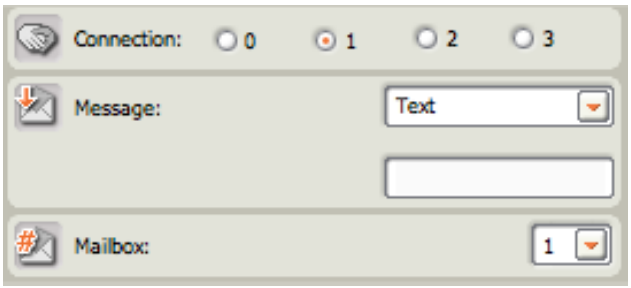
- **Port:** la porta controllata dal blocco.
- **Direction:** il senso di rotazione del motore.
- **Action:** l'azione da compiere:
  - **Constant:** il motore gira a velocità costante.
  - **Ramp Up:** il motore accelera fino a raggiungere la potenza impostata.
  - **Ramp Down:** il motore decelera fino a fermarsi.
- **Power:** la potenza del motore.
- **Control:** attiva la funzione di controllo della potenza che cerca di mantenere la velocità di rotazione del motore costante compensando resistenza e slittamento.
- **Wait:** se l'opzione *Wait for complention* è selezionata le istruzioni successive al blocco Motor verranno eseguite solo alla fine dell'azione del motore.
- **Duration:** la durata del movimento, comprende quattro impostazioni possibili:
  - **Unlimited:** durata del movimento indefinita, i motori gireranno fino a quando non verranno fermati.
  - **Degrees:** durata espressa in gradi, i motori effettuano una rotazione pari ai gradi specificati e poi si fermano.
  - **Rotations:** durata espressa in rotazioni, i motori girano effettuano le rotazioni specificate e poi si fermano.
  - **Seconds:** durata espressa in secondi, i motori girano per il tempo specificato e poi si fermano.
- **Next Action:** l'azione da compiere alla fine della rotazione:
  - **Brake:** alla fine della rotazione i motori si bloccano.
  - **Coast:** alla fine della rotazione i motori rallentano gradualmente fino a fermarsi.

Il blocco *Send Message* permette di inviare un messaggio ad un altro NXT tramite l'interfaccia Bluetooth.



1. Mailbox.
2. Connessione.
3. Tipo di messaggio.

Il pannello di configurazione del blocco *Send Message*:



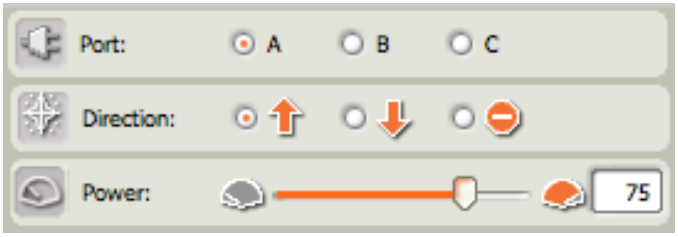
- **Connection:** il numero della connessione da utilizzare. Il numero di connessione corrisponde all'indirizzo di un NXT specifico.
- **Message:** il tipo e il contenuto del messaggio.
  - **Text:** invia una stringa di testo.
  - **Number:** invia un numero.
  - **Logic:** invia un messaggio booleano (vero, falso).
- **Mailbox:** il numero di casella in cui depositare il messaggio. Ogni NXT possiede dieci *mailbox* in cui i messaggi possono essere depositati; ognuna può contenere fino a cinque messaggi. Se una mailbox è piena, la ricezione di un nuovo messaggio provoca la cancellazione del messaggio più vecchio.

*Motor**

Il blocco *Motor** permette di comandare in un singolo motore RCX collegato ad una porta di uscita tramite il cavo di conversione NXT/RCX.

1. Porta controllata.
2. Potenza.
3. Direzione.

Il pannello di configurazione del blocco *Motor**:



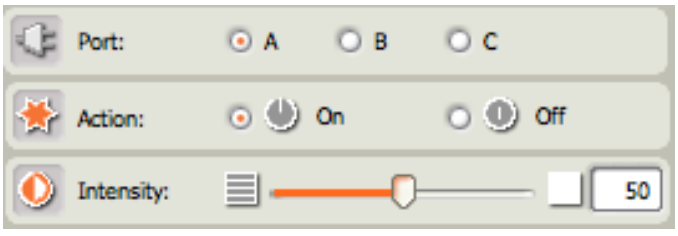
- **Port:** la porta controllata dal blocco.
- **Direction:** il senso di rotazione del motore.
- **Power:** la potenza del motore.

*Lamp**

Il blocco *Lamp** permette di comandare una lampada collegata ad una porta di uscita tramite il cavo di conversione NXT/RCX.

1. Porta controllata.
2. Potenza.
3. On/Off.

Il pannello di configurazione del blocco *Lamp**:



- **Port:** la porta controllata dal blocco.
- **Action:** lampada accesa o spenta.
- **Intensity:** l'intensità luminosa.

Blocchi Sensor

I blocchi *Sensor* servono a leggere un ingresso, esiste un blocco *Sensor* per ogni tipo di sensore e sorgente di dati. Questi blocchi leggono il valore dell'ingresso specificato dal programmatore e lo forniscono ad altri blocchi tramite un *data wire*. I blocchi dedicati ai sensori permettono anche di produrre un segnale logico prodotto confrontando il valore letto con dei valori impostati dal programmatore.

Touch Sensor:
Legge lo stato di un sensore tattile.

Sound Sensor:
Legge il valore di un sensore sonoro.

Light Sensor:
Legge il valore di un sensore di luce.

Ultrasonic Sensor:
Legge la distanza misurata dal sensore ad ultrasuoni.

NXT Buttons:
Legge lo stato dei pulsanti del NXT.

Rotation Sensor:
Legge la posizione di un motore. Permette anche il reset del sensore di rotazione.

Timer:
Legge il valore di uno dei tre timer del NXT. Permette anche l'azzeramento di un timer.

Receive Message:
Permette di ricevere un messaggio tramite Bluetooth.

Touch* Sensor:
Legge lo stato di un sensore tattile RCX.

Rotation* Sensor:
Legge lo stato di un sensore di rotazione RCX.

Light* Sensor:
Legge lo stato di un sensore di luce RCX.

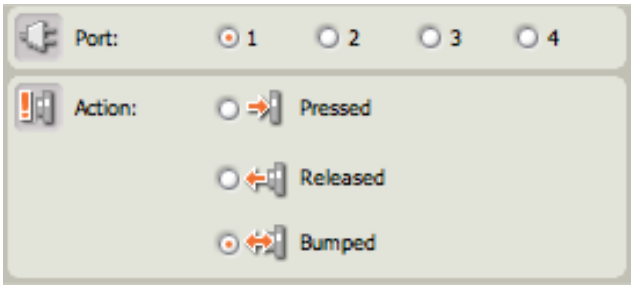
Temperature* Sensor:
Legge lo stato di un sensore di temperatura RCX.

Il blocco *Touch Sensor* serve a leggere lo stato di un sensore tattile.

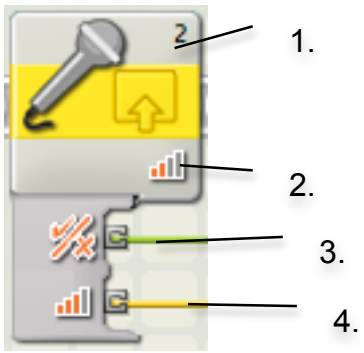
1. Porta letta.
2. Azione.
3. Stato.

Il pannello di configurazione del blocco *Touch Sensor*:

- **Port:** il numero della porta da leggere.
- **Action:** l'azione da attendere:
  - **Pressed:** l'uscita (*data wire*) del blocco assume il valore *true* se il sensore è premuto, altrimenti *false*.
  - **Released:** l'uscita del blocco assume il valore *true* se il sensore è rilasciato, *false* altrimenti.
  - **Bumped:** l'uscita del blocco assume il valore *true* quando il sensore viene premuto e rilasciato, *false* altrimenti.

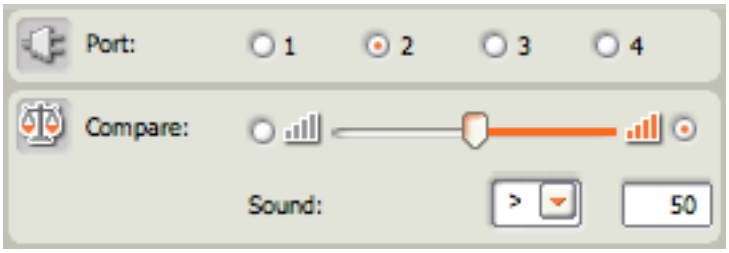


Il blocco *Sound Sensor* legge il valore misurato da un sensore sonoro.

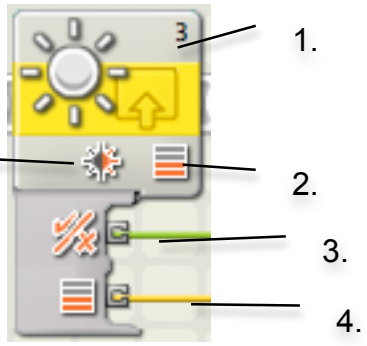


1. Porta di ingresso.
2. Livello di soglia.
3. Uscita Stato: assume il valore *true* quando il livello di soglia viene sorpassato.
4. Uscita Livello sonoro.

Il pannello di configurazione del blocco *Sound Sensor*:

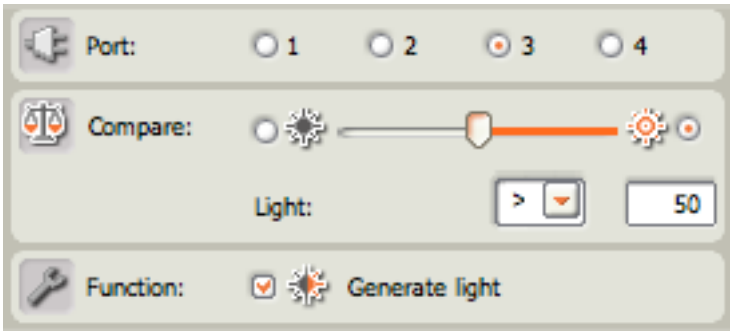


- **Port:** il numero della porta da leggere.
- **Compare:** il livello di soglia.



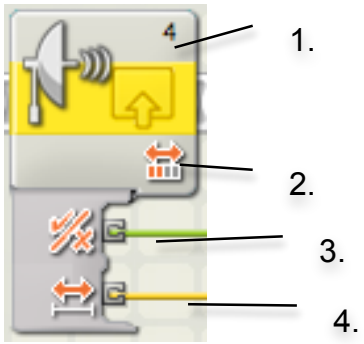
1. Porta di ingresso.
2. Livello di soglia.
3. Uscita Stato: assume il valore *true* quando il livello di soglia viene sorpassato.
4. Uscita Intensità luminosa.
5. Generate Light.

Il pannello di configurazione del blocco *Light Sensor*:



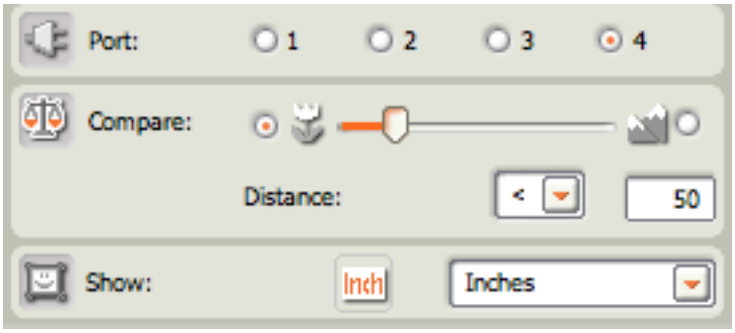
- **Port:** il numero della porta da leggere.
- **Compare:** il livello di soglia.
- **Function:** se l'opzione *Generate light* è attivata il led del sensore viene acceso e il sensore misura la luce riflessa, altrimenti viene misurata la luminosità dell'ambiente

Il blocco *Ultrasonic Sensor* legge la distanza misurata da un sensore ad ultrasuoni.



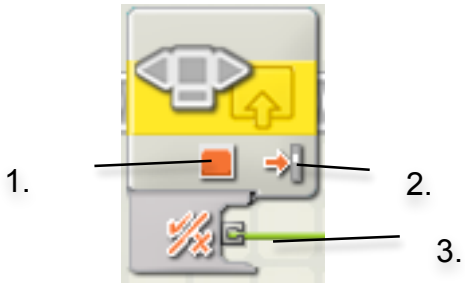
1. Porta di ingresso.
2. Soglia.
3. Uscita Stato: assume il valore *true* quando la distanza di soglia viene sorpassata.
4. Uscita Distanza.

Il pannello di configurazione del blocco *Ultrasonic Sensor*:



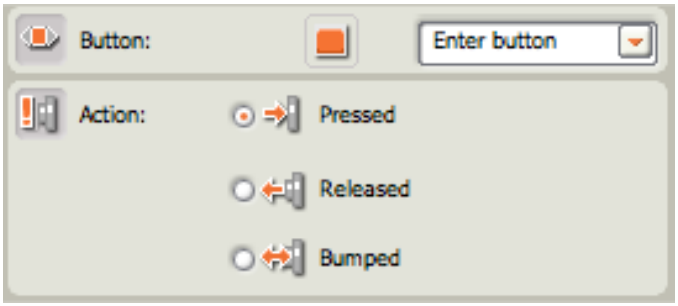
- **Port:** il numero della porta da leggere.
- **Compare:** la distanza di soglia.
- **Show:** unità di misura (pollici/centimetri).

Il blocco *NXT Buttons* fornisce lo stato dei bottoni del NXT.



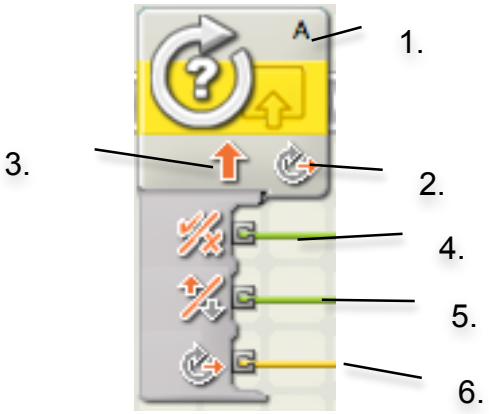
1. Bottone.
2. Azione.
3. Uscita Stato.

Il pannello di configurazione del blocco *NXT Buttons*:



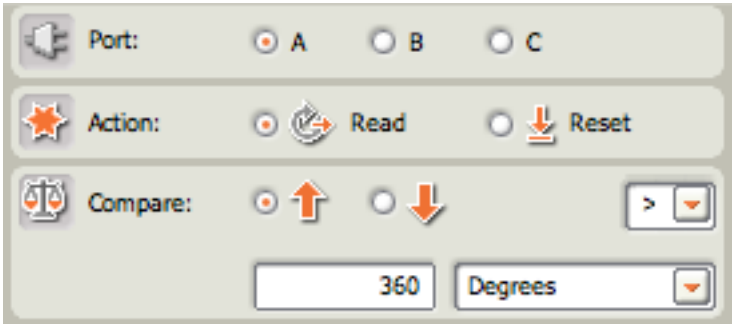
- **Button:** il bottone.
- **Action:** l'azione da monitorare:
  - **Pressed:** l'uscita (*data wire*) del blocco assume il valore *true* se il bottone è premuto, altrimenti *false*.
  - **Released:** l'uscita del blocco assume il valore *true* se il bottone è rilasciato, *false* altrimenti.
  - **Bumped:** l'uscita del blocco assume il valore *true* quando il bottone viene premuto e rilasciato, *false* altrimenti.

Il blocco *Rotation Sensor* fornisce la rotazione di un motore e premette di azzerare il contatore.



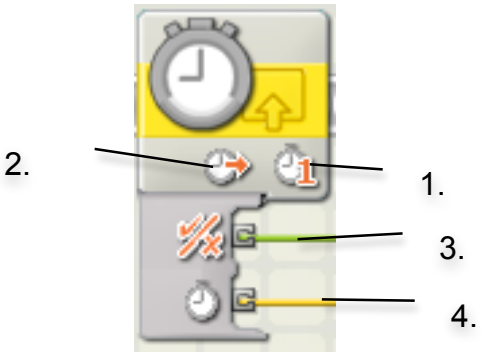
1. Porta.
2. Azione.
3. Direzione.
4. Uscita Stato (soglia superata).
5. Uscita Direzione.
6. Uscita Rotazione (Gradi)

Il pannello di configurazione del blocco *Rotation Sensor*:



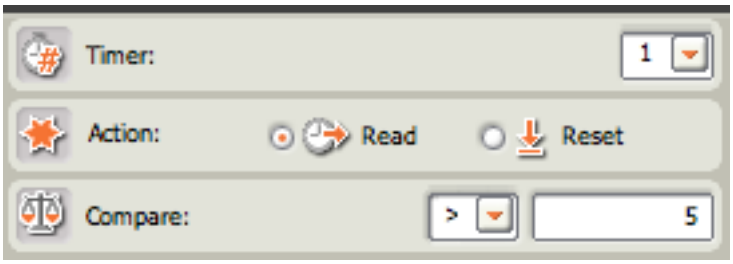
- **Port:** la porta di uscita.
- **Action:** l'azione da compiere:
  - **Read:** legge la posizione del sensore.
  - **Reset:** azzera il contatore del sensore di rotazione.
- **Compare:** il valore di soglia oltre il quale l'uscita logica Stato assume valore *true*.

Il blocco *Timer* permette di leggere lo stato di uno dei tre timer del NXT e di confrontarlo con un valore. I timer partono automaticamente ogni volta che viene eseguito un programma. Il blocco *Timer* può anche essere utilizzato per azzerare un timer.



1. Numero di Timer.
2. Azione.
3. Uscita Stato.
4. Uscita Tempo (millisecondi).

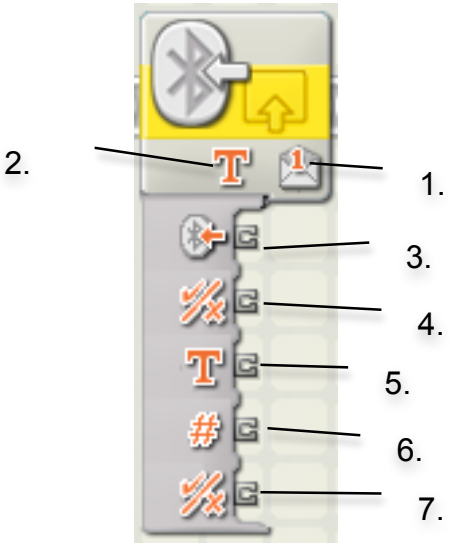
Il pannello di configurazione del blocco *Timer*:



- **Timer:** il timer (1,2 o 3).
- **Action:** l'azione da compiere:
  - **Read:** legge il timer.
  - **Reset:** azzera il timer.
- **Compare:** la soglia del timer espressa in secondi. Quando il valore del timer supera la soglia, l'uscita Stato assume valore *true*.

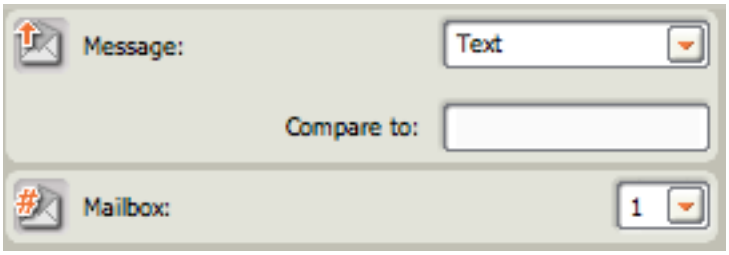
Receive Message

Il blocco *Receive Message* serve a verificare la ricezione di un messaggio e può essere utilizzato anche per confrontare il contenuto del messaggio ricevuto con un valore.



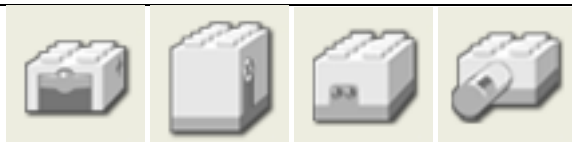
1. Numero di *Mailbox*.
2. Tipo di messaggio.
3. Uscita Messaggio ricevuto.
4. Uscita Logico.
5. Uscita Testo.
6. Uscita Numero.
7. Uscita Stato.

Il pannello di configurazione del blocco *Timer*:



- **Message:**
  - **Tipo di messaggio**:
    - **Text:** riceve una stringa di testo.
    - **Number:** riceve un numero.
    - **Logic:** riceve un messaggio booleano (vero, falso).
  - **Compare to**: un valore da confrontare con il contenuto del messaggio. L'usctita Stato assume il valore *true* se i due valori coincidono.
- **Mailbox:** il numero di mailbox da leggere (1 - 10).

Blocchi RCX Sensor*

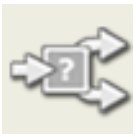


Permettono di utilizzare i sensori del vecchio kit Mindstorms RCX e funzionano in modo analogo ai blocchi *Touch Sensor, Rotation Sensor e Light Sensor* già trattati in precedenza.

Blocchi Flow

I blocchi nel gruppo *Flow* servono a controllare il flusso logico del programma.

Wait:
Attende l'avverarsi di una condizione.



Switch:
Implementa la struttura di controllo di selezione.

Loop:
Implementa la struttura di controllo di iterazione.

Stop:
Arresta l'esecuzione del programma.

Nota: I blocchi *Wait*, *Loop* e *Switch* sono presenti anche nella common palette e sono quindi già stati trattati in precedenza.

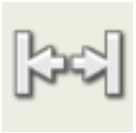
Stop

Il blocco *Stop* provoca l'arresto dell'esecuzione del programma e spegne tutte le porte di uscita del NXT. Il blocco *Stop* non è parametrizzabile e non possiede quindi un pannello di configurazione.

Blocchi Data

Questi blocchi permettono di eseguire diverse operazioni sui dati e possono interagire con altri blocchi tramite *data wire*.

Logic:
Esegue operazioni logiche And, Or, Xor e Not.



Range:
Verifica se un valore numerico è compreso in un intervallo.

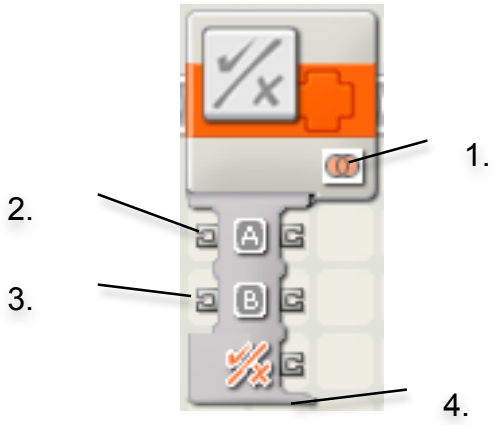
Math:
Esegue le operazioni aritmetiche di somma, sottrazione, moltiplicazione e divisione.

Random:
Genera numeri casuali.

Compare:
Implementa gli operatori di confronto “uguale”, “maggiore di” e “minore di”.

Variable:
Permette l’accesso in lettura e scrittura alle variabili.

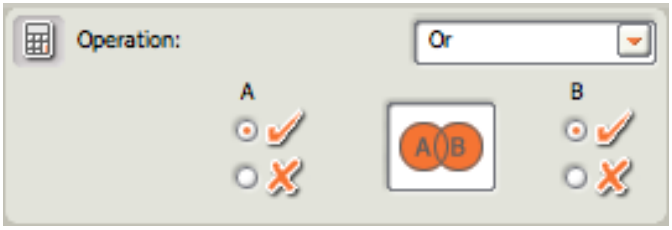
Il blocco *Logic* esegue le quattro operazioni logiche AND (congiunzione), OR (disgiunzione), XOR (disgiunzione esclusiva) e NOT (negazione) e produce un risultato di tipo logico (vero/falso) sul *data wire* di uscita. Gli operandi possono essere forniti tramite i *data wire* di tipo logico A e B, oppure predefiniti nel pannello di configurazione del blocco.



1. Operazione.
2. A: Primo operando.
3. B: Secondo operando.
4. Risultato.

Il pannello di configurazione del blocco *Logic* permette di decidere quale operatore logico applicare agli operandi e permette anche di definire il valore di *default* per gli operandi:

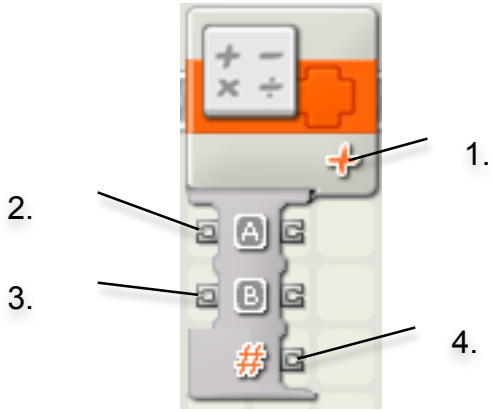
- **Operation:** operazione da svolgere:
  - **AND**: l'uscita del blocco assume valore *vero* se A e B valgono entrambi *vero*.
  - **OR**: l'uscita del blocco assume valore *vero* se A o B o entrambi valgono *vero*.
  - **XOR**: l'uscita del blocco assume valore *vero* se A è diverso da B.
  - **NOT**: si applica ad un solo operando (A). L'uscita assume il valore negato dell'ingresso.
- **A**: valore di default per il primo operando. Se all'ingresso A non è collegato nessun *data wire* il primo operando assume il valore definito qui.
- **B**: valore di default per il secondo operando.



Operazioni logiche:

Ingresso A	Ingresso B	A AND B	A OR B	A XOR B	NOT A
Falso (0)	Falso (0)	Falso (0)	Falso (0)	Falso (0)	Vero (1)
Falso (0)	Vero (1)	Falso (0)	Vero (1)	Vero (1)	Vero (1)
Vero (1)	Falso (0)	Falso (0)	Vero (1)	Vero (1)	Falso (0)
Vero (1)	Vero (1)	Vero (1)	Vero (1)	Falso (0)	Falso (0)

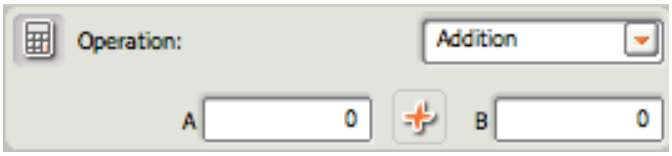
Il blocco *Math* serve per svolgere operazioni aritmetiche di somma, sottrazione, moltiplicazione e divisione. Questo blocco produce un risultato di tipo numerico sul *data wire* di uscita. Gli operandi devono essere di tipo numerico e possono essere forniti tramite i *data wire* di tipo logico A e B, oppure predefiniti nel pannello di configurazione del blocco.

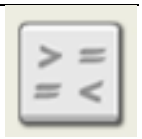


1. Operazione.
2. A: Primo operando.
3. B: Secondo operando.
4. Risultato.

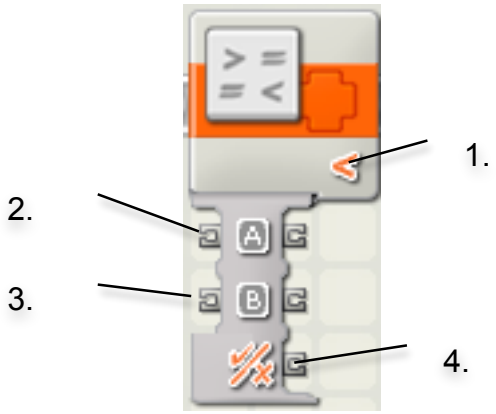
Il pannello di configurazione del blocco *Math* permette di decidere quale operazione svolgere e di definire il valore di *default* per gli operandi:

- **Operation:** operazione da svolgere:
  - **Addition**: l'uscita del blocco assume valore A+B.
  - **Subtraction**: l'uscita del blocco assume valore A-B.
  - **Multiplication**: l'uscita del blocco assume valore A*B.
  - **Division**: l'uscita del blocco assume valore A/B.
- **A**: valore di default per il primo operando. Se all'ingresso A non è collegato nessun *data wire* il primo operando assume il valore definito qui.
- **B**: valore di default per il secondo operando.





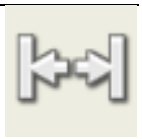
Il blocco *Compare* serve a confrontare due valori numerici. Gli operandi possono essere forniti tramite i *data wire* di tipo logico A e B, oppure predefiniti nel pannello di configurazione del blocco. Il blocco *Logic* compara i due operandi e produce un risultato di tipo logico (vero/falso) sul *data wire* di uscita.



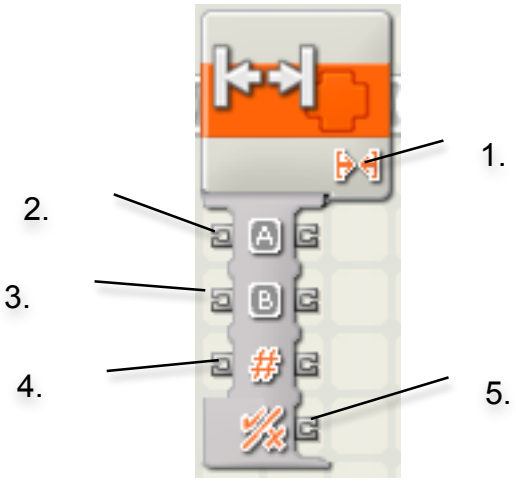
1. Operazione.
2. A: Primo operando.
3. B: Secondo operando.
4. Risultato.

Il pannello di configurazione del blocco *Compare* permette di decidere quale operatore relazionle applicare agli operandi. Inoltre permette di definire il valore di *default* per gli operandi:

- **Operation:** operazione da svolgere:
  - **Less than**: l'uscita del blocco assume valore *vero* se A è minore di B.
  - **Greater than**: l'uscita del blocco assume valore *vero* se A è maggiore di B.
  - **Equals**: l'uscita del blocco assume valore *vero* se A è uguale a B.
- **A**: valore di default per il primo operando. Se all'ingresso A non è collegato nessun *data wire* il primo operando assume il valore definito qui.
- **B**: valore di default per il secondo operando.



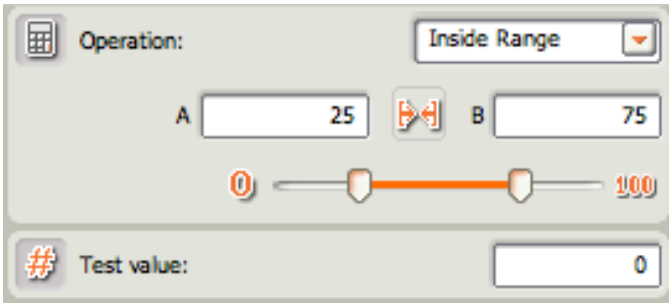
Il blocco *Range* serve a verificare se un valore numerico fornito in ingresso è compreso in un intervallo. I limiti dell'intervallo possono essere introdotti nel pannello di configurazione del blocco oppure passati tramite *data wire*. Il blocco *Compare* produce un risultato di tipo logico (vero/falso) sul *data wire* di uscita.



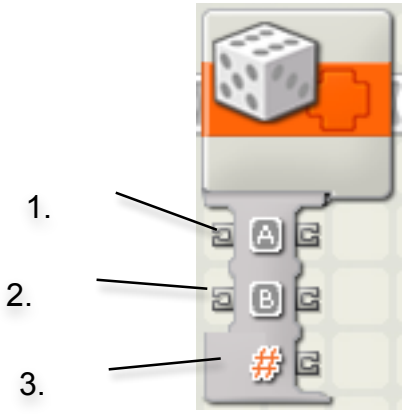
1. Operazione.
2. A: limite inferiore.
3. B: limite superiore.
4. #: Valore.
5. Risultato.

Il pannello di configurazione del blocco *Range* permette di definire i limiti dell'intervallo, il tipo di operazione da svolgere e il *valore di test*:

- **Operation:** operazione da svolgere:
  - **Inside Range**: l'uscita del blocco assume valore *vero* se il valore in ingresso è compreso nell'intervallo.
  - **Outside Range**: l'uscita del blocco assume valore *vero* se il valore in ingresso non è compreso nell'intervallo.
- **A**: il limite inferiore dell'intervallo. Se all'ingresso A non è collegato nessun *data wire* viene utilizzato il valore definito qui.
- **B**: il limite superiore dell'intervallo.
- **Test value**: il valore da applicare nel caso non ci sia nessun *data wire* collegato alla porta di ingresso #.



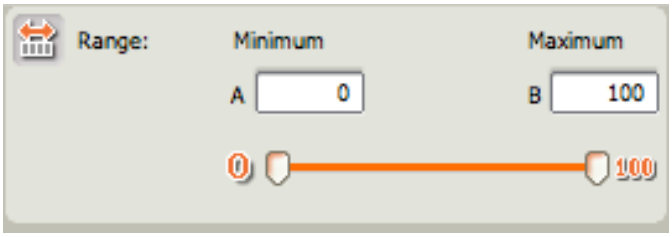
Il blocco *Random* genera un numero aleatorio compreso in un intervallo definito tramite due *data wire* oppure nel pannello di configurazione. Il numero generato viene prodotto sull'uscita del blocco.



1. A: limite inferiore.
2. B: limite superiore.
3. #: Numero casuale.

Il pannello di configurazione del blocco *Random* permette di definire il valore massimo e il valore minimo entro cui deve essere compreso il numero generato:

- **Range:** il range entro cui generare il valore casuale:
  - **Minimum**: il valore minimo del numero generato.
  - **Maximum**: il valore massimo del numero generato.



Il blocco *Variable* fornisce l'accesso a una *variabile*. Una *variabile* è un "posto" dove memorizzare un'informazione nella memoria del NXT. NXT-G supporta tre tipi di variabili:

- **Logic**: può memorizzare un'informazione di tipo logico (vero/falso).
- **Number**: per memorizzare un numero.
- **Text**: per memorizzare una stringa di testo.

Tutti i programmi dispongono di tre variabili di default, una per ogni tipo, che possono essere utilizzate con il blocco *Variable*. All'occorrenza altre variabili possono essere create utilizzando la funzione *Define Variables* che si trova nel menu *Edit* dell'ambiente di sviluppo. Il blocco *Variable* fornisce due modi di accesso alle variabili: *Read* per leggere il valore di una variabile e *Write* per modificarlo. La lettura e la scrittura dei valori delle variabili avviene tramite *data wire*.

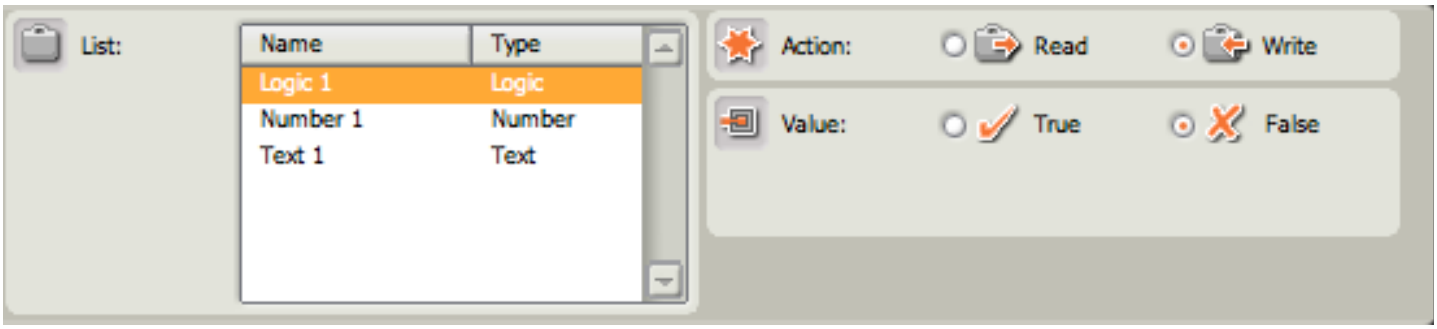
Read

1. Operazione (lettura).
2. Valore variabile.

Write

1. Operazione (scrittura).
2. Ingresso.

Il pannello di configurazione del blocco *Variable* consente di scegliere quale variabile accedere e che tipo di accesso effettuare. In modalità *Write* il pannello permette anche di definire un valore per la variabile:



- **List**: lista delle variabili disponibili, permette di scegliere quale variabile accedere.
- **Action**: il tipo di accesso**:**
  - **Read**: legge il valore della variabile. Il valore della variabile viene prodotto sull'uscita del blocco.
  - **Write**: scrive il valore della variabile. La variabile assume il valore passato all'ingresso del blocco.
- **Value**: in modalità *Write* permette di assegnare un valore alla variabile.

Blocchi Advanced

Il gruppo *Advanced* contiene dei blocchi che permettono di effettuare alcune operazioni avanzate.

Text:
Concatena fino a tre stringe di testo.

Number to Text:
Converte valori numerici in stringhe di testo.

Keep Alive:
Inibisce il timer di spegnimento automatico del NXT.

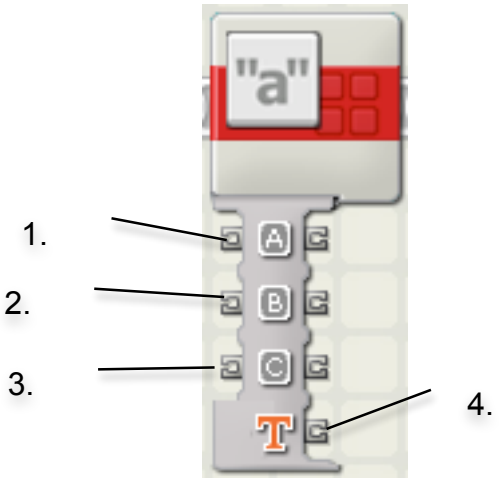
File Access:
Permette l'accesso a file.

Calibrate:
Effettua la calibrazione di un sensore di luce o sonoro.

Reset Motor:
Inibisce il meccanismo di correzione automatica di uno o più motori.

Text

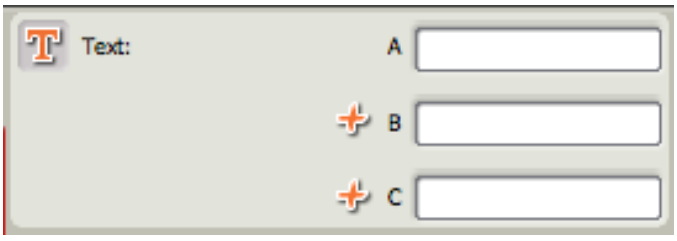
Il blocco *Text* concatena fino a tre stringhe di testo fornite tramite *data wire*.



1. A: Ingresso.
2. B: Ingresso.
3. C: Ingresso.
4. Risultato (A+B+C).

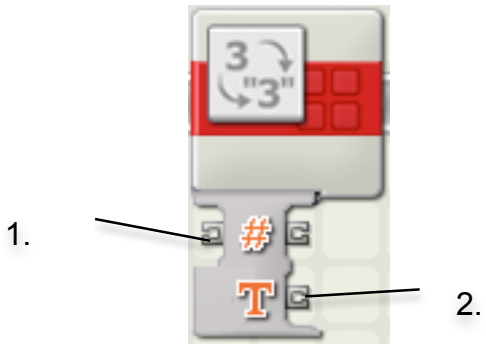
Il pannello di configurazione del blocco *Text* permette di definire il valore di *default* per le stringhe A, B e C:

- **Text:** Il valore di default per le stringhe A, B e C. Se un ingresso non è collegato nessun *data wire* la relativa stringa assume questo valore.



Number to Text

Questo blocco converte il valore numerico in ingresso in una stringa di testo che produce sul *data wire* di uscita.

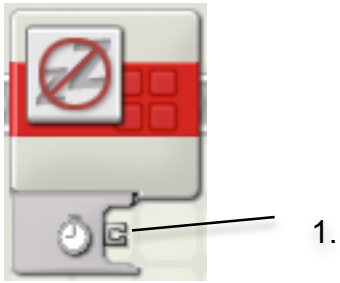


1. #: Ingresso (numero).
2. Risultato (testo).

Il pannello di configurazione del blocco *Number to Text* permette di definire il valore di *default* del valore numerico in ingresso.

Keep Alive

Questo blocco inibisce il timer di spegnimento automatico del NXT. L'utilizzo di questo blocco permette di scrivere dei programmi la cui esecuzione deve durare più a lungo del tempo impostato nel timer di spegnimento automatico.

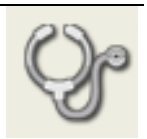


1. *Sleep Time* (ms).

Questo blocco non dispone di un pannello di configurazione.

File Access

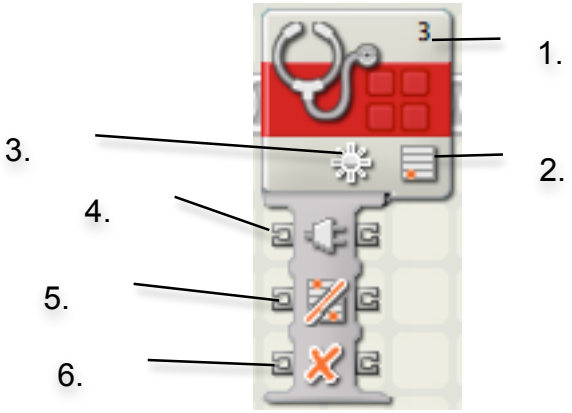
Permette le operazioni di accesso al *file system* del NXT. Questo blocco può essere utilizzato per accedere in lettura e/o scrittura ai file presenti nella memoria del NXT.



Il blocco *Calibrate* serve alla calibrazione automatica di sensori ottici e sonori. Permette di calibrare il valore massimo e il valore minimo misurati da un sensore o di cancellare una calibrazione già esistente.

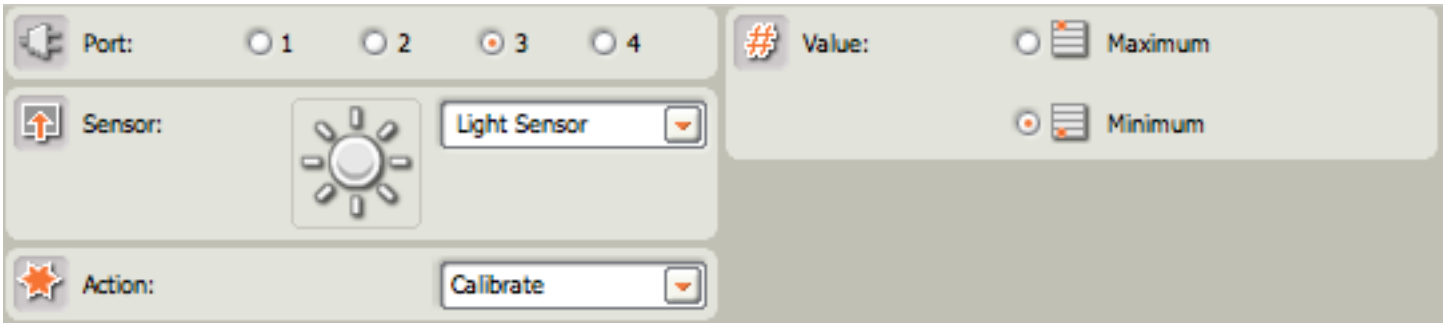
Tra le prime istruzioni di un programma si possono inserire due blocchi *Calibrate* per registrare il valore minimo ed il valore massimo da applicare alla lettura del sensore. In questo modo la sensibilità del sensore viene adattata alle condizioni ambientali, evitando al programmatore di dovere ricalcolare il valori di soglia dei sensori e di ricompilare i propri programmi ogni volta che tali condizioni cambiano.

Tutti i parametri della calibrazione possono essere definiti tramite i *data wire* di ingresso del blocco oppure tramite il pannello di configurazione:



1. Indicazione numero della porta di ingresso del sensore.
2. Valore calibrato (massimo o minimo).
3. Ingresso: numero di porta.
4. Ingresso: valore calibrato.
5. Ingresso: cancellazione della calibrazione.

Il pannello di configurazione del blocco *Calibrate*:



- **Port**: il numero di porta cui è connesso il sensore da calibrare.
- **Sensor**: il tipo di sensore da calibrare (*Sound* o *Light*).
- **Action**: l'azione da compiere:
  - **Calibrate**: calibra il sensore.
  - **Delete**: cancella la calibrazione del sensore selezionato.
- **Value**: il valore da calibrare:
  - **Maximum**: viene registrato il valore massimo che il sensore dovrà misurare.
  - **Minimum**: viene registrato il valore minimo che il sensore dovrà misurare.

Questo blocco inibisce il meccanismo di correzione automatica di uno o più motori. L'utilizzo del blocco *Reset Motor* consente quindi di disattivare il sistema automatico che controlla la rotazione di un motore quando esso è in stato *floating* (quando cioè le uscite sono spente ma l'asse del motore gira per qualche ragione).

Programmazione avanzata

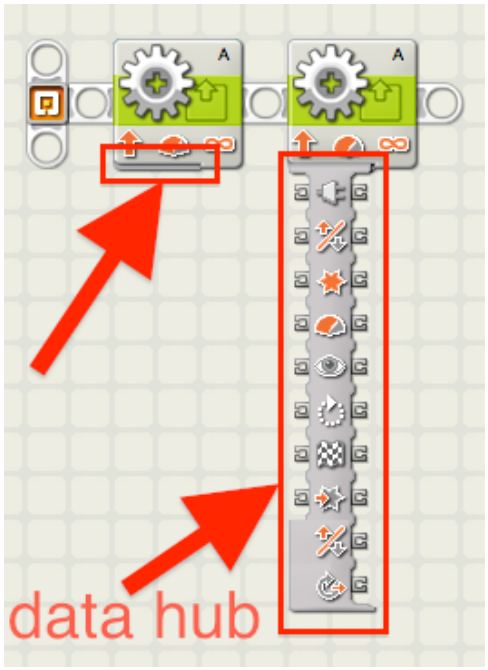
In questo capitolo vengono trattate alcune nozioni basilari della *programmazione* e alcuni aspetti della programmazione avanzata nel linguaggio NXT-G.

Data wire

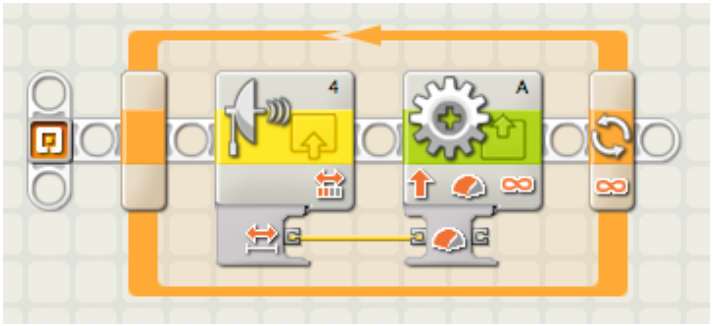
I *data wire* servono a trasferire dati da un blocco all'altro. Molti blocchi posseggono infatti delle porte di ingresso e di uscita, dette *pin,* che possono essere collegate tramite dei "fili virtuali" detti appunto *data wire*.

Data hub

I *pin* di ingresso e di uscita si trovano nei *data hub* dei blocchi. I *data hub* sono normalmente chiusi ma possono essere aperti cliccando sul profilo inferiore del blocco:



Generalmente i *pin* di ingresso permettono di modificare i parametri di un blocco in modo *programmatico* (durante l'esecuzione del programma). I pin di uscita permettono invece di prelevare dei dati da un blocco; nell'esempio seguente il valore misurato dal blocco *Ultrasonic Sensor* viene utilizzato per modificare la potenza del blocco *Motor* successivo:

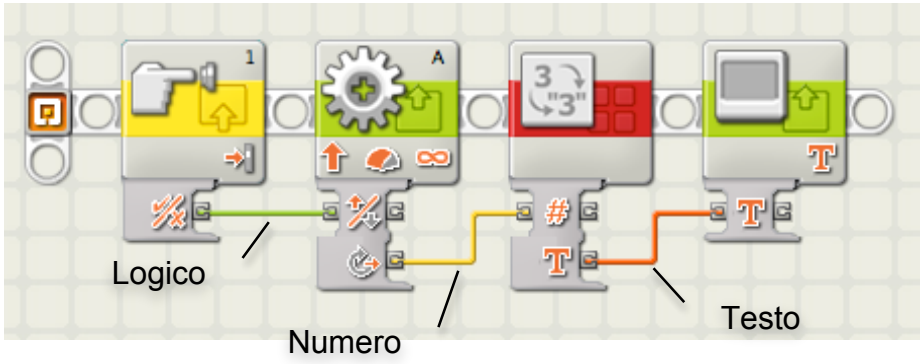


L'elenco dei *pin* disponibili è reperibile nella documentazione di ogni singolo blocco (help).

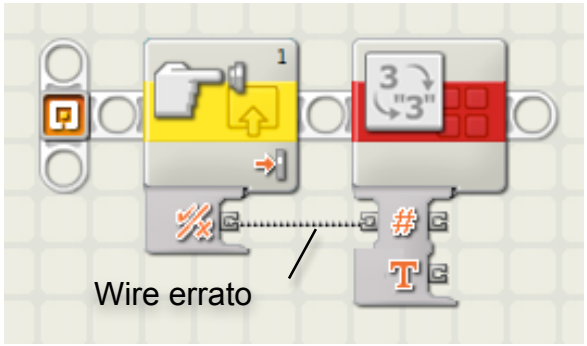
Tipi di dati

I dati trasferiti tra i blocchi possono essere di *tipo* differente in funzione della natura dell'informazione da comunicare. Il linguaggio NXT-G prevede tre tipi di dati:

- **Numerici:** Numeri (ad esempio i valori prodotti da un sensore, la potenza di un motore). Sono rappresentati in colore giallo.
- **Logici:** Valori booleani (Vero/Falso, come lo stato di un sensore tattile, il risultato di un'operazione logica). Rappresentati in verde.
- **Testo:** Delle stringhe di testo (come una parola da stampare sullo schermo, il nome di un file); disegnati in arancio.



Un *pin* di uscita di un blocco si può collegare ad un *pin* di ingresso di un altro blocco a condizione che i due *pin* siano dello stesso tipo. Se in un programma sono presenti *data wire* connessi in modo errato (o connessi ad un solo *pin*) il filo viene rappresentato tramite un tratteggio grigio:

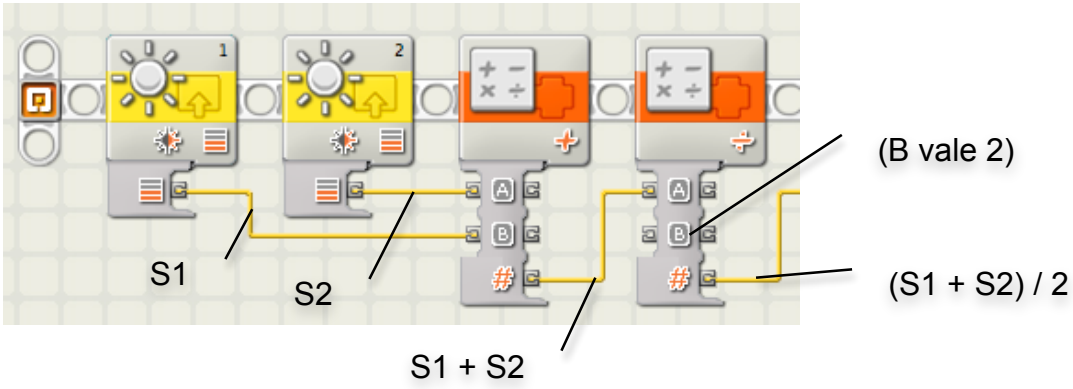


Operazioni su dati numerici

È possibile effettuare delle operazioni algebriche e di confronto utilizzando i blocchi *Math*, *Compare* e *Range* (Complete palette>Data).

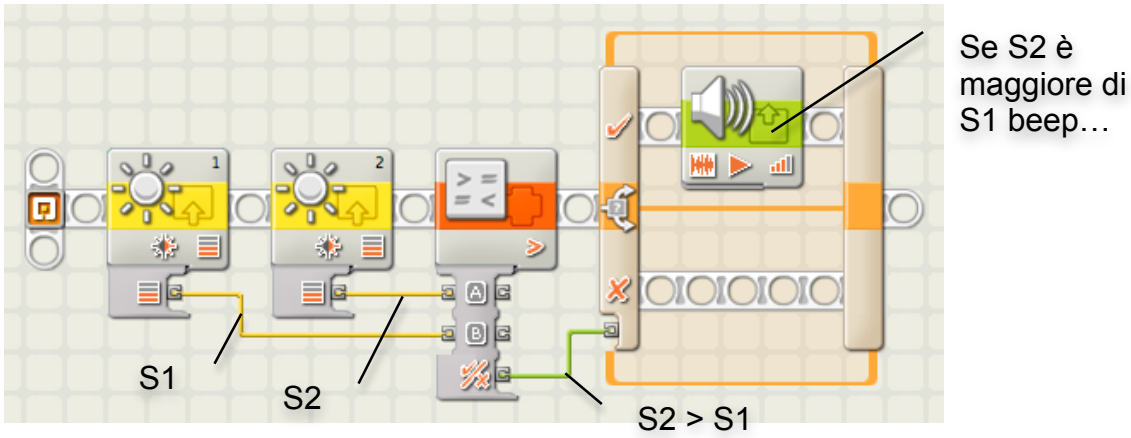
Operazioni algebriche

Il blocco *Math* implementa le operazioni algebriche si somma, sottrazione, prodotto e divisione. Nell'esempio seguente viene calcolata la media del valore letto da due sensori di luce:

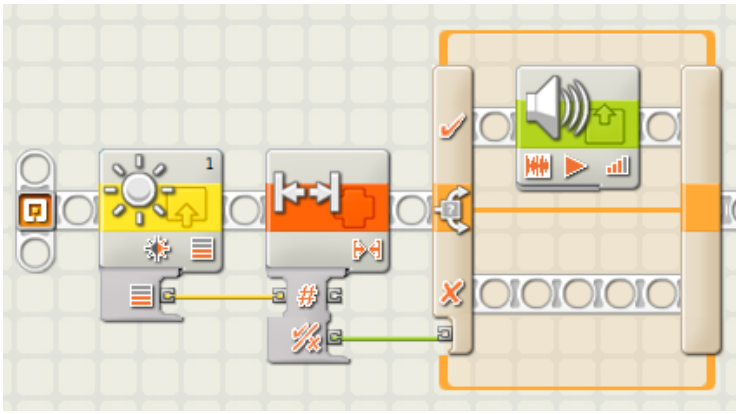


Operazioni di confronto

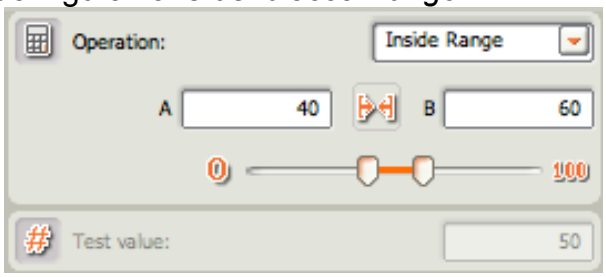
Gli operatori relazionali *minore, maggiore e uguale* sono implementati dal blocco C*ompare* che può essere utilizzato per valutare due valori di tipo numerico e produce un risultato di tipo logico (Vero/falso). Il codice rappresentato a seguito confronta il valore prodotto da due sensori di luce e emette un suono se il valore di S2 è maggiore rispetto a quello di S1:



Oltre al blocco *Compare* esiste il blocco *Range* che permette di verificare l'appartenenza di un valore ad un intervallo numerico definito nel pannello di configurazione del blocco o tramite i *data wire* di ingresso. Nell'esempio seguente il blocco *Range* viene utilizzato per verificare se il valore letto da un sensore di luce è compreso nell'intervallo [40;60]:

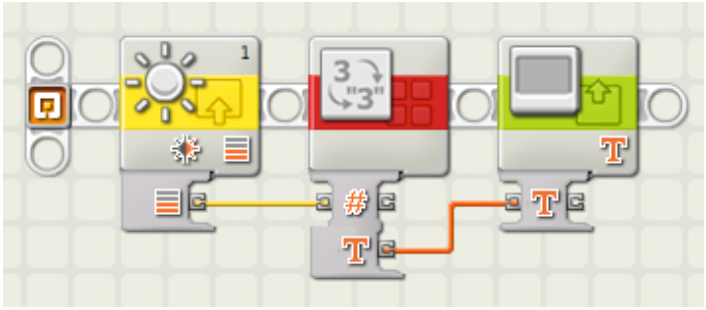


Configurazione del blocco *Range*:



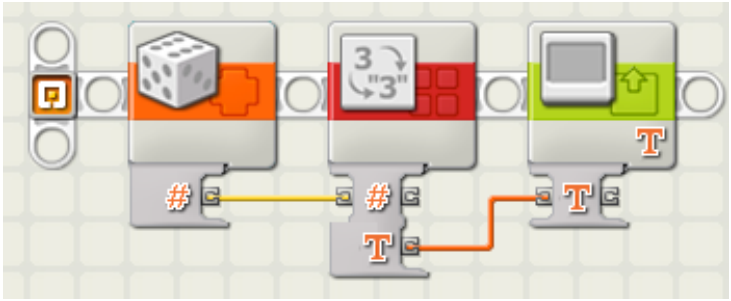
Conversione da numeri a testo

Volendo stampare sul display del NXT un valore numerico (e in altre rare occasioni) è necessario effettuare una conversione da numero a testo. Il blocco *Display* non prevede infatti un ingresso di tipo numerico. La conversione da numero a testo è realizzata dal Blocco *Number To Text*:



Generazione di numeri casuali

Il blocco *Random* genera un numero aleatorio compreso in un intervallo definito tramite due *data wire* oppure nel pannello di configurazione. Il numero generato viene prodotto sull'uscita del blocco:



Operazioni su dati logici

In programmazione viene fatto un ampio utilizzo dell'algebra booleana, in modo particolare dei *connettivi logici*. Grazie al lavoro di Boole è possibile rappresentare la logica tramite espressioni algebriche, perfette per la definizione di *condizioni logiche* all'interno dei programmi.

Enunciati

Gli oggetti dell'algebra di Boole gli enunciati. Un enunciato è una proposizione che può essere soltanto vera o falsa. Alcuni esempi:

- "il sensore S1 è premuto", "il valore di S2 è maggiore di 50" sono enunciati.
- "che ore sono", "dove siete stati?" non sono enunciati in quanto non sono ne veri ne falsi

La verità o falsità di un enunciato sono dette *valori di verità*; un enunciato può essere vero o falso, ma non entrambe le cose.

Ovviamente in programmazione siamo interessati dagli **enunciati** proprio per la loro natura binaria: possono essere solo veri o falsi.

Connettivi logici

Gli enunciati possono essere composti, vale a dire che possono essere formati da sottoenunciati collegati fra loro da connettivi. Esempio:

"il sensore S1 è premuto **oppure** il sensore S2 è premuto "

è un enunciato composto dai due sottoenunciati "oggi piove" e "oggi nevica" e dal connettivo "oppure".

Il valore di verità di un enunciato composto è definito dai valori di verità dei suoi sottoenunciati e dai connettivi che li uniscono. Vi sono tre *connettivi* fondamentali tramite i quali si può definire qualsiasi logica possibile.

Nell'algebra di Boole, per indicare gli enunciati si usano di solito le lettere p, q, r…

Congiunzione: AND

Due enunciati possono essere collegati dal connettivo *"e"*. In inglese ed in informatica la *congiunzione* viene spesso espressa come "and", "AND", oppure con il simbolo "&" o ancora "&&".

Un enunciato composto da due sottoenunciati p e q è detto "congiunzione di p e q" e viene espresso in simboli come "p and q".

I valori di verità di un *enunciato composto* possono essere espressi tramite una *tavola della verità* analogamente alle equazioni logiche:

p	q	p and q
V	V	V
V	F	F
F	V	F
F	F	F

Tavola della verità della congiunzione (AND). V indica vero mentre F indica falso.

La prima riga indica in modo sintetico che se *p* è vera e *q* è vera allora *"p and q"* è vera.
Si osservi che *"p and q"* è vera solo nel caso in cui siano veri entrambi i sottoenunciati.

Disgiunzione: OR

Il connettivo "*o*", viene spesso espresso quale "or" e "OR" in inglese e dai simboli "|" oppure "||" in informatica. Un enunciato composto dal connettivo *or* è detto *disgiunzione* degli enunciati di partenza. Il *valore di verità* di "*p or q*" è dato dalla seguente tabella della verità che costituisce la definizione della disgiunzione:

p	q	p or q
V	V	V
V	F	V
F	V	V
F	F	F

Tavola della verità della disgiunzione (OR). V indica vero mentre F indica falso.

Si osservi che *p or q* è falsa solo nel caso in cui sono falsi entrambi i sottoenunciati.

Negazione: NOT

Dato un enunciato *p*, è possibile formare un altro enunciato che si indica con *not p* e che è detto *negazione* di *p*. Nel linguaggio corrente la *negazione di p* si ottiene anteponendo a p "non è vero che…" oppure inserendo in p la parola "non". La *negazione* viene spesso espressa quale "*not p*", " *p'* ", "*!p*" o "¬*p*". Il valore di verità di *not p* è dato dalla tavola:

p	**not p**
V	F
F	V

Tavola della verità della negazione (NOT). V indica vero mentre F indica falso.

Disgiunzione Esclusiva: XOR

Il connettivo xor viene rappresentato in algebra booleana tramite il simbolo "⊕" e in alcuni linguaggi di programmazione con "^". Un enunciato composto dal connettivo x*or* è detto *disgiunzione esclusiva* degli enunciati di partenza. Il *valore di verità* di "*p xor q*" è dato dalla seguente tabella della verità:

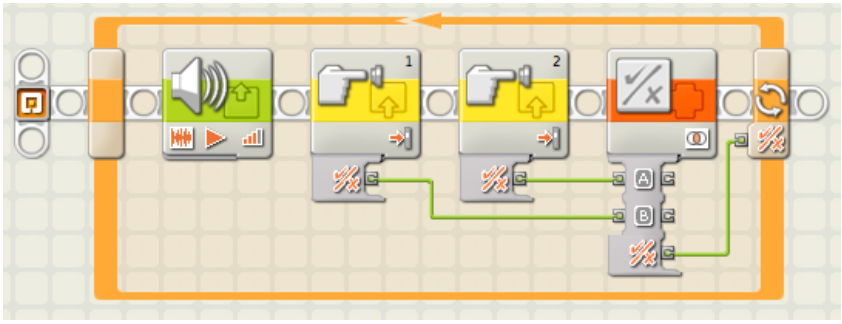
p	**q**	**p xor q**
V	V	F
V	F	V
F	V	V
F	F	F

Tavola della verità della disgiunzione esclusiva (XOR). V indica vero mentre F indica falso.

Si osservi che *p xor q* è vera quando *p* è diversa da *q*.

Il blocco Logic

Il linguaggio NXT –G implementa le funzioni logiche tramite il blocco *Logic* (*Complete Palette* > *Data* > *Logic*). Tale blocco permette di comporre enunciati (espressioni logiche) con quattro connettivi logici AND, OR, NOT e XOR. Pur non essendo un *operatore logico fondamentale* il connettivo XOR (detto anche disgiunzione esclusiva) è molto utilizzato in programmazione. Il blocco *Logic* può quindi essere utilizzato per comporre delle espressioni logiche che possono poi essere utilizzate quali condizioni per *cicli* e *selezioni*. Prendiamo ad esempio il programma seguente:



In questo esempio il ciclo termina quando entrambi i sensori tattili 1 e 2 vengono premuti:

Pseudocodice

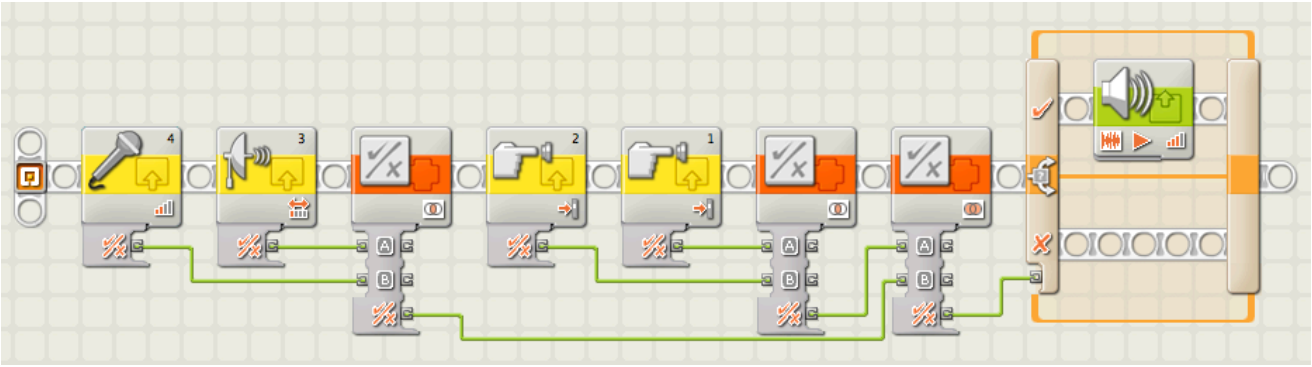
```
inizio
     esegui
          emetti un suono;
     ripeti finché NOT(S1=1 AND S2=1)
fine
```

Tabella della verità

S2	S1	S1 AND S2
0	0	0
0	1	0
1	0	0
1	1	1

In modo analogo il *data wire* che esce da un blocco *Logic* può essere utilizzato per governare una selezione (blocco *Switch* di tipo logico) oppure immesso in un altro blocco *Logic* per comporre un'espressione logica più complessa.

Nell'esempio seguente vengono utilizzati 3 blocchi *Logic* per comporre una condizione che governa un blocco *Switch*:



Pseudocodice

```
inizio
     se (S4 AND S3) OR (S2 AND S1) allora
          emetti un suono;
     fine se
fine
```

Tabella della verità

S4	S3	S2	S1	S4 AND S3 OR S2 AND S1
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Operazioni su stringhe di testo

Le uniche operazioni su stringhe disponibili in NXT-G sono quelle implementate dal blocco *Number to Text* e dal blocco *Text*. Il blocco *Text* esegue la concatenazione di due stringhe di testo.

Appendice A: Esempi di codice

Loop

Loop infinito

Questo programma ripete l'istruzione *Sound Block* all'infinito:

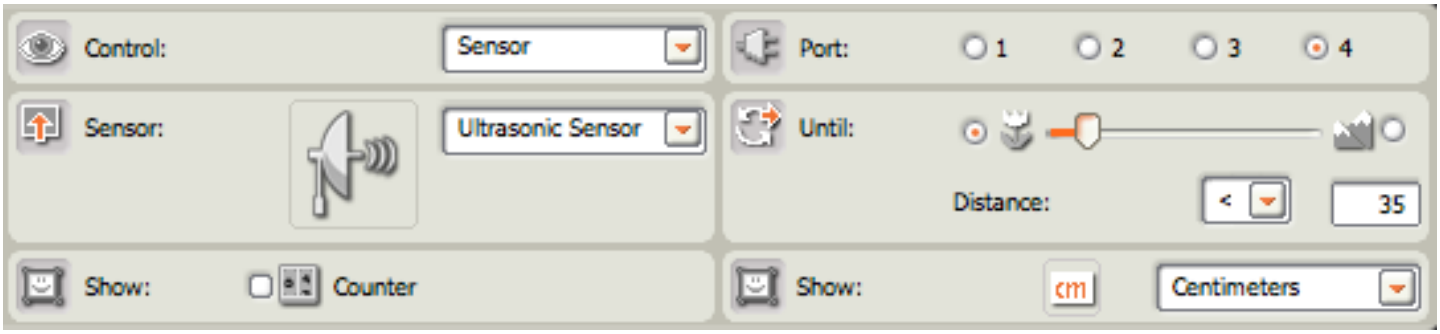
Loop di tipo Sensor con sensore touch

Questo programma ripete l'istruzione *Sound Block* fino a quando il sensore tattile collegato all'ingresso numero uno viene premuto:

Loop di tipo Sensor con sensore ultrasuoni

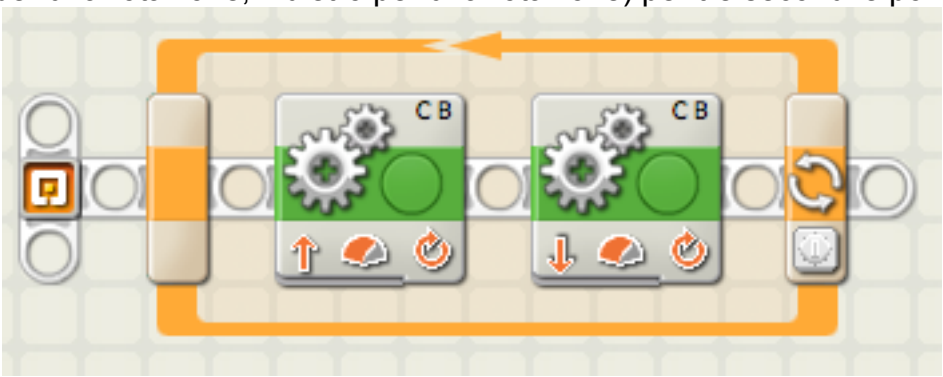
In questo esempio il loop è governato dal sensore ad ultrasuoni. Quando la distanza misurata scende al di sotto di trentacinque centimetri la ripetizione termina:

Configurazione del *loop*:

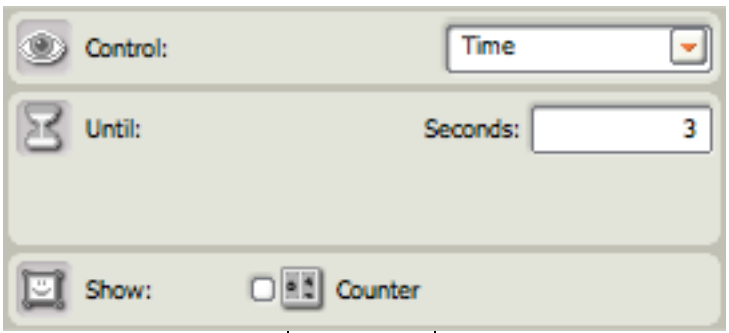


Loop di tipo Time

Questo esempio utilizza un *loop* controllato dal tempo (in modo *Time*). Il programma ripete due istruzioni (avanti per una rotazione, indietro per una rotazione) per tre secondi e poi termina:

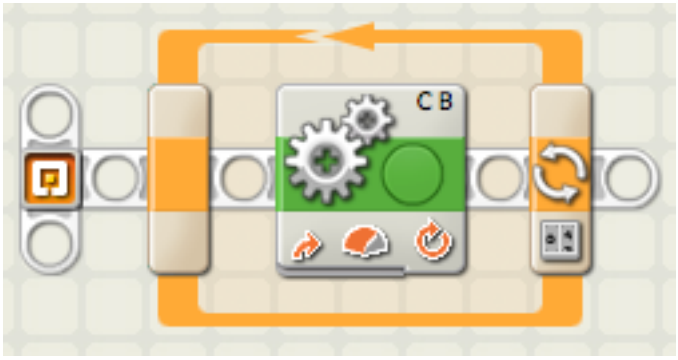


Configurazione del *loop*:

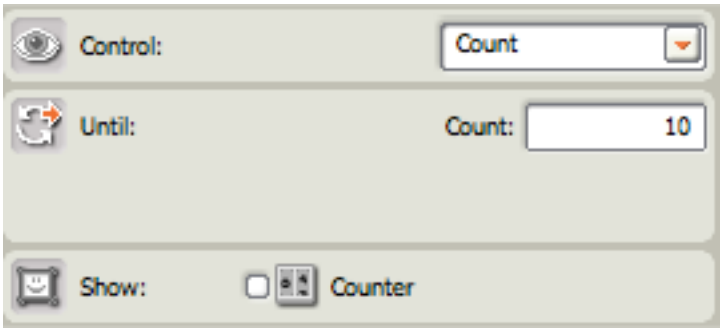


Loop di tipo Count

Nei *loop* di tipo *count* la ripetizione avviene un numero finito di volte. In questo esempio le istruzioni contenute nel ciclo vengono ripetute 10 volte:

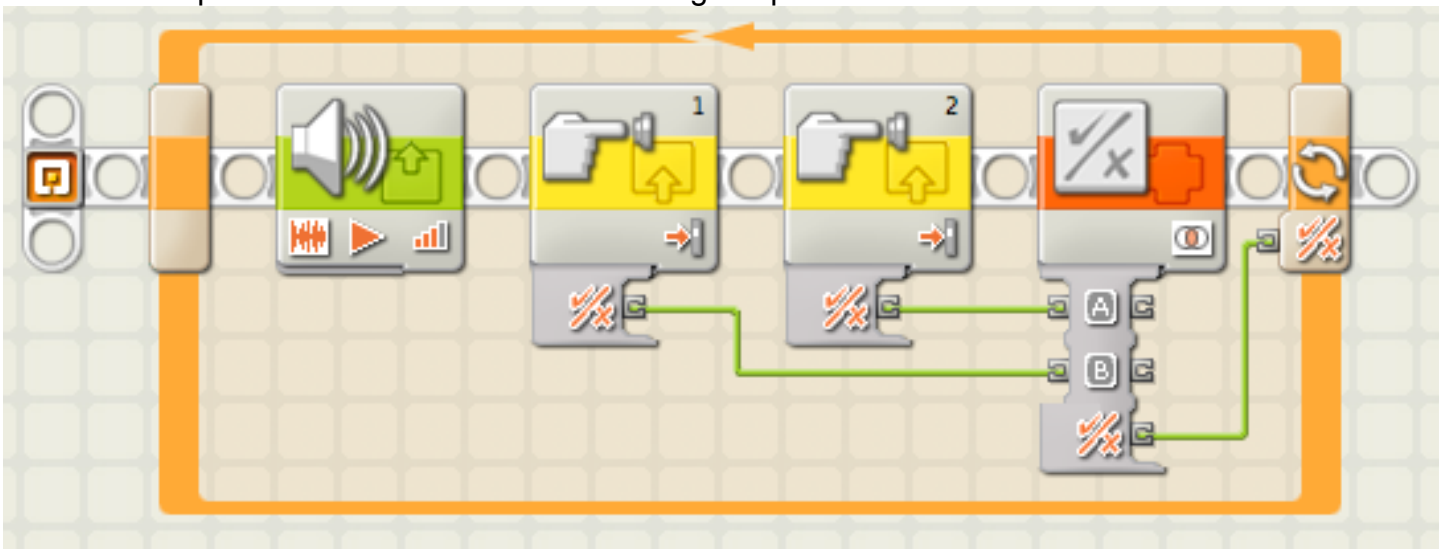


Configurazione del *loop*:

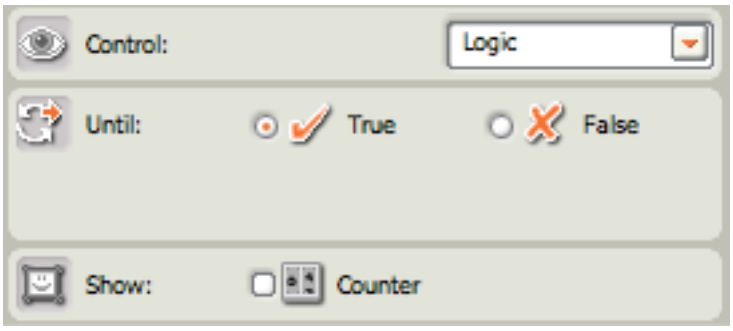


Loop di tipo Logic

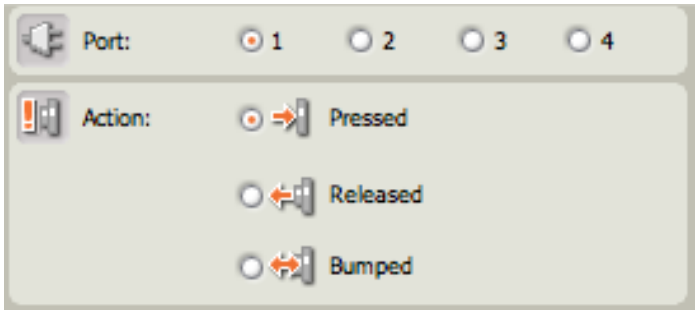
Questo tipo di loop è governato da un valore logico fornito tramite un *data wire*. In questo esempio il ciclo termina quando entrambi i sensori tattili vengono premuti:

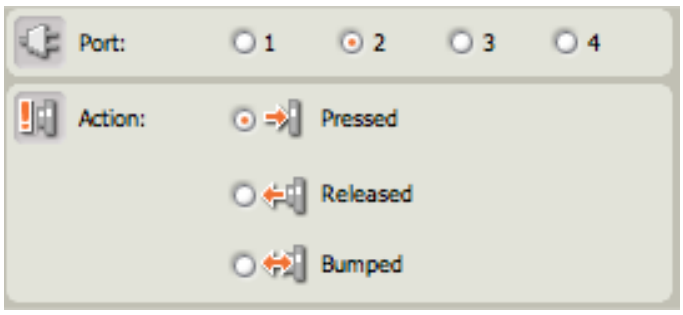


Configurazione del blocco *loop*:



Configurazione dei due blocchi *Touch Sensor*:



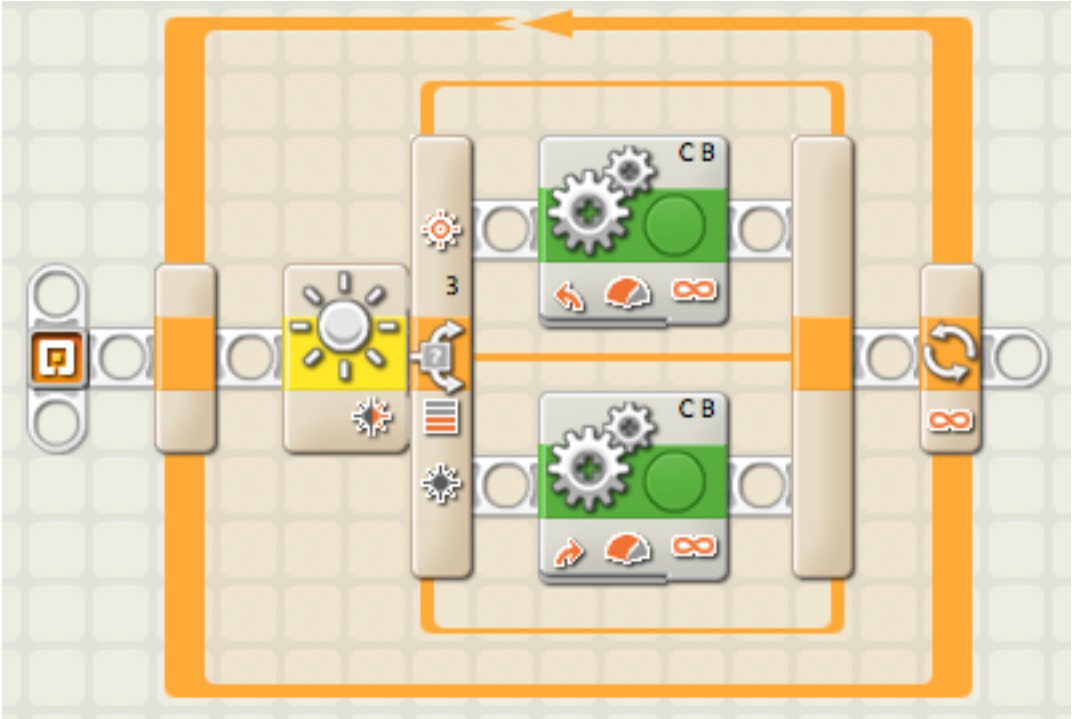


Configurazione del blocco *Logic*:

Switch

Switch di tipo Sensor con sensore di luce

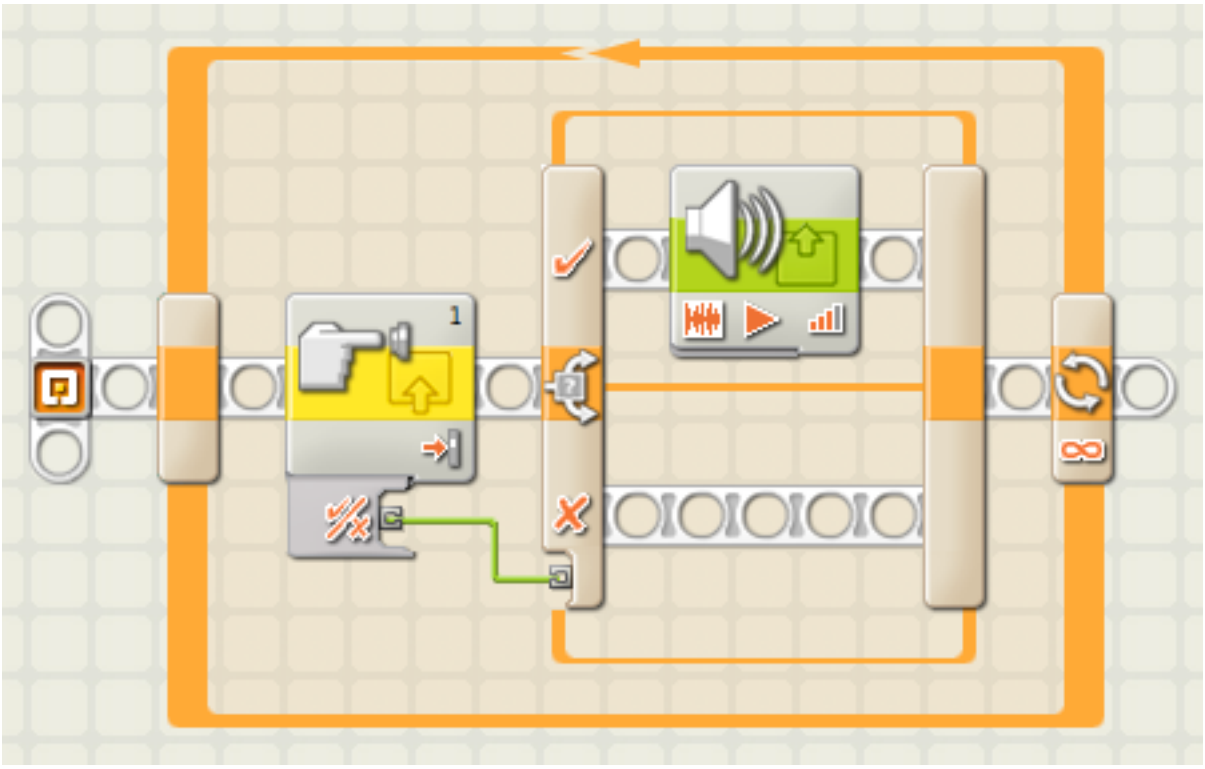
Questo programma contiene uno *switch* basato sul sensore di luce. Se il sensore collegato alla porta numero tre legge un valore superiore al cinquanta per cento il robot gira a sinistra, altrimenti gira a destra:



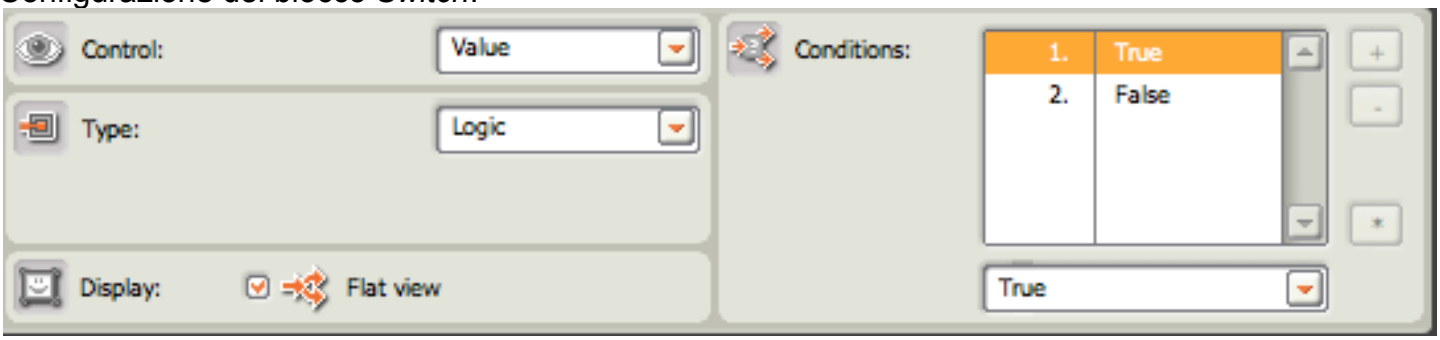
Configurazione del blocco *switch*:

Switch di tipo Logic

Questo esempio illustra l'utilizzo di un blocco *Switch* in modalità *Logic*. Se il sensore tattile è premuto, viene prodotto un suono, altrimenti non succede niente.

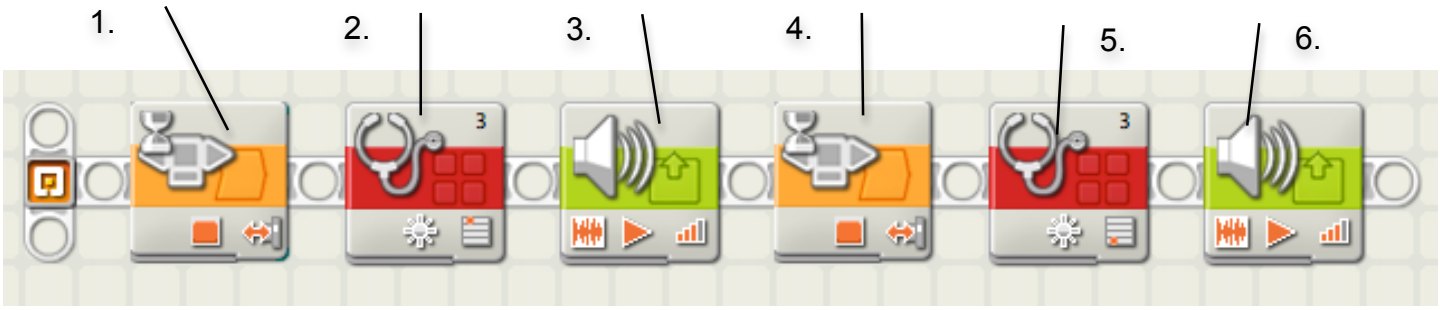


Configurazione del blocco *Switch*:



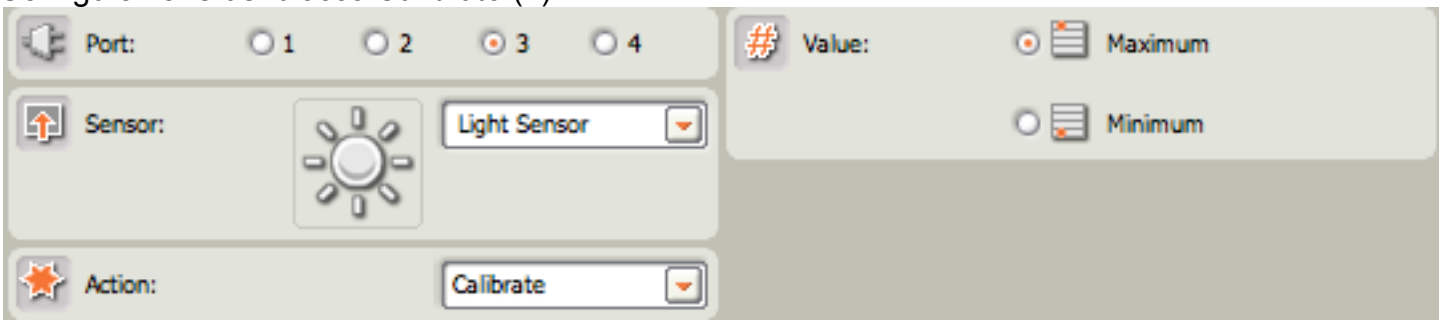
Calibrazione di un sensore

Questo esempio mostra come calibrare un sensore di luce tramite due blocchi *Calibrate*:

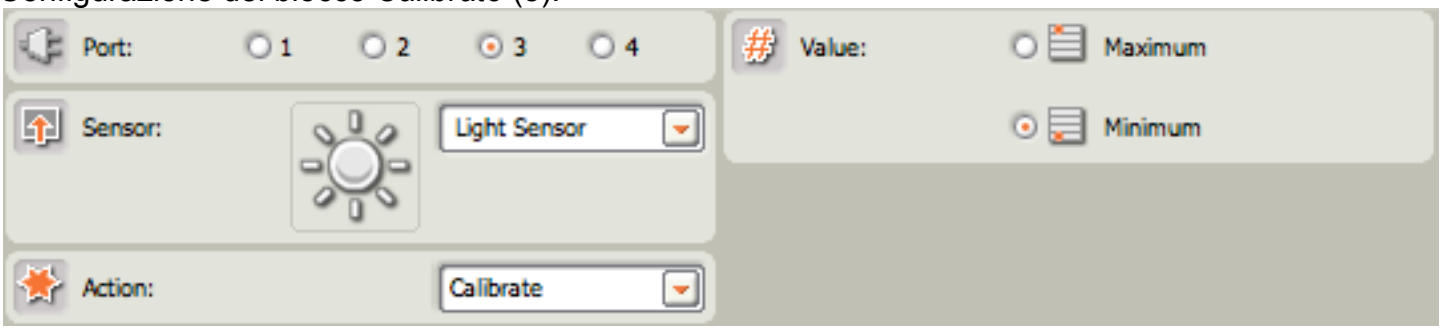


1. Attende che l'utente ponga il sensore di fronte alla superficie più chiara e poi prema il tasto arancione sul NXT.
2. Calibra il valore massimo del sensore.
3. Emette un suono per confermare l'avvenuta calibrazione.
4. Attende che l'utente ponga il sensore di fronte alla superficie più scura e poi prema il tasto arancione sul NXT.
5. Calibra il valore minimo del sensore.
6. Emette un suono per confermare l'avvenuta calibrazione.

Configurazione del blocco *Calibrate* (2):



Configurazione del blocco *Calibrate* (5):



Appendice B: Calibrazione dei sensori

Le condizioni ambientali possono influenzare le prestazioni dei sensori di luce e dei sensori sonori. Ad esempio in un locale molto luminoso un sensore di luce non calibrato fornisce valori più alti di quelli forniti in un locale buio. È perciò importante calibrare i sensori in modo adeguato all'ambiente per ottenere le migliori prestazioni possibili da essi.

Ci sono due possibili modi di calibrare un sensore; il primo consiste nell'utilizzo della funzione *Calibrate Sensors* che si trova nel menu *Tools* dell'ambiente di sviluppo. Utilizzando questa funzionalità è possibile calibrare i sensori una sola volta, la calibrazione verrà applicata ogni volta che viene eseguito un programma.

Il secondo sistema di calibrare un sensore è attraverso l'utilizzo del blocco *Calibrate*, in questo modo la calibrazione viene effettuata ogni volta che si esegue programma.

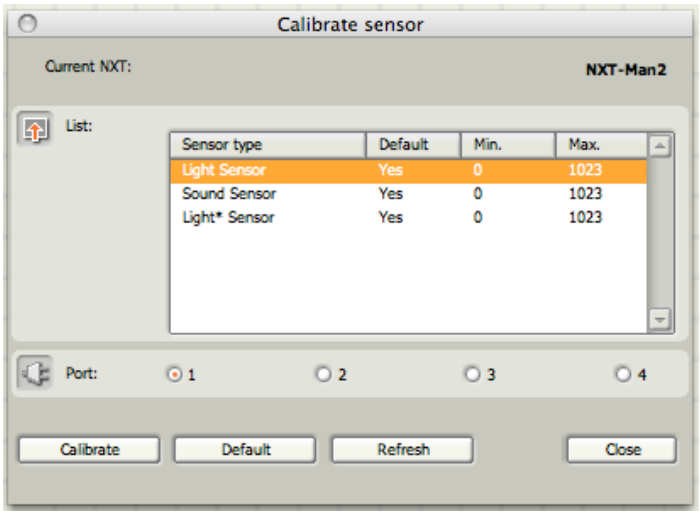
Nota 1: I valori di calibrazione registrati tramite la funzione *Calibrate Sensors* possono essere sovrascritti da quelli registrati dal blocco *Calibrate* e viceversa. La calibrazione più recente è quella che fa stato e resta effettiva fino a quando non viene sovrascritta o cancellata.

Nota 2: La calibrazione di un sensore resta valida indipendentemente dal numero della porta cui esso è connesso. Se un sensore viene calibrato quando è connesso ad una porta, la sua calibrazione verrà applicata anche quando il sensore é connesso ad una porta differente.

Utilizzo della funzione Calibrate Sensors

Per utilizzare questa funzione il NXT deve essere acceso e connesso al computer:

- Connettere il sensore che si intende calibrare deve essere connesso al NXT prestando attenzione al numero della porta di ingresso utilizzata.
- Nel menu *Tools* dell'ambiente di sviluppo selezionare la voce *Calibrate Sensors* per aprire il pannello di calibrazione (se non vi sono NXT connessi al computer la finestra compare disattivata):



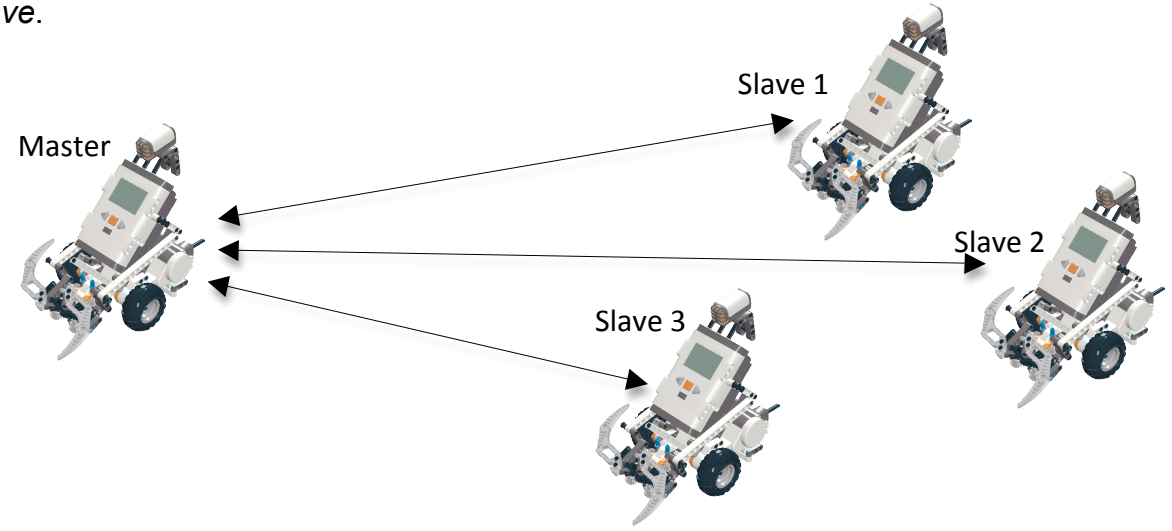
Calibrazione di un sensore di luce

Per calibrare un sensore ottico:

- Selezionare il sensore *Light Sensor* nella lista e il numero di porta cui è connesso. Poi cliccare il bottone *Calibrate*. A questo punto il programma di calibrazione viene caricato sul NXT e eseguito automaticamente.
- Sullo schermo dell'NXT compare il testo "Min Value:". Puntare il sensore verso l'oggetto più scuro, quello che il sensore dovrebbe vedere come *nero*. Premere il tasto color arancio sul NXT.
- Sullo schermo dell'NXT compare il testo "Max Value:". Puntare il sensore verso l'oggetto più chiaro, quello che il sensore dovrebbe vedere come *bianco*. Premere il tasto color arancio sul NXT.
- La calibrazione è terminata.

Appendice C: Comunicazione Bluetooth

Un NXT, che assume il ruolo di *master*, può comunicare con altri NXT (massimo tre) che assumono il ruolo di *slave*.



Configurazione della comunicazione Bluetooth

Nel menu principale del NXT *master* selezionare la voce “Bluetooth” e scegliere “Search” per attivare la ricerca di altri dispositivi (che devono essere accesi e avere il Bluetooth attivato). Finita la ricerca, scegliere quale numero di connessione attribuire al NXT *slave* (1, 2 o 3). La prima volta che avviene una connessione tra due NXT, sullo schermo di entrambi i dispositivi compare la richiesta di introdurre una password. Una volta introdotta la password, la connessione è impostata. Ora selezionando la voce “Connections” sul NXT *master* dovrebbe comparire il nome dello *slave* e il relativo numero di connessione. A questo punto l’NXT *master* è pronto per iniziare la comunicazione con lo *slave*. Se il *master* deve comunicare con più di uno *slave* questa procedura deve essere ripetuta.

La configurazione della comunicazione avviene solo sul NXT *master*. I dispositivi *slave* aggiornano automaticamente la propria configurazione quando il *master* assegna loro il numero di connessione.

Se un dispositivo deve comunicare con più di uno *slave* è necessario introdurre una pausa di almeno un secondo nei programmi per dare tempo alla radio Bluetooth di cambiare canale prima di spedire i messaggi.

Appendice D: Elenco attività

In questa sezione trovate una lista di attività:

1. Alla scoperta del kit Mindstorm.
2. Inventario pezzi e disposizione scatola.
3. Alla scoperta del NXT (accendere, inserire le batterie…).
4. Il firmware del NXT.
5. Try Me.
6. View.
7. Costruire il modello base.
8. Riflessioni su funzioni sincrone e asincrone.
9. Robot con sensore sonoro e relativa riflessione.
10. Introduzione all'ambiente di sviluppo.
11. Installazione Mindstorms.
12. Collegare il robot e trasferire il primo software.
13. Aggiornare il firmware (SOLO DOCENTI).
14. Lego digital builder & Co.
15. Robot semplificato, un possibile esempio.
16. Programmazione di base, qualche esempio:
    a. Battimano.
    b. LineFollower.
    c. Linea start stop.
    d. LineFollower avanzato:
        i. Detezione ostacolo.
        ii. Gira.
        iii. Calibrazione sensore ottico.
        iv. Verde.
17. Rappresentazione di algoritmi:
    a. Diagrammi di flusso e nassi.
        i. Ricetta.
        ii. Termostato.
        iii. …
18. Analisi:
    a. Top down.
        i. Frittata.
        ii. …
19. Data wire e tipi di dati:
    a. Stampa a display di un valore numerico.
    b. Stampa a display di un valore logico.
    c. Stampa a display di una stringa di testo.
    d. Ev. rivedere esercizio "Misuratore di conduttività".
    e. Concatenazione e manipolazione di stringhe.
20. Variabili:
    a. Contatore di click.
    b. Delta.
    c. Media.
    d. Gioco delle caselline.