Sensori

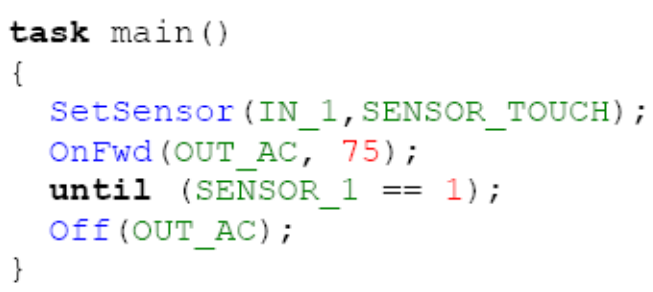
Possiamo connettere dei sensori all' NXT per consentirgli di agire in base ad informazioni provenienti dall' esterno prima che io vi spieghi come, dovete modificare il robot, aggiungendo un sensore di tatto.

Come prima, fare riferimento al manuale di istruzione per modificare il vostro tribot aggiungendo un sensore frontale.

In attesa di un valore dal sensore

Partiamo con un esempio elementare, nel quale il robot si muove diritto in avanti, finche non si imbatte in qualcosa.

Testo copiabile	Screenshot del BricxCC

task main()
{
SetSensor(IN_1,SENSOR_TOUCH);
OnFwd(OUT_AC, 75);
until (SENSOR_1 == 1);
Off(OUT_AC);
}
	

Ci sono due righe importanti in questo codice: La prima riga del programma dice al robot che tipo di sensore è connesso.

IN_1 è il numero della porta cui è connesso il sensore. Le altre porte di ingresso sono ovviamente chiamate IN_2, IN_3 e IN_4. SENSOR_TOUCH indica che alla porta IN_1 è connesso un sensore di tocco.Se avessimo connesso un sensore di luce avremmo dovuto indicare SENSOR_LIGHT.Dopo aver specificato il tipo di sensore, il programma avvia entrambi i motori ed il robot inizia a mouversi in avanti. La prossima istruzione è davvero utlile: attende finchè la condizione specificata tra parentesi si avvera.

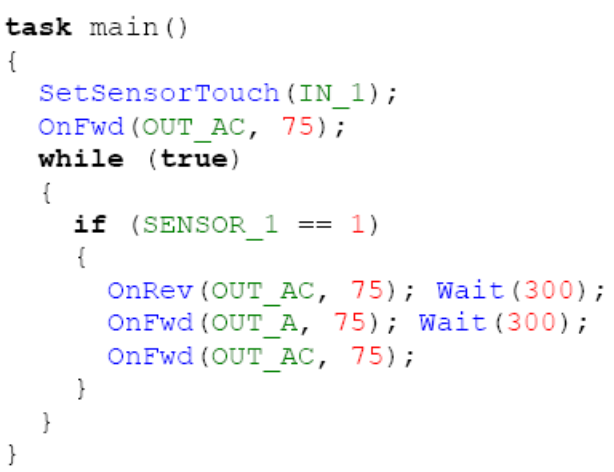
La condizione specificata in questo caso è ce il valore letto dal sensore SENSOR_1 deve valere 1, e cioè che il sensore di tocco sia stato premuto. Se il sensore non è premuto, il valore letto sarà 0. Dunque, westa istruzione aspetta fino a quando il sensore di tocco viene premuto;se il sensore è premuto i motori sono spenti ed il task termina.

Utilizzare un sensore di contatto

Vediamo adesso come si può fare per fare evitare al robot un ostacolo. Quando il robot urta un ostacolo, lo

faremo indietreggiare un po' poi girare, e infine continuare il suo percorso. Ecco il programma:

Testo copiabile	Screenshot del BricxCC

task main()
{
SetSensorTouch(IN_1);
OnFwd(OUT_AC, 75);
while (true)
{
if (SENSOR_1 == 1)
{
OnRev(OUT_AC, 75);
Wait(300);
OnFwd(OUT_A, 75);
Wait(300);
OnFwd(OUT_AC, 75);
}
}
}
	

Come nell' esempio precedente, per prima cosa indichiamo il tipo di sensore connesso. Quindi, il robot parte diritto in avanti. Durante il ciclo infinito, noi controllando di continuo se il sensore è stato premuto. Se ciò accade, il robot si muove all' indietro per 300 millisecondi,gira a destra per 300 millisecondi, e ricomincia ad avanzare diritto.

Sensore di luce

Oltre al sensore di tocco, l'NXT può essere collegato anche a dei sensori di luce, di suono, oppure a dei sensori di ultrasuoni. Il sensore di luce può essere regolato in modo da emettere egli stesso luce, oppure no, in modo da misurare la luce riflessa da una superficie oppure la luce ambientale.

Utilizzare il sensore per misurare la luce riflessa è molto utile quando vogliamo fargli seguire una linea tracciata sul pavimento.

E' quello che faremo nell'esempio seguente. (Per poter continuare a sperimentare, finiamo la costruzione del Tribot.)

Connettiamo il sensore di luce alla porta 3, il sensore sonoro alla 2, ed il sensore di ultrasuoni alla 4.

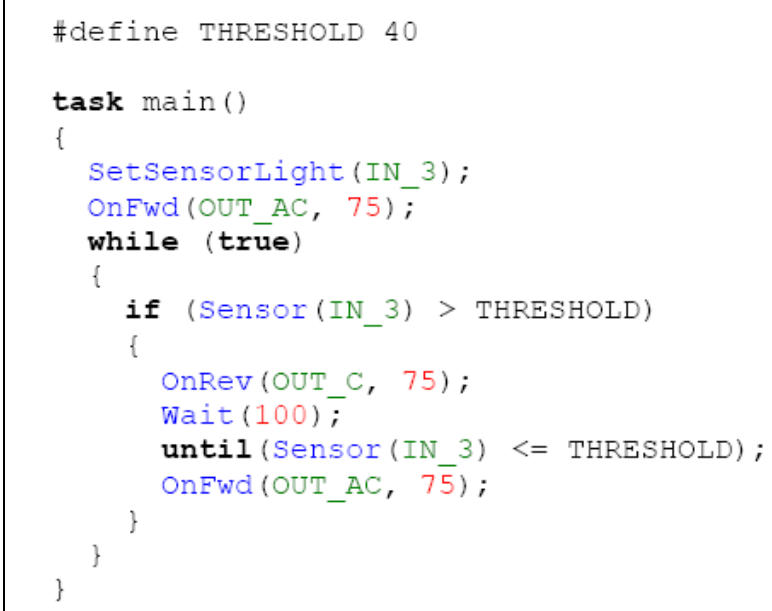
Abbiamo anche bisogno di una superficie chiara, su cui è riportata una traccia scura, che il robot possa riconoscere e seguire.Possiamo utilizzare quella in dotazione all' NXT oppure autocostruirla. Il principio di

funzionamento della routine utilizzata per far seguire la linea scura al robot è il seguente: il robot cerca di rimanere col sensore sul bordo della linea scura. In questa zona la luce riflessa ha un valore medio: metà della luce prodotta dal sensore si riflette sulla superficie chiara esterna alla traccia, l'altra metà sulla superficie scura della traccia stessa.

Ogni variazione di traiettoria rispetto alla direzione corretta da seguire comporterà un aumento di luminosità (se il sensore va a finire completamente sulla parte chiara esterna alla linea), oppure una diminuzione di luminosità (se il sensore si trova completamente sopra al nero).

Qui sotto c'è un esempio di un semplice programma in grado di seguire una linea tramite un valore di soglia di luminosità.

Testo copiabile	Screenshot del BricxCC

#define THRESHOLD 40
task main()
{
SetSensorLight(IN_3);
OnFwd(OUT_AC, 75);
while (true)
{
if (Sensor(IN_3) > THRESHOLD)
{
OnRev(OUT_C, 75);
Wait(100);
until(Sensor(IN_3) <= THRESHOLD);
OnFwd(OUT_AC, 75);
}
}
}
	

La prima cosa che fa il programma è dire all' NXT che sulla porta 3 c'è un sensore di luce.Poi, ordina al robot di avanzare ed entra in un ciclo infinito.

Quando il valore di luce letta dal sensore è maggiore di 40 (usiamo una costante, in questo caso, perché il parametro è adattabile: dipende dalle condizioni di luce ambientale che possono cambiare) uno dei motori viene mosso al contrario, fino a quando il sensore sarà riposizionato sulla traccia ed il valore della luce letta sarà corretto.

Vi accorgerete che l'andatura del robot non sarà molto scorrevole, quindi aggiungete l'istruzione Wait (100) prima del comando UNTIL , per rendere l'avanzamento del robot un po' migliore.

Il programma funziona, ma non per movimenti in senso antiorario. Per renderlo universale abbiamo bisogno di complicarlo un pochino.

Per leggere la luce ambientale, con il led spento, configuriamo il sensore nel modo seguente:

Testo copiabile	Screenshot del BricxCC

SetSensorType(IN_3,IN_TYPE_LIGHT_INACTIVE);
SetSensorMode(IN_3,IN_MODE_PCTFULLSCALE);
ResetSensor(IN_3);
	`SetSensorType(IN_3,IN_TYPE_LIGHT_INACTIVE);`
	`SetSensorMode(IN_3,IN_MODE_PCTFULLSCALE);`
	`ResetSensor(IN_3);`

Sensore di suono

Usando il sensore di suono, potrete trasformare il vostro costoso robot NXT in un interruttore sonoro :)

Scriveremo un programma che aspetta un forte suono, e fa avanzare il robot fino a quando esso udrà un altro suono. Collegate il sensore di suono alla porta di ingresso N° 2, come specificato nelle istruzioni di montaggio del Tribot

Testo copiabile	Screenshot del BricxCC

#define THRESHOLD 40
#define MIC SENSOR_2
task main()
{
SetSensorSound(IN_2);
while(true)
{
until(MIC > THRESHOLD);
OnFwd(OUT_AC, 75);
Wait(300);
until(MIC > THRESHOLD);
Off(OUT_AC);
Wait(300);
}
}
	`#define THRESHOLD 40`
	`#define MIC SENSOR_2`
	
	`task main()`
	`{`
	`  SetSensorSound(IN_2);`
	`  while(true){`
	`  until(MIC > THRESHOLD);`
	`  OnFwd(OUT_AC, 75);`
	`  Wait(300);`
	`  until(MIC > THRESHOLD);`
	`  Off(OUT_AC);`
	`  Wait(300);`
	`  }`
	`}`

Per prima cosa definiremo una costante di soglia (Treshold) ed un alias per SENSOR_2.Nel Main Task configuriamo la porta 2 per leggere il sensore di suono, e facciamo poi partire un ciclo infinito. Usando l'istruzione **until**, il programma aspetta che il livello sonoro diventi maggiore della soglia che abbiamo scelto.

Fate caso che SENSOR_2 non è semplicemente un nome, ma è una macro, che quando è chiamata restituisce il valore sonoro letto dal sensore. Se viene captato un forte suono il robot parte diritto, fin quando un altro suonoforte lo fermerà.

E' stata inserita un' istruzione **wait** , affinchè il robot non parta e si fermi istantaneamente. In effetti,in asenza di questa istruzione l' NXT è così veloce nel leggere il sensore, che un singolo suono anche di durata brevissima viene letto più volte dal sensore, col risultato di attivare e subito disattivare il movimento del robot. Provate a commentare e decommentare la prima e la seconda istruzione **wait**, e potrete capire meglio il concetto.

Una alternativa all' uso dell' istruzione **until** per bloccare l'esecuzione del codice fino a quando una certa condizione è verificata, è l'uso dell' istruzione **while**.

Usando **while**, è necessario porre tra parentesi la condizione che si deve verificare per uscire dal ciclo. ES: **while** (MIC <= THRESHOLD).

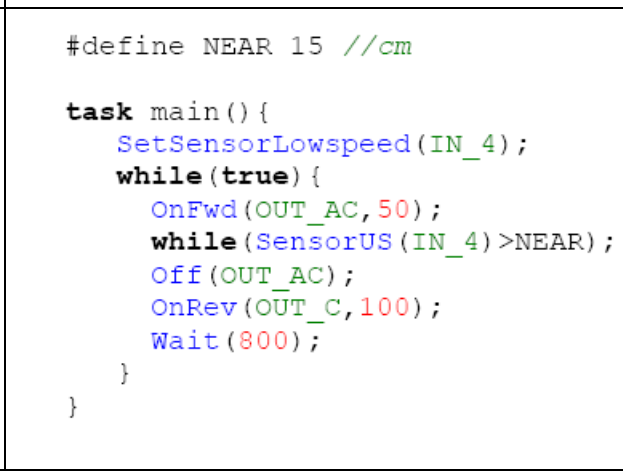
Non c'è molto altro da sapere sui sensori analogici dell' NXT; ricordate solamente che sia il sensore di luce che di suono restituiscono valori che sono compresi tra lo 0(zero) ed il 100(cento).

Sensore di Ultrasuoni

Il sensore di ultrasuoni lavora come un radar. Per capirci, esso invia dei segnali ultrasonici ed attende che tornino riflessi dall'oggetto di cui vuole misurare la distanza. Il tempo intercorso tra la partenza ed il ritorno dei segnali,visto che la velocità del suono in aria è nota , darà una misura abbastanza corretta della distanza dell'oggetto.

Questo tipo di sensore è un sensore digitale, intendendo con ciò che contiene in sé un circuito autonomo in grado di calcolare le distanze in base al tempo impiegato dal segnale sonoro ad andare e tornare.(cioè ha un "cervello" proprio, in grado di preelaborare i dati raccolti prima di trasferirli all' NXT).Con questo sensore possiamo realizzare un robot che riveli ed eviti ostacoli senza toccarli, come accade invece col sensore di tatto.

Testo copiabile	Screenshot del BricxCC

#define NEAR 15 //cm
task main()
{
SetSensorLowspeed(IN_4);
while(true)
{
OnFwd(OUT_AC,50);
while(SensorUS(IN_4)>NEAR);
Off(OUT_AC);
OnRev(OUT_C,100);
Wait(800);
}
	

}	

Gestione avanzata dei sensori

Abbiamo analizzato la gestione di base dei sensori. Ma coi sensori si può fare ben di più.

In questo capitolo discuteremo della differenza tra **sensor mode** e **sensor type**, vedremo come utilizzare i vecchi (e compatibili) sensori RCX, utlilizzandoli in abbinamento ai cavi di conversione per l' NXT.

Sensor Type

Il comando **SetSensor()** che conosciamo già , fa due cose: seleziona il tipo di sensore, e seleziona il modo in cui il sensore opera.

Selezionando però separatamente il tipo di sensore ed il modo in cui esso opera,possiamo controllare il comportamento del sensore con maggiore precisione, il che, per particolari operazioni, è davvero importante.

Il tipo di sensore viene selezionato tramite il comando **SetSensorType()**.Ci sono un sacco di tipi di sensore, ma qui riportiamo i più comunemente usati:

- **SENSOR_TYPE_TOUCH**, è il sensore di tocco,
-
- **SENSOR_TYPE_LIGHT_ACTIVE**, sensore di luce (con led attivo),
-
- **SENSOR_TYPE_SOUND_DB**, sensore sonoro,
-
- **SENSOR_TYPE_LOWSPEED_9V**, sensore di ultrasuoni.

Selezionare il giusto tipo di sensore è importante per indicare se il sensore ha bisogno di una sorgente di alimentazione elettrica (ad esempio per accendere il led del sensore di luce), o per indicare all' NXT che il sensore è digitale e necessita di essere letto tramite il bus seriale I2C.

E' possibile usare i vecchi sensori dell' RCX connessi all' NXT:

- **SENSOR_TYPE_TEMPERATURE**, sensore di temperatura,
- **SENSOR_TYPE_LIGHT** sensore di luce,
- **SENSOR_TYPE_ROTATION** peril sensore di rotazione dell' RCX (di quest' ultimo sensore parleremo più avanti).

Sensor Mode

Il modo di utilizzo dei sensori è gestito dal comando **SetSensorMode()**. Ci sono 8 differenti modalità.

La più importante è

- **SENSOR_MODE_RAW**. In questa modalità il valore restituito dal sensore

sarà un numero compreso tra 0 e 1023. E' il valore "grezzo" restituito dal sensore. L'interpretazione del significato del valore "grezzo" restituito dipende dal tipo di sensore. Ad esempio: per il sensore di tatto, quando non c'è pressione alcuna il valore restituito è 1023. Quando il sensore è premuto del tutto il valore restituito è circa 50. Quando il sensore è parzialmente premuto, esso restiturà un numero compreso tra 1000 e 50. Quindi, se selezioniamo il modo **RAW** per un sensore di tocco, siamo adesso in grado di capire se il sensore è stato premuto parzialmente.

Se,invece, il sensore applicato è un sensore di luce, il valore restituito in modalità **RAW** varia da 300 (molta luce) ad 800 (buio). Questo consente una misura molto più precisa , rispetto all' uso del comando **SetSensor()**.

Per ulteriori dettagli fate riferimento alla guida alla programmazione dell' NXT.

La seconda modalità di utilizzo dei sensori è

- **SENSOR_MODE_BOOL**. In questa modalità il vaolre restituito dai sensori è 0 oppure 1.

Se il valore RAW fosse superiore a 562, il valore restituito in modalità BOOL è 0, altrimenti il valore è 1.

- **SENSOR_MODE_BOOL** è la modalità di utilizzo prestabilita per il sensore di tocco, ma può essere utilizzato anche per altri sensori,se non ci interesa prendere in considerazione i valori analogici restituiti ma se invece ci interessa il raggiungimento di un valore di soglia.

Le modalità

- **SENSOR_MODE_CELSIUS** e
-
- **SENSOR_MODE_FAHRENHEIT** sono utilizzate per il sensore di temperatura, solamente per indicare al sensore di fornire le temperature in gradi Celsius o Fahrenheit.
-
- **SENSOR_MODE_PERCENT** converte il valore RAW in un valore percentuale da 0 a 100.E' il valore di default per il sensore di luminosità.
-
- **SENSOR_MODE_ROTATION** è una modalità adoperata per leggere i sensori di rotazione (vedi sotto).
-

Altre due interessanti modalità:

- **SENSOR_MODE_EDGE** e

- **SENSOR_MODE_PULSE**.

Queste modalità contano le transizioni, e cioè il cambiamento di stato di un sensore. Per esempio, se premiamo un sensore di tocco, il suo stato **RAW** passerà da alto a basso. Se lo rilasceremo, cambierà il suo output **RAW** in senso chiaramente opposto. Settando il sensore in modalità**SENSOR_MODE_PULSE**,verranno registrati solo i cambiamenti di fare da basso ad alto, per cui, ad esempio, ogni pressione e conseguente rilascio del sensore di tocco produrrà un cambiamento che verrà conteggiato una sola volta.

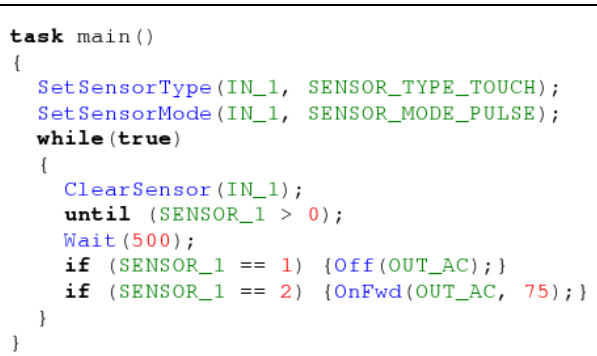
SENSOR_MODE_EDGE, viceversa, farà in modo che ogni variazione di stato venga conteggiata per cui, permendo e rilasciando il sensore di tocco avremo due cambiamenti entrambi registrati dal sensore. Sensori utilizzati in queste modalità possono ad esempio servire per capire quanto spesso viene premuto un sensore di tocco, oppure, tramite un sensore di luce, quando viene accesa e spenta una (intensa) lampada.

Ovviamente, quando conteggiate un numero di accensioni o spegnimenti, o un qualsiasi evento, dovete essere in grado di azzerare il contatore.

Per fare ciò esiste il comando **ClearSensor()**,che resetta a zero il valore del contatore del sensore specificato.

Vediamo ora un esempio. Il programma che segue usa un sensore per guidare il robot. Connettete il sensore di tocco tramite un lungo cavo alla porta di ingraesso n°1. Se toccate il sensore rapidamente per due volte consecutive il robot partirà in avanti, se lo toccherete una volta sola cesserà di muoversi. Notate: prima settiamo il tipo di sensore, e poi la modalità di utilizzo. Questo sembra essere essenziale, in quanto il cambio del tipo di sensore influenza anche le sue modalità di utilizzo.

Testo copiabile	Screenshot del BricxCC

task main()
{
SetSensorType(IN_1,
SENSOR_TYPE_TOUCH);
SetSensorMode(IN_1,
SENSOR_MODE_PULSE);
while(true)
{
ClearSensor(IN_1);
until (SENSOR_1 > 0);
Wait(500);
if (SENSOR_1 == 1) {Off(OUT_AC);}
if (SENSOR_1 == 2) {OnFwd(OUT_AC,
75);}
}
}
	
	`task main()`
	`{`
	`  SetSensorType(IN_1, SENSOR_TYPE_TOUCH);`
	`  SetSensorMode(IN_1, SENSOR_MODE_PULSE);`
	`  while(true)`
	`  {`
	`  ClearSensor(IN_1);`
	`  until (SENSOR_1 > 0);`
	`  Wait(500);`
	`  if (SENSOR_1 == 1) {Off(OUT_AC);}`
	`  if (SENSOR_1 == 2) {OnFwd(OUT_AC, 75);}`
	`  }`
	`}`

Il Sensore di rotazione NXT

Ogni motore della Lego NXT ha al suo interno un ENCODER che può essere utiizzato per programmare il movimento del robot in maniera molto precisa. L'encoder ha una risoluzione di 360 conteggi per rotazione oppure un impulso per ogni grado . Ogni rotazione della ruota corrisponde a 360 conteggi .
che cos'è un encoder?

A encoder is a measurement device which converts mechanical motion into electronic signals. The encoder is connected to the motors and outputs digital pulses which are converted by the NXT into usable information for programming our robots.

```
task main()
{
 ResetRotationCount(OUT_A);//reset the value of encoder
 while( MotorRotationCount(OUT_A); < 720)//while encoderB is less than 720
  {
    OnFwd(OUT_A, 30);
   }
 Off(OUT_A);
 }
```

```
// Display RPM of motor attached to the port MOTOR while running at full speed.
// The program runs continously until stopped by pressing the gray NXT button.
// Requires NXT firmware 1.28 or later (uses floating point arithmetic).
// CurrentTick returns milliseconds in a long integer.
// MotorRotationCount returns degrees in a long integer.

#define MOTOR OUT_A
#define FULL_SPEED 100
#define DEG_TO_RPM 166.6667 // converts degrees per millisecond to RPM
long prev_tick;
long prev_deg = 0;
task main()
{
prev_tick = CurrentTick();
OnFwd(MOTOR, FULL_SPEED);
while (true)
{
Wait(MS_500); // update display every 0.5 seconds
long dt = CurrentTick() - prev_tick;
long deg = MotorRotationCount(MOTOR) - prev_deg;
float rpm = deg * DEG_TO_RPM / dt;
```

```
prev_deg = MotorRotationCount(MOTOR);
prev_tick = CurrentTick();
 NumOut(0, LCD_LINE2, rpm);
   Wait(MS_500);      // update display every 0.5 seconds

}
}
```

Il sensore di rotazione (RCX)

Il sensore di rotazione è un tipo di sensore davvero utile: è un encoder ottico, più o meno lo stesso che è contenuto nei servomotori dell' NXT.

Questo sensore ha un foro, nel quale si può infilare un asse che può ruotare liberamente,la cui posizione angolare relativa viene misurata. Una rotazione completa dell' asse equivale a 16 conteggi, (o -16 se giriamo dalla parte opposta), il che significa che ad ogni passo corrisponde una rotazione di 22,5 gradi.

Molto poco rispetto alla precisione di un grado offerta dal servomotore.

Questo vecchio tipo di sensore può comunque risultare utile per contare la rotazione di un asse, senza sprecare un motore per eseguire questo compito. Considerate inoltre che utilizzare un motore come sensore di rotazione, utilizzando il suo encoder interno equivale a perdere un sacco di energia per far ruotare il tutto, mentre il vecchio sensore di rotazione è facilissimo da far ruotare.Se desiderate una precisione maggiore di un passo ogni 22,5 gradi potete sempre aumentare meccanicamente il numero di giri tramite un ingranaggio.

L'esempio seguente è preso dal vecchio tutorial dell' RCX.

Un' applicazione standard consiste nell avere due sensori di rotazione sulle due ruote di un robot, mosse ciascuna da un motore. Se volete far andare il robot diritto in avanti le due ruote si devono muovere alla stessa velocità.Sfortunatamente, i motori non sono uguali e quindi non si muovomno alla stessa velocità.Usando il sensore di rotazione possiamo vedere quale delle due ruote gira più velocemente

Possiamo a questo punto fermare (Meglio usando **Float()**) il motore più veloce, fin quando l'altro motore ha compiuto lo stesso numero di giri. Il programma seguente fa esattamente questo.Semplicemente, fa muovere il robot in linea retta. Per utilizzarlo, cambiata il vostro robot connettendo a ciascuna ruota un sensore di rotazione. Connetete i sensori alle porte 1 e 3.

Testo copiabile	Screenshot del BricxCC

```
task main()
{
SetSensor(IN_1, SENSOR_ROTATION);
ClearSensor(IN_1);
SetSensor(IN_3, SENSOR_ROTATION);
ClearSensor(IN_3);
while (true)
{
if (SENSOR_1 < SENSOR_3)
{OnFwd(OUT_A, 75); Float(OUT_C);}
else if (SENSOR_1 > SENSOR_3)
{OnFwd(OUT_C, 75); Float(OUT_A);}
else
{OnFwd(OUT_AC, 75);}
}
}
```

```
task main()
{
  SetSensor(IN_1, SENSOR_ROTATION); ClearSensor(IN_
  SetSensor(IN_3, SENSOR_ROTATION); ClearSensor(IN_
  while (true)
  {
    if (SENSOR_1 < SENSOR_3)
      {OnFwd(OUT_A, 75); Float(OUT_C);}
    else if (SENSOR_1 > SENSOR_3)
      {OnFwd(OUT_C, 75); Float(OUT_A);}
    else
      {OnFwd(OUT_AC, 75);}
  }
}
```

Il programma per prima cosa stabilisce che entrambi i sensori sono sensori di rotazione, e resetta il loro valore a zero.Poi inizia un ciclo infinito. All' interno del ciclo si controlla se la lettura di entrambi i sensori è uguale. Se la lettura è la stessa, il robot si muove in avanti in linea retta. Se invece uno dei due è maggiore, il motore corrispondente viene arrestato fin quando la lettura dei due sensori ritorna uno stasso valore. Ovviamente questo è un programma molto semplice; potete modificarlo per far percorrere al robot distanze molto precise, oppure per percorrere dei cerchi perfetti.