Corso di programmazione per le scuole con Arduino – PARTE 4

Siamo arrivati all'ultima puntata di questo corso: la volta scorsa abbiamo parlato di come gestire i suoni (tramite microfoni e buzzer). Stavolta presentiamo un singolo progetto che permette di mettere in pratica tutti i concetti di base imparati nelle puntate precedenti. Ma soprattutto, speriamo di stimolare la fantasia degli studenti con un dispositivo che può essere facilmente personalizzato e migliorato. Vedremo, infatti, come realizzare un semplice robot capace di riconoscere le linee disegnate sul pavimento o su un foglio di carta e di muoversi seguendo tali linee. Il dispositivo che progettiamo è molto semplice, ma proprio per questo ogni studente può lavorarci sopra per aggiungere nuove caratteristiche, impiegando in modo creativo le competenze che ha acquisito finora.

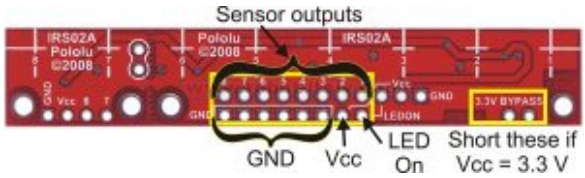
Table of Contents

7 – Un line following robot

Costruire un robot è meno complicato di quanto si possa immaginare, in fondo bastano due servomotori a rotazione continua (continuous rotation servo) e il gioco è fatto. Più

complicato può essere inventarsi un sistema per controllare lo spostamento del robot, ma esiste sempre il meccanismo del "line following", ovvero del seguire le linee. L'idea è semplice: basta disegnare sul pavimento una linea con un buon contrasto (per esempio con un pennarello nero su un foglio di carta bianco). Esiste un metodo piuttosto semplice per permettere ad una scheda Arduino di riconoscere la linea nera: dei sensori infrarossi. Banalmente, basta accoppiare un led a infrarossi con una fotoresistenza (sempre a infrarossi), montandoli sotto al robot. In questo modo, quando la luce infrarossa del led colpisce una zona bianca, la fotoresistenza riceverà un riflesso molto luminoso, mentre quando la luce del led colpisce un punto nero la fotoresistenza riceverà un riflesso molto debole o addirittura nullo (perché il bianco riflette la luce ed il nero la assorbe). Naturalmente il meccanismo funzionerebbe anche con dei normali led colorati, come quelli che abbiamo già utilizzato in precedenza. Però per non avere interferenze dovremmo far correre il robot in una stanza buia, perché la luce del Sole o delle lampade si sovraporrebbe a quella del led. Utilizzando dei led ad infrarossi il problema è risolto, e il vantaggio è che si tratta comunque di banalissimi led, che funzionano come tutti gli altri. Esistono addirittura dei set già pronti con led infrarossi e fotoresistenze, come il QTR-8A, che abbiamo preso come base per il nostro esempio. Il QTR-8A è molto semplice da utilizzare: basta fissarlo sotto al nostro robot, in mezzo ai due servomotori. Il pin Vcc va collegato al pin 5V di Arduino, mentre il pin GND va connesso al GND di Arduino. Poi sono disponibili ben otto pin di segnale: infatti il QTR-8A dispone di 8 coppie di led e fotoresistenze infrarosse, ed ogni fotoresistenza ha un pin che offre a Arduino il proprio segnale analogico, cioè la lettura della luminosità del pavimento. Però Arduino Uno ha soltanto 6 pin analogici: poco male, collegheremo ad Arduino soltanto 6 dei pin del QTR-8A. Per non fare confusione li collegheremo in ordine: il pin 1 del QTR-8A andrà connesso al pin analogico 0 di Arduino, il pin 2 del QTR-8A andrà collegato al pin analogico 1 di

Arduino, e così via. Se avete una scheda più grande, come Arduino Mega, potete utilizzare tutti i pin del QTR-8A, perché un Arduino Mega ha ben 16 pin analogici. In realtà, però, per la maggioranza delle applicazioni 6 coppie di led e fotoresistenze infrarosse sono più che sufficienti.



I contatti della scheda QTR-8A prevedono il Vcc (5V), il GND, e gli input analogici di Arduino

L’idea di base è molto semplice: abbiamo una fila di 6 sensori sotto al robot, ed in teoria vorremmo che la riga nera si trovasse sempre in corrispondenza della metà dei sensori (ovvero tra il terzo ed il quarto). Questo significa che, se facciamo una media della luminosità rilevata dai primi tre sensori ed una media di quella rilevata dagli ultimi tre sensori, è ovvio che dovrebbero essere più o meno uguali. Se la media dei primi tre sensori (che possono essere quelli di sinistra) è più alta significa che la linea nera si trova dalla loro parte, e quindi dovremo far ruotare il robot nella giusta direzione (per esempio destra) per far tornare la linea nera al centro. Infatti, i sensori del QTR-8A funzionano al contrario rispetto ad una normale fotoresistenza: offrono il valore massimo quando la luminosità ricevuta è bassa, e viceversa. Naturalmente, destra e sinistra dipendono dal verso in cui si monta il QTR-8A sul robot: se il robot non si comporta come dovrebbe, basta ruotare di 180° la scheda con tutti i sensori infrarossi.

Utilizzare un radiocomando

Un altro metodo per controllare un robot realizzato con Arduino è utilizzare un radiocomando: i radiocomandi non sono altro che un insieme di potenziometri (le varie leve presenti sul radiocomando sono potenziometri) i cui valori vengono trasmessi a distanza tramite onde radio. Quindi basta collegare il ricevitore del radiocomando ad Arduino, alimentandolo con i pin 5V e GND, e connettendo tutti i vari pin di segnale (ce n'è uno per ciascuna leva del radiocomando) ai pin analogici di Arduino. Arduino può poi leggere i valori dei potenziometri, e dunque capire se sia stata spostata una levetta in tempo reale e reagire di conseguenza (per esempio spegnendo od accendendo uno dei due servomotori).
http://www.instructables.com/id/RC-Control-and-Arduino-A-Complete-Works/?ALLSTEPS

Il codice per il line following

Il codice che fa funzionare questo programma è probabilmente il più complesso che abbiamo analizzato finora, ma è comunque abbastanza semplice da capire:

Prima di tutto si include nel programma la libreria necessaria per il funzionamento dei servomotori.

Dichiariamo poi due variabili speciali, degli "oggetti" (ne avevamo già parlato, sono variabili che possono avere proprietà e funzioni tutte loro), che rappresenteranno i due servomotori. Come abbiamo già accennato, il nostro robot ha un

totale di due ruote motrici, ciascuna mossa da un servomotore: una a destra (right) e l'altra a sinistra (left). Possiamo poi aggiungere una terza ruota centrale, per esempio una rotella delle sedie da ufficio, solo per tenere il robot in equilibrio. Una sorta di triciclo al contrario, visto che le ruote motrici sono due e non una.

Per poter eseguire i nostri calcoli, dovremo tarare i sensori: dovremo capire quale sia il valore massimo (**mx**), minimo (**mn**), e medio (**mid**) di luminosità rilevabile dai vari sensori. Quindi dichiariamo delle variabili che ci serviranno per memorizzare questi valori.

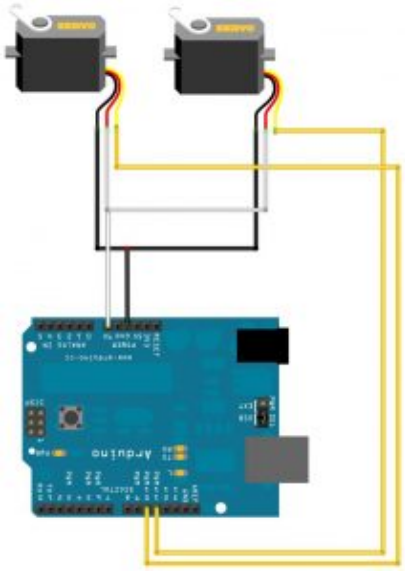
Cominciamo ora a scrivere la funzione **setup**, che viene eseguita all'avvio di Arduino.

Dobbiamo assegnare i due oggetti di tipo servo, **left** e **right**, ai pin digitali di Arduino: abbiamo deciso di collegare il pin **9** al servomotore sinistro ed il pin **10** al servomotore destro. Ricordiamo che devono essere pin PWM, indicati sulla scheda Arduino col simbolo tilde (cioè ~). Indichiamo anche i due valori di minimo e massimo per gli impulsi che faranno muovere il servomotore: di solito non è necessario specificarli (lo fa automaticamente Arduino), ma può essere utile per evitare problemi quando si usano servomotori a rotazione continua come nel nostro caso.

Attiviamo anche la comunicazione sulla porta seriale, così da poter inviare messaggi ad un computer per capire se qualcosa

non stia funzionando nel nostro robot.

Visto che siamo all'inizio, spegniamo il led collegato al pin digitale numero **13** di Arduino (è un led di segnalazione saldato sulla scheda Arduino). Spegniamo anche i due servomotori, così il robot resterà fermo. Con dei servomotori a rotazione continua, lo spegnimento si esegue dando il valore **90** alla funzione write di ciascun oggetto **left** e **right**.



Collegare due servomotori ad Arduino è molto semplice

Calibrare i sensori infrarossi

Attenderemo 5 secondi, ovvero 5000 millisecondi, eseguendo una misura della luminosità di ciascun sensore ogni millisecondo, grazie ad un semplice ciclo **for** che viene ripetuto per 5000 volte.

Accendiamo il led connesso al **pin 13**, quello saldato su Arduino, per avvisare che la calibrazione dei sensori è in atto.

Dobbiamo ora capire quali siano i valori massimo e minimo che si possano misurare con i nostri sensori: per farlo scorriamo tutti i sensori, dal pin analogico **0** al pin analogico **5** di Arduino, leggendo con **analogRead** il valore misurato al momento. Si suppone che il robot si trovi già sopra alla linea, e che almeno alcuni dei sensori siano proprio sopra alla linea nera mentre altri siano sopra al pavimento bianco. Il valore di ogni sensore viene inserito nella variabile **val**. Se tale variabile è maggiore dell'attuale valore massimo (**mx**), allora essa viene considerata il nuovo massimo, assegnando il suo valore a **mx**. Allo stesso modo, se **val** è minore del valore più basso finora registrato **mn**, alla variabile **mn** viene assegnato al valore della variabile **val**.

Prima di terminare il ciclo **for** da 0 a 5000 inseriamo la funzione **delay** chiedendole di attendere 1 millisecondo. Così siamo sicuri che il ciclo durerà in totale almeno 5000 millisecondi, ovvero 5 secondi. In realtà durerà un po' di più, perché le varie operazioni richiedono una certa quantità di tempo, ma sarà probabilmente impercettibile.

Ottenuti i valori massimi e minimi che i sensori possono fornire, possiamo calcolare il valore medio, da memorizzare nella variabile **mid**.

Per segnalare che la calibrazione dei sensori è terminata, spegniamo il led connesso al pin digitale **13** di Arduino.

Finora abbiamo solo calibrato i sensori, ma il robot è ancora fermo. Cominciamo ora la funzione di **loop**, quella che farà muovere il nostro robot.

Per cominciare leggiamo i valori di tutti i sensori, inserendoli in apposite variabili chiamate **s0**, **s1**, **s2**, eccetera.

Iniziamo a muovere il robot: per far girare un servomotore a rotazione continua si può indicare un numero da **0** a **90** oppure da **90** a **180**.
Il valore **180** rappresenta la massima velocità in una direzione, **0** rappresenta la massima velocità nell'altra direzione, e **90** rappresenta la posizione di stallo (quindi il servomotore è fermo).

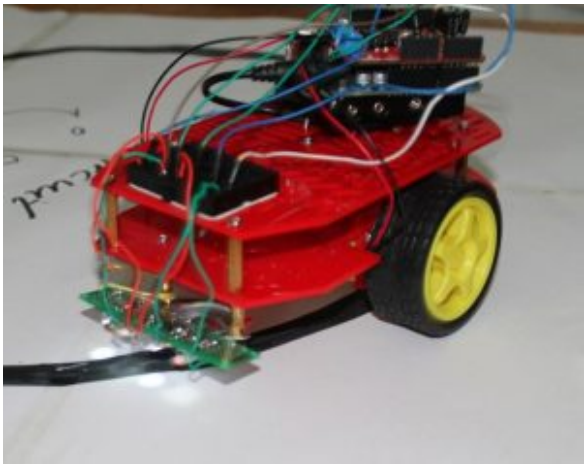
Un semplice miglioramento

Un semplice modo per migliorare il robot può consistere nel rendere variabile la velocità. Se infatti i motori sono sempre impostati a 0 o 180 il robot va sempre alla massima velocità. Si può utilizzare, per esempio, un sensore a ultrasuoni per ridurre la velocità se ci si sta avvicinando a un ostacolo, mappando (funzione **map**) la distanza in un valore da 90 a 0 e da 90 a 180.

Siccome i nostri servomotori sono montati in modo da essere uno speculare all'altro, è ovvio che per far andare il robot avanti uno dei servomotori girerà in una direzione e l'altro nell'altra, così alla fine le due ruote gireranno all'unisono.

Attendiamo un millisecondo, soltanto per essere sicuri che il comando di movimento dei servomotori sia stato applicato.

Abbiamo memorizzato nelle variabili **s0**, **s1**, eccetera, i valori dei vari sensori. Però, come abbiamo detto prima di cominciare a scrivere il programma, noi vogliamo semplicemente comparare la media dei sensori di sinistra con quella dei sensori di destra. Abbiamo deciso che i sensori di **sinistra** siano quelli che sono collegati ai pin analogici **0**, **1**, e **2** di Arduino, mentre quelli di **destra** siano i sensori connessi ai pin analogici **3**, **4**, e **5** (ovviamente dipende da come montiamo il **QTR-8A** sotto al robot). Le due medie si calcolano banalmente con la classica formula matematica: si fa la somma e si divide per 3. Ovviamente, la media potrebbe essere un numero con decimali (con la virgola), ma a noi basta un numero intero: siccome abbiamo definito le due variabili come tipo **int**, Arduino arrotonderà automaticamente i decimali al numero intero più vicino.



Il robot posizionato sopra

la linea nera disegnata su
un pavimento bianco

La linea è a destra o a sinistra?

Normalmente, il robot continua a muoversi in avanti. Però, se la media dei sensori di sinistra è maggiore di quelli dei sensori di destra significa che la linea nera sul pavimento si trova dalla parte sinistra del robot.

Abbiamo indicato anche un fattore correttivo, pari a **240**, per avere un certo lasco: se avessimo scritto soltanto **averageLeft>averageRight** il blocco **if** verrebbe eseguito anche per variazioni minime dei sensori infrarossi tra la parte destra e quella sinistra del robot. Ma vi sarà sempre qualche piccola variazione, anche solo per minime interferenze o oscillazioni nella corrente. Inserendo un fattore correttivo ci assicuriamo che la condizione di **if** venga attivata soltanto se la differenza tra la parte destra e sinistra del robot è notevole.

Ovviamente, se la linea nera è alla sinistra del robot, dovremo ruotare il robot verso sinistra in modo da riportarlo in una posizione in cui la linea nera sia esattamente al centro del robot stesso. E per far ruotare il robot verso sinistra dobbiamo tenere fermo il servomotore di sinistra, dandogli il valore **90**, di modo che faccia da perno, e muovere il servomotore di destra con un valore vicino a **180**. Scegliamo un valore inferiore a 180 perché vogliamo che il servomotore

di destra si muova ma non alla sua massima velocità, così il movimento è più lento e più facile da controllare (se si esagera si rischia di finire al di fuori della linea nera).

Prima di concludere il blocco **if** attendiamo una certa quantità di millisecondi, ottenuta come la metà della differenza delle due medie. L'idea è che in questo modo maggiore è la differenza tra i sensori di destra e quelli di sinistra, maggiore è dunque la dimensione della linea nera, e quindi maggiore è il tempo necessario durante lo spostamento del robot per riuscire ad arrivare ad avere la linea nera al centro del robot stesso.
Siccome la differenza dei due valori potrebbe essere un numero negativo, utilizziamo la funzione **abs** per ottenere il valore assoluto, cioè ottenere la differenza senza segno (in pratica, un eventuale valore **-500** diventerebbe semplicemente **500**).

Se la media dei sensori di sinistra è inferiore a quella dei sensori di destra (tenuto sempre conto del solito fattore correttivo), allora significa che la linea nera è posizionata dalla parte destra del robot. Quindi faremo esattamente l'opposto del precedente **if**: terremo fermo il servomotore destro e muoveremo quello sinistro (anche in questo caso non imposteremo la sua velocità al valore massimo, cioè **0**, ma un po' meno, cioè 40). Anche prima di concludere questo blocco if, attenderemo di nuovo una manciata di millisecondi con lo stesso calcolo precedente.

I percorsi disegnati con nastro adesivo nero su un pavimento chiaro possono anche essere molto lunghi ed elaborati

La fine del percorso

Abbiamo detto al robot cosa fare se la linea nera si trova alla destra oppure alla sinistra del robot stesso. E se invece la linea nera coprisse tutto il pavimento? Significherebbe che il percorso è terminato. Infatti, quando disegniamo il percorso sul pavimento, a meno che non sia un circuito chiuso su se stesso come quelli automobilistici, possiamo indicarne la fine dipingendo di colore nero un bel rettangolo perpendicolare all'ultima parte della linea.

In questo modo quando il robot ci arriva sopra si accorgerà che tutti i suoi sensori, in particolare il sensore **s0** ed il sensore **s5**, che sono il primo e l'ultimo, hanno un valore che è superiore alla media. Questo significa che tutti i sensori sono contemporaneamente sopra alla linea nera, e quindi si è raggiunto il termine del percorso.

In questo caso dobbiamo chiaramente fermare il robot, dando il valore **90** a entrambe i servomotori (come abbiamo già detto,

questo valore provoca l’arresto immediato dei servomotori a rotazione continua).

Per segnalare di avere raggiunto quello che riteniamo essere il termine del percorso (e che quindi il robot non si è fermato per un errore), facciamo lampeggiare rapidamente il led collegato al pin digitale **13** di Arduino, quello saldato sulla scheda. Per farlo lampeggiare 50 volte basta un ciclo **for** che si ripete per l’appunto 50 volte, e ad ogni iterazione non fa altro che accendere il led portando il suo pin al valore **HIGH**, attendere **100** millesimi di secondo, spegnere il led scrivendo il valore **LOW** sul suo pin, ed poi attendere altri **100** millisecondi prima di passare all’iterazione successiva.

Ora attendiamo 5 secondi per essere sicuri che tutte le operazioni necessarie allo spegnimento dei servomotori siano state portate a termine. Durante questi 5 secondi si può tranquillamente spostare il robot, magari posizionandolo di nuovo all’inizio del percorso per farlo ripartire.

La funzione loop si conclude qui, e con essa il programma. Naturalmente, ricordiamo che la funzione loop viene ripetuta di continuo, quindi il robot continuerà a muoversi lungo la linea nera tracciata sul pavimento bianco finché non lo fermiamo facendolo passare sopra ad un rettangolo nero largo tanto quanto l’intera scheda **QTR-8A**, in modo che tutti i sensori infrarossi si trovino contemporaneamente sopra al colore nero.

Costruire un vero Go-Kart con Arduino

Il robot che presentiamo è pensato per essere piccolo e semplice. Ma il bello di Arduino è che si può facilmente salire di scala: basta avere motori più potenti ed un buon telaio, e si può costruire un Go Kart capace di trasportare una persona, controllando il motore elettrico (ne basta uno, visto che la trazione è posteriore) con Arduino. Il progetto che vi suggeriamo utilizza un motore brushless, un Savox BSM5065 450Kv. I motori brushless sono una via di mezzo tra servomotori ed i normali motorini elettrici a spazzole. I motori brushless eseguono rotazioni continue (quindi non si fermano a 180°), e sono controllabili con Arduino (si può controllare la direzione e la velocità). Hanno anche il notevole vantaggio di poter fornire molta potenza e dare dunque una notevole velocità al mezzo su cui vengono montati. Inoltre sono quasi immuni all'usura, rispetto ai motori a spazzole. Ne esistono di piccolissimi, per progetti di modellismo, oppure di enormi (le automobili elettriche usano motori brushless).
http://www.instructables.com/id/Electric-Arduino-Go-kart/?ALLSTEPS

Il codice sorgente

Potete trovare il codice sorgente dei vari programmi presentati in questo corso di introduzione alla programmazione con Arduino nel repository:
https://www.codice-sorgente.it/cgit/arduino-a-scuola.git/tree/
Il file relativo a questo articolo è 7-robot-linefollowing.ino.