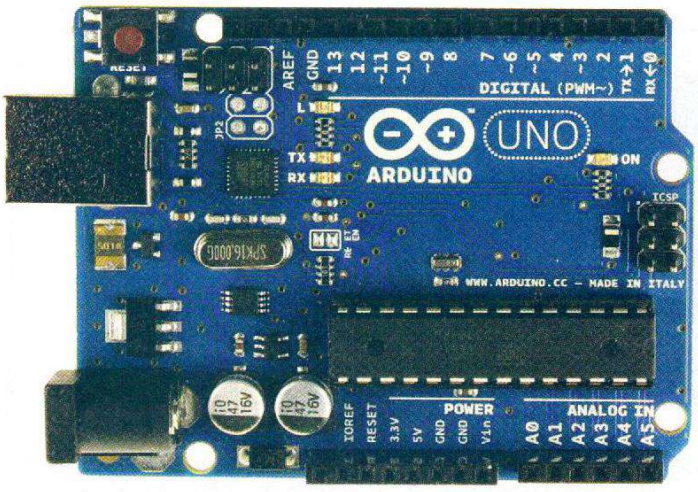
Un cingolato semovente

Nella prima parte di questo tutorial farete la conoscenza con Arduino e organizzerete il materiale per realizzare un fantastico progetto di automazione

L'autore

Silvio Umberto Zanzi
szanzi@informazione.biz

Non posso nasconderlo, sono sempre stato profondamente affascinato da coloro che per studio o lavoro operano nell'ambito dell'automazione. Saper imprimere movimento in maniera coerente e ordinata a un manufatto implica competenze di programmazione, di elettronica e di meccanica. Non è certamente qualcosa alla portata di tutti visto l'aspetto multidisciplinare di questo settore. In genere ci si specializza in un ambito molto preciso, perdendo, per scelta, la visione a più ampio raggio. Non è mai stato tra l'altro semplice sperimentare in questo campo. Non è il settore informatico, diventato così accessibile a tutti a partire dagli anni ottanta, da quando è stato possibile acquistare un computer con una modica spesa. La mia personale attività è oggi possibile proprio in virtù della facilità con cui era possibile iniziare a muovere i primi passi con i computer senza la necessità di avere alcuna attrezzatura "di contorno". Lo stesso può essere affermato per milioni di professionisti dell'IT in tutto il mondo e per centinaia di milioni di appassionati di informatica a vario titolo. L'accessibilità è inoltre progredita nel tempo.
Oggi è infatti ancora più semplice costruirsi una carriera nel settore informatico visti i bassi costi dell'hardware, le enormi potenzialità dei dispositivi oggi in commercio, la quantità illimitata di informazioni presenti su Internet e l'ottima letteratura tecnica disponibile nelle biblioteche e nelle librerie. Vi sono pochissimi settori che vantano questo orientamento "democratico". In quasi tutti i campi è necessario acquisire conoscenze dettagliate e investire in modo cospicuo prima di osservare i risultati. L'automazione è appunto uno di questi campi. Almeno lo era fino a pochi anni fa, prima che un gruppo di italiani desse l'avvio al movimento di elettronica libera



› Scheda Arduino (tratto dal sito ufficiale di Arduino). Si notino le morsettiere con i pin digitali (in alto) e quelle con i pin analogici e i pin di alimentazione (in basso)

› Il Kit motori Tamiy è disponibile presso Pololu (www.pololu.com), uno dei più famosi negozi online per maker

denominato "Arduino" e permettesse a qualunque persona dotata di basi di programmazione ed elettronica di accedere a questo campo affascinante. Non si parlerà in questa sede della storia di questo progetto nato a Torino tra buone birre in un locale ispiratore e i banchi di un istituto di design. Non si mancherà però di accennare al perché una scheda elettronica da 40 euro possa rendere l'automazione a portata di una massa ampia di persone. Serve però una breve digressione. Per automatizzare processi che implicano oggetti tangibili e reali come per esempio un motore elettrico è necessario che un computer sia in grado di "leggere" lo stato dell'oggetto di interesse ma anche di comandarne le funzioni. Il sistema informatizzato deve perciò avere la capacità di interfacciarsi con il mondo esterno attraverso una serie di porte di input e di output. Queste porte sono collegate con l'oggetto. Quest'ultimo deve essere realizzato in modo che possa fornire informazioni alle porte di input del sistema informatico e di ricevere comandi veicolati attraverso la porta di output. Una scheda Arduino fornisce proprio questo capacità di interfacciamento verso il mondo esterno. Sulla scheda di Arduino Uno sono presenti quattordici pin in grado di operare in modalità di input oppure di output. Attraverso il codice è cioè possibile specificare il comportamento di ogni singolo pin, se deve cioè essere usato per leggere dati da un dispositivo collegato o se deve essere utilizzato per impartirne. Questi pin sono di tipo digitale e quindi possono leggere stati binari di tipo ON/OFF oppure possono impartire comandi digitali di accensione/spegnimento. Per fare un esempio tangibile, è possibile rilevare se un interruttore collegato a uno dei quattordici pin è stato azionato e sfruttare un altro pin di Arduino per accendere un led in conseguenza del comportamento sull'interruttore. Sulla scheda sono presenti anche sei pin analogici in grado di leggere valori continui dall'ambiente esterno. La scheda Arduino diventa il ponte tra la programmazione e gli oggetti del mondo reale. Tutta la complessità è schermata dalla scheda e non è quindi necessario disporre delle conoscenze ingegneristiche imprescindibili per progettare una scheda di questo tipo.

La scheda Uno integra allo scopo un microcontrollore commerciale di grande diffusione e tutta l'elettronica necessaria per il funzionamento di questo dispositivo, oltre a quanto è necessario per l'interfacciamento tramite i pin di input/output. La scheda possiede inoltre una memoria non volatile contenente il firmware che presiede al funzionamento dell'ambiente. Generalmente il progettista di una scheda per l'automazione deve anche realizzare il software di controllo ma in questo caso è stato tutto realizzato dal gruppo che promuove il progetto. L'ambiente di controllo è perciò pronto all'uso. Basta alimentare Arduino tramite il jack a 5 Volt a cui si collega un normale alimentatore. In alternativa è possibile usare delle normali pile, meglio se ricaricabili. L'interfacciamento con il computer per le fasi di programmazione avviene tramite la porta USB.

I pin digitali

I pin digitali operano in TTL, ovvero con i segnali digitali di 0 Volt e 5 Volt, rispettivamente per indicare le condizioni di OFF e di ON. Tecnicamente i voltaggi di soglia operano entro un range più ampio, ma non è questa la sede per i dettagli di elettronica digitale. La programmazione avviene tramite un IDE liberamente scaricabile dal sito del progetto all'indirizzo **www.arduino.cc**. Il linguaggio è il celebre C a cui sono state aggiunte le librerie per l'operatività con Arduino. Il sito ufficiale contiene numerosi esempi ben commentati all'indirizzo **http://arduino.cc/en/Tutorial/HomePage**. Si può notare l'efficacia dell'ambiente osservando la modalità per accendere un LED collegato al pin 13. Il codice seguente è tratto dal sito ufficiale. Basta compilarlo e caricarlo sulla scheda Uno:

```
/*
Blink
Turns on an LED on for onesecond, then off for onesecond,
repeatedly.

This example code is in the public domain.
*/

// Pin 13 has an LED connected on most Arduino boards.
// giveit a name:
int led = 13;

// the setup routine runs once whenyou press reset:
void setup() {
  // initialize the digital pin as an output.
pinMode(led, OUTPUT);
}

// the loop routine runs over and over againforever:
voidloop() {
digitalWrite(led, HIGH);   // turn the LED on (HIGH is the
voltagelevel)
  delay(1000);               // wait for a second
digitalWrite(led, LOW);    // turn the LED off by making the
voltage LOW
  delay(1000);               // wait for a second
}
```

È evidente la semplicità con cui si può comandare il mondo esterno. Il fulcro dell'esempio è la funzione digitalWrite() attraverso cui impostare lo stato dei pin digitali, in questo caso mettere il pin 13 in stato di on, erogando i 5 Volt necessari all'accensione del diodo LED. Si consiglia di esaminare gli altri esempi forniti sul sito di Arduino e leggere la documentazione tecnica scritta in maniera diretta ed essenziale. Si diventerà fluenti nell'ambiente in pochissimo tempo.

Componenti necessari

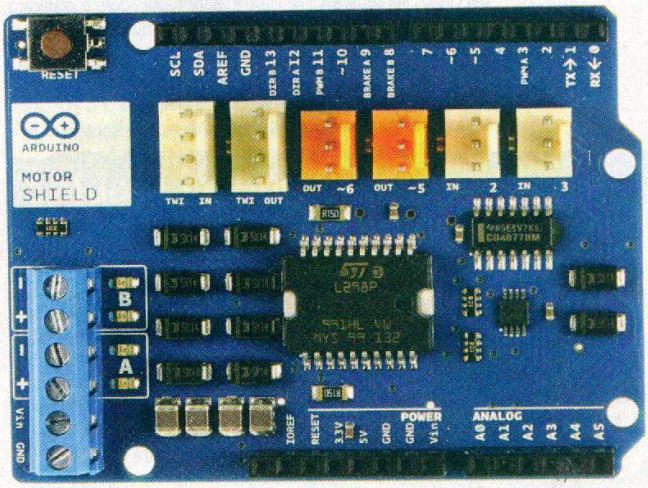
- 1 Tamiya Track& Wheel Set (item 70100-600)
- 3 Tamiya Universal Plate Set (item 70098-360)
- 1 Tamiya Twin-motorGearbox (Item 70097-840)
- 2 motori Solarbotics RM3
- 1 Arduino Uno R3
- 1 Arduino Motor Shield R3
- 1 breadboard 8 x 5,5 cm
- 2 interruttori on/off a scorrimento
- 1 LED a tre stati
- 1 resistenza 330 Ohm
- 1 buzzer acustico
- 1 sensore Ping di Parallax
- Fili elettrici
- Pile, meglio se ricaricabili
- Portapile

Un sistema semovente

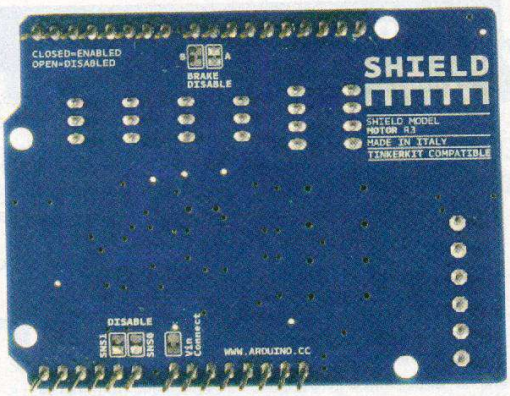
Arduino ha stimolato la fantasia degli utenti di tutto il mondo e ha in qualche modo sancito la nascita del movimento dei *maker*, persone che costruiscono oggetti per risolvere necessità pratiche oppure per assecondare la voglia di divertimento geek oppure per una forma moderna di ispirazione artistica. Basta fare alcune ricerche con Google per rendersi conto di quanto il movimento dei maker sia vasto e variegato. I progetti più diffusi implicano la realizzazione di oggetti semoventi. Piccoli robot in grado di interagire in qualche modo con il mondo reale. In questo articolo saranno illustrati gli aspetti generali necessari per la costruzione di un piccolo robot cingolato. Nel prossimo numero si esaminerà il codice di controllo. L'oggetto deve svolgere alcuni semplici compiti:

- Muoversi in avanti in linea retta
- Eseguire una scansione ciclica per rilevare eventuali ostacoli
- In caso di ostacolo fermarsi, ritornare indietro di alcuni centimetri e cambiare direzione. Poi proseguire nuovamente in linea retta

Il compito è semplice e di facile comprensione. Il robot sarà dotato di cingoli, azionati da due motori elettrici. I cingoli saranno montati su una basetta forata utilizzata come chassis su cui installare il sensore per rilevare gli ostacoli, la batteria e la scheda Arduino Uno.



Motor Shield (tratto dal sito ufficiale di Arduino). Si noti come i pin di Arduino sono stati riportati sullo shield. I motori si collegano alla morsettiera di colore blu



› Facciata posteriore del Motor Shield (tratto dal sito ufficiale di Arduino). Per disaccoppiare l'alimentazione dell'elettronica da quella dei motori si deve tagliare il contatto Vin Connect visibile nella parte inferiore

Le scatole di montaggio

Il meccanismo cingolato e la basetta forata impiegati per questo progetto sono realizzati da Tamiya, uno dei nomi più famosi nell'ambito del modellismo. Il produttore giapponese ha in catalogo una serie di scatole di montaggio specifiche per i progettisti, realizzate con buona qualità costruttiva e vendute a costi molto accessibili. Per il progetto serve un kit cingolato, i motori, gli ingranaggi e tre basette forate. Qui di seguito sono elencati i nomi delle scatole di montaggio:

» TamiyaTrack& Wheel Set (item 70100-600)
» Tamiya Universal Plate Set (item 70098-360)
» Tamiya Twin-motorGearbox (Item 70097-840)

Le scatole possono essere ordinate presso negozi di modellismo oppure online sui siti dedicati ai maker. I kit Tamiya sono un riferimento importante per via del rapporto costo/qualità e sono in genere reperibili con facilità. Una volta acquisite le scatole si deve mondare il kit "Twin-motorGearbox" contenente due motori elettrici e una serie di ingranaggi che permettono di ottenere i rapporti 58:1 oppure 203:1. La scelta del rapporto avviene in fase di montaggio posizionando i semiassi nelle apposite fessure della scatola di cambio. In questo caso si è scelto il rapporto 58:1.

Il complesso così realizzato dovrà essere posizionato nella basetta forata "Universal Plate Set" e fissato con le apposite viti fornite nel kit. Si dovranno poi fissare le ruote di trasmissione dei cingoli sui semiassi, installare le altre ruote libere e applicare i cingoli. Tutti questi elementi sono contenuti nel kit "Track& Wheel Set". Le istruzioni precise per il montaggio sono presenti all'interno delle scatole dei kit Tamiya in lingua giapponese oppure in inglese. Sui motori si dovrà saldare con uno stagnatore due fili elettrici nelle apposite sedi dei contatti elettrici. Questi fili andranno poi collegati ai pin di controllo di Arduino. In molti forum e siti dedicati ai maker viene consigliato di cambiare i motori inclusi nel kit Tamiya con i Solarbotics RM3. Si tratta di motori del tutto compatibili nelle dimensioni e nei voltaggi di esercizio ma che hanno un assorbimento inferiore di corrente elettrica durante alcune modalità di esercizio. La scelta comporta un consumo inferiore della batteria. I motori sono disponibili all'indirizzo **https://solarbotics.com/product/rm3**.

Il telaio

Per la realizzazione del prototipo si è deciso di impiegare tre basette forate sovrapposte, tenute a distanza da supporti cilindrici filettati. Lo spazio tra le basi risulta così di circa tre centimetri, molto comodo per posizionare i componenti e le batterie. In questo caso si impiega la prima basetta per fissare il blocco del motore e le ruote. Sulla seconda basetta si alloggiano le batterie mentre sull'ultima viene posizionata l'elettronica di controllo. Si è naturalmente liberi di scegliere il numero di basette che si reputa più comodo. Naturalmente maggiore è la quantità di "materia" che si inserisce nel progetto e maggiore sarà la massa che i motori dovranno spostare, comportando consumi più elevati. I cilindri distanziatori possono anch'essi essere scelti liberamente. Vi è ampia scelta di lunghezza e diametro nei cataloghi dei negozi online di elettronica o nei, purtroppo sempre più rari, negozi fisici di componenti. Personalmente ho impiegato distanziatori di tre centimetri di lunghezza e mezzo centimetro di diametro in materiale plastico. Le viti di serraggio coordinate sono anch'essere di plastica, sufficienti per la creazione del telaio per un simile prototipo. Vista la locomozione a pile non vi saranno particolari sollecitazioni meccaniche da tenere in considerazione. Nelle numerose pagine disponibili sulla Rete si trovano progetti analoghi realizzati con tavole di compensato, fogli di plexiglass o perfino con cartone robusto. È a discrezione personale valutare se usare un supporto plastico come il prodotto Tamiya suggerito o indirizzarsi verso soluzioni artigianali. Naturalmente dipende anche dalla maestria personale e dalla disponibilità di strumenti per il taglio e la foratura.

Il motore

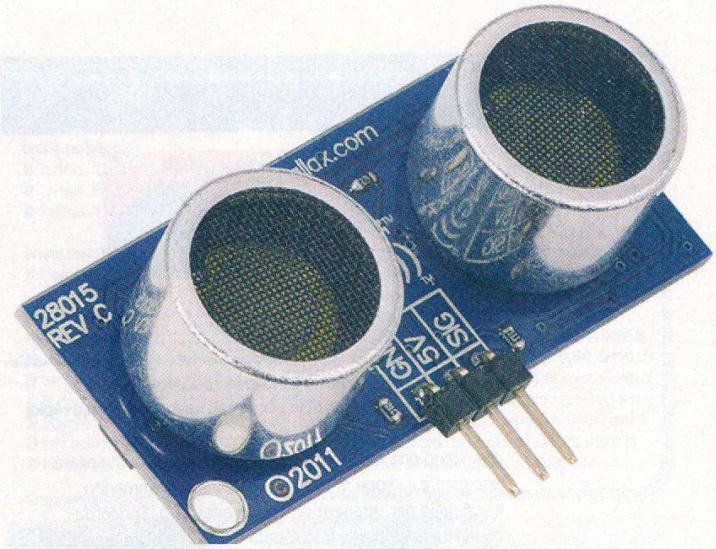
Il kit impiegato per imprimere movimento rotatorio ai cingoli impiega due motori. Questi devono essere indipendenti. Si deve cioè avere la facoltà di comandare singolarmente ogni motore per rendere possibile la rotazione del prototipo a destra e a sinistra. Per farlo si devono azionare i motori in direzioni opposte. Questo comporterà la rotazione intorno all'asse ortogonale al piano su cui poggia il prototipo. Azionando i motori nella stessa direzione si potrà invece ottenere il moto in avanti oppure indietro. I motori elettrici hanno un assorbimento elettrico significativo durante la partenza da fermo, in presenza di ostacoli oppure in salita. Si avrà un notevole consumo della batteria in queste fasi e una sollecitazione elettrica nei circuiti elettronici che presiedono alla gestione del motore. Per questo motivo si consiglia di impiegare i motori compatibili segnalati nel link più in alto. Questi hanno un assorbimento inferiore e rendono più facile lo sviluppo del prototipo, soprattutto all'inizio quando non si ha molta dimestichezza con i componenti. I motori impiegati sono a rotazione continua. Questo significa che è sufficiente fornire tensione ai due capi del motore per ottenere il movimento

› Batteria ricaricabile da aeromodellismo. HobbyKing (www.hobbyking.com) ha un catalogo ampio di batterie a costi accessibili

in maniera costante. La velocità di rotazione è proporzionale alla tensione fornita, entro il limite massimo supportato dal motore, oltre al quale si danneggia il componente. La scelta tra rotazione in avanti o indietro si ottiene invertendo la polarità ai capi del motore. I motori elettrici non possono essere semplicemente collegati a uno dei pin di input/output di Arduino viste le correnti che attraversano il componente. Facendolo si rischia di danneggiare la scheda elettronica. Serve piuttosto un circuito di controllo dotato di un circuito appositamente costruito per lo scopo. La progettazione di questo elemento non è banale ma fortunatamente esiste anche in questo caso una soluzione già pronta, concepita dal gruppo di Arduino.

Arduino e gli shield

La scheda Arduino Uno è un elemento autonomo che può essere utilizzato per progetti di automazione che implichi la lettura di segnali digitali da dispositivi esterni, per esempio sensori o l'invio di segnali di tipo ON/OFF. L'interfacciamento con elementi più complessi o che operano con tensioni e correnti non compatibili con Arduino devono essere gestiti attraverso gli "shield". Si tratta di schede appositamente progettate per funzionare con Arduino, dedicate a un compito ben preciso. Vi sono shield per fornire funzionalità Wi-Fi, shield per avere la compatibilità Ethernet oppure shield specifici per pilotare motori elettrici. I shield si innestano direttamente su Arduino. Questi sono dotati di pin che si inseriscono esattamente nelle morsettiere presenti sulla scheda Arduino. Non si perde capacità di interfacciamento perché i pin di Arduino sono riportati direttamente sulla scheda shield per l'utilizzo da parte del progettista. Non tutti i pin sono comunque utilizzabili liberamente perché alcuni sono impiegati per le finalità dello shield. I dettagli sul "Motor Shield" sono forniti sul sito ufficiale di Arduino, all'indirizzo **http://arduino.cc/en/Main/ArduinoMotorShieldR3**. Questo shield integra un circuito L298P capace di pilotare due motori a rotazione continua con assorbimento massimo di 2 Ampere per motore. Un motore viene comandato tramite il pin digitale 12 (motore A) e l'altro con il pin digitale 13 (motore B). Non si devono però usare i pin sulla morsettiera nera che riporta i segnali di Arduino sul shield ma la morsettiera di colore blu presente sul bordo della scheda. Due di questi pin riportano la serigrafia "A" e gli altri due riportano la serigrafia "B". Su questi morsetti si devono collegare con un filo i due contatti dei due motori. Impostando il pin 12 e13 al segnale di ON oppure di OFF si otterrà la rotazione del motore in avanti oppure in direzione inversa. Per frenare il motore si utilizzano i pin digitale 9 (motore A) e 8 (motore B). Impostando il pin su ON si aziona il freno, fermando immediatamente il motore mentre con un segnale OFF si disattiva il freno. La velocità di esercizio del motore si imposta attraverso i pin 3 (PWM motore A) e 11 (PWM motore B). Si deve indicare un valore tra 1 e 255. Maggiore è il valore e più elevata sarà la velocità di rotazione del motore. Prima di assemblare il prototipo in forma definitiva si consiglia di sperimentare i valori da assegnare ai due PWM. Non tutto lo spettro dei valori può essere impiegato. Il motore reagisce correttamente solo a una gamma ristretta. Serve quindi una fase di sperimentazione per comprendere quali sono i valori ottimali per i motori che si stanno utilizzando. Potrebbe anche essere necessario impiegare valori differenti per i due motori. Per via di piccole differenze di tolleranza meccanica o piccole imprecisioni nell'assemblaggio si potrebbe avere un fenomeno di deriva verso un lato nei momenti in cui entrambi i motori sono attivi in direzione frontale. Pur azionando entrambi i motori si potrebbe avere un lento movimento verso destra o sinistra. Per compensare la deriva potrebbe essere necessario far funzionare uno dei motori a una velocità di poco inferiore a quella dell'altro. Anche in questo caso serve una fase di sperimentazione e tentativi. Se si utilizzano motori che funzionano con tensioni maggiori o superiori a 9 Volt è necessario usare una seconda alimentazione dedicata ai motori. Questa alimentazione dovrà essere collegata ai pin Vin e GND della morsettiera blu. È necessario però anche girare il Motor Shield e tagliare la connessione "Vin Connect" con un cutter. In questo modo si impedisce che l'alimentazione inviata ai motori interessi la scheda Arduino che è progettata per essere alimentata con una tensione tra i 7 e i 12 Volt massimi. La divisione delle alimentazioni è anche utile per evitare cali di tensione all'elettronica di controllo durante il funzionamento del motore in diverse condizioni di stress. Anche isolando le alimentazioni si ha un limite di tensione che può essere all'applicato all'ingresso del Motor Shield ai morsetti blu. Il valore documentato è di 18 Volt. Per l'alimentazione dei motori si consiglia di impiegare batterie ricaricabili per modellismo. Vi è una scelta molto ampia con capacità elevate, utile per aumentare l'autonomia del cingolato. Naturalmente la controparte è un inevitabile incremento del peso e la maggior energia necessaria per il movimento. Altrimenti si possono usare normali batterie disponibili presso i supermercati. Le batterie a 9 Volt sono pratiche da utilizzarsi e da fissare al prototipo vista la loro forma



› **Sensore a ultrasuoni Ping di Parallax**

Saldare i componenti?

I progetti di prototipazione, come l'esempio illustrato in queste pagine non richiede attività di saldatura in quanto utilizza una breadboard e i contatti di realizzazione per semplice inserimento del filo elettrico. Si consiglia a tal proposito di acquistare fili per prototipazione, già tagliati in piccole lunghezze e dotati di boccola finale, comoda per l'inserimento. L'unica saldatura necessaria nel progetto riguarda i motori elettrici. Si devono saldare due fili su ogni motore. Serve perciò uno stagnatore per elettronica e un rotolo di stagno. Si consiglia di eseguire l'operazione se si ha la necessaria esperienza onde evitare di danneggiare il motore oppure comportare danni. Uno stagnatore se utilizzato in modo non attento può causare ustioni o incendi. Si consiglia di chiedere assistenza a persone competenti.

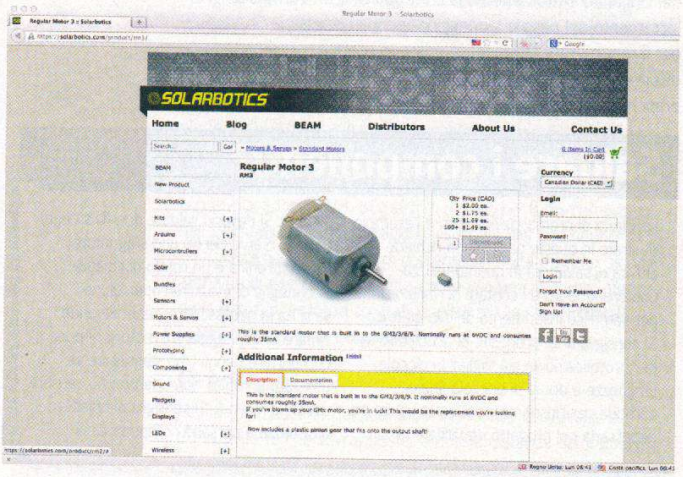
› Il sito ufficiale di Arduino contiene molta documentazione essenziale e ben scritta, utile per essere operativi in tempi rapidi

ma hanno una capacità molto limitata. È meglio impiegare una configurazione di batterie da 1.5 Volt, sommando il voltaggio per ottenere un valore adeguato. Si ricorda che la velocità dei motori è data dal valore di voltaggio. Si incrementa il voltaggio collegando le batterie in serie. Sarebbe utile mettere in parallelo più gruppi di batterie in serie per ottenere una maggiore capacità di carica. Ben presto però si noterà quanto spazio diventa necessario sul prototipo per alloggiare le batterie e quanto peso comportano. Si noterà pure il costo di sostituzione delle batterie esauste. Per questo motivo si consigliano le batterie ricaricabili ai polimeri di Litio (LiPo) da aeromodellismo. Sono disponibili in diversi voltaggi in multipli da 3.7 V, con capacità a scelta. Hanno un costo di acquisto discreto ma si ripagano in breve tempo con tutte le batterie normali che non vengono più gettate via.

Sensori

Esiste una disponibilità enorme di sensori che possono essere impiegati con Arduino. Non è infatti necessario acquisire prodotti specifici per Arduino oppure soluzioni hobbistiche. La scheda Arduino Uno implementa un microcontrollore di tipo industriale che può interfacciarsi con l'immensa disponibilità di sensori che l'industria elettronica offre per finalità professionali. Sensori di luminosità, di movimento, di pressione, di temperatura, di rilevazione gas, di flessione, di intensità luminosa, di prossimità, ecc. Questa componentistica professionale è in genere disponibile in gradi lotti ma possono oggi essere acquistati in negozi online dedicati ai maker in pezzi singoli e a costi generalmente molto accessibili. È interessante rilevare che questi oggetti possono essere forniti in forma "discreta", ovvero solamente il sensore oppure integrati in una basetta con tutta la componentistica necessaria al loro funzionamento. Questo formato è definito *springboard* ed è concepito dall'industria proprio per necessità di sperimentazione e prototipazione. Le modalità per pilotare i sensori e la documentazione tecnica sono disponibili presso i siti dei produttori nel formato di datasheet, ovvero di documentazione tecnica essenziale. La lettura di questa documentazione implica una conoscenza di base dell'elettronica e della programmazione. In alternativa è disponibile molta documentazione nei siti dedicati alla prototipazione con Arduino o altre piattaforme di sviluppo. Il sistema di sviluppo di Arduino include inoltre molti esempi già pronti di pilotaggio dei sensori più comuni. Per cominciare si consiglia di esplorare i siti di vendita di prodotti per maker, per esempio **www.homotix.it**. Questo sito ha un'ampia sezione di sensori, utile per farsi una panoramica di cosa offre l'industria. Individuato il prodotto di propria necessità è possibile eseguire l'acquisto anche di un singolo pezzo. Per il progetto di questo articolo è necessario impiegare un sensore in grado di rilevare la presenza di un ostacolo e di comunicare questa informazione ad Arduino in modo che il codice possa prendere provvedimenti per evitare la collisione con un ostacolo, in questo caso frenare il motore, ritornare indietro di qualche centimetro, cambiare direzione e azionare nuovamente il moto in avanti. Si utilizzerà un sensore a ultrasuoni, nello specifico il prodotto "Ping" fornito da Parallax (**http://www.parallax.com/product/28015**), un'azienda americana impegnata da anni nella fornitura di prodotti di elettronica per l'industria e il settore educational. Il sensore funziona emettendo un impulso a ultrasuoni e poi misurando il tempo con cui questo segnale ritorna indietro dopo essere stato riflesso da un ostacolo. Attraverso un semplice algoritmo è possibile stabilire la distanza dell'ostacolo misurando il tempo. Il range di distanza misurabile spazia dai due centimetri ai tre metri. Il sensore è fornito su una basetta di piccole dimensioni con tre pin per l'alimentazione e il controllo. La basetta contiene l'elettronica di controllo essenziale e due "cilindri" per l'emissione degli ultrasuoni e la rilevazione degli stessi eventualmente riflessi su uno ostacolo. È necessario fornire una tensione stabile di 5 Volt per alimentare l'oggetto attraverso gli appositi pin GND e 5V. Il pin SIG dovrà essere usato in alternanza tra modalità di scrittura e di lettura. Per cominciare si deve impostare la modalità di scrittura sul pin di Arduino a cui è collegato il pin SIG del sensore, mantenere il pin in stato di HIGH per alcuni microsecondi, porre il pin in modalità di lettura e rilevare un eventuale segnale HIGH, indice di un ostacolo. Il valore ottenuto dalla lettura è il tempo trascorso dall'emissione del segnale alla sua rilevazione. Minore il tempo e più vicino si trova l'ostacolo. Queste operazioni devono essere eseguite in modo continuo all'interno di un ciclo. Il sensore deve cioè "campionare" l'ambiente costantemente durante il moto in avanti del prototipo semovente. L'ambiente di sviluppo di Arduino contiene un esempio funzionante di utilizzo del sensore Ping. Questo è disponibile dal menu **File**, voce **Esempi**, sottovoce **06_sensors**. L'esempio si chiama **Ping**. Il codice è chiaro e ben commentato. Può essere incluso direttamente nel proprio progetto evitando qualunque difficoltà nella gestione del sensore. Il Ping può essere montato tramite un supporto

› Solarbotics (www.solarbotics.com) fornisce un motore compatibile con il kit Tamiya dotato di assorbimento elettrico contenuto



apposito realizzato in metallo da Parallax oppure integrato con qualche sistema "artigianale" al prototipo. Il sensore deve essere posizionato nel punto alto dello chassis e in posizione ortogonale rispetto al piano di movimento del prototipo. Si deve prestare attenzione al fatto che l'angolo di lavoro del sensore è stretto. La capacità di rilevazione è cioè di alcuni gradi rispetto all'asse centrale del cilindro del sensore. Il sistema è perfetto per misurare ostacoli frontali mentre non rileva ostacoli laterali. È importante tenere in considerazione che il sensore Ping necessita di una alimentazione regolare di 5V per effettuare le misurazioni. Tensioni inferiori ne pregiudicano il funzionamento. Bisogna tener conto di questo dettaglio in relazione all'assorbimento dei motori. Questa è una seconda motivazione per implementare due alimentazioni distinte per motori ed elettronica di controllo. In questo modo l'assorbimento elettrico del motore non inficerà la lettura della distanza per via di cali di tensione. Quando la carica della batteria sarà prossima all'esaurimento si noterà che il sensore Ping smetterà di essere operativo, generalmente prima di Arduino Uno.

Cablaggio dei componenti

Per la realizzazione elettrica del cingolato si è utilizzato una breadboard di piccole dimensioni (8 x 5,5 cm) dove ospitare componenti e cablaggi. Vicino alla breadboard è stata fissata la scheda Arduino con il Motor Shield agganciato. Sono utilizzate due alimentazioni separate, come indicato nella pagine precedenti. Le batterie sono collegate sulle breadboard con un cavo e dalla breadboard si innestano nella scheda Arduino per l'alimentazione dell'elettronica di controllo e nei morsetti blu di alimentazione del Motor Shield. I cavi dalla batteria potrebbero essere collegati direttamente alle schede, semplificando il cablaggio e riducendo il caos generato dai cavi volanti. Sfruttando la breadboard è però possibile impiegare due interruttori on/off e avere un meccanismo per l'accensione e lo spegnimento più comodo. L'alimentazione dedicata alla scheda Arduino viene impiegata anche per alimentare il sensore Ping. Perciò i tre pin sul sensore saranno anch'essi riportati sulla breadboard, due saranno collegati all'alimentazione mentre il pin di controllo sarà portato a uno dei pin digitali di Arduino, in questo caso il pin 7. Questo cablaggio potrebbe essere sufficiente per far funzionare il cingolato. Per avere una maggiore conoscenza di quanto sta avvenendo nel sistema si può impiegare un LED a tre stati, ovvero un LED in grado di essere negli stati di spento, di emissione del colore rosso o di emissione del colore verde. Il LED è dotato di tre pin. Uno di questi va collegato a massa mentre gli altri due servono ad accendere i relativi colori se alimentati. Durante il funzionamento normale del semovente si farà lampeggiare il LED di colore verde. In questo modo si potrà avere percezione del fatto che il programma sta funzionando regolarmente e non c'è stato un crash per errori di programmazione. Se il sensore incontra uno ostacolo entro una soglia prescelta, fissata in 25 centimetri, il LED si accende di colore rosso e si attiva la procedura per evitare l'ostacolo. In questo modo si ha la percezione visiva della rilevazione dell'ostacolo da parte del sensore e della conseguente attivazione del codice di gestione della collisione. Vengono usati i pin 2 e 4 rispettivamente per attivare il pin che controlla il colore verde del LED tristate e per il pin del colore rosso. Generalmente quando si collega un diodo LED a un segnale a 5 Volt è necessario impiegare una resistenza per evitare che il LED si bruci. Può essere sufficiente una resistenza da 330 Ohm tra il pin di Arduino che controlla il LED e il LED stesso. La resistenza non ha polarità e non c'è perciò un "verso" da rispettare. Questo non è vero per i LED, che sono componenti polarizzati. Per convenzione il piedino più corto va collegato a massa. Se si inverte la polarità non si comportano danni al componente. Semplicemente non si accenderà. All'interno del cingolato è possibile impiegare anche un buzzer e fare in modo che il semovente emetta una segnalazione acustica quando rileva un ostacolo. Il buzzer è una capsula cilindrica disponibile in vari formati e con differenti capacità di emissione acustica. Per il progetto si è scelto un modello di piccole dimensioni, nella fattispecie un centimetro di diametro. Vi sono due pin, uno connesso a massa e l'altro collegato al pin digitale 5 di Arduino. Il buzzer è posizionato sulla breadboard per comodità. Per emettere il suono non è sufficiente mettere il pin in stato di ON. Si deve far oscillare gli stati di ON e di OFF con una breve pausa tra il passaggio dei vari stati. Nel prossimo numero della rivista sarà illustrato il codice per far funzionare il cingolato secondo le modalità indicate nelle pagine precedenti. LXP

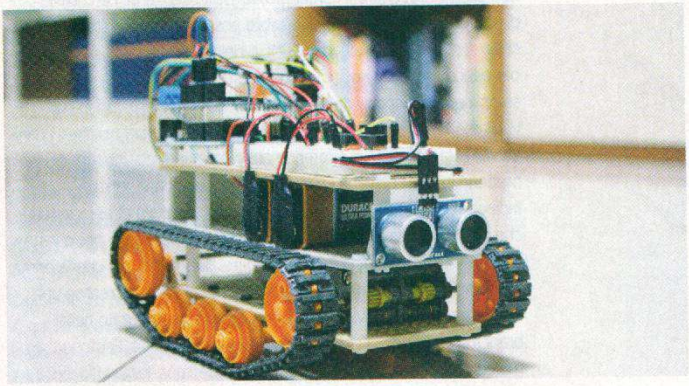
Costi componenti

www.polulu.com
- Tamiya 70100 Track and wheel set **$7,95**
- Tamiya 70097 Twin-Motor Kit **$8,95**
- Tamiya 70157 Universal Plate Set **$8,75**

www.solarbotics.com
- Motore Solarbotics RM3 **$2,00**

www.homotix.it
- Arduino Uno R3 **€22,99**
- Arduino Motor Shield R3 **€24,20**
- Sensore ultrasuoni Parallax Ping **€36,30**
- Supporto montaggio Parallax Ping **€14,52**
- Basetta breadboard 400 punti **€3,17**
- Led bicolore (10 pezzi) **€3,43**
- Kit resistenze assortite (160 pezzi) **€5,49**
- Kit ponticelli per breadboard **€3,63**

I cilindri distanziali per il montaggio delle basette possono essere acquistati da **http://it.rs-online.com** a circa **€15,00** per 100 distanziali. Le viti, in kit da 100 pezzi, hanno costi variabili in base al materiale. Il costo si attesta intorno ai **€10,00**. Se non si desidera usare normali batterie per alimentare i motori si può acquistare una batteria ricaricabile LiPo da 11.1 Volt presso **http://www.hobbyking.com**. Il costo varia in base al modello e la marca, intorno ai **$20.00**. Serve un apposito caricabatteria disponibile presso il sito al costo di circa **$10.00**. Per alimentare Arduino si possono usare normali batterie da 9 Volt visto il basso consumo della scheda. Dotarsi anche in questo caso dell'apposito portabatteria.



Il prototipo semovente assemblato, in una delle sue versioni di sviluppo

Versioni di Arduino

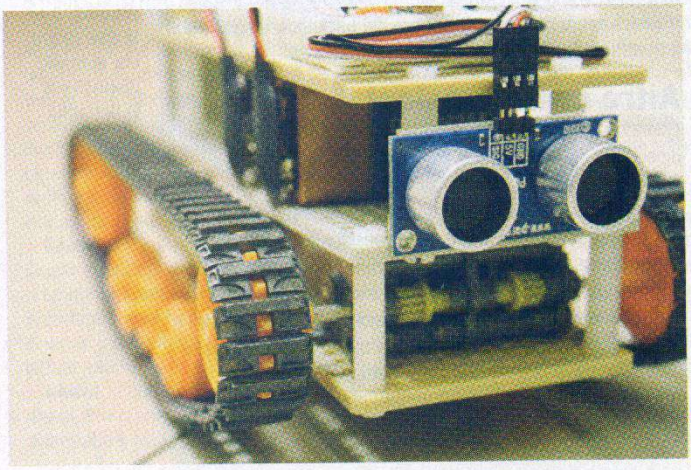
La scheda Arduino Uno è un prodotto di successo disponibile da molto tempo. La scheda è stata aggiornata nel tempo e la versione corrente è la R3. Anche il Motor Shield è ora in versione R3.

PARTE 2 Nella seconda parte di questo tutorial vediamo come completare il prototipo basato sulla tecnologia da hobbisty del momento...

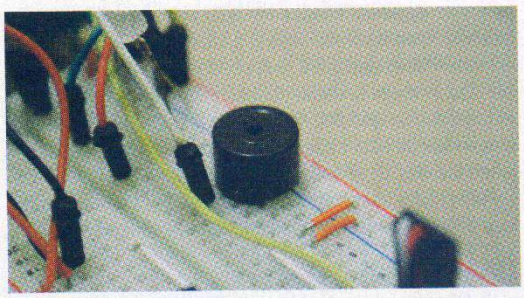
L'ambiente di prototipazione Arduino permette ai maker di realizzare progetti di elettronica in modo veloce, riducendo la complessità e il tempo che sarebbero altrimenti necessari per finalizzare i prototipi. Nel numero precedente della rivista sono state fornite alcune indicazioni per creare da un punto di vista hardware un robot semovente dotato di un sensore a ultrasuoni. Grazie al sensore è possibile misurare la distanza che separa il semovente da eventuali ostacoli presenti frontalmente e fare in modo che il veicolo eviti l'ostacolo. Il prototipo semovente è stato realizzato con una serie di kit della Tamiya, nome importante nel settore del modellismo, una scheda Arduino Uno, un Motor Shield ufficiale Arduino, un sensore a ultrasuoni Ping e alcuni componenti elettronici di contorno, assemblati tramite una basetta breadboard e alcuni ponti elettrici. In questo numero saranno fornite alcune indicazioni dal punto di vista software per realizzare il codice di controllo del semovente. Come presupposto si richiede la conoscenza del linguaggio di programmazione C, necessario per la realizzazione di codice per Arduino e un poco di familiarità con l'IDE di Arduino. Si deve anche sapere come caricare codice direttamente all'interno della scheda tramite un cavo USB collegato a un computer. Se non si hanno conoscenze specifiche sull'IDE si consiglia di consultare la documentazione online presso il sito ufficiale di Arduino su **http://arduino.cc/en/Guide/HomePage**.

Logica generale di funzionamento

Il cingolato funziona attraverso un loop principale. Il ciclo aziona il movimento in avanti del mezzo e contestualmente attiva una scansione attraverso il sensore a ultrasuoni. Se viene rilevato un ostacolo entro una distanza di sicurezza prefissata, in questo caso 25 centimetri, viene attivato il freno, impartito un breve moto all'indietro del cingolato e poi un cambio direzione. Completato questo semplice meccanismo per evitare l'ostacolo si riattiva il moto in avanti e il loop generale si ripete nuovamente. Si riporta qui di seguito il codice caricato sul cingolato per implementare il comportamento descritto.



› **Sensore a ultrasuoni Ping montato frontalmente sul prototipo**



› **Dettaglio della capsula per l'emissione della segnalazione acustica di ostacolo rilevato**

```
void loop()
{
 healthLed();
 delay(100);

 if (!dangerCheck())          // Nessun ostacolo
 {
 rilasciaFreno();
 avanti();
 }
 else                         // Ostacolo!!!
 {
 frena();
 emettiSuono();
 rilasciaFreno();
 indietro();
 delay(1000);
 indietroSX();
 }
}
```

La funzione loop() è uno standard di Arduino e deve essere rispettato. Tutti gli sketch operano infatti all'interno di un loop principale. Non si deve perciò alterare il nome di questa funzione. Come si può osservare, viene impiegata una collezione di funzioni per il funzionamento. La funzione healthLed() provvede a far lampeggiare il LED collegato al pin 2 della scheda. Questo permette di avere un riscontro visivo del funzionamento del software. Fino a quando si vedrà lampeggiare il LED, significa che il loop principale del programma sta funzionando correttamente. Se il LED smette di alternare gli stati acceso/spento significa che probabilmente il software è andato in crash o c'è qualche problema alla scheda oppure all'alimentazione. Se viene rilevato un ostacolo entro la distanza di sicurezza impostata si attiva una segnalazione visiva di colore rosso, più precisamente un LED di allarme di prossimità collegato al pin 4 di Arduino Uno. Superato l'ostacolo si spegne il LED rosso. Il LED verde del loop di programma continuerà così

a lampeggiare regolarmente. Immediatamente dopo la funzione healthLed() vi è una breve pausa tramite la funzione di sistema delay() e poi un blocco condizionale di tipo if...else. Viene testata come condizione il valore di ritorno, logicamente negato, della funzione dangerCheck(). Questa ritorna zero se non ci sono ostacoli e un valore diverso da zero se vi è un ostacolo entro la distanza di sicurezza. Nel caso non ci siano ostacoli si disattiva il freno tramite la funzione rilasciaFreno() e si attiva il movimento in avanti attraverso la funzione avanti(). In caso di ostacolo si impartiscono una serie di operazioni per segnalare l'evento e per evitare la collisione: si frena il semovente immediatamente con la funzione frena(), si aziona il buzzer per avere un segnale sonoro, si rilascia il freno, si attiva il movimento all'indietro con l'omonima funzione per allontanarsi dall'ostacolo e si attende un secondo impiegando la funzione delay() con il valore 1000. Per evitare che il semovente si imbatta ancora una volta nell'ostacolo si impartisce un movimento laterale per cambiare direzione. Si ritorna a questo punto al loop principale del programma nuovamente. Il sistema alternerà perciò in modo indefinito un moto lineare in avanti con una sequenza di moto all'indietro e cambio direzione per evitare eventuali ostacoli. Non si tratta evidentemente di un algoritmo "intelligente" ma di un semplice metodo per portare il cingolato lontano dall'ostruzione rilevata sulla direzione del moto. Si evita in questo modo una collisione certa e si raggiunge quindi l'obiettivo.

La gestione dei motori

Il prototipo cingolato è dotato di due motori indipendenti. Il movimento in avanti o indietro avviene segnalando al Motor Shield di far ruotare entrambi i motori in modo coordinato. Per cambiare direzione si attivano invece i due motori in senso alternato. Questo genererà una rotazione intorno all'asse del cingolato. Per gestire i motori attraverso il Motor Shield di Arduino si devono utilizzare sei pin, quattro digitali e due analogici. Questi vengono dichiarati nella parte iniziale del sorgente, insieme alla dichiarazione di altri pin e variabili utili al funzionamento del sistema:

```
const int pwm_a = 3;
const int pwm_b = 11;
const int dir_a = 12;
const int dir_b = 13;
const int brake_a = 9;
const int brake_b = 8;
```

Le variabili dir_a e dir_b servono a specificare rispettivamente la direzione del primo e del secondo motore. In modo analogo brake_a e braker_b servono a frenare i rispettivi motori. I pin 3 e 11 assegnati a pwm_a e pwm_b servono a regolare la velocità di rotazione dei motori. Nella funzione di setup del programma deve essere assegnato un valore tra 0 e 255. Serve un po' di sperimentazione per individuare i valori più idonei. Valori troppo bassi non sono funzionali e probabilmente non sortiranno alcun effetto. Allo stesso modo valori troppo elevati non sono consigliabili. Durante lo sviluppo del prototipo sono stati provati diversi valori procedendo per tentativi. I due valori non devono essere necessariamente gli stessi. Per errori di assemblaggio, attriti nei componenti, differenze costruttive e altre variabili si potrebbe avere una tendenza del semovente a muoversi verso una direzione laterale piuttosto che seguire una retta. Dal momento che questa deviazione è data da errori sistematici è possibile operare una correzione tramite le variabili pwm_a e pwm_b impostando valori diversi ai due cingoli. Attraverso una serie di tentativi si potrà correggere la tendenza alla deviazione laterale. Nel semovente impiegato per questo tutorial sono stati impiegati i seguenti valori:

```
analogWrite(pwm_a, 190);
analogWrite(pwm_b, 210);
```

Questi valori sono forniti a titolo di esempio. È necessario individuare i valori corretti per il proprio prototipo. Le altre variabili necessarie al funzionamento dei motori sono state inizializzate nel modo seguente:

```
pinMode(pwm_a, OUTPUT);
pinMode(pwm_b, OUTPUT);
pinMode(dir_a, OUTPUT);
pinMode(dir_b, OUTPUT);
pinMode(brake_a, OUTPUT);
pinMode(brake_b, OUTPUT);
digitalWrite(brake_a, LOW);
digitalWrite(brake_b, LOW);
```

Tutti i pin sono stati dichiarati come di uscita e si è disattivato il freno impostandolo a LOW.

Inizializzare LED e buzzer

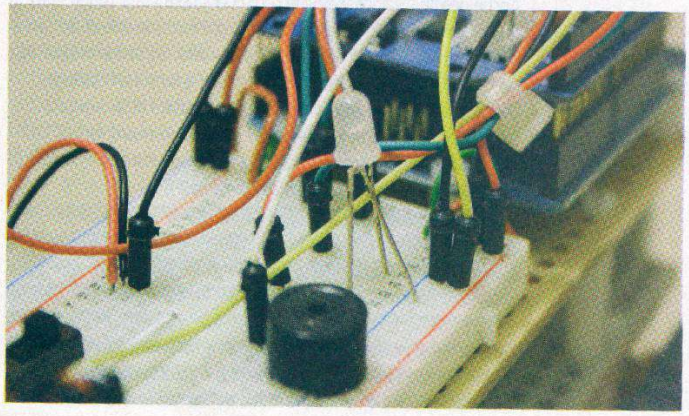
Il prototipo è dotato di due LED di segnalazione, come indicato prima. Uno di questi LED è usato per segnalare il funzionamento del loop principale mentre l'altro viene acceso quando il sensore ha rilevato un ostacolo. I LED necessitano di una inizializzazione. Il semovente dispone anche di un piccolo buzzer sonoro. Anche questo deve essere inizializzato.

```
pinMode(health_led, OUTPUT);
pinMode(prox_led, OUTPUT);
digitalWrite(prox_led, LOW);
digitalWrite(health_led, LOW);
pinMode(buzzer, OUTPUT);
digitalWrite(buzzer, LOW);
```

La procedura di inizializzazione richiede l'indicazione della modalità di output per i pin relativi e l'impostazione al valore LOW per azzerare lo stato di LED e buzzer. Il LED dedicato alla segnalazione del funzionamento del ciclo di programma lampeggerà durante il movimento del semovente. Un LED connesso a un pin di Arduino può essere acceso o spento. Per avere un comportamento alternato si deve scrivere una breve porzione di codice che alterna continuamente lo stato del LED. Per lo scopo viene definita una variabile di stato denominata health_output con scope globale per tutto il codice.

```
boolean health_output;
```

La variabile viene usata per contenere lo stato attuale del LED. Questa variabile sarà usata da una funzione che ne legge lo stato e imposta il LED in accordo con il valore contenuto. Poi viene invertito il valore. In questo modo al passaggio seguente su questa porzione di codice il LED sarà impostato nella condizione opposta. Si avrà così un'alternanza di stati accesi e spenti. Maggiori dettagli su questa funzione sono forniti più avanti in questo articolo.



La segnalazione verde per il ciclo di programma e la segnalazione rossa di rilevazione ostacolo è generata da un LED a tre stati

Dettaglio degli ingranaggi per la trasmissione del moto dai motori ai cingoli

Gestione dei motori

Il ciclo principale di funzionamento è stato scritto in maniera modulare, richiamando funzioni specifiche per ogni funzionalità del prototipo semovente. Il ciclo principale risulta così più ordinato, più facile da leggere e più comodo da mantenere nel tempo. Scrivere codice in modo modulare rientra tra le buone abitudini della programmazione. L'utilizzo di funzioni implica un certo overhead durante le frequenti chiamate ma questo non inficia il funzionamento del prototipo, trattandosi di un'applicazione con tolleranze ampie e tempi di reazione dell'ordine dei secondi. Per ottenere il moto in avanti o indietro è sufficiente impostare le variabili abbinate ai pin dei motori allo stato di HIGH oppure di LOW, come evidenziato dalle relative funzioni implementate per il prototipo:

```
void indietro()
{
  digitalWrite(dir_a, HIGH);
  digitalWrite(dir_b, HIGH);
}

void avanti()
{
  digitalWrite(dir_a, LOW);
  digitalWrite(dir_b, LOW);
}
```

Il moto nella direzione impostata viene mantenuto fino a quando non si cambia direzione o si attivano i pin per il freno:

```
void frena()
{
  digitalWrite(brake_a, HIGH);
  digitalWrite(brake_b, HIGH);
}
```

Anche questa funzione, come le controparti dedicate al movimento, risulta estremamente semplice. Si tratta dell'impostazione di un valore digitale sui pin relativi di Arduino. La funzionalità di freno implementata nel Motor Shield di Arduino è molto efficace e ferma la rotazione del motore in modo pressoché istantaneo. Si tratta della modalità preferenziale per fermare il prototipo. Bisogna poi ricordarsi di disattivare il freno prima di attivare il moto del prototipo. È stata realizzata una funzione specifica per lo scopo.

```
void rilasciaFreno()
{
  digitalWrite(brake_a, LOW);
  digitalWrite(brake_b, LOW);
}
```

Quando viene incontrato un ostacolo viene fermato il prototipo, si attiva il moto indietro per un secondo e si cambia direzione. Quest'ultimo comportamento è realizzato tramite una funzione:

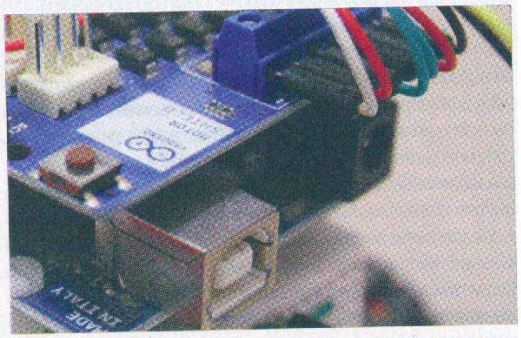
```
void indietroSX()
{
  digitalWrite(brake_a, HIGH);
  delay (750);
  digitalWrite(brake_a, LOW);
}
```

Viene frenato il motore A per 750 millisecondi e poi il freno è rilasciato. Dal momento che è stato rilevato un ostacolo, il semovente sta procedendo all'indietro. Frenando un solo motore si ottiene una rotazione intorno all'asse perpendicolare al piano.

Misurare la distanza

Il sensore Ping montato nella parte frontale del prototipo è dotato di tre pin. Due di questi sono dedicati all'alimentazione. Il terzo pin serve per l'emissione dell'impulso e la seguente lettura. Il pin è cioè utilizzabile sia in modalità di output che di input. Per utilizzare il sensore si deve quindi impostare il pin in modalità di scrittura e poi metterlo in stato di HIGH per alcuni microsecondi. Il codice di esempio fornito con l'IDE di Arduino impiega un impulso di cinque microsecondi. Una volta emesso il segnale si impostata il pin in modalità di lettura e si "ascolta" il sensore per un secondo per verificare se ritorna un impulso. Questa evenienza implica che il segnale è rimbalzato su un ostacolo posizionato davanti al semovente e ha investito il prototipo. Si deve quindi valutare il tempo trascorso tra la fine dell'emissione dell'impulso e la fine della lettura del segnale rimbalzato per determinare la distanza dell'ostacolo. Per trasformare un lasso di tempo in un valore di distanza si tiene conto della velocità media del suono nell'aria, circa 340 metri al secondo. Con le opportune conversioni si determina che il suono viaggia per circa un centimetro ogni 29 microsecondi. Si deve quindi dividere il valore ottenuto dalla misurazione del tempo di ritorno del segnale per 29 e poi per 2. La divisione per due è necessaria perché il valore rilevato è dato dalla somma del tempo che il suono ha impiegato a viaggiare dal sensore Ping all'ostacolo più il tempo necessario a percorrere il tragitto opposto dall'ostacolo al sensore. La misurazione è naturalmente grossolana e non si dovrebbero tenere conto dei decimali. È un grado di accuratezza del tutto adeguato per l'applicazione in oggetto. Si consiglia di fare alcune prove con il sensore prima di impiegarlo nel semovente. Si potrà così appurare quanto la misurazione sia buona considerando tra l'altro il costo modesto del sensore impiegato. Il codice impiegato nel prototipo per la misurazione è basato sul sorgente fornito con l'IDE di Arduino:

```
boolean dangerCheck()
{
  // The PING))) is triggered by a HIGH pulse of 2 or more
microseconds.
```



Porta USB sulla scheda Arduino per la programmazione del firmware del prototipo

```
  // Give a short LOW pulse beforehand to ensure a HIGH pulse:
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);

  // The same pin is used to read the signal from the PING))): a HIGH
  // pulse whose duration is the time (in microseconds) from the
sending
  // of the ping to the reception of its echo off of an object.
  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);

  // convert the time into a distance
  cm = microsecondsToCentimeters(duration);

  if (cm <= DISTANZA_SICUREZZA)
    {
     digitalWrite(prox_led, HIGH);
     digitalWrite(health_led, LOW);
     return(true);
    }
   else
    {
     digitalWrite(prox_led, LOW);
     return(false);
    }
 }
```

Nella parte finale della funzione è stato inserito un costrutto if...else per la valutazione della distanza. Se questa è uguale o inferiore al valore DISTANZA_SICUREZZA viene acceso il LED per la segnalazione dell'ostacolo e viene spento quello di segnalazione di funzionamento del ciclo principale. Viene poi richiesta l'uscita dalla funzione tramite la parola chiave return() indicando un valore booleano di verità. Sarà così possibile rilevare la condizione di allarme di prossimità dal ciclo principale del programma. Se la distanza di sicurezza non è stata violata viene spento o mantenuto spento il LED di segnalazione ostacolo e viene ritornato un valore logico falso.

Segnalazione ciclo e buzzer

Il corretto funzionamento del ciclo principale di programma è visualizzato attraverso un LED di colore verde che alterna gli stati acceso/spento. La funzione preposta a questo comportamento si chiama healthLed() ed è stata realizzata con poche righe di codice:

```
 void healthLed()
 {
  if (cm > DISTANZA_SICUREZZA)
     {
      digitalWrite(health_led, health_output);
      health_output = !health_output;
     }
   else digitalWrite(health_led, LOW);
 }
```

La funzione si basa su un costrutto if...else. Viene eseguito un test per comprendere se il valore contenuto nella variabile cm è maggiore della distanza di sicurezza impostata. La variabile cm ha scope globale nel programma ed è gestita dalla funzione dangerCheck() che di fatto controlla il sensore a ultrasuoni. La variabile contiene in ogni istante la distanza aggiornata in centimetri degli ostacoli rilevati frontalmente.

Se la distanza rilevata è superiore a quella di sicurezza viene alternato lo stato booleano del LED verde di segnalazione ciclo. In caso contrario il LED viene spento. Le porzioni restanti del programma provvederanno quindi ad accendere il LED rosso di allarme prossimità e gestire la situazione. Oltre alla segnalazione visiva viene anche emesso un breve suono tramite il buzzer montato sul cingolato e collegato al pin cinque di Arduino. È stata scelta una semplice capsula che opera attraverso la vibrazione di una membrana. Si ottiene così un suono udibile nei pressi del prototipo di natura non intrusiva. Si tratta cioè di un "beep" e non di un allarme. La gestione della capsula avviene con una funzione apposita denominata emettiSuono().

```
 void emettiSuono()
 {
  int c = 0;
  boolean stato = LOW;

  for (c=0; c < 200; ++c)
     {
      digitalWrite(buzzer, stato);
      stato = !stato;
      delay(2);
     }
 }
```

Per fare in modo che la capsula emetta il suono si deve far alternare lo stato del pin 5 all'interno di un ciclo. Per ogni ciclo si alterna lo stato e si attende un breve lasso di tempo, in questo caso 2 millisecondi. Alterando la pausa si modifica la frequenza del suono. Il numero di iterazioni del ciclo, in questo caso duecento, determina invece la lunghezza del suono. È consigliabile impiegare valori brevi per non occupare un tempo significativo della routine di gestione delle collisioni.

Compilazione e test

Il codice presentato in queste pagine e allegato con la rivista in forma digitale occupa poco più di 2 Kbyte una volta compilato. Resta quindi molto spazio libero all'interno della memoria Flash montata sulla scheda Arduino Uno. È possibile quindi espandere il codice nel tempo per migliorare il comportamento del cingolato, gestire nuovi sensori o adottare attuatori per incrementare la capacità di interazione con l'ambiente circostante. È evidente la semplicità con cui si possono leggere i valori di ritorno di sensori commerciali e agire su dispositivi quali motori a rotazione continua, LED e capsule sonore. È sufficiente una conoscenza generale della scheda Arduino, qualche competenza di elettronica e un po' di esperienza con il linguaggio C. Le conoscenze di elettronica richieste non sono di livello universitario. Sono sufficienti le nozioni e l'esperienza che vengono erogate nelle scuole superiori a indirizzo tecnico che contemplano l'elettronica digitale nel proprio percorso didattico. Le competenze di programmazione in linguaggio C si possono acquisire con un buon libro e un po' di pratica. Il C è un linguaggio imperativo classico senza estensioni a oggetti, veloce da apprendere. L'utilizzo generale di Arduino non richiede l'utilizzo dei puntatori del C, l'area di maggiore criticità nell'apprendimento del linguaggio. Questo è un altro dettaglio che semplifica l'apprendimento del linguaggio di programmazione. Arduino si rivela quindi un potente mezzo per la realizzazione di prototipi in modo facile, veloce ed estremamente economico. Si possono perciò trascorrere molte ore di puro divertimento imparando nozioni di programmazione, elettronica e automazione con un approccio pratico. Uno splendido hobby che può essere trasformato nel tempo in un percorso di studi ed eventualmente in un lavoro nell'ambito della tecnologia. LXP