2019

Scheda d’utilizzo

UNA GUIDA PER INTRODURRE I GIOVANI NEL MONDO DI ARDUINO

DI MATTIA LAZZARONI & MATTIA TOSCANELLI

Indice

Scopo

Lo scopo di questo manuale è quello di facilitare la vostra comprensione riguardo circuiti e librerie da noi implementate. Queste sono:

1. Potenziometro
2. Servo Motore
3. Ultrasuono

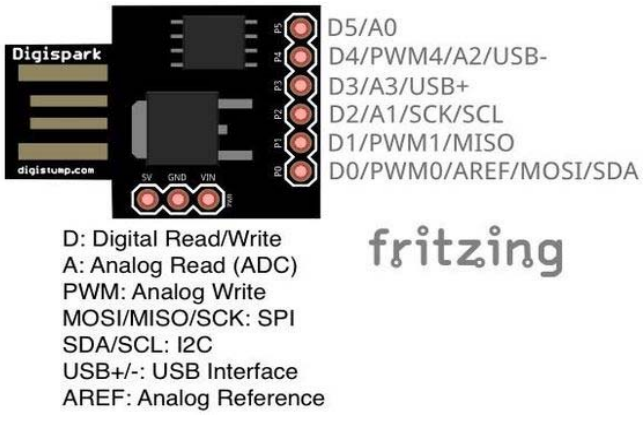
Nella prima parte di questo manuale vi verranno descritti i componenti utilizzati nelle librerie e soltanto in seguito otterrete le descrizioni delle singole librerie.

Componenti

Arduino Digispark

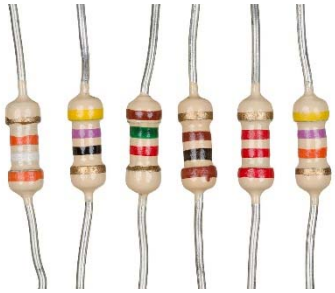
Il **Digispark** è una scheda di sviluppo di piccole dimensioni e possiede un microcontrollore. Questo microcontrollore è simile a quello presente su un classico Arduino, soltanto che quello del Digispark costa meno ed è un po' più scarso. Nell'immagine qua sotto, sulla sinistra, si può vedere il Digispark sui due fronti. Come si può notare la scheda è già assemblata, tranne per i due connettori che si possono saldare facilmente. Lo scopo di questa scheda è quello di realizzare dei dispositivi di controllo o degli automatismi. Digispark dà la possibilità di usare l'IDE (software per la programmazione) di Arduino, che è gratuito e accessibile a tutti. Se si vuole collegare la scheda a dei componenti bisogna utilizzare i vari pin presenti sul Digispark. Nella parte inferiore troviamo tre pin. Il primo è il pin dell'alimentazione ed è indicato con "5V", il secondo pin è il pin della massa (o del -) ed è indicato con "GND" che sta per "Ground" (tradotto in italiano "Terra") e il terzo è il pin per alimentare la scheda da un alimentazione esterna ed è indicato con "Vin". I restanti 6 che si trovano sulla parte destra (dal P0 al P5) sono i pin "programmabili". Questi possono scrivere (cioè emettere corrente) o leggere (cioè ricevere corrente), inoltre possono assumere due tipologie diverse: Digitali, cioè che che lavorano soltanto con 0 (niente corrente) e 1 (corrente) oppure Analogici, cioè che lavorano con un intervallo di corrente che va da da 0 a 1023. Tutti i pin possono assumere la funzione digitale sia di lettura sia di scrittura, ma non tutti i pin possono assumere la funzione analogica. Infatti per quanto riguarda la scrittura analogica possono eseguirla solamente i pin: P0, P1 e P4. Per quanto riguarda la lettura analogica possono eseguirla solamente i pin: P2, P3, P4, P5. Normalmente la lettura analogica su Digipark ha una struttura diversa da quella che si può pensare infatti per leggere dal pin P2 bisogna scrivere 1, il pin P5 bisogna scrivere 0, il pin P4 bisogna scrivere 2 e per il pin P3 bisogna scrivere 3. Però grazie alle nostre librerie tutta questa struttura strana si può anche dimenticare, ma basta seguire la tabella qui di seguito:

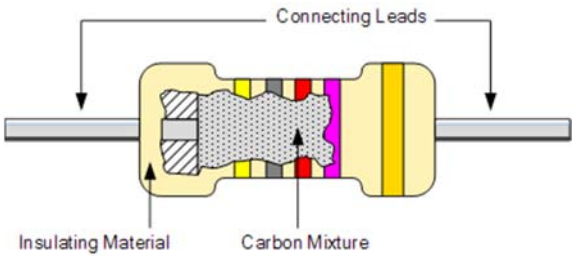
Pin	Digital Read/Write	Analog Read	Analog Write
0	X		X
1	X		X
2	X	X	
3	X	X	
4	X	X	X
5	X	X	



Resistenza

Una **resistenza** è un piccolo componente progettato per fornire una quantità specifica di resistenza in un circuito elettronico. Dato che la resistenza è un elemento essenziale di quasi tutti i circuiti elettronici, utilizzerai i resistori in praticamente tutti i circuiti che creerai. Sebbene i resistori siano disponibili in una varietà di dimensioni e forme, il tipo più comune di resistenza per l'elettronica per hobby è la resistenza *carbon film*. Questi resistori sono costituiti da uno strato di carbonio depositato su un materiale isolante e contenuto in un piccolo cilindro, con fili metallici attaccati ad entrambe le estremità. Dato che le resistenze non hanno polarità si possono collegare ad un circuito in entrambi i sensi.





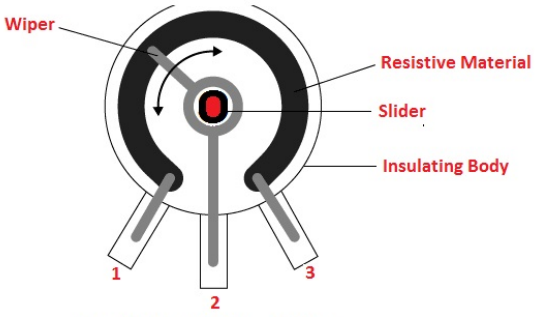
Potenziometro

Un **potenziometro** è un componente che è in grado di opporre una resistenza variabile al passaggio di un segnale. Consente di variare la velocità di intermittenza del LED senza modificare alcun componente nel circuito. I potenziometri possono avere varie forme, dimensioni e valori ma tutti hanno le seguenti cose in comune:

- Tre terminali (o punti di connessione).
- Una manopola, una vita o un cursore che può essere spostato per variare la resistenza tra il terminale centrale e uno dei terminali esterni.
- La resistenza tra i due terminali esterni è una resistenza fissa ed è la resistenza massima del potenziometro.
- La resistenza tra il terminale centrale e uno dei terminali esterni varia da 0 Ohm alla resistenza massima del potenziometro, quando la manopola, la vite o il cursore vengono spostati.

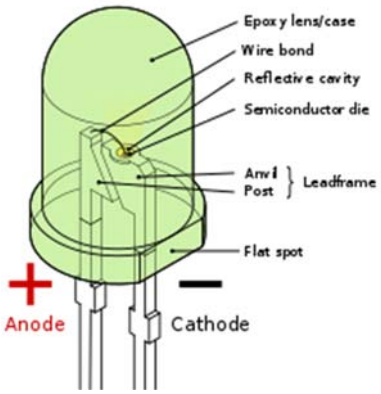
Nel software Arduino si misura la rotazione del potenziometro tramite il metodo *analogRead()* che ritorna un valore fra 0 e 1023. Quando il potenziometro è in fase di “riposo” il metodo analogRead ritorna 0, mentre quando è roteato al massimo ritorna 1023. Nella nostra libreria del potenziometro abbiamo realizzato una metodo più semplificato che verrà spiegato in seguito.

Nei potenziometri il due terminali esterni (1 e 3 nell’immagine) vengono collegati al “+” e al “-“, mentre quello centrale indica il valore della rotazione. È indifferente quale dei terminali viene collegato al “+” e quale al “-“, in ogni caso se il potenziometro non dovesse funzionare bisognerebbe invertire i terminali.



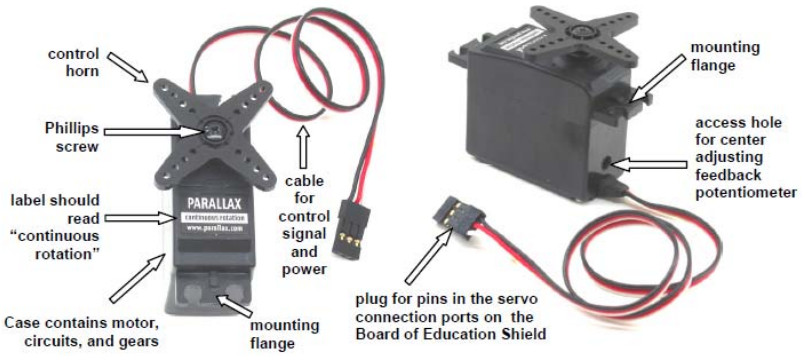
Led

Un **led**, che letteralmente significa “Light Emitting Diode”, ovvero “Diodo ad Emissione di Luce” è un semiconduttore che emette luce quando viene attraversato da una corrente elettrica. In pratica la luce emessa è data dal passaggio di elettroni da una parte all’altra del led (tra anodo e catodo). La corrente entra dall’anodo ed esce dal catodo. Se un led non dovesse accendersi provare ad invertire l’anodo col catodo. Il terminale più lungo indica l’anodo, quello più corto il catodo.



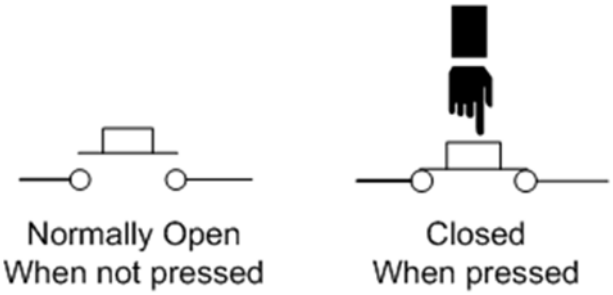
Servomotore

Un **servomotore** è un attuatore o motore rotante che consente un controllo preciso in termini di posizione angolare, accelerazione e velocità. Tutte queste sono caratteristiche che un normale motore non ha. Fa uso di un motore normale e lo accoppia con un sensore per il feedback di posizione. Il controller è la parte più sofisticata del servomotore, in quanto è specificamente progettato per lo scopo. Il servomotore è costituito da tre fili: un filo nero/marrone collegato alla massa, un filo bianco/giallo/arancio collegato all’unità di controllo e un filo rosso collegato all’alimentazione.



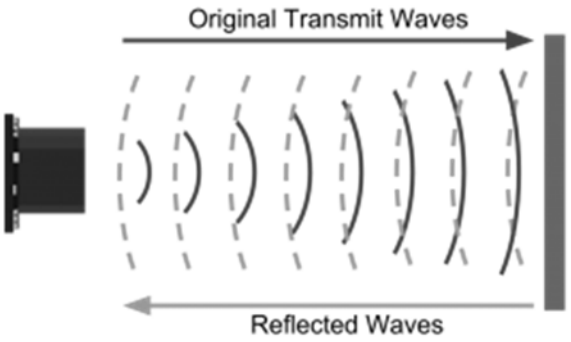
Pulsante

Un **pulsante** in elettronica è un interruttore che serve a collegare due fili. Quando il bottone non è premuto ed è quindi in stato di riposo i due fili sono scollegati e la corrente non circola. Nel momento in cui si preme il pulsante i fili si collegano e la corrente può circolare.



Ultrasuoni

Un sensore ad **ultrasuoni** misura la distanza utilizzando onde ultrasoniche.
La testina del sensore emette un'onda ultrasonica e riceve l'onda riflessa dall'obiettivo. I sensori ad ultrasuoni misurano la distanza dal bersaglio misurando il tempo tra emissione e ricezione. Un sensore a ultrasuoni utilizza un singolo elemento a ultrasuoni sia per le emissioni che per la ricezione. In un sensore ad ultrasuoni a modello riflettente, un singolo oscillatore emette e riceve alternativamente onde ultrasoniche.

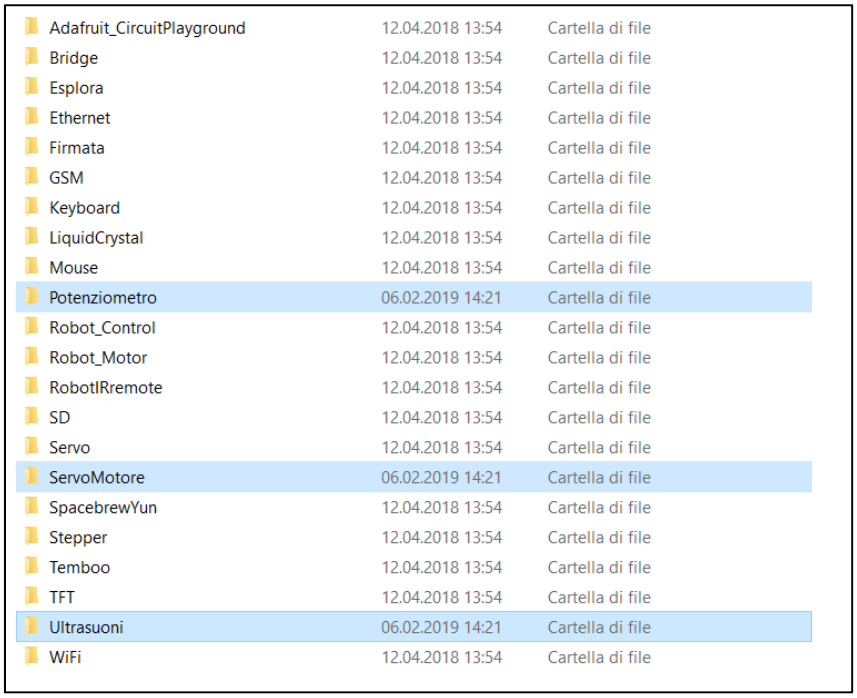


Installazione driver per Digispark

Per cominciare bisogna recarsi nel seguente sito:
https://github.com/digistump/DigistumpArduino/releases/download/1.6.7/Digistump.Drivers.zip
Una volta scaricato lo zip entrare nella cartella “Digistump Drivers” e aprire il file “Install Driver.exe”. Partirà quindi l’installazione guidata dei driver. Premere “Avanti >”, “Installa” e “Fine”. Fatto ciò recarsi nell’IDE (Arduino), premere “File” in alto a sinistra, successivamente “Preferences” e nella casella contrassegnata da “Additional Boards Manager URLs:” aggiungere il seguente link: “http://digistump.com/package_digistump_index.json” e fare clic su “OK”.
Premere ora il menu “Tools”, poi il sotto menu “Board: ….” e infine cliccare “Boards Manager…”.
Nella barra di ricerca inserire la parola “Digistump”, a questo punto selezionare il primo pacchetto che uscirà e premere “Install” in basso a destra. Quando l’installazione sarà terminata chiudere la finestra “Boards Manager”, tornare nel menu “Tools”, poi nel sotto menu “Board: ….” e selezionare la scheda “Digispark (Default – 16.5mhz)”.

Aggiungere le librerie per Arduino

Premere il tasto Windows (il tasto in cui c’è l’icona di Windows) e digitare “Arduino”. Premere ora tasto destro sul primo risultato e poi “Apri percorso file”. Verrà aperta una cartella contenente il collegamento al software “Arduino.exe” che sarà selezionato. Premere quindi tasto destro sul file selezionato e cliccare di nuovo “Apri percorso file”. A questo punto si potrà notare una cartella chiamata “libraries” che andrà aperta. All’interno di questa cartella vanno aggiunte tre cartelle chiamate con i nomi delle tre librerie come mostrato dall’immagine. In queste tre cartelle vanno aggiunti i rispettivi files .cpp e .h. Quindi nella cartella “Potenziometro” aggiungere i files “Potenziometro.cpp” e “Potenziometro.h” e così anche per le altre due cartelle.



Libreria 1 – Potenziometro

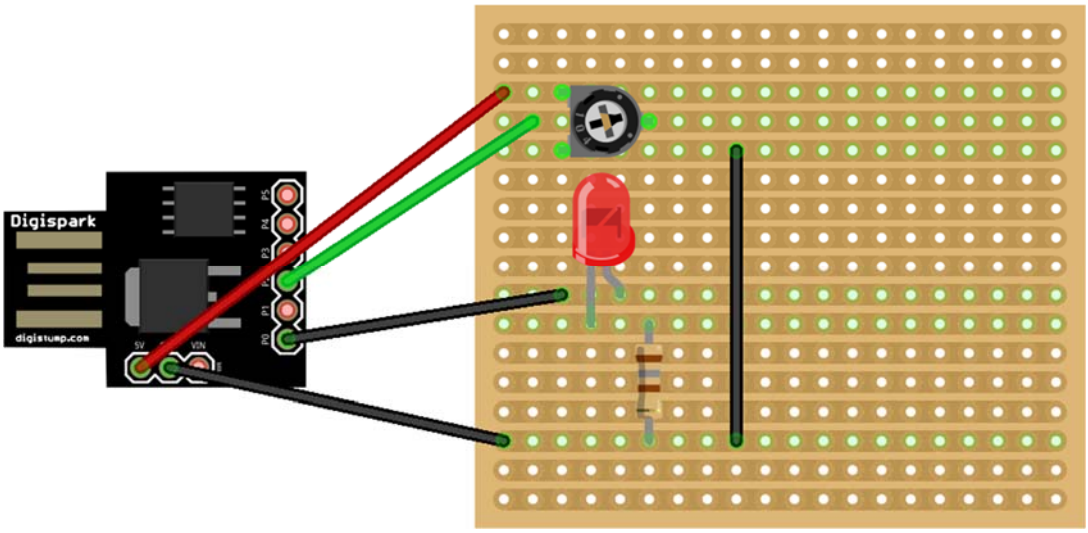
La prima libreria che abbiamo realizzato è per il potenziometro. Abbiamo inoltre creato un metodo “valoreLetto()” per ritornare il valore del potenziometro.

Materiale

- 1 Arduino Digispark
- 1 Potenziometro da 10 kΩ
- 1 Resistenza da 180 Ω
- 1 Led

Montaggio del circuito

Per facilitare la montatura del circuito aiutarsi con l’immagine qua sotto. Per prima cosa prendere il proprio Digispark ed inserire un cavo rosso nel pin indicato da “5V”. Questo cavo deve giungere sulla breadboard come da immagine ed essere infilato in un foro presente sulla stessa riga della parte più in alto del potenziometro. Nella riga che sta in mezzo al potenziometro collegare un cavo che giunge al pin P2. Nel pin P0 aggiungere invece un filo che deve arrivare sulla stessa riga del “+” del led, ovvero il punto più in alto del led. Infine dovrà venire collegato un ultimo filo nero nel pin con scritto “GND” (quello accanto al “5V”) che dovrà arrivare sulla stessa riga della parte più in basso del potenziometro e del filo nero.



Struttura

Per realizzare questa libreria abbiamo un’interfaccia chiamata “Potenziometro.h”, in cui definiamo “Potenziometro_h” se esso non è ancora definito, e la libreria “Potenziometro.cpp” che estende l’interfaccia. Entrambi i files includono “Arduino.h”.

Metodi

- ***Potenziometro(int pin)****;* Questo metodo costruttore riceve come parametro il pin a cui è collegato il potenziometro. Fatto ciò esso si occupa di impostare il pin di lettura del valore del potenziometro in base ai pin analogici del Digispark.
- ***valoreLetto()****;* Questo metodo restituisce il valore letto dal potenziometro. Il valore viene letto e successivamente modificato in un numero intero che va da 0 a 255.

Esempio 1 – Fade

Il primo esempio realizzato permette di dissolvere la luminosità del led a dipendenza della rotazione del potenziometro. Per cominciare andiamo ad includere la nostra libreria chiamata “Potenziometro.h”. In un secondo momento inizializziamo un oggetto associato alla classe Potenziometro,di nome potenziometro e successivamente dichiariamo una variabile led che conterrà la un valore intero (senza virgola) che indica la porta a cui è collegato il led. Nel setup creiamo una variabile potenziometro che andiamo poi ad istanziare e tramite il metodo pinMode passiamo come parametro la variabile led che contiene appunto il numero del pin che vogliamo impostare come output. Nel metodo loop che contiene una sequenza d’istruzioni che viene ripetuta continuamente andiamo a “scrivere” sul led (e quindi a cambiarne la luminosità) il valore che leggiamo dal potenziometro. Questa scrittura sul led avviene tramite il metodo “analogWrite()”. Il primo parametro di questo metodo indica il pin su cui scriviamo il valore e il seconda indica il valore da scrivere. Il valore che si va a scrivere viene letto dal potenziometro tramite il metodo “valoreLetto()” che abbiamo spiegato prima.

Codice

```
#include "Potenziometro.h"

//Inizializza un oggetto di tipo Potenziometro di nome potenziometro.
Potenziometro* potenziometro;

int led = 0;

void setup() {
  //Istanzia la variabile potenziometro assegnado la porta analogica all'interno
delle parentesi. (in questo caso 2)
  potenziometro = new Potenziometro(2);
  pinMode(led, OUTPUT);
}

void loop() {
  //Scrive sul LED il valore lettro dal potenziometro.
  analogWrite(led, potenziometro->valoreLetto());
}
```

Libreria 2 – ServoMotore

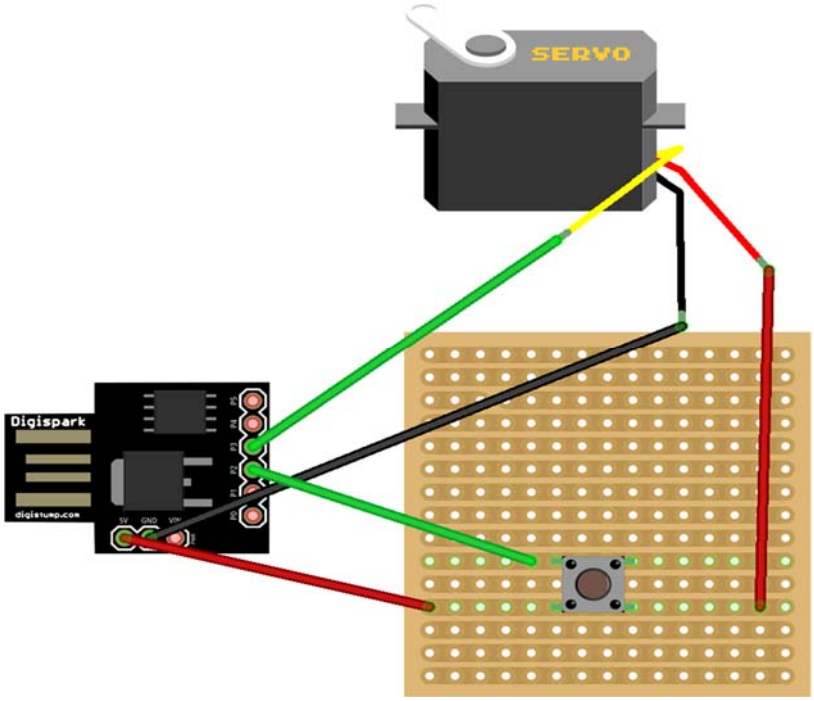
La seconda libreria che abbiamo realizzato è per il ServoMotore. Abbiamo creato tre metodi: uno per impostare la velocità del ServoMotore, uno per ottenere la posizione del ServoMotore e l'ultimo che si occupa di avviare il motore.

Materiale

- 1 Arduino Digispark
- 1 ServoMotore
- 1 Pulsante

Montaggio del circuito

Per facilitare la montatura del circuito aiutarsi con l'immagine qua sotto. Per prima cosa prendere il proprio Digispark ed inserire un cavo rosso nel pin indicato da "5V". Questo cavo deve giungere sulla breadboard come da immagine ed essere infilato in un foro presente sulla stessa riga della parte più in basso del pulsante. Collegare il cavo nero del ServoMotore nel pin accanto al "5V", che sarà contrassegnato da "GND". Nel pin P2 del Digispark inserire un cavo che deve collegarsi in un foro sulla stessa riga della parte più alta del pulsante. L'ultimo cavo che avanza del ServoMotore (che sarà bianco, giallo o arancio) va collegato al pin P3.



Struttura

Per realizzare questa libreria abbiamo un'interfaccia chiamata "ServoMotore.h", in cui definiamo "ServoMotore_h" se esso non è ancora definito, e la libreria "ServoMotore.cpp" che estende l'interfaccia. Entrambi i files includono "Arduino.h".

Metodi

- ***ServoMotore(int pin, int velocita)****;* Questo metodo costruttore riceve come parametro il pin a cui è collegato il ServoMotore e la velocità di movimento dell'elica. Fatto ciò esso si occupa di impostare il pin di scrittura del valore da inviare al ServoMotore.

- ***settaVelocita(int velocita)****;* Questo metodo riceve come parametro la velocità da impostare al ServoMotore. Se la velocità passata come parametro è inferiore a -100 viene limitata a -100. Nel caso in cui la velocità si maggiore di 100 viene limitata a 100. Nel caso che invece la velocità passata a parametro sia compresa tra -100 e 100 la variabile _velocita viene impostata con questo valore.
- ***ottieniPosizione();*** Questo metodo ritorna la posizione del ServoMotore con un valore che va da 0 a 180 gradi.

Esempio 1 – Base

Il primo esempio realizzato per questa libreria fa muovere il ServoMotore avanti e indietro. Per cominciare andiamo ad includere la nostra libreria chiamata “ServoMotor.h”. In un secondo momento inizializziamo un oggetto associato alla classe ServoMotore, di nome servoMotore e successivamente dichiariamo una variabile velocità, che conterrà un valore intero (senza virgola), di default 20, che indica la velocità di movimento del motore. Nel metodo setup istanziamo la variabile servoMotor assegnando il pin nelle parentesi (in questo caso il pin 3) e come secondo parametro (dopo la virgola) passiamo la velocità di movimento che varrà inizialmente 20. Nel metodo loop facciamo girare il servo motor inizialmente con une velocità positiva fino a che la posizione raggiunge il valore di 180, a quel punto la velocità diventa negativa fino a che la posizione arriva a 0 e la velocità ritorna positiva. Alla fine delle condizioni avviamo il servo motor tramite il metodo avviaServo(). Questi passaggi vengono ripetuti all’infinito, gestiti tramite i costrutti if ed else if e i valori delle velocità vengono modificati con il metodo “settaVelocita()”.

Codice

```
#include "ServoMotore.h"

//Inizializza un oggetto di tipo Potenziometro di nome potenziometro.
ServoMotore* servoMotore;

int velocita = 20;

void setup() {
  //Istanzia la variabile servoMotor assegnando il pin nelle parentesi (in questo caso
il pin 3) e
  //dopo la virgola la volecita iniziale.
  servoMotore = new ServoMotore(3, velocita);
}

void loop() {
  //Verifica se il servo motore si trova a una posizione uguale o supriore di 180
gradi, se si entra nell'if.
  if(servoMotore->ottieniPosizione() >= 180){
     //Setta la velocità al servoMotore al negativo (cosi torna indietro).
    servoMotore->settaVelocita(-velocita);
  //Verifica se il servo motore si trova a una posizione uguale o inferiore di 0 gradi,
se si entra nell'else if.
  }else if(servoMotore->ottieniPosizione() <= 0){
    //Setta la velocità al servoMotore in positivo (cosi va avanti).
    servoMotore->settaVelocita(velocita);
  }
  //Avvia il servo motore.
  servoMotore->avviaServo();
}
```

Libreria 3 – Ultrasuoni

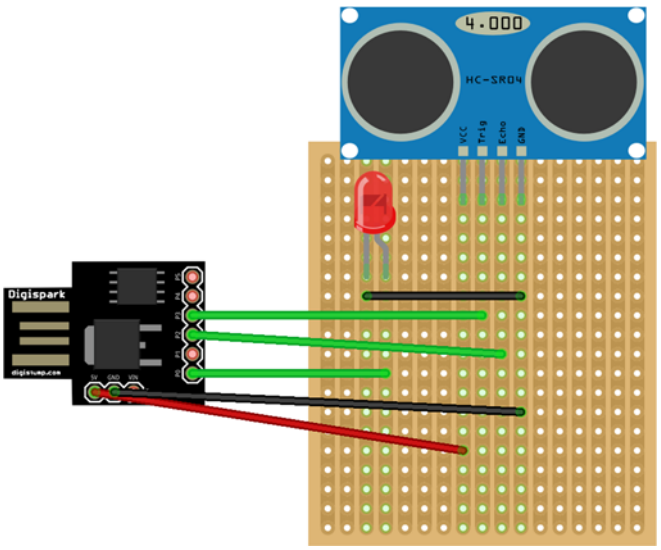
La terza libreria che abbiamo realizzato è per l'ultrasuoni. Abbiamo inoltre creato un metodo "distanzaLetta()" per ritornare la distanza letta dal ultrasuoni.

Materiale

- 1 Arduino Digispark
- 1 Sensore ad Ultrasuoni HC-SR04
- 1 Led

Montaggio del circuito

Per facilitare la montatura del circuito aiutarsi con l'immagine qua sotto. Per prima cosa prendere il proprio Digispark e inserire un cavo rosso nel pin indicato da "5V". Questo cavo deve giungere sulla breadboard come da immagine ed essere infilato in un foro presente sulla stessa riga del "VCC" dell'Ultrasuoni. Collegare il cavo nero nel pin accanto al "5V", che sarà contrassegnato da "GND" in un foro presente sulla stessa riga del "GND" dell'Ultrasuoni. Nel pin P0 del Digispark inserire un cavo che deve collegarsi in un foro sulla stessa riga della parte più a destra del led (quindi il "+" del led). Nel pin P2 aggiungere un cavo che deve collegarsi a un foro presenta sulla stessa riga del "Echo" dell'Ultrasuoni. Un ultimo cavo deve venire collegato al pin P3 e deve giungere in un foro presenta sulla riga del "Trig" dell'Ultrasuoni.



Struttura

Per realizzare questa libreria abbiamo un'interfaccia chiamata "Ultrasuoni.h", in cui definiamo "Ultrasuoni_h" se esso non è ancora definito, e la libreria "Ultrasuoni.cpp" che estende l'interfaccia. Entrambi i files includono "Arduino.h".

Metodi

- ***Ultrasuoni(int echo, int trig)***; Questo metodo costruttore riceve come parametro il pin a cui è collegato il ServoMotore e la velocità di movimento dell'elica. Fatto ciò esso si occupa d'impostare il pin di scrittura del valore da inviare al ServoMotore.
- ***distanzaLetta()***; Questo metodo trasforma la distanza che l'ultrasuoni legge in un valore intero in centimetri e lo ritorna.

Esempio 1 – Base

Il primo esempio realizzato per questa libreria accende il led se la distanza letta dal sensore è maggiore di 50, altrimenti lo spegne. Per cominciare andiamo a includere la nostra libreria chiamata “Ultrasuoni.h”. In un secondo momento inizializziamo un oggetto associato alla classe Ultrasuoni, di nome ultrasuoni e successivamente dichiariamo tre variabili. La prima è “trigPin”, che conterrà un valore intero (senza virgola), di default 2, che indica il pin trig dell’Ultrasuoni. La seconda è “echoPin”, che conterrà un valore intero (senza virgola), di default 3, che indica il pin echo dell’Ultrasuoni. La terza è “ledPin”, che conterrà un valore intero (senza virgola), di default 0, che indica il pin trig del led. Nel metodo setup istanziamo la variabile ultrasuoni assegnando l’echo pin nelle parentesi (in questo caso il pin 3) e come secondo parametro (dopo la virgola) passiamo il pin trig che nel nostro caso sarà il pin 2. Nel metodo loop leggiamo la distanza dal sensore ultrasuoni tramite il metodo “distanzaLetta()”. In seguito controlliamo se la distanza è maggiore di 50 centimetri, in quel caso accendiamo il led tramite il metodo “digitalWrite(ledPin, HIGH)”. Se invece la distanza è minore di 50 centimetri spegniamo il led con il metodo “digitalWrite(ledPin, LOW)”.

Codice

```
#include "Ultrasuoni.h"

//Inizializza un oggetto di tipo Ultrasuoni di nome ultrasuoni.
Ultrasuoni* ultrasuoni;
//Viene inizializzata il pin trig del Ultrasuoni.
int trigPin = 2;
//Viene inizializzata il pin trig del Ultrasuoni.
int echoPin = 3;
//Viene inizializzata il pin trig del led.
int ledPin = 0;

void setup() {
  //Istanzia la variabile ultrasuoni assegnando il pin di echo come primo valore nelle
parentesi e il pin
  //di trig come secibdi valore delle parentesi
  ultrasuoni = new Ultrasuoni(echoPin, trigPin);
}
void loop() {
  //Viene letta la distanza dal sensore ultrasioni.
  int distance = ultrasuoni->distanzaLetta();
  //Se la distanza è superiore di 50 cm entra nell'if.
  if(distance > 50){
    //Accende il led se la distanza letta dal sensore ultrasuoni è superiore di 50 cm.
    digitalWrite(ledPin, HIGH);
  }else{
    //Spegne il led se la distanza letta dal sensore ultrasuoni è inferiore di 50 cm.
    digitalWrite(ledPin, LOW);
  }
}
```