Sistema didattico per Arduino con libreria per attuatori e relativa documentazione

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

Analisi 4

1.4 Analisi del dominio 4

1.5 Analisi e specifica dei requisiti 4

1.6 Analisi dei costi 7

1.7 Pianificazione 7

1.8 Analisi dei mezzi 8

1.8.1 Software 8

1.8.2 Hardware 8

2 Progettazione 9

2.1 Design degli schemi elettrici 9

2.1.1 Schema elettrico potenziometro 9

2.1.2 Schema elettrico ServoMotore 10

2.1.3 Schema elettirco Ultrasuoni 11

2.2 Design delle librerie 12

2.2.1 Libreria Potenziometro 12

2.2.2 Libreria ServoMotore 13

2.2.3 Libreria Ultrasuoni 14

3 Implementazione 15

3.1 Libreria Potenziometro 15

3.1.1 Schema circuito Potenziometro 15

3.1.2 Creazione libreria 15

3.1.3 Sviluppo libreria Potenziometro 16

3.2 Libreria ServoMotore 17

3.2.1 Schema circuito ServoMotore 17

3.2.2 Sviluppo libreria ServoMotore 18

3.3 Libreria Ultrasuoni 19

3.3.1 Schema circuito Ultrasuoni 19

3.3.2 Sviluppo liberia Ultrasuoni 20

4 Test 21

4.1 Protocollo di test 21

4.2 Risultati test 21

4.3 Mancanze/limitazioni conosciute 21

5 Consuntivo 21

6 Conclusioni 21

6.1 Sviluppi futuri 22

6.2 Considerazioni personali 22

7 Bibliografia 22

7.1 Sitografia 22

8 Allegati 22

# Introduzione

## Informazioni sul progetto

**Titolo**: Sistema didattico per Arduino con libreria per attuatori e relativa documentazione

**Allievi**: Mattia Lazzaroni, impiegato nello svolgimento del progetto.

Mattia Toscanelli, impiegato nello svolgimento del progetto.

**Classe**: I3AC

**Docenti**: Adriano Barchi, Luca Muggiasca, Francesco Mussi, Massimo Sartori

**Sezione scuola**: Scuola arti e mestieri Trevano

**Materia**: Modulo 306 – Progetti

**Data inizio**: 14.11.2018

**Fine**: 08.02.2019

## Abstract

E’ una breve e accurata rappresentazione dei contenuti di un documento, senza notazioni critiche o valutazioni. Lo scopo di un abstract efficace dovrebbe essere quello di far conoscere all’utente il contenuto di base di un documento e metterlo nella condizione di decidere se risponde ai suoi interessi e se è opportuno il ricorso al documento originale.

Può contenere alcuni o tutti gli elementi seguenti:

* **Background/Situazione iniziale**
* **Descrizione del problema e motivazione**: Che problema ho cercato di risolvere? Questa sezione dovrebbe includere l'importanza del vostro lavoro, la difficoltà dell'area e l'effetto che potrebbe avere se portato a termine con successo.
* **Approccio/Metodi**: Come ho ottenuto dei progressi? Come ho risolto il problema (tecniche…)? Quale è stata l’entità del mio lavoro? Che fattori importanti controllo, ignoro o misuro?
* **Risultati**: Quale è la risposta? Quali sono i risultati? Quanto è più veloce, più sicuro, più economico o in qualche altro aspetto migliore di altri prodotti/soluzioni?

Esempio di abstract:

*As the size and complexity of today’s most modern computer chips increase, new techniques must be developed to effectively design and create Very Large Scale Integration chips quickly. For this project, a new type of hardware compiler is created. This hardware compiler will read a C++*

## Scopo

Lo scopo del progetto è quello di realizzare un prodotto didattico dedicato agli allievi di terza media che arriveranno nelle giornate di porte aperte Promtec. Il nostro progetto consentirà all’utente di ambientarsi con la piattaforma elettronica Arduino (nel nostro caso mini DigiSpark) grazie a delle guide semplici ed intuitive che spiegheranno come utilizzarlo e come programmarlo. Per far capire al meglio il concetto di programmazione il prodotto offre delle librerie create da noi contenenti dei metodi a cui sono stati attribuiti nomi semplici ed intuitivi.   
Al nostro gruppo sono stati assegnati i seguenti componenti:  
- Potenziometro  
- ServoMotore  
- Ultrasuoni

## Analisi

## Analisi del dominio

Bisogna trovare un metodo che aiuti i ragazzi di terza media, giunti alla scuola arti e mestieri Trevano durante le porte aperte Promtec, ad approciarsi al mondo della programmazione. All’inzio della giornata lo studente riceverà un DigiSpark (un Arduindo molto piccolo, ma che svolge le stesse funzioni). L’obbiettivo è quello di creare delle librerie per quest’utlimo contenteti metodi che semplificano la stesura del codice. Tutto ciò viene viene accompagnato con dei manuali utente in cui sono scritte dettagliate descrizioni su come funziona la liberiria e su come devono venir utilizzati i vari metodi con tanto di esempi. Alla fine della giornata Il ragazzo tornarà a casa con il DigiSpark, un piccolo manuale d’uso e sicuramente con delle conoscenze in più per quanto riguarda la programmazione.

## Analisi e specifica dei requisiti

Il committente deve ricevere un DigiSpark e un guida per capirne il funzionamento. Il prodotto deve svolgere determinate azioni, come far accendere un led tramite un potenziometro. Queste funzioni sono implementate tramite il software Arduino 1.8.7. L’utente dovrebbe consultare la guida per capire come montare il circuito e come usare il proprio Arduino e le librerie presenti in esso da noi implementate. L’allievo consultando la guida dovrà capire tutto ciò che dovrà fare tramite una spiegazione semplice ed efficace, anche se egli non si è mai applicato prima nell’elettronica. Come minimo il prodotto deve far si che tutte le librerie presenti funzionino e raggiungano il risultato atteso al 100%, senza bug o mancanze. Ovviamente il circuito dovrà essere sicuro e senza la presenza di rischi che possano ferire un visitatore. Quando l’utente avrà finito la sua attività avrà la possibilità di tenere il lavoro da lui svolto come ricordo della sua visita.

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Saldamento dei pin DigiSpark |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | Si necessita un modulo da 6 pin femmina. |
| **002** | Si necessita di un saldatore. |
| **003** | Si necessita dello stagno. |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Creazione delle libreria Potenziometro |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Per poter implementare le librerie bisogna completare il requisitio REQ-001. |
| **Sottorequisiti** | |
| **001** | Il codice deve essere ben indentato. |
| **002** | Il codice deve essere commentato. |
| **003** | I nomi dei metodi devono essere di facile compresione da parte dell’utente. |
| **004** | Si devono creare 3 codici d’esempio sull’utilizzo della libreria. |
| **005** | La libreria deve contenere un metodo che semplifichi la lettura del valore dal Potenziometro. |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Creazione delle libreria ServoMotore |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Per poter implementare le librerie bisogna completare il requisitio REQ-001. |
| **Sottorequisiti** | |
| **001** | Il codice deve essere ben indentato. |
| **002** | Il codice deve essere commentato. |
| **003** | I nomi dei metodi devono essere di facile compresione da parte dell’utente. |
| **004** | Si devono creare 3 codici d’esempio sull’utilizzo della libreria. |
| **005** | La libreria deve contenere dei metodi che semplificano il muovimento dell’elica del ServoMotore. |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Creazione delle libreria Ultrasuoni |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Per poter implementare le librerie bisogna completare il requisitio REQ-001. |
| **Sottorequisiti** | |
| **001** | Il codice deve essere ben indentato. |
| **002** | Il codice deve essere commentato. |
| **003** | I nomi dei metodi devono essere di facile compresione da parte dell’utente. |
| **004** | Si devono creare 3 codici d’esempio sull’utilizzo della libreria. |
| **005** | La libreria deve contenere un metodo che semplifichi la distanza letta dal sensore Ultrasuoni. |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Creazione della guida |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Per poter scrivere la guida bisogna completare i requisiti REQ-002, REQ-003, REQ-004. |
| **Sottorequisiti** | |
| **001** | Bisogna spiegare come installare i dirver della la scheda Digispark. |
| **002** | Deve essere comprensibile a tutti gli utenti. |
| **003** | Deve contenere la spiegazione di tutte le librerie. |
| **004** | Deve avere degli esempi per aiutare la comprensione dell’utente. |

## Analisi dei costi

Ci siamo divisi i lavori:

* Mattia Toscanelli si occupa di creare delle librerie molto semplici in modo da aiutare gli allievi di terza media ed inoltre si occupa di documentare tutti i passaggi eseguiti.
* Mattia Lazzaroni si occupa di generare la guida delle librerie dove vengono spiegati i vari componenti utilizzati e i metodi semplificati ed inoltre si occupa di eseguire i vari test case.

|  |  |  |  |
| --- | --- | --- | --- |
| Categoria | Ore di lavoro | Costo all’ora | Costo totale |
| Manodopera Mattia Lazzaroni | 70 h | 62 CHF | 4340 CHF |
| Manodopera Mattia Toscanelli | 70 h | 62 CHF | 4340 CHF |

## Pianificazione

Abbiamo inserito due pietre miliari, una a fine progettazione, cioè quando il progetto inizierà ad essere concreto; l’altra a fine progetto, quando potrà essere consegnato al cliente. La fase di implementazione sarà quella in cui dedicheremo più tempo, in particolare l’attività “Sviluppo librerie”, in quanto necessita di maggiore cura e inoltre per funzionare correttamente richiede di effetturare molti test.

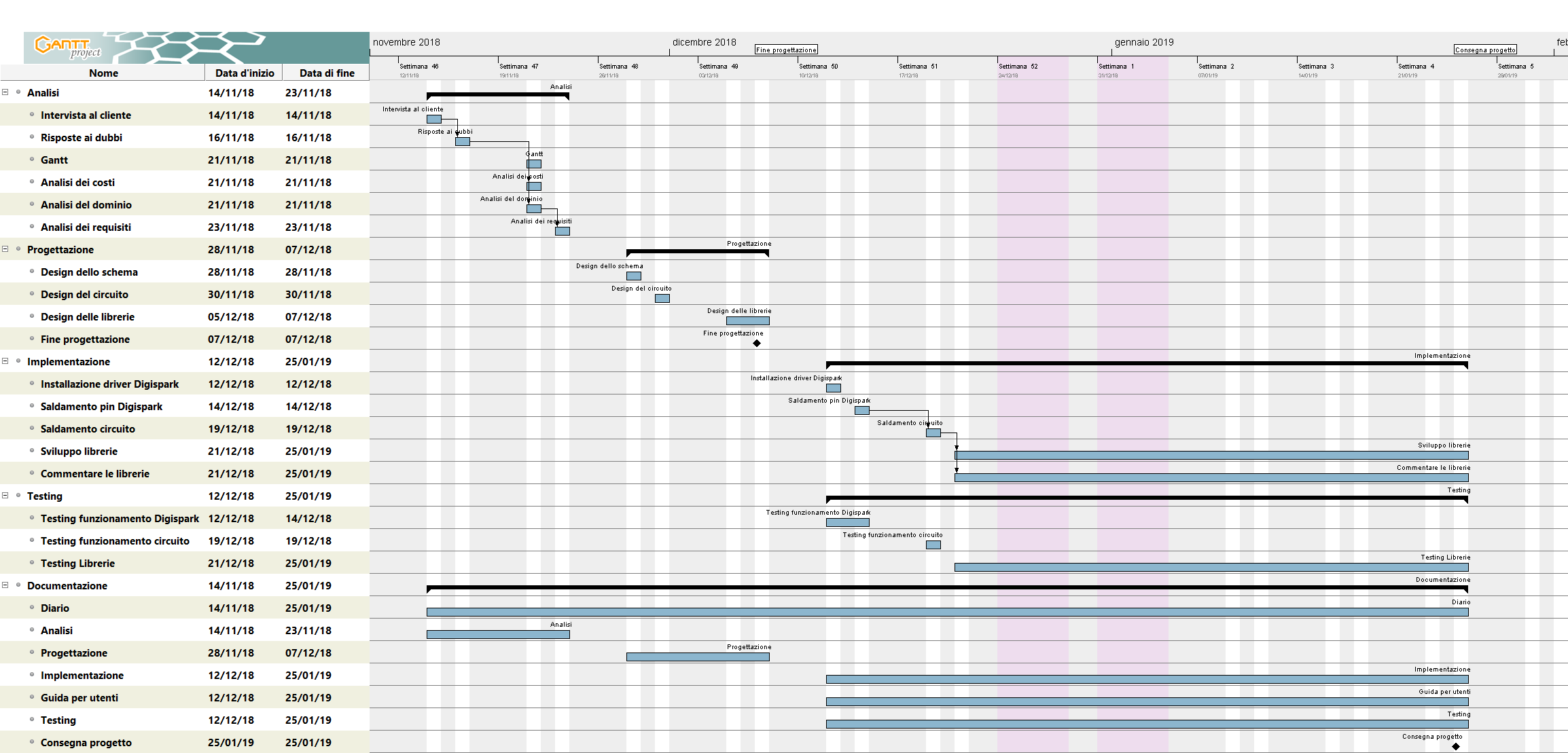


Figura 1, Gannt di pianificazione.

## Analisi dei mezzi

Per la realizzazione di questo progetto si ha bisogno di:

* Un arduino o simile
* Un PC con performance in grado di far girare il software di Arduino.

### Software

I software utilizzati per la realizzazione di questo progetto sono:

* Word 2016
* Fritzing 0.9.3b
* Arduino 1.8.7
* Power Point 2016
* GanttProject 2.8.9
* Google Chrome

### Hardware

I componenti hardaware utilizzati per la realizzazione del progetto sono:

* Digispark USB Development Board
* Caratteristiche computer Mattia Toscanelli:
  + Modello: Huawei Matebook X Pro
  + Processore: i7 8550u
  + RAM: 8GB LPDDR3 2133 MHz
  + GPU: NVIDIA MX150
  + SSD: 512 GB
* Caratteristiche computer Mattia Lazzaroni:
  + Modello: Acer Aspire E 15
  + Processore: i7 7500u
  + RAM: 16GB LPDDR4 2133 MHz
  + GPU: NVIDIA 940MX
  + SSD: 256 GB
* Digispark USB Development Board
  + Alimentazione tramite USB 5V oppure da sorgente estera 7-35V (conisgliati 12V)
  + Regolatore On-board da 5V 500mA
  + Porta di collegamento a computer USB
  + 6 pin I/O (se il programma comunica attivamente tramite USB 2 pin vengono utilizzati per la porta USB, altrimenti è possibile utilizzare tutte 6, anche se si sta programmando via USB)
  + Memoria Flash da 8k (circa 6k dopo il bootloader)
  + Pin di I2C e SPI
  + 3 pin PWM (altri possibili con il software PWM)
  + 4 pin ADC
  + LED di alimentazione
  + LED di test/stato

# Progettazione

## Design degli schemi elettrici

### Schema elettrico potenziometro

Il funzionamento di questo circuito è molto semplice. Per farlo si necessita un diodo rosso, una resistenza da 180Ω e un potenziometro da 10kΩ. Lo scopo di questo circuito è il cambiamento di stato del LED grazie alla rotazione del potenziometro. Per effetture questo circuito bisogna collegare il terminale 3 del potenziometro (vedi Figura 2) a un pin analogico del Digispark (in questo caso il P2). Per avere un cambiamento di luminosità del LED bisogna collegare anch’esso a un pin analogico, nel nostro caso al pin P0.

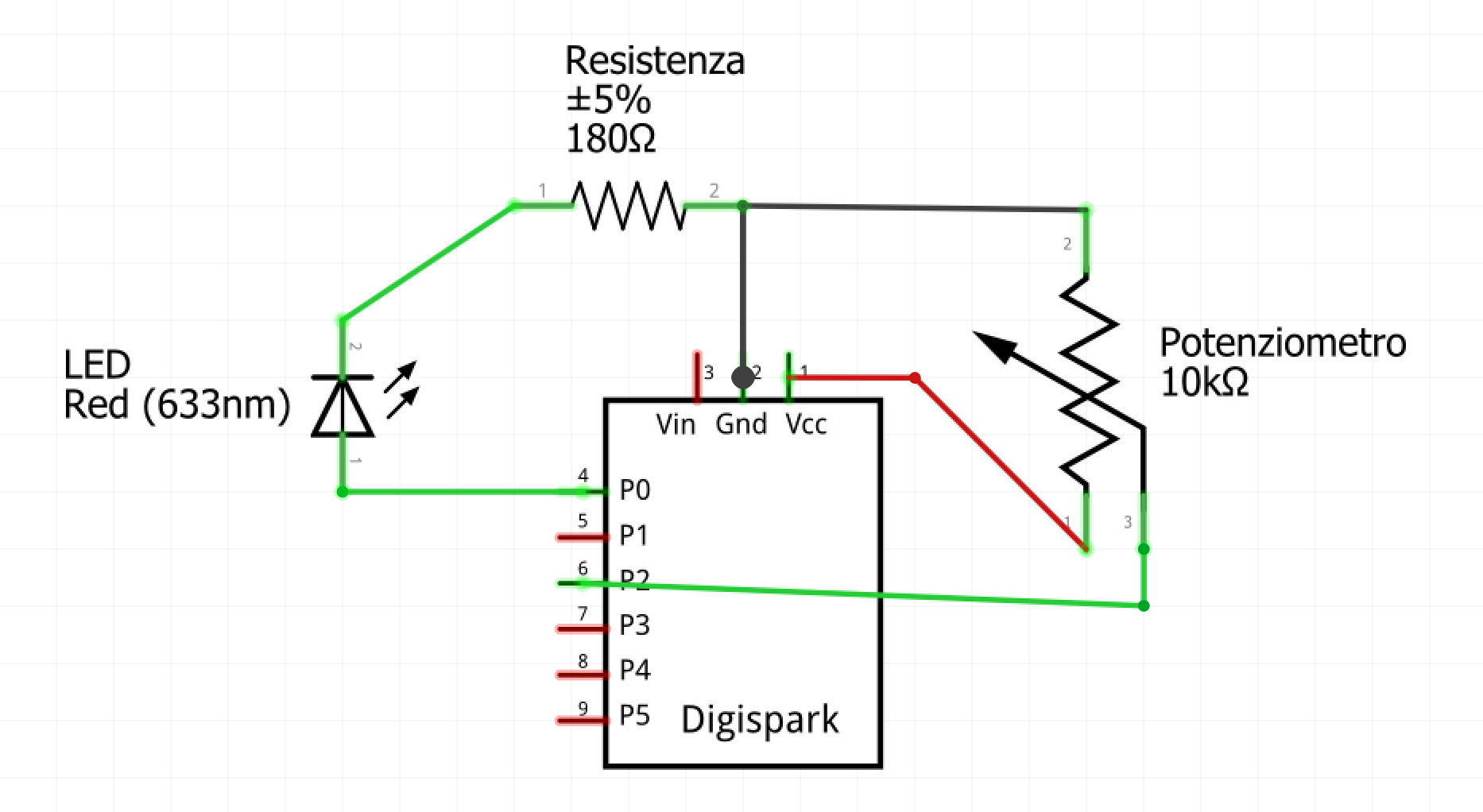


Figura , Schema elettrico del Potenziometro

### Schema elettrico ServoMotore

Per fare questo circuito si necessita un pulsante e un ServoMotore. Lo scopo di questo circuito è il muovimento del Servomotore grazie alla pressione o meno del pulsante. Per effetture questo circuito bisogna collegare il terminale di pulse del ServoMotore (vedi Figura 3) a un pin analogico del Digispark (in questo caso il pin P3). In seguito bisogna collegare a un pin del Digispark uno dei due terminali 1 del pulsante (vedi Figura 3) per capire quando fa cambiare lo stato del ServoMotore.

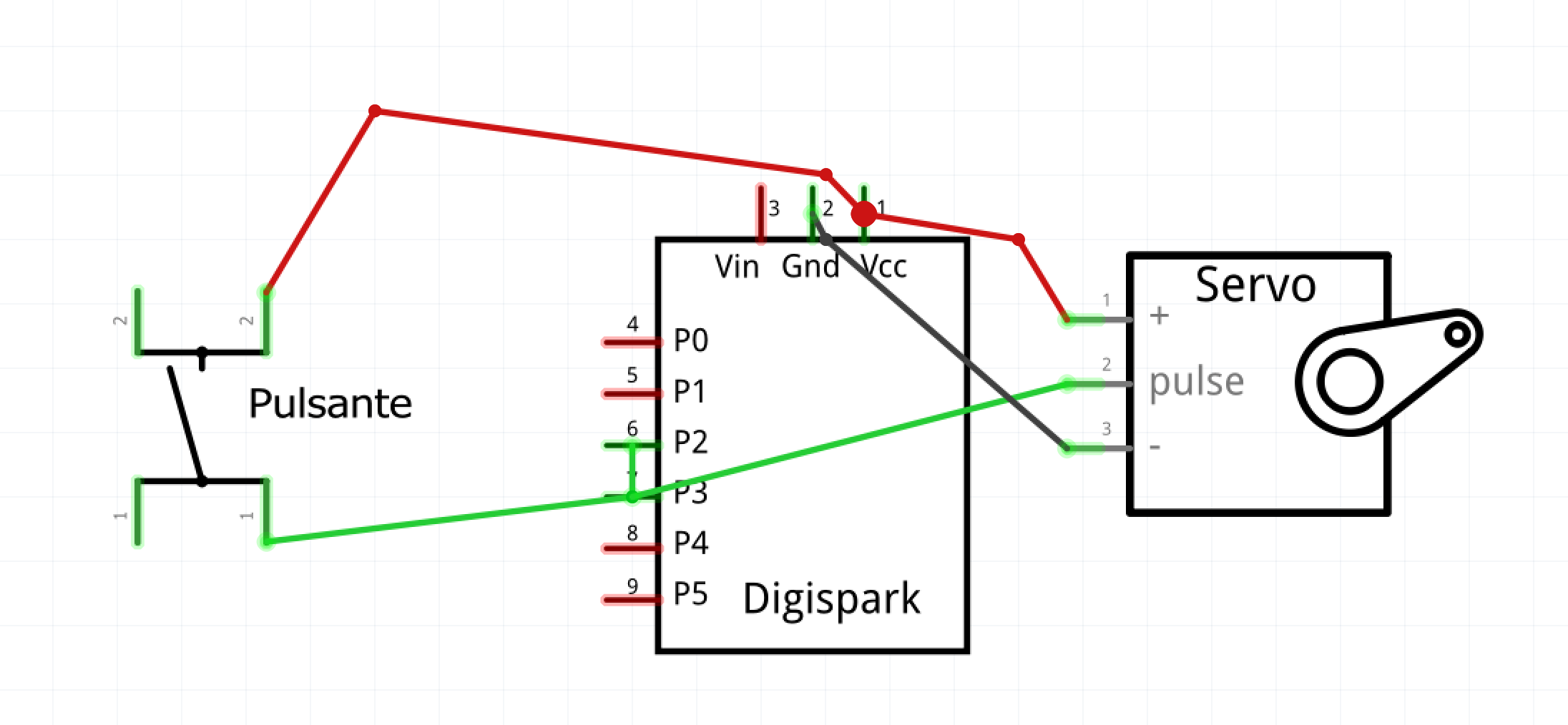


Figura 3, Schema elettrico del ServoMotore

### Schema elettirco Ultrasuoni

Per fare questo circuito si necessita un sensore Ultrasuoni e un LED. Lo scopo di questo circuito è il cambiamento di stato del LED in base a quello che legge il sensore Ultrasuoni. Per effetture questo circuito bisogna collegare il terminale di echo dell’Ultrasuoni (vedi Figura 4) a un pin analogico del Digispark (in questo caso il pin P2). In seguito bisogna collegare il pin di trig ad qualsiasi pin del Digispark (in questo caso il pin P3).

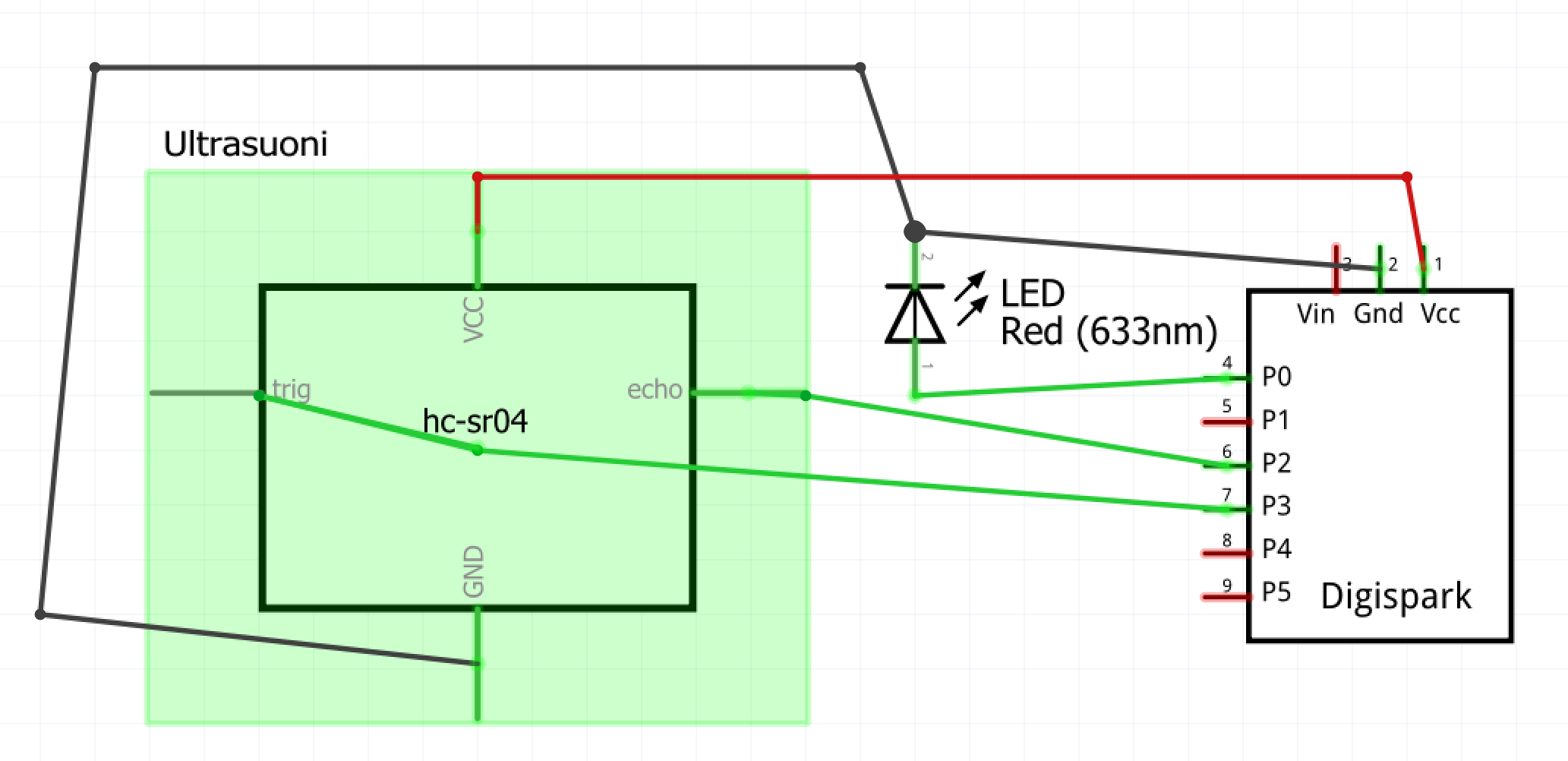


Figura 4, Schema elettrico dell’Ultrasuoni

## Design delle librerie

### Libreria Potenziometro

Questa libreria semplica l’utilizzo di un Potenziometro. Contiene un metodo che permette la lettura del Potenziometro in modo rapito e in modo più semplificato.

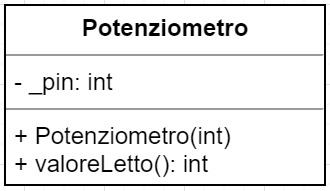


Figura 5, Libreria del potenziometro

|  |  |  |
| --- | --- | --- |
| Nome | Tipo | Descrizione |
| \_pin | Variabile | L’attributo \_pin conterrà il numero del pin a cui sarà collegato il terminale di lettura del potenziometro. |
| Potenziometro(int) | Metodo costruttore | È il metodo costruttore della classe Potenziometro. È composto da un solo paramertro e quest’ultimo corrisponde al terminale di lettura del potenziometro. |
| valoreLetto(): int | Metodo | Questo metodo ritorna il valore letto dal pin analogico in cui è collegato il terminale del potenziometro. Il valore viene letto e covetito in un intervallo che parte da 0 fino a 255. |

### Libreria ServoMotore

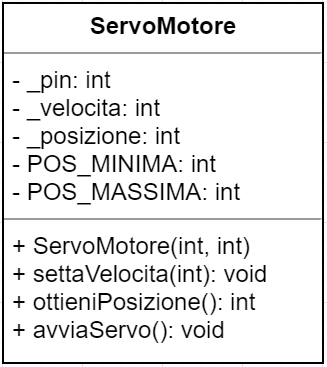
Questa libreria semplica l’ultizzo del ServoMotore. Contiene dei metodi che ottimizzano l’utilizzo del ServoMotore, come il settaggio della velocita e l’ottenimento della posizione.

Figura 6, Libreria del ServoMotore

|  |  |  |
| --- | --- | --- |
| Nome | Tipo | Descrizione |
| \_pin | Variabile | L’attributo \_pin conterrà il numero del pin a cui sarà collegato il terminale di lettura del ServoMotore. |
| \_velocita | Variabile | L’attributo \_velocita conterrà la velocita del ServoMotore. Questo valore può variare da -100 a 100. |
| \_posizione | Variabile | L’attributo \_posizione conterrà la posizione dell’elica del ServoMotore. Questo valore parte da 0° fino a 180° |
| POS\_MINIMA | Variabile costante | L’attibuto POSIZIONE\_MIMIMA conterrà la posizione minima dell’elica del ServoMotore (cioè la posizione corrispondente a 0°). |
| POS\_MASSIMA | Variabile costante | L’attibuto POSIZIONE\_MASSIMA conterrà la posizione massima dell’elica del ServoMotore (cioè la posizione corrispondente a 180°). |
| ServoMotore(int,int) | Metodo costruttore | È il metodo costruttore della classe ServoMotore. È composto da due parametri: il primo esprime il pin a cui è collegato il ServoMotore e il secondo esprime la velocità dell’elica. |
| settaVelocita(int) | Metodo | Questo metodo permette si settare la velocità al ServoMotore. All’interno di esso viene controllata la velocità passata come parametro ed eventualemente viene corretta. |
| ottieniPosizione() | Metodo | Questo metodo ritorna la posizione dell’elica del ServoMotore. |
| avviaServo() | Metodo | Questo metodo fa muovere il ServoMotore. |

### Libreria Ultrasuoni

Questa libreria semplifica l’utlizzo dell’Utrasuoni. Contiene un metodo che permettere di sapere velocemente la velocità letta dall’Ultrasuoni.

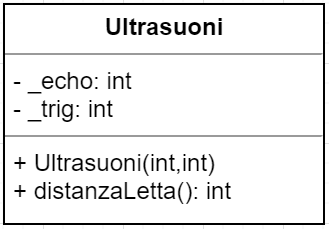


Figura 7, Libreria dell’Ultrasuoni

|  |  |  |
| --- | --- | --- |
| Nome | Tipo | Descrizione |
| \_echo | Variabile | L’attributo \_echo conterrà il numero del pin a cui sarà collegato il terminale di echo. |
| \_trig | Variabile | L’attributo \_echo conterrà il numero del pin a cui sarà collegato il terminale di trig. |
| Ultrasuoni(int, int) | Metodo costruttore | È il metodo costruttore della classe Ultrasuoni. È composto da due parametri: il primo esprime il pin di echo e il secondo il pin di trig. |
| distanzaLetta(): int | Metodo | Questo metodo ritorna la ditanza letta dall’ultrasuoni e quest’ultima viene convertita in centimetri. |

# Implementazione

## Libreria Potenziometro

### Schema circuito Potenziometro

La prima libreria che ci è stata assegnata è quella del potenziometro. Inanzitutto ho preparato con Fritzing lo schema del circuito.

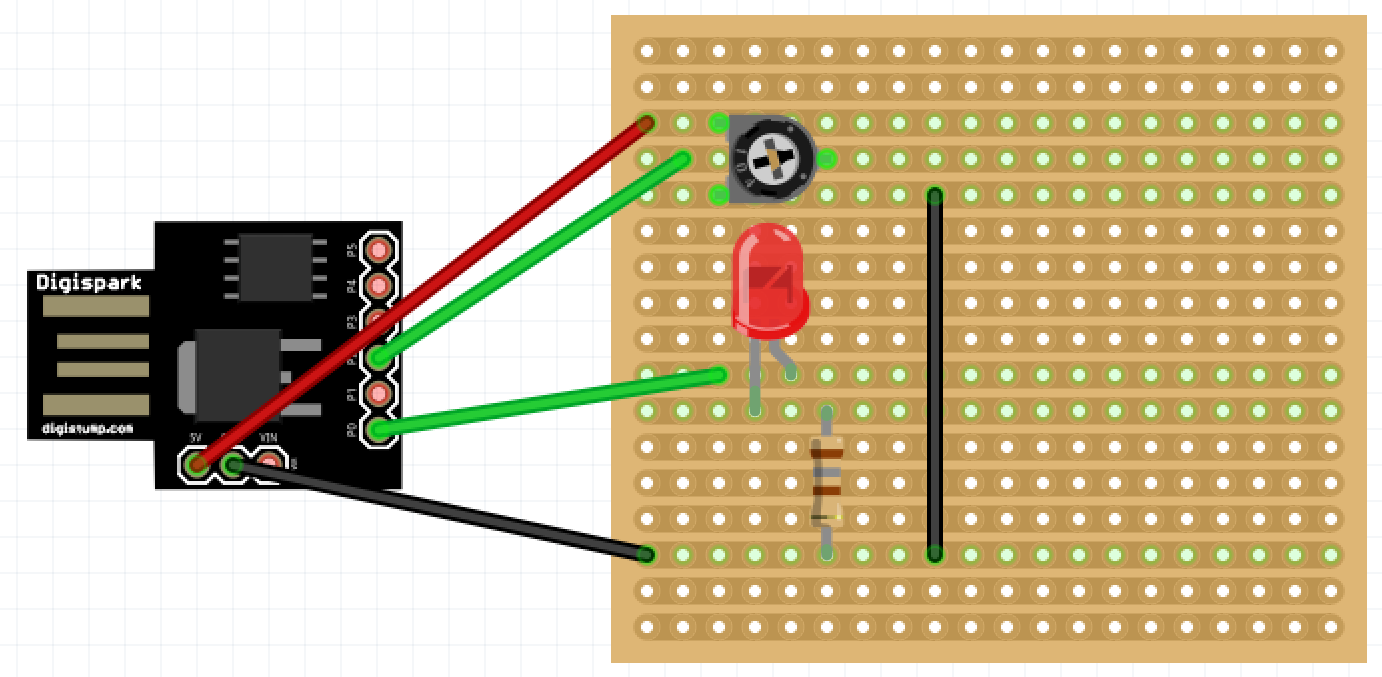


Figura 8, Schema del circuito con potenziometro e led.

Per iniziare abbiamo collegato il VCC del Digispark al terminale superiore del potenziometro, mentre il terminale inferiore del potenziomerto è collegato ad un cavo che giunge alla massa. Il pin 2 del Digispark è collegato al terminale centrale del potenziometro così da poter leggere il suo valore in modo analogico. Successivamente abbiamo congiunto il pin 0 del Digispark al anodo del led rosso, mentre il catodo giunge alla resistenza di 180Ω che a sua volta si reca alla massa.

### Creazione libreria

Essendo che non avevamo mai scritto una liberia per arduino ci siamo documentati su come la si faceva. Abbiamo scopero che per effettuarle bisogna creare due file distinti per ognuna. Il primo di questi è un .h e ha la funzione di esprimere le varibili e le funzioni per il secondo tipo di file, come una vera e propria interfaccia. Il secono file è un .cpp, in questo file dobbiamo includere l’interfaccia definta dal .h e “riempire” i metodi che poi verranno utilizzati. Questi due file vengono scritti con il linguaggio C.

Per includere un .h bisogna fare nel seguente modo:

#include "Potenziometro.h"

### Sviluppo libreria Potenziometro

Abbiamo creato come prima cosa il file Potenziometro.h e abbiamo stabilito che metodi sarebbero stati utili per la semplificazione dell’utilizzo del componente. Questo è il risultato:

#ifndef Potenziometro\_h

#define Potenziometro\_h

#include "Arduino.h"

class Potenziometro{

  public

    Potenziometro (int pin);

int valoreLetto();

  private:

    int \_pin;

};

#endif

La libreria è composta da 2 metodi pubblici e un singolo attributo privato. L’attributo \_pin contiene il pin a cui è collegato il digispark e quindi dove verrà letto il valore del potenziometro.

Il metodo costruttore ha un singolo parametro e questo specifica appunto il numero del pin a cui viene collegato il potenziometro. Il pin devere essere di tipo analogico e essendo che si dovrà leggere si dovrà fare attenzione alla mappatura di Digisprark in quanto è diversa da quella classica, infatti il pin2 corrisponde al 1, il pin3 corrisponde al 3, il pin4 corrisponde al 2 e il pin5 corrisponde al 0. Questo viene gesito dentro al metodo costruttore ridefinito nel file .ccp in questo modo:

if(pin == 2){

    \_pin = 1;

}else if(pin == 4){

    \_pin = 2;

}else if(pin == 5){

    \_pin = 0;

}else{

    \_pin = 3;

}

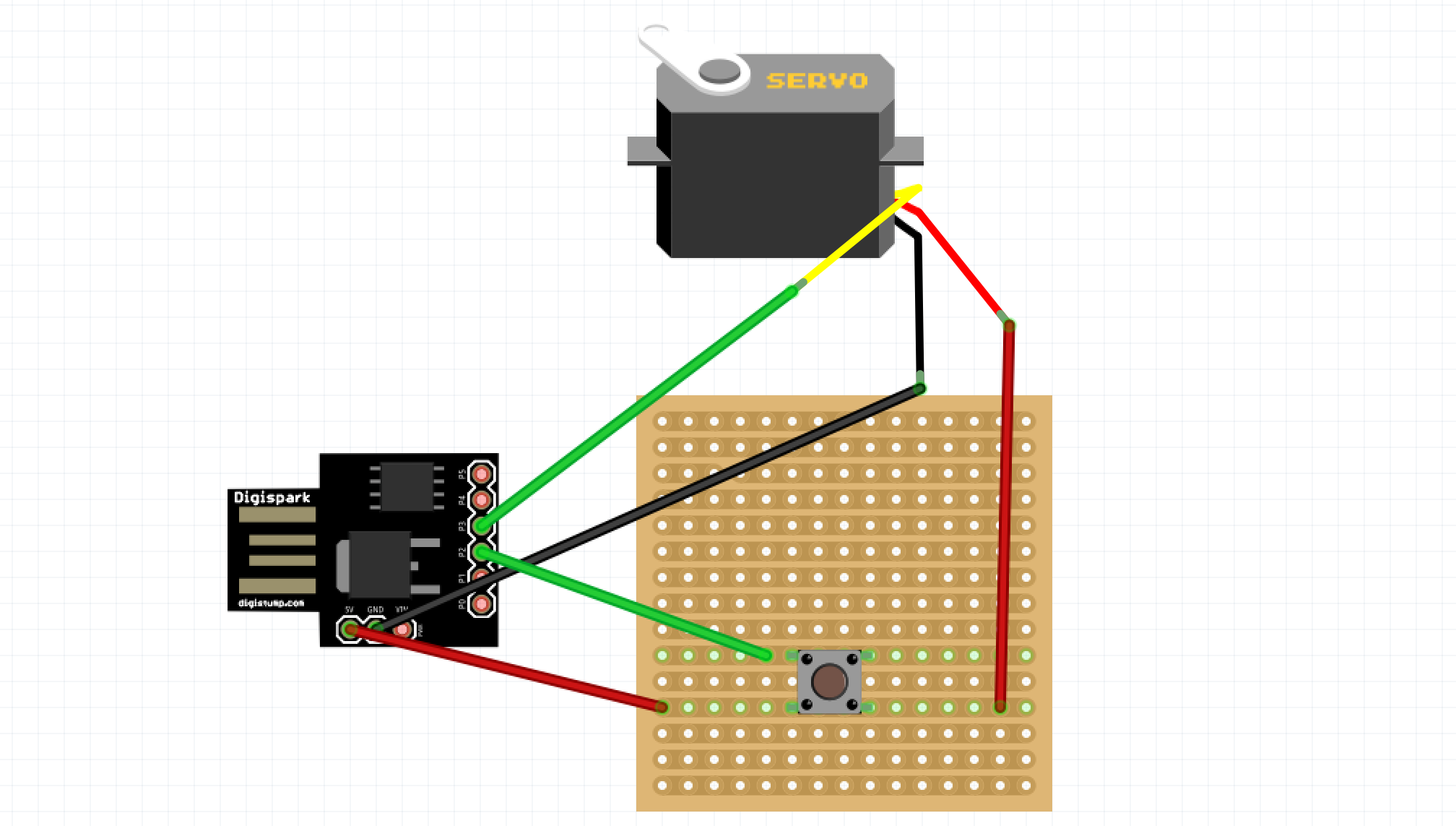
Infine c’è il metodo valoreLetto() che semplifica la lettura del potenziometro, infatti i valori analogici solitamente vanno da 0 a 1023 e questo metodo ristabilisce il range che parte da 0 e arriva fino a 255.

## Libreria ServoMotore

### Schema circuito ServoMotore

La seconda libreria che ci è stata assegnata è quella del ServoMotore. Come nel primo caso abbiamo preparato lo schema del circuito con Fritzing.

Figura 9, Schema del circuito con ServoMotore



Per iniziare abbiamo collegato il VCC del Digispark ai terminali inferiori del pulsante e al cavo di alimentazione rosso del ServoMotore, mentre al GND del Digispark abbiamo collegato il cavo di massa nero del ServoMotore. Il pin 2 del Digispark è collegato ai terminali superiori del pulsante così quanto quest’ultimo viene premuto o rilasciato si può leggere il suo valore in modo digitale. Successivamente abbiamo congiunto il pin 3 del Digispark al cavo di scrittura del ServoMotore in modo analogico.

### Sviluppo libreria ServoMotore

Abbiamo creato come per la libreria del potenziometro il file contente l’intefaccia del .cpp dove abbiamo stabilito i metodi che sarebbero stati utili per la semplificazione dell’utilizzo del componente. Questo è il risultato:

#ifndef ServoMotore\_h

#define ServoMotore\_h

#include "Arduino.h"

class ServoMotore {

public:

ServoMotore(int pin, int velocita);

void settaVelocita(int velocita);

int ottieniPosizione();

void avviaServo();

private:

int \_pin;

const int POS\_MINIMA = 544;

const int POS\_MASSIMA = 2400;

int \_velocita;

int \_posizione;

};

#endif

A differenza della prima libreria che abbiamo creato questa seconda è stata quello che ci ha richiesto più tempo in quanto più complessa. Esisteva gia una libreria che permetteva l’utilizzo di questo componete però come ci è stato detto da un supervisore abbiamo provveduto a crearne una tutta nostra. La libreria è composta da 4 metodi pubblici e 5 attributi privati. Partendo con gli attiributi, la variabile \_pin, come nella precedente libreria del potenziometro, conterrà il pin a cui è collegato il cavo di scrittura del ServoMotore. Questo deve essere analogico ma in questo caso non si eseguirà la mappatura descritta nel capitolo 3.1.2 essendo che bisognerà scirvere e non leggere. Poi abbiamo creato due variabili costatanti (POS\_MINIMA, POS\_MASSIMA) che esprimono, come dice il termine, la posizione minima dell’elica del ServoMotore e viceversa. Poi abbiamo creato una variabile \_velocita in cui sarà descitta la velocità dell’elica e per fine una variabile \_posizione dove appunto conterrà la posizione dell’elica.

Passando con i metodi, il costruttore ha due parametri, cioè il pin a cui è collegato il cavo di scrittura del ServoMotore e la velocità dell’elica da settare come partenza ed infine viene settata anche la posizione iniziale (quella minima).

Poi abbiamo creato il metodo settaVelocità(int) in cui si spuò settare appunto la velocità dell’elica del ServoMotore nel corso del programma. Inoltre all’interno del metodo vengono eseguiti dei controlli sul valore passato come parametro ed eventualmente corretti con un numero che va da -100 a 100.

Come penultimo metodo abbiamo creato ottientiPosizione() che ritorna la posizione dell’elica in gradi (da 0 a 180), come descritto nell’istruzione qui di seguito:

return map(\_posizione, POS\_MINIMA, POS\_MASSIMA, 0, 180);

In pratica grazie alla funzione map() possiamo ridefinire il range in uno più accettabile.

Infine abbiamo creato il metodo avviaServo() che appunto sarà l’istuzione che fara muovere il ServoMotore con le caratteristiche settate precedentemente.

## Libreria Ultrasuoni

### Schema circuito Ultrasuoni

L’ultima libreria che ci è stata assegnata è quella dell’Ultrasuoni. Come nelle altre due libreria abbiamo preparato come prima cosa lo schema in Fritzing

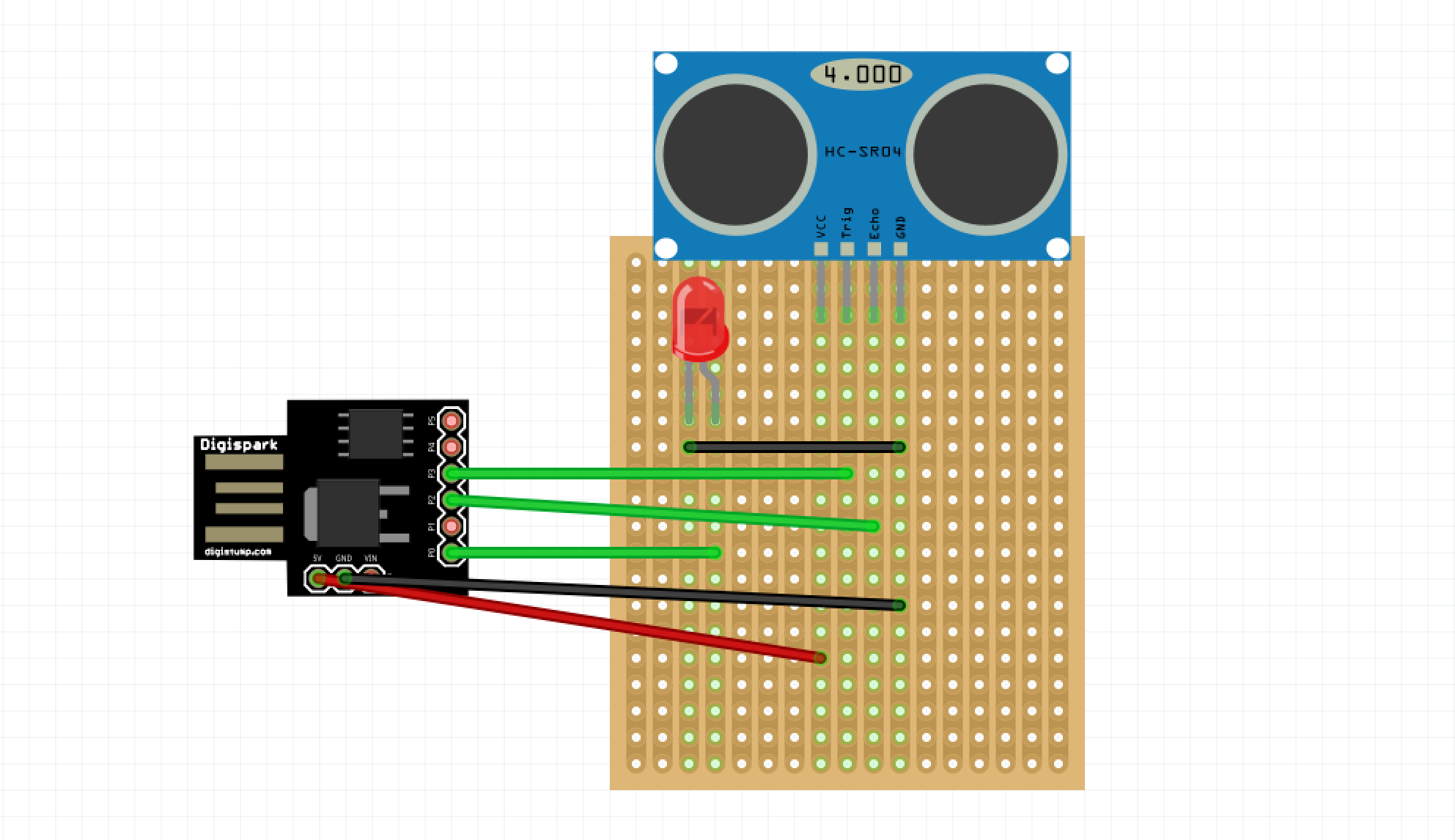


Figura 0, Schema del circuito con Ultrasuoni

Per iniziare abbiamo collegato il VCC del Digispark al terminale VCC dell’Ultrasuoni, mentre abbiamo collegato il pin di GND del Digispark al terminale GND dell’Ultrasuoni e al cadoto del LED. Il pin 2 digitale del Digispark è collegato al terminale di trig dell’Ultrasuoni, questo serve per inviare un impulso alto per almeno 10 microsecondi cosicchè il sensore invii il ping sonoro e aspetti il ritorno delle onde riflesse. Per poi rispondere sul terminale di Echo del sensore, collegato al pin 2 sul Digispark, con un impulso alto della durata corrispondente a quella di viaggio delle onde sonore.Infine abbiamo collegato al pin 0 del Digispark l’anodo del LED il quale cambiare stato in funzione della lettura del sensore Ultrasuoni.

### Sviluppo liberia Ultrasuoni

Abbiamo creato come prima cosa il file Ultrasuoni.h e abbiamo stabilito che metodi sarebbero stati utili per la semplificazione dell’utilizzo del componente. Questo è il risultato:

#ifndef Ultrasuoni\_h

#define Ultrasuoni\_h

#include "Arduino.h"

class Ultrasuoni {

public:

Ultrasuoni(int echo, int trig);

int distanzaLetta();

private:

int \_echo;

int \_trig;

};

#endif

La libreria è composta da 2 metodi pubblici e un 2 attributi privati. A differenza delle due librerie precedenti in questa libreria teniamo conto di due pin e non uno solo, infatti nella variabile \_echo verrà contenuro il numero del pin a cui è collegato il terminale di echo del sensore ultrasuoni e nella variabile \_trig verrà contenuro il numero del pin a cui è collegato il terminale di trig.

Il metodo costruttore ha due parametri, il pin di echo e il pin di trig i quali specificano appunto il numero dei pin a cui vengono collegati.

Infine abbiamo creato il metodo distanzaLetta() che ha questa serie di istruzioni:

digitalWrite(\_trig, LOW);

delayMicroseconds(2);

digitalWrite(\_trig, HIGH);

delayMicroseconds(10);

digitalWrite(\_trig, LOW);

long durata = pulseIn(\_echo, HIGH);

return (durata \* 0.034 / 2);

In pratica viene mandato l’impulso di tirgger, cioè quello descritto nel capitoletto precedente, ed in seguito viene letta la durata del viggio delle onde sonore, per poi essere divisa per 2 (avanti + indietro).

## Esempi di utilizzo librerie

Per conculudere abbiamo creato per ogni libreria una serie di 3 esempi, cosi da avere un totale di 9 esempi. Ogni gruppo della libreria presenta un esempio semplice, uno intermendio e per finire uno più difficile. Questi non ci hanno creato particolarmente problemi in quanto per persone del nostro livello sono cose abbastanza basilari. Il primo esempio di ogni libreria viene descritto nel manuale utente.

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Verifica di funzionamento della scheda Digispark |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Verifica di funzionamento della scheda Digispark |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Verifica di funzionamento della scheda Digispark |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Verifica di funzionamento della scheda Digispark |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

|  |  |
| --- | --- |
| Nome Requisito | Risultato |
| TC-001 |  |
| TC-002 |  |
| TC-003 |  |
| TC-004 |  |

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap 1.7) (ad esempio Gannt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? Ecc

# Bibliografia

## Sitografia

* <https://www.adrirobot.it/arduino/digispark/digispark.htm>, *Scheda Digispark*, 30-11-2018.
* <https://github.com/digistump/DigistumpArduino/releases/download/1.6.7/Digistump.Drivers.zip>, [*Digistump Arduino Release 1.6.7*](https://github.com/digistump/DigistumpArduino/releases/tag/1.6.7)*, 30-11-2018.*
* <https://www.instructables.com/id/Digispark-DIY-The-smallest-USB-Arduino/>, *Digispark DIY: the Smallest USB Arduino, 07-12-2018*
* <http://digistump.com/wiki/digispark>, *Getting Started with your Digispark or Digispark Pro, 12-12-2018.*
* <http://fritzing.org/download/>, *Fritzing, 12-12-2018.*
* <http://www.radiofo.it/files/articoli/sku207366_programmare_att.pdf>, *Scheda tecnica Attinity85*

*31-01-2010*

* <http://www.dmf.unisalento.it/~denunzio/allow_listing/ARDUINO/HC-SR04-GDN.pdf>, *Manuale s’uso Sensore Ultrasuoni HC-SR04, 02-02-2019*

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o Qdc
* Prodotto
* …