Scheda d’utilizzo

Una guida per introdurre I giovani nel mondo di arduino

DI Mattia Lazzaroni & Mattia toscanelli

2019

Indice

[Scopo 2](#_Toc536188676)

[Arduino Digispark 2](#_Toc536188677)

[Componenti 2](#_Toc536188678)

[Potenziometro 2](#_Toc536188679)

[Esempio 1 – Fade 2](#_Toc536188680)

[Esempio 2 – Blink 3](#_Toc536188681)

[Esempio 3 – FadePro 3](#_Toc536188682)

[Servo Motor 3](#_Toc536188683)

[Esempio 1 - ServoMotorStart 4](#_Toc536188684)

[Esempio 2 - ServoMotoreIncrement 4](#_Toc536188685)

[Esempio 3 - ServoMotoreIncrement 5](#_Toc536188686)

[Ultrasuoni 5](#_Toc536188687)

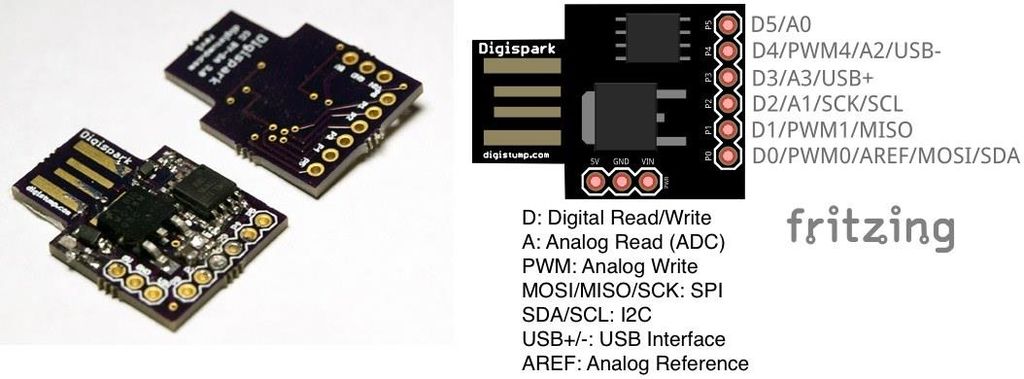
# Scopo

Lo scopo di questo manuale è quello di facilitare la comprensione dell’utente riguardo i circuiti e le librerie da noi implementate, che sono:

1. Potenziometro
2. Servo Motore
3. Ultrasuono

# 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. Digispark dà la possibilità di usare l’IDE (software per la programmazione) di Arduino, che è gratuito ed 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 secodo pin



# Componenti

## Potenziometro

La prima libreria che abbiamo realizzato è per il potenziometro, questa presenta due metodi pubblici. Il primo è “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. Il pin 2 viene impostato a 1, il 4 a 2, il 5 a 0 e il 3 rimane 3. Il secondo metodo, chiamato “valoreLetto()” non riceve nessuna parametro e ritorna un int. La prima cosa che fa è impostare la variabile “sensorValue” al valore letto dal pin analogico del potenziometro specificato tramite il costruttore. Successivamente la variabile viene divisa per 4, questo perché il valore del potenzimetro va da 0 a 1023 mentre quello del led va da 0 a 255. Il metodo ritorna poi questa variabile che sarà il valore del led. Quindi nel caso che il potenziometro venga roteato al punto da possedere una valore di 1023, la variabile contenete il valore verrà divisa per 4 e raggiungerà 255, che sarà poi il valore ritornato dal metodo e che imposterà il led alla luminoosità massima.

### Esempio 1 – Fade

Il primo esempio realizzato fa 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 di istruzioni che viene ripetuta continuamente andiamo a “scrivere” sul led il valore che leggiamo dal potenziometro. Il primo parametro del metodo indica il pin su cui scrviviamo il valore e il seconda indica il valore da scrivere. Noi usiamo il metodo valoreLetto() per sapere il valore del potenziometro.

### Esempio 2 – Blink

Il secondo esempio che abbiamo realizzato fa lampeggiare il 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 andiamo inizialmente a far fare un pausa della durata del valore letto dal potenziometro (in millisecondi) moltiplicato per due. Il valore andrà quindi da 0 a 510 che saranno i millisecondi di pausa. Successivamente accendiamo il led tramite il metodo digitalWrite. Come primo parametro passiamo il led da accendere e come seconda parametro la parola chiave “HIGH” che indica appunto che il led è acceso. Adesso svolgiamo come in precedenza una pausa della durata in millisecondi letta dal potenziometro e poi moltiplicata per due. Infine sempre con il metodo digitalWrite andiamo a spegnere il led usando questa volta la parola chiave “LOW” come secondo parametro per indicare che il led viene spento.

### Esempio 3 – FadePro

Il terzo esempio che abbiamo realizzato è una versione alternativa del primo esempio in cui andiamo a far dissolvere la luminosità del led in base alla 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 creiamo una variabile “valore” in cui leggiamo il valore del potenziometro (da 0 a 255) ora tramite i costrutti if ed else andiamo a controllare se il valore del potenziometro è minore di 128 o maggiore. Nel primo caso scriviamo accendiamo il led con un intensità pari al valore del potenziometro moltiplicato per due. Nel secondo caso scriviamo nel led il valore massimo moltiplicato per due, meno il valore letto moltiplicato per 2 (così a meta “corsa” il led comincia a spegnersi).

## Servo Motor

La seconda libreria che abbiamo realizzato è per il servo motor, questa presenta quattro metodi pubblici. Il primo è “ServoMotor(int pin, int velocita)”; questo metodo costruttore riceve come parametro il pin a cui è collegato il potenziometro e si occupa di impostare il pin di lettura del valore del potenziometro in base ai pin analogici del Digispark. Il pin 2 viene impostato a 1, il 4 a 2, il 5 a 0 e il 3 rimane 3. Il secondo parametro che il metodo riceve è “velocita”, indica la velocità di movimento dell’elica del servo motor. Successivamente troviamo il metodo “setVelocita(int velocita)” che serve a modificare la velocità del motore in base al valore del parametro “velocita” e il metodo getPosizione() che ritorna la posizione mappata del motore. L’ultimo metodo, “startServo()”, serve ad avviare il motore del servo motor. Nella libreria sono presenti cinque variabili di tipo intere e private. Due di queste sono delle costanti, una è per la posizione iniziale e l’altra per la posizione finale. I nomi sono “POS\_INIZIALE” e “POS\_FINALE”. Le altre tre variabili sono: “\_pin” che è un intero che indica il pin a cui è collegato il potenziometro, “\_velocita” che indica la velocità di movimento dell’elica ed infine “\_posizione” che rappresenta la posizione del servo motor.

### Esempio 1 - ServoMotorStart

Il primo esempio realizzato per questa libreria fa muovere il servo motor avanti e indietro. Per cominciare andiamo ad includere la nostra libreria chiamata “ServoMotor.h”. In un secondo momento inizializziamo un oggetto associato alla classe ServoMotor, di nome servoMotor e successivamente dichiariamo una variabile velocità, che conterrà la un valore intero (senza virgola), di default 20, che indica la velocità di movimento del motore. Nel 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 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. Questi passaggi vengono ripetuti all’infinito, gestiti tramite i costrutti if ed else if e azionati con il metodo “setVelocita()”. Alla fine del codice avviamo il servo motor tramite il metodo startServo().

### Esempio 2 - ServoMotoreIncrement

Il secondo esempio realizzato per questa libreria fa muovere il servo motor incrementando sempre di più la velocità. Per cominciare andiamo ad includere la nostra libreria chiamata “ServoMotor.h”. In un secondo momento inizializziamo un oggetto associato alla classe ServoMotor, di nome servoMotor e successivamente dichiariamo una variabile i, che conterrà un valore intero (senza virgola), inizialmente di 1, che indica la velocità iniziale di movimento del motore. Nel 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 iniziale di movimento che varrà inizialmente 1. Nel loop controlliamo con un if se la posizione supera i 180, in quel caso andiamo ad incrementare la velocità di 1 e la rendiamo negativa in modo da far tornare indietro il servo motor, a quel punto il servo procede fino a che la posizione arriva a 0 e a quel punto la velocità viene incrementata ancora e ritorna positiva. Questi passaggi vengono ripetuti all’infinito e modificati con il metodo “setVelocita()”. Alla fine del codice avviamo il servo motor tramite il metodo startServo().

### Esempio 3 - ServoMotoreIncrement

Il terzo esempio realizzato per questa libreria fa muovere il servo motor se il bottone viene premuto fino alla posizione 180. Al rilascio del bottone il servo motor torna alla posizione 0. Per cominciare andiamo ad includere la nostra libreria chiamata “ServoMotor.h”. In un secondo momento inizializziamo un oggetto associato alla classe ServoMotor, di nome servoMotor e successivamente dichiariamo una variabile velocità, che conterrà la un valore intero (senza virgola), di default 100, che indica la velocità di movimento del motore. Successivamente dichiariamo una variabile “val” che verificherà che il bottone è stato premuto o no. Viene inizalizzata infine la variabile “bottone” che specifica il pin del bottone. Nel 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 100. Nel loop controlliamo con un if se il bottone è premuto e la posizione del servo motor è superiore o uguale a 0, in quel caso andiamo ad impostare la velocità al servo motor in negativo (così torna indietro) e andiamo ad avviare il servo motor. Con un else if controlliamo se il bottone non è premuto e se la posizione del servo motor è inferiore o uguale a 180. In questo caso invece rendiamo la velocità del servo motor positiva così da farlo andare avanti e successivamente avviamo il motore con il metodo “startServo()”. Questi passaggi vengono ripetuti all’infinito.

## Ultrasuoni

La terza libreria