

PowerShield 6+6 T800

Obiettivo: Utilizzare la scheda PowerShield 6+6 T800 per controllare dei carichi in corrente continua con Arduino. Caso applicativo: controllo di velocità di una ventola mediante PWM.

Componenti elettronici:

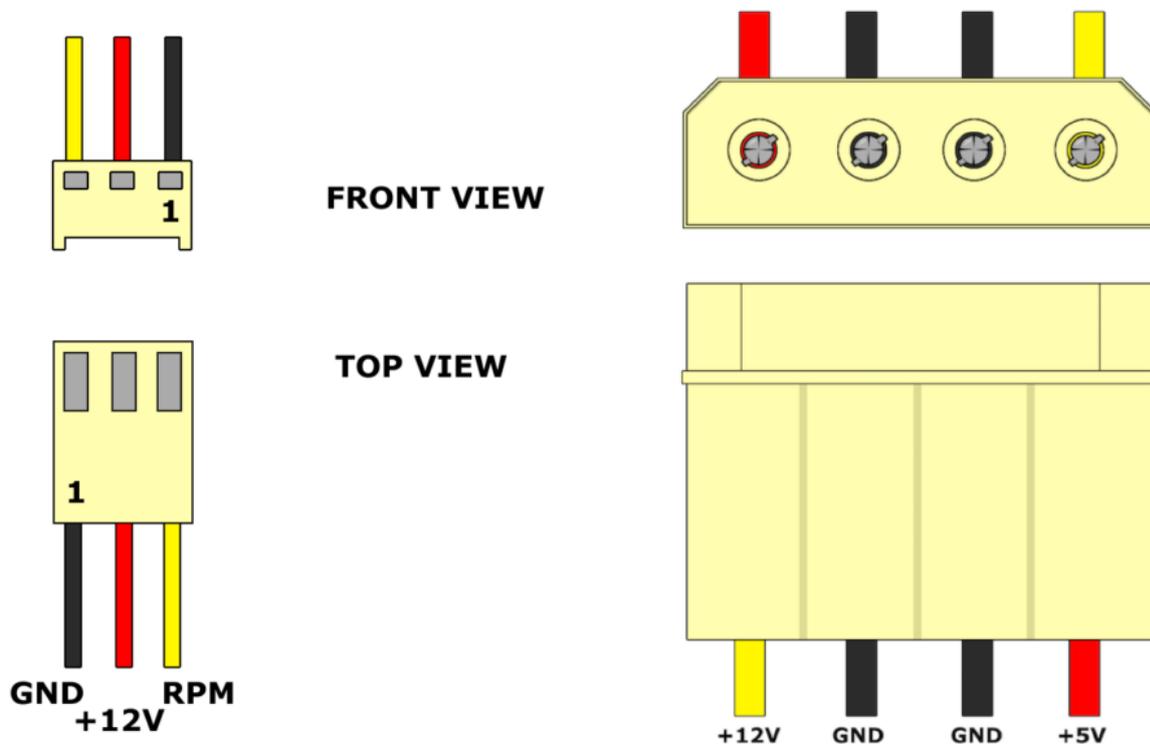
- Arduino UNO
- PowerShield 6+6 T800
- 1 Ventola da PC
- 1 Alimentatore da Banco (corrente continua)
- 1 Trimmer per controllare la velocità della ventola

Pre-requisiti:

[Controllo di un LED mediante un Potenziometro](#)

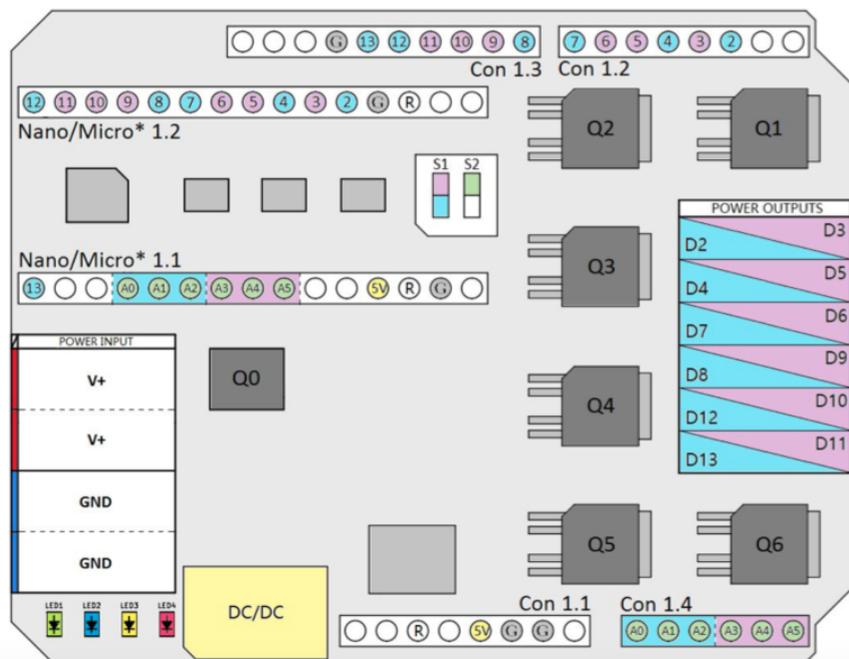
Teoria: Obiettivo di questa dimostrazione è controllare una ventola per PC utilizzando la PowerShield 6+6 T800. E' importante considerare che esistono due differenti tipologie di ventole le quali si differenziano nel numero di cavi utilizzati per alimentarle (vedi la seguente figura). Nel caso specifico è stata utilizzata una ventola a 3 cavi:

- Cavo Nero: GND
- Cavo Rosso: Alimentazione
- Cavo Giallo: RPM per il controllo della velocità



Poiché la tensione di alimentazione è pari a 12 volt non è possibile utilizzare esclusivamente la scheda Arduino per pilotare la ventola ma è necessaria una fonte di alimentazione esterna ed un dispositivo di controllo (e.g., Relè, Transistor). Nel caso specifico è stato utilizzato la PowerShield 6+6 T800.

PowerShield 6+6 T800: Sviluppata da Vytautas Janušonis e Valdas Mikėnas, questa scheda è stata progettata ed ideata per Arduino (UNO, MEGA, NANO) con l'obiettivo di aiutare l'utilizzatore nella gestione dei carichi che richiedono particolari valori di tensione (input voltage range 6.5 – 32Volt) e corrente (fino a 25 Ampere). Grazie alla tecnologia Mosfet la scheda supporta la gestione di elevate frequenze in



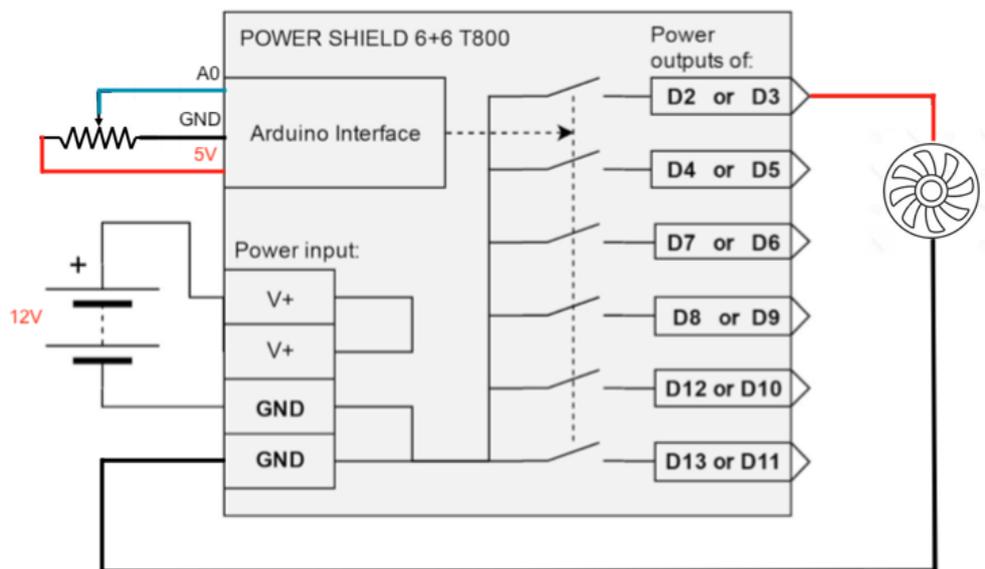
PowerShield 6+6 T800 Pin Configuration

Nella board sono presenti due switch:

- **Switch1:** Permette di scegliere le uscite. Quando lo switch1 è in posizione OFF i morsetti sono pilotati i pin 2, 4, 7, 8, 12e 13. Quando invece lo switch2 è in posizione ON, sono portati sui morsetti di uscita i PIN PWM 3, 5, 6, 9, 10, 11.
- **Switch2:** Permette di ottenere il feedback sugli ingressi analogici di quello che sta accadendo in uscita. Nello specifico, quando lo switch2 è in posizione ON e lo switch1 è in posizione OFF sul pin A2 viene riporta la misura della corrente che sta passando nel morsetto di uscita, sul pin A1 è riportato un warning (ON/OFF), ed infine sul pin A0 si indica se l'uscita è in protezione. Differentemente quando lo switch1 è in posizione ON queste informazioni sono riportate rispettivamente sui pin A3, A4, e A5. Tutti questi feedback possono essere visualizzati utilizzando il monitor Seriale.

Nel caso specifico del controllo di velocità di una ventola da PC lo Switch1 è impostato su ON (per abilitare i PIN PWM) mentre lo Switch2 è impostato su OFF.

Collegamento Circuitale:



Collegamento Circuitale

Considerazioni: La **PowerShield 6+6 T800** è una ottima scheda che permette di ampliare le possibilità di Arduino trasformandolo in un dispositivo efficace anche dal punto di vista della gestione di carichi che richiedono l'utilizzo di correnti elevate. In molte applicazioni infatti il limite di utilizzo del controllore Arduino è proprio legato all'impossibilità di gestire carichi che richiedono tensioni maggiori di 5volt e/o correnti superiori ad un Ampere.

Riavviare Arduino in modo Hardware

Obiettivo: Utilizzare la porta Reset per riavviare Arduino.

Teoria: In alcune applicazioni potrebbe essere utile avere la possibilità di riavviare l'esecuzione dello sketch via software. In questo modo si può ripristinare la board ad una condizione iniziale certa.

Un esempio potrebbe essere la gestione del meccanismo di sicurezza di una macchina industriale dove, premendo il pulsante di arresto, il dispositivo deve ritornare ad uno stato iniziale ben definito.

In questo specifico caso, è possibile **utilizzare** la porta Reset di arduino, che se opportunamente attivata, permette di riavviare il programma.

Nel dettaglio è importante specificare che la porta di reset è **attiva bassa** ovvero se portato a livello basso (0V) esegue il reset della scheda.

Per effettuare il reset della scheda viene introdotto un pulsante esterno collegato in serie con una resistenza di pull-up. In questa configurazione il pulsante permette di gestire in modo corretto (attivo basso) la porta di reset.

Collegamento Circuitale:

Riavviare Arduino in modo Software

Obiettivo: Utilizzare una “funzione di reset” per riavviare Arduino via codice.

Teoria:

In alcune applicazioni potrebbe essere utile avere la possibilità di riavviare l'esecuzione dello sketch via software. In questo modo si può ripristinare la board ad una condizione iniziale certa.

Un esempio potrebbe essere la gestione del meccanismo di sicurezza di una macchina industriale dove, premendo il pulsante di arresto, il dispositivo deve ritornare ad uno stato iniziale ben definito.

In questo specifico caso, è possibile **definire** ed **utilizzare** una funzione di Reset, che se richiamata, permette di riavviare il programma.

Nel dettaglio prima della funzione di setup viene dichiarato un **puntatore a funzione** il quale punta alla posizione zero. Richiamando questa funzione Arduino esegue il codice come se fosse stato appena avviato.

Codice: Viene in seguito riportato il codice necessario per riavviare via software il microcontrollore Arduino. Nello specifico il programma, una volta avviato, attende per cinque secondi prima di effettuare il reset software. Il monitor seriale viene utilizzato per fornire i feedback relativi alle fasi di start e di reset.

Realizzazione di un Gioco a Quiz – La macchina a stati finiti

Obiettivo: Realizzare un gioco a Quiz mediante pulsanti e display LCD 16×2 (basato su un Driver Hitachi HD44780). L'attività prevede la realizzazione software di una macchina a stati.

Componenti elettronici:

- Arduino UNO
- Breadboard
- 1 Display LCD (e.g., 1602A) compatibile con Driver Hitachi HD44780
- 1 Resistenza da 220 Ohm (per il display)
- 1 Trimmer da 10 kOhm (per il display)
- 2 Pulsanti
- 2 Resistenze da 1kOhm (per i pulsanti)

Pre-requisiti:

[Il Display LCD](#)

[LED e Pulsante](#)

Teoria: L'utilizzo di una progettazione basata sul concetto di macchina a stati finiti permette la facile realizzazione di apparati elettronici e sistemi capaci di fornire output desiderati a partire da specifici input ricevuti. Questo paradigma si basa sulla definizione degli stati di funzionamento del dispositivo da realizzare e sul codice associato allo stato stesso. A seguire sono riportati alcuni esempi pratici di stati di sistemi generici:

- Semaforo: verde, giallo, rosso
- Ascensore: primo piano, secondo piano, etc
- Cannello: aperto, chiuso

Da un punto di vista pratico si potrebbe affermare che: **“Attraverso una macchina a stati è possibile avere tanti loop quanti sono gli stati del dispositivo in questione”**. Tuttavia è importante considerare che solamente uno di questi loop è attivo. (La macchina non può trovarsi in più stati contemporaneamente).

Nel caso specifico di un gioco a quiz gli stati sono

rappresentati dalle:

- Domande: la domanda deve essere visualizzata sul display
- Risposte: l'utente deve premere il pulsante e il controllore deve verificare se la risposta è corretta.

Pertanto si avranno tanti stati quante sono il numero di domande moltiplicato per due.

E' infine importante considerare che, nel caso della realizzazione di un quiz a risposta multipla, l'impiego di una macchina a stati permette di risolvere il problema annoso del delay. Questa funzione può infatti essere tranquillamente utilizzata negli stati associati alle domande (l'utente deve avere tempo di leggere la domanda) mentre non deve essere assolutamente impiegata negli stati relativi alle risposte.

La realizzazione di una macchina a stati mediante il controllore Arduino è relativamente semplice. Infatti, il concetto di stato viene implementato grazie all'utilizzo di una variabile globale (tipicamente denominata state) e attraverso una serie di if o mediante uno switch case si seleziona il codice da eseguire in quello specifico stato.

Collegamento Circuitale:

risposte.

Controllo del Contrasto di un Display LCD mediante PWM

Obiettivo: Controllare il contrasto di un Display LCD 16×2 (basato su un Driver Hitachi HD44780) mediante PWM. (Se non possiedi un Trimmer puoi utilizzare questa strategia basata su PWM e filtro passa-basso).

Componenti elettronici:

- Arduino UNO
- Breadboard
- 1 Display LCD (e.g., 1602A) compatibile con Driver Hitachi HD44780
- 1 Resistenza da 220 Ohm
- 1 Resistenza da 1k0hm (per filtro passa basso)
- 1 Condensatore elettrolitico da 22uF (per filtro passa basso)

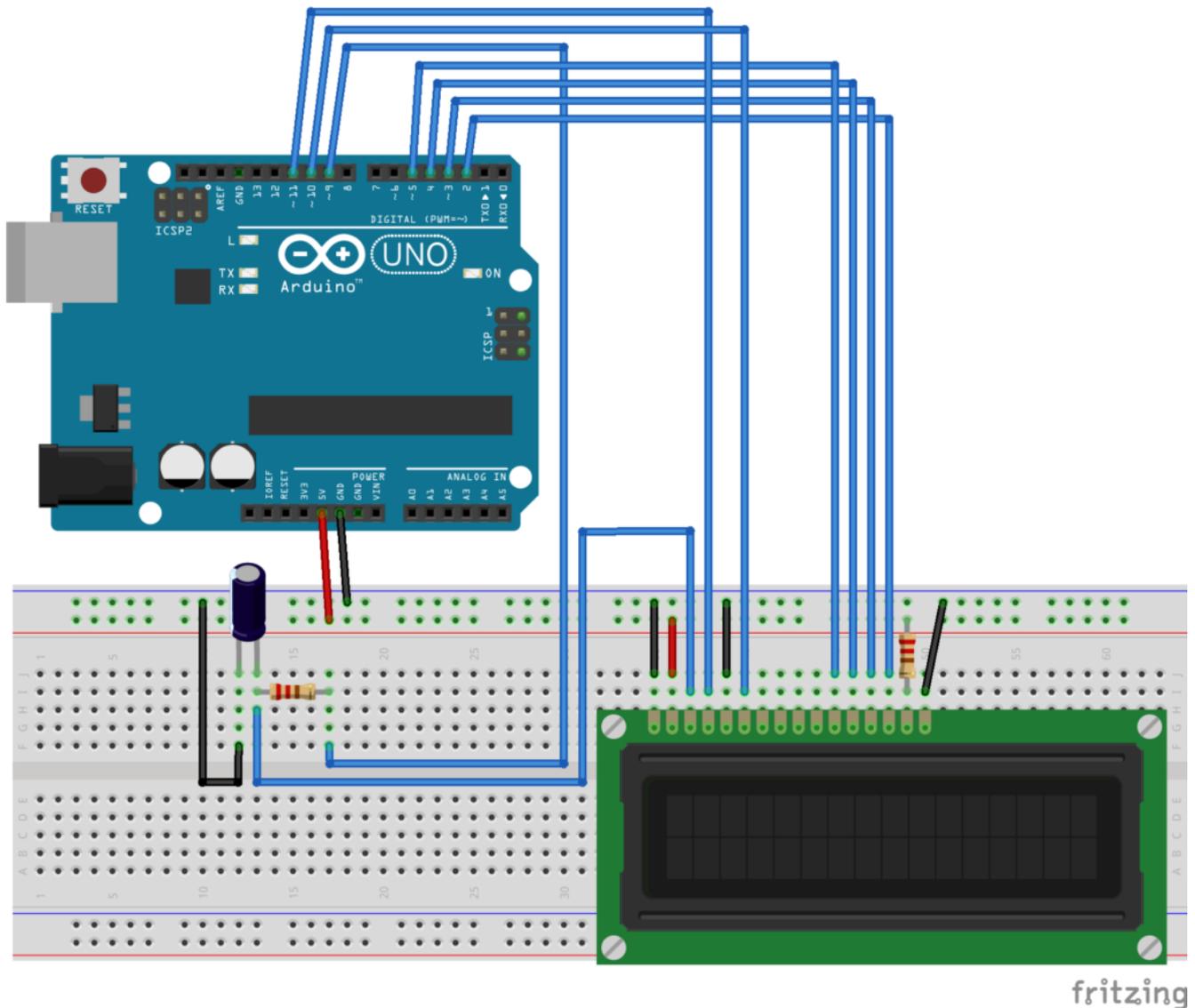
Pre-requisiti:

[Il Display LCD](#)

Teoria: In un display LCD basato su Driver HD44780 il pin numero 3 è utilizzato per gestire il contrasto. Questo pin viene tipicamente collegato ad un potenziometro con tensione variabile nel range 0 a 5v. Pertanto variando la posizione del Trimmer cambia il livello di contrasto del display. Il contrasto è un parametro di fondamentale importanza nella gestione di un dispositivo elettronico che utilizza un display LCD. Un'errata regolazione del livello di contrasto può rendere un'immagine troppo o poco dettagliata con il rischio che le aree più chiare e/o quelle più scure possano scomparire rendendo il testo non leggibile. Pertanto, nel caso in cui non si possedesse un Trimmer, l'utilizzo di un display LCD potrebbe essere facilmente compromesso a causa della incapacità di settare un livello di contrasto corretto.

Tuttavia, è importante considerare che, esistono delle alternative all'utilizzo di un trimmer per generare una tensione variabile compresa tra 0 e 5 Volts. Nel dettaglio, in questo articolo viene presentata una tecnica basata sull'utilizzo della PWM e di un filtro passa basso. L'impiego della tecnica PWM permette di generare un segnale con un duty cycle regolabile. Questa tecnica viene utilizzata anche nell'istruzione analogWrite per creare dei segnali apparentemente analogici partendo da segnali digitali. Il segnale PWM viene in seguito filtrato utilizzando un filtro RC passa basso del primo ordine. Attraverso questa operazione è infatti possibile ottenere la componente continua del segnale PWM necessaria per regolare il contrasto.

Collegamento Circuitale:



Collegamento Circuitale

Codice: Attraverso la variabile *contrast* (regolabile nel range 0-255) è possibile modificare il valore del contrasto da software fino ad ottenere l'effetto desiderato.

Personalizzazioni: E' possibile introdurre due pulsanti per modificare il contrasto del display utilizzando il valore PWM (0-255). Un pulsante può incrementare il valore della variabile *contrast* mentre l'altro pulsante può decrementare il

suo valore.

Utilizzare e Creare una Libreria per il Sensore ad Ultrasuoni

Obiettivo: Utilizzare e creare una libreria (file header e cpp) per un Sensore a Ultrasuoni (HC-SR04) utilizzato per misurare la distanza.

Puoi scaricare i file di libreria cliccando nel seguente link:
<http://www.arduinofacile.it/wp-content/uploads/2020/10/UltrasonicSensor.zip>

I file scaricati devono essere inseriti all'interno della cartella di progetto insieme al file .ino

Componenti elettronici:

- Arduino UNO
- Breadboard
- 1 Sensore Ultrasuoni (e.g., HC-SR04)

Pre-requisiti:

Il Sensore a Ultrasuoni

Teoria: la realizzazione di funzioni di libreria permette di facilitare l'operazione di **riutilizzo del codice** rendendo più veloce e più rapido lo sviluppo. Nel caso specifico la funzione di libreria implementata sarà costituita da un file header (.h) e da un file sorgente (.cpp).

Un file header è un file di testo che contiene i prototipi dei metodi (funzioni) definite nel relativo file sorgente. Nel caso in questione il file header contiene anche la dichiarazione della classe "UltrasonicSensor" utilizzata per modellare il sensore ad ultrasuoni.

Tale classe sarà caratterizzata da 2 attributi:

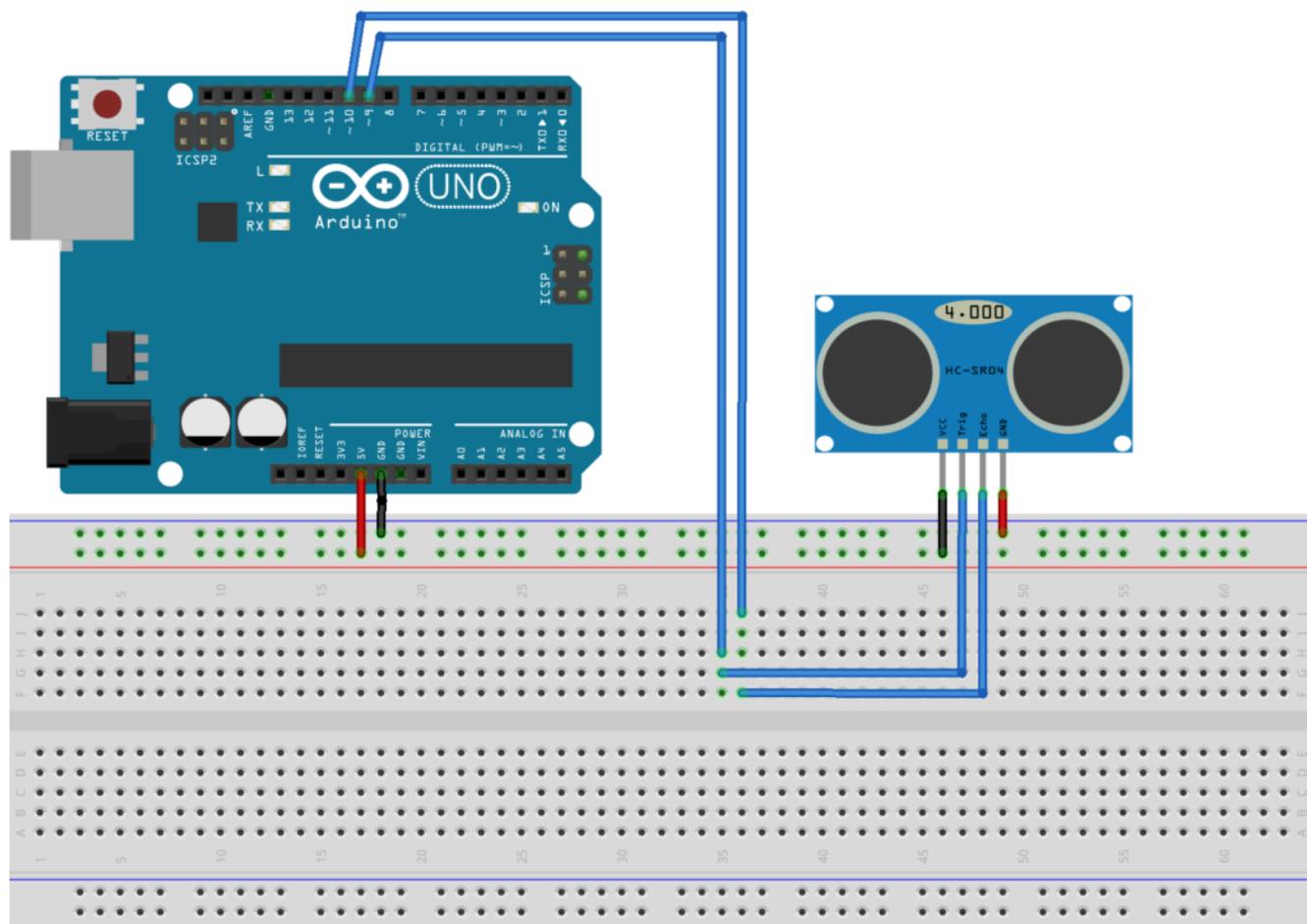
- **int pinEcho:** il pin di echo
- **int pinTrigger:** il pin di trigger

e da 4 metodi:

- **void SetPinEcho (int pinEcho):** metodo utilizzato per settare il pin di echo
- **void SetPinTrigger(int pinTrigger):** metodo utilizzato per settare il pin di trigger:
- **long GetDistance():** metodo utilizzato per effettuare la misura di distanza (restituisce un valore di tipo long)
- **long GetAverageDistance(int numIteration):** metodo utilizzato per effettuare la misura di distanza media su un numero dato di misure (restituisce un valore di tipo long)

Nel file sorgente viene invece riportata l'implementazione dei prototipi delle funzioni dichiarate nel file header.

Collegamento Circuitale:



fritzing

Collegamento Circuitale

Codice:

Vengono in seguito riportate le tre porzioni di codice utilizzate per creare la funzione di libreria relativa al sensore ad ultrasuoni HC-SR04.

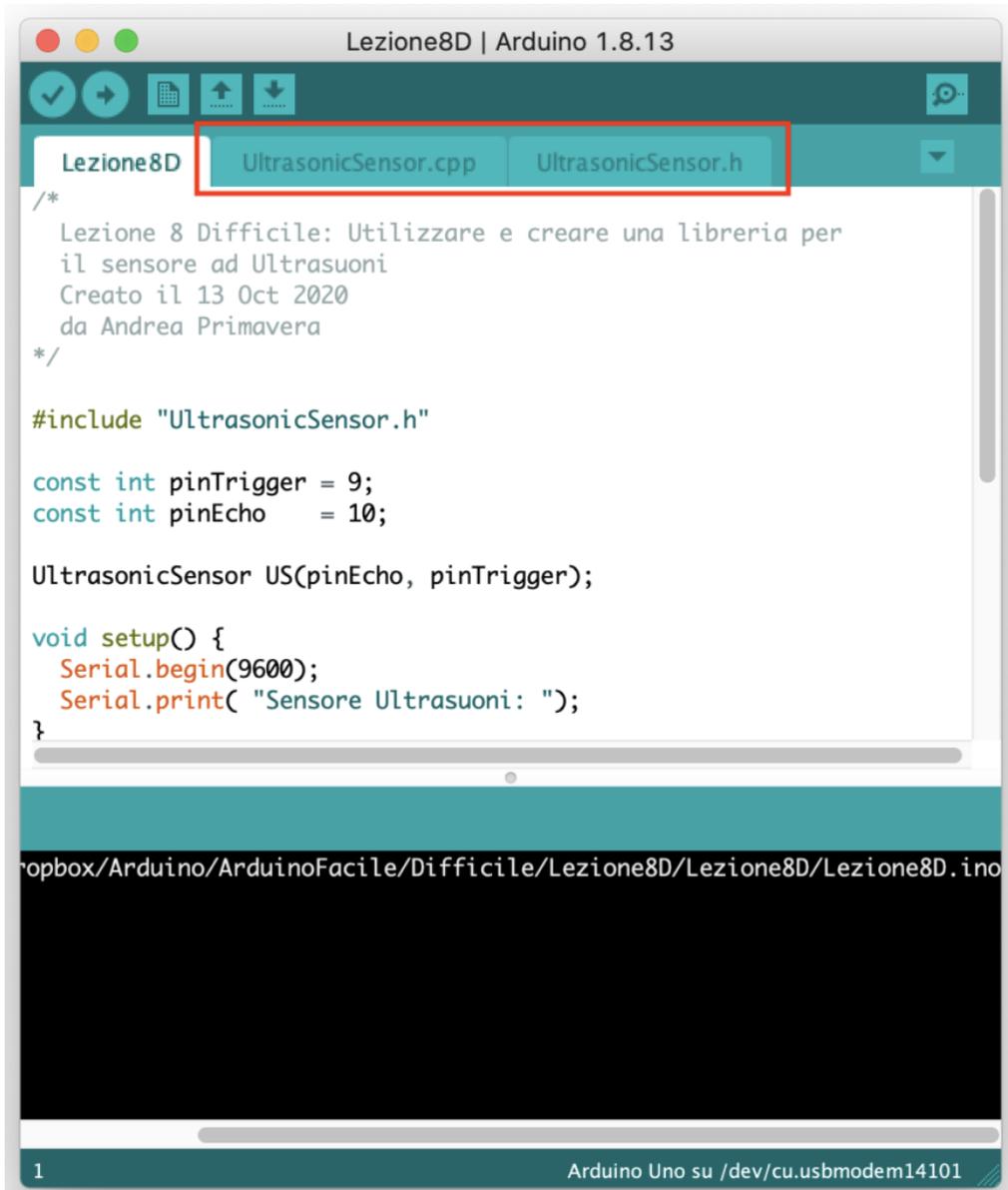
- File Header: contiene la definizione della classe con i

propri attributi (i.e., pinEcho e pinTrigger) ed i prototipi dei relativi metodi.

- File Sorgente: contiene le implementazioni dei metodi riportati nel file header.

- File Arduino: Utilizzato per fornire un esempio di come utilizzare la libreria per la gestione del sensore ad ultrasuoni.

Se tutti i file sono correttamente posizionati sullo stesso livello all'interno della cartella di progetto, due nuove tab compariranno nell'ambiente di sviluppo utilizzato per programmare Arduino. Attraverso queste tab sarà possibile visionare e modificare il file sorgente (.cpp) ed il file header (.h)



```
Lezione8D | Arduino 1.8.13
Lezione8D UltrasonicSensor.cpp UltrasonicSensor.h
/*
  Lezione 8 Difficile: Utilizzare e creare una libreria per
  il sensore ad Ultrasuoni
  Creato il 13 Oct 2020
  da Andrea Primavera
*/

#include "UltrasonicSensor.h"

const int pinTrigger = 9;
const int pinEcho    = 10;

UltrasonicSensor US(pinEcho, pinTrigger);

void setup() {
  Serial.begin(9600);
  Serial.print( "Sensore Ultrasuoni: ");
}

/home/robox/Arduino/ArduinoFacile/Difficile/Lezione8D/Lezione8D/Lezione8D.ino
1 Arduino Uno su /dev/cu.usbmodem14101
```

L'Impianto Elettrico dell'APE con Arduino

Obiettivo: Realizzare l'Impianto Elettrico di un APE Car utilizzando il microcontrollore Arduino

Componenti elettronici:

- Arduino UNO
- Breadboard
- 4 Resistenze da 1k0hm per i pulsanti
- 3 Resistenze da 100 Ohm per i LED.
- 1 Buzzer Passivo (per la realizzazione di una melodia)
- 3 LED (Gialli per frecce, Bianchi per luci di posizione)
- 4 Pulsanti

Pre-requisiti:

[Pulsante come Interruttore](#)

[Blinking Led Senza Delay: MILLIS\(\)](#)

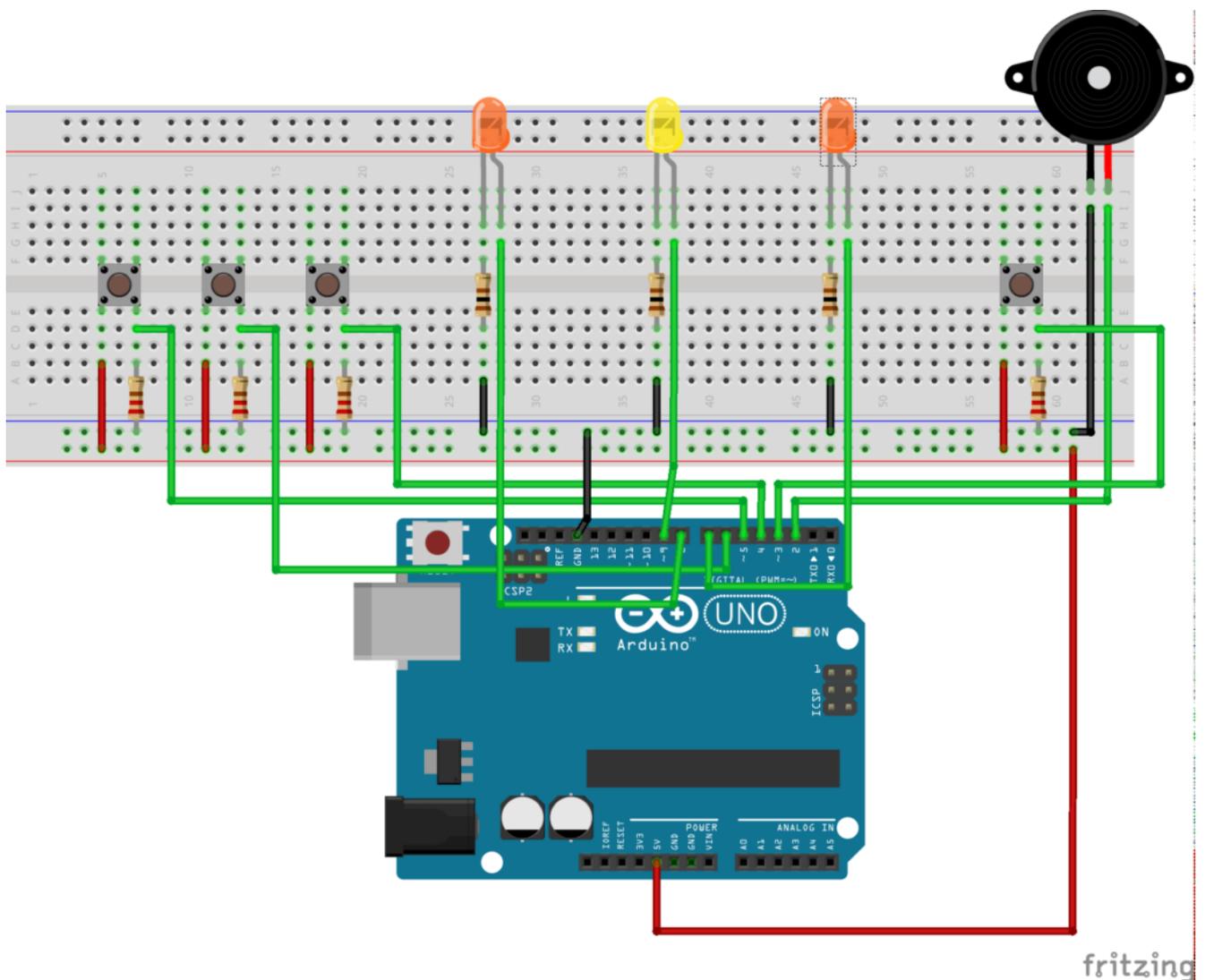
[Buzzer Passivo](#)

Teoria: Attraverso l'utilizzo di quattro semplici pulsanti utilizzati come interruttori (vedi pre-requisiti) l'utente può abilitare i vari attuatori tipici dell'impianto elettrico di un autoveicolo.(i.e., ape piaggio).

E' importante considerare che per fare lampeggiare le frecce è

stata utilizzata la funzione millis() a discapito della tradizionale delay. L'impiego della funzione millis() permette infatti una maggiore reazione dell'impianto elettrico. Le letture vengono fatte in tempo reale e pertanto appena si preme un pulsante l'attuttore associato reagisce immediatamente.

Collegamento Circuitale:



Codice:

Personalizzazioni: E' possibile modificare l'impianto elettrico inserendo melodie personalizzate per il clacson oppure introducendo i fari posteriori.

Blinking Led Senza Delay: MILLIS()

Obiettivo: Realizzazione del classico blinking led senza utilizzare la funzione Delay

Componenti elettronici:

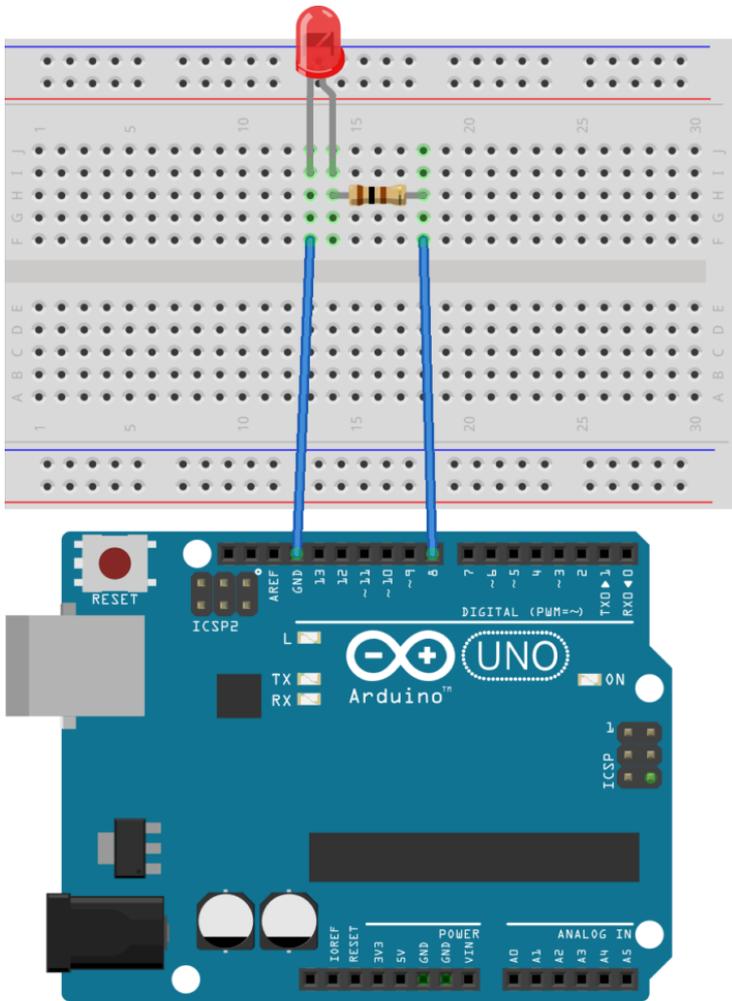
- Arduino UNO
- Breadboard
- Led
- Resistenza (100 Ohm)

Teoria: Se da un certo punto di vista l'impiego della funzione delay è particolarmente utile per la realizzazione di semplici applicativi; da un altro punto di vista molteplici potrebbero essere gli inconvenienti legati all'utilizzo di questa funzione.

E' infatti, molto importante, considerare che l'istruzione delay è un'istruzione bloccante. Questa istruzione "congela" Arduino nel suo stato corrente, pertanto negli istanti di delay non è possibile fare aggiornamenti, gestire input attraverso letture o comandare attuatori mediante scritture. Ad esempio, nel caso della realizzazione di un semaforo per perdoni, dove è possibile effettuare la chiamata mediante la pressione di un pulsante, l'impiego della funzione delay è altamente sconsigliato per la gestione del semaforo perchè questo renderebbe invisibile la pressione del pulsante. Per tutte queste ragioni è opportuno scrivere codici senza l'impiego della funzione delay. Nel dettaglio è opportuno sostituire la funzione delay con la più funzionale ma meno pratica funzione [millis](#).

La funzione millis restituisce il numero di millisecondi che sono passati da quando la board Arduino ha eseguito il programam corrente. Questo numero si riazzerà dopo circa 50 giorni.

Collegamento Circuitale:



fritzing

Collegamento Circuitale

Codice:

A seguire viene riportato il codice utilizzato per fare lampeggiare il led senza delay. Nello specifico il codice memorizza l'ultimo istante in cui è avvenuta una azione (il led ha cambiato stato) nella variabile `previousMillis` mentre nella variabile `currentMillis` viene memorizzato il tempo corrente. Attraverso la differenza tra questi due tempi si comprendere se il led deve cambiare stato (accendersi o spegnersi). Nel dettaglio, se la differenza tra la variabile `currentMillis` e `previousMillis` è maggiore di 1000 millisecondi allora lo stato del led cambia.

Personalizzazioni: E' possibile modificare il comportamento del circuito in questione intervenendo sul valore della variabile *interval*. Modificando il suo valore infatti cambia la frequenza di lampeggiamento del LED.

E' inoltre possibile modificare il pin digitale utilizzato per pilotare il LED cambiando rispettivamente hardware e software.

L'utilizzo di una resistenza, in serie al LED, serve appunto per limitare la quantità di corrente presente sul diodo emettitore di luce.

Controllo di un LED Mediante Telecomando SAMSUNG

Obiettivo: Controllo ON/OFF di un LED mediante il telecomando SAMSUNG ad infrarossi.

Componenti elettronici:

- Arduino
- Telecomando di un televisore SAMSUNG
- Ricevitore IR
- 1 Resistenza 100 Ohm
- 1 LED

Pre-requisiti:

Per utilizzare il sensore ad Infrarossi ed il telecomando è necessario installare la libreria IRremote. Per scoprire come installare la libreria IRremote consultare la seguente lezione:

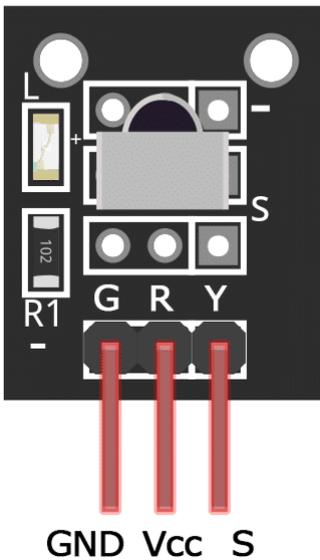
[Come Installare una Libreria \(IRremote Library\)](#)

Teoria: Il telecomando è un dispositivo elettronico, sviluppato negli anni 50, che permette di inviare (ma non di ricevere) segnali ad un altro dispositivo situato a distanza per comandarlo.

In genere, i telecomandi tradizionali sono in grado di trasmettere il segnale fino ad una distanza di circa 20 metri sotto forma di raggi infrarossi codificati.

Affinché il dispositivo da comandare possa ricevere i segnali inviati dal telecomando è necessario utilizzare un ricevitore ad infrarossi tipicamente fornito con il telecomando. La maggior parte dei ricevitori ad infrarossi in commercio sono dotati di 3 pin. Nel caso specifico, viene utilizzato il sensore KY 022, sul quale in prossimità dei connettori sono riportate tre lettere (G, R, Y)

- Alimentazione (R)
- Ground (G)
- Uscita (Y)



Ricevitore IR

Il pin di uscita (Y) del ricevitore permette di inviare al microcontrollore (al quale è collegato il ricevitore) il segnale ricevuto dal telecomando. E' importante considerare come ad ogni pulsante del telecomando sia associato un codice univoco. Tali codici variano in funzione dei vari telecomandi; pertanto prima di realizzare il programma finale è importante ottenere il valore del codice associato ai vari pulsanti.

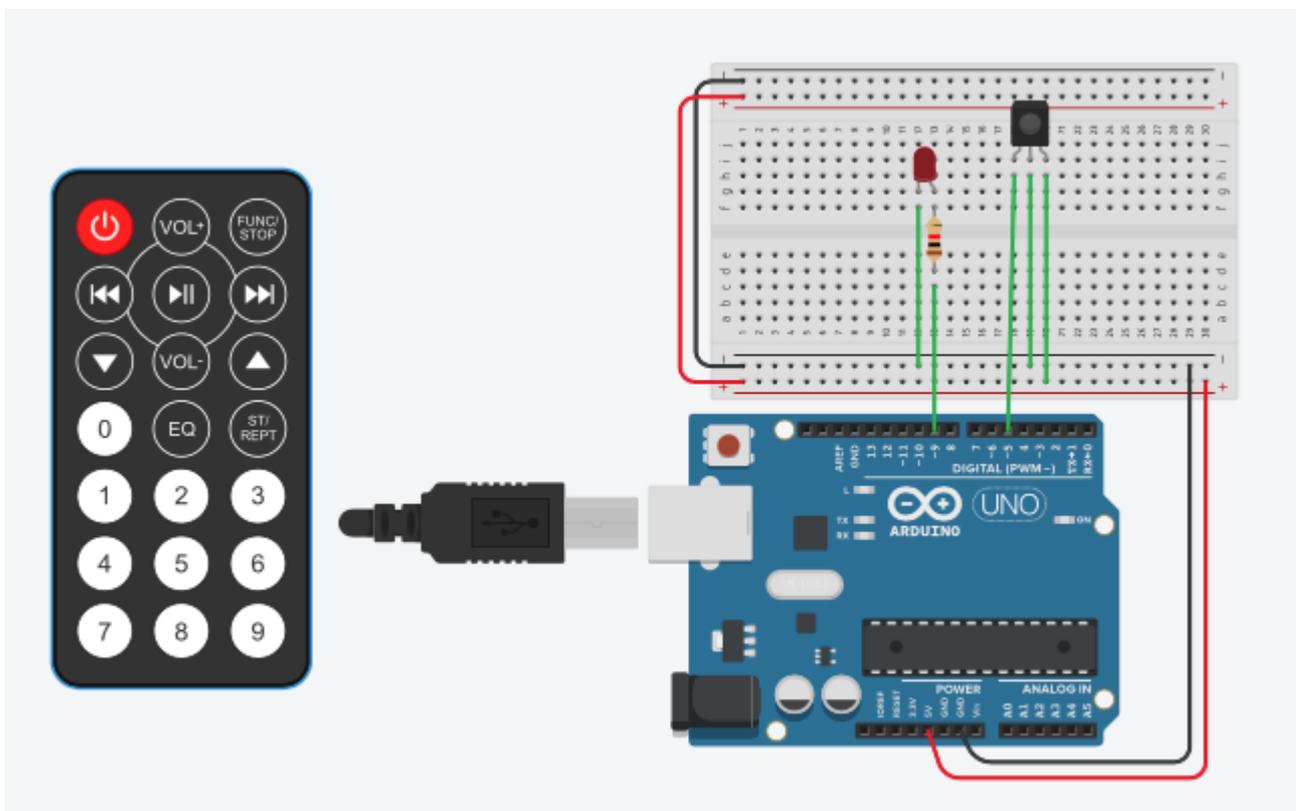
Nel caso specifico, si è deciso di utilizzare un telecomando SAMSUNG di una SMART TV. Utilizzando il software riportato in seguito è stato possibile determinare i codici associati ai vari pulsanti:

| Pulsante | Codice |
|---------------|------------|
| ON/OFF Button | 3772793023 |
| UP Button | 3772795063 |
| DOWN Button | 3772778743 |
| Number 0 | 3772811383 |
| Number 1 | 3772784863 |
| Number 2 | 3772817503 |
| Number 3 | 3772801183 |

| | |
|----------|------------|
| Number 4 | 4120482440 |
| Number 5 | 3772813423 |
| Number 6 | 3772797103 |
| Number 7 | 3772788943 |
| Number 8 | 3772821583 |
| Number 9 | 3772805263 |
| MENU | 3772799143 |

Se il telecomando a disposizione è differente da quello proposto in questa lezione è possibile ottenere i codici eseguendo il software riportato in seguito. Una volta ottenuti i codici, è possibile attraverso il seguente collegamento circuitale scrivere il programma per comandare un LED attraverso il pulsante di ON/OFF del telecomando.

Collegamento Circuitale:



Schema Circuitale

Codice:

A seguire viene riportato il software utile per determinare i codici associati ad ogni pulsante del telecomando a IR.

Ottenuto il codice associato al pulsante desiderato è possibile modificare il software per comandare l'azionamento di un LED mediante telecomando a IR. Il programma è molto simile al precedente viene solamente aggiunta la parte di codice relativa alla gestione del LED ed una istruzione IF per determinare se il pulsante premuto è quello di ON/OFF. E' importante infatti considerare che il LED si accenderà solamente quando il pulsante di ON/OFF è premuto.

Personalizzazioni:

E' possibile aggiungere più LED e comandare i vari LED con i vari pulsanti del telecomando.

Controllo di un LED Mediante Telecomando ELEG00

Obiettivo: Controllo ON/OFF di un LED mediante il telecomando ELEG00 ad infrarossi.

Componenti elettronici:

- Arduino
- Telecomando IR Elegoo
- Ricevitore IR
- 1 Resistenza 100 Ohm
- 1 LED

Pre-requisiti:

Per utilizzare il sensore ad Infrarossi ed il telecomando è necessario installare la libreria IRremote. Per scoprire come installare la libreria IRremote consultare la seguente lezione:

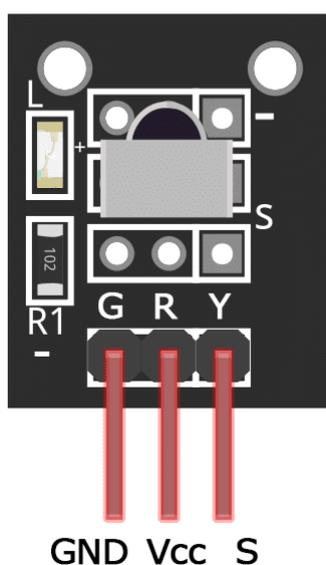
[Come Installare una Libreria \(IRremote Library\)](#)

Teoria: Il telecomando è un dispositivo elettronico, sviluppato negli anni 50, che permette di inviare (ma non di ricevere) segnali ad un altro dispositivo situato a distanza per comandarlo.

In genere, i telecomandi tradizionali sono in grado di trasmettere il segnale fino ad una distanza di circa 20 metri sotto forma di raggi infrarossi codificati.

Affinché il dispositivo da comandare possa ricevere i segnali inviati dal telecomando è necessario utilizzare un ricevitore ad infrarossi tipicamente fornito con il telecomando. La maggior parte dei ricevitori ad infrarossi in commercio sono dotati di 3 pin. Nel caso specifico, viene utilizzato il sensore KY 022, sul quale in prossimità dei connettori sono riportate tre lettere (G, R, Y)

- Alimentazione (R)
- Ground (G)
- Uscita (Y)



Ricevitore IR

Il pin di uscita (Y) del ricevitore permette di inviare al microcontrollore (al quale è collegato il ricevitore) il

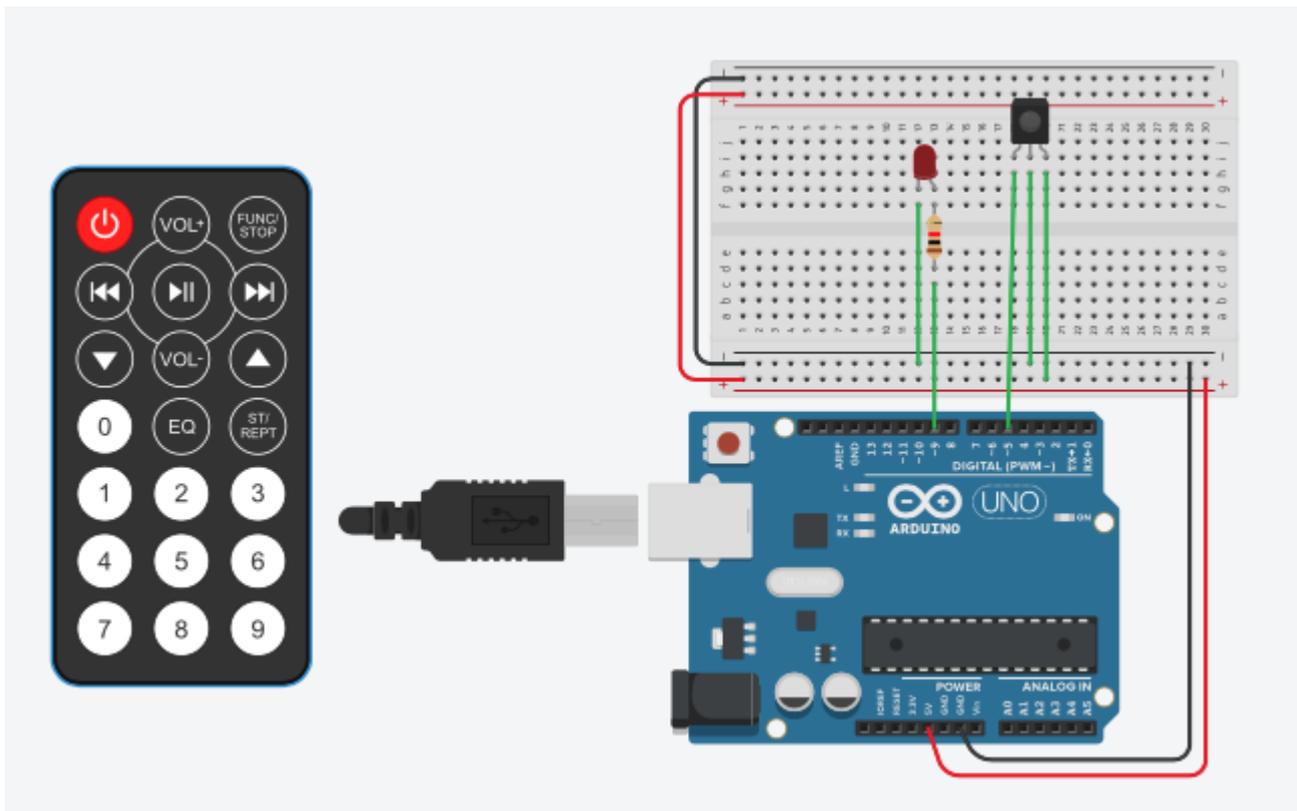
segnale ricevuto dal telecomando. E' importante considerare come ad ogni pulsante del telecomando sia associato un codice univoco. Tali codici variano in funzione dei vari telecomandi; pertanto prima di realizzare il programma finale è importante ottenere il valore del codice associato ai vari pulsanti.

Nel caso specifico del telecomando ELEG00 i codici associati ai vari pulsanti (ottenuti con il programma Arduino riportato in seguito) sono presentati nella seguente tabella:

| Pulsante | Codice |
|---------------|----------|
| ON/OFF Button | 16753245 |
| UP Button | 16748655 |
| DOWN Button | 16769055 |
| Number 0 | 16738455 |
| Number 1 | 16724175 |
| Number 2 | 16718055 |
| Number 3 | 16743045 |
| Number 4 | 16716015 |
| Number 5 | 16726215 |
| Number 6 | 16734885 |
| Number 7 | 16728765 |
| Number 8 | 16730805 |
| Number 9 | 16732845 |
| STOP | 16769565 |

Se il telecomando a disposizione è differente da quello proposto in questa lezione è possibile ottenere i codici eseguendo il software riportato in seguito. Una volta ottenuti i codici, è possibile attraverso il seguente collegamento circuitale scrivere il programma per comandare un LED attraverso il pulsante di ON/OFF del telecomando.

Collegamento Circuitale:



Schema Circuitale

Codice:

A seguire viene riportato il software utile per determinare i codici associati ad ogni pulsante del telecomando a IR.

Ottenuto il codice associato al pulsante desiderato è possibile modificare il software per comandare l'azionamento di un LED mediante telecomando a IR. Il programma è molto simile al precedente viene solamente aggiunta la parte di codice relativa alla gestione del LED ed una istruzione IF per determinare se il pulsante premuto è quello di ON/OFF. E'

importante infatti considerare che il LED si accenderà solamente quando il pulsante di ON/OFF è premuto.

Personalizzazioni:

E' possibile aggiungere più LED e comandare i vari LED con i vari pulsanti del telecomando.