Antifurto con funzioni domotiche

Obiettivo: Realizzare un antifurto con funzioni domotiche, con annesso un secondo Arduino in collegamento seriale in grado di comandare un cancello scorrevole.
Progetto realizzato dallo studente **Sanchini Simone**, dell'Università Politecnica delle Marche

Componenti elettronici:

- 2 Arduino UNO
- 2 Breadboard
- 4 Resistenze da 220 Ohm per i led
- 4 LED
- 6 Interruttori
- 1 Resistenza (2.2kOhm) per Fotoresistenza
- 1 Fotoresistenza
- 2 Motori DC
- 1 Piezo

- 1 Sensore PIR
- 1 Sensore IR
- 1 Telecomando IR
- 1 KeyPad 4×4

Pre-requisiti:

1. Password di accesso con KeyPad 4×4 e Arduino – (Tinkercad)
2. Crepuscolare (Smart Lamp)
3. Controllo di un LED Mediante Telecomando ELEGOO
4. Blinking Led Senza Delay: MILLIS()
5. Collegamento Seriale Arduino (non trovato su Arduinofacile)
6. Il Sensore di Presenza

Teoria:

Come abbiamo detto, lo scopo dell'esercitazione è quello realizzare un antifurto con funzioni domotiche utilizzando poi un secondo Arduino in collegamento seriale in grado di comandare un cancello scorrevole.

ARDUINO 1:

Attraverso il Keypad sarà possibile inserire tre tipi di codici diversi:

1. 1234: permette di inserire l'allarme (Stato=1), al suo inserimento le finestre e la porta di casa devono essere chiuse, queste ultime vengono simulate da due interruttori. All'inserimento ci saranno 5 secondi di delay in modo di permettere all'utente di uscire e chiudere la porta, dopodiché le luci presenti (simulate dal led) vengono spente, le tapparelle abbassate e attraverso il collegamento seriale viene dato il comando al secondo Arduino di aprire il cancello. Lo stop delle tapparelle e del cancello avviene attraverso dei finecorsa (simulati da due interruttori).

2. 1235: permette di togliere l'allarme (Stato=0), al momento del disinnesco, verranno aperte le tapparelle se giorno, oppure accese le luci se notte, utilizzando come riferimento il sensore crepuscolare.

3. 1236: permette di inserire l'allarme notturno (Stato=2), con la differenza dal primo che in questo caso esso non tiene in considerazione dell'apertura delle finestre. Al momento dell'inserimento attraverso il collegamento seriale viene dato il comando al secondo Arduino di

chiudere il cancello.

Indicazioni LED:

Rosso: allarme inserito;

Rosso-Verde: allarme notturno inserito;

Rosso-Giallo-Verde (lampeggio): codice errato;

Giallo: finestre/porta aperta;

Cosa succede in caso di apertura di finestre o porta quando il nostro allarme è inserito?

Il sistema passerà allo stato di intrusione (Stato=3), settando cosi un timer realizzato dalla funzione millis() di tempo t, nella quale è possibile inserire il codice di sblocco; In caso il codice di sblocco non è inserito entro il tempo limite la “sirena” inizia a suonare finché l’antifurto non verrà sbloccato.

La stessa cosa succede in caso di intrusione dalle finestre quando l’allarme notturno è inserito.

ARDUINO 2:

Il secondo Arduino permette di comandare un cancello attraverso o i comandi mandati dal primo, oppure attraverso un telecomando IR, con un solo pulsante (a causa di un problema di lettura di tinkercad).

Premendo il pulsante lo stato cambierà lo stato del cancello in base a quello precedente:

Stato=0 -> Cancello fermo;

Stato=1 -> Cancello in chiusura;

Stato=2 -> Cancello fermo;

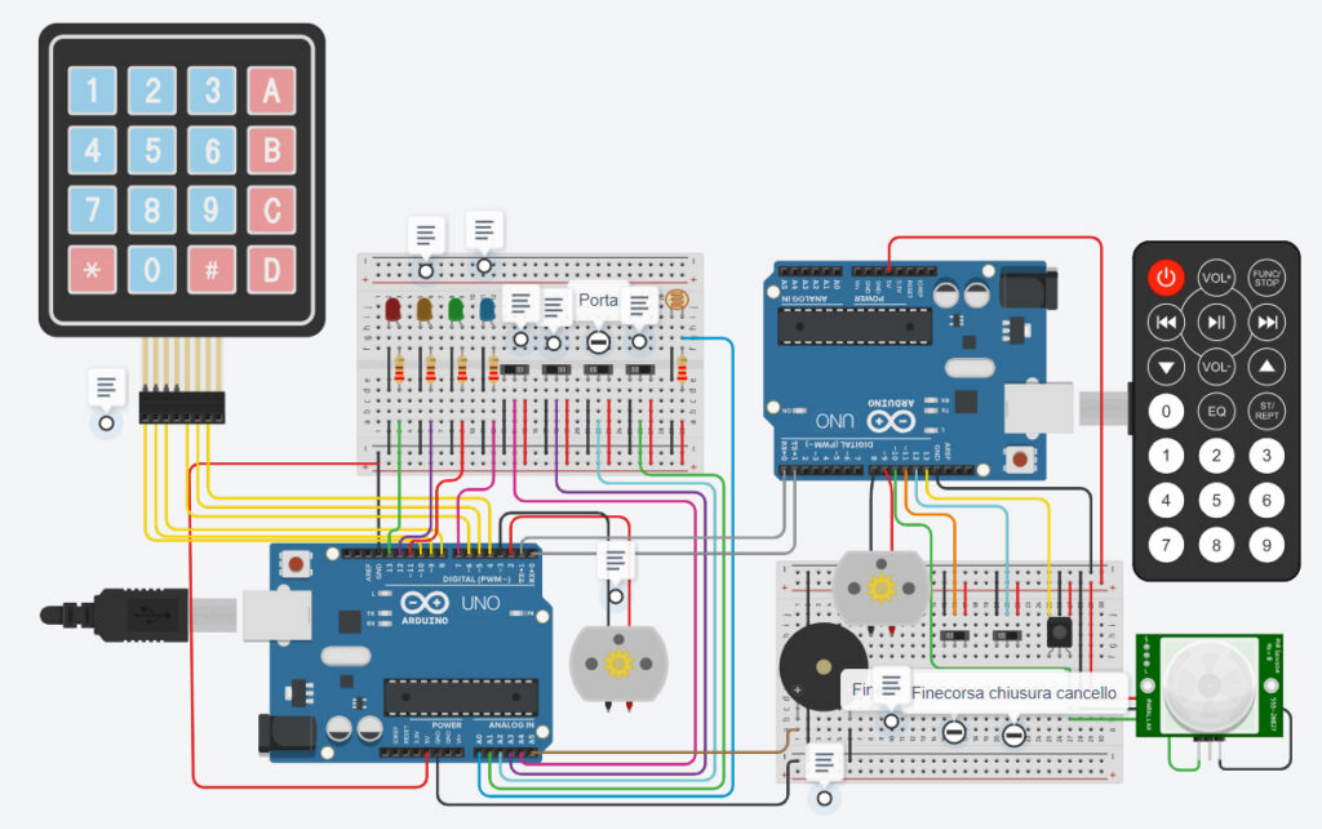
Stato=3 -> Cancello in apertura;

Il cancello è fermato in apertura e chiusura da due finecorsa, simulati da due interruttori.

Se in fase di chiusura il sensore PIR rileva qualcosa il cancello torna allo stato di apertura.

Quando il cancello finisce l’apertura, premendo il finecorsa, parte un timer di tempo t, ed alla fine di esso il cancello torna in fase di chiusura automatica.

Collegamento Circuitale:



TINKERCAD:

https://www.tinkercad.com/things/3aMc1kLCsu0-definitivo-sanchini/editel?sharecode=_gcKkLKcUpJEY1lHHIETPlS8ldhHe4IcM9bkXZUF48k

Codice:

Utilizzare e Creare una Libreria per il Display a 7 Segmenti

Obiettivo: Utilizzare e creare una libreria (file header e cpp) per un Display a Sette Segmenti.

Puoi scaricare i file di libreria cliccando nel seguente link: http://www.arduinofacile.it/wp-content/uploads/2021/03/SevenSegment.zip
I file scaricati devono essere inseriti all'interno della cartella di progetto insieme al file .ino

Componenti elettronici:

- Arduino UNO
- Breadboard
- 1 Display a 7 Segmenti
- 8 Resistenze

Pre-requisiti:

Creare funzioni con Arduino … per un Display a 7 Segmenti

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 costituta 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 "SevenSegment" utilizzata per modellare il display a sette segmenti.
Tale classe sarà caratterizzata da 10 attributi:

- **int pinA**: il pin A del display a sette segmenti
- **int pinB**: il pin B del display a sette segmenti
- **int pinC**: il pin C del display a sette segmenti
- **int pinD**: il pin D del display a sette segmenti
- **int pinE**: il pin E del display a sette segmenti
- **int pinF**: il pin F del display a sette segmenti
- **int pinG**: il pin G del display a sette segmenti
- **int pinDP**: il pin DP del display a sette segmenti
- **bool isCommonAnode**: indica se il display è di tipo anodo comune oppure no

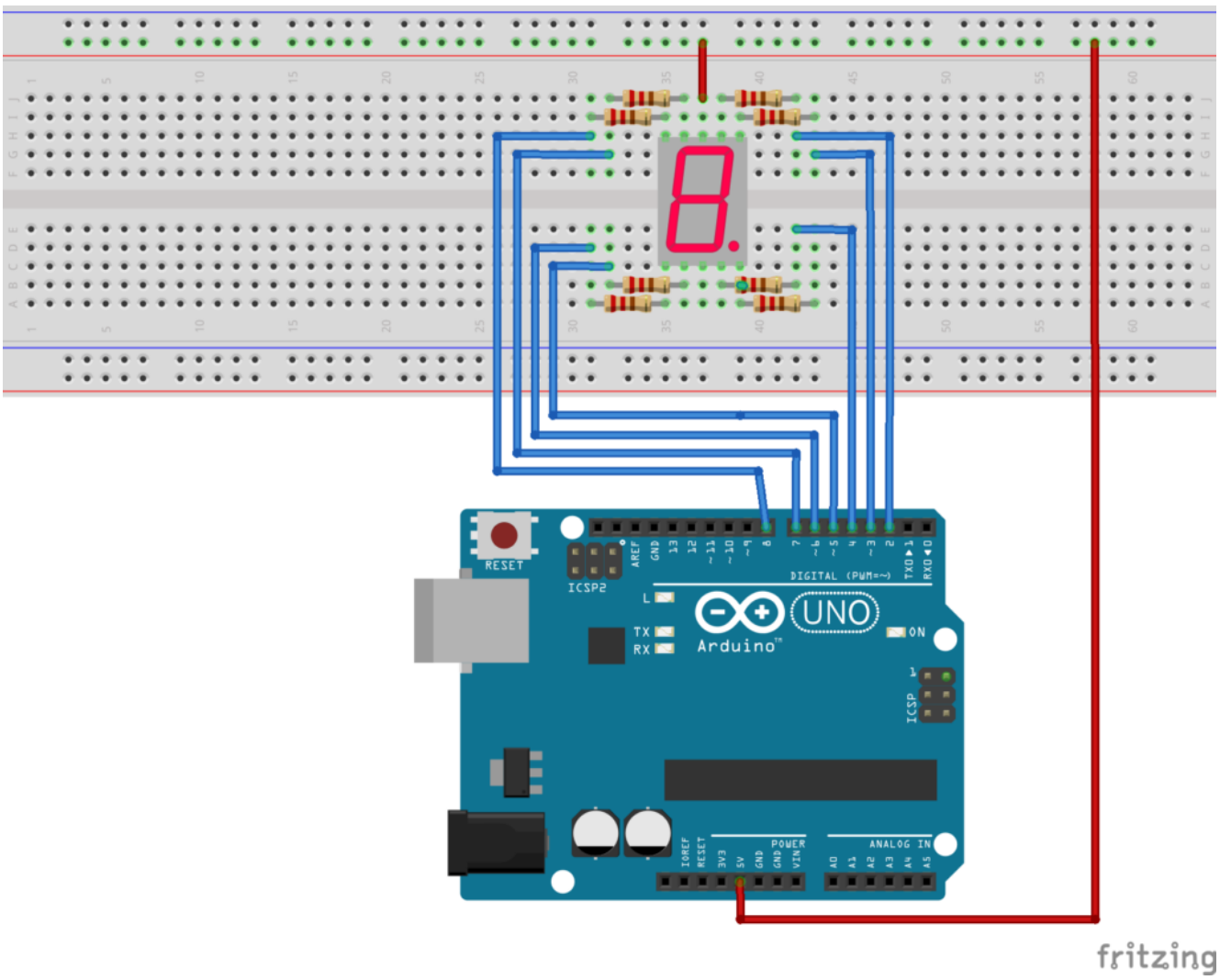
e da 11 metodi

- **void Print0()**: metodo utilizzato per stampare il numero 0
- **void Print1()**: metodo utilizzato per stampare il numero 1
- **void Print2()**: metodo utilizzato per stampare il numero 2
- **void Print3()**: metodo utilizzato per stampare il numero

3
- **void Print4()**: metodo utilizzato per stampare il numero 4
- **void Print5()**: metodo utilizzato per stampare il numero 5
- **void Print6()**: metodo utilizzato per stampare il numero 6
- **void Print7()**: metodo utilizzato per stampare il numero 7
- **void Print8()**: metodo utilizzato per stampare il numero 8
- **void Print9()**: metodo utilizzato per stampare il numero 9
- **void CountDown()**: metodo utilizzato per eseguire il countdown.

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

Collegamento Circuitale:



Collegamento Circuitale

Codice:

Vengono in seguito riportate le tre porzioni di codice utilizzate per creare la funzione di libreria relativa al display a sette segmenti.

- File Header: contiene la definizione della classe con i propri attributi (i.e., pinA, pinB, etc) 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 display a sette segmenti.

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)

IDE con l’utilizzo della libreria SevenSegment.h

Pilotare le uscite GPIO di

Raspberry tramite Server TCP/IP sviluppato in Java e client Android

Obiettivo: Accendere e spegnere tre LED tramite Raspberry via Server Java TCP/IP e Client Android.

Pilotare le uscite GPIO di Raspberry tramite Server TCP/IP sviluppato in Java e client Android

Componenti:

- Raspberry Pi 3 Model B+ oppure Raspberry Pi 4 Model B+
- N.1 LED Rosso
- N.1 LED Giallo
- N.1 LED Verde
- N.3 resistenze da 220 ohm

Teoria:

Alla base di questa esercitazione c'è Raspberry e la libreria Pi4J.

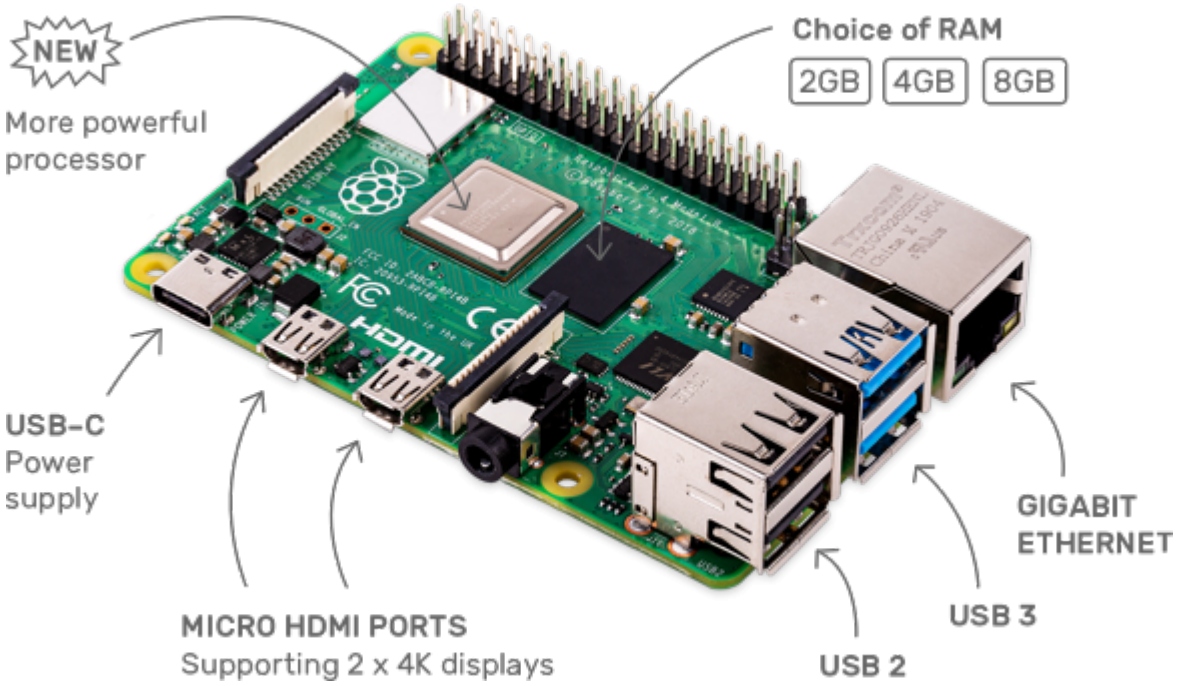
Raspberry Pi 3 Model B+ è la versione finale della famiglia Raspberry Pi 3 (https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/).

Queste le caratteristiche:

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- Extended 40-pin GPIO header
- Full-size HDMI
- 4 USB 2.0 ports
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- 4-pole stereo output and composite video port
- Micro SD port for loading your operating system and storing data
- 5V/2.5A DC power input
- Power-over-Ethernet (PoE) support (requires separate PoE HAT)

Raspberry Pi 4 Model B è l’ultima versione di Raspberry

(https://www.raspberrypi.org/products/raspberry-pi-4-model-b/).



Queste le caratteristiche:

- Processore
  1.5GHz quad-core 64-bit ARM Cortex-A72 CPU (about 3x performance)
- 1GB,
  2GB o 4GB di LPDDR4 SDRAM
- Gigabit
  Ethernet
- Dual-band
  802.11ac wireless rete
- Bluetooth
  5.0
- 2
  porte USB 3.0 e 2 porte USB 2.0
- Supporto
  dual monitor, con risoluzione fino a 4K

- VideoCore
  VI Graphics supporta OpenGL ES 3.x
- 4Kp60
  hardware decode di HEVC video
- Compatibilità
  con i precedenti prodotti Raspberry Pi

Ed infine la libreria to Pi4J (https://pi4j.com/1.2/index.html).

Questa
libreria ha lo scopo di fornire un'API di I/O orientata agli oggetti implementata
per i programmatori Java per accedere alle funzionalità di I/O complete della
piattaforma Raspberry Pi. Questo progetto astrae l'integrazione nativa di basso
livello e il monitoraggio degli interrupt per consentire ai programmatori Java
di concentrarsi sull'implementazione della logica di business dell'applicazione.

Come funziona il progetto:

Il progetto è suddiviso in 2 gruppi, il server e il client.

Il server viene realizzato tramite Raspberry sul quale viene

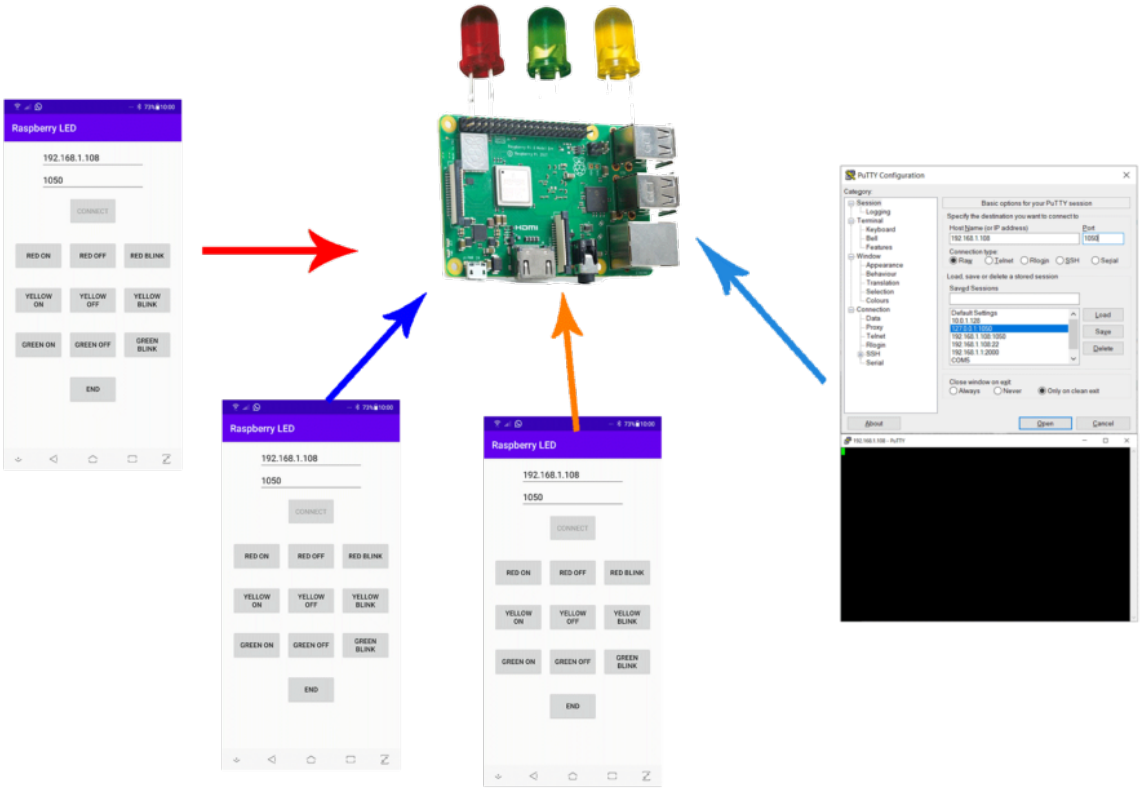
fatto girare il Server TCP/IP Multithreading.

Il server in base a semplici comandi tipo “ON RED”, “OFF RED”, “BLINK RED”, “ON YELLOW”, “OFF YELLOW”, “BLINK YELLOW”, “ON GREEN”, “OFF GREEN”,
“BLINK GREEN” tutti ovviamente senza virgolette, accende, spegne e fa
lampeggiare i vari LED collegati alle porte GPIO di Raspberry.

Il client invece è realizzato tramite Android Studio.

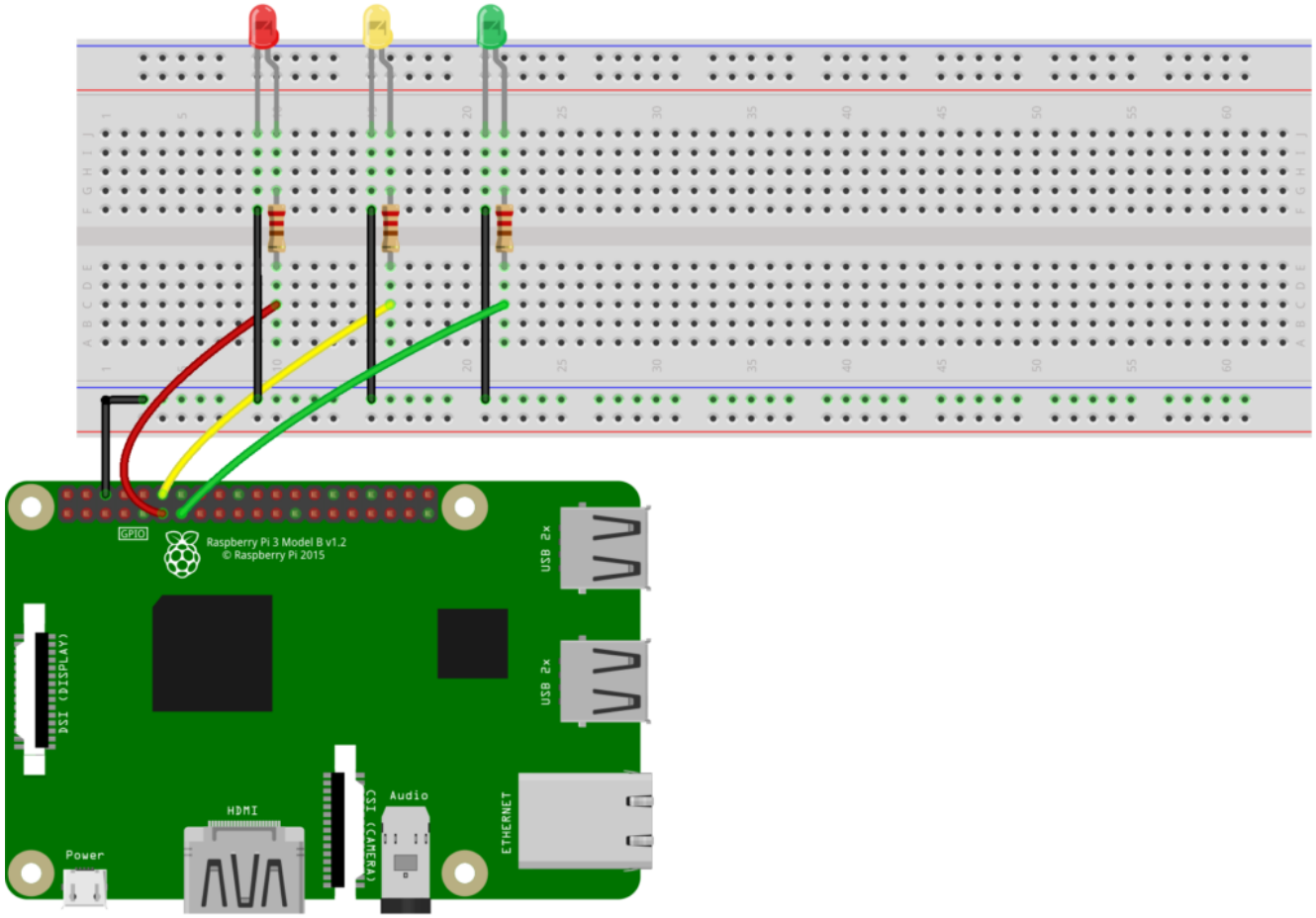
Il client ha una Activity contenente due Editbox per
digitare l’indirizzo IP e la porta di funzionamento del server e 11 bottoni in
grado di eseguire la connessione, accendere, spegnere, far lampeggiare i LED e
disconnettersi dal server.

È possibile anche utilizzare Putty da un qualsiasi PC, connettersi in modalità RAW all’indirizzo di Raspberry e alla porta 1050.

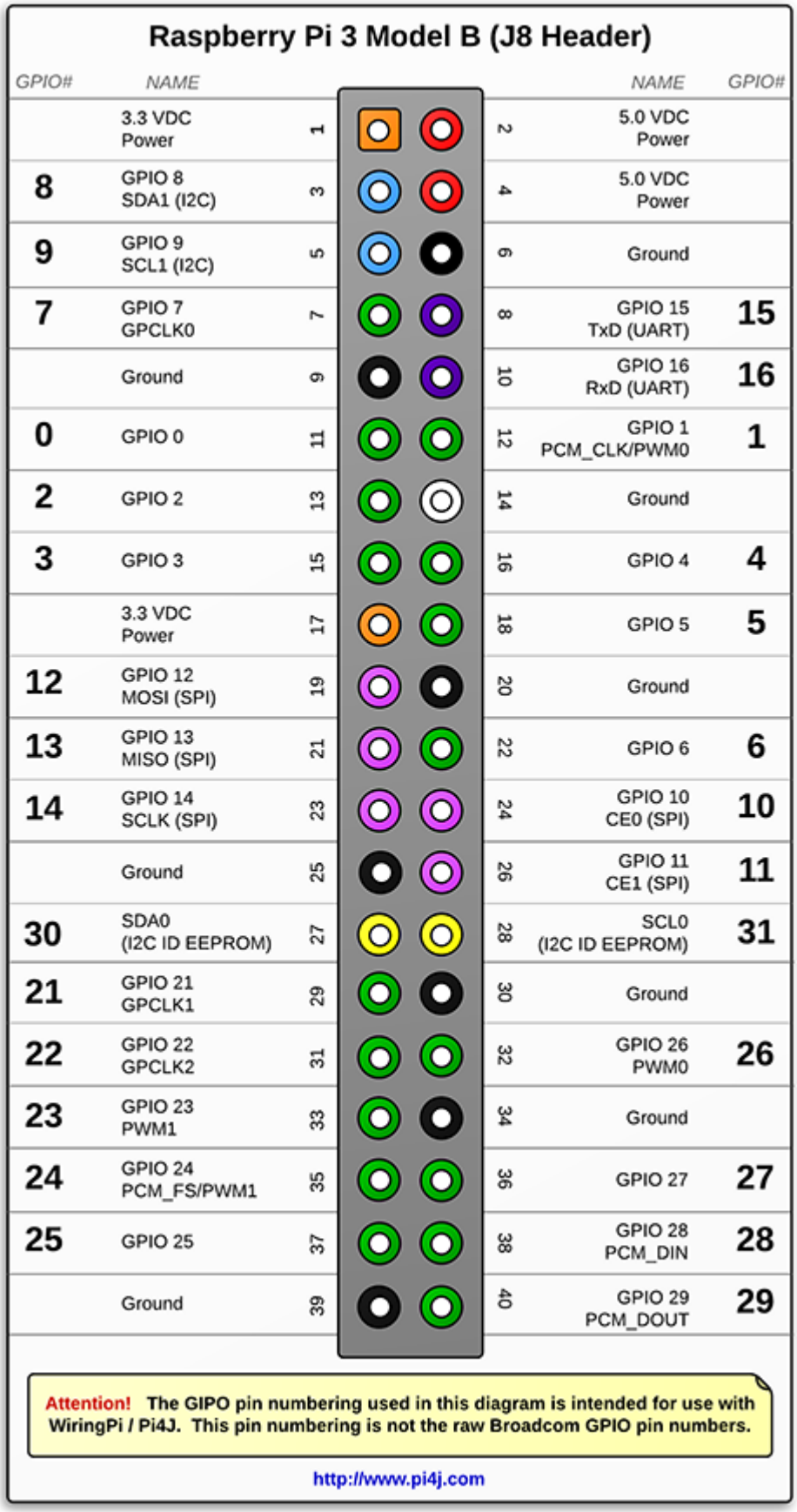


Schema di funzionamento

Schema Server:

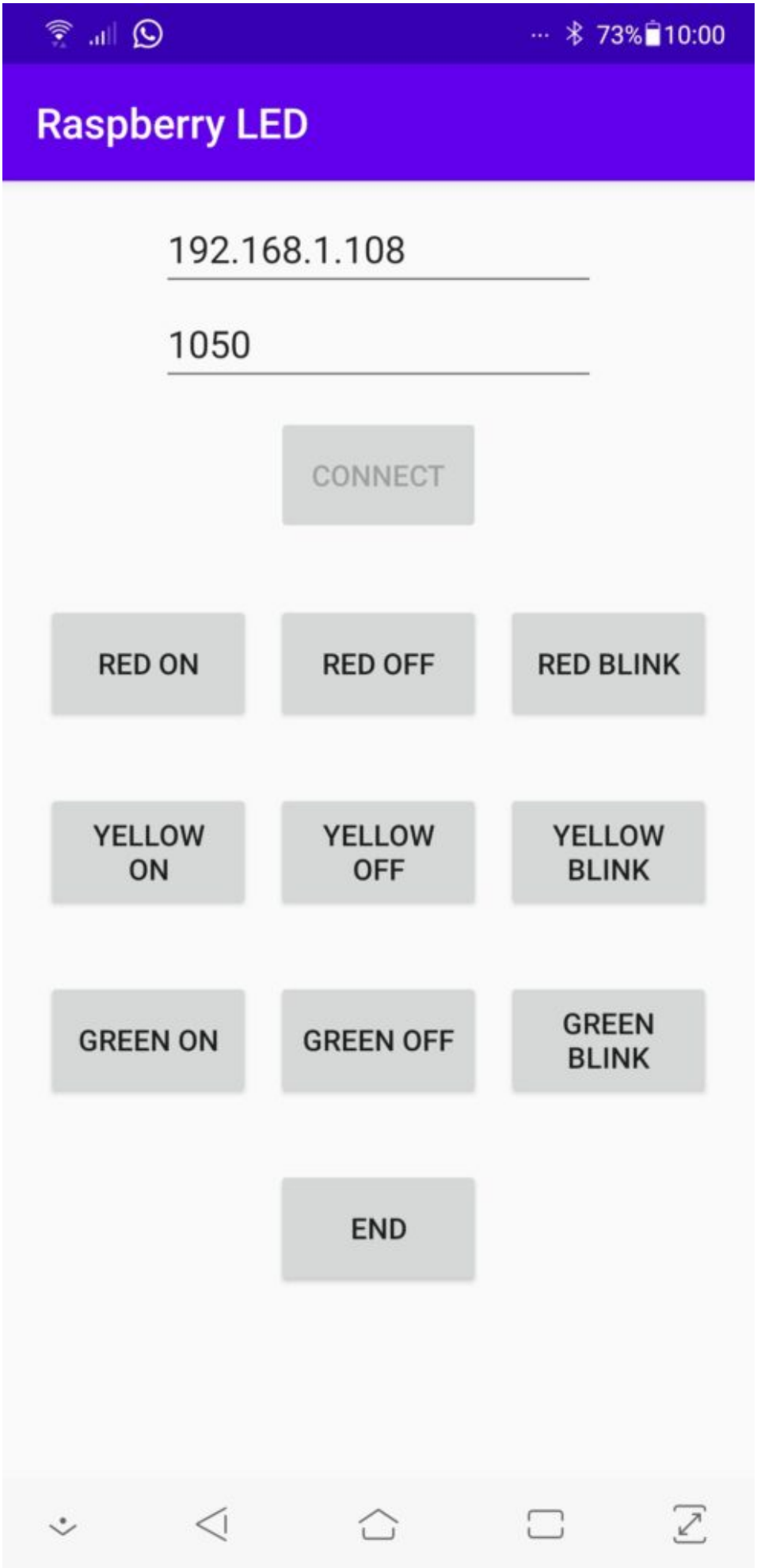


Schema di collegamento di Raspberry ai LED



Piedinatura delle GPIO di Raspberry

Schermata Client:


73% 10:00
Raspberry LED
192.168.1.108
1050
CONNECT
RED ON
RED OFF
RED BLINK
YELLOW ON
YELLOW OFF
YELLOW BLINK
GREEN ON
GREEN OFF
GREEN BLINK
END

Codice Sorgente:

Istruzioni installazione PI4J scrittura Server, compilazione ed esecuzione

Download Server TCP/IP Java

Download Client Android

Istruzioni per eseguire il server all'accensione di Raspberry come servizio

---

Creare un richiamo per Birdwatching tramite Arduino e lettore MP3 DFPlayer Mini

Obiettivo: Far suonare dei file MP3 di versi di uccelli tramite Arduino e lettore MP3 DFPlayer Mini visualizzando il numero della traccia su un display TM1637 e il nome il volume e lo stato su un display LCD 16×2 I^2C con possibilità di cambiare la traccia tramite telecomando IR

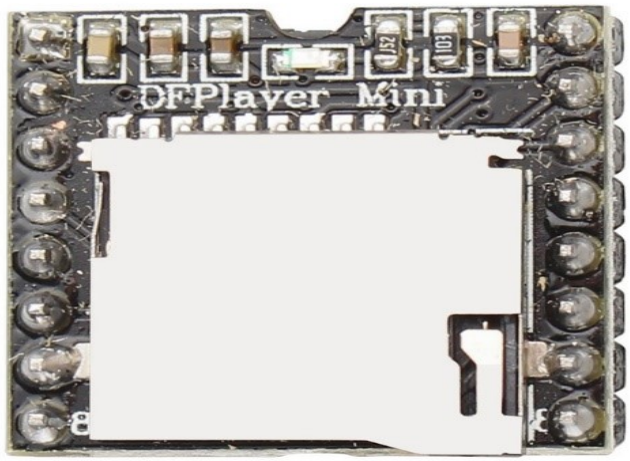
Creare un richiamo per Birdwatching tramite Arduino

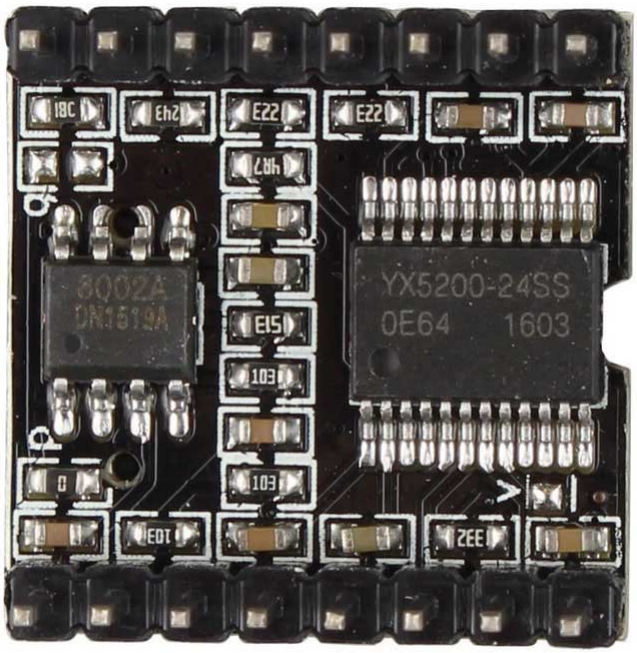
Componenti:

- Arduino UNO
- 1 DFPlayer Mini
- 1 TM1637
- 1 Display LCD 16×2 I^2C
- 1 Trasmettitore IR
- 1 Ricevitore IR

Teoria:

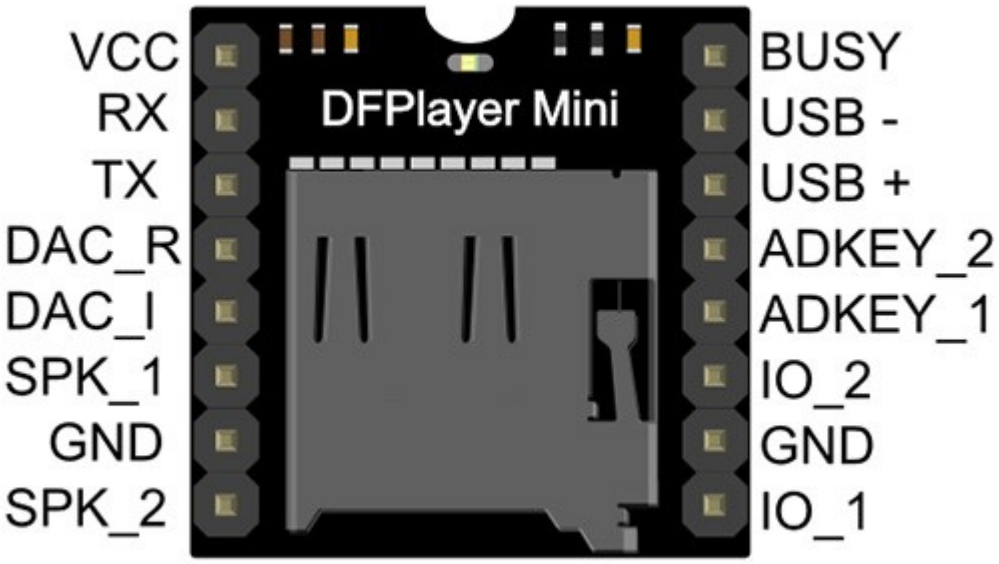
Alla
base di questa esercitazione c’è il modulo DFPlayer Mini





Sul [sito del produttore] è possibile studiare le principali informazioni che permettono di utilizzare in modo semplice il lettore.

Si possono notare il pinout a 16 pin di collegamento, necessario per interconnettere il DFPlayer mini a pulsanti per essere utilizzato senza microcontrollore, le alimentazioni, le uscite DAC per essere collegato ad un amplificatore esterno, e le uscite dirette ad un altoparlante.



Pin	Description	Note
VCC	Input Voltage	DC3.2~5.0V;Type: DC4.2V
RX	UART serial input	
TX	UART serial output	
DAC_R	Audio output right channel	Drive earphone and amplifier
DAC_L	Audio output left channel	Drive earphone and amplifier
SPK2	Speaker-	Drive speaker less than 3W
GND	Ground	Power GND
SPK1	Speaker+	Drive speaker less than 3W
IO1	Trigger port 1	Short press to play previous（long press to decrease volume）
GND	Ground	Power GND
IO2	Trigger port 2	Short press to play next（long press to increase volume）
ADKEY1	AD Port 1	Trigger play first segment
ADKEY2	AD Port 2	Trigger play fifth segment
USB+	USB+ DP	USB Port
USB-	USB- DM	USB Port
BUSY	Playing Status	Low means playing \High means no

Caratteristiche del modulo:

- Frequenze di campionamento supportate (kHz): 8 / 11.025 / 12/16 / 22.05 / 24/32 / 44.1 / 48
- Uscita DAC a 24 bit, supporto per gamma dinamica 90 dB, supporto SNR 85 dB
- Supporta pienamente FAT16, file system FAT32, supporto massimo 32G della scheda TF, supporto 32G di disco U, 64M byte NORFLASH
- Vasta varietà di modalità di controllo, modalità di controllo I/O, modalità seriale, modalità di controllo tramite pulsanti
- Funzione di attesa sonora pubblicitaria, la musica può essere sospesa. quando la pubblicità è finita nella musica continua
- Dati audio ordinati per cartella, supporta fino a 100 cartelle, ogni cartella può contenere fino a 255 canzoni
- 30 livelli di volume regolabile, 6 livelli EQ regolabili

Modalità di controllo

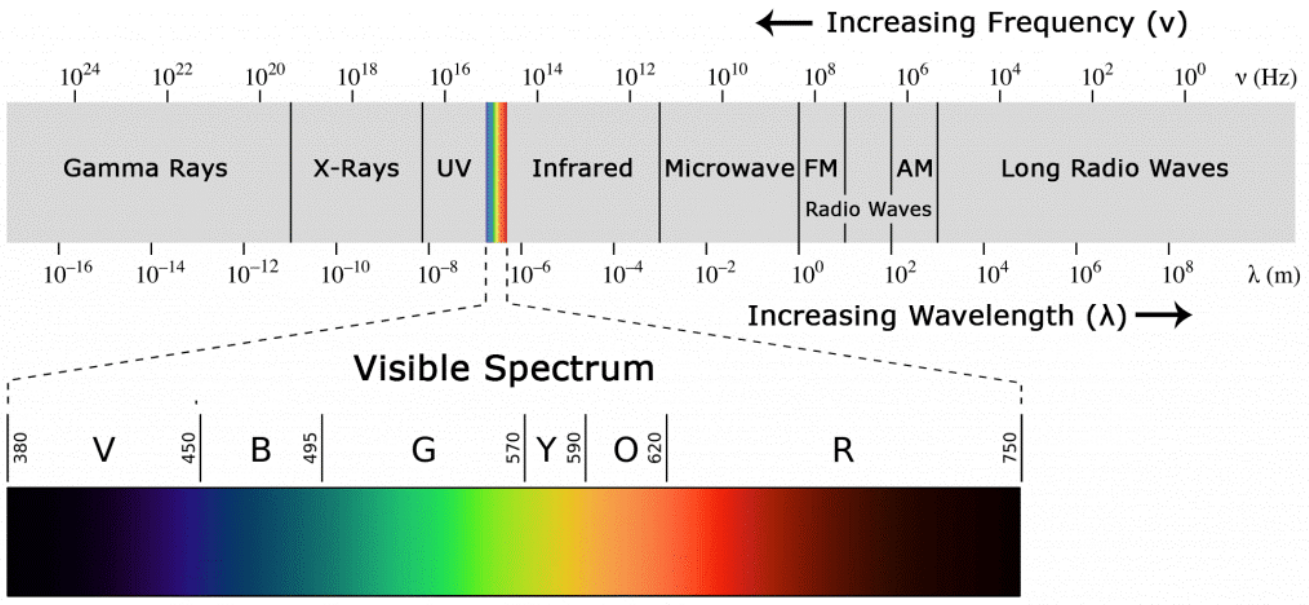
Sempre sul sito del produttore sono possibili vedere le tre modalità di funzionamento del lettore DFPlayer Mini che sono: Modalità Seriale, AD KEY Mode, I/O Mode.

Noi utilizzeremo la Modalità Seriale per interconnettere Arduino al modulo DFPlayer Mini e al tempo stesso utilizzare anche altri moduli quali Diplay LCD I^2C 16×2, Display TM1637, Ricevitore IR

Una nota aggiuntiva deve essere anche posta al ricevitore IR che ci permetterà di acquisire il codice dal trasmettitore IR in grado di far avviare il file MP3 desiderato, spostandoci tra i file MP3, avendo la possibilità di mettere in pausa e in play il suono, alzare e abbassare il volume.

Cosa sono gli Infrarossi

La
radiazione infrarossa è una forma di luce simile alla luce che vediamo tutto
intorno a noi. L'unica differenza tra la luce IR e la luce visibile è la
frequenza e la lunghezza d'onda. La radiazione infrarossa si trova al di fuori
della gamma della luce visibile, quindi gli esseri umani non possono vederla.

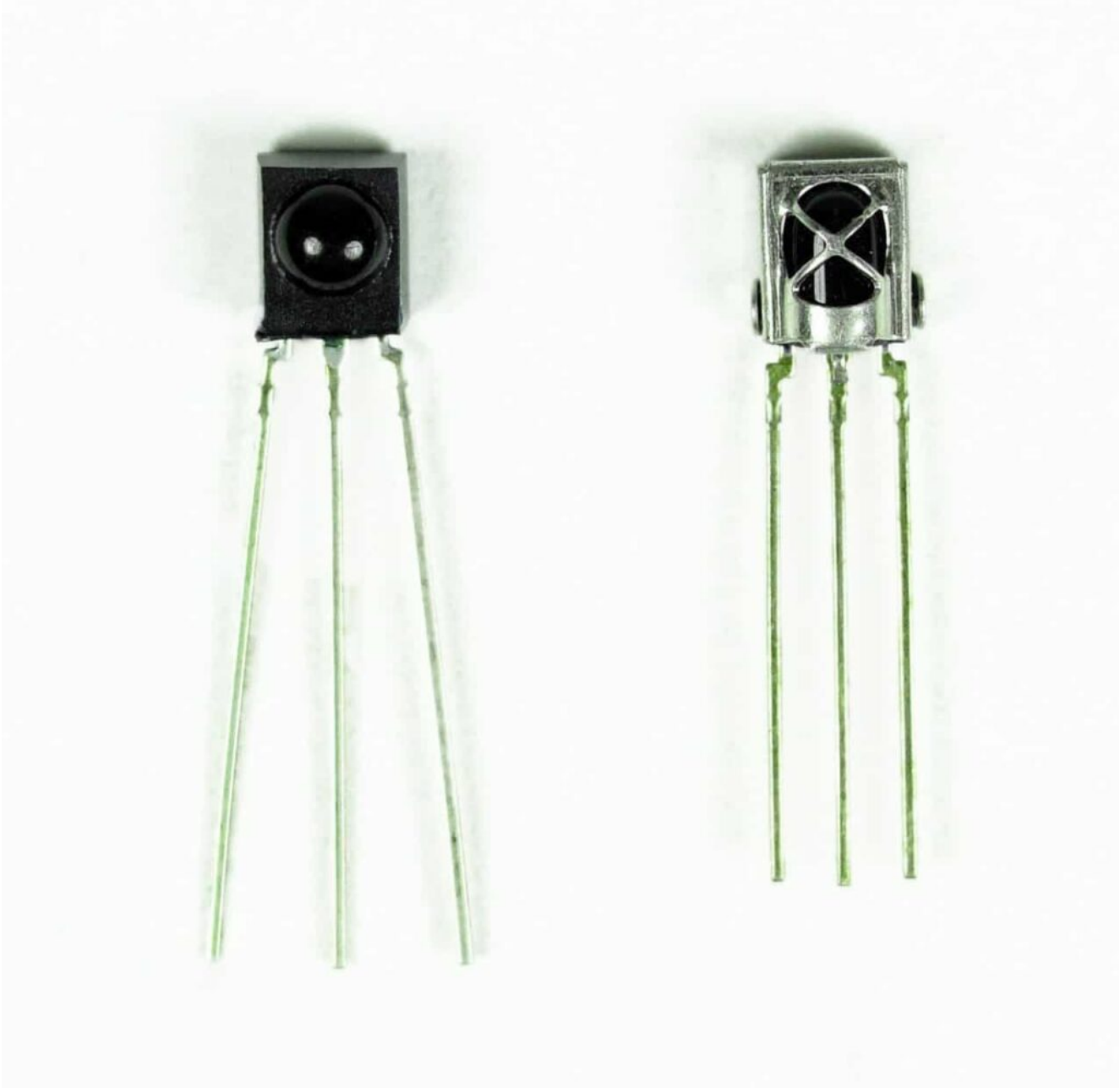


Poiché l'infrarosso è un tipo di
luce, la comunicazione IR richiede una linea visiva diretta

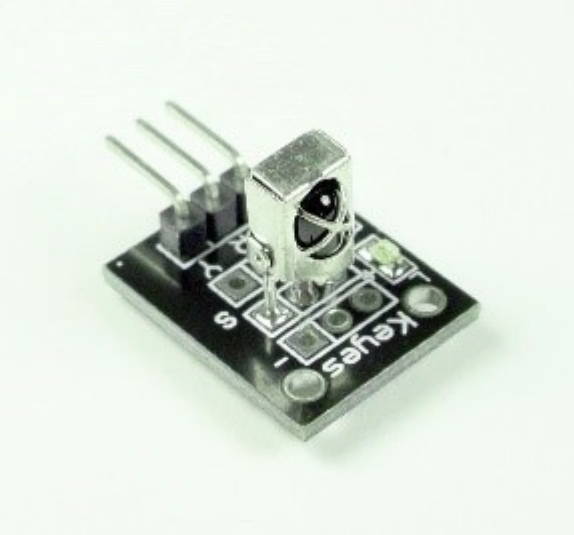
dal ricevitore al
trasmettitore quindi non è possibile trasmettere attraverso muri o altri
materiali come WiFi o Bluetooth.

Un tipico sistema di
comunicazione a infrarossi richiede un trasmettitore IR e un ricevitore IR. Il
trasmettitore ha l’aspetto di un LED standard, tranne per il fatto che produce
luce nello spettro IR invece che nello spettro visibile. Se si osserva la parte
anteriore del telecomando di un televisore, si vedrà il LED del trasmettitore
IR.

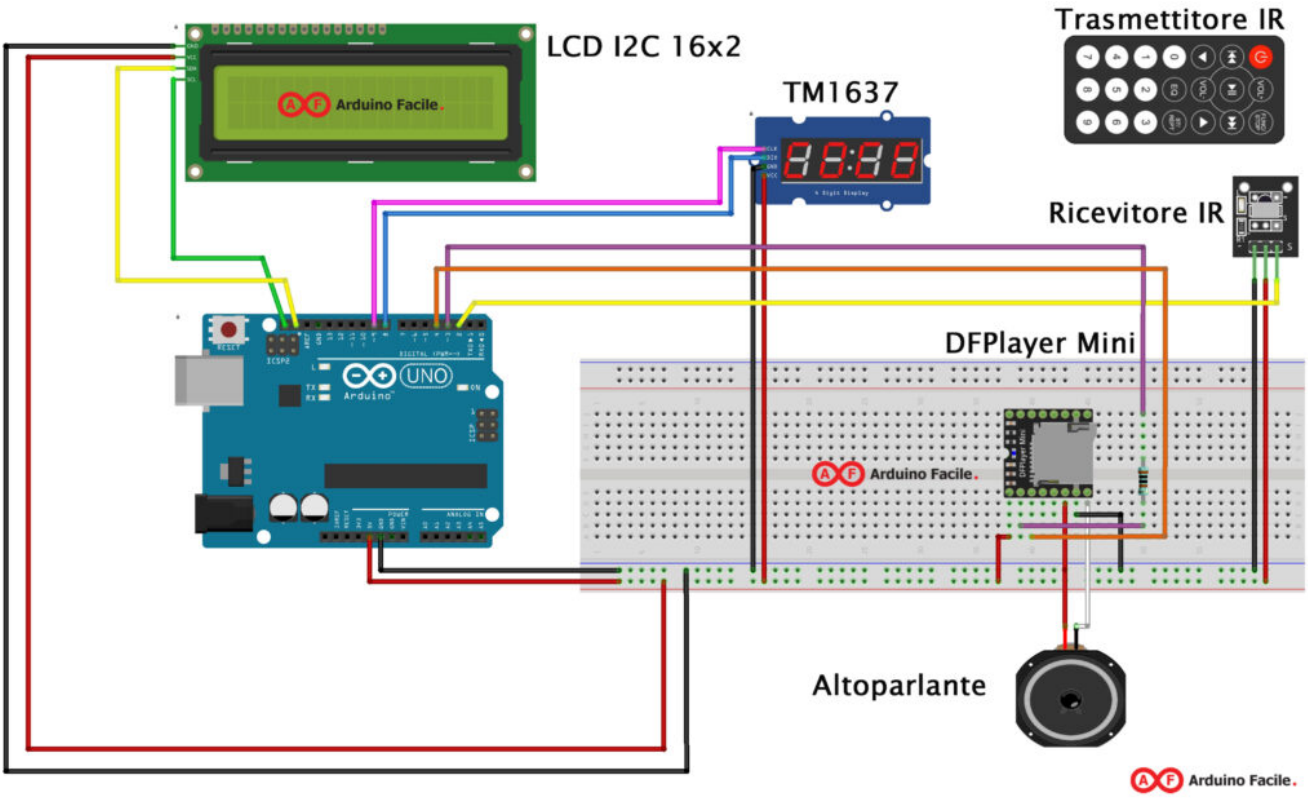
Il ricevitore IR è un fotodiodo e un preamplificatore che converte la luce IR in un segnale elettrico. I diodi del ricevitore IR in genere hanno questo aspetto:



Nel nostro caso è stato usato questo modulo preso nei soliti KIT per Arduino:



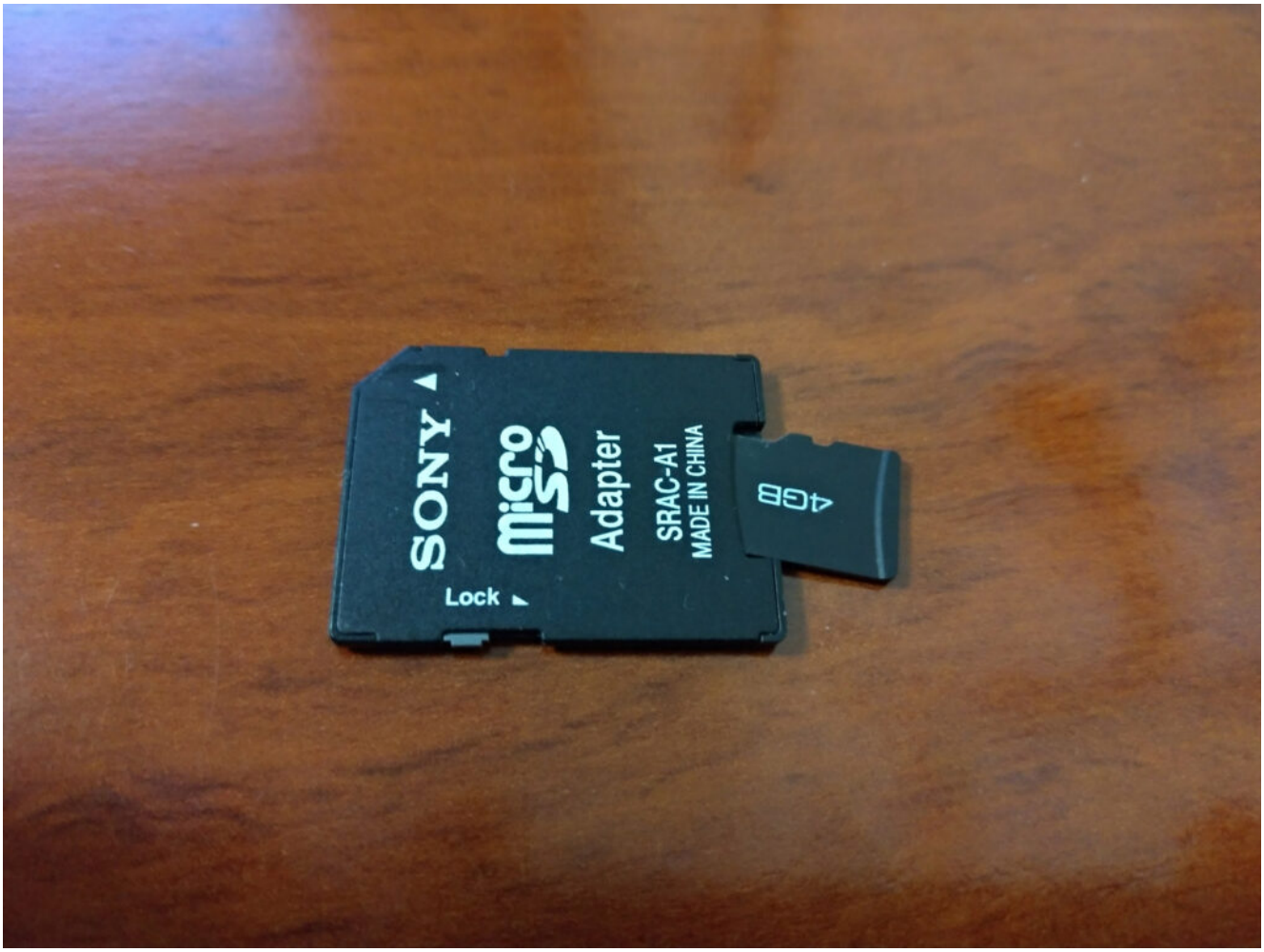
Schema elettronico



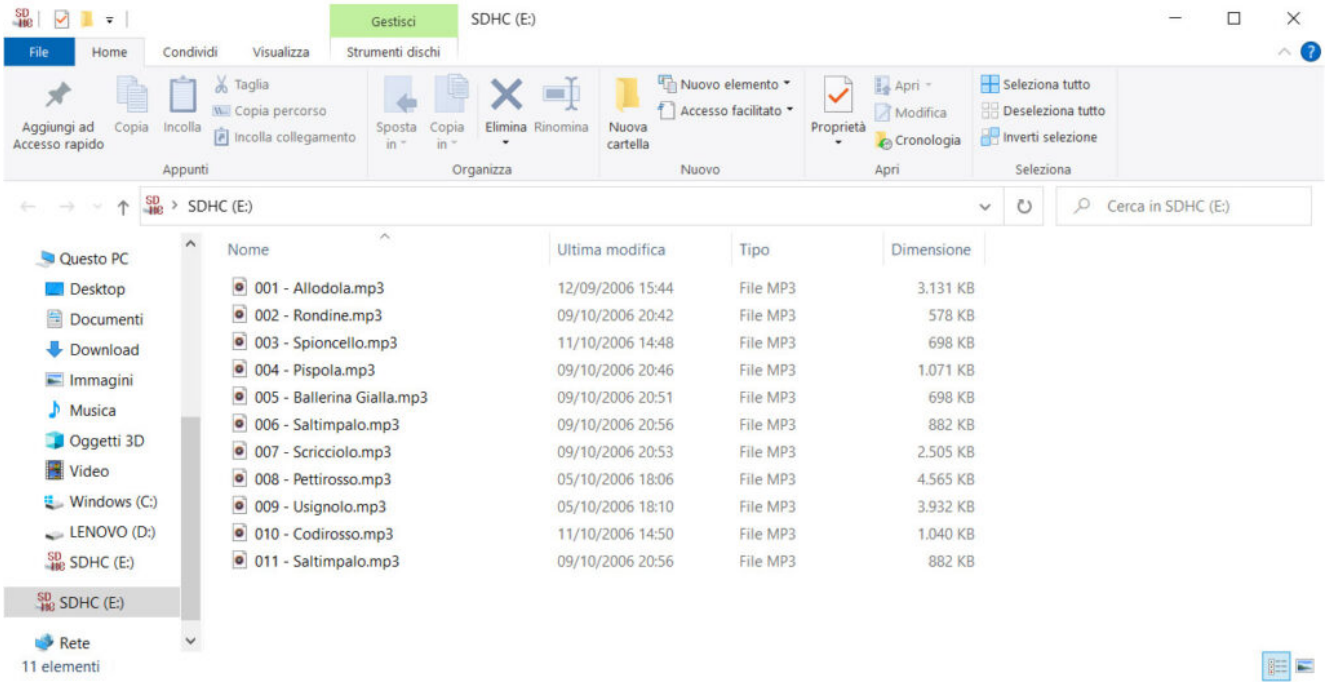
Preparazione Scheda MicroSD

La MicroSD card potrà essere di dimensioni massime di 32Gb è

deve essere formattata con filesystem Fat16 o Fat32 oppure se possedete un Mac OS X, selezionare ExFat e poi puoi copiare i file MP3 che desiderate.
E’ conveniente che i file audio siano numerati per definirne l’ordine di esecuazione prograssiva. Al termine del trasferimento, si potrà estrarre la SD Card dal computer per puoi inserla nel DFPlayer mini.



Micro SD



Struttura della MicroSD

Codice sorgente

Quanto Tempo Hai Premuto il Pulsante?

Obiettivo: Determinare per quanto tempo un pulsante è stato premuto.

Componenti elettronici:

- Arduino UNO

- Breadboard
- Pulsante
- Resistenza (1kOhm)

Prerequisiti

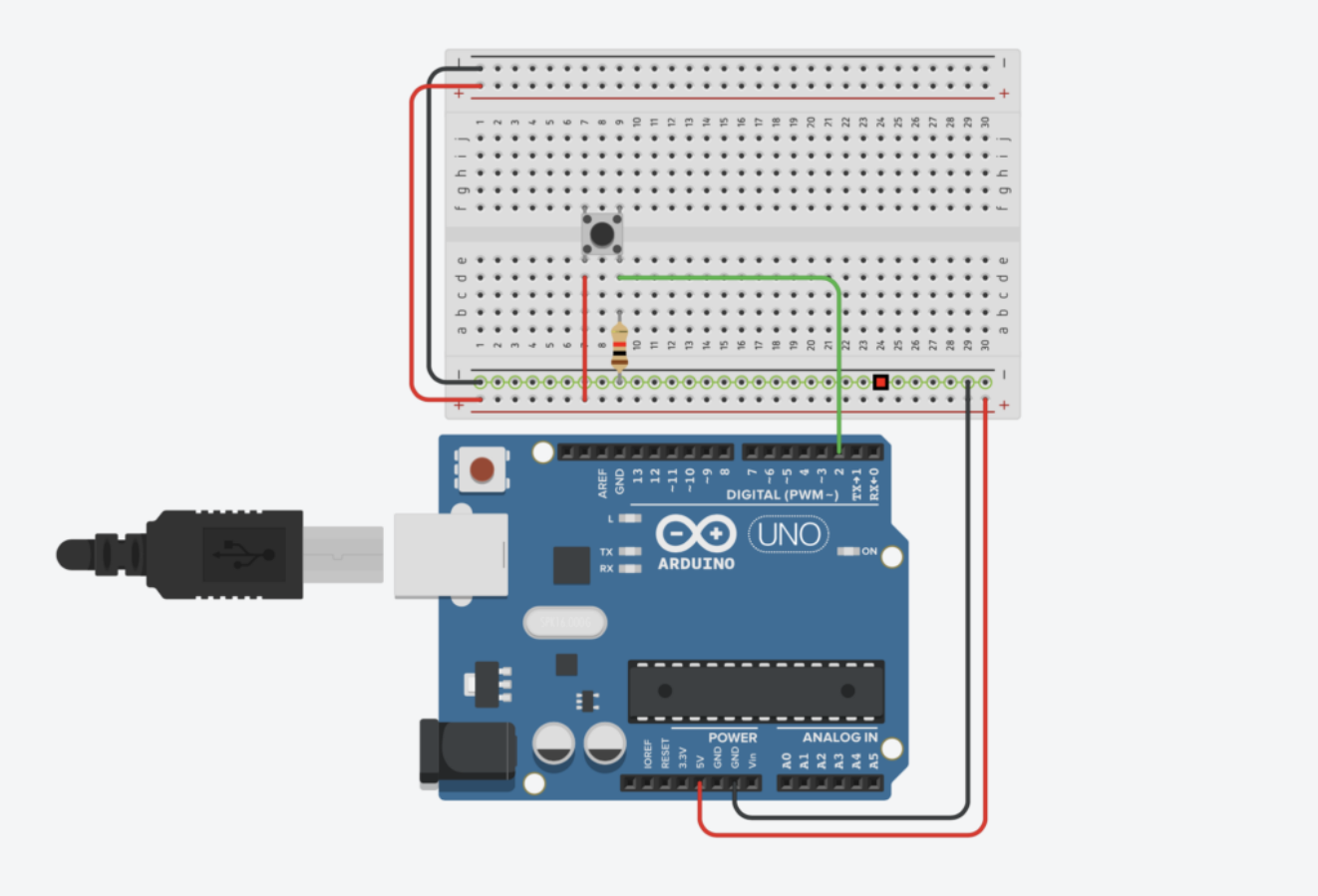
Blinking Led Senza Delay: MILLIS()

Pulsante come Interruttore

Teoria: Poter misurare il tempo di pressione di un tasto può risultare utile in molte applicazioni. Infatti, questa informazione permette di discriminare le differenti modalità di iterazione con il pulsante come il click (tasto premuto) ed il long click (tasto premuto a lungo). Potere discriminare questi comportamenti permette di abilitare il pulsante a differenti funzioni. Ad esempio il single click potrebbe essere utilizzato per accendere un led mentre il long click potrebbe essere utile per farlo lampeggiare.
Dal punto di vista hardware il circuito necessario per realizzare questa applicazione è molto semplice ed è costituito dal singolo pulsante collegato a vcc e al ground mediante resistenza di pull-down.
Elemento centrale di questa esercitazione è la scrittura di un codice corretto che permetta di misurare esattamente lo scorrere quel tempo. Questo codice si basa sull'impiego di due elementi fondamentali:

- La funzione millis: questa funzione restituisce il numero di millisecondi che sono passati da quando la board Arduino ha eseguito il programma corrente. Questo numero si riazzera dopo circa 50 giorni. L'impiego di questa funzione è fondamentale per misurare il tempo di pressione del pulsante. Nel dettaglio, questa operazione può essere svolta semplicemente eseguendo la differenza tra le misure temporali prese quando il pulsante è premuto e quando il pulsante è rilasciato.
- La variabile di stato: questa variabile (di natura globale) permette di determinare quando un pulsante è premuto e quando il pulsante è rilasciato. Nello specifico grazie a questa variabile è possibile determinare un passaggio dallo stato logico basso a quello alto e viceversa.

Collegamento Circuitale:



Collegamento Circuitale

Codice: A seguire viene riportato il codice utilizzato per determinare il tempo di pressione di un pulsante. Nello specifico il codice utilizza la variabile di stato "*valButtonOld*" per memorizzare lo stato del pulsante relativo al ciclo passato.
Quando i valori di "*valButton*" *e* "*valButtonOld*" differiscono, allora c'è stato un passaggio di stato.

ValButtonOld	ValButton	Evento
LOW	HIGH	Il pulsante è stato premuto
HIGH	LOW	Il pulsante è stato rilasciato

Nel caso specifico del passaggio di stato viene effettuata una misura del tempo trascorso mediante la funzione millis(). Per determinare il tempo trascorso basta semplicemente effettuare una differenza tra le due misure realizzate.

Personalizzazioni: E' possibile modificare l'hardware introducendo due led. Quando il pulsante viene premuto per meno di un secondo deve accendersi il primo led, quando invece il pulsante viene premuto per più di un secondo deve accendersi il secondo.

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:

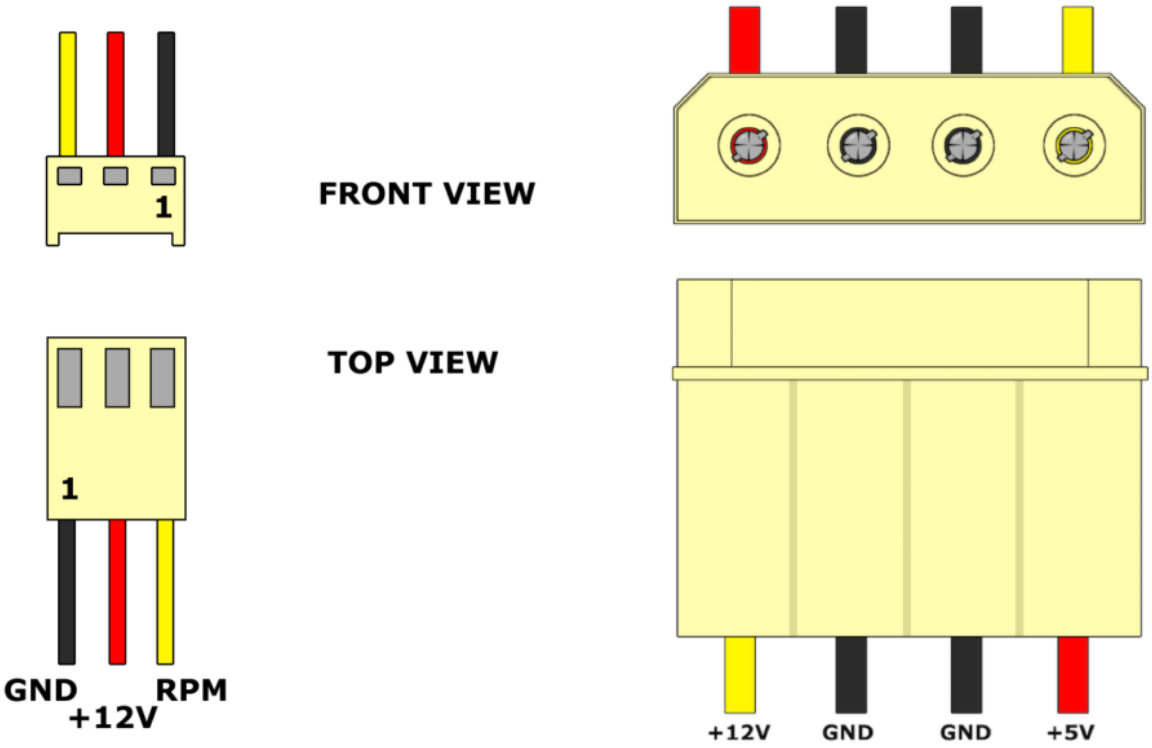
- Arduino UNO
- PowerShield 6+6 T800
- 1 Ventola da PC
- 1 Alimentatore da Banco (corrente continua)
- 1 Trimmer per controllare la velocità della ventola

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 stato utilizzata una ventola a 3 cavi:

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



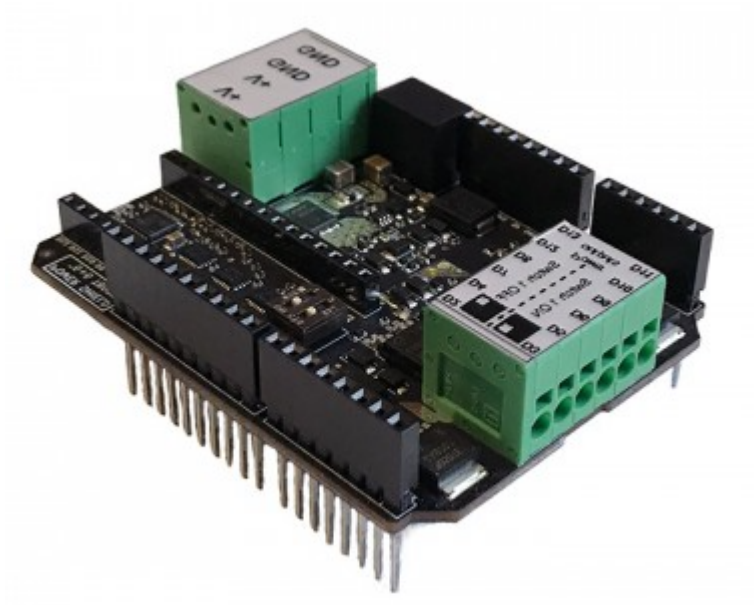
Poiché la tensione di alimentazione è pari a 12 volt non è possibile utilizzare esclusivamente la scheda Arduino per pilotare la ventola ma è necessaria una fonte di alimentazione esterna ed un dispositivo di controllo (e.g., Relè, Transistor). Nel caso specifico è stato utilizzato la PowerShield 6+6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

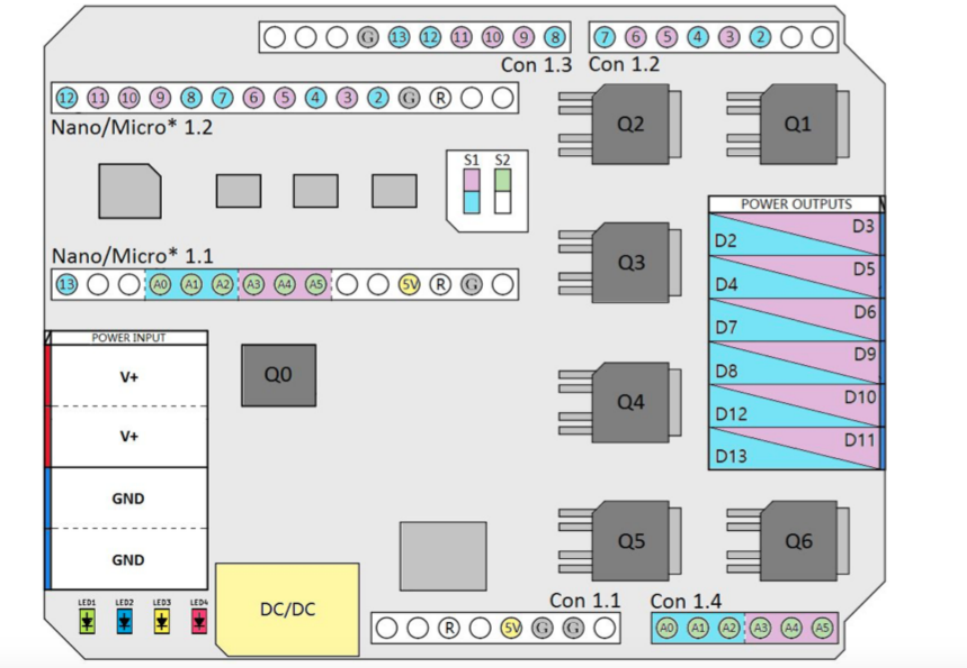
uscita e garantisce il supporto PWM fornito da Arduino. In seguito sono riportate le principali specifiche tecniche della scheda:

- Tensione in ingresso fino a 32V con un supporto massimo di corrente pari a 25A.
- Le uscite possono controllare differenti tensioni
- PWM fino a 100kHz
- Fino a 7A per canale
- Autoalimentazione della scheda controllore Arduino.
- Circuito Integrato di MultiProtezione

Sono riportate in seguito le immagini relative alla scheda T800 e alla sua configurazione dei PIN



PowerShield 6+6 T800:



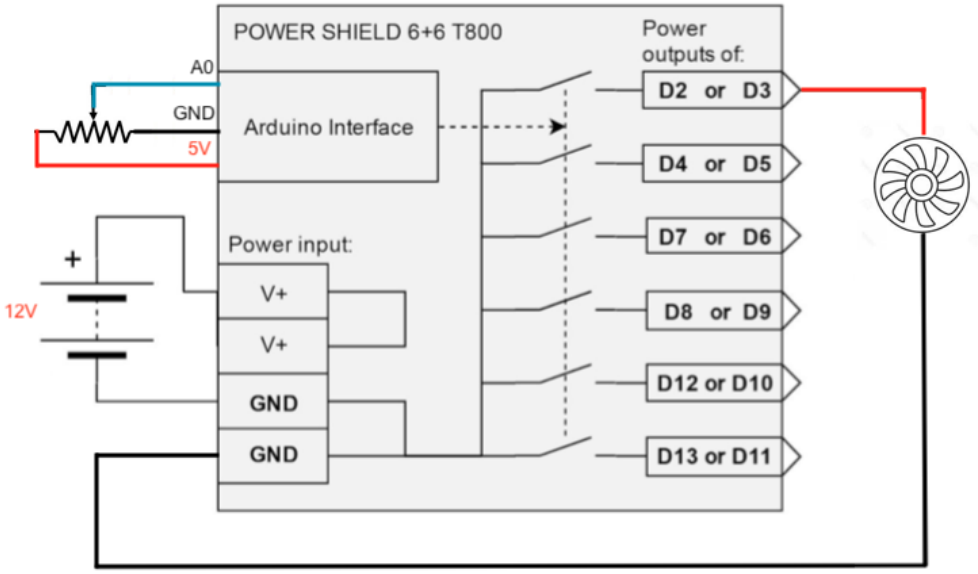
PowerShield 6+6 T800 Pin Configuration

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.

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 1kOhm (per filtro passa basso)
- 1 Condensatore elettrolitico da 22uF (per filtro passa basso)

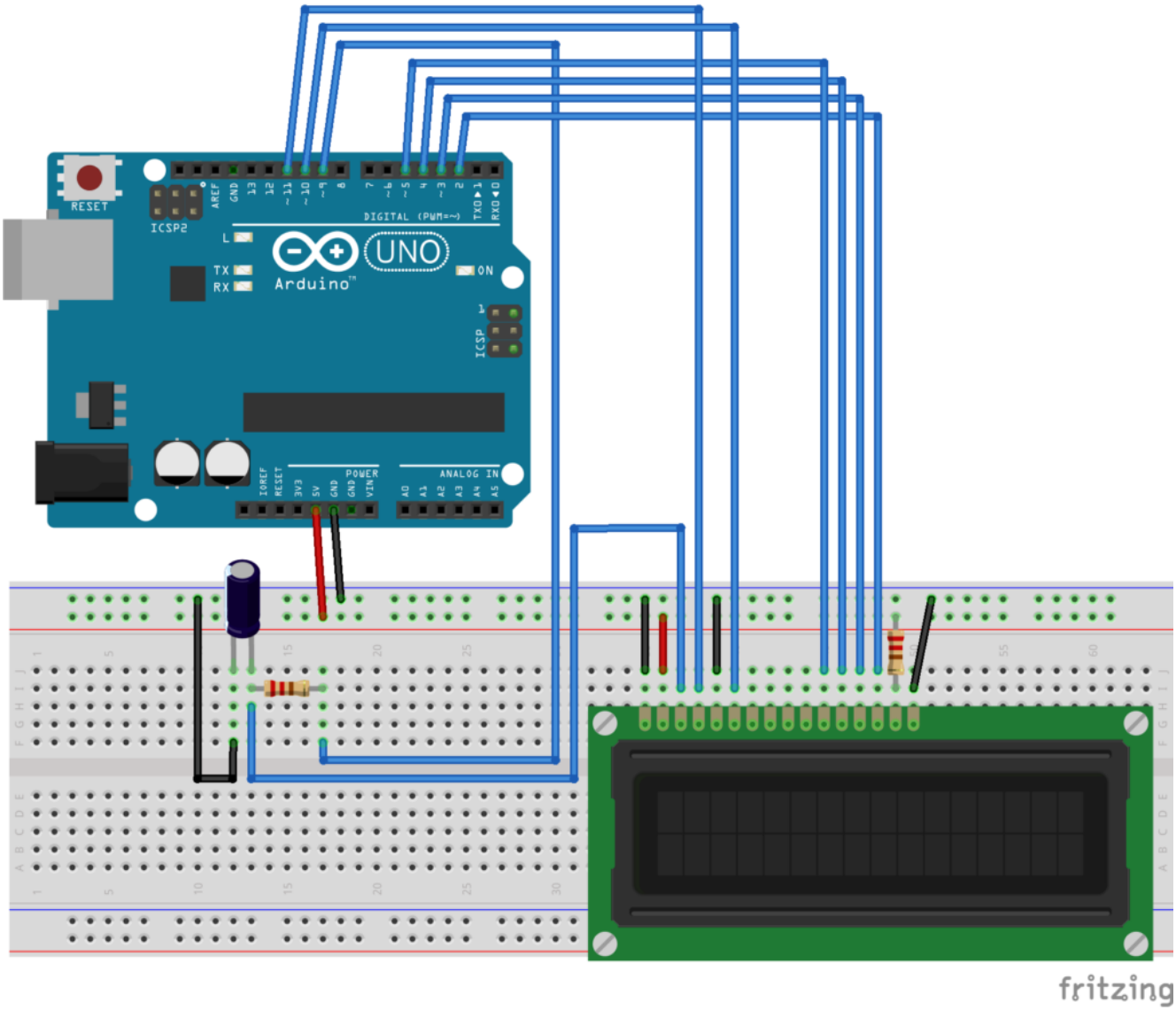
Pre-requisiti:

Come Collegare un Display LCD ad Arduino

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 una dispositivo elettronico che utilizza un display LDC. 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.

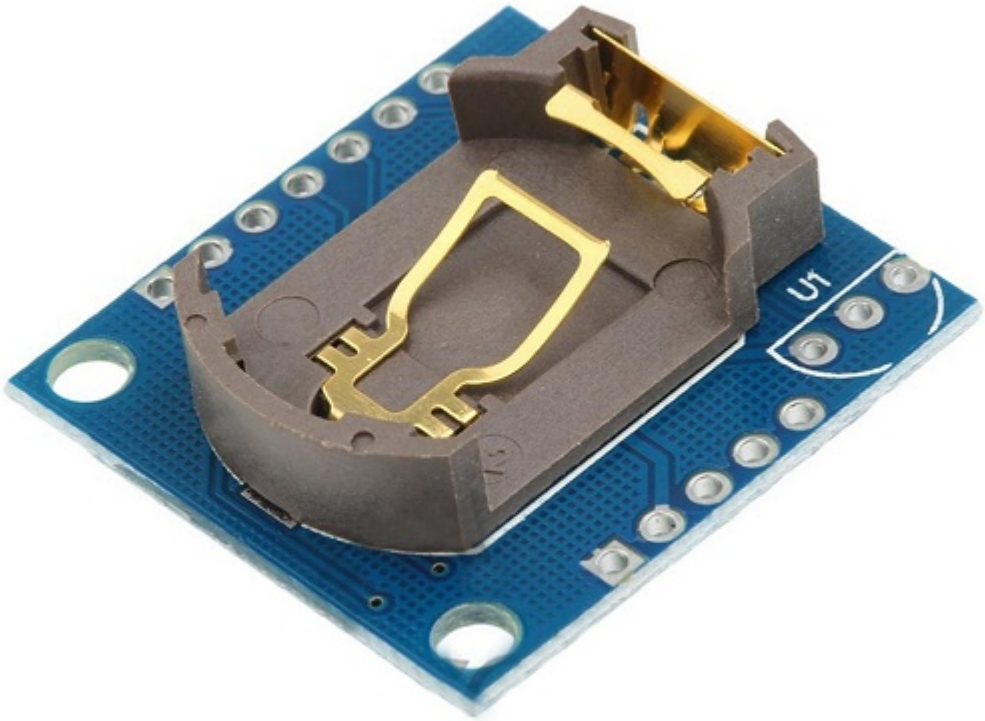
Real Time Clock per Arduino

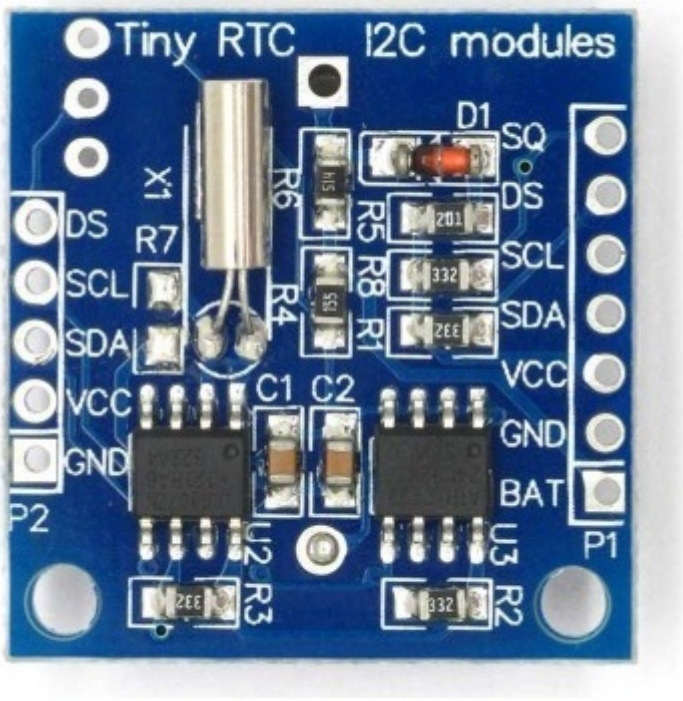
Obiettivo: Scrivere e poi leggere l'ora e la data da una scheda RTC Real Time Clock.

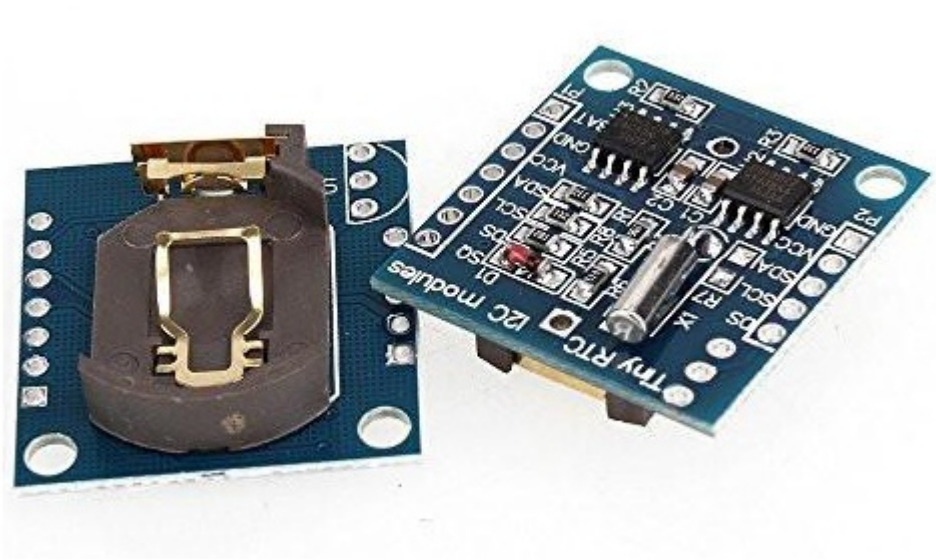
Componenti elettronici:

- Arduino UNO
- 1 TinyRTC con chip DS1307 (link1 – link2)

Teoria: Alla base di questa esercitazione c'è il modulo TinyRTC con chip DS1307 ed EEPROM: 24C32 32K I^2C con interfaccia I2C.


U1


Tiny RTC
I2C modules
D1
SQ
DS
SCL
SDA
VCC
GND
BAT
X1
R6
R5
R8
R1
R7
R4
C1
C2
U2
U3
R3
R2
P2
P1



Caratteristiche Hardware

- Chip RTC: DS1307
- Memoria EEPROM: 24C32 32K I2C
- Comunicazione: I2C (possibilità di collegare altri dispositivi I2C in cascata)
- Batteria di backup supportata (NON inclusa): LIR2032
- Ciclo di funzionamento del modulo (a piena carica): 1 anno
- Dimensioni: 28 x 28 x 9mm
- Peso: 4g

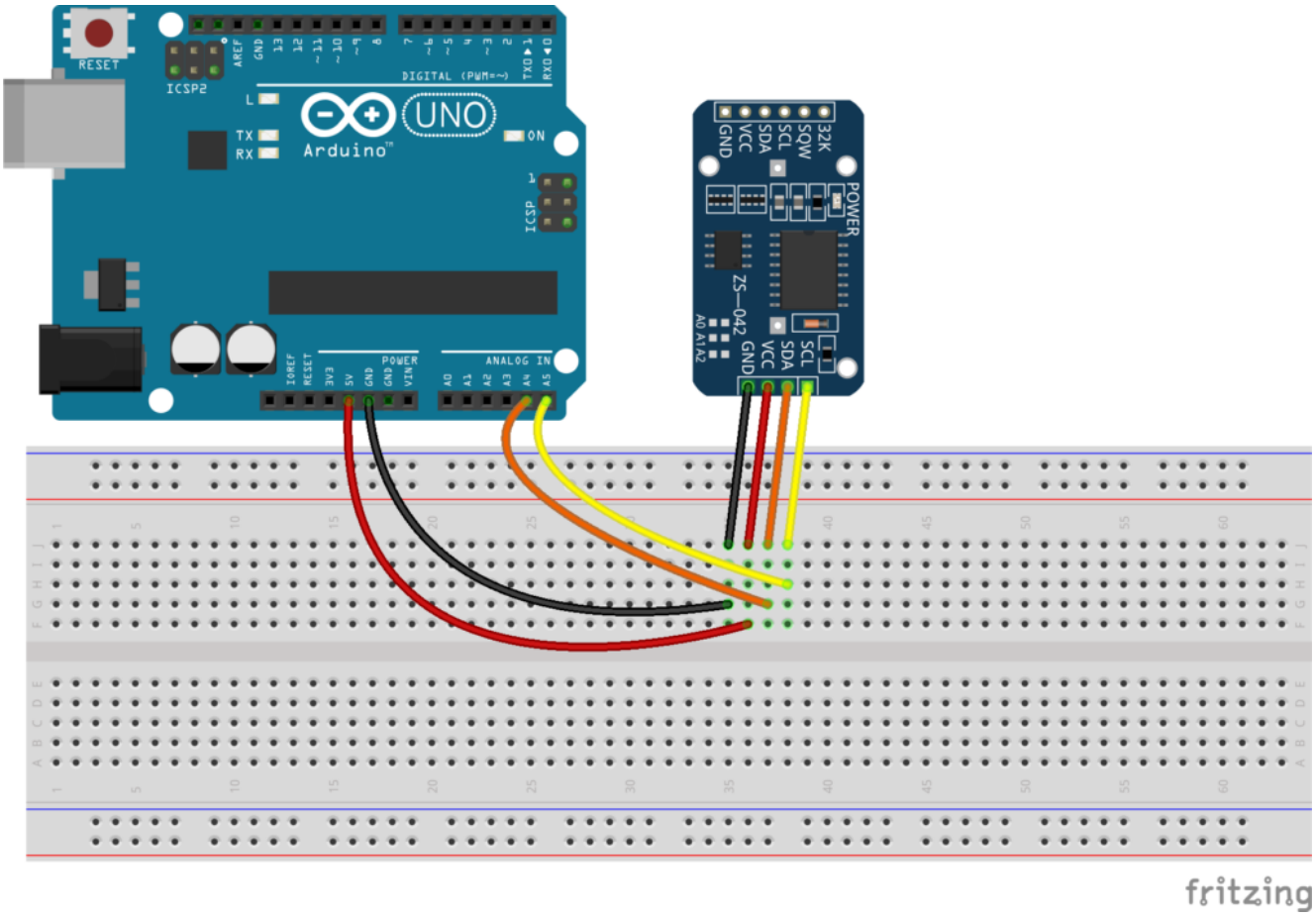
Se avete bisogno nei vostri progetti di un sistema per la gestione di data e orario questa scheda è quello che fa per voi.

Per utilizzare tale scheda viene utilizzata la libreria RTClib.h

Tramite tale libreria è possibile settare nella funzione di setup la data e l'ora personalizzata (riga codice n.25), oppure quella di compilazione (riga codice n.22) , la prima volta che si carica il codice in Arduino, in modo tale che la scheda TinyRTC si configuri con un orario e data opportuni, poi si deve ricaricare lo stesso sketch commentando la riga di codice (n.22 o n.25).

Da questo momento in poi Arduino sarà in grado di utilizzare un orologio real time in grado di gestire ore, minuti, secondi, anni, mesi e
giorni!

Collegamento Circuitale:



Collegamento Circuitale

Codice

Leggere informazioni dal GPS BN-880 o uBlox M8N

Obiettivo:

Leggere tutte le informazioni provenienti dal GPS BN-880 utilizzando la libreria TinyGPS++

Video:

Componenti elettronici:

- Arduino UNO
- 1 GPS BN-880 (link 1 – link 2) o uBlox NEO-7M/M8N (link 3 – link 4)

Teoria:

Alla base di questa esercitazione c’è il modulo GPS Baitian BN-880, ma andrebbe benissimo anche un modulo uBlox NEO-7M o M8N o equivalenti.

- 

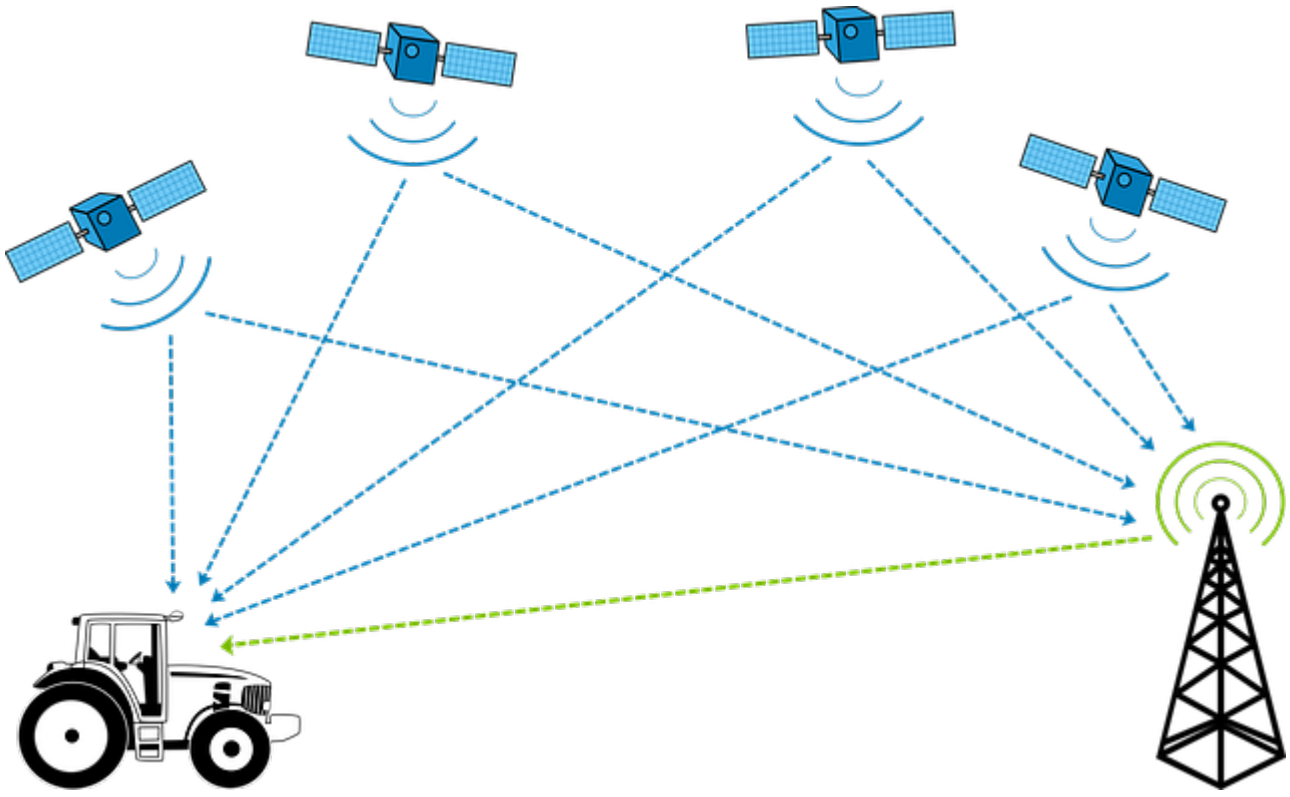
- 

Il modulo in oggetto oltre ad avere integrato il GPS,
oggetto dell’esercitazione, ha anche integrato una bussola (compass) HMC5883l comunicante
tramite il bus/protocollo I2C.

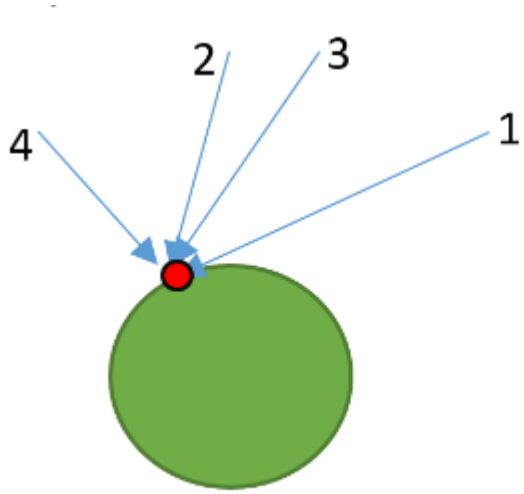
Ecco alcune caratteristiche:

- Inexpensive,
  light, Dual Mode GPS and GPS/Compass using UBLOX M8N module
- Receiving
  Format: GPS, GLONASS, Galileo, BeiDou, QZSS and SBAS
- Frequency:
  GPS L1,GLONASS L1,BeiDou B1,SBAS L1,Galileo E1
- Channels: 72
- Data Protocol: NMEA-0183 or UBX, Default
  NMEA-0183
- Single GNSS: 1Hz-18Hz
- Concurrent GNSS: 1Hz-10Hz
- Compass IC: HMC5883L

Il sistema GPS (Global Positioning System):



GPS sta per **Global Positioning System** cioè un Sistema di Posizionamento Globale, la cui funzione primaria è quella di determinare con precisione la posizione sul pianeta Terra di un ricevitore (come ad esempio uno smartphone o un apparecchio dedicato). Il ricevitore GPS riceve continuamente segnali da lontani satelliti in orbita attorno alla Terra. Attorno alla Terra, a circa 20.200 km di altitudine, orbitano 31 satelliti del sistema GPS. Ogni satellite ha al suo interno un precisissimo orologio atomico. Ogni satellite invia continuamente un segnale radio contenente la sua posizione, e l'orario di trasmissione del segnale. Il ricevitore GPS confronta questi segnali, e tramite alcune equazioni riesce a stabilire la sua posizione.



Il pallino rosso è il ricevitore GPS. Alle ore 12:00 tutti i satelliti GPS inviano un segnale che dice "io mi trovo nella posizione XYZ, e
sono le 12:00". Poiché i satelliti hanno distanze diverse dal ricevitore, il
segnale "io mi trovo nella posizione XYZ, e sono le 12:00" arriva al ricevitore
in momenti diversi: quello del satellite 4 arriva per primo; poi quello del

satellite 2; poi 3; e poi 1. Sapendo quanto ci ha messo il segnale ad arrivare,
e quindi la distanza fra il ricevitore e i vari satelliti, il ricevitore GPS,
con una relativamente semplice triangolazione, riesce a stimare la propria
posizione sul pianeta Terra.

Che cosa trasmette il modulo GPS:

Il modulo GPS trasmette tramite Seriale TTL (0-5 Volt) delle "sentenze" tramite il protocollo NMEA0183.

NMEA 0183 (o più comunemente NMEA) è uno standard di comunicazione di dati utilizzato soprattutto in nautica e nella comunicazione di dati satellitari GPS. L'ente che gestisce e sviluppa il protocollo è la National Marine Electronics Association. Questo protocollo si basa sul principio che la fonte, detta talker, può soltanto inviare i dati (sentences) e la ricevente, detta listener, può soltanto riceverli.

Ad esempio la seguente sentenza $GGA trasmette tantissime informazioni oltre alla Latitudine e Longitudine:

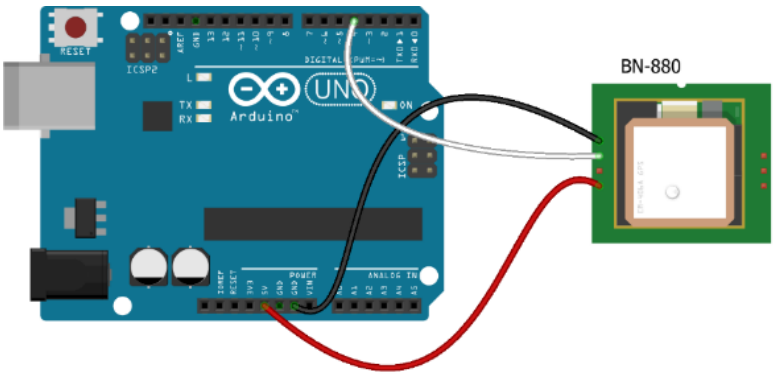
GGA **Global Positioning System Fix Data. Time, Position and fix related data for a GPS receiver**

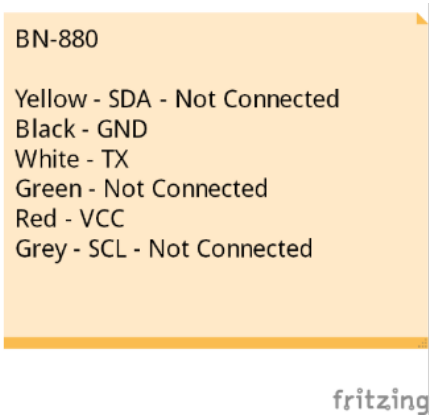
```
                                                      11
       1         2       3 4        5 6 7  8   9  10 |  12 13  14   15
       |         |       | |        | | |  |   |   | |   | |   |    |
$--GGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh
```

1) Time (UTC)
2) Latitude
3) N or S (North or South)
4) Longitude
5) E or W (East or West)
6) GPS Quality Indicator,
   0 - fix not available,
   1 - GPS fix,
   2 - Differential GPS fix
7) Number of satellites in view, 00 - 12
8) Horizontal Dilution of precision
9) Antenna Altitude above/below mean-sea-level (geoid)
10) Units of antenna altitude, meters
11) Geoidal separation, the difference between the WGS-84 earth ellipsoid and mean-sea-level (geoid), "-" means mean-sea-level below ellipsoid
12) Units of geoidal separation, meters
13) Age of differential GPS data, time in seconds since last SC104 type 1 or 9 update, null field when DGPS is not used
14) Differential reference station ID, 0000-1023
15) Checksum

Tramite
la libreria TinyGPS++ è possibile intercettare tutti i tipi di sentenze GPS e
inoltre nell'ultima versione anche di elaborare le sentenze particolari, ad
esempio prelevate da altre fonti NMEA0183.

Schema Elettronico (Fritzing):





Codice:

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 costituta 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:

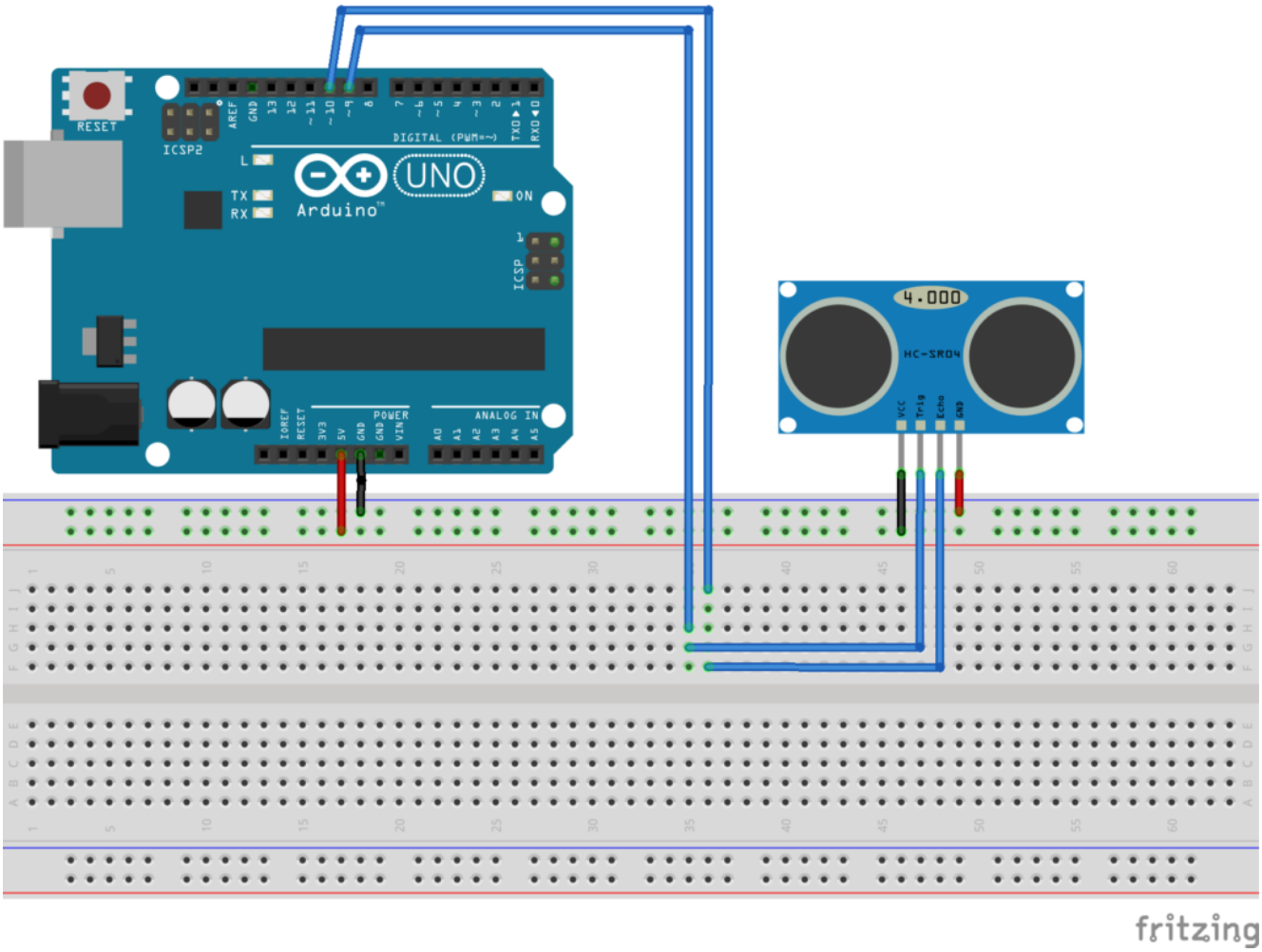
- **int pinEcho**: il pin di echo
- **int pinTrigger**: il pin di trigger

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:



Collegamento Circuitale

Codice:

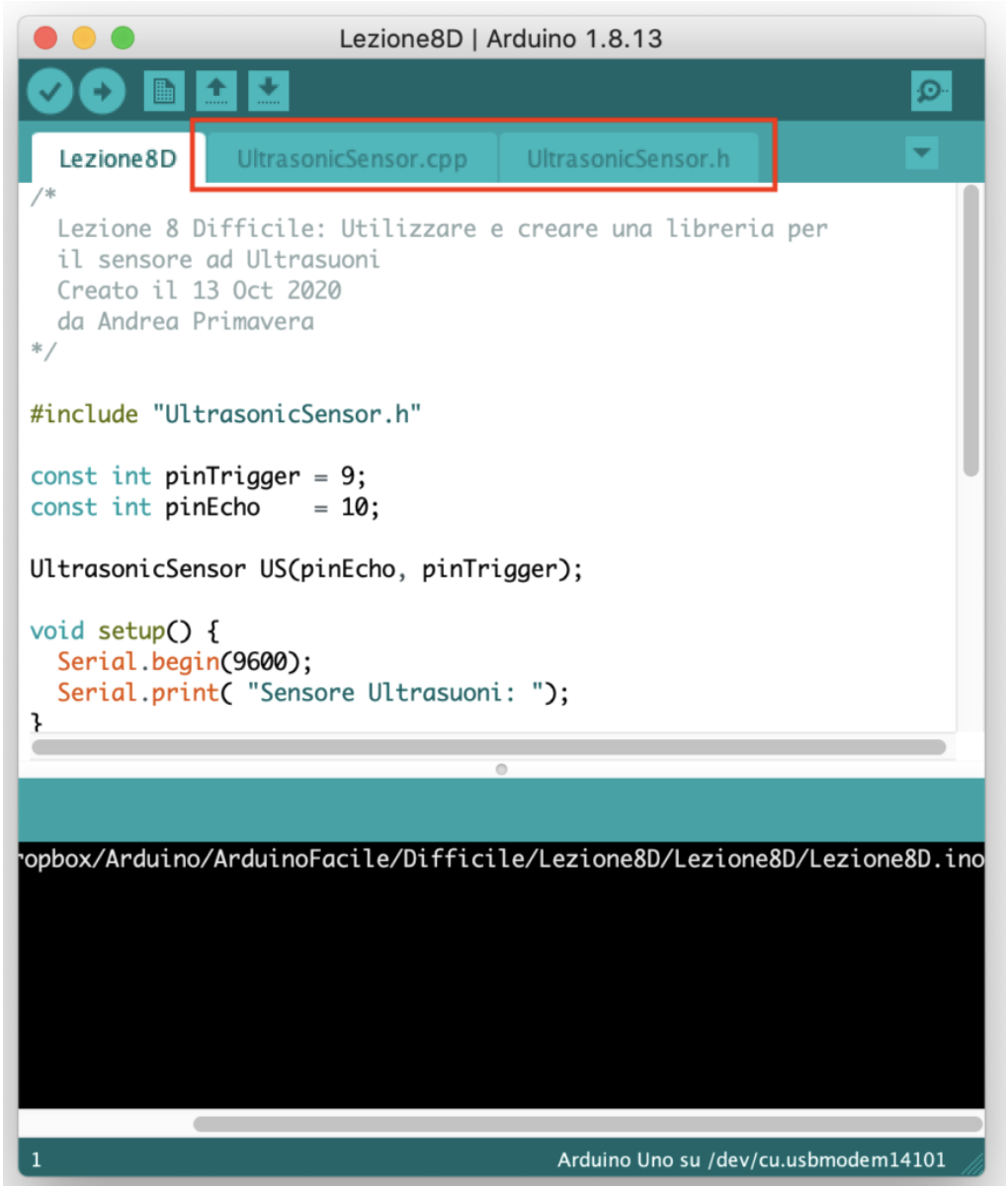
Vengono in seguito riportate le tre porzioni di codice utilizzate per creare la funzione di libreria relativa al sensore ad ultrasuoni HC-SR04.

- 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: ");
}
ropbox/Arduino/ArduinoFacile/Difficile/Lezione8D/Lezione8D/Lezione8D.ino
1
Arduino Uno su /dev/cu.usbmodem14101