Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Titolo Progetto: “Teaching and Learning Arduinos in Vocational Training”

Acronimo Progetto: “ ARDUinVET ”

Progetto N: “2020-1-TR01-KA202-093762”

*** *ARDUinVET Guidebook* ***

"Bu proje Erasmus+ Programı kapsamında Avrupa Komisyonu tarafından desteklenmektedir. Ancak burada yer alan görüşlerden Avrupa Komisyonu ve Türkiye Ulusal Ajansı sorumlu tutulamaz".

"Questo progetto è finanziato dal Programma Erasmus+ dell'Unione Europea. Tuttavia, la Commissione europea e l'Agenzia nazionale turca non possono essere ritenute responsabili per l'uso che può essere fatto delle informazioni in esso contenute".

COORDINATORE:

Gölbaşı Mesleki ve Teknik Anadolu Lisesi (Ankara / TURCHIA)

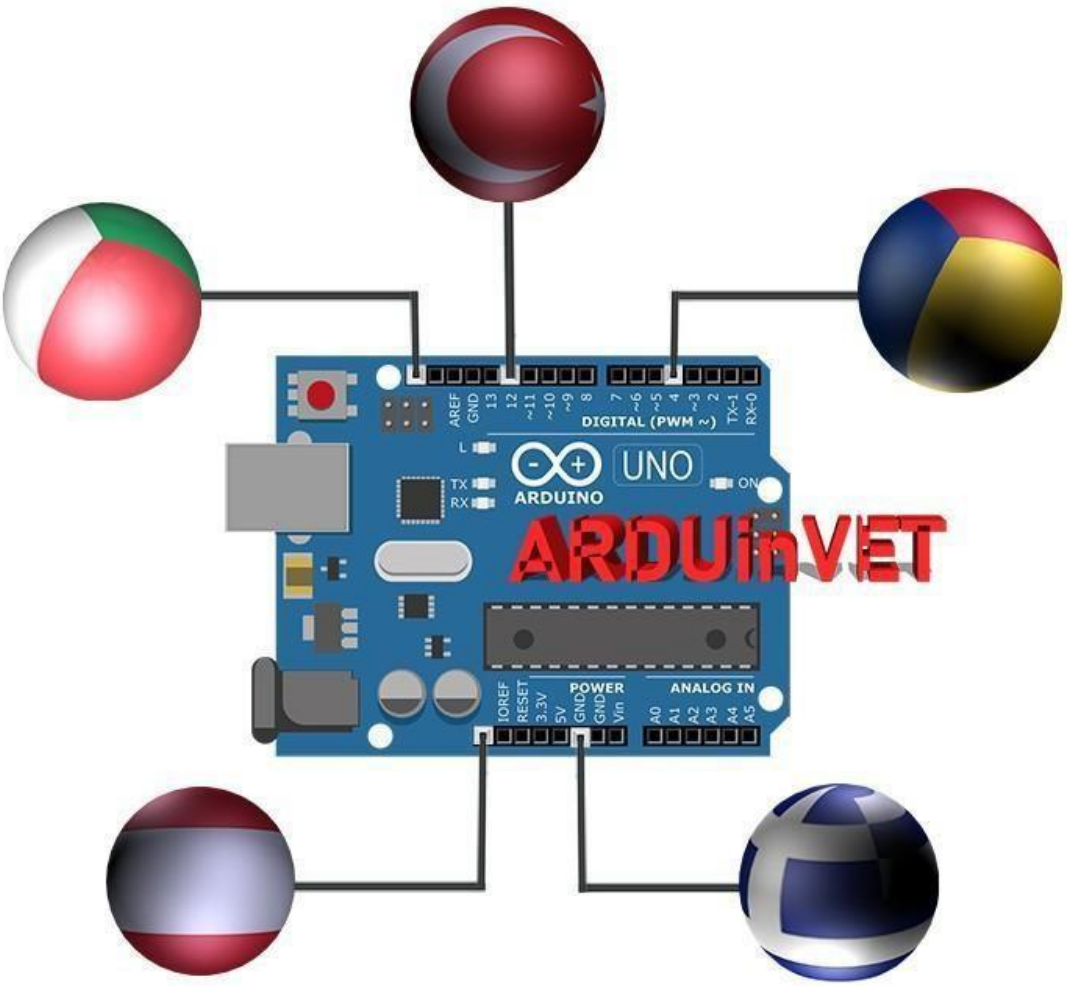
PARTNER:

2 EK Peiraia (Pireo / Grecia)

Liceul Tehnologic Grigore Moisil Braila (Braila / Romania)

Istituto di Istruzione Superiore Einstein De Lorenzo Potenza (Potenza /Italy)

HTBLA Wolfsberg / (College for Engineering Wolfsberg) (Wolfsberg/ Austria)



Sintesi del progetto

"Insegnamento e apprendimento di Arduino nella formazione professionale".

La tecnologia si sviluppa molto rapidamente giorno dopo giorno. È un dato di fatto che gli sviluppi tecnologici costringono l'uomo a svilupparsi continuamente nella sua professione. Più di ogni altro campo, le tecnologie elettroniche, dell'informazione e della comunicazione si stanno sviluppando più rapidamente. Questi sviluppi hanno portato alla produzione di sistemi di controllo più complessi.
Per controllare questi sistemi complessi si utilizzano ora gli Arduino.

Poiché siamo istituzioni educative che insegnano la tecnologia, dobbiamo seguire gli sviluppi del mondo. Il progetto "Teaching and Learning Arduinos in Vocational Training" mira ad adattare le applicazioni di Arduino alla formazione professionale e a sviluppare un set di formazione più efficiente e una guida per i laboratori e le officine degli studenti dell'istruzione professionale e tecnica.

L'obiettivo principale di questo progetto è quello di sviluppare una guida alle buone pratiche e un set di formazione Arduino, di presentare i modelli di formazione Arduino agli altri partecipanti durante le loro visite nel Paese ospitante, di confrontare i diversi sistemi educativi e i metodi di formazione con le altre scuole partecipanti e di condividere le migliori pratiche.
Partecipano ai progetti: insegnanti di elettricità, elettronica, TIC e automazione. I partecipanti insegnano Arduino nelle scuole di formazione professionale. Ci sono un coordinatore e un totale di quattro partner nel progetto e i partner sono scuole professionali provenienti da Turchia, Italia, Romania, Austria e Grecia. Il numero totale di mobilità nel progetto è di 40 persone. Un numerototale di 5 TPM sarà realizzato nel corso del progetto e ogni partner parteciperà a ogni TPM con2 partecipanti. Un TPM si terrà in ogni Paese.

Con le attività del progetto, si garantirà la condivisione delle migliori pratiche per l'insegnamento di Arduino e i partner le adatteranno ai loro ambienti educativi. I partner del progetto produrranno kit sperimentali Arduino e una guida alle migliori pratiche. Le nostre attività di progetto mirano a fornire un ambiente di apprendimento e insegnamento di successo per tutti gli insegnanti e gli studenti dell'IFP. Come condizione per il successo dell'apprendimento, gli insegnanti devono rafforzare il loro ruolo nel facilitare l'apprendimento. Gli insegnanti hanno bisogno di nuovi kit sperimentali e moduli per l'innovazione, il lavoro di gruppo, il feedback e la valutazione. Agli insegnanti deve essere data questa opportunità di sviluppo professionale continuo.

I principali risultati del progetto sono: un set di formazione Arduino per lezioni di Arduino nell'istruzione professionale e un libro guida per questo set, un sito web del progetto, un DVD del progetto. La metodologia da utilizzare nell'esecuzione del progetto è "Make-Develop-Share" (“Crea-Sviluppa-Condividi”). Nel nostro progetto, le buone pratiche saranno prima realizzate, poi sviluppate e infine condivise. Nel nostro progetto tutto è aperto e trasparente. Ogni compito e responsabilità sarà registrato o scritto nel progetto. I prodotti del progetto non sono solo prodotti scritti o documentari, ma anche un set di formazione pratica su Arduino e kit.

A lungo termine, ci proponiamo di diffondere il progetto a insegnanti, studenti e scuole di formazione professionale, istituzioni educative locali, mercato del lavoro elettronico e ICT. Crediamo che i risultati e i prodotti dei nostri progetti avranno un impatto a breve e lungo termine sull'istruzione professionale e sul mercato del lavoro grazie alle attività di divulgazione. Il budget totale del progetto con 5 partner è di 59.000 euro.

CONTENUTI:

Co-funded by the
Erasmus+ Programme
of the European Union

MODULI DI PROGETTO e KIT di ARDUinVET

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Titolo Progetto: “Teaching and Learning Arduinos in Vocational Training”

Acronimo Progetto: “ ARDUinVET ”

Progetto N: “2020-1-TR01-KA202-093762”

INTRODUZIONE ARDUINO
(Fondamenti di ARDUINO)

INTRODURRE l'ARDUINO

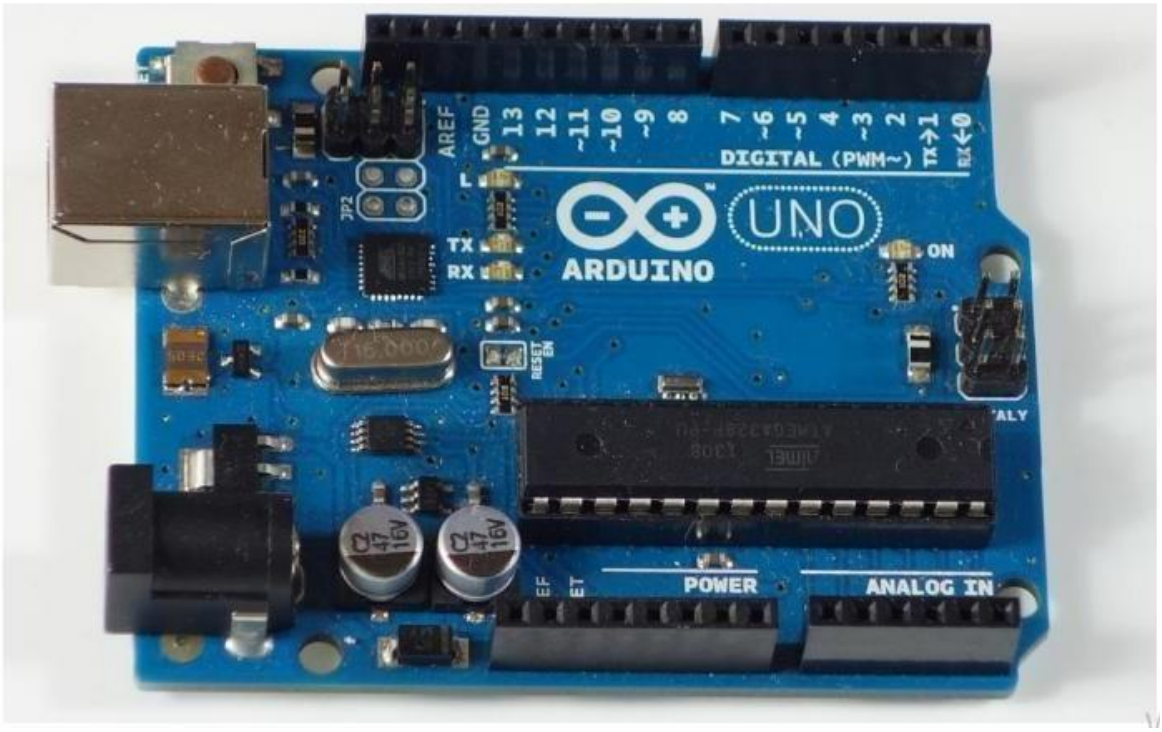
Arduino è una piattaforma di prototipi (open-source) basata su un hardware e un software di facile utilizzo. Consiste in una scheda di circuito che può essere programmata (chiamata microcontrollore) e in un software già pronto chiamato Arduino IDE (Integrated Development Environment), che viene utilizzato per scrivere e caricare il codice del computer sulla scheda fisica.

In altre parole, Arduino è un piccolo computer che si può programmare per leggere informazioni dal mondo circostante e inviare comandi al mondo esterno. Tutto questo è possibile perché è possibile collegare diversi dispositivi e componenti ad Arduino per fare ciò che si desidera. Si possono realizzare progetti straordinari, non c'è limite a ciò che si può fare e con l'immaginazione tutto è possibile!

In parole povere, Arduino è un piccolo sistema informatico che può essere programmato con le vostre istruzioni per interagire con varie forme di input e output. L'attuale modello di scheda Arduino, la Uno, ha dimensioni piuttosto ridotte rispetto alla mano umana media.

Che cos'è Arduino?

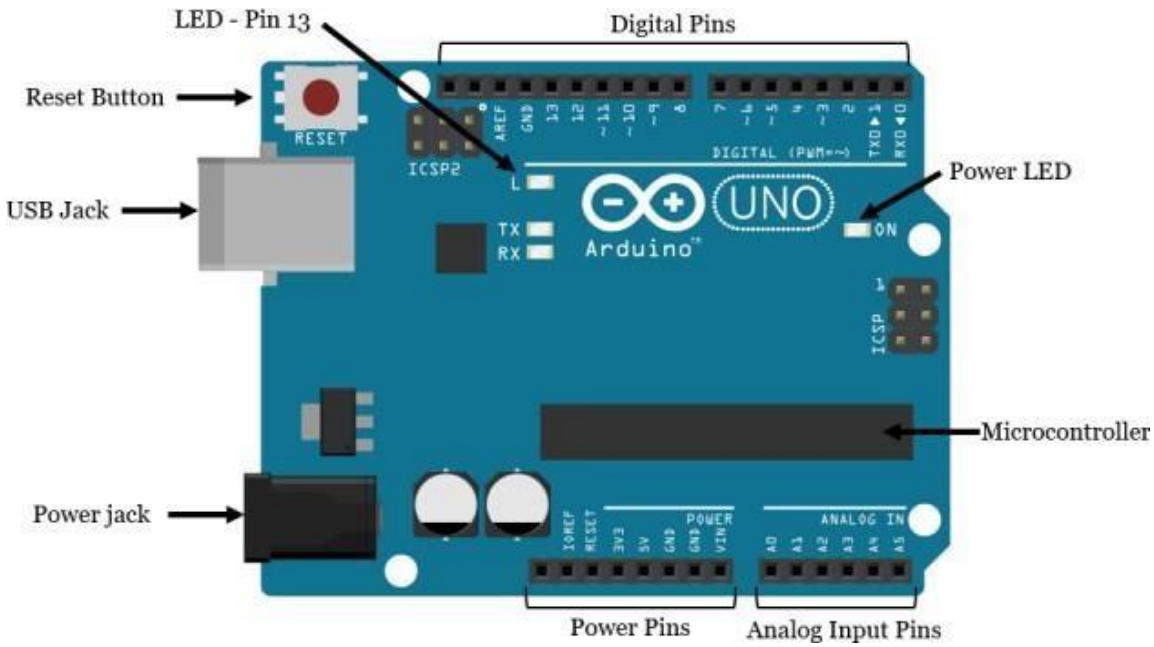
Arduino è la scheda mostrata nella figura seguente.



In pratica, si tratta di una piccola scheda di sviluppo con un cervello (noto anche come microcontrollore) che si può collegare ai circuiti elettrici. In questo modo è facile leggere gli ingressi - leggere i dati dall'esterno - e controllare le uscite - inviare un comando all'esterno. Il cervello di questa scheda (Arduino Uno) è un chip ATmega328p in cui è possibile memorizzare i programmi che diranno ad Arduino cosa fare.

Esplorare la scheda Arduino Uno

Nella figura che segue, è possibile vedere l'etichetta di una scheda Arduino. Vediamo cosa fa ogni parte.



• **Microcontrollore:** l'ATmega328p è il cervello di Arduino. Tutto ciò che è presente sulla scheda Arduino è destinato a supportare questo microcontrollore. È qui che si memorizzano i programmi per dire ad Arduino cosa fare.

• **Pin digitali:** Arduino dispone di 14 pin digitali, etichettati da 0 a 13, che possono fungere da ingressi ouscite.

o Quando sono impostati come ingressi, questi pin possono leggere la tensione. Possono leggere solo duestati: ALTO o BASSO.

o Quando sono impostati come uscite, questi pin possono applicare una tensione. Possono applicare solo 5V (HIGH) o 0V (LOW).

• **Pin PWM: Si** tratta di pin digitali contrassegnati da un ~ (pin 11, 10, 9, 6, 5 e 3). PWM è l'acronimo di "pulse width modulation" (modulazione dell'ampiezza degli impulsi) e consente ai pin digitali di emettere "finte" quantità di tensione variabili. Più avanti si scoprirà qualcosa di più sulla PWM.

• **Pin TX e RX:** pin digitali 0 e 1. La T sta per "trasmissione" e la R per "ricezione". Arduino utilizza questi pin per comunicare con altri dispositivi elettronici via seriale. Arduino utilizza questi pin anche per comunicare con il computer quando carica nuovo codice. Evitare di utilizzare questi pin per altri compiti diversi dalla comunicazione seriale, a meno che non si sia a corto di pin.

• **LED collegato al pin digitale 13:** è utile per facilitare il debug degli sketch di Arduino.

• **LED TX e RX:** questi LED lampeggiano quando vengono inviate informazioni tra il computer e Arduino.

• **Pin analogici:** i pin analogici sono etichettati da A0 a A5 e sono spesso utilizzati per leggere sensori analogici. Possono leggere diverse quantità di tensione comprese tra 0 e 5V. Inoltre, possono essere utilizzati come pin di uscita/ingresso digitale, come i pin digitali.

• **Pin di alimentazione:** Arduino fornisce 3,3 V o 5 V attraverso questi pin. Questo è molto utile perché la maggior parte dei componenti richiede 3,3 V o 5 V per funzionare. I pin etichettati come "GND" sono i pin di massa.

• **Pulsante di reset:** quando si preme questo pulsante, il programma attualmente in esecuzione in Arduino si riavvia. È presente anche un pin di reset accanto ai pin di alimentazione che funge da pulsante di reset. Quando si applica una piccola tensione a questo pin, si resetta Arduino.

• **LED di accensione:** si accende quando Arduino è alimentato.

• **Presa USB:** per caricare i programmi dal computer alla scheda Arduino è necessario un cavo USB A maschio - USB B maschio (mostrato nella figura seguente). Questo cavo alimenta anche Arduino.

• **Presa di alimentazione:** è possibile alimentare Arduino attraverso la presa di alimentazione. La tensione di ingresso consigliata è compresa tra 7V e 12V. Esistono diversi modi per alimentare Arduino: ad esempio, batterie ricaricabili, batterie usa e getta, batterie a muro e pannelli solari.

Caratteristiche di Arduino

Arduino Uno; ha il microcontrollore Atmel Atmega 328P e dispone anche di ingresso per la connessione USB, ingresso per la presa di alimentazione, pulsante di reset. Arduino ha tutto ciò che un microcontrollore dovrebbe avere.

Microcontrollore	Atmega328P
Tensione di lavoro	5V
Tensione di ingresso (consigliata)	7-12V
Tensione d'ingresso (limite)	6-20V
Pin di ingresso/uscita digitale	14
Pin di ingresso/uscita PWM	6
Pin di ingresso analogico	6
Corrente continua per pin di ingresso/uscita	20mA
Corrente CC per 3,3 V	50mA
Memoria flash	32 KB
Sram	2KB
EEPROM	1 KB
Velocità di clock	16 MHz
Lunghezza	68,6 mm
Larghezza	53,4 mm
Peso	25 g

Figura: Caratteristiche diArduino Uno

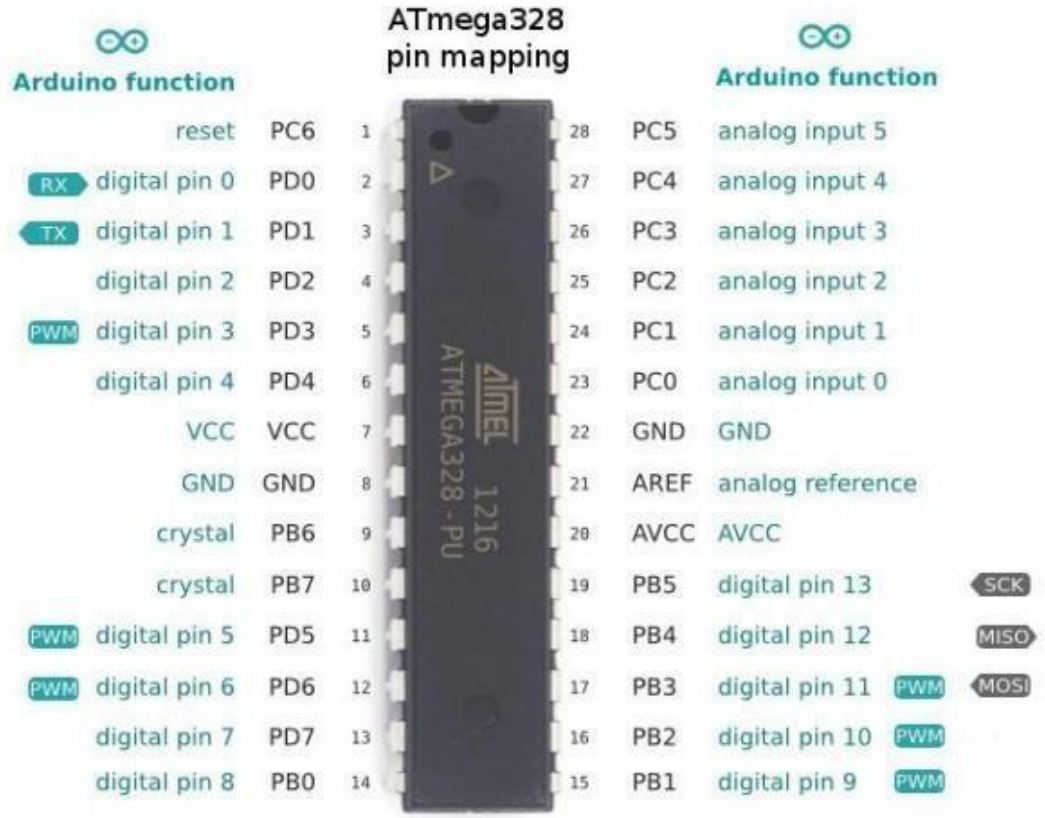


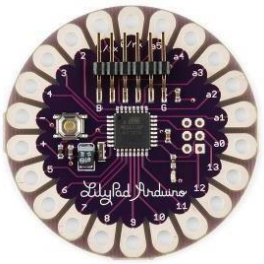
Figura: Pin dell'Atmega 328P

ALTRI TIPI DI ARDUINO

ARDUINO MEGA

È dotato di un microcontrollore Atmega 2560. Dispone di 54 pin di ingresso e uscita digitali, 16 ingressi analogici, 4 porte seriali hardware e un oscillatore a cristallo da 16 mhz. È alimentata sia da USB che da un adattatore CC. In genere questa scheda, che ha le stesse caratteristiche dell'Arduno UNO, viene preferita nei progetti più grandi perché dispone di un maggior numero di pin.	

ARDUINO LILYPAD

Lilypad è stato progettato per essere cucito su abiti e tessuti. In questo modo, può essere utilizzato in progetti interessanti che possono essere progettati per essere indossati. È dotato di un microcontrollore Atmega 168V.	

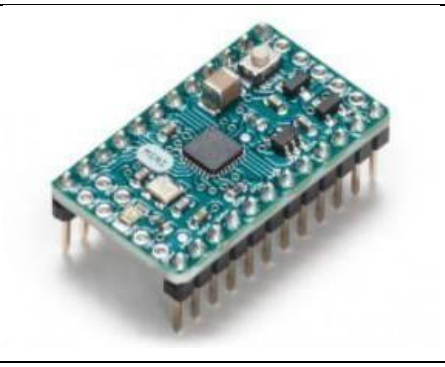
ARDUINO ETHERNET

Dispone di un chip Ethernet e di una porta Ethernet per la realizzazione di progetti connessi a Internet. È presente anche uno slot per scheda SD, che ha il modello Atmega 328 come microcontrollore.	

ARDUINO BLUETOOTH

Su Arduino BT è presente un modulo Bluetooth, ideale per realizzare applicazioni che comunicano con il protocollo Bluetooth. Questo modulo può anche essere utilizzato per programmare Arduino tramite Bluetooth.	

ARDUINO MİNİ

È un modello Arduino progettato per essere utilizzato su una breadboard o integrato in un altro progetto. È dotato di microcontrollore modello Atmega 168 o Atmega 328. È ideale per le applicazioni in cui le dimensioni ridotte sono particolarmente importanti.	

ARDUINO NANO

Si tratta di un modello molto piccolo e progettato per applicazioni su circuito stampato, dotato di microcontrollore Atmega 328 o Atmega 168, regolatore di tensione, chip di conversione da seriale a USB, porta di ingresso di tensione CC e porta mini USB.	

ARDUINO LEONARDO

È una delle schede Arduino che contiene un microcontrollore Atmega 32u4 su Arduino Leonardo e non richiede un chip aggiuntivo per la connessione USB. Con 20 ingressi/uscite digitali e 12 ingressi analogici,il microcontrollore sulla scheda ha un coperchio a montaggio superficiale. Grazie alle sue capacità di connessione USB, Leonardo può essere collegato al computer come mouse o tastiera.	

ARDUINO ESPLORA

Esplora è una scheda Arduino che contiene diversi sensori, a differenza delle altre. Grazie ai sensori presenti sulla scheda, è possibile eseguire molte applicazioni senza bisogno di altre aggiunte e di eccessive conoscenze elettroniche. Esplora è dotata di un potenziometro a scorrimento, un sensore di luce e suono, un sensore di temperatura, un generatore di suoni, un mini joystick analogico a 2 assi, un LED a 3 colori e un accelerometro. Esplora è inoltre dotato di un microcontrollore AVR Atmega 32U4 come Leonarda. È possibile sviluppare applicazioni che fungano da mouse o tastiera quando sono collegate a un computer con la sua connessione micro USB.	

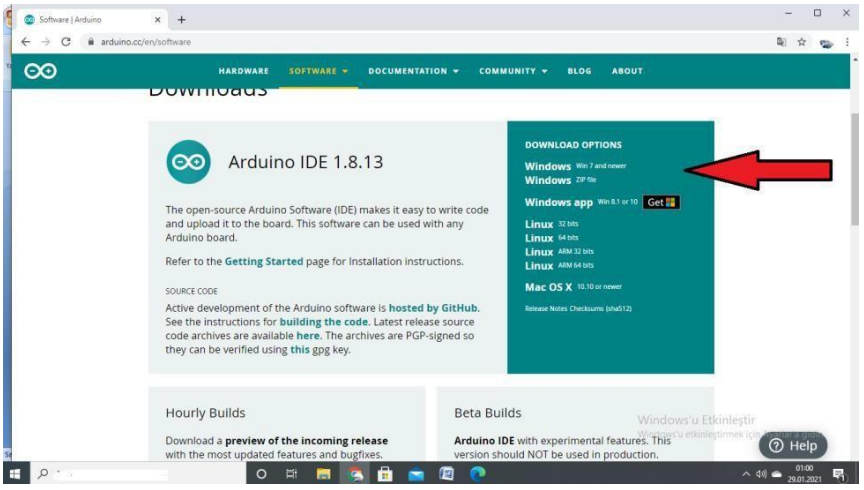
Scaricare l'IDE Arduino

L'IDE (Integrated Development Environment) di Arduino è il luogo in cui si sviluppano i programmi che diranno ad Arduino cosa fare.

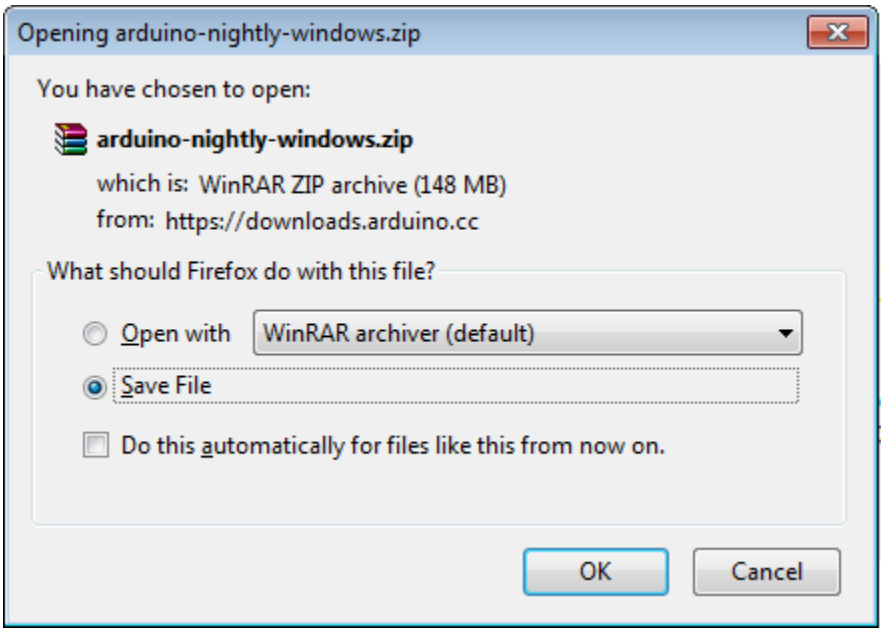
Per installare l'IDE Arduino per Windows, dobbiamo seguire le istruzioni.

È possibile caricare nuovi programmi sul chip principale, l'ATmega328p, via USB utilizzando l'IDE Arduino.

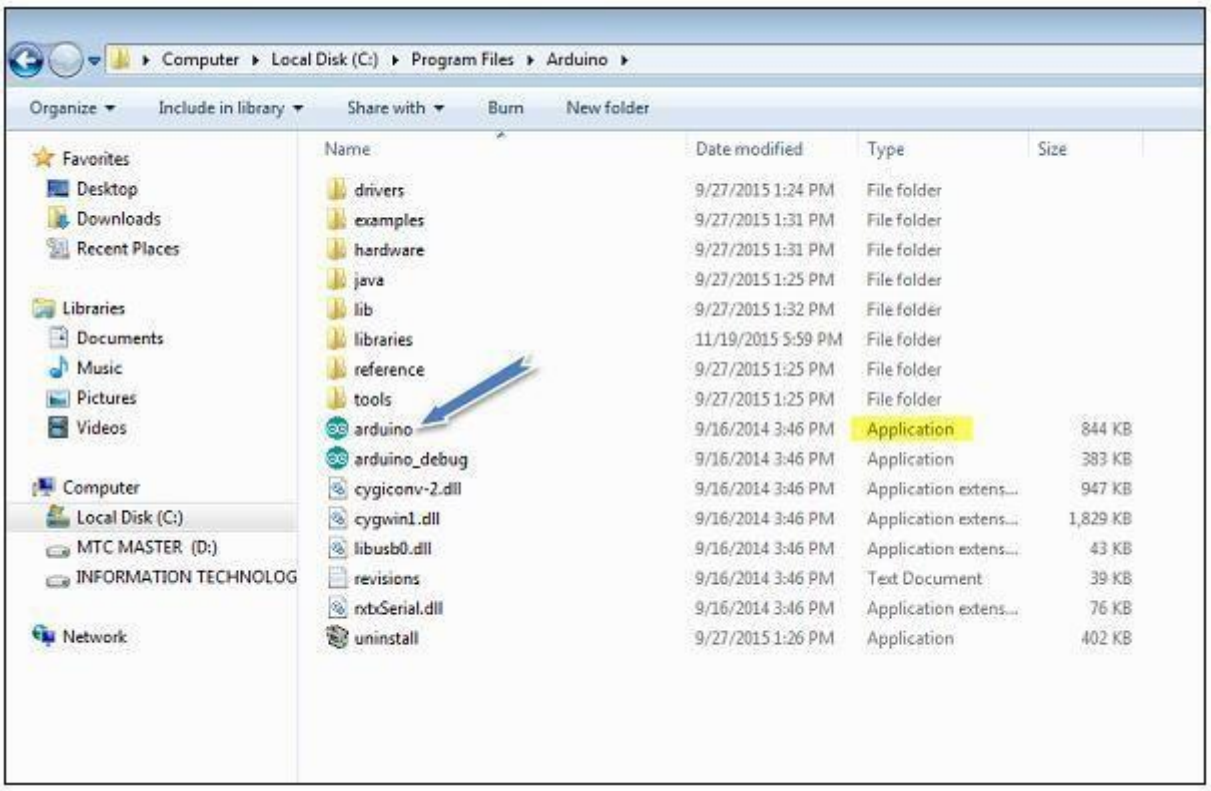
Per scaricare l'IDE Arduino, fare clic sul seguente link:
https://www.arduino.cc/en/Main/Software.



Selezionare il sistema operativo in uso e scaricarlo. Dopo aver scaricato il software Arduino IDE, dobbiamo decomprimere la cartella.



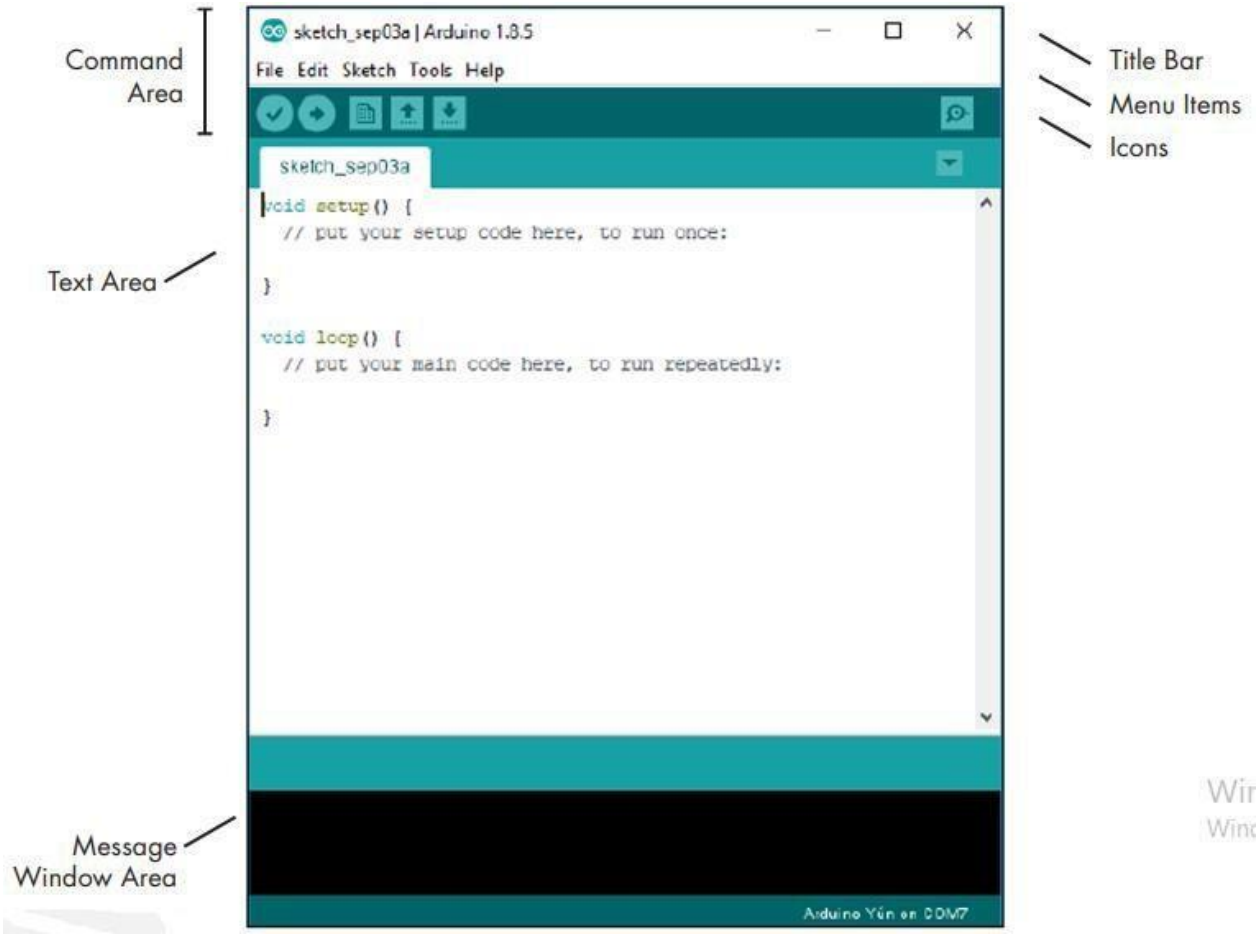
All'interno della cartella si trova l'icona dell'applicazione con un'etichetta infinita (application.exe). Fare doppio clic sull'icona per avviare l'IDE. Quindi, è sufficiente seguire la procedura guidata per installare l'IDE Arduino.



Finestra IDE Arduino per scrivere programmi

Quando si apre per la prima volta l'IDE Arduino, si dovrebbe vedere qualcosa di simile alla figuraseguente.

Come mostrato nella Figura seguente, l'IDE di Arduino assomiglia a un semplice word processor. L'IDE è diviso in tre aree principali: l'area dei comandi, l'area del testo e l'area della finestra dei messaggi.



Voci di menu

Come per qualsiasi elaboratore di testi o editor di testo, è possibile fare clic su una delle voci di menu per visualizzare le varie opzioni.

File: contiene le opzioni per salvare, caricare e stampare gli schizzi, una serie completa di schizzi di esempio da aprire e il sottomenu Preferenze.

Modifica: contiene le consuete funzioni di copia, incolla e ricerca comuni a qualsiasi elaboratore di testi

Schizzo: Contiene la funzione di verifica dello schizzo prima dicaricarlo su una scheda e alcune opzioni di importazione e cartella degli schizzi.

Strumenti: Contiene una serie di funzioni e i comandi per selezionare il tipo di scheda Arduino e la porta USB.

Aiuto: Contiene collegamenti a vari argomenti di interesse e alla versione dell'IDE.

Che cos'è il disegno

Uno schizzo Arduino è un insieme di istruzioni create per svolgere un compito particolare; in altre parole, uno schizzo è un programma.
Lo sketch non è altro che un insieme di istruzioni che Arduino deve eseguire. Gli schizzi creati con l'IDE Arduino vengono salvati come file .pde. Per creare uno sketch, è necessario realizzare le tre parti principali: La dichiarazione delle variabili, la funzione Setup e la funzione Loop principale.

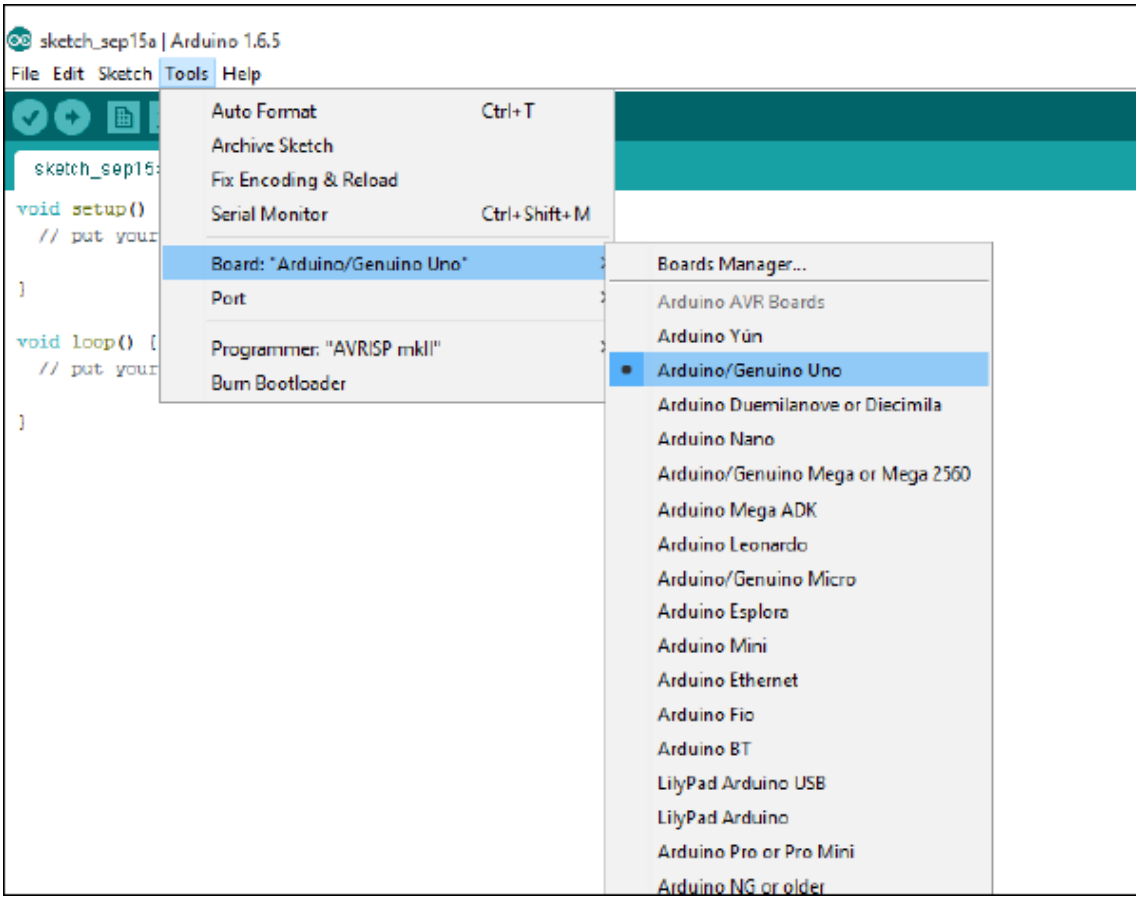
Pulsanti della barra degli strumenti dell'IDE Arduino

Sotto la barra degli strumenti del menu sono presenti sei icone. Passare il mouse su ciascuna icona per visualizzarne il nome. Le icone, da sinistra a destra, sono le seguenti:

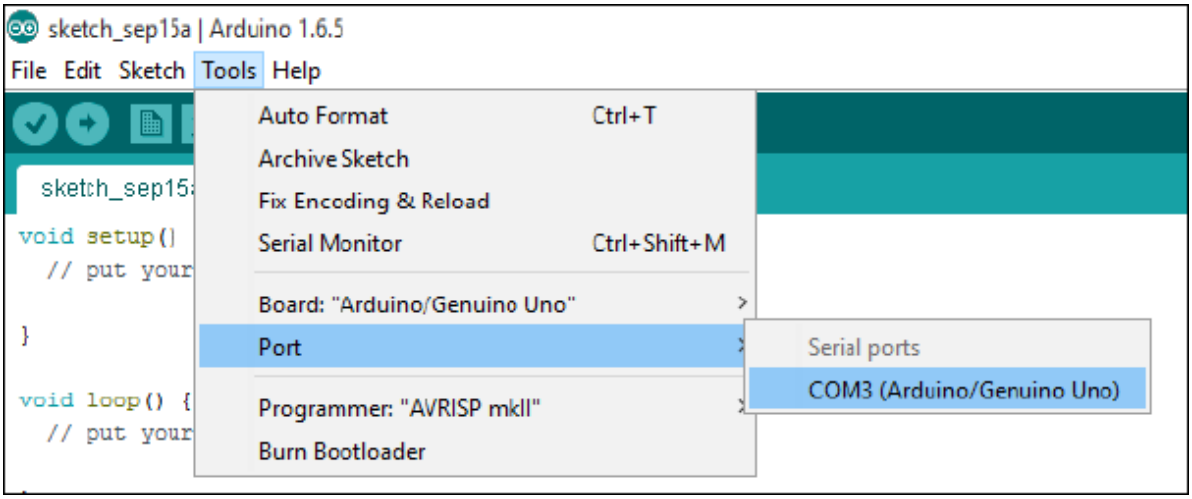
	**Verifica (Compilazione):** Fare clic su questa opzione per verificare che lo schizzo di Arduino sia valido e non contenga errori di programmazione.
	**Nuovo:** Fare clic su questo pulsante per aprire un nuovo schizzo vuoto in una nuova finestra.
	**Apri:** Apri Fare clic su questa opzione per aprire uno schizzo salvato.
	**Salva:** Fare clic su questo pulsante per salvare lo schizzo aperto. Se lo schizzo non ha un nome, verrà richiesto di crearne uno.
	**Carica:** Fare clic su questo pulsante per verificare e caricare lo sketch sulla scheda Arduino.
	**Monitor seriale:** Fare clic su questo pulsante per aprire una nuova finestra da utilizzare per inviare e ricevere dati tra Arduino e l'IDE.

Collegamento di Arduino

Collegare Arduino UNO al computer tramite USB.
Dopo aver collegato Arduino con un cavo USB, è necessario assicurarsi che l'IDE Arduino abbia selezionato la scheda giusta.
Nel nostro caso, stiamo usando Arduino Uno, quindi dobbiamo andare su **Strumenti → Scheda: → Arduino/Genuino Uno.**



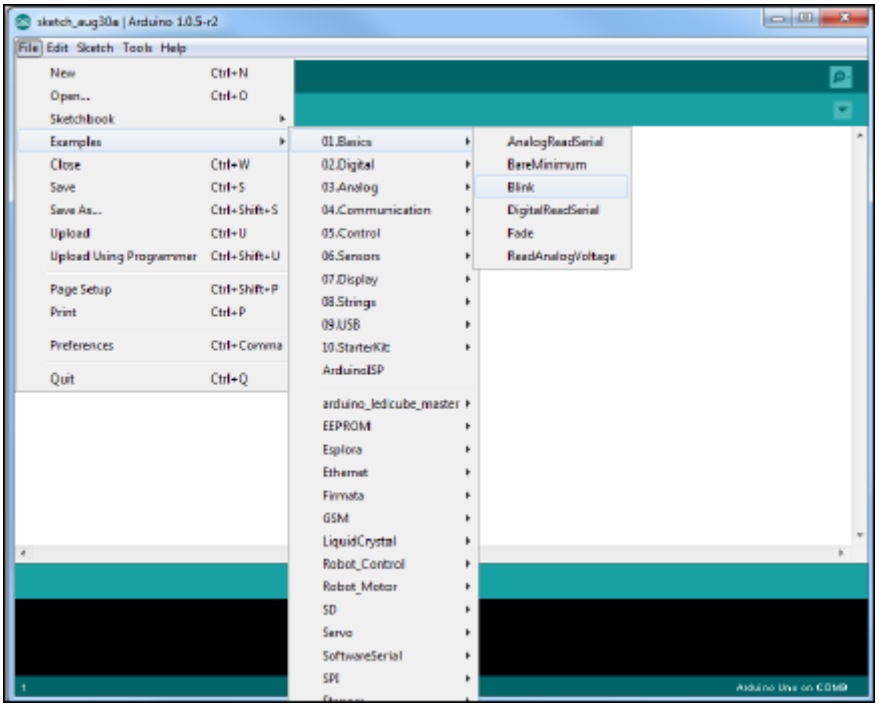
Quindi, si deve selezionare la porta seriale a cui è collegato Arduino. Andare a **Strumenti → Porta** e selezionare la porta giusta.



TÜRKİYE CUMHURİYETİ AVRUPA BİRLİĞİ BAKANLIĞI
REPUBLIC OF TURKEY MINISTRY FOR EU AFFAIRS
TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Co-funded by the Erasmus+ Programme of the European Union

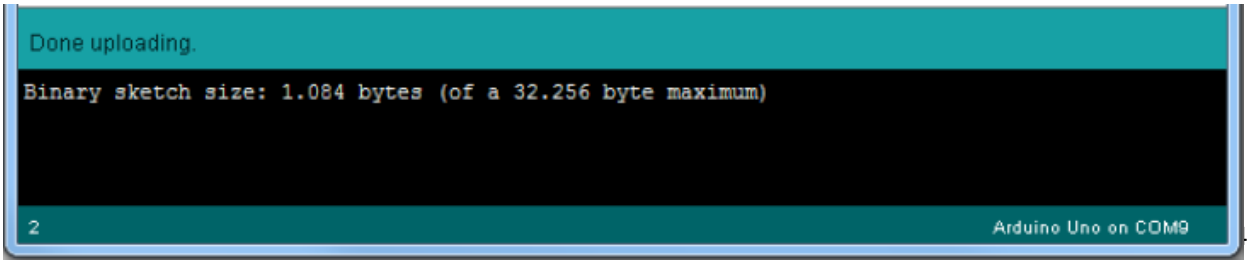
Caricamento di uno schizzo Arduino

Per mostrarvi come caricare il codice sulla vostra scheda Arduino, vi mostreremo un semplice esempio. Si tratta di uno degli esempi più elementari: consiste nel far lampeggiare il LED sulla breadboard o del pin digitale 13 ogni secondo.
1. Aprire l'IDE Arduino.
2. Vai su **File → Esempi → 01.Basics -> Blink**



Per impostazione predefinita, l'IDE Arduino è preconfigurato per Arduino UNO. Fare clic sul pulsante **Carica** e attendere qualche secondo.

Dopo qualche secondo, dovrebbe apparire un messaggio di **caricamento completato**.



Questo codice non fa altro che far lampeggiare il LED di bordo di Arduino UNO (evidenziato in rosso). Si dovrebbe vedere il piccolo LED accendersi per un secondo e spegnersi ripetutamente per un altro secondo.

Controllo di un'uscita e lettura di un ingresso

Una scheda Arduino contiene pin digitali, pin analogici e pin PWM.

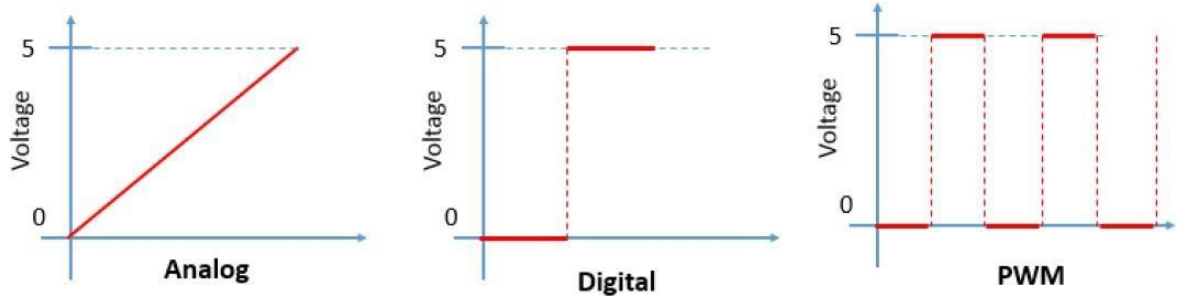
Differenza tra digitale, analogico e PWM

Nei **pin digitali** sono possibili solo due stati: on o off. Questi possono anche essere indicaticome Alto o Basso, 1 o 0 e 5V o 0V.

Ad esempio, se un LED è acceso, il suo stato è Alto o 1 o 5V. Se è spento, lo stato è Basso, ovvero 0 o 0V.

Nei **pin analogici**, gli stati possibili sono illimitati e compresi fra 0 e 1023. Ciò consente di leggere i valori dei sensori. Ad esempio, con un sensore di luce, se è molto scuro si leggerà1023, se è molto luminoso si leggerà 0 Se c'è una luminosità compresa tra scuro e molto luminoso si leggerà un valore compreso tra 0 e 1023.

I pin PWM sono pin digitali, quindi emettono 0 o 5V. Tuttavia, questi pin possono emettere "finti" valori di tensione intermedi tra 0 e 5 V, perché sono in grado di eseguire la "Pulse Width Modulation" (PWM). La PWM consente di "simulare" livelli di potenza variabili facendo oscillare la tensione di uscita di Arduino.



Controllo di un'uscita

Per controllare un'uscita digitale si utilizza la funzione digitalWrite() e tra le parentesi si scrive il pin che si desidera controllare e poi HIGH o LOW.

Per controllare un pin PWM si utilizza la funzione analogWrite() e tra le parentesi si scrive il pin che si vuole controllare e un numero compreso tra 0 e 255.

Lettura di un ingresso

Per leggere un ingresso analogico si utilizza la funzione analogRead() e per un ingresso digitale si utilizza digitalRead().

Nota: il modo migliore per imparare Arduino è fare pratica. Quindi, realizzate molti progetti e iniziate a costruire qualcosa.

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

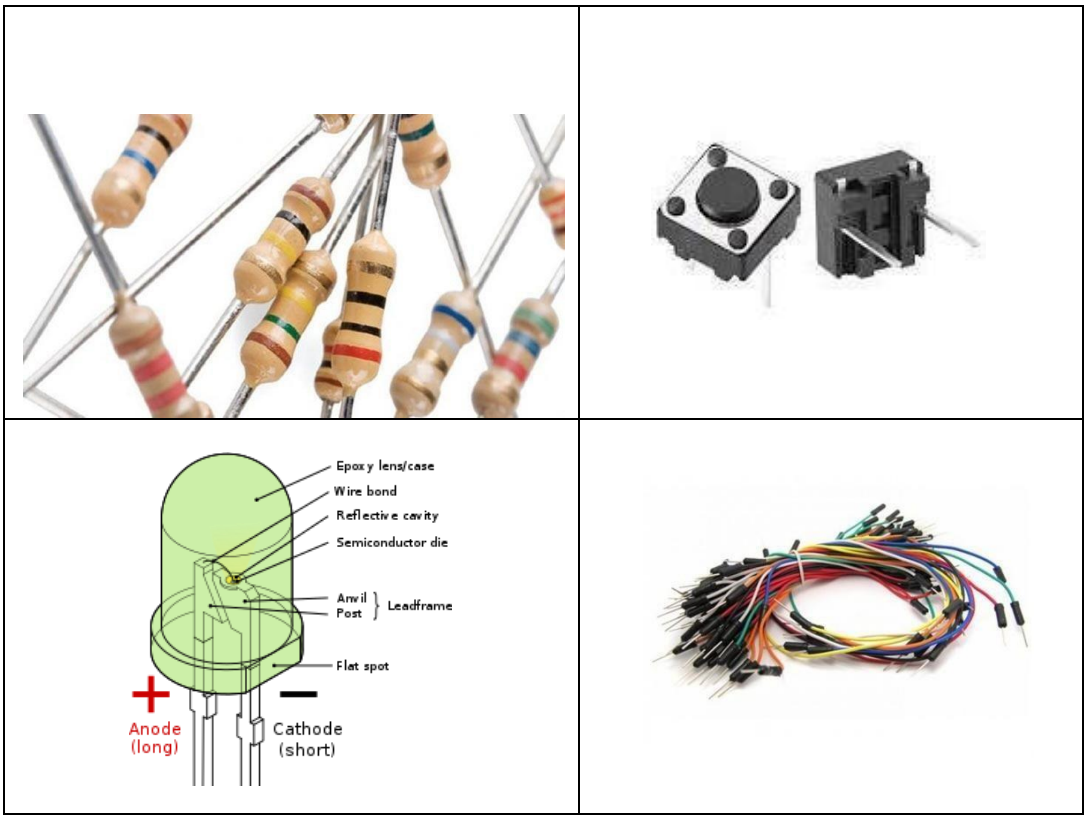
Strategic Partnerships Project for Vocational Education and Training

Titolo Progetto: “Teaching and Learning Arduinos in Vocational Training”

Acronimo Progetto: “ ARDUinVET ”

Progetto N: “2020-1-TR01-KA202-093762”

Arduino Input/Output Module and Kit



Pianificazione dei progetti

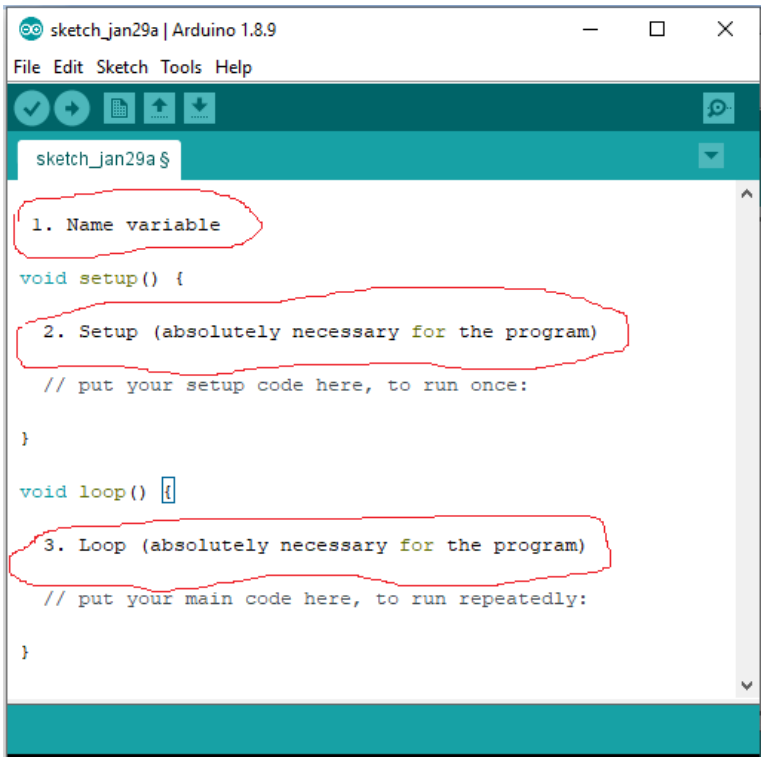
Quando si iniziano i primi progetti, si può essere tentati di scrivere il bozzetto subito dopo aver avuto una nuova idea. Ma prima di iniziare a scrivere, è necessario eseguire alcuni passi preparatori di base. Dopo tutto, la scheda Arduino non legge nel pensiero; ha bisogno di istruzioni precise e, anche se queste istruzioni possono essere eseguite da Arduino, i risultati potrebbero non essere quelli attesi se si trascura anche un piccolo dettaglio.
Sia che stiate creando un progetto che si limita a far lampeggiare una luce o un segnale ferroviario automatizzato, un piano dettagliato è la base del successo. Quando progettate i vostriprogetti Arduino, seguite questi passi fondamentali:

1. Definite il vostro obiettivo. Stabilite cosa volete ottenere.
2. Scrivete il vostro algoritmo. Un algoritmo è un insieme di istruzioni che descrive come realizzare il progetto. L'algoritmo elenca i passaggi necessari per raggiungere l'obiettivo delprogetto.
3. Selezionare l'hardware. Determinare il modo in cui si collegherà ad Arduino.
4. Scrivere lo sketch. Creare il programma iniziale che dice ad Arduino cosa fare.
5. Collegare il tutto. Collegare l'hardware, i circuiti e altri elementi alla scheda Arduino.
6. Test e debug. Funziona? In questa fase si identificano gli errori e se ne individuano le cause, che siano nello sketch, nell'hardware o nell'algoritmo.

Più tempo si dedica alla pianificazione del progetto, più facile sarà la fase di test e debug.

Struttura di base di uno schizzo

Il programma Arduino è chiamato "sketch". Uno sketch può essere suddiviso in tre parti.



1. Variabile nome:
Nella prima parte vengono nominati gli elementi del programma. Questa parte non è assolutamente necessaria

2. Impostazione (assolutamente necessaria per il programma):
La configurazione verrà eseguita una sola volta. Qui si indica al programma, ad esempio,quale pin (slot per i cavi) deve essere un ingresso e quale un'uscita delle schede.
Definito come uscita: Il pin deve emettere una tensione. Ad esempio: Con questo pin siintende accendere un LED.
Definito come ingresso: La scheda deve leggere una tensione. Ad esempio: Un interruttore viene azionato. La scheda lo riconosce perché riceve una tensione sul pindi ingresso.

3. Loop (assolutamente necessario per il programma):
Questa parte del ciclo viene ripetuta continuamente dalla scheda. Assimila lo schizzo dall'inizioalla fine e ricomincia dall'inizio e così via.

Ulteriori regole di sintassi

È necessario prestare attenzione a queste regole durante la scrittura dei programmi Arduino. In caso contrario, il nostro programma fallirà.

; (Punto e virgola):

; (punto e virgola) viene utilizzato per terminare un'istruzione. Se si dimentica di terminare una riga con un punto e virgola, si ottiene un errore del compilatore.

Esempio: int a=13;

{} (parentesi graffe):

Le parentesi graffe sono una parte importante del linguaggio di programmazione Arduino. Vengono utilizzate in diversi costrutti e questo può talvolta confondere i principianti. Una parentesi graffa di apertura "{" deve sempre essere seguita da una parentesi graffa di chiusura "}".

I principali usi delle parentesi graffe: Funzioni, cicli, dichiarazioni condizionali

Esempio:

```
void myfunction(argomento datatype)
{          dichiarazioni (s)          }
```

// (commento a una riga) e /* */ (commento a più righe):

Si tratta di righe del programma utilizzate per informare se stessi o altri sul funzionamento del programma. Vengono ignorate dal compilatore e non vengono esportate nel processore, quindi non occupano spazio sul chip Atmega.
I commenti hanno l'unico scopo di aiutare l'utente a capire (o ricordare) il funzionamento del programma o per informare gli altri sul funzionamento del programma. Esistono due modi diversi percontrassegnare una riga come commento:

Esempio:

```
x = 5;      // Questo è un commento su una sola riga. Qualsiasi cosa dopo gli slash è un commento
```

// alla fine della riga

x = 5; /***Questo** è un commento su più righe.
..... Questa è la fine di un
commentosu più righe. ***/**

#define:

#define consente al programmatore di dare un nome a un valore costante prima che il programma venga compilato. Le costanti definite in arduino non occupano spazio nella memoria del programma sul chip. In generale, la parola chiave const è preferibile per definire le costanti e dovrebbe essere usata al posto di #define.

Esempio:

#define ledPin 3 //Il compilatore sostituirà ogni riferimento a ledPin con il valore 3 in fase di compilazione.

#include:

#include viene utilizzato per includere librerie esterne nello sketch. Ciò consente al programmatore di accedere a un ampio gruppo di librerie C standard (gruppi di funzioni preconfezionate) e anche a librerie scritte appositamente per Arduino.

Esempio:

#includere <servo.h>

Arduino - Tipi di dati

I tipi di dati vengono utilizzati per dichiarare variabili o funzioni di tipo diverso. Il tipo di variabile determina lo spazio che occupa nella memoria e l'interpretazione del modello di bit memorizzato.

Le espressioni utilizzate per memorizzare qualsiasi informazione in memoria e che possono cambiare il valore durante il flusso del programma sono chiamate variabili. Le variabili possono essere numeri, caratteri o espressioni logiche. Il tipo di dati appropriato deve essere selezionato in base al tipo di variabile. A seconda del tipo di dati utilizzati per definire la variabile in memoria, viene allocata una determinata area.

La tabella seguente fornisce tutti i tipi di dati che verranno utilizzati durante la programmazione di Arduino.

Tipo	Contiene
booleano	può contenere sia vero che falso
carbone	Da -128 a 127
byte	Da 0 a 255
carattere senza segno	Da 0 a 255
int	-Da 32.768 a 32.767
int senza segno	Da 0 a 65.535
parola	(come l'int senza segno)
lungo (o lungo int)	da -2.147.483.648 a 2.147.483.647
lungo senza segno	Da 0 a 4.294.967.295
galleggiante	da -3,4028235E+38 a 3,4028235E+38
doppio	(come il galleggiante)

Arduino - Variabili e costanti

Durante la definizione della variabile, è necessario determinare il nome della variabile, il valore della variabile e il tipo di dati appropriato per la variabile.

La definizione è fatta come si vede nell'esempio seguente:

intLED =12;

Qui: int=tipo di **dato** LED=nome **variabile** 12=valore **della variabile**

Le variabili, che Arduino utilizza, hanno una proprietà chiamata scope. Un ambito è una regione del programma e ci sono tre posti in cui le variabili possono essere dichiarate. Essi sono:

- All'interno di una funzione o di un blocco, sono chiamate variabili locali.
- Nella definizione dei parametri delle funzioni, che si chiamano parametri formali.
- Al di fuori di tutte le funzioni, le cosiddette variabili globali.

Una costante è un *qualificatore di* variabile che modifica il comportamento della variabile, rendendola "*di sola lettura*". Ciò significa che la variabile può essere utilizzata come qualsiasi altra variabile del suo tipo, ma il suo valore non può essere modificato. Si otterrà un errore del compilatore se si tenta di assegnare un valore a una variabile const.

Le costanti definite con la parola chiave const obbediscono alle regole di scoping delle variabili che regolano le altre variabili. Questo aspetto e le insidie dell'uso di #define rendono la parola chiave const un metodo superiore per la definizione delle costanti ed è preferibile all'uso di #define.

Esempio:

```
const float pi = 3,14;
```

Note:

#define **o** const: È possibile utilizzare const o #define per creare costanti numeriche o di stringa. Per gli array, è necessario usare const. In generale, const è preferibile a #define per definire le costanti.

Arduino - Operatori

Un operatore è un simbolo che indica al compilatore di eseguire specifiche funzioni matematiche o logiche. In Arduino si possono usare i seguenti tipi di operatori.

Operatori aritmetici:

Supponiamo che la variabile A contenga 10 e la variabile B contenga 20, allora -

Nome dell'operatore	Operatore semplice	Esempio
operatore di assegnazione	=	A = B
aggiunta	+	A + B darà 30
sottrazione	-	A - B darà -10
moltiplicazione	*	A * B darà 200
divisione	/	B / A darà 2
modulo	%	B % A darà 0

Operatori di confronto:

Supponiamo che la variabile A contenga 10 e la variabile B contenga 20, allora -

Nome dell'operatore	Simbolo dell'operatore	Esempio
pari a	==	(A == B) non è vero
non uguale a	!=	(A != B) è vero
meno di	<	(A < B) è vero
maggiore di	>	(A > B) non è vero
inferiore o uguale a	<=	(A <= B) è vero
maggiore o uguale a	>=	(A >= B) non è vero

Operatori booleani

Supponiamo che la variabile A contenga 10 e la variabile B contenga 20, allora -

Nome dell'operatore	Operatore semplice	Esempio
e	&&	(A & B) è vero
o	||	(A || B) è vero
non	!	!(A && B) è falso

Operatori bitwise

Supponiamo che la variabile A contenga 60 e la variabile B contenga 13, quindi -

Nome dell'operatore	Operatore semplice	Esempio
e	&	(A & B) darà 12 che è 0000 1100
o	|	(A | B) darà 61 che è 0011 1101
xor	^	(A ^ B) darà 49 che è 0011 0001
non	~	(~A) darà -60 che è 1100 0011
spostamento a sinistra	<<	A << 2 darà 240 che è 1111 0000
spostamento a destra	>>	A >> 2 darà 15 che è 0000 1111

Arduino - Funzioni di I/O (comandi)

Le funzioni consentono di strutturare i programmi per eseguire compiti individuali. Il caso tipico per la creazione di una funzione è quando è necessario eseguire la stessa azione più volte in un programma. Le funzioni diventeranno più chiare quando mostreremo esempi concreti di programmi in circuiti e programmi Arduino. Per aiutare a spiegare le varie funzioni di comando,le abbiamo suddivise in comandi separati

I pin della scheda Arduino possono essere configurati come ingressi o uscite. Di seguito spiegheremo il funzionamento dei pin in queste modalità. È importante notare che la maggior parte dei pin analogici di Arduino può essere configurata e utilizzata esattamente come i pin digitali.

Funzione pinMode():

La funzione pinMode() serve a configurare un pin specifico in modo che si comporti come ingresso o come uscita. È possibile abilitare le resistenze di pull-up interne con la modalità INPUT_PULLUP.

Sintassi: pinMode(pin, mode)

```
Void setup () {
 pinMode (pin , mode);
```

}

Parametri:
pin: il numero del pin di Arduino da impostare per la
modalità: INPUT, OUTPUT o INPUT_PULLUP.

Esempi:
pinMode(13, OUTPUT); //imposta il pin digitale 13 come uscita
pinMode(5, INPUT); //imposta il pin digitale 5 come ingresso

Funzione digitalWrite():

La funzione digitalWrite() serve a scrivere un valore HIGH o LOW su un pin digitale. Se il pin è stato configurato come OUTPUT con pinMode(), la sua tensione sarà impostata sul valore corrispondente: 5V per HIGH, 0V (massa) per LOW. Se il pin è configurato come INGRESSO, digitalWrite() abilita (HIGH) o disabilita (LOW) il pullup interno sul pin di ingresso. Si consiglia di impostare pinMode() su INPUT_PULLUP per abilitare la resistenza di pull-up interna.

Se non si imposta pinMode() su OUTPUT e si collega un LED a un pin, quando si richiama digitalWrite(HIGH), il LED potrebbe apparire poco luminoso. Senza impostare esplicitamente pinMode(), digitalWrite() avrà attivato la resistenza di pull-up interna, che agisce come una grande resistenza di limitazione della corrente.

Sintassi:
digitalWrite (pin ,valore);

pin: il numero del pin di cui si desidera impostare il

valore: HIGH (1) o LOW (0).

Esempio:

digitalWrite(LED, HIGH); //accendere il led
digitalWrite(LED, LOW); //spegnere il led

Funzione digitalRead():

La funzione digitalRead() legge il valore di un pin digitale specificato, ALTO o BASSO.

Sintassi: digitalRead(pin)
pin: il numero del pin di Arduino che si desidera leggere.

Esempi:

val = digitalRead(inPin); //lettura del pin d'ingresso

Nota: i pin di ingresso analogici possono essere utilizzati come pin digitali, denominati A0, A1, ecc.

Funzione delay():

La funzione dalay() mette in pausa il programma per il tempo (in millisecondi) specificato come parametro. (In un secondo ci sono 1000 millisecondi).

Sintassi: delay(ms):
ms: il numero di millisecondi di pausa. Tipi di dati consentiti: unsigned long.

Esempi:

```
delay(1000);          //attende per 1 secondo
delay(2000);          //attender per 2 secondi
```

Funzione analogWrite():

La funzione analogWrite() scrive un valore analogico (onda PWM) su un pin. Può essere utilizzata per illuminare un LED a luminosità variabile o per azionare un motore a varie velocità. Dopo una chiamata a analogWrite(), il pin genera un'onda rettangolare costante del duty cycle specificato fino alla chiamata successiva.
a analogWrite() (o una chiamata a digitalRead() o digitalWrite()) sullo stesso pin.

Esempi:
Imposta l'uscita al LED proporzionale al valore letto dal potenziometro.

```
val = analogRead(analogPin);      //lettura del pin d'ingresso
analogWrite(ledPin, val/4);       //i valori di lettura vanno da 0 a 1023, i valori di scritta da 0 a 255
```

Funzione analogRead( )

Arduino è in grado di rilevare se c'è una tensione applicata a uno dei suoi pin e di segnalarlo attraverso la funzione digitalRead(). C'è una differenza tra un sensore on/off (che rileva la presenza di un oggetto) e un sensore analogico, il cui valore cambia continuamente. Per leggere questo tipo di sensore, abbiamo bisogno di un tipo diverso di pin.

Nella parte inferiore destra della scheda Arduino si trovano sei pin contrassegnati dalla dicitura "Analog In". Questi pin speciali non solo indicano se vi è applicata una tensione, ma anche ilsuo valore. Utilizzando la funzione analogRead(), possiamo leggere la tensione applicata a uno dei pin.

Questa funzione restituisce un numero compreso tra 0 e 1023, che rappresenta tensioni tra 0 e 5 volt. Ad esempio, se al pin numero 0 è applicata una tensione di 2,5 V, analogRead(0) restituisce 512.

Sintassi: analogRead(pin);

pin: numero del pin di ingresso analogico da leggere, da 0 a 5.

Esempio:

```
val = analogRead(analogPin);      //leggere il pin di
ingresso Serial.println(val);     // valore di debug
```

Istruzione if:

L'istruzione if controlla una condizione ed esegue l'istruzione o l'insieme di istruzioni successivese la condizione è "vera".

Sintassi:
se (condizione) {
//dichiarazione/i
}

condizione: un'espressione booleana (cioè, può essere vera o falsa).

Esempi: Le parentesi possono essere omesse dopo un'istruzione if. In questo caso, la riga successiva (definita dal punto e virgola) diventa l'unica dichiarazione condizionale.

```
if (x > 120) digitalWrite(LEDpin, HIGH);

if (x > 120)
digitalWrite(LEDpin, HIGH);

if (x > 120) {digitalWrite(LEDpin, HIGH);}

if (x > 120) {
        digitalWrite(LEDpin1, HIGH);
        digitalWrite(LEDpin2, HIGH);
}
```

Comando if-else:

L'istruzione if...else consente un maggiore controllo sul flusso del codice rispetto all'istruzione ifdi base, permettendo di raggruppare più test. Una clausola else (se esiste) viene eseguita se la condizione nell'istruzione if risulta falsa. L'else può procedere con un altro test if, in modo da poter eseguire contemporaneamente più test che si escludono a vicenda.

Ogni test procede al successivo fino a quando non viene incontrato un test vero. Quando viene trovato un test vero, viene eseguito il blocco di codice associato e il programma salta alla riga successiva all'intero costrutto if/else. Se nessun test risulta vero, viene eseguito il blocco else predefinito, se presente, e viene impostato il comportamento predefinito. È consentito un numero illimitato di rami else if.

Sintassi:

```
if (la condizione è VERA) {
        // fai la cosa A
}
else
{
        //altrimenti, fai la cosa B
}
```

```
if (condizione 1) {
        // fai la cosa A
} else if (condizione 2) {
        // fai la cosa B
} else {
        // fai la cosa C
}
```

Esempi: (Di seguito è riportato un estratto di un codice per un sistema di sensori di temperatura.)

```
if (temperatura >= 70) {
        // Pericolo! Spegnere il sistema.
} else if (temperatura >= 60) { // 60 <= temperatura < 70
        // Attenzione! Richiesta attenzione dell'utente.
} else { // temperatura < 60
        // Sicuro! Continuare con i compiti attuali
}
```

Comando for:

L'istruzione for viene utilizzata per ripetere un blocco di istruzioni racchiuse tra parentesi graffe. Di solito viene utilizzato un contatore di incremento per incrementare e terminare il ciclo.

L'istruzione for è utile per qualsiasi operazione ripetitiva e viene spesso utilizzata in combinazione con gli array per operare su collezioni di dati/perni.

Sintassi:

```
for (inizializzazione; condizione; incremento) {
  // dichiarazione/i;
}
```

Parametri:

inizializzazione: avviene per prima ed esattamente una volta.

condizione: ogni volta che si attraversa il ciclo, la condizione viene testata; se è vera, l'istruzione si blocca e l'incremento viene eseguito, quindi la condizione viene testata di nuovo. Quando la condizione diventa falsa, il ciclo termina.

Incremento: eseguito ogni volta che la condizione è vera.

Esempi:

```
for (int i = 0; i <= 255; i++) {
        analogWrite(PWMpin, i);
}
for (int x = 2; x < 100; x = x * 1.5) {
        println(x);
}
for (int i = 0; i > -1; i = i + x) {
        analogWrite(PWMpin, i);
}
```

Comando switch...case

Come le istruzioni **if**, switch/ case controllano il flusso dei programmi consentendo ai programmatori di specificare codice diverso da eseguire in varie condizioni. In particolare, un'istruzione switch confronta il valore di una variabile con i valori specificati nelle istruzioni case. Quando viene trovata un'istruzione case il cui valore corrisponde a quello della variabile, viene eseguito il codice contenuto in quell'istruzione case.

La parola chiave **break** esce dall'istruzione switch ed è tipicamente usata alla fine di ogni caso.Senza un'istruzione di interruzione, l'istruzione switch continuerà a eseguire le espressioni successive ("falling-through") fino a quando non verrà effettuata un'interruzione o verrà raggiunta la fine dell'istruzione switch.

Sintassi

```
switch (var) {
        case etichetta1:
                // istruzioni
                break;
        case etichetta2:
                // istruzioni
                break;
        default:
                // istruzioni
                 break;
}
```

Codice di esempio

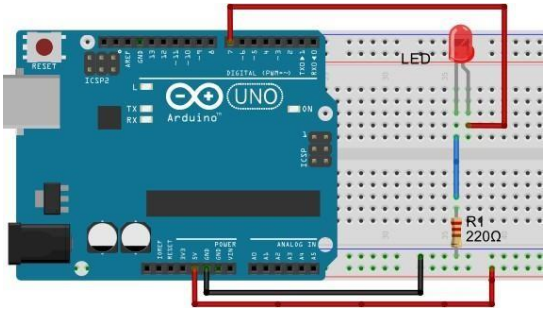
```
switch (var) {
        case 1:
                //fai qualcosa
quando var è uguale a 1
                break;
        case 2:
                //fai qualcosa
quando var è uguale a 2
                break;
        default:
                // se nient'altro
corrisponde, fai il default
                // default è opzionale
        break;
}
```

var: una variabile il cui valore deve essere confrontato con i vari casi. Tipi di dati ammessi: int, char.
etichetta1, etichetta2: costanti. Tipi di dati ammessi: int, char.

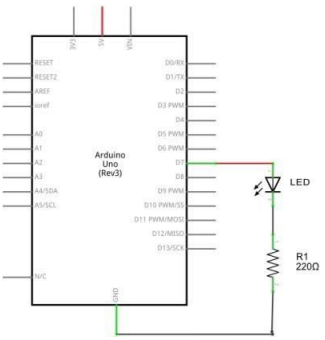
Nota:

I circuiti Arduino sono mostrati in duemodi:

1. Vista su scheda elettronica
2. Vista schematica



1) Vista su scheda elettronica



2) Vista Schematica

Utilizzeremo la vista Breadboard che è più utilizzata.

Basta parlare! Facciamo qualcosa!

Circuiti del kit di moduli LED e pulsanti Arduino

La maggior parte dei pin di Arduino può essere configurata come ingresso o uscita. Il Kit modulo pulsanti e LED è il primo KIT di formazione di ARDUinVET per far apprendere agli studenti i sistemi di I/O di Arduino. Pertanto, può essere denominato KIT di formazione I/O Arduino. In questo kit di formazione, come mostrato di seguito, 6 pulsanti sono collegati ad Arduino come ingressi. Come uscite sono collegati un cicalino, un connettore a 2 pin e 6 LED.

Poiché gli studenti possono utilizzare questo kit di formazione per imparare i sistemi di I/O di Arduino, questo kit di formazione può rendere più facile il test dei circuiti Arduino. Inoltre, possono utilizzare una breadboard per testare i circuiti Arduino e gli esperimenti.

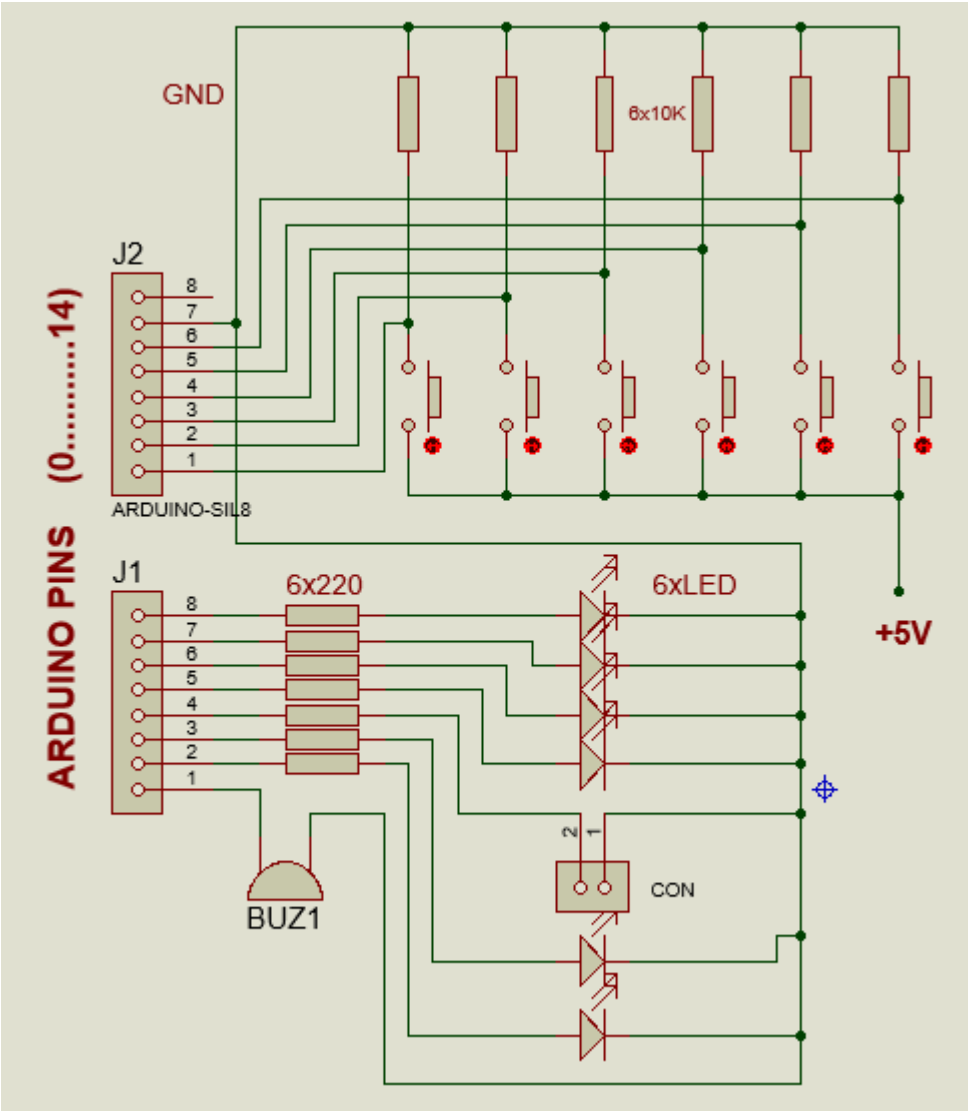


Figura: Circuito aperto del kit di pulsanti e moduli LED

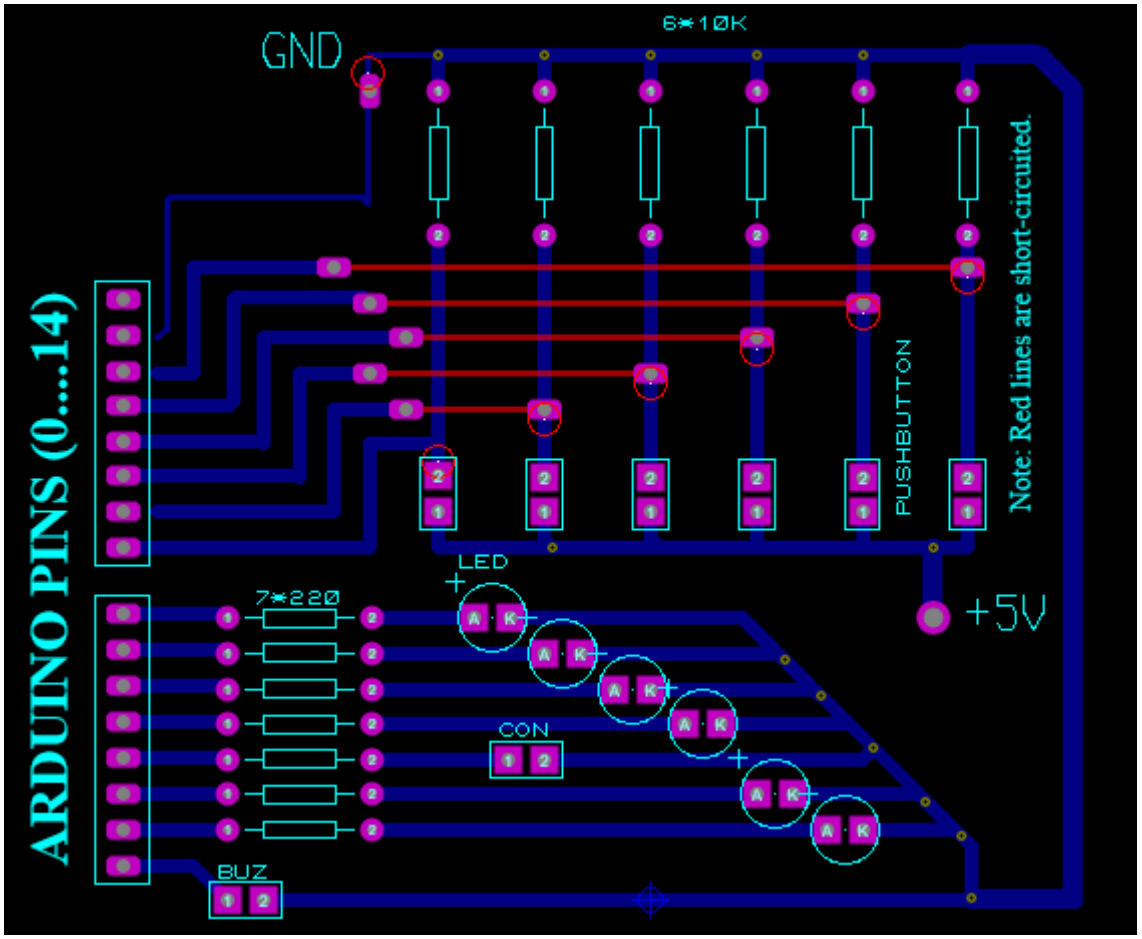


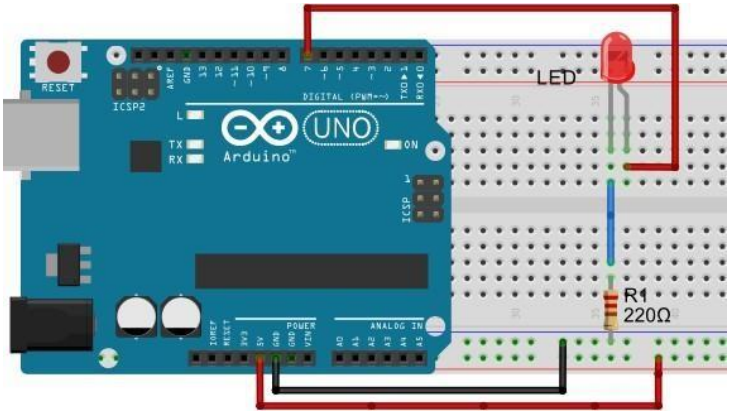
Figura: Schema PCB del kit di moduli pulsanti e LED

Esempi di circuiti e programmi Arduino

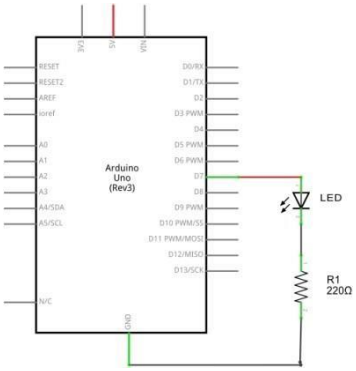
Circuito 1:

Titolo del circuito: LED lampeggiante Programma

Descrizione del circuito: Un LED è collegato al 13° pin di Arduino. Il LED lampeggia continuamente con un intervallo di 1 secondo.



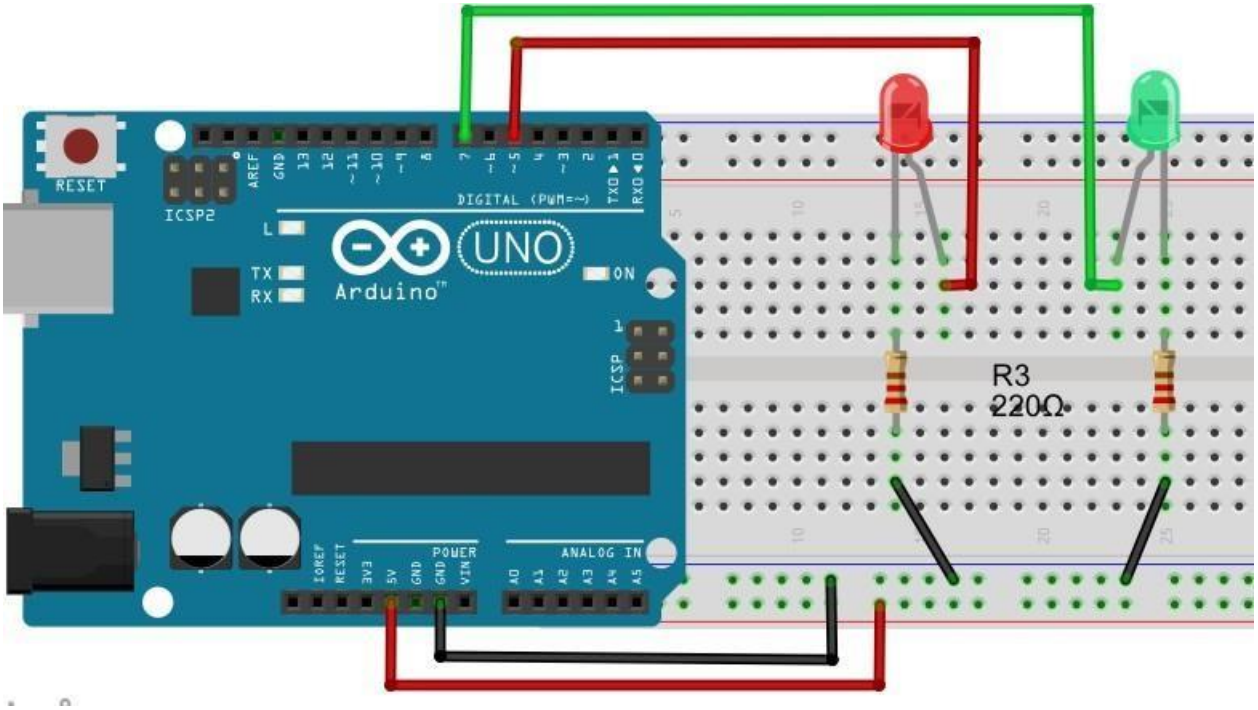
1) Vista su scheda elettronica



2) Vista schematica

```
/*Programma di lampeggiamento del LED,
accensione e spegnimento di un LED*/

int led = 7;          //viene dichiarata la variabile intera led
void setup() {                  // il metodo setup() viene eseguito una sola volta
pinMode(led, OUTPUT);     // il PIN del led è dichiarato come uscita digitale
}

void loop() {                   // il metodo loop() viene ripetuto
digitalWrite(led, HIGH);     // accensione del led
delay(1000);                    // arresta il programma per 1000 millisecondi
digitalWrite(led, LOW);       // spegnimento del led
delay(1000);                    // arresta il programma per 1000 millisecondi

}
```

Circuito 44
:
Titolo del circuito: Flip-Flop, 2 LED lampeggianti Programma
Descrizione del circuito: 2 LED lampeggiano in sequenza con intervalli di 2 secondi.

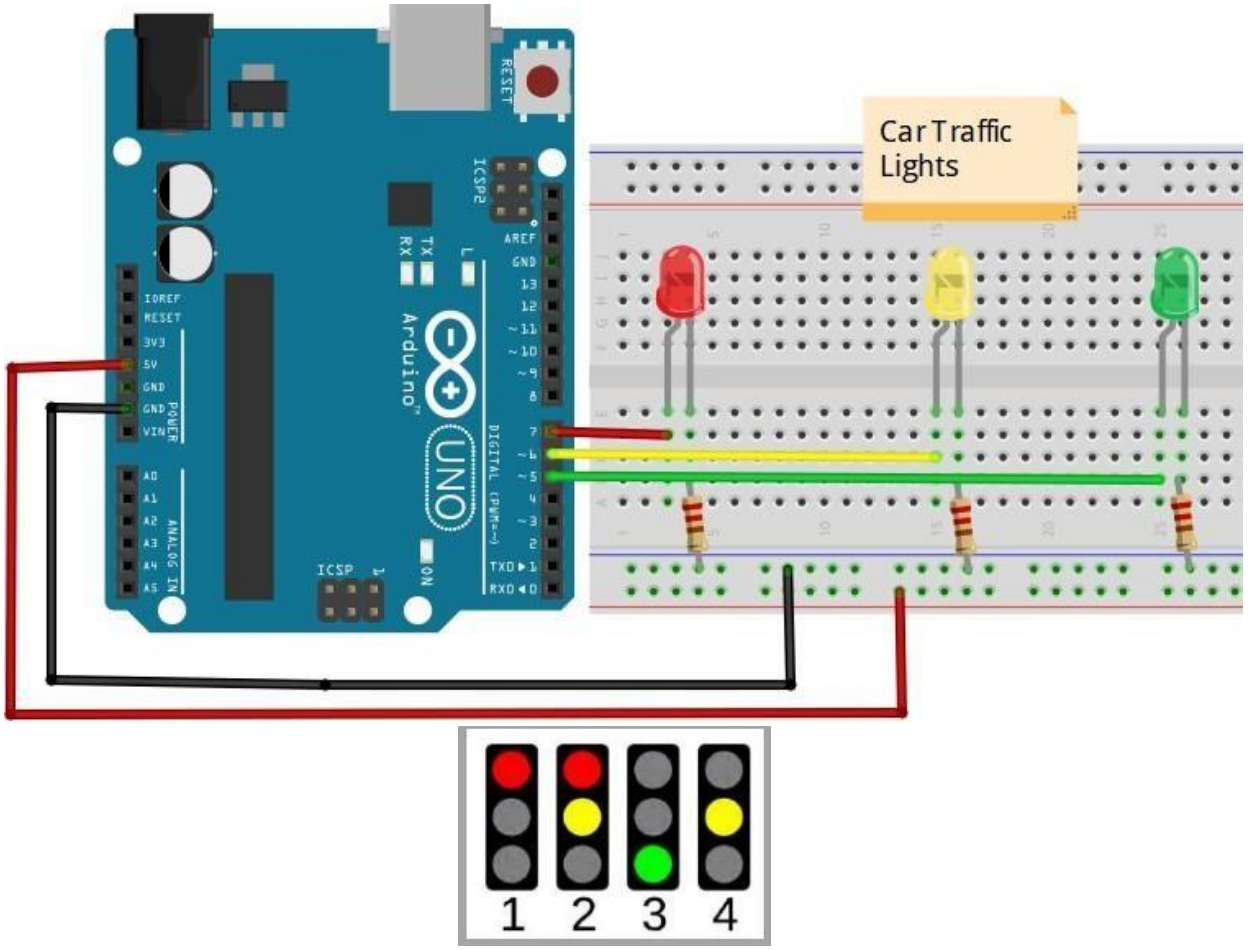



```
/*Flip Flop                */

int greenLED=5;                          // Pin a cui è collegato il LED verde
int redLED=7;                            // Pin a cui è collegato il LED rosso

void setup() {
pinMode(greenLED, OUTPUT);               // il pin del LED verde è inizializzato come OUTPUT
pinMode(redLED, OUTPUT);                 // il pin del LED rosso è inizializzato come OUTPUT
}

void loop(){
digitalWrite(greenLED, HIGH);        // accensione del LED verde
digitalWrite(redLED, LOW);           // spegnimento del LED rosso
pausa(2000);

digitalWrite(greenLED, LOW);                 // spegne il LED verde
digitalWrite(redLED, HIGH);                  // accende il LED rosso
pausa(2000);
}
```

Circuito 45 :

Titolo del circuito: Semafori

Descrizione del circuito: In questo progetto costruiremo un sistema di semafori. Ci sono 3 LED di colore diverso (verde, giallo e rosso).



/*Luci deltraffico*/

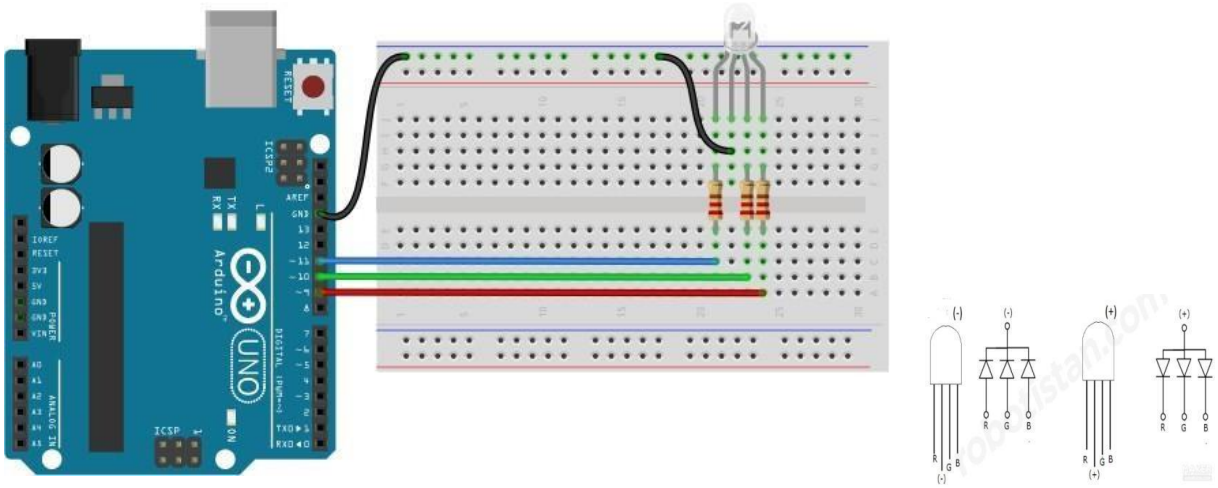
```
int redLED = 7;
int yellowLED =6;
int greenLED = 5;

void setup() {                          // qui, stiamo inizializzando i nostri pin come uscite
pinMode(redLED, OUTPUT);
pinMode(yellowLED, OUTPUT);
pinMode(greenLED, OUTPUT);
}

void loop() {
digitalWrite(redLED, HIGH);   // redLED è acceso per 9 secondi
```

```
digitalWrite(yellowLED, LOW);
digitalWrite(greenLED, LOW);
delay(9000);

digitalWrite(redLED, HIGH);          // redLED e yellowLED sono accesi per 2secondi
digitalWrite(yellowLED, HIGH);
digitalWrite(greenLED, LOW);
delay(2000);

digitalWrite(redLED, LOW);           // greenLED è acceso per 9 secondi
digitalWrite(yellowLED,  LOW);
digitalWrite(greenLED,  HIGH);
delay(9000);

digitalWrite(redLED, LOW);           // Anche in questo caso, il LED giallo è acceso per 2 secondi.
digitalWrite(yellowLED, HIGH);
digitalWrite(greenLED, LOW);
delay(2000);

/* Il ciclo ricomincia        */
 }
```

Circuito 4:

Titolo del circuito: APPLICAZIONE LED RGB

Spiegazione del circuito: Il LED RGB è costituito da 3 LED, collegati in comune, posti in un unico contenitore. È possibile controllare l'intensità luminosa dei tre colori in modo digitale. Inoltre, è possibile ottenere i colori desiderati utilizzando la tecnica PWM.



/* Ogni colore del LED RGB lampeggia a intervalli di 1 secondo. Se vogliamo visualizzare una luce bianca, dobbiamo accendere tutti i LED..*/

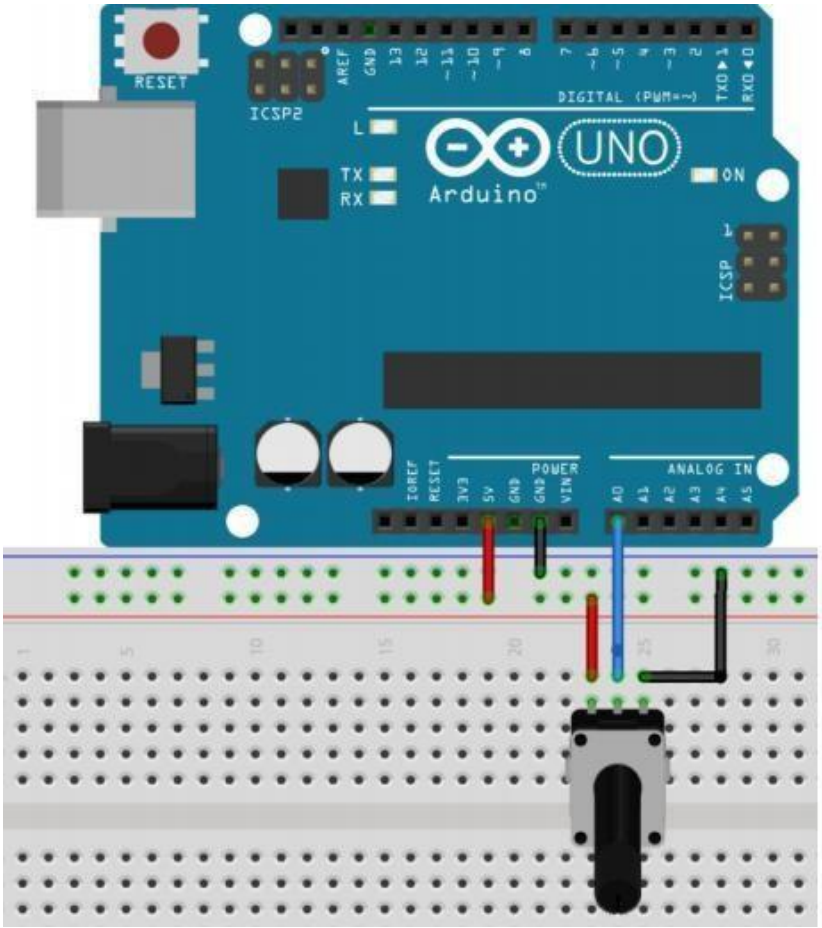
```
const int BlueLed=11;        // colleghiamo il led blu al pin-11
const int GreenLed=10;       // colleghiamo il led verde al pin-10
const int RedLed=9;          // colleghiamo il led rosso al pin-11

        // Assegniamo i pin a cui sono collegati i LED come uscite.
void setup() {
pinMode(BlueLed,OUTPUT);
pinMode(GreenLed,OUTPUT);
pinMode(RedLed,OUTPUT);     }

        // Il ciclo inizia qui.

void loop() {
digitalWrite(BlueLed, LOW);        // RedLed è ON.
digitalWrite(GreenLed, LOW);
digitalWrite(RedLed, HIGH);
delay(1000);

 digitalWrite(BlueLed, LOW );      // GreenLed èacceso.
digitalWrite(GreenLed, HIGH);
 digitalWrite(RedLed, LOW );
delay(1000);

 digitalWrite(BlueLed, HIGH);          // BlueLed è acceso.
digitalWrite(GreenLed, LOW);
digitalWrite(RedLed, LOW);
delay(1000);

        // Visualizziamo il colore bianco attivando tutti i led.
digitalWrite(BlueLed, HIGH);
digitalWrite(GreenLed, HIGH);
digitalWrite(RedLed, HIGH);
delay(1000);
}
```

Circuito 5:

Titolo del circuito: Lettura della tensione analogica, lettura del valore del potenziometro

Spiegazione del circuito: Qui impariamo a leggere un ingresso analogico sul pin analogico-0. L'ingresso viene convertito da analogRead() in tensione e stampato sul monitor seriale dell'IDE Arduino.

Potenziometro: Il potenziometro (o vaso) è un semplice trasduttore elettromeccanico. Converte il movimento rotatorio o lineare dell'operatore in ingresso in una variazione di resistenza. Questa variazione viene (o può essere) utilizzata per controllare qualsiasi cosa, dal volume di un sistema hi-fi alla direzione di un'enorme nave container.	



/* ReadAnalogVoltage: legge un ingresso analogico sul pin 0, lo converte in tensione e stampa il risultato sul monitor seriale.
La rappresentazione grafica è disponibile con il plotter seriale (menu Strumenti → Plotter seriale).Collegare il pin centrale di un potenziometro al pin A0 e i pin esterni a +5V e a massa. */

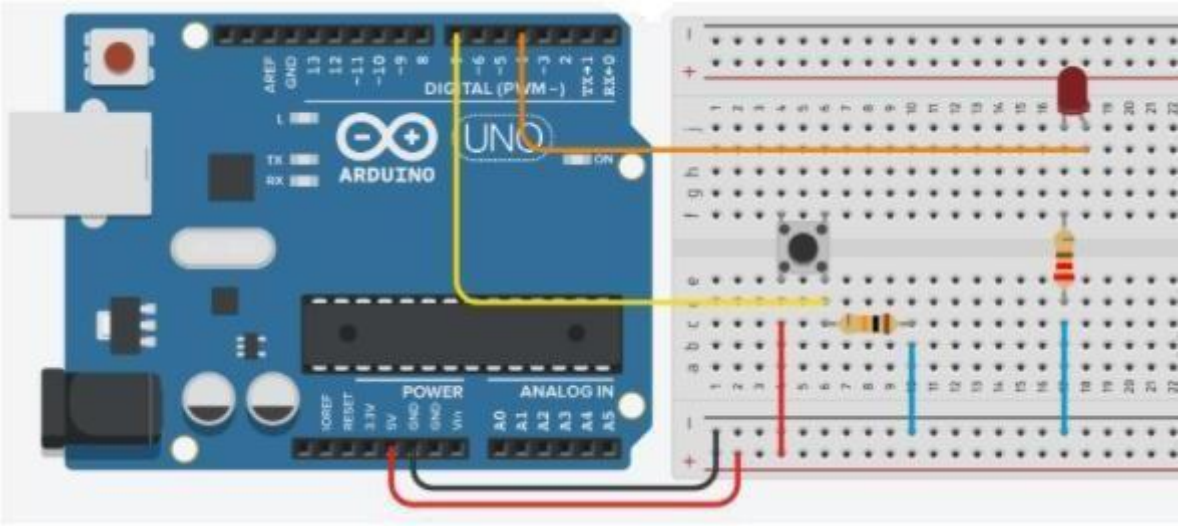
// la routine di impostazione viene eseguita una volta quando si preme il tasto reset:

```
void setup()        {
          // inizializzare la comunicazione seriale a 9600 bit al secondo:
  Serial.begin(9600); }
          // la routine del ciclo si ripete all'infinito:
void loop() {
           // leggere l'ingresso sul pin analogico 0:
  int sensorValue = analogRead(A0);
          // stampare il valore letto:
  Serial.println(sensorValue);
  delay(1000);        // ritardo tra le letture per garantire la stabilità
}
```

Circuito 6:

Titolo del circuito: Uso dei pulsanti su Arduino

Spiegazione del circuito: Quando si preme un pulsante collegato al pin 7, il pulsante accende e spegne un (LED), collegato al pin digitale 4,



/*Pulsante Accende e spegne un diodo luminoso (LED) collegato al pin digitale 4, quando si preme un pulsante collegato al pin 7. */

```
vuoto setup()
{
  pinMode(4, OUTPUT); // il pin-4 è un'uscita
  pinMode(7, INPUT);     // Il pin7 è un ingresso. 7
}
vuoto loop()
{
   if (digitalRead(7) == HIGH)              // Se il pin7 è alto,
    digitalWrite(4, HIGH);                  // Il led è acceso,
    if (digitalRead(7) == LOW)              // Se il pin7 è basso,
     digitalWrite(4, LOW);                  // Il led è spento.
}
```

NOTA: il comando IF-THEN può essere utilizzato anche per collegare i pulsanti ai pin di ingresso di Arduino. Il collegamento può essere effettuato in due modi diversi. Collegamento Pull_Up e Pull-Down.

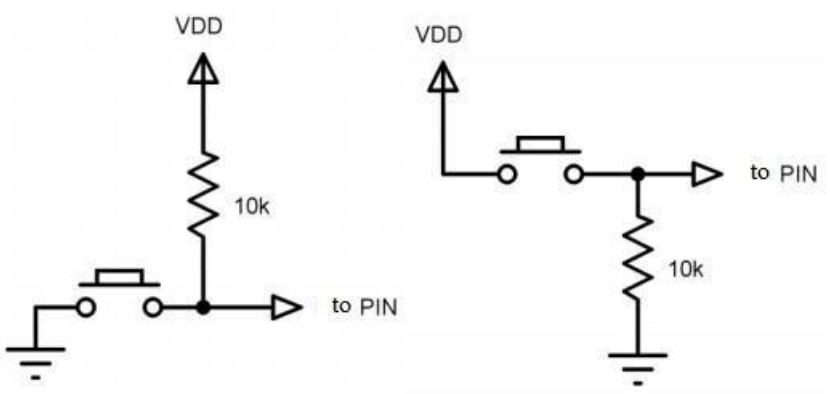
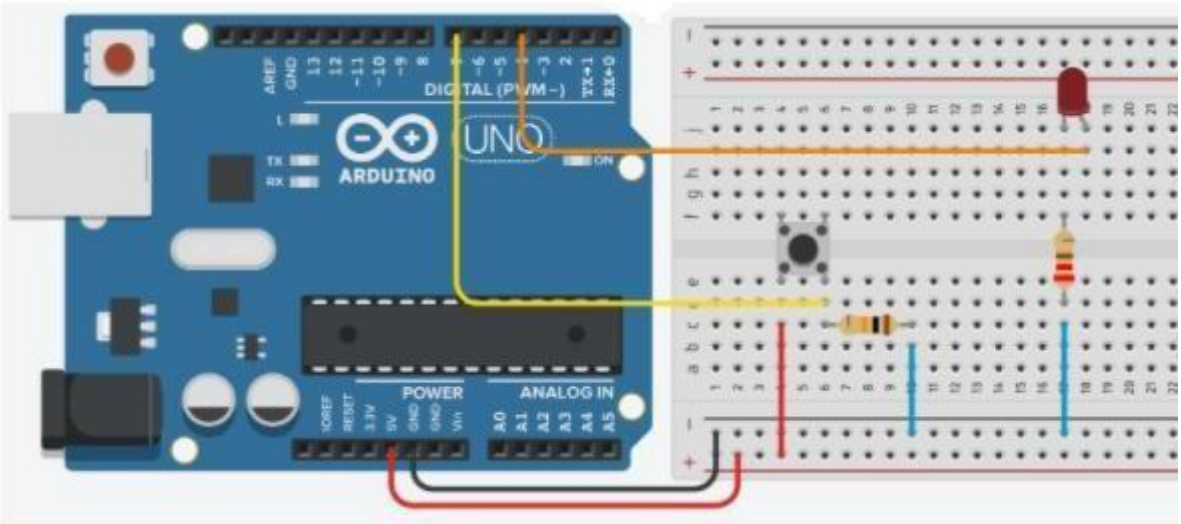


Figura: Interruttori o pulsanti che possono essere utilizzati per il comando IF...THEN

Circuito 7:

Titolo del circuito: Utilizzo del comando ELSE su Arduino

Spiegazione del circuito: Premendo un pulsante collegato al pin 7, il pulsante accende e spegne un (LED), collegato al pin digitale 4. Inoltre, impareremo a determinare i pin con la variabile "int".



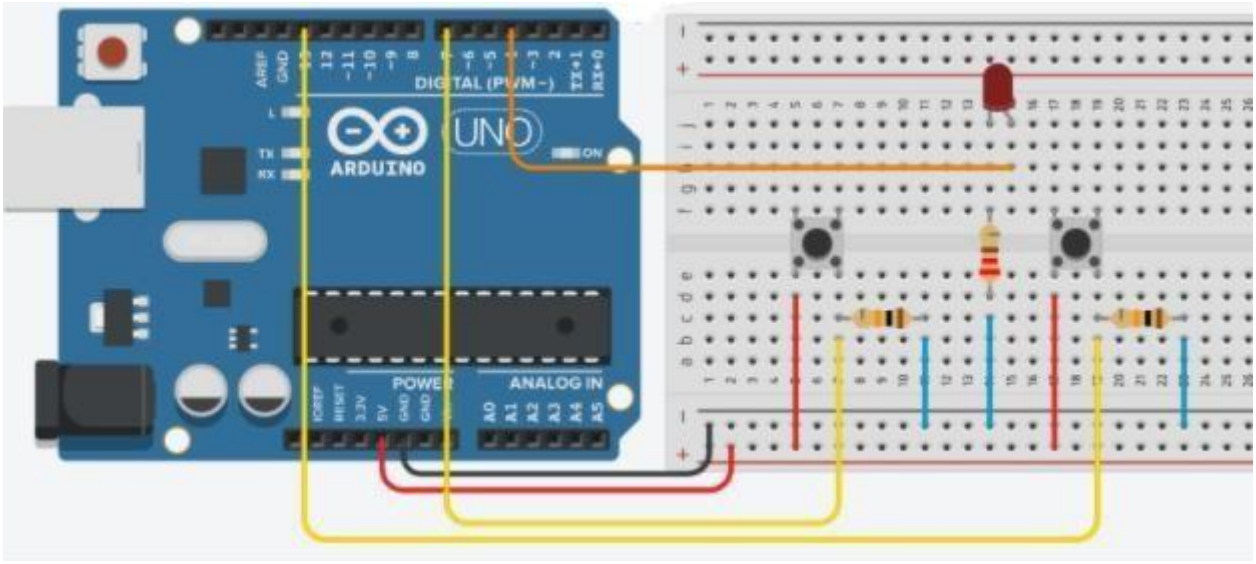

```
 int led = 4;
int buton =7;
void setup()
```

```
{
  pinMode(led, OUTPUT);
  pinMode(buton, INPUT);
}
vuoto loop()
{
    if (digitalRead(buton) == HIGH)          // Il pulsante è attivo,
       digitalWrite(led, HIGH);              // Il led è acceso
     else                                    // In caso contrario,
       digitalWrite(led, LOW);               // Il led è spento.
}
```

Circuito 8:

Titolo del circuito: 2 pulsanti contro 1 LED

Spiegazione del circuito: Un pulsante accende il LED, un altro pulsante lo spegne.




```
/*2 pulsanti contro 1 LED
I pulsanti sono collegati con resistenze da 10K in serie. */

int led = 4;                    // Il pin-4 è assegnato come "led".
int button1 = 7;                // Il pin-7 è assegnato come "button1"
int button2 = 13;               // Il pin-13 è assegnato come "button2"
void setup()
{
  pinMode(led, OUTPUT);         // led=Output
  pinMode(button1, INPUT);      // pulsante1=ingresso
  pinMode(button2, INPUT);      // pulsante2=ingresso
}
vuoto loop()
{
  if (digitalRead(button1) == HIGH)     // SE il "button1" è ON (attivo),
     digitalWrite(led, HIGH);           // Il led è acceso.
   if (digitalRead(button2) == HIGH)    // SE il "button2" è ON (attivo),
     digitalWrite(led, LOW);            // Il led è spento.
}
```

COME USARE IL MONITOR SERIALE ARDUINO

L'IDE Arduino ha una funzione che può essere di grande aiuto per il debug degli sketch o per il controllo di Arduino dalla tastiera del computer.

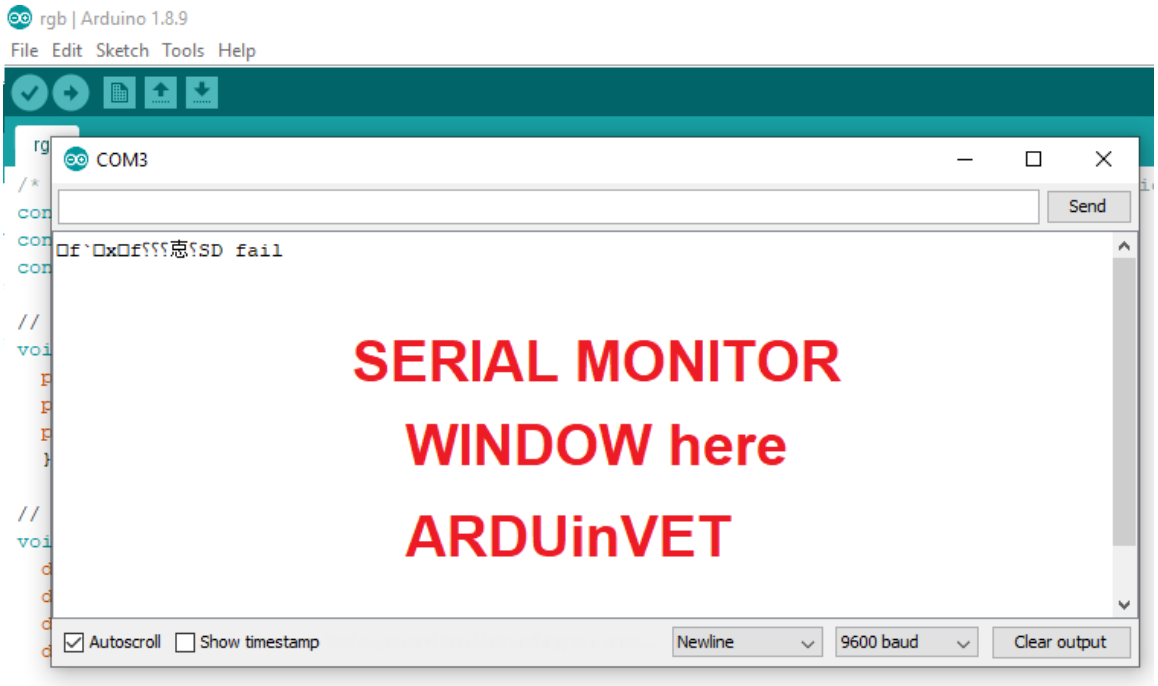
Il Serial Monitor è una finestra pop-up separata che agisce come un terminale separato che comunica ricevendo e inviando dati seriali. Vedere l'icona all'estrema destra dell'immagine qui 

.

I dati seriali vengono inviati su un singolo filo (ma di solito viaggiano su USB nel nostro caso) e consistono in una serie di 1 e 0 inviati sul filo. I dati possono essere inviati in entrambe le direzioni (nel nostro caso su due fili).

Il Serial Monitor viene utilizzato per eseguire il debug degli Sketch software di Arduino o per visualizzare i dati inviati da uno Sketch funzionante. Per poter attivare il Serial Monitor è necessario che Arduino sia collegato via USB al computer.

Aprire il Serial Monitor facendo clic sulla casella Serial Monitor nell'IDE. L'aspetto dovrebbe essere quello della schermata seguente. Assicurarsi che il baud (velocità) sia impostato su 9600. Si trova nell'angolo in basso a destra. (L'importante è che sia impostato allo stesso modo nel nostro programma e qui. Poiché l'impostazione predefinita è 9600, abbiamo impostato il nostro programma



Comandi principali per l'utilizzo di Serial Monitor

Serial.begin(); Imposta la velocità di trasmissione dei dati seriali in bit al secondo (baud). Per comunicare con Serial Monitor, assicuratevi di utilizzare una delle velocità di trasmissione elencate nel menu in basso a destra della schermata. È tuttavia possibile specificare altre velocità, ad esempio per comunicare sui pin 0 e 1 con un componente che richiede una particolare velocità di trasmissione.

Sintassi: Serial.begin(velocità);

Esempio: Serial.begin(9600);

Serial.print();

Stampa i dati sulla porta seriale come testo ASCII leggibile dall'uomo. Questo comando può assumere diverse forme. I numeri vengono stampati utilizzando un carattere ASCII per ogni cifra. I numeri a virgola sono stampati come cifre ASCII, con due cifre decimali.I byte vengono inviati come un singolo carattere. I caratteri e le stringhe vengono inviati così come sono.

Ad esempio:

Serial.print(78;) restituisce "78"
Serial.print('N'); restituisce "N"
Serial.print("Ciao ArduinVet."); restituisce "Ciao ArduinVet".

Serial.println();

Stampa i dati sulla porta seriale come testo ASCII leggibile dall'uomo, seguito da un carattere di ritorno a capo (ASCII 13, o '\r') e da un carattere di newline (ASCII 10, o '\n'). Questo comando assume le stesse forme di Serial.print().

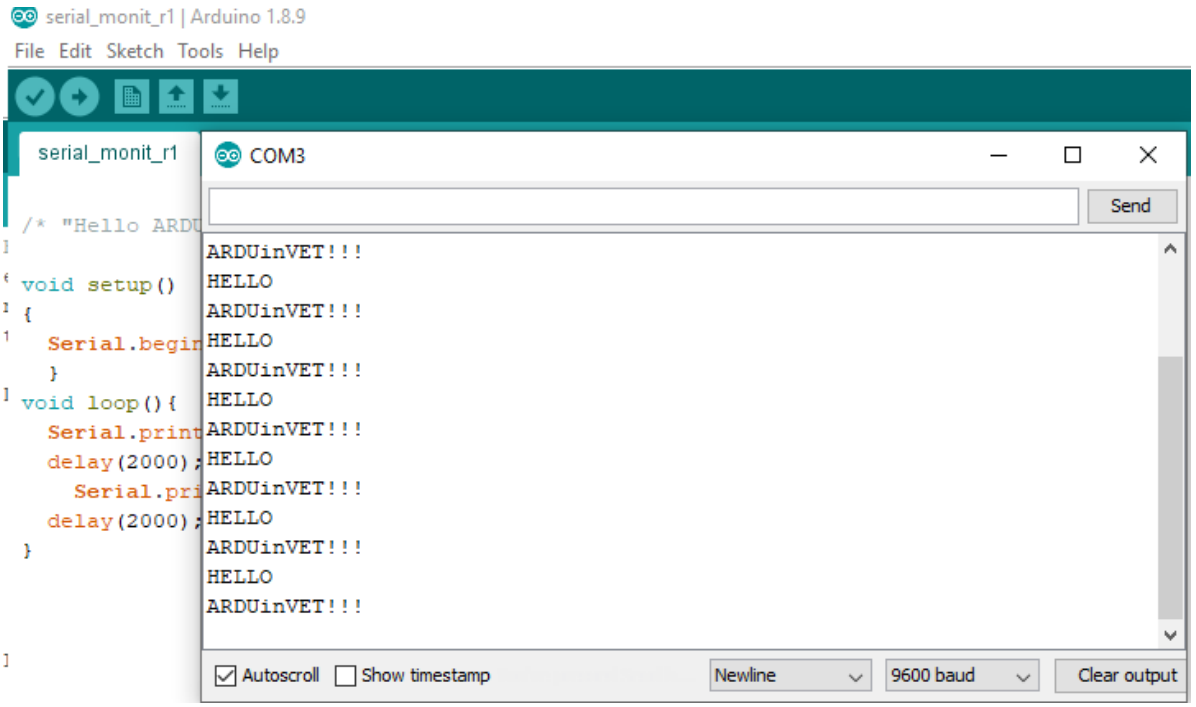
Serial.println(val);
Serial.println(val, format);
Serial.println(13, BIN); restituisce "1101", valore binario di 15 sul monitor seriale.

NOTA: L'unica differenza tra Serial.print e Serial.println è che Serial.println significa che la prossima cosa inviata dalla porta seriale dopo questa inizierà sulla riga successiva. C'è una terza novità che potreste aver notato. C'è qualcosa tra virgolette ("). Si tratta di una stringa.

Circuito 56:
Titolo del circuito: "Applicazione "Hello ARDUinVET" in Serial Monitor".

Spiegazione del circuito: Sul monitor seriale viene visualizzato il messaggio Hello ARDUinVET.

```
/*Applicazione  "Hello ARDUinVET" nel monitor seriale */
void setup()
{
  Serial.begin(9600);     // Velocità di comunicazione seriale
}
vuoto loop()
{
  Serial.println(" CIAO ");
  delay(2000);
  Serial.println("ARDUinVET!!!");
  delay(2000);
}
```

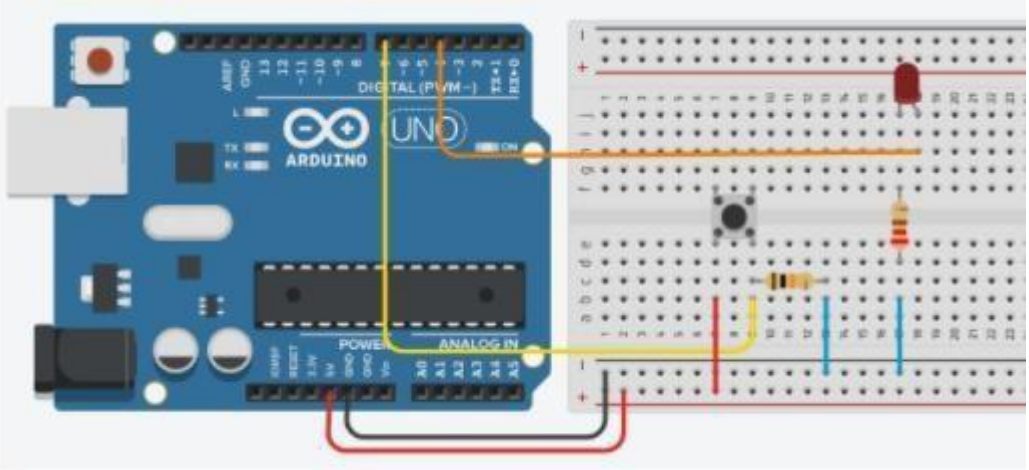


Circuito 57:

Titolo del circuito: "Analisi dello stato dei pulsanti su un monitor seriale".

Spiegazione del circuito: Se si preme il pulsante, sul monitor seriale viene visualizzato il messaggio "Led is ligthing, LED=ON" e il LED si accende.

Se il pulsante non viene premuto, sul monitor seriale viene visualizzato il messaggio ("Pulsante non premuto") e il LED si accende.




```
/*Analisi dello stato dei pulsanti sul monitor seriale
*/ void setup()              {
pinMode(4, OUTPUT);
pinMode(7, INPUT);
Serial.begin(9600);        }
void loop()                      {
 se (digitalRead(7) == HIGH)        // Leggere il segnale digitale sul pin-7
  {
  digitalWrite(4, HIGH);
  Serial.println("Led acceso, LED=ON");
  delay(250);
  }
 altro                    // Se la condizione precedente non è soddisfatta.
  {
```

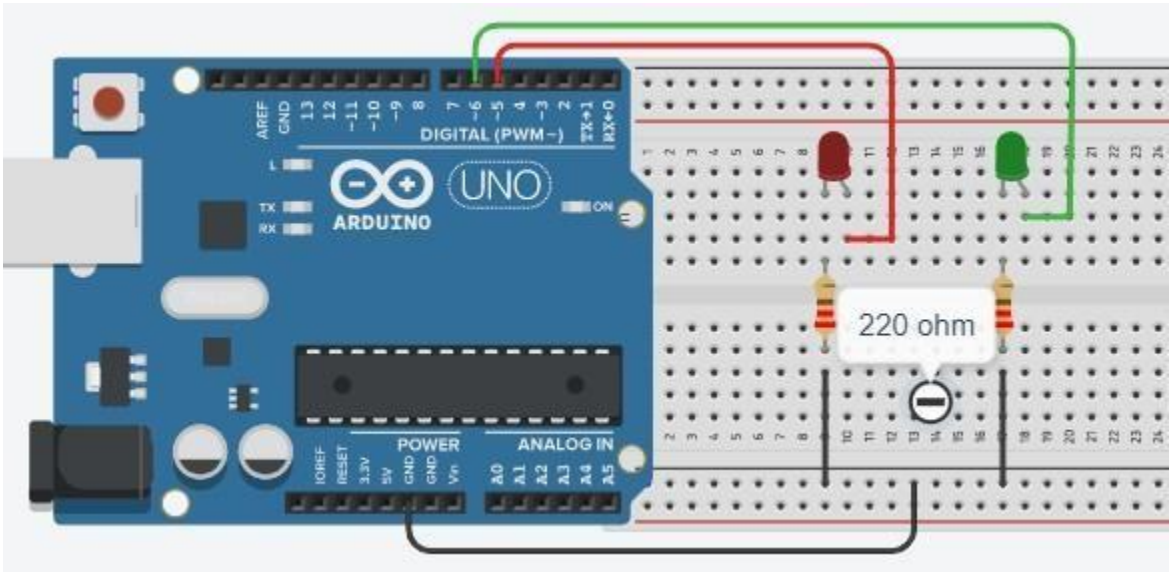
```
digitalWrite(4, LOW);
Serial.println("Il pulsante non è premuto");
delay(1000);
  }
}                          //La vista del monitor seriale è visibile di seguito.
```

```
Button is not pressed
Button is not pressed
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Button is not pressed
```

Circuito 11:

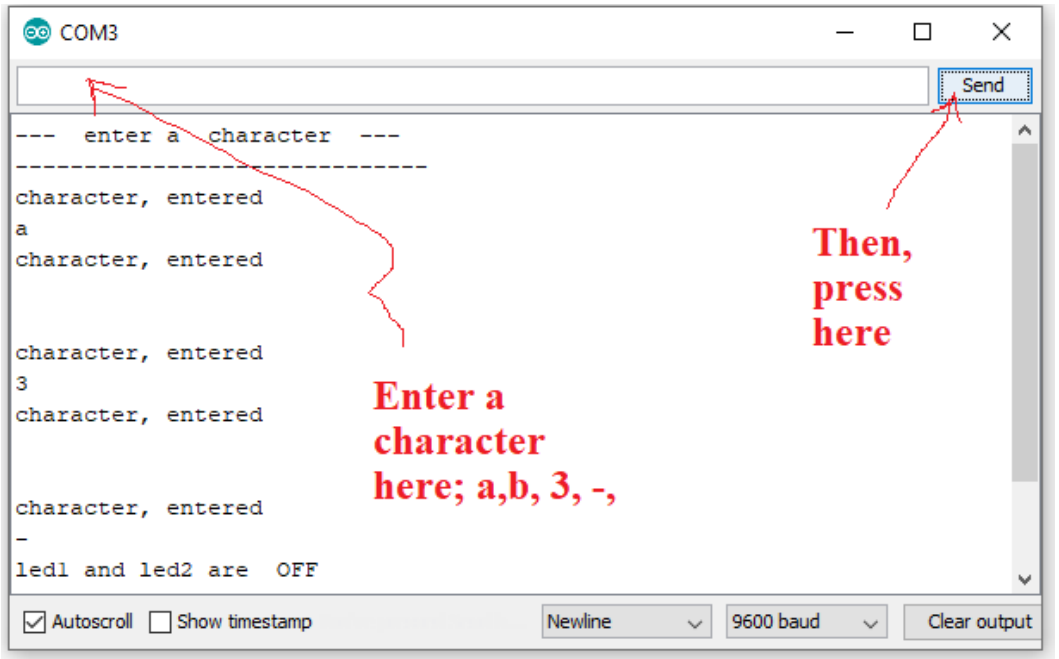
Titolo del circuito: " Invio di dati dal monitor seriale".

Spiegazione del circuito: Se si preme il carattere "A" sulla tastiera, il led1 si accende. Se si preme il carattere "3" della tastiera, si accende il led2.
Se si preme il carattere "-" sulla tastiera, i led1 e led2 sono spenti.



/*Invio di dati dal monitor seriale */

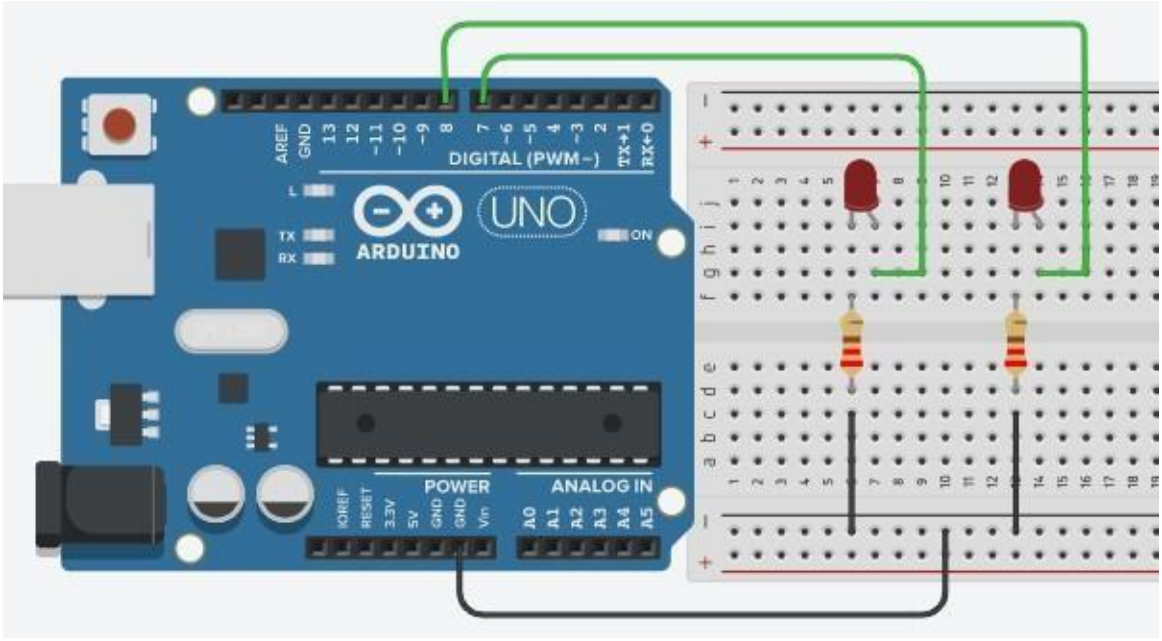
```
char carattere;
#define led1 5
#define led2 6

 void setup() {
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
Serial.begin(9600);
Serial.println ("--- inserire un carattere ---“);
Serial.println ("------------------------------");
}

vuoto loop()
{                                                                    "
if (Serial.available()>0)
{
carattere=Serial.read();
Serial.println ("carattere,  inserito");
Serial.println(carattere);

 if (carattere=='A') digitalWrite (led1, 1);
if (carattere=='a') digitalWrite (led1, 1);
if (carattere=='3') digitalWrite (led2, 1);
if (carattere=='-')
{
digitalWrite(led1,0);
digitalWrite(led2,0);
Serial.println ("led1 e led2 sono spenti");
}}}
```



Circuito 12:

Titolo del circuito: "Imparare il comando FOR"

Spiegazione del circuito:



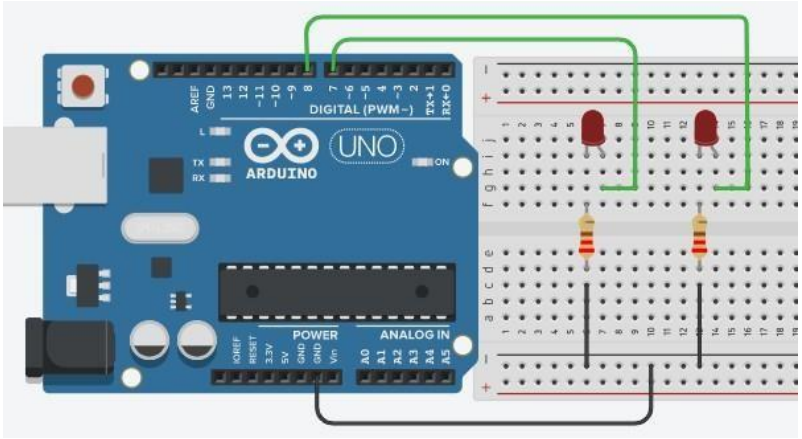

```
/*     "Imparare il comando FOR" */

int led1 = 7;
int led2 = 8;

void setup() {
Serial.begin(9600);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);
}

vuoto loop()
{
for(int i=1; i<6; i++)
{
Serial.println(i);
digitalWrite(led1, 1);
digitalWrite(led2, 1);
delay(1000);

 digitalWrite(led1, 0);
digitalWrite(led2, 0);
delay(1000);
}
}
```

Circuito 13:

Titolo del circuito: " Comando FOR contro Monitor seriale"

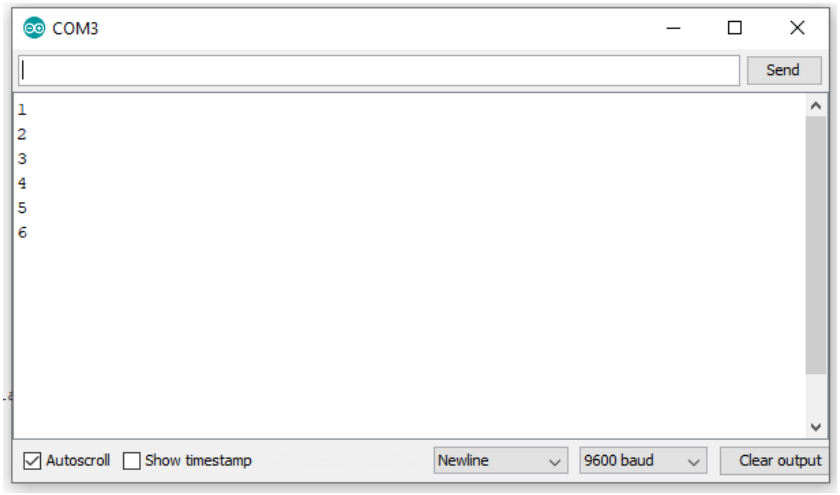
Spiegazione del circuito: In questa applicazione, i LED lampeggiano 6 volte. I numeri 1, 2, 3, 4, 5, 6 appariranno sul monitor seriale. Poi si entrerà nel ciclo while uscendo dal ciclo for. Il programma si ferma qui. Per riavviare il programma è necessario premere il pulsante di reset.

In questo caso, utilizziamo lo stesso circuito del precedente.




```
/* "Comando FOR contro Monitor seriale" */

int led1 = 7;
int led2 = 8;

void setup()                    {
Serial.begin(9600);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);       }

void loop() {

  for(int i=1; i<=6; i++)              // il valore di i = 1, 2,3, 4,5, 6

  {

 Serial.println(i);                    // Scrivere i valori di i, sul monitor seriale
digitalWrite(led1, 1);
digitalWrite(led2, 1);
delay(1000);
digitalWrite(led1, 0);
digitalWrite(led2, 0);
delay(1000);
}
while(1);                  // il ciclo for non entra in un ciclo infinito.
}
```

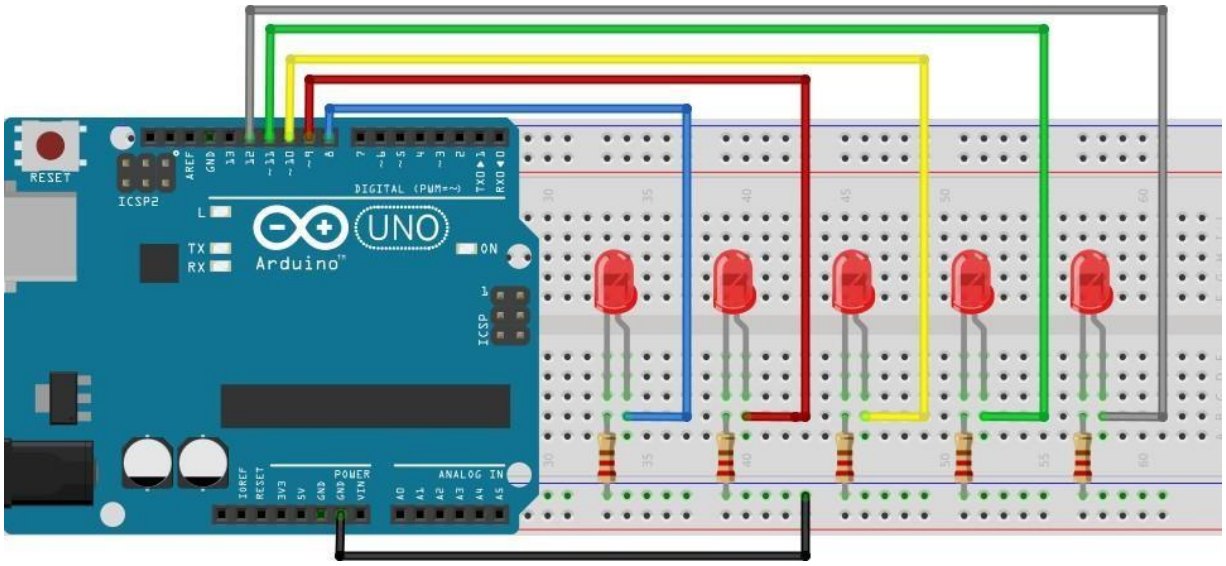
NOTA: per riavviare, è necessario premere il pulsante di reset.



Circuito 14:

Titolo del circuito: " Knight Rider con 5 led utilizzando il comando FOR ".

Spiegazione del circuito:



/*Cavaliere con 5 led utilizzando il comando FOR */

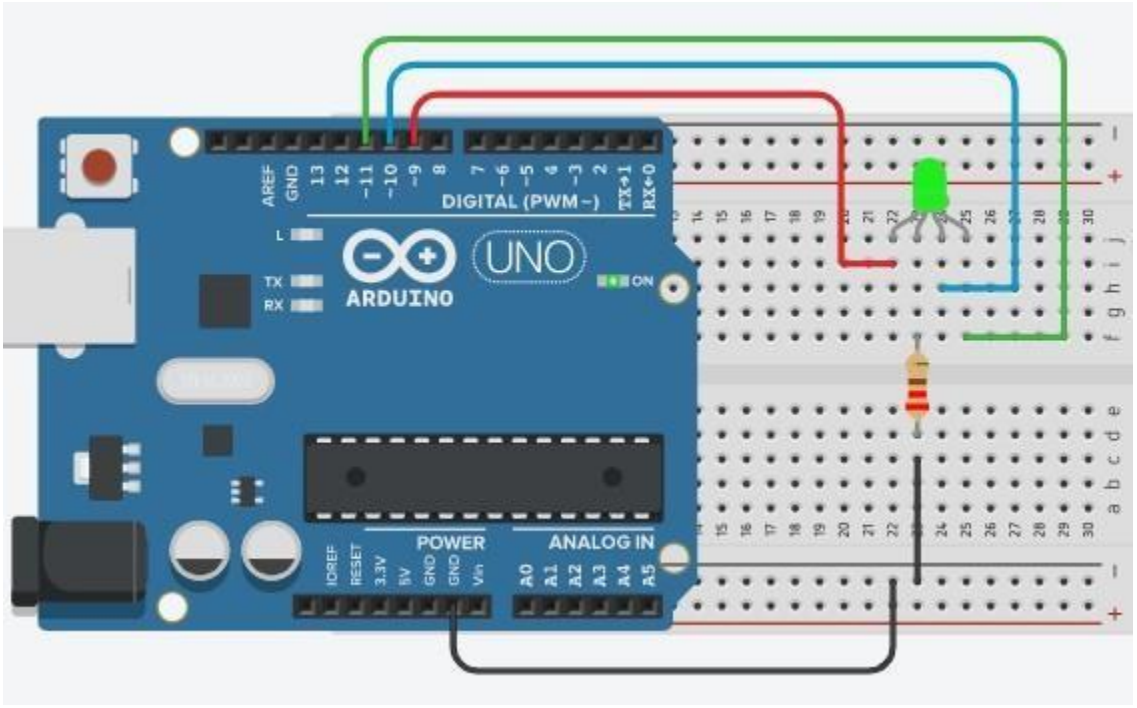
```
void setup() {
pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
pinMode(10, OUTPUT);
pinMode(11, OUTPUT);
pinMode(12, OUTPUT);          }
```

```
void loop() {

 for (int b=8; b<=12; b++)                    // L'upcounter inizia qui
 {
  digitalWrite(b, HIGH);
 delay(150);
 digitalWrite (b, LOW);
 delay(150);
 }

 for (int b=12; b>=8; b--)                    // Il downcounter inizia qui
 {
  digitalWrite (b, HIGH);
 delay(150);
 digitalWrite (b, LOW);
 delay(150);
 }
 }
```

Circuito 15:

Titolo del circuito: "Produzione di colori casuali con led RGB, utilizzando il comando switch/case".

Spiegazione del circuito: Nell'applicazione vengono utilizzati il comando Random e il comando Switch/Case. In questo circuito, i colori rosso, verde e blu saranno ottenuti in modo casuale.



NOTA:

random() : La funzione random genera numeri casuali.

Sintassi: random(max), random(min, max)

min: limite inferiore del valore casuale (opzionale).
max: limite superiore del valore casuale.

/*Produzione di colori casuali con i led RGB, utilizzando il comando switch/case */

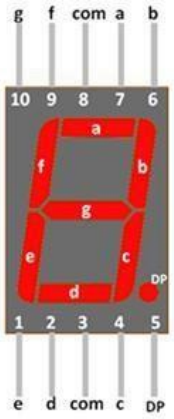
```
#define R 9
#define G 10
#define B 11
int colore;
int dly=3000;

void setup() {
pinMode(R, OUTPUT);
pinMode(G, OUTPUT);
pinMode(B, OUTPUT);
Serial.begin(9600);
}

vuoto loop()
{
colore=random(4);
Serial.println(colore);

switch(colore) {

 case 0:
digitalWrite (R, 1);              //RedLED=ON
digitalWrite (G, 0);
digitalWrite (B, 0);
delay(dly);
break;

 case 1:
digitalWrite (R, 0);            /LED verde=ON
digitalWrite (G, 1);
digitalWrite (B, 0);
delay(dly);
break;

 case 2:                          //BlueLED=ON
digitalWrite (R, 0);
digitalWrite (G, 0);
digitalWrite (B, 1);
delay(dly);
break;

 case 3:                           //Nessun colore
digitalWrite (R, 0);
digitalWrite (G, 0);
digitalWrite (B, 0);
delay(dly);
break;}}
```

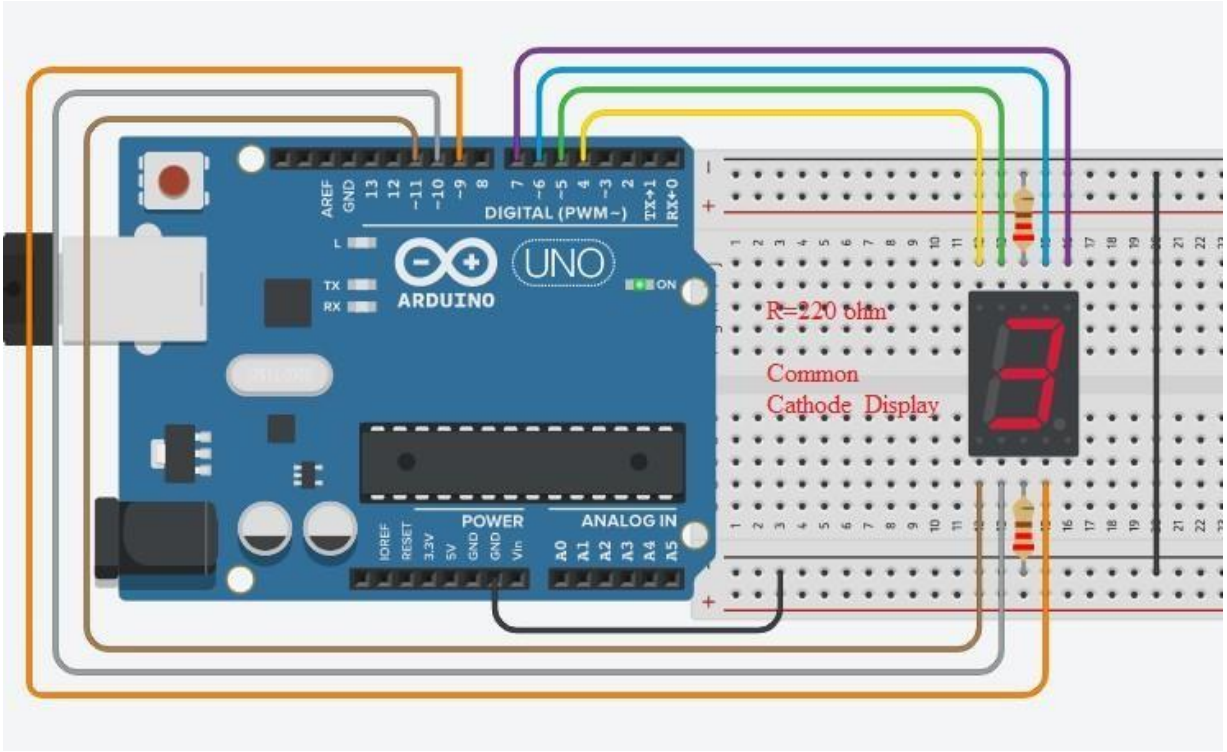
Circuito 16:

Titolo del circuito: "Contaimpulsi 0-9 con display a 7 segmenti ad anodo comune".

Spiegazione del circuito: Per visualizzare un numero sul display, si accende il LED corrispondente tra i 7 LED rappresentati dalle lettere a, b, c, d, e, f, g.

NOTA: un display a sette segmenti è un dispositivo di visualizzazione elettronica per la visualizzazione di numeri decimali.





```
/*    " Contatore 0-9 con display a 7 segmenti Common-Cathoode".    */

int       a=6,   b=7, c=9,      d=10,      e=11,    f=5,     g=4;
int  numero;

void setup() {
pinMode(a, OUTPUT);
pinMode(b, OUTPUT);
pinMode(c, OUTPUT);
pinMode(d, OUTPUT);
pinMode(e, OUTPUT);
pinMode(f, OUTPUT);
pinMode(g, OUTPUT);
}

void loop() {
for(numero=0; numero< =9; numero++)      {
delay(1000);
switch(numero) {

 caso 0:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, HIGH);
digitalWrite (g, LOW);
break;

caso 1:
digitalWrite (a, LOW);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
break;

caso 2:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, LOW);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, LOW);
digitalWrite (g, HIGH);
break;

caso 3:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, HIGH);
break;

caso 4:
digitalWrite (a, LOW );
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;

caso 5:
digitalWrite (a, HIGH);
digitalWrite (b, LOW);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, LOW);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;
```

```
caso 6:
digitalWrite (a, HIGH);
digitalWrite (b, LOW);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;

caso 7:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
break;

caso 8:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;

caso 9:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, LOW);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;
}
}
}
```

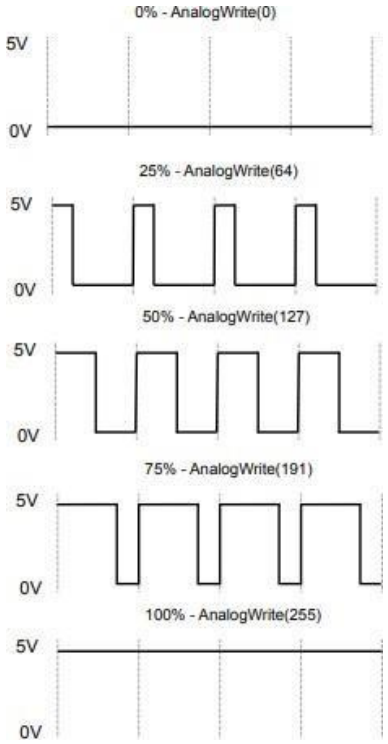
Circuito 17:

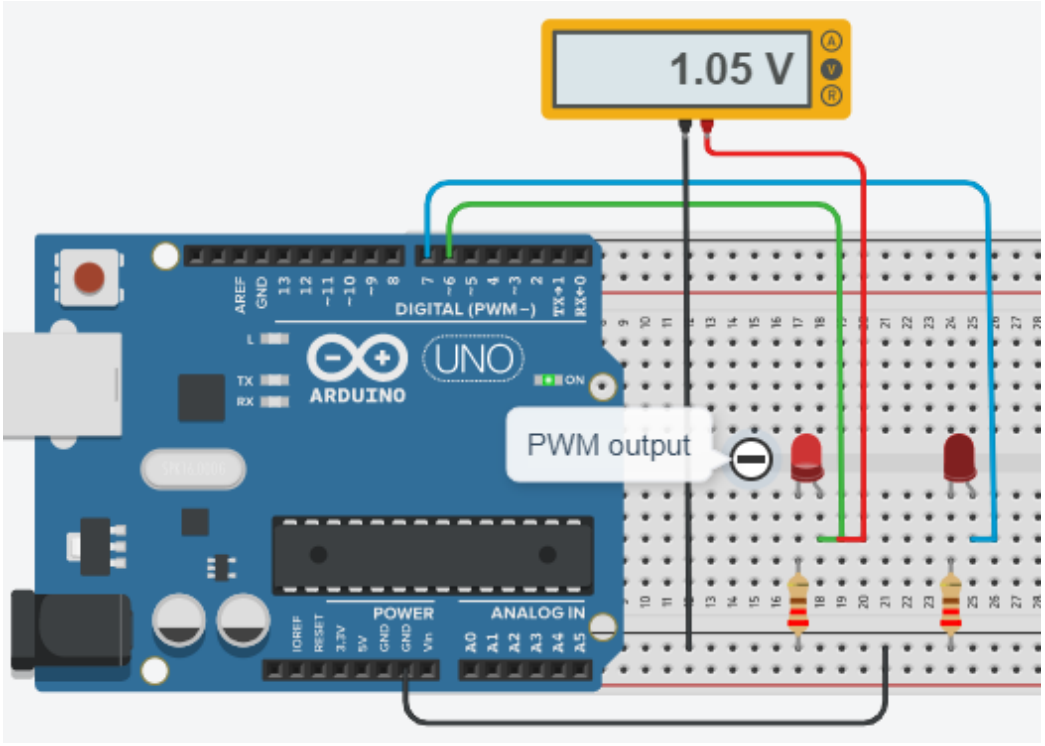
Titolo del circuito: " Utilizzo della tecnica PWN"

Spiegazione del circuito: In pratica, vengono utilizzati il pin-6 come uscita PWM e il pin-7 come uscita digitale. Pertanto, si vuole osservare la differenza tra i due. Applicazioni come la regolazione della luminosità dei led e il controllo della velocità del motore possono essere eseguite con il metodo PWM.

NOTA: Arduino supporta il PWM (su alcuni pin contrassegnati da una tilde (~) sulla scheda Arduino - pin 3, 4, 5, 9, 10 e 11) a 500Hz (500 volte al secondo). 0 significa che non è mai a 5V. 255 significa che è sempre a 5 V. A tale scopo, è necessario effettuare una chiamata a analogWrite() con il valore. Il rapporto tra il tempo "ON" e il tempo totale è chiamato "dutycycle". Un'uscita PWM che è attiva per metà del tempo ha un duty cycle del 50%.

Si può pensare che la PWM sia attiva per x/255, dove x è il valore inviato con analogWrite(). Diseguito è riportato un esempio che mostra l'aspetto degli impulsi:


0% - AnalogWrite(0)
5V
0V
25% - AnalogWrite(64)
5V
0V
50% - AnalogWrite(127)
5V
0V
75% - AnalogWrite(191)
5V
0V
100% - AnalogWrite(255)
5V
0V

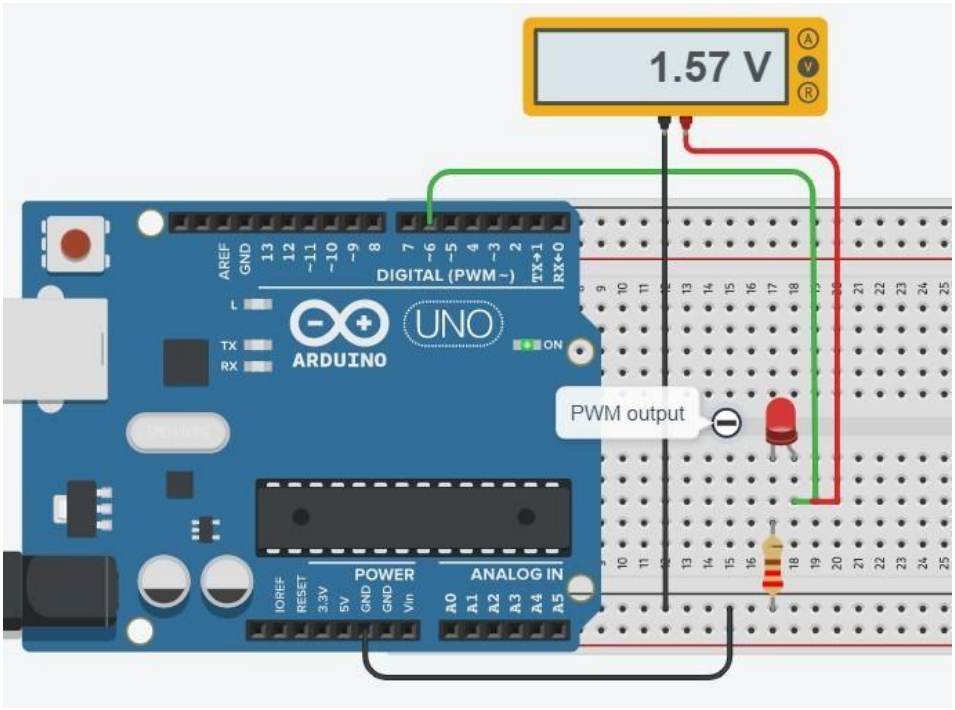

1.05 V
DIGITAL (PWM~)
UNO
ARDUINO
PWM output
POWER
ANALOG IN

```
/* Apprendimento della tecnica PWN tramite analogWrite() */

#definito1 6
#definito2 7

void setup() {
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);       }

void loop()                  {

 analogWrite(led1, 60);        // Il valore 60 viene inviato al pin Led1, ovvero 1,05 V.

 analogWrite(led2,60);         // Il valore 60 viene inviato al pin Led2, ovvero 1,05 V.

delay (100);                   // si accende solo il led1.
}
```

Circuito 18:

Titolo del circuito: "Cambiare la luminosità di un LED dal minimo al massimo".

Spiegazione del circuito: Utilizzando la tecnica PWM, si modifica la luminosità di un LED dal minimo al massimo. In questo caso, verranno utilizzati insieme i comandi analogWrite() e for.

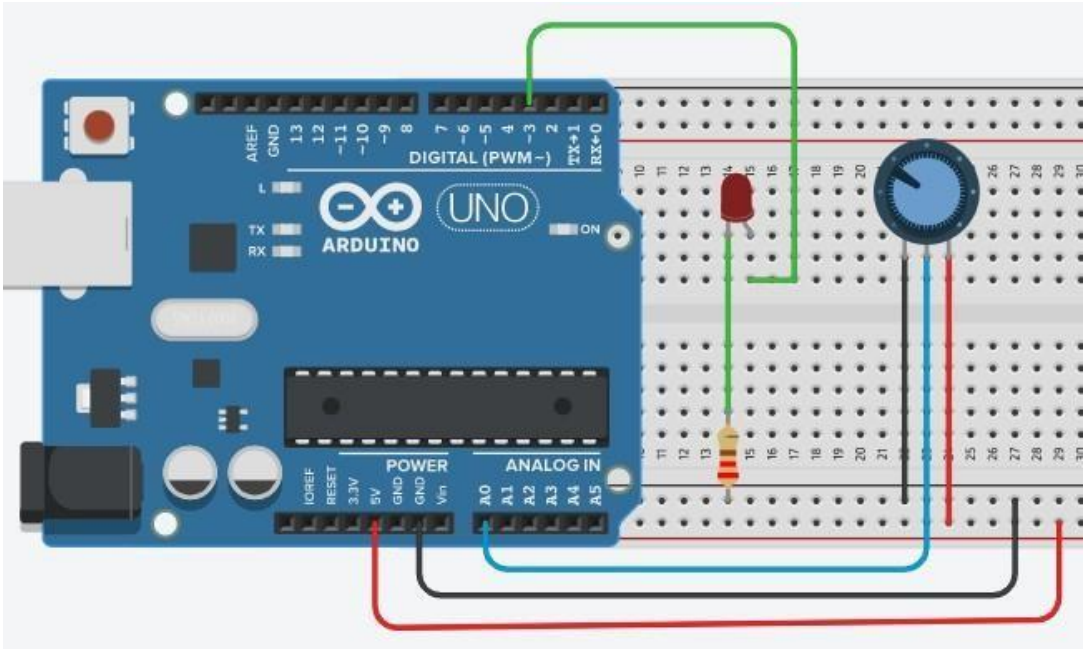



```
/* Per modificare la luminosità di un LED dal minimo al massimo */

#define led16
int a;

void setup()          {
Serial.begin(9600);
pinMode(led1, OUTPUT);    }

void loop() {

 for (a=0; a<=255; a++) {

Serial.println(a);

analogWrite(led1, a);

 ritardo (100);
}               }
```

NON: Sul voltmetro vengono visualizzati valori compresi tra 0 e 5 Volt. Sul monitor seriale vengono visualizzati i numeri da 0 a 255.

Circuito 19:

Titolo del circuito: "Fotenziometro dissolvenza led".
Spiegazione del circuito: Utilizzando il potenziometro (10K), si modifica la luminosità di un LED dal minimo al massimo. In questo caso viene utilizzata la funzione map().



Nota:

Funzione map():

Ri-mappa un numero da un intervallo a un altro. In altre parole, un valore di **fromLow** viene mappato su **toLow**, un valore di **fromHigh** su **toHigh**, valori intermedi su valori intermedi, ecc.

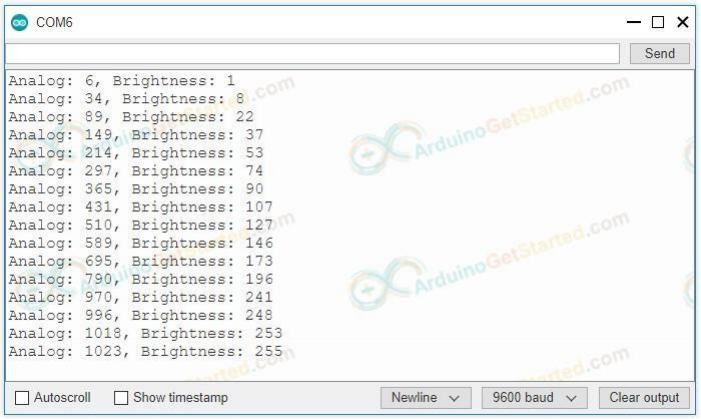
Non vincola i valori all'interno dell'intervallo, perché a volte sono previsti e utili valori fuori dall'intervallo. La funzione constrain() può essere utilizzata prima o dopo questa funzione, se si desidera limitare gli intervalli.

La funzione map() utilizza la matematica degli interi e quindi non genera frazioni, anche se la matematica potrebbe indicare che dovrebbe farlo. I resti frazionari vengono troncati e non vengono arrotondati o mediati.

Sintassi: map(valore, fromLow, fromHigh, toLow, toHigh);

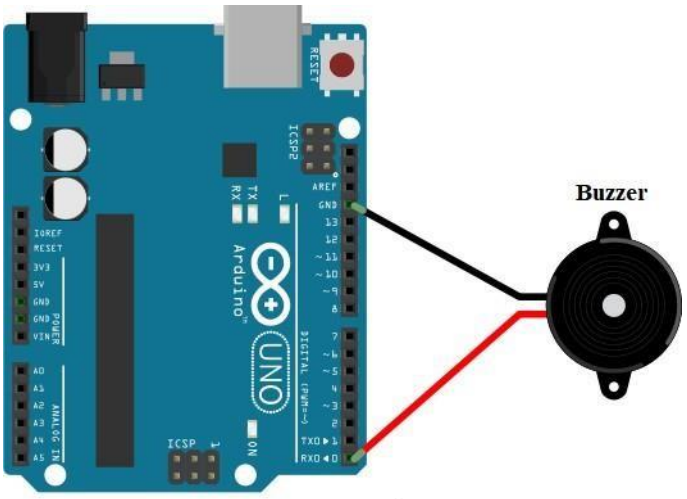
Esempio: val = map(val, 0, 1023, 0, 255);

```
/*Il fotenziometro fa sfumare il led*/
int LED_PIN = 3;
void setup() {
Serial.begin(9600);
pinMode(LED_PIN, OUTPUT);      }
void loop() {
  // legge l'ingresso sul pin analogico A0 (valore compreso tra 0 e 1023)
 int analogValue = analogRead(A0);
  // scala alla luminosità (valore compreso tra 0 e 255)
 int luminosità = map(analogValue, 0, 1023, 0, 255);
  // imposta la luminosità del LED collegato al pin 3
 analogWrite(LED_PIN, luminosità);
  // stampare il valore
  Serial.print("Analogico: ");
  Serial.print(analogValue);
  Serial.print(" Luminosità: ");
  Serial.println(luminosità);
 ritardo(100);
// Schermata Serial Monitor i sotto
}
```



Circuito 20:
Titolo del circuito: " Utilizzare un cicalino"

Spiegazione del circuito: Un cicalino è un dispositivo di segnale audio che può essere meccanico, elettromeccanico o piezoelettrico (in breve piezo). L'uso tipico dei cicalini nell'industria è quello di dispositivi di allarme, che emettono un ronzio o un bip mentre hanno bisogno di ronzare.

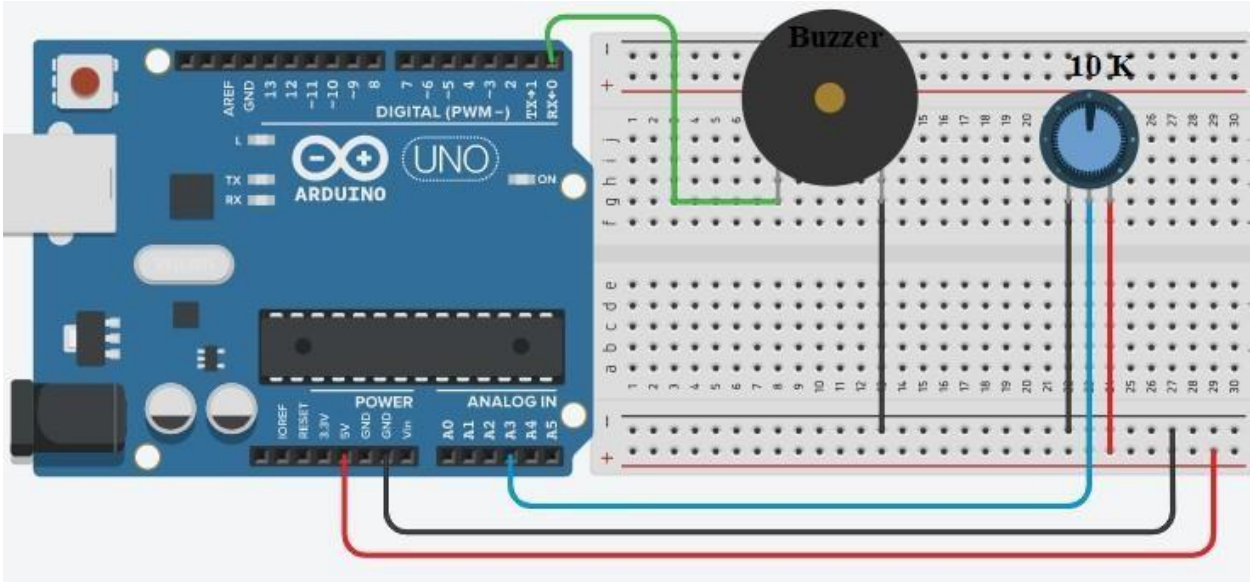


/* Per utilizzare un cicalino nei circuiti Arduino*/

```
#define buzzer_pin 0

 void setup() {

 pinMode(buzzer_pin, OUTPUT);         }
void loop()       {
digitalWrite(buzzer_pin, HIGH);
delay(500);
 digitalWrite(buzzer_pin, LOW);
 delay(500);      }
```

Circuito 21:
Titolo del circuito: " Controllo di un cicalino con un potenziometro".

Spiegazione del circuito: "Quando il valore del potenziometro è superiore a 500, il cicalino suona".



/* Per controllare un cicalino con il potenziometro */

```
const int POT_PIN = A3;              // pin di Arduino collegato al pin del Pot
const int BUZZER_PIN = 0;            // pin di Arduino collegato al pin del
Buzzer const int ANALOG_THRESHOLD = 500;
int analogValue;

void setup() {

  pinMode(BUZZER_PIN, OUTPUT);       // imposta il pin di arduino in modalità di uscita

}

void loop() {

 analogValue = analogRead(POT_PIN);              // leggere l'ingresso sul pin analogico

se(analogValue > ANALOG_THRESHOLD)
digitalWrite(BUZZER_PIN, HIGH);                  // attiva il
cicalino

altro
digitalWrite(BUZZER_PIN, LOW);                   // spegne il cicalino

}
```

Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Titolo Progetto: “Teaching and Learning Arduinos in Vocational Training”

Acronimo Progetto: “ ARDUinVET ”

Progetto N: “2020-1-TR01-KA202-093762”

Modulo LCD e KIT di formazione

LCD Module e Training KIT

In questo modulo vogliamo spiegare come visualizzare i messaggi di stato o le letture dei sensori di Arduino sui display LCD. Sono estremamente comuni e rappresentano un modo rapido per aggiungere un'interfaccia leggibile al vostro progetto.
LCD è l'abbreviazione di Liquid Crystal Display. È un'unità di visualizzazione che utilizza cristalli liquidi per produrre un'immagine visibile. Quando la corrente viene applicata a questo speciale tipo di cristallo, esso diventa opaco bloccando la retroilluminazione che si trova dietro lo schermo. Di conseguenza, quella particolare area diventa più scura rispetto alle altre. È così che i caratteri vengono visualizzati sullo schermo.

Interfacciamento del modulo LCD 16×2 caratteri con Arduino

Esistono diversi tipi di display LCD che è possibile collegare ad Arduino. Il più comune è quello basato sul chip di controllo LCD con interfaccia parallela di Hitachi, chiamato HD44780.
Questi LCD sono ideali per la visualizzazione di solo testo/caratteri, da cui il nome "LCD a caratteri". Il display è dotato di una retroilluminazione a LED e può visualizzare 32 caratteri ASCII su due righe con 16 caratteri su ogni riga.
Sul display è presente un piccolo rettangolo per ogni carattere, ognuno dei quali è una griglia di 5×8 pixel.
Sebbene visualizzino solo testo, sono disponibili in molti formati e colori: ad esempio, 16×1, 16×4, 20×4, con testo bianco su sfondo blu, con testo nero su sfondo verde e molti altri.
La comunità di Arduino ha già sviluppato una libreria per gestire gli LCD HD44780 (**LiquidCrystal Library)**; quindi li interfacceremo in breve tempo.

Di seguito è riportata la piedinatura dell'LCD:

- **VSS:** è un pin di massa e deve essere collegato alla massa di Arduino;
-
- **VDD:** collegato a 5 V;
- **VD:** per la regolazione del contrasto (collegato al pin centrale del potenziometro) ;
- **RS:** controlla in quale zona della memoria dell'LCD vengono memorizzati i dati inviati;
- **R/W:** pin per selezionare la modalità di lettura/scrittura;
- **E:** se abilitato, consente al modulo LCD di eseguire istruzioni speciali;
- **Da D0 a D7:** trasmissione dati;
- **A e K:** anodo e catodo per fornire la retroilluminazione al modulo LCD.

Arduino - Funzioni LCD (comandi)

LiquidCrystal lcd() - Crea una variabile di tipo LiquidCrystal. Il display può essere controllato utilizzando 4 o 8 linee di dati. Nel primo caso, omettere i numeri dei pin da d0 a d3 e lasciare queste linee non collegate. Il pin RW può essere legato a massa invece di essere collegato a un pin di Arduino; in tal caso, ometterlo dai parametri di questa funzione. Sintassi:

LiquidCrystal(rs, enable, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)

Parametri:
rs: il numero del pin di Arduino collegato al pin RS dell'LCD.

rw: il numero del pin di Arduino collegato al pin RW dell'LCD (facoltativo)

enable: il numero del pin di Arduino collegato al pin di abilitazione dell'LCD

d0, d1, d2, d3, d4, d5, d6, d7: i numeri dei pin di Arduino collegati ai corrispondenti pin di dati dell'LCD. d0, d1, d2 e d3 sono opzionali; se omessi, l'LCD sarà controllato solo dalle quattro linee di dati (d4, d5, d6, d7).

lcd.begin() - Inizializza l'interfaccia con lo schermo LCD e specifica le dimensioni (larghezza e altezza) del display. begin() deve essere chiamato prima di qualsiasi altro comando della libreria LCD. Sintassi: lcd.begin(cols, rows)

Parametri:
lcd: una variabile di tipo LiquidCrystal

cols: il numero di colonne del display

righe: il numero di righe del display

lcd.print() - Stampa il testo sull'LCD. Sintassi:

lcd.print(dati)
lcd.print(dati, BASE)

Parametri:
lcd: una variabile di tipo LiquidCrystal

dati: i dati da stampare (char, byte, int, long o stringa)

BASE (opzionale): la base in cui stampare i numeri: BIN per binario (base 2), DEC per decimale (base 10), OCT per ottale (base 8), HEX per esadecimale (base 16).

Restituzioni
byte print() restituirà il numero di byte scritti, anche se la lettura di tale numero è facoltativa

lcd.setCursor() - Posiziona il cursore dell'LCD, ovvero imposta la posizione in cui verrà visualizzato il testo scritto sull'LCD. Sintassi:

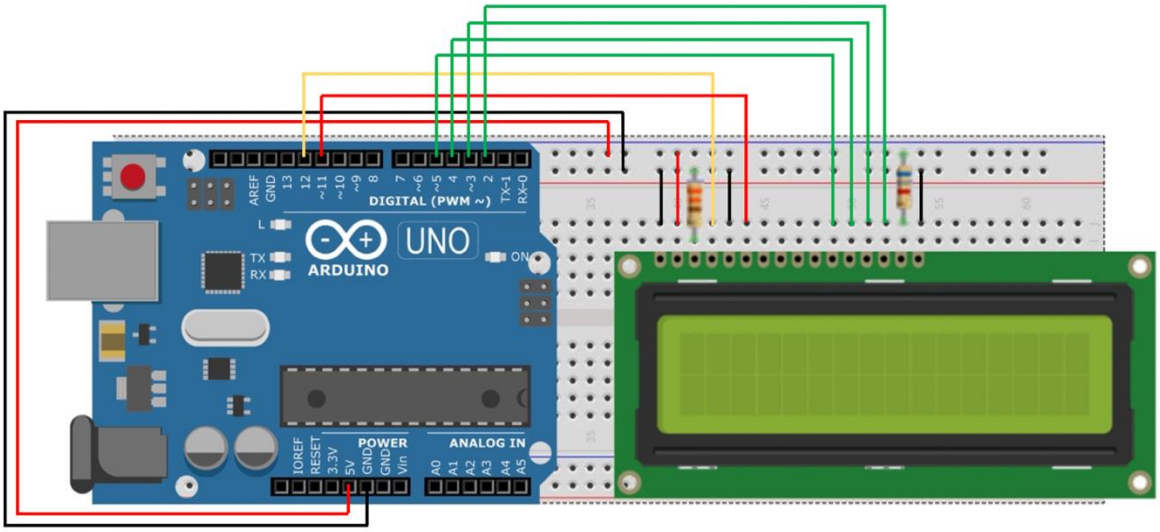
lcd.setCursore(col, row)

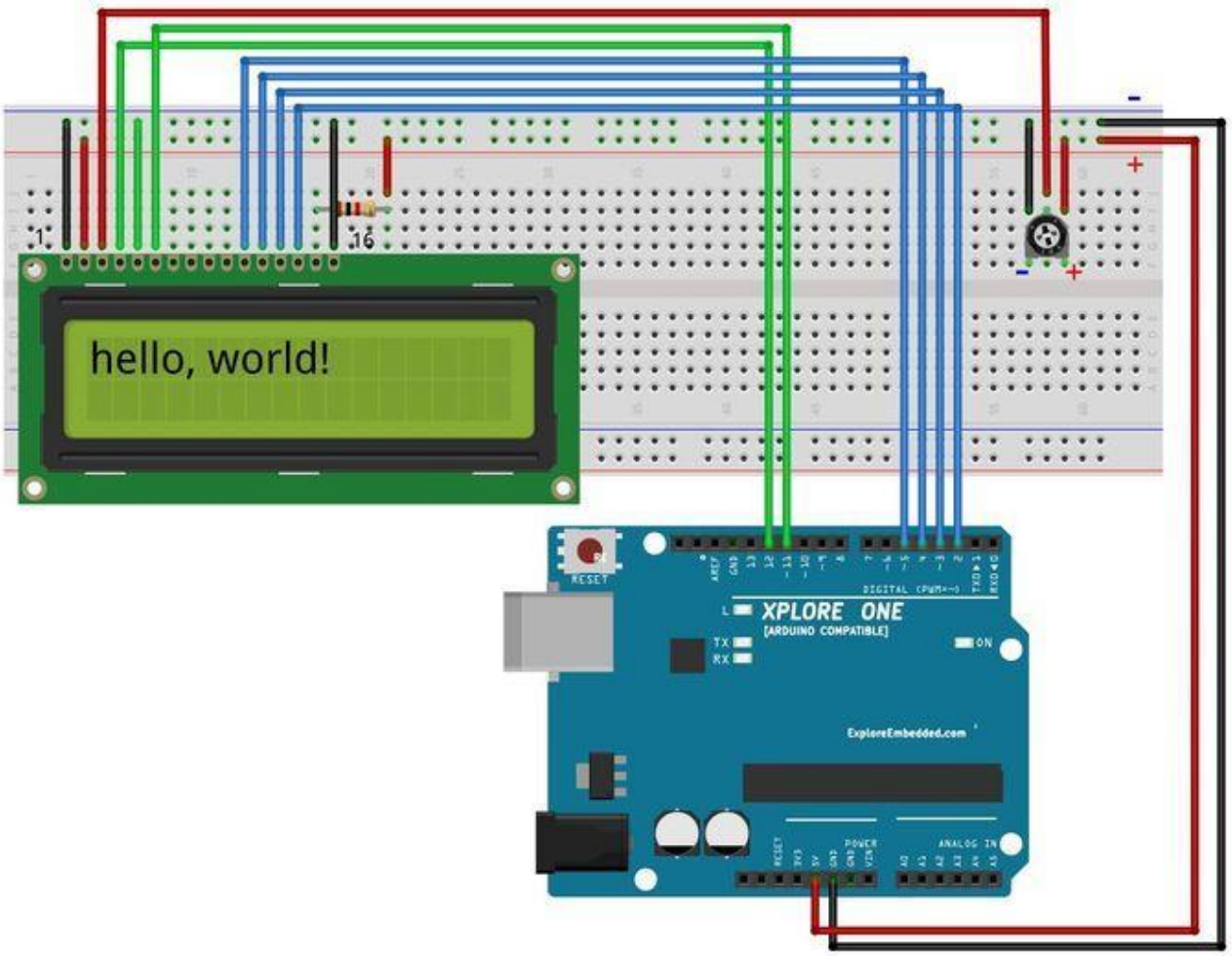
Parametri:
lcd: una variabile di tipo LiquidCrystal

col: la colonna su cui posizionare il cursore (con 0 come prima colonna)

riga: la riga in cui posizionare il cursore (0 è la prima riga)
lcd.clear(); - Cancella lo schermo LCD e posiziona il cursore nell'angolo superiore sinistro.
Sintassi: lcd.clear()
Parametri: lcd: una variabile di tipo LiquidCrystal

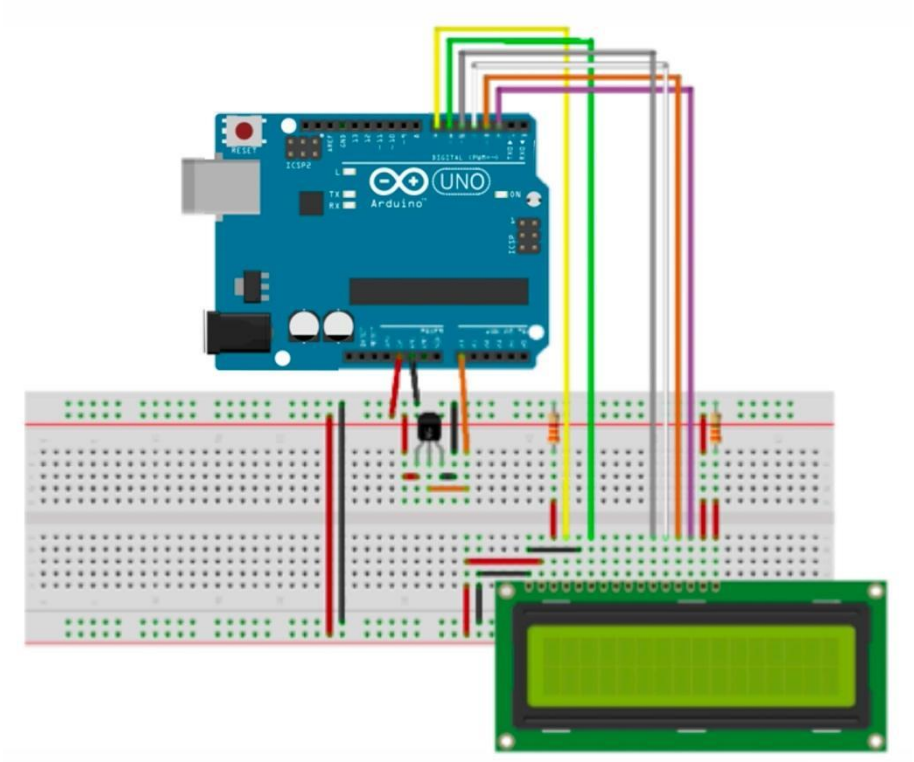
Circuito 1:
Titolo del circuito: "Stampa di un messaggio sul display LCD"
Spiegazione del circuito: è possibile collegare un display LCD al microcontrollore e stampare sullo schermo i messaggi desiderati.
Nota: è necessario includere la libreria LiquidCrystal.h per utilizzare le funzioni LCD descritte di seguito.

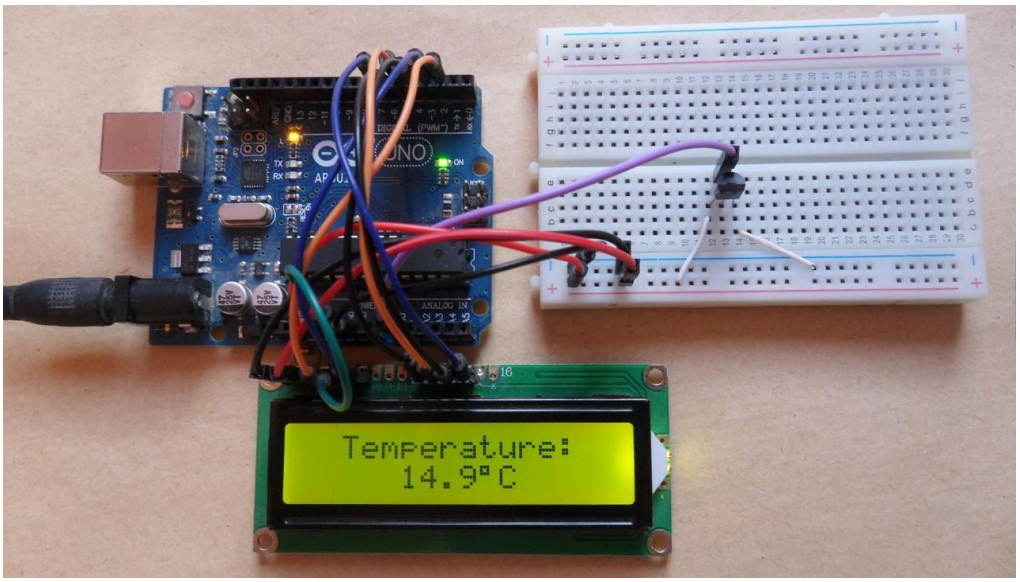




```
/* Stampa un messaggio */

#include < LiquidCrystal.h> // include il codice della libreria

//costanti per il numero di righe e colonne dell'LCD

const int numRows = 2;

const int numCols = 16;

// inizializzare la libreria con i numeri dei pin dell'interfaccia

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

vuoto setup()

{

lcd.begin(numCols, numRows);

lcd.print("hello, world!"); // Stampa un messaggio sull'LCD.

}
```

Circuito 2:
Titolo del circuito: "Termometro digitale"
Spiegazione del circuito: creare un termometro digitale con Arduino. La temperatura verrà rilevata con il sensore di temperatura LM35 e dovrà essere visualizzata su un display LCD e aggiornata ogni secondo.
Il pin Vo del display LCD regola il contrasto dei caratteri visualizzati sul display. Come già detto, deve essere collegato a una resistenza da 3,3 kΩ collegata a GND. Tuttavia, in alcuni display può essere necessario collegare il pin Vo direttamente a GND.
Nota: il sensore LM35 ha 3 terminali: uno per l'alimentazione, uno per la massa e uno per l'uscita della tensione proporzionale alla temperatura rilevata, che è pari a 10 mV per ogni grado centigrado, ed è calibrata in gradi Celsius.



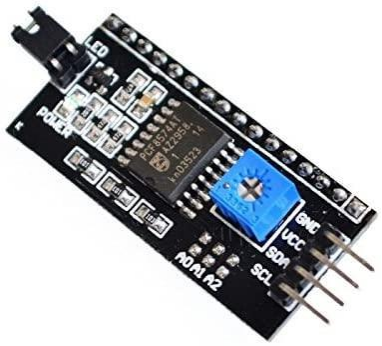



```
/* Termometro digitale */

#include < LiquidCrystal.h> //Libreria per pilotare il display LCD

#define pin_temp A0 //Pin di collegamento al piede del sensore di temperatura Vout

float temp = 0; //Variabile in cui verrà memorizzata la temperatura rilevata

LiquidCrystal lcd(7, 6, 5, 4, 3, 2); //Inizializzazione della libreria con i pin del display LCD
```

```
vuoto setup()

{

  lcd.begin(16, 2); /Impostazione del numero di colonne e righe del display LCD lcd.setCursor(0,
  0); /Spostamento del cursore sulla prima riga (riga 0) e sulla prima colonna lcd.print
  ("Temperatura:"); Stampa del messaggio 'Temperatura:' sulla prima riga

 //*Vref dell'ADC a 1,1V
 (per una maggiore precisione nel calcolo della temperatura)

  IMPORTANTE: se si utilizza Arduino Mega, sostituire INTERNAL con INTERNAL1V1 */
  analogReference(INTERNAL);
}


vuoto loop()
{
  /*calcolare la temperatura ==========================================*/
  temp = 0;

  per (int i = 0; i < 5; i++) { //Esegue l'istruzione successiva 5 volte
    temp += (analogRead(pin_temp) / 9,31); // Calcola la temperatura e la somma alla variabile 'temp'.

  }
  temp /= 5; //Calcola la media matematica dei valori di temperatura
  /*=================================================================*/
  /*Vedo la temperatura sul display LCD
  ==========================================*/
  lcd.setCursor(0, 1); //lcd.print(temp); Spostare il cursore sulla prima colonna e sulla
  seconda riga lcd.print(temp);
  // Stampo su display LCD temperatura

  lcd.print(" C"); // stampa uno spazio e il carattere 'C' sul display
 /*=========================================*/

  delay(1000); /Ritardo di un secondo (può essere modificato)

}
```

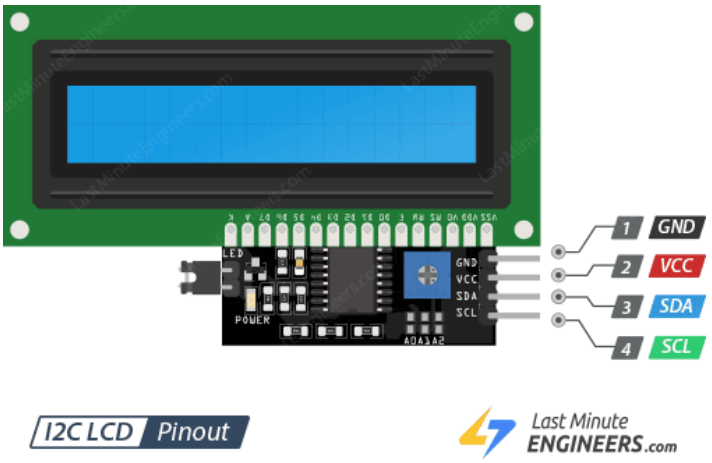
Interfacciamento di LCD I2C con Arduino

Se si desidera collegare un display LCD ad Arduino, è necessario consumare molti pin su Arduino. Anche in modalità a 4 bit, Arduino richiede un totale di sette connessioni, ovvero la metà dei pin di I/O digitali disponibili.
La soluzione consiste nell'utilizzare un display LCD che si interfaccia con il protocollo I2C. Consuma solo due pin di I/O che possono anche non far parte di un set di pin di I/O digitali e possono essere condivisi anche con altri dispositivi I2C.
Il display LCD I2C più diffuso è costituito da un display LCD a caratteri basato su HD44780 e da un adattatore LCD I2C.



La parte più importante dell'adattatore è il chip I/O Expander a 8 bit - PCF8574. Questo chip converte i dati I2C di Arduino nei dati paralleli richiesti dal display LCD. La scheda è dotata anche di un piccolo potenziometro per effettuare regolazioni fini del contrasto del display. Se si utilizza più di un dispositivo sullo stesso bus I2C, potrebbe essere necessario impostare un indirizzo I2C diverso per la scheda, in modo che non entri in conflitto con un altro dispositivo I2C. Per questo motivo, la scheda dispone di tre ponticelli a saldare (A0, A1 e A2).

Un LCD I2C ha solo 4 pin che lo interfacciano con Arduino.



La piedinatura è la seguente:

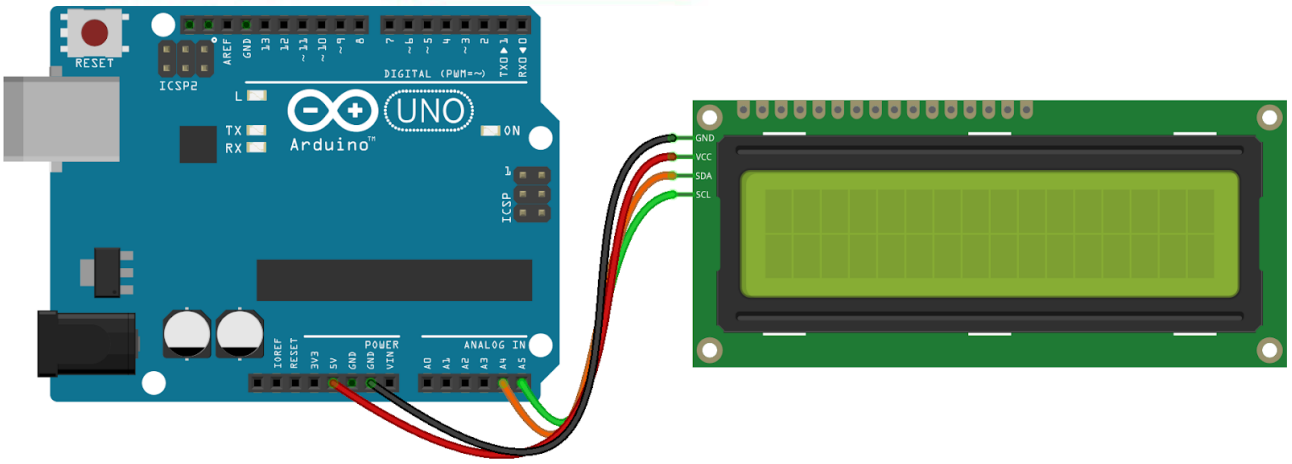
- **GND:** è un pin di massa e deve essere collegato alla massa di Arduino;
- **VCC:** alimenta il modulo e l'LCD. Collegarlo all'uscita 5V di Arduino o a un alimentatore separato;
- **SDA:** è un pin di dati seriali. Questa linea viene utilizzata sia per la trasmissione che per la ricezione. Collegare al pin SDA di Arduino;

- **SCL:** è un pin di clock seriale. È un segnale di temporizzazione fornito dal dispositivo Bus Master. Collegare al pin SCL di Arduino.

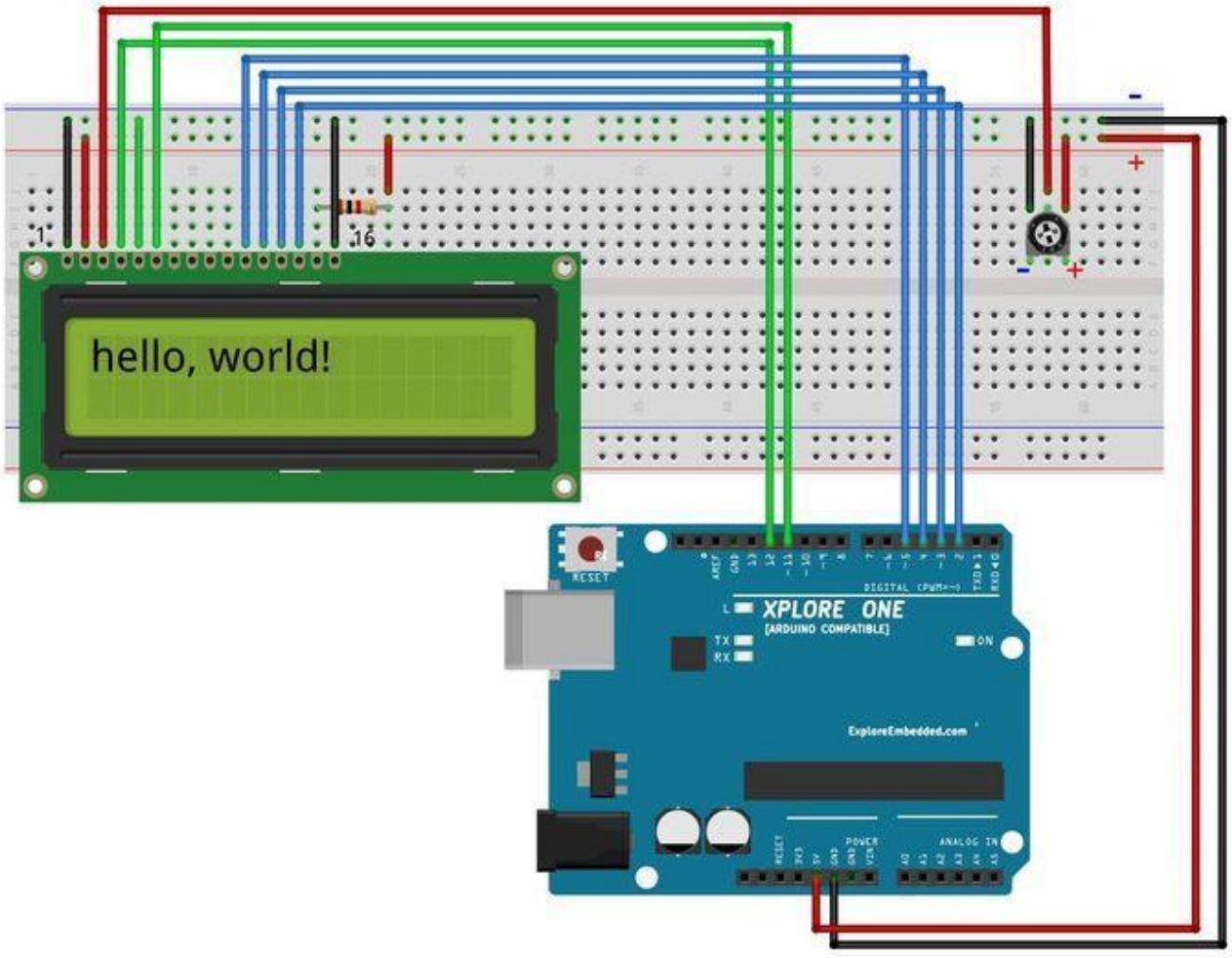
Sulle schede Arduino con layout R3, le linee SDA (linea dati) e SCL (linea di clock) si trovano sui pin vicino al pin AREF. Sono noti anche come A5 (SCL) e A4 (SDA). Fare riferimento alla tabella seguente per identificare i pin corretti a seconda del modello di Arduino.

	SCL	SDA
Arduino Uno	A5	A4
Arduino Nano	A5	A4
Arduino Mega	21	20
Leonardo/Micro	3	2

Circuito 3:
Titolo del circuito: "Stampa di un messaggio sul display LCD I2C".
Spiegazione del circuito: è possibile collegare un display LCD al microcontrollore con soli 4 pin e stampare sullo schermo i messaggi desiderati.
Nota: è necessario installare una libreria chiamata LiquidCrystal_I2C. Questa libreria è una versione migliorata della libreria LiquidCrystal fornita con l'IDE Arduino.



fritzing




```
/* Stampa di un messaggio su un display I2C */
#includere <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F,16,2); // imposta l'indirizzo LCD su 0x3F per un display a 16 caratteri e 2 righe
void setup() {
  lcd.init();
  lcd.clear();
```

```
  lcd.backlight(); // Assicurarsi che la retroilluminazione sia attiva
  // Stampa un messaggio su entrambe le righe dell'LCD.
  lcd.setCursor(2,0); /Imposta il cursore sul carattere 2 della riga 0
  lcd.print("Ciao mondo!");
  lcd.setCursor(2,1); /Spostamento del cursore sul carattere 2 della riga 1
  lcd.print("con protocollo I2C! ");
}
void loop() {
}
```

Interfacciamento del modulo di visualizzazione grafica OLED con Arduino

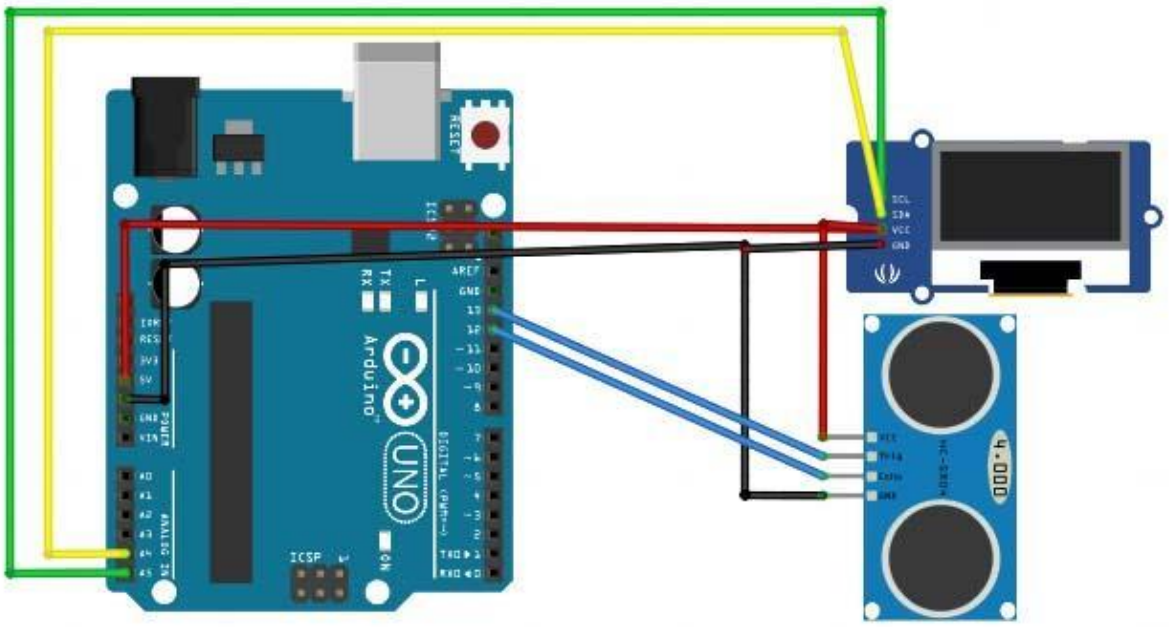
Un'altra possibilità di utilizzare il protocollo I2C è la scelta di un display OLED (Organic Light-Emitting Diode). Sono leggerissimi e sottili e producono immagini più luminose e nitide.

Un display OLED funziona senza retroilluminazione. Per questo motivo il display ha un contrasto così elevato, un angolo di visione estremamente ampio e può visualizzare livelli di nero profondo. L'assenza di retroilluminazione riduce significativamente l'energia necessaria per il funzionamento dell'OLED.
Come si può vedere dalla figura, la piedinatura è la classica interfaccia I2C a quattro pin già vista in precedenza.
La comunità Arduino ha già sviluppato alcune librerie per gestire questi display OLED, come la libreria SSD1306 di Adafruit. Per installare la libreria, andare in Sketch > Include Library > Manage Libraries... Attendere che Library Manager scarichi l'indice delle librerie e aggiorni l'elenco delle librerie installate.

Arduino - Modulo display grafico OLED Funzioni (comandi)

```
pinMode();
display.begin()
display.clearDisplay();
```

Circuito 4:
Titolo del circuito: "Sensore di distanza"
Spiegazione del circuito: La distanza viene rilevata con il sensore a ultrasuoni HC-SR04 e visualizzata su un display OLED.
Nota: I pin di cui ci si deve preoccupare sull'HC-SR04 sono solo quattro: VCC (alimentazione), Trig (attivazione), Echo (ricezione) e GND (massa).

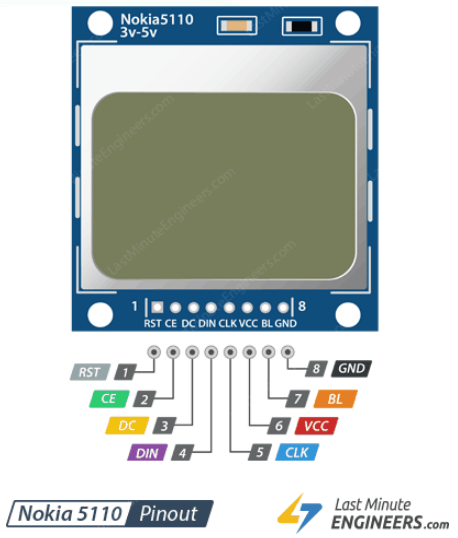



```
/* Sensore di distanza */

#include <SPI.h>    // questa libreria consente di comunicare con i dispositivi SPI, con Arduino come dispositivo master.

#include < Wire.h>    // questa libreria consente di comunicare con i dispositivi I2C / TWI

#include < Adafruit_GFX.h>          // le librerie del display OLED
#includere <Adafruit_SSD1306.h>

#define CommonSenseMetricSystem
#define trigPin 13//    definire i pin del sensore
#define echoPin 12


void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //inizializza con l'addr I2C 0x3C (128x64)
  display.clearDisplay();
```

```
}
void loop() {
  lunga durata, distanza;

  digitalWrite(trigPin, LOW);
  ritardoMicrosecondi(2);
  digitalWrite(trigPin, HIGH);
  ritardoMicrosecondi(10);
  digitalWrite(trigPin, LOW);

  durata = pulseIn(echoPin, HIGH);
  distanza = (durata/2) / 29,1;
  display.setCursor(22,20); //il cursore di impostazione del display a colori
  display.setTextSize(3); //dimensione del testo
  display.setTextColor(WHITE);        //se si scrive in nero si cancellano le cose
  display.println(distanza);      //stampa la nostra variabile
  display.setCursore(85,20);
  display.setTextSize(3);

    display.println("cm");

  display.display();

  ritardo(500);
  display.clearDisplay();

  Serial.println(distanza);//debug


}
```

Interfacciamento del display LCD grafico del Nokia 5110 con Arduino

È possibile interfacciare Arduino con piccoli LCD simili a quelli utilizzati da Nokia nei telefoni cellulari 3310 e 5110. Questi display sono piccoli (solo circa 1,5″), economici, facili da usare, a basso consumo (solo 6-7 mA) e possono visualizzare sia testo che bitmap.



Si tratta di display grafici di 84×48 pixel. Si interfacciano con i microcontrollori attraverso un'interfaccia bus seriale simile all'SPI. L'LCD è dotato anche di una retroilluminazione in diversi colori, come rosso, verde, blu e bianco. La retroilluminazione non è altro che quattro LED disposti intorno ai bordi del display.
Dispone di 8 pin che lo interfacciano con Arduino; la piedinatura è la seguente:

- **RST:** azzera il display. È un pin attivo basso, il che significa che è possibile resettare il display tirandolo verso il basso. È anche possibile collegare questo pin al reset di Arduino in modo che resetti automaticamente lo schermo;

- **CE (Chip Enable**): viene utilizzato per selezionare uno dei dispositivi collegati che condividono lo stesso bus SPI;

- **D/C (Data/Command): il** pin indica al display se i dati ricevuti sono comandi o dati visualizzabili;

- **DIN:** è un pin di dati seriali per l'interfaccia SPI;

- **CLK:** è un pin di clock seriale per l'interfaccia SPI;

- **VCC:** alimenta l'LCD e si collega al pin da 3,3 V di Arduino;

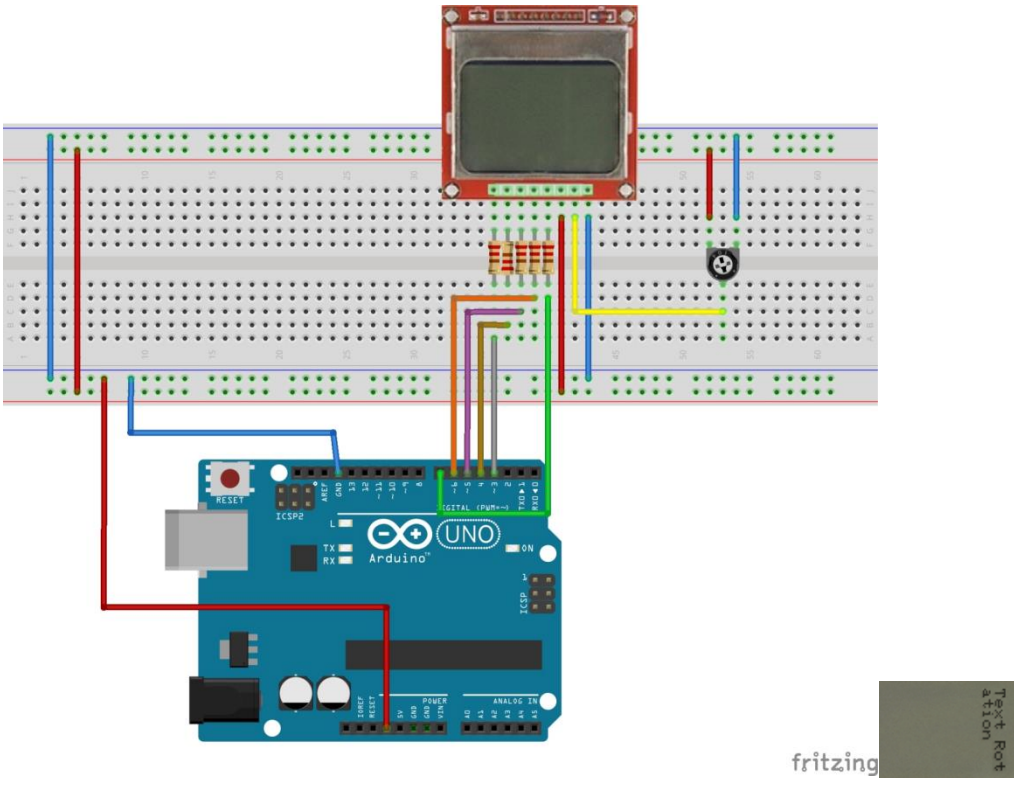
- **BL(Backlight):** controlla la retroilluminazione del display. Per controllarne la luminosità, è possibile aggiungere un potenziometro o collegare questo pin a un qualsiasi pin di Arduino con capacità PWM;

- **GND:** è un pin di massa e deve essere collegato alla massa di Arduino.

È possibile collegare i pin di trasmissione dati a qualsiasi pin di I/O digitale. L'LCD ha livelli di comunicazione a 3 V, quindi non possiamo collegare direttamente questi pin ad Arduino. Un modo è quello di aggiungere resistenze in linea con ciascun pin di trasmissione dati. È sufficiente aggiungere resistenze da 10kΩ tra i pin CLK, DIN, D/C e RST e una resistenza da 1kΩ tra CE. Il pin della retroilluminazione (BL) è collegato a 3,3 V tramite una resistenza di limitazione della corrente da 330Ω. È possibile aggiungere un potenziometro o collegare questo pin a qualsiasi pin Arduino con capacità PWM, se si desidera controllarne la luminosità.
La comunità Arduino ha già sviluppato alcune librerie per gestire questi display NOKIA, come la libreria PCD8544 Nokia 5110 LCD di Adafruit. Per installare la libreria, andare in Sketch > Include Library > Manage Libraries... Attendere che Library Manager scarichi l'indice delle librerie e aggiorni l'elenco delle librerie installate.

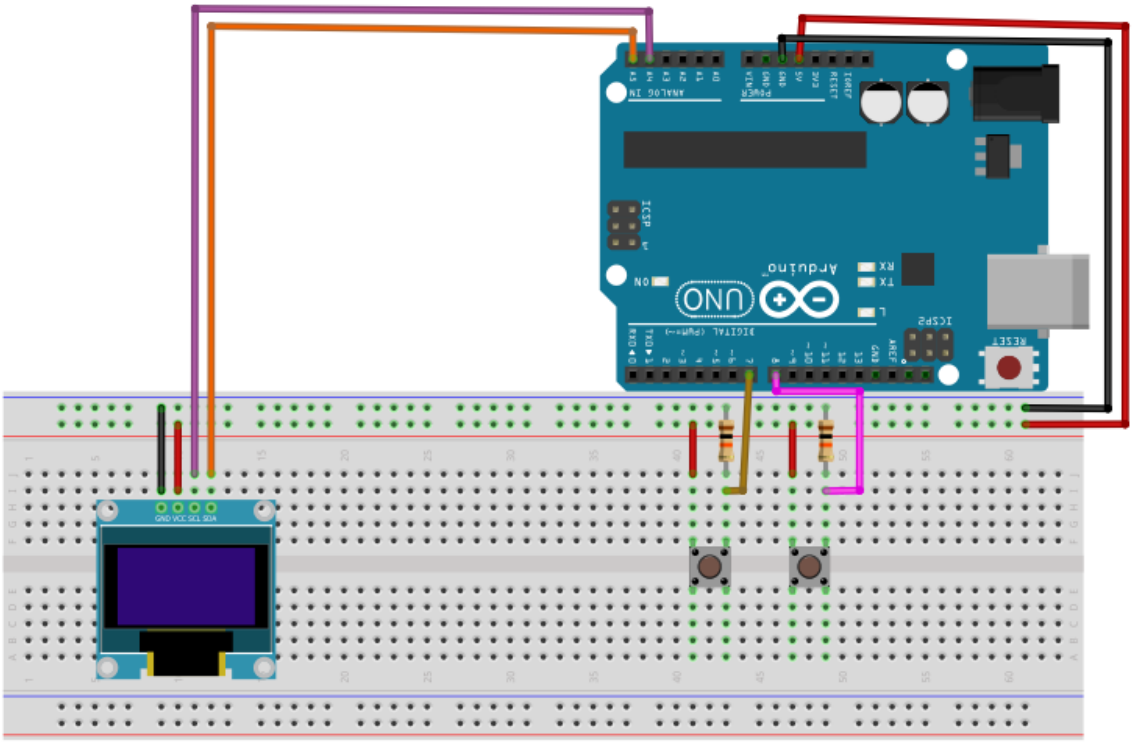
Circuito 5:
Titolo del circuito: "Rotazione del testo"
Spiegazione del circuito: È possibile ruotare il contenuto del display richiamando la funzione setRotation(). Ciò consente di visualizzare il display in modalità verticale o di capovolgerlo.
Nota: la funzione accetta un solo parametro che corrisponde a 4 rotazioni cardinali. Questo valore può essere un qualsiasi numero intero non negativo a partire da 0. Ogni volta che si aumenta il valore, il contenuto del display viene ruotato di 90 gradi in senso antiorario. Ad esempio:

- 0 - Mantiene l'orientamento standard dello schermo in orizzontale.
- 1 - Ruota lo schermo di 90° verso destra.
- 2 - Capovolge lo schermo.
- 3 - Ruota lo schermo di 90° a sinistra.



```
/* Rotazione del testo */

#include < SPI.h>     // questa libreria permette di comunicare con i dispositivi SPI, con Arduino come dispositivo master.

#include < Adafruit_GFX.h>            // le librerie del display OLED
#includere <Adafruit_PCD8544.h>

// Dichiarare l'oggetto LCD per il software SPIAdafruit_PCD8544(CLK,DIN,D/C,CE,RST);
Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);

void setup() {
        Serial.begin(9600);

        //Inizializza il display
        display.begin();

display.setContrast(57);        // si può modificare il contrasto per adattare la visualizzazione

        display.clearDisplay();         // Cancella il buffer.

// Rotazione del testo
mentre(1)
{
        display.clearDisplay();
        display.setRotation(rotatetext);
        display.setTextSize(1);
        display.setTextColor(BLACK);
        display.setCursore(0,0);
        display.println("Rotazione del testo");
        display.display();
        ritardo(1000);
        display.clearDisplay();
        rotatetext++;
}
}

void loop() {}
```

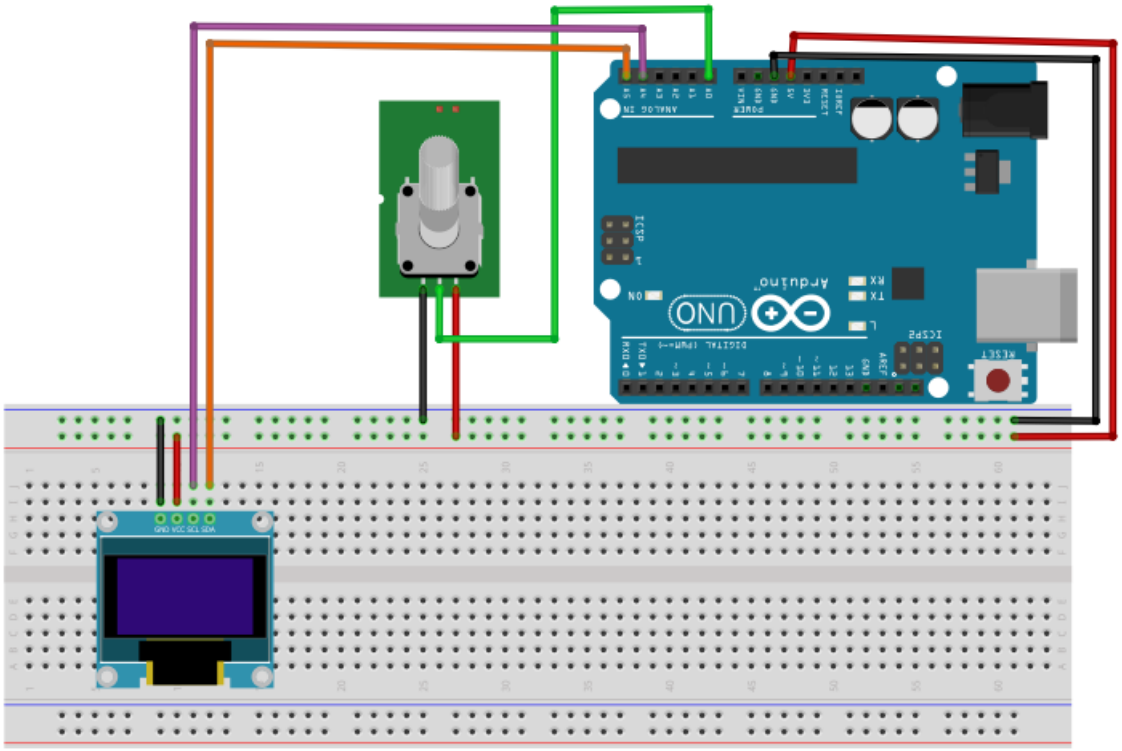
Circuito 6:
Titolo del circuito: "Contapassi"
Spiegazione del circuito: un display OLED che mostra un numero, che può essere incrementato e decrementato facendo clic su due diversi pulsanti.



```
#include <U8glib.h> //lcd libraries#include
"U8glib.h "
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0); //modello di
visualizzazione
int button_plus = 8, button_minus = 7; //dichiarazione dei pulsanti dei pin
int stato_più, stato_meno, numero=0;
void setup() {
  pinMode(pulsante_più,INPUT); pinMode(pulsante_meno,INPUT);
  Serial.begin(9600);
  u8g.setFont(u8g_font_fub25n); /font da utilizzare per la scrittura del numero}.
void write_number(void) {
u8g.setPrintPos(10,50);
```

```
  u8g.print(numero);
}
void loop() {
  //pulsanti stato_più = digitalRead(pulsante_più); stato_meno = digitalRead(pulsante_meno);
  if(state_plus==1)
  {
  number++;
  delay(50);
  }
  if(stato_minimo==1)
{
numero--;
ritardo(50);
}
  //scrivere il numero sul display
  u8g.firstPage();
do { write_number();
} while ( u8g.nextPage() );
    u8g.firstPage();
}
```

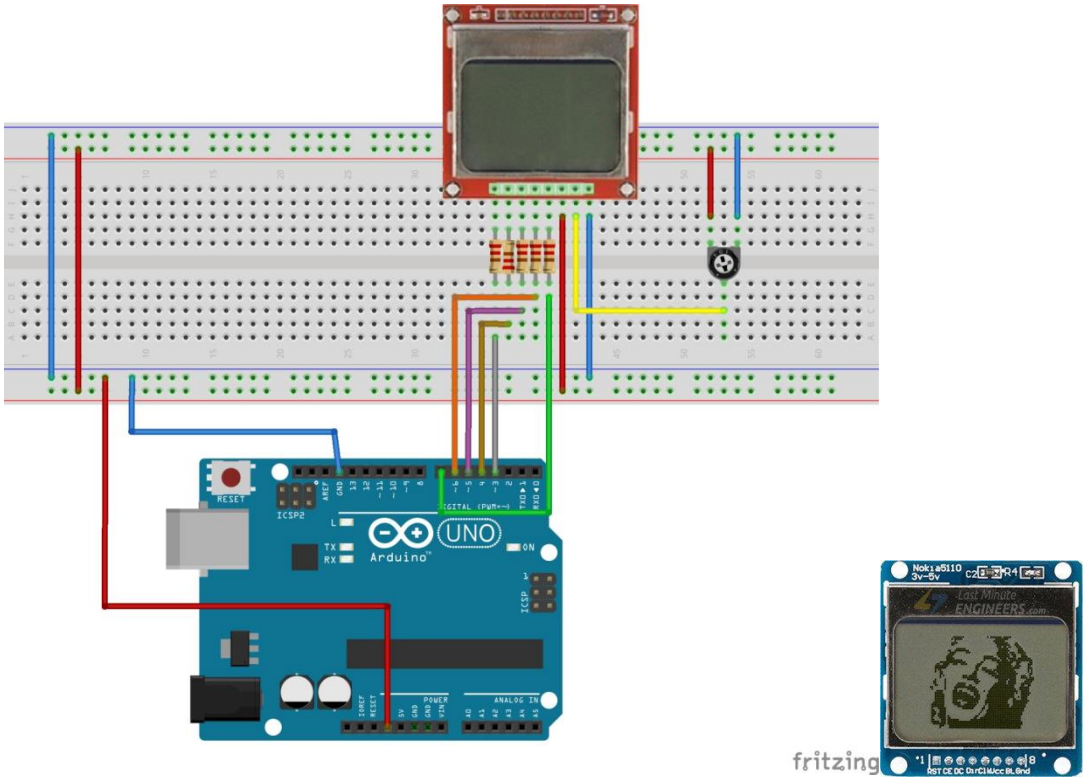
Circuito 7:
Titolo del circuito: "Contatore con potenziometro"
Spiegazione del circuito: un display OLED che mostra un numero, che aumenta e diminuisce quando si ruota un potenziometro.



```
#include <U8glib.h> //Lcd librerie
#include "U8glib.h"
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0); // modello di visualizzazione

int potenziometro = 0; //dichiarazione e potenziometro
int state_pot, stop_pot, differenza, numero=0;

void setup() {
  Serial.begin(9600);
  u8g.setFont(u8g_font_fub25n);/font da utilizzare per la scrittura del numero
}

void write_number(void) {
  u8g.setPrintPos(10,50);
  u8g.print(numero);
}

void loop() {
  stato_pot = analogRead(potenziometro);
```

```
  //potenziometro
  stop_pot = state_pot;//salva il valore quando è stop per vedere se il potenziometro gira in senso orario o antiorario
  ritardo(100);
  stato_pot = analogRead(potenziometro);
  differenza = stop_pot-state_pot;
  if((differenza>2)||(differenza<2))//viene utilizzato per evitare di effettuare operazioni se il potenziometro è fermo
  {
    numero=numero-differenza/5;//se la differenza è maggiore di 0 allora gira in senso orario e aggiunge altrimenti sottrae
    ritardo(100);
  }

  //scrivere il numero sul display
  u8g.firstPage();
  fare {
    write_number();
  } while
  ( u8g.nextPage() );
    u8g.firstPage();
}
```

Circuito 8:
Titolo del circuito: "Immagine Marilyn Bmp"
Spiegazione del circuito: Come disegnare immagini bitmap sul display LCD del Nokia 5110. In questo esempio è presente un ritratto di Marilyn Monroe.
Nota: la risoluzione dello schermo del Nokia 5110 LCD è di 84×48 pixel, quindi le immagini più grandi non verranno visualizzate correttamente. Per visualizzare un'immagine bitmap sul display LCD del Nokia 5110 è necessario richiamare la funzione drawBitmap(). La funzione richiede sei parametri: coordinata X dell'angolo superiore sinistro, coordinata Y dell'angolo superiore sinistro, array di byte della bitmap monocromatica, larghezza della bitmap in pixel, altezza della bitmap in pixel e Colore. Nel nostro esempio, l'immagine bitmap ha una dimensione di 84×48 pixel. Pertanto, le coordinate X e Y sono impostate a 0, mentre la larghezza e l'altezza sono impostate a 84 e 48.




```
/* Immagine Marilyn Bmp */

#include < SPI.h>
#includere < Adafruit_GFX.h>
#includere <Adafruit_PCD8544.h>

Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);

// 'Marilyn Monroe 84x48', 84x48px
const unsigned char MarilynMonroe [] PROGMEM = {
```

```
        0x00, 0x00, 0x00, 0x7f, 0x00, 0x02, 0xfe, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xbe,
0x00,
        0x00, 0x1f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x3f, 0x80, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0xf0, 0x00, 0x00, 0x1f, 0xe1, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0xc0,
        0x00, 0x00, 0x0f, 0xf1, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0e, 0xd8, 0xe0,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x80, 0x00, 0x07, 0xe0, 0x70, 0x00, 0x00, 0x00, 0x00, 0x00,
0x03,
        0x3f, 0xe0, 0x00, 0x07, 0xf0, 0x78, 0x00, 0x00, 0x00, 0x00, 0x01, 0xe0, 0x70, 0x00, 0x0f, 0xee,
        0x7c, 0x00, 0x00, 0x00, 0x00, 0x03, 0xc0, 0x00, 0x00, 0x0f, 0xf7, 0x1c, 0x00, 0x00, 0x00, 0x00,
0x00,
        0x07, 0x80, 0x00, 0x0f, 0xc7, 0xf3, 0x1e, 0x00, 0x00, 0x00, 0x00, 0x07, 0xc0, 0x00, 0x0f, 0xf3,
        0xdf, 0x7f, 0x80, 0x00, 0x00, 0x00, 0x07, 0xfe, 0x00, 0x08, 0x7d, 0xef, 0xff, 0xc0, 0x00, 0x00,
        0x00, 0x7f, 0xff, 0x80, 0x30, 0x0f, 0xfc, 0xe0, 0xc0, 0x00, 0x00, 0x01, 0x9e, 0x73, 0xc0, 0xe0,
        0x07, 0xf8, 0xc1, 0xc0, 0x00, 0x00, 0x03, 0xfc, 0x00, 0x01, 0xc0, 0x0f, 0xfd, 0xe1, 0x80, 0x00,
        0x00, 0x03, 0xf8, 0x00, 0x01, 0x9c, 0x0f, 0xff, 0xc1, 0xc0, 0x00, 0x00, 0x02, 0xc0, 0x00, 0x01,
        0x9f, 0xbf, 0xfe, 0x01, 0x40, 0x00, 0x00, 0x02, 0x60, 0x00, 0x03, 0x07, 0xef, 0xff, 0x01, 0x40,
        0x00, 0x00, 0x00, 0x60, 0x00, 0x07, 0x01, 0xf7, 0xff, 0x80, 0xc0, 0x00, 0x00, 0x00, 0x50, 0x01,
        0xdf, 0x00, 0x7f, 0xff, 0x1c, 0x80, 0x00, 0x00, 0x00, 0x40, 0x01, 0xff, 0x00, 0x1f, 0xff, 0x1e,
        0xe0, 0x00, 0x00, 0x02, 0x08, 0x00, 0x3f, 0x80, 0x07, 0xef, 0x03, 0xe0, 0x00, 0x00, 0x06, 0x08,
        0x00, 0x03, 0xc0, 0x07, 0xdf, 0x07, 0xc0, 0x00, 0x00, 0x06, 0x08, 0x0f, 0x81, 0x80, 0x1f, 0xdf,
        0x1f, 0x80, 0x00, 0x00, 0x03, 0x08, 0x1f, 0x98, 0x00, 0x3f, 0xfe, 0x19, 0x80, 0x00, 0x00, 0x18,
        0x08, 0x3f, 0xfe, 0x00, 0x7f, 0xfe, 0x3f, 0x00, 0x00, 0x00, 0x08, 0x08, 0x30, 0x3f, 0x00, 0xff,
        0xff, 0x3f, 0x00, 0x00, 0x00, 0x01, 0xe0, 0x76, 0x0f, 0x89, 0xff, 0xff, 0x9f, 0x00, 0x00, 0x00,
0x00,
        0x03, 0xe0, 0x7f, 0xc3, 0x81, 0xff, 0xfe, 0x9f, 0x80, 0x00, 0x00, 0x03, 0xf0, 0x7f, 0xf3, 0xc3,
        0xff, 0xfe, 0x1f, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x7f, 0xfd, 0xc3, 0xff, 0xfe, 0x5e, 0x00, 0x00,
        0x00, 0x03, 0xf0, 0x7f, 0xff, 0xc3, 0xff, 0xf3, 0x1e, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x71, 0xff,
        0x87, 0xff, 0xe3, 0xff, 0x00, 0x00, 0x00, 0x07, 0xf0, 0x7c, 0x3f, 0x87, 0xff, 0xe3, 0xfe, 0x00,
        0x00, 0x00, 0x0f, 0xf0, 0x3c, 0xff, 0x05, 0xff, 0xf3, 0xfc, 0x00, 0x00, 0x00, 0x0f, 0x0f, 0x0f,
0x0f,
        0xfe, 0x09, 0xff, 0xf7, 0xfc, 0x00, 0x00, 0x00, 0x08, 0xf8, 0x01, 0xfc, 0x19, 0xff, 0xff, 0xf8,
        0x00, 0x00, 0x00, 0x0c, 0x78, 0x00, 0x00, 0x00, 0x13, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x0e,
0x78,
        0x00, 0x00, 0x23, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x0e, 0xf8, 0x00, 0x00, 0x47, 0xff, 0xff,
        0xf0, 0x00, 0x00, 0x00, 0x0c, 0xfa, 0x00, 0x01, 0x8f, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x08,
        0x7b, 0x00, 0x03, 0x3f, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x0c, 0x3b, 0xf8, 0x0f, 0xff, 0xff,
        0xff, 0xe0, 0x00, 0x00, 0x0f, 0xbb, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00,
        0x07, 0xfb, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x71, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x41, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00
};

void setup() {
        Serial.begin(9600);

        display.begin();

        display.setContrast(57);

        display.clearDisplay();
```

```
        // Visualizzazione della bitmap
        display.drawBitmap(0, 0, MarilynMonroe, 84, 48, NERO);
        display.display();

        // Invertire il display
        //display.invertDisplay(1);
}

void loop() {}
```

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

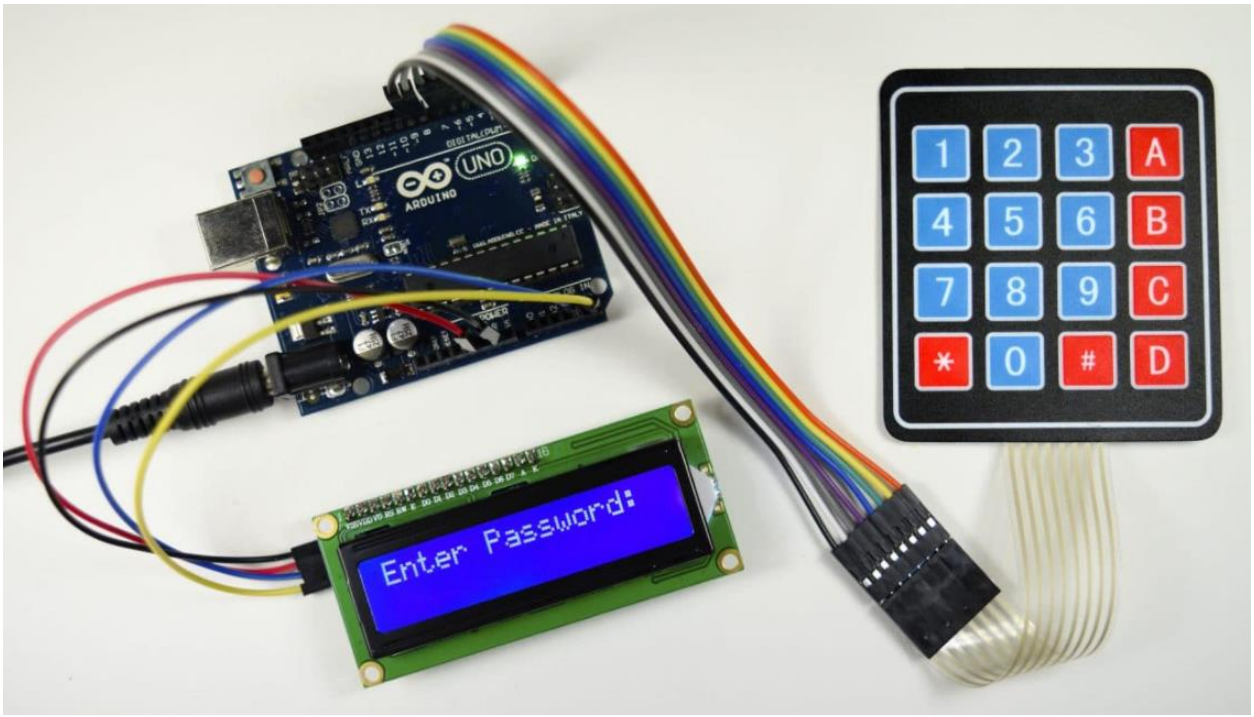
Strategic Partnerships Project for Vocational Education and Training

Titolo Progetto: “Teaching and Learning Arduinos in Vocational Training”

Acronimo Progetto: “ ARDUinVET ”

Progetto N: “2020-1-TR01-KA202-093762”

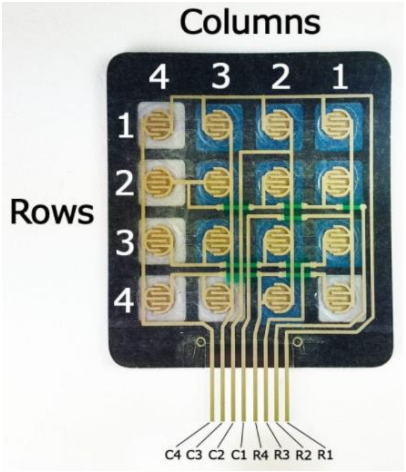
Keypad Module and Training Kit



What a keypad is?

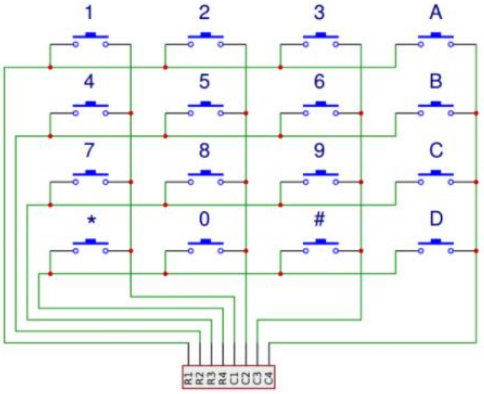
Una tastiera è un sistema di pulsanti disposti in una matrice che funziona come un dispositivo di commutazione per fornire un collegamento tra una linea e una colonna.

Useremo una tastiera a membrana con matrice 4X4, perché è sottile e ha un supporto adesivo, in modo che possa essere incollata sulla maggior parte delle superfici piane.
Sotto ogni tasto c'è un interruttore a membrana. Ogni interruttore in una fila è collegato agli altri interruttori della fila da una traccia conduttiva sotto il pad. Ogni interruttore in una colonna è collegato allo stesso modo - un lato dell'interruttore è collegato a tutti gli altri interruttori in quella colonna da una traccia conduttiva. Ogni fila e colonna è collegata a un singolo pin, per un totale di 8 pin su una tastiera 4X4.

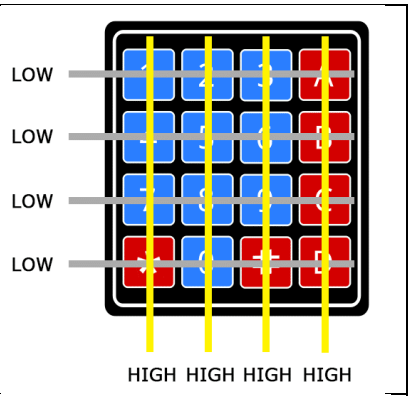
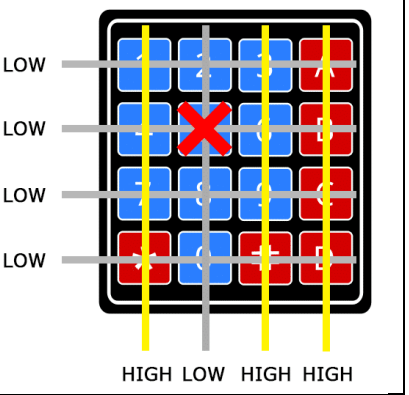
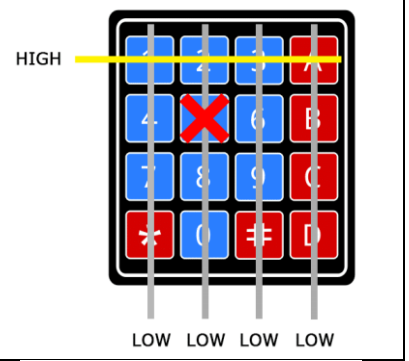
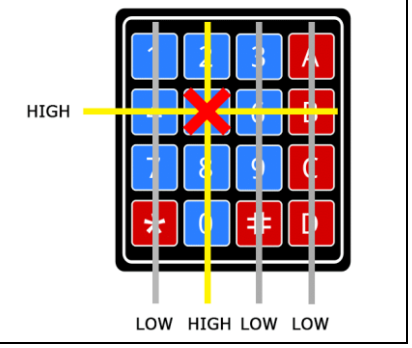


La pressione di un pulsante chiude l'interruttore tra una traccia di colonna e una di fila, permettendo alla corrente di fluire tra un pin di colonna e uno di fila.

Lo schema di una tastiera 4X4 mostra come sono collegate le righe e le colonne:



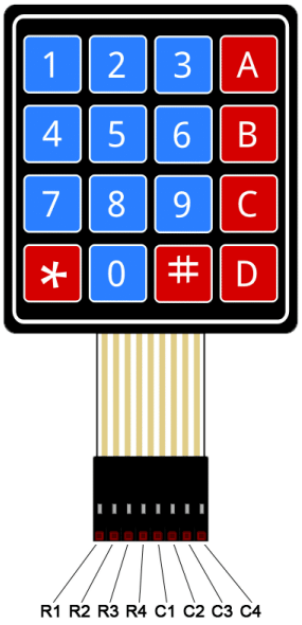
Arduino rileva quale pulsante è stato premuto rilevando il pin di riga e colonna che è collegato al pulsante.
Questo avviene in quattro passi:

1. In primo luogo, quando nessun pulsante è premuto, tutti i pin della colonna sono tenuti ALTO, e tutti i pin della fila sono tenuti BASSO	
2. Quando viene premuto un pulsante, il pin di colonna viene tirato a BASSO poiché la corrente dalla colonna ALTA scorre verso il pin di fila BASSO:	
3. Arduino ora sa in quale colonna si trova il pulsante, quindi ora ha solo bisogno di trovare la fila in cui si trova il pulsante. Lo fa commutando ciascuno dei pin della fila in ALTO, e allo stesso tempo leggendo tutti i pin della colonna per rilevare quale pin della colonna ritorna in ALTO	
4. Quando il pin della colonna va di nuovo ALTO, Arduino ha trovato il pin della fila che è collegato al pulsante:	

Dal diagramma qui sopra, si può vedere che la combinazione della riga 2 e della colonna 2 può significare solo che è stato premuto il pulsante numero 5.

COLLEGARE LA TASTIERA AD ARDUINO

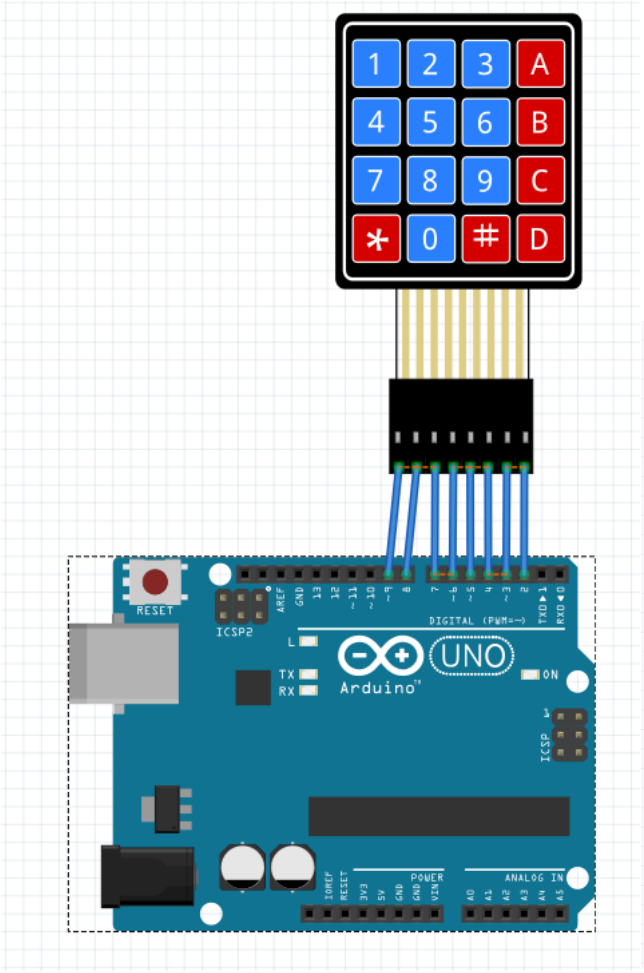
La disposizione dei pin per la maggior parte delle tastiere a membrana sarà simile a questa:


1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D
R1 R2 R3 R4 C1 C2 C3 C4

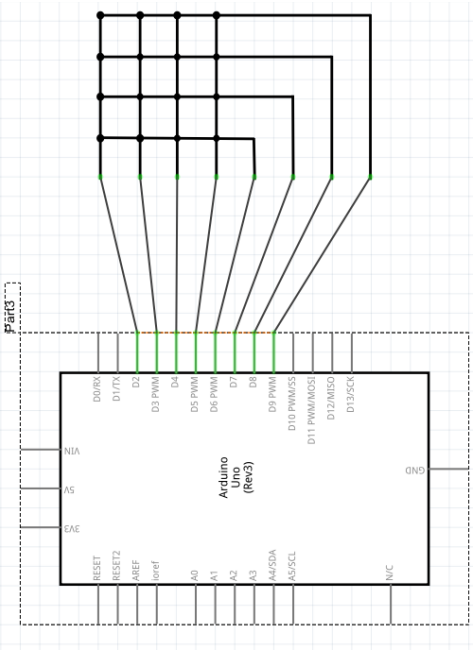
Circuito 1

Titolo del circuito: Monitor seriale visualizzare il tasto premuto

Descrizione del circuito: Il programma ci mostrerà come stampare ogni tasto premuto sul monitor seriale.



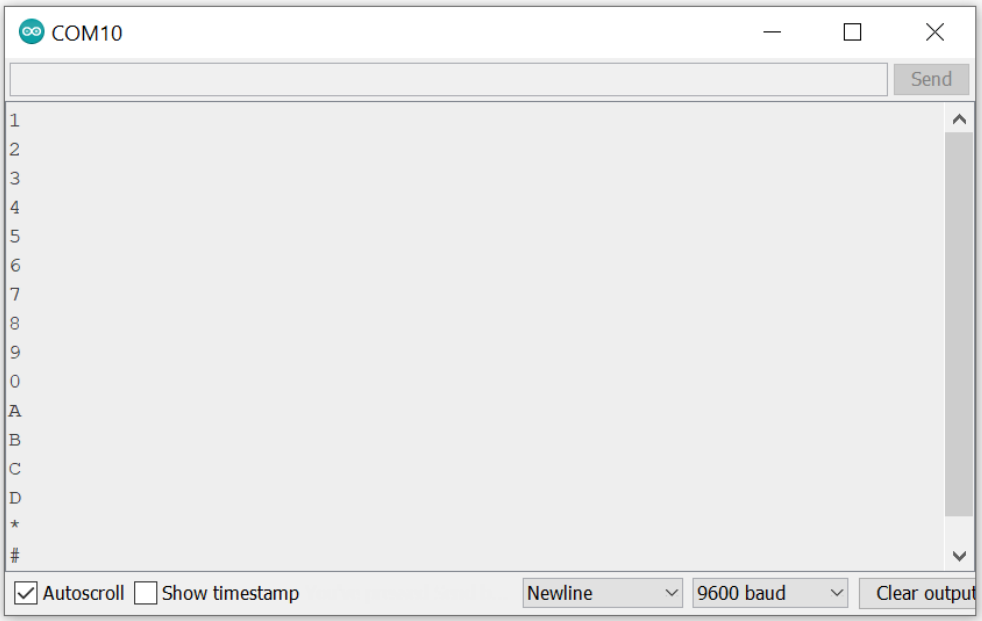
1-) Breadboard view



2-) Schematic view

```
/*     "Serial monitor display the pressed key"    */
#include <Keypad.h>
const byte ROWS = 4;
const byte COLS = 4;
char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
```

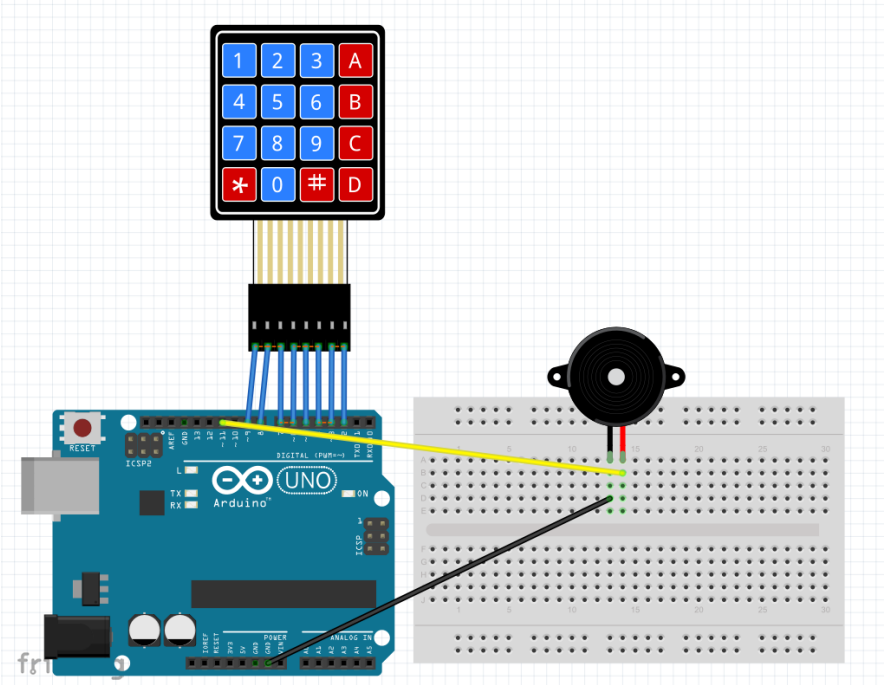
```
void setup(){
  Serial.begin(9600);
}
void loop(){
  char customKey = customKeypad.getKey();
  if (customKey){
    Serial.println(customKey);
  }
}
```



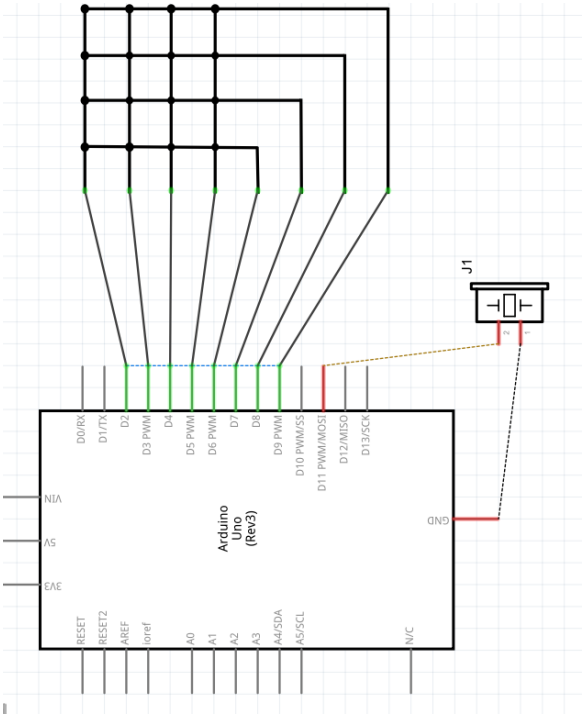
Circuito 2

Titolo del circuito: Arduino Keypad Beep

Descrizione del circuito: Quando un tasto della tastiera viene premuto, il cicalino piezoelettrico emette un segnale acustico



1-) Breadboard view



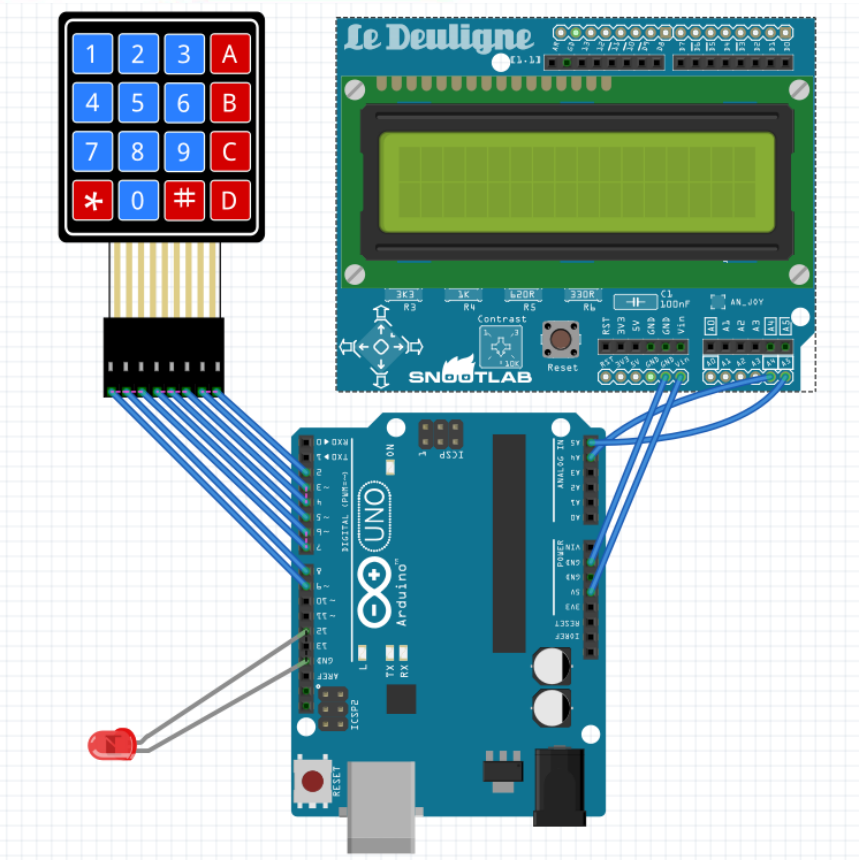
2-) Schematic view

```
/*    "Arduino Keypad Beep"   */
#include <Keypad.h>
#include <ezBuzzer.h>
const int BUZZER_PIN = 11;
const int ROW_NUM    = 4; // four rows
const int COLUMN_NUM = 4; // four columns
char keys[ROW_NUM][COLUMN_NUM] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte pin_rows[ROW_NUM] = {9, 8, 7, 6};      // connect to the row pinouts of the keypad
byte pin_column[COLUMN_NUM] = {5, 4, 3, 2}; // connect to the column pinouts of the keypad
Keypad keypad = Keypad(makeKeymap(keys), pin_rows, pin_column, ROW_NUM,
COLUMN_NUM );
ezBuzzer buzzer(BUZZER_PIN); // create ezBuzzer object that attach to a pin;
void setup() {
  Serial.begin(9600);
}
void loop() {
  buzzer.loop(); // MUST call the buzzer.loop() function in loop()
  char key = keypad.getKey();
  if (key) {
    Serial.print(key); // prints key to serial monitor
    buzzer.beep(100);  // generates a 100ms beep
  }
}
```

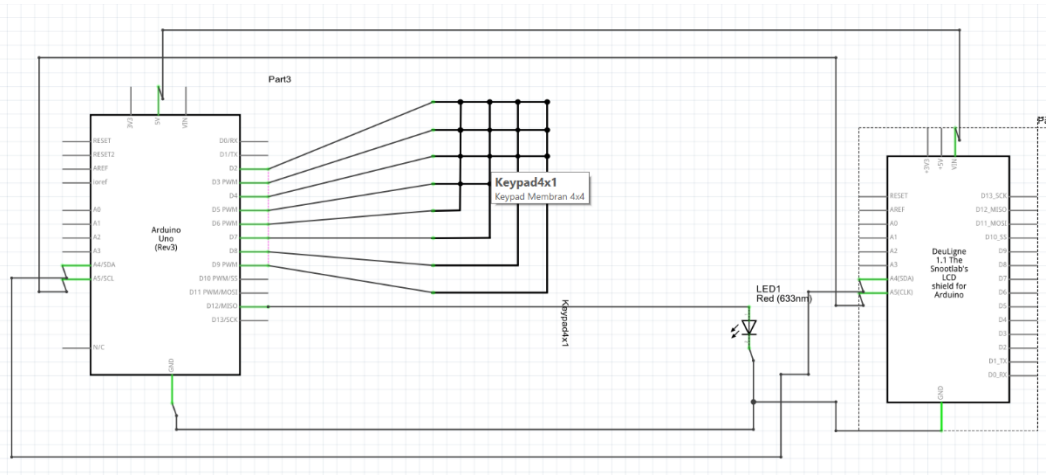
Circuito 2

Titolo del circuito: Arduino Keypad Beep

Descrizione del circuito: Quando un tasto della tastiera viene premuto, il cicalino piezoelettrico emette un segnale acustico



1-) Breadboard view



2-) Schematic view

```
/* Arduino Unlocking code */
#include <Keypad.h> //Libraries you can download them via Arduino IDE
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#define Solenoid 12          //Actually the Gate of the transistor that controls the solenoid
                             //in my case I use a simple LED
#include <liquidcrystal_i2c.h></liquidcrystal_i2c.h></lcd.h></wire.h></keypad.h>
```

```
#define I2C_ADDR 0x27 // LCD i2c Adress and pins
#define BACKLIGHT_PIN 3
#define En_pin 2
#define Rw_pin 1
#define Rs_pin 0
#define D4_pin 4
#define D5_pin 5
#define D6_pin 6
#define D7_pin 7
LiquidCrystal_I2C lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);
const byte numRows= 4; //number of rows on the keypad
const byte numCols= 4; //number of columns on the keypad
int code = 1234;  //here is the code
int tot,i1,i2,i3,i4;
char c1,c2,c3,c4;
//keymap defines the key pressed according to the row and columns just as appears on the
keypad char keymap[numRows][numCols]=
{
{'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'}
};
//Code that shows the keypad connections to the arduino terminals
byte rowPins[numRows] = {9,8,7,6}; //Rows 0 to 3
byte colPins[numCols]= {5,4,3,2}; //Columns 0 to 3
//initializes an instance of the Keypad class
Keypad myKeypad= Keypad(makeKeymap(keymap), rowPins, colPins, numRows, numCols);
void setup()
        {
         lcd.begin (16,2);
         lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
         lcd.setBacklight(HIGH);
         lcd.home ();
         lcd.print("ROMANIA DoorLock");
         lcd.setCursor(9, 1);
         lcd.print("Standby");
          pinMode(Solenoid,OUTPUT);
         delay(2000);
        }
void loop()
{
    char keypressed = myKeypad.getKey();  //The getKey fucntion keeps the program runing, as
long you didn't press "*" the whole thing bellow wouldn't be triggered
      if (keypressed == '*')          // and you can use the rest of you're code simply
         {
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Enter Code");              //when the "*" key is pressed you can enter the
passcode
```

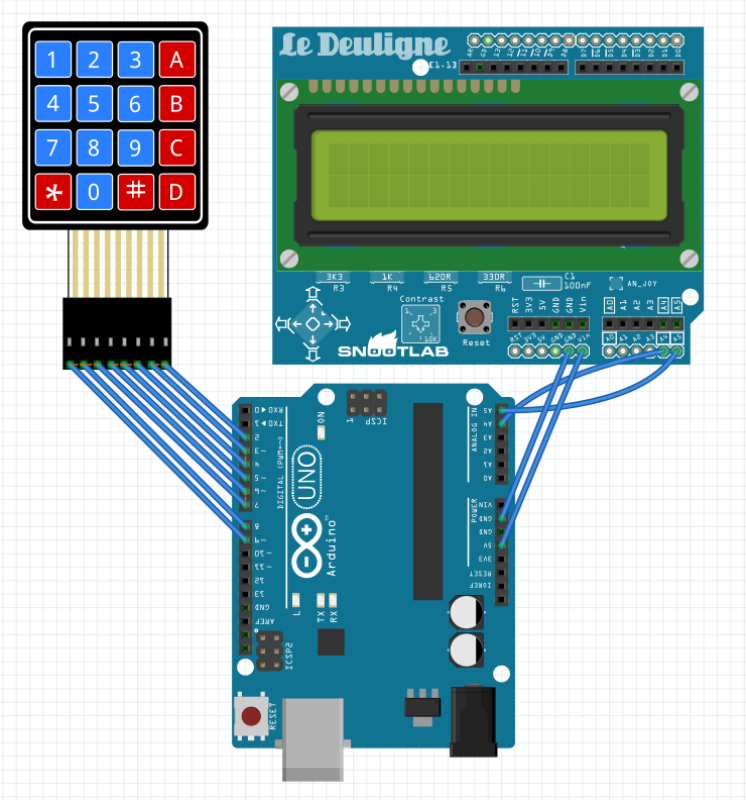
```
            keypressed = myKeypad.waitForKey();  // here all programs are stopped until you
enter the four digits then it gets compared to the code above
            if (keypressed != NO_KEY)
              {
               c1 = keypressed;
               lcd.setCursor(0, 1);
               lcd.print("*");
               }
            keypressed = myKeypad.waitForKey();
            if (keypressed != NO_KEY)
              {
               c2 = keypressed;
               lcd.setCursor(1, 1);
               lcd.print("*");
               }
             keypressed = myKeypad.waitForKey();
           if (keypressed != NO_KEY)
              {
               c3 = keypressed;
               lcd.setCursor(2, 1);
               lcd.print("*");
               }
              keypressed = myKeypad.waitForKey();
           if (keypressed != NO_KEY)
              {
               c4 = keypressed;
               lcd.setCursor(3, 1);
               lcd.print("*");
               }
             i1=(c1-48)*1000;       //the keys pressed are stored into chars I convert them to int
then i did some multiplication to get the code as an int of xxxx
              i2=(c2-48)*100;
              i3=(c3-48)*10;
              i4=c4-48;
              tot=i1+i2+i3+i4;
          if (tot == code) //if the code is correct you trigger whatever you want here it just print a
message on the screen
       {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Welcome");
        digitalWrite(Solenoid,HIGH);
  delay(3000);
  digitalWrite(Solenoid,LOW);
        lcd.setCursor(7, 1);
        lcd.print("ROMANIA DoorLock");

        delay(3000);
        lcd.clear();
        lcd.print("ROMANIA DoorLock");
```

```
        lcd.setCursor(9, 1);
        lcd.print("Standby");

      }
      else //if the code is wrong you get another thing
      {
       lcd.clear();
       lcd.setCursor(0, 0);
       lcd.print("WRONG CODE");
       delay(3000);
       lcd.clear();
       lcd.print("ROMANIA DoorLock");
       lcd.setCursor(9, 1);
       lcd.print("Standby");
       }
         }
}
```

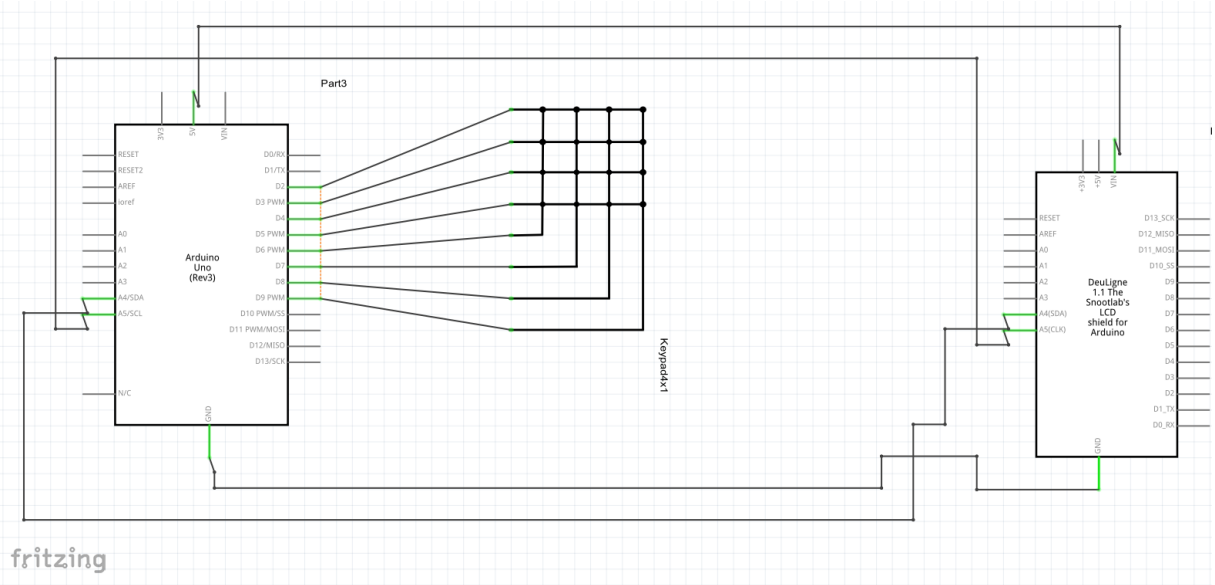
Circuito 4

Titolo del circuito: Calcolatrice Arduino

Descrizione del circuito: Il programma fa calcoli matematici di base



1-) Breadboard view



2-) Schematic view

/* Arduino calculator */
#include <Keypad.h>

```
#include <EEPROM.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#define I2C_ADDR 0x27          //I2C adress
#define BACKLIGHT_PIN 3        // Declaring LCD Pins
#define En_pin 2
#define Rw_pin 1
#define Rs_pin 0
#define D4_pin 4
#define D5_pin 5
#define D6_pin 6
#define D7_pin 7
const byte ROWS = 4; // Four rows
const byte COLS = 4; // Four columns
// Define the Keymap
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {9,8,7,6}; //Rows 0 to 3
byte colPins[COLS]= {5,4,3,2}; //Columns 0 to 3
LiquidCrystal_I2C lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);
Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS ); //  Create the
Keypad
long Num1,Num2,Number;
 char key,action;
 boolean result = false;
 void setup()
        {
         lcd.begin (16,2);
         lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
           lcd.setBacklight(HIGH); //Lighting backlight
        lcd.print("Calculator Ready"); //Display a intro message
  lcd.setCursor(0, 1);   // set the cursor to column 0, line 1
  lcd.print("A=+ B=- C=* D=/"); //Display a intro message
   delay(6000); //Wait for display to show info
    lcd.clear(); //Then clean it
                  //         for(i=0 ; i<sizeof(code);i++){       //When you upload the code the first time
keep it commented
//           EEPROM.get(i, code[i]);           //Upload the code and change it to store it in the
EEPROM
//           }       //Then uncomment this for loop and reupload the code (It's done only once)
        }
void loop() {
  key = kpd.getKey(); //storing pressed key value in a char
if (key!=NO_KEY)
DetectButtons();
if (result==true)
```

```
CalculateResult();
DisplayResult();
}
void DetectButtons()
{
    lcd.clear(); //Then clean it
   if (key=='*') //If cancel Button is pressed
   {Serial.println ("Button Cancel"); Number=Num1=Num2=0; result=false;}
       if (key == '1') //If Button 1 is pressed
   {Serial.println ("Button 1");
   if (Number==0)
   Number=1;
   else
   Number = (Number*10) + 1; //Pressed twice
   }
       if (key == '4') //If Button 4 is pressed
   {Serial.println ("Button 4");
   if (Number==0)
   Number=4;
   else
   Number = (Number*10) + 4; //Pressed twice
   }
       if (key == '7') //If Button 7 is pressed
   {Serial.println ("Button 7");
   if (Number==0)
   Number=7;
   else
   Number = (Number*10) + 7; //Pressed twice
   }
     if (key == '0')
   {Serial.println ("Button 0"); //Button 0 is Pressed
   if (Number==0)
   Number=0;
   else
   Number = (Number*10) + 0; //Pressed twice
   }
       if (key == '2') //Button 2 is Pressed
   {Serial.println ("Button 2");
    if (Number==0)
   Number=2;
   else
   Number = (Number*10) + 2; //Pressed twice
   }
       if (key == '5')
   {Serial.println ("Button 5");
    if (Number==0)
   Number=5;
   else
   Number = (Number*10) + 5; //Pressed twice
   }
```

```
      if (key == '8')
  {Serial.println ("Button 8");
   if (Number==0)
  Number=8;
  else
  Number = (Number*10) + 8; //Pressed twice
  }
    if (key == '#')
  {Serial.println ("Button Equal");
  Num2=Number;
  result = true;
  }
      if (key == '3')
  {Serial.println ("Button 3");
   if (Number==0)
  Number=3;
  else
  Number = (Number*10) + 3; //Pressed twice
  }
      if (key == '6')
  {Serial.println ("Button 6");
  if (Number==0)
  Number=6;
  else
  Number = (Number*10) + 6; //Pressed twice
  }
      if (key == '9')
  {Serial.println ("Button 9");
  if (Number==0)
  Number=9;
  else
  Number = (Number*10) + 9; //Pressed twice
  }
    if (key == 'A' || key == 'B' || key == 'C' || key == 'D') //Detecting Buttons on Column 4
 {
  Num1 = Number;
  Number =0;
  if (key == 'A')
  {Serial.println ("Addition"); action = '+';}
   if (key == 'B')
  {Serial.println ("Subtraction"); action = '-'; }
   if (key == 'C')
  {Serial.println ("Multiplication"); action = '*';}
   if (key == 'D')
  {Serial.println ("Division"); action = '/';}
  delay(100);
 }
}
void CalculateResult()
{
```

```
  if (action=='+')
    Number = Num1+Num2;
  if (action=='-')
    Number = Num1-Num2;
  if (action=='*')
    Number = Num1*Num2;
  if (action=='/')
    Number = Num1/Num2;
}
void DisplayResult()
{
  lcd.setCursor(0, 0);   // set the cursor to column 0, line 1
  lcd.print(Num1); lcd.print(action); lcd.print(Num2);
    if (result==true)
  {lcd.print(" ="); lcd.print(Number);} //Display the result
    lcd.setCursor(0, 1);   // set the cursor to column 0, line 1
  lcd.print(Number); //Display the result
}
```

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Titolo Progetto: "Teaching and Learning Arduinos in Vocational Training"

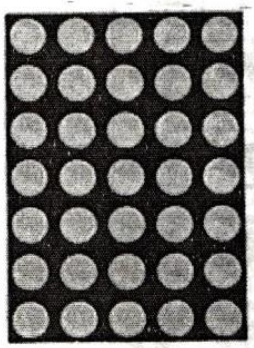
Acronimo Progetto: " ARDUinVET "

Progetto N: "2020-1-TR01-KA202-093762"

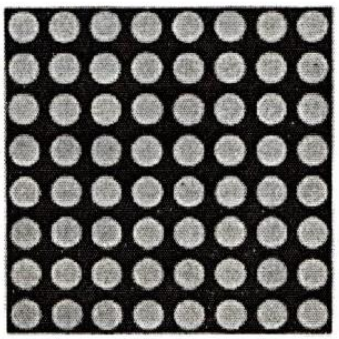
Dot Matrix Module and Training Kit

Cos'è un display Dot Matrix?

Il display a matrice di punti è un gruppo di LED disposti in un sistema specifico. Una matrice LED standard è composta da 5x7 o 8x8 LED disposti in righe e colonne come mostrato nella figura qui sotto. Una matrice 5*7 DOT ha 5 righe e 7 colonne di LED e una matrice 8*8 DOT ha 8 righe e 8 colonne di LED che sono collegate tra loro.

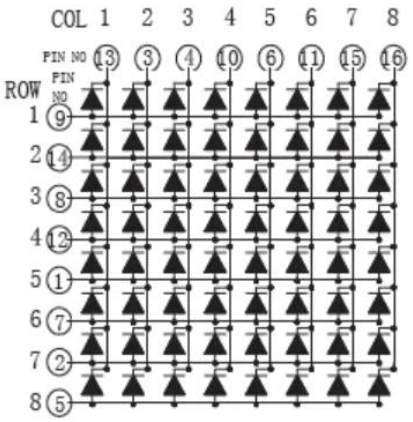


5*7 DOT Matrix Display

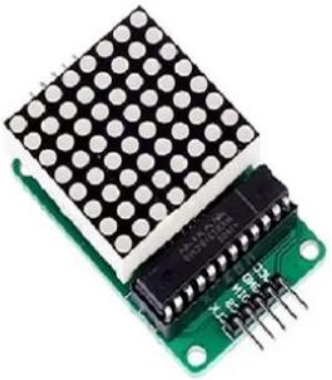


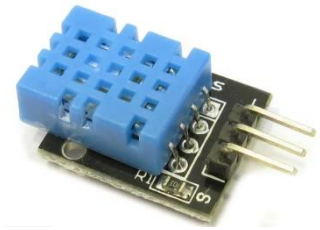
8*8 DOT Matrix Display

Se guardiamo un pezzo della matrice di punti 8x8, contiene 16 pin in cui 8 pin usati per le righe e 8 per le colonne. Questo significa che nelle righe e nella colonna un totale di 64 LED. Iniziamo dal pin numero 1 al pin numero 8. Il pin numero 1 è R5 (Row-5) e il pin numero 8 è R3 (Row-3) nella parte inferiore.
Al lato superiore Da Pin 9 (Row-1) a Pin 16 (column-1) situato. Ma un principiante confonde sempre e inizia da zero, perché sappiamo l'immagine/diagramma. spesso otteniamo da qualche fonte, anche noi dobbiamo risolvere quale +VE e -VE. potrebbe essere un esperto può capire dal tipo comune catodo/anodo.



Il display a matrice di punti MAX7219 LED è uno di quelli popolari disponibili sul mercato e utilizzato da studenti, hobbisti di elettronica e soprattutto in applicazioni di visualizzazione industriali. Ci sono due tipi di moduli generalmente disponibili. Questi sono il modulo generico.
Ogni modulo consiste di due unità. Una è la matrice di punti 8X8 LED e l'altra è il circuito integrato MAX7219.
MAX7219 è un IC driver di display a catodo comune con ingressi e uscite seriali. Ha una capacità di corrente regolabile

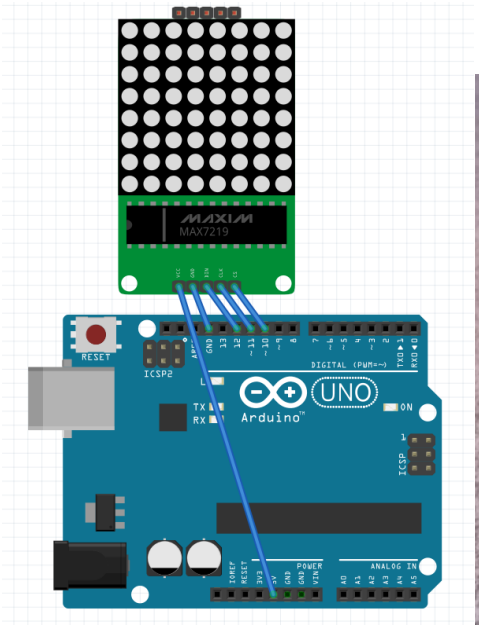
che può essere impostata utilizzando solo una resistenza esterna. Inoltre, ha un'interfaccia seriale a quattro fili che può essere facilmente collegata a tutti i microprocessori. Può pilotare 64 LED individuali collegati ai suoi pin di uscita usando solo 4 fili usando Arduino. Inoltre, può pilotare display a matrice di punti, display a 7 segmenti e grafici a barre. Inoltre, MAX7219 ha un decoder BCD integrato che lo rende facile da usare con display numerici a sette segmenti. Inoltre, ha una RAM statica 8×8 che possiamo usare per memorizzare i numeri. È uno dei più popolari IC di driver di visualizzazione.	

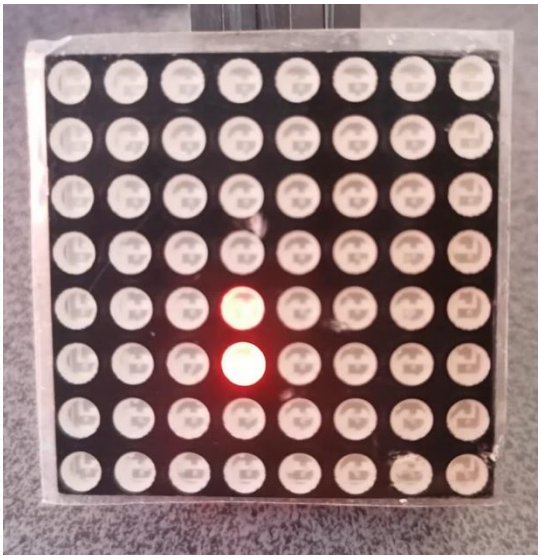
	Il DHT11 è un sensore digitale di temperatura e umidità. Utilizza un sensore di umidità capacitivo e un termistore per misurare l'aria circostante e sputa fuori un segnale digitale sul pin dati (nessun pin di ingresso analogico necessario).

Circuito 1:

Titolo del circuito: Visualizzare tutti i LED uno per uno

Descrizione del circuito: Il Dot Matrix visualizzare tutti i LED uno per uno





1.) Breadboard view

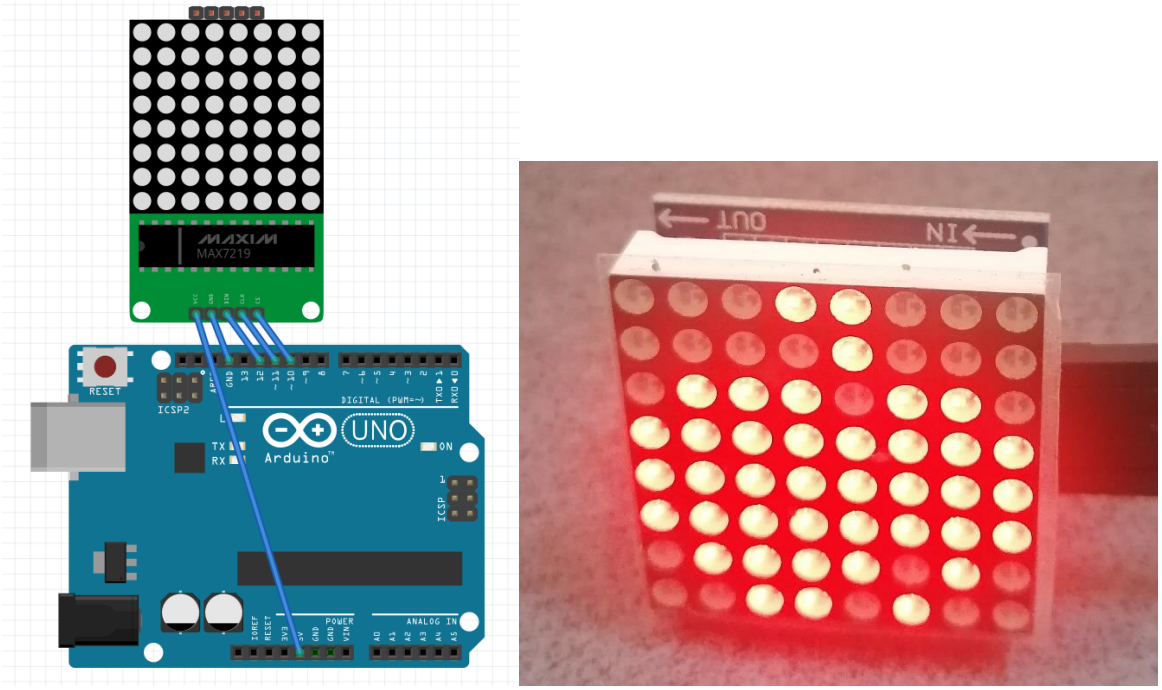
/* Display all LEDs one by one */

```
#include <LedControl.h>
int DIN = 12;
int CS = 10;
int CLK = 11;
LedControl lc=LedControl(DIN, CLK, CS,0);
void setup() {
  lc.shutdown(0,false);
  lc.setIntensity(0,0);
  lc.clearDisplay(0);}
void loop() {
  for(int j=0;j<8;j++){
     for(int i=0;i<8;i++){
       lc.setLed(0,j,i,true);
       delay(100);
       lc.setLed(0,j,i,false); } }}
```

Circuito 2:

Titolo del circuito: visualizzare l'icona della mela

Descrizione del circuito: L'interfaccia Dot Matrix visualizza l'icona Apple



1-) Breadboard view

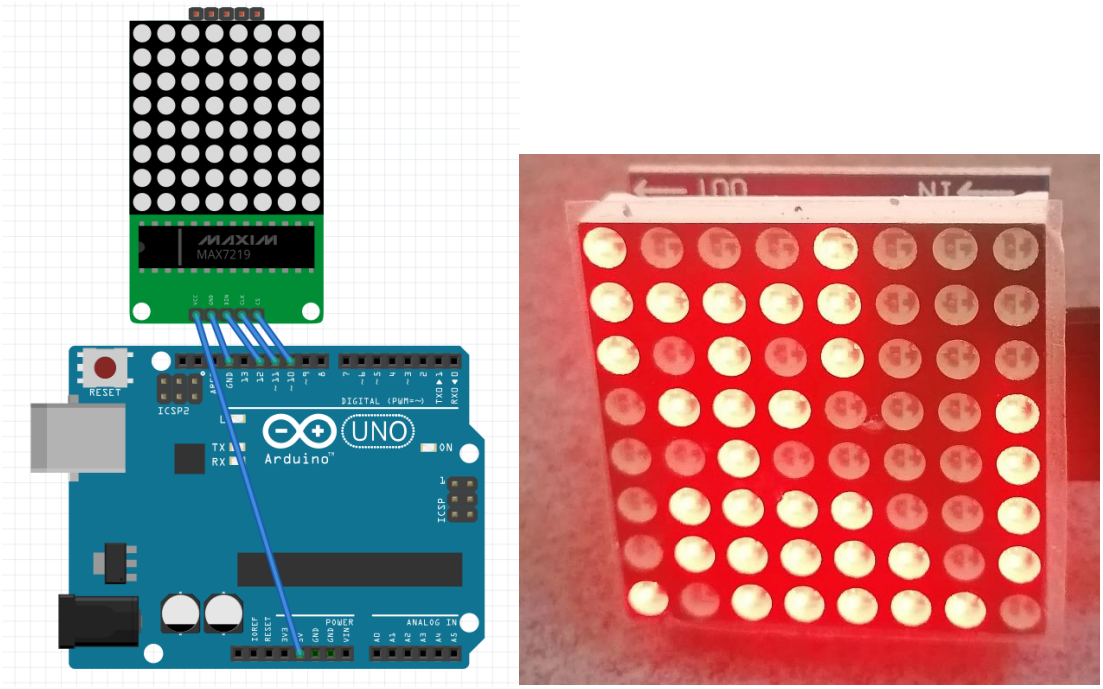
```
/*  Display the Apple icon*/
#include <LedControl.h>
int DIN = 12;
int CS = 10;
int CLK = 11;
LedControl lc=LedControl(DIN, CLK, CS,0);
```

```
byte Apple
[8]={B00011000,B00001000,B01110110,B11111111,B11111111,B11111111,B01111010,B0011010
0};
void setup() {
  lc.shutdown(0,false);
  lc.setIntensity(0,0);
  lc.clearDisplay(0);   }
  void loop(){
  for(int i=0;i<8;i++) lc.setRow(0,i,Apple[i]);   }
```

Circuito 3:

Titolo del circuito: Visualizzare l'icona di un gatto

Descrizione del circuito: L'interfaccia a matrice di punti visualizza un'icona di gatto



1-) Breadboard view

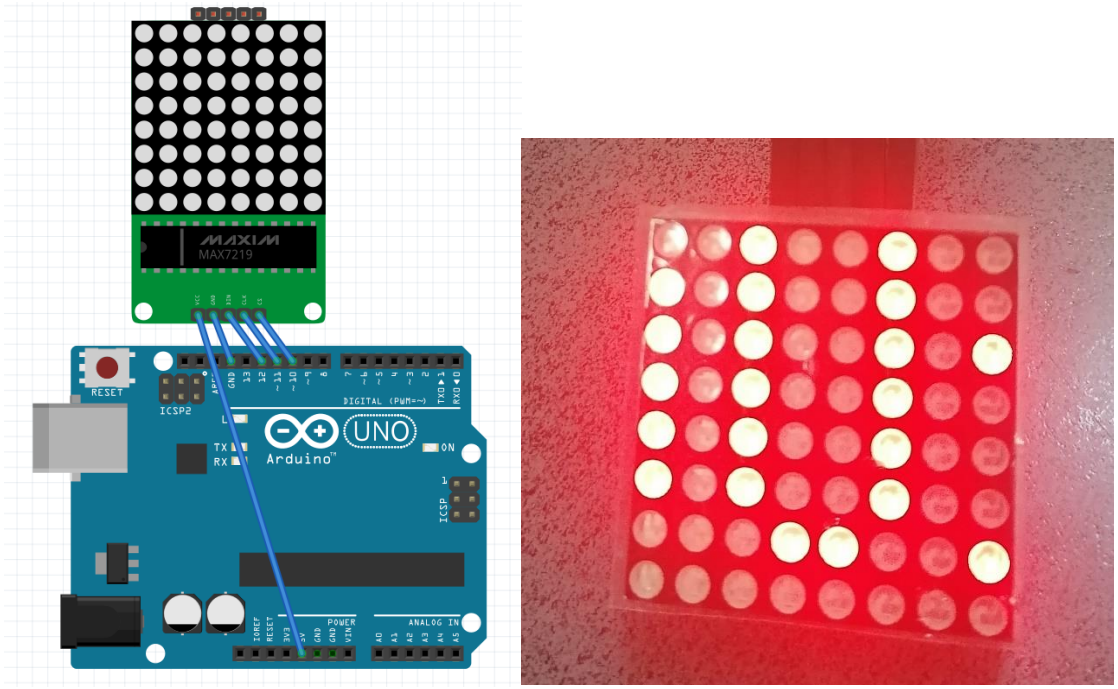
```
/*  Display a cat icon*/
#include <LedControl.h>
int DIN = 12;
int CS = 10;
int CLK = 11;
LedControl lc=LedControl(DIN, CLK, CS,0);
int Cat[8]
={B10001000,B11111000,B10101000,B01110001,B00100001,B01111001,B01111101,B10111110
};
void setup() {
  lc.shutdown(0,false);
  lc.setIntensity(0,0);
  lc.clearDisplay(0);   }
  void loop(){
```

for(int i=0;i<8;i++) lc.setRow(0,i,Cat[i]); }

Circuito 4:

Titolo del circuito: Visualizzazione del testo scorrevole ARDUINVET

Descrizione del circuito: L'interfaccia Dot Matrix visualizza il testo scorrevole ARDUINVET



1-) Breadboard view

```
/*  Display flowing text ARDUINVET*/
//D10=CS
//D11=CLK
//D12=DIN
#include <MaxMatrix.h> //include matrix library
#include <avr/pgmspace.h>
#include <stdlib.h>
PROGMEM const unsigned char CH[] = {
3, 8, B00000000, B00000000, B00000000, B00000000, B00000000, // space
1, 8, B01011111, B00000000, B00000000, B00000000, B00000000, // !
3, 8, B00000011, B00000000, B00000011, B00000000, B00000000, // "
5, 8, B00010100, B00111110, B00010100, B00111110, B00010100, // #
4, 8, B00100100, B01101010, B00101011, B00010010, B00000000, // $
5, 8, B01100011, B00010011, B00001000, B01100100, B01100011, // %
5, 8, B00110110, B01001001, B01010110, B00100000, B01010000, // &
1, 8, B00000011, B00000000, B00000000, B00000000, B00000000, // '
3, 8, B00011100, B00100010, B01000001, B00000000, B00000000, // (
3, 8, B01000001, B00100010, B00011100, B00000000, B00000000, // )
5, 8, B00101000, B00011000, B00001110, B00011000, B00101000, // *
5, 8, B00001000, B00001000, B00111110, B00001000, B00001000, // +
2, 8, B10110000, B01110000, B00000000, B00000000, B00000000, // ,
4, 8, B00001000, B00001000, B00001000, B00001000, B00000000, // -
```

```
2, 8, B01100000, B01100000, B00000000, B00000000, B00000000, // .
4, 8, B01100000, B00011000, B00000110, B00000001, B00000000, // /
4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // 0
3, 8, B01000010, B01111111, B01000000, B00000000, B00000000, // 1
4, 8, B01100010, B01010001, B01001001, B01000110, B00000000, // 2
4, 8, B00100010, B01000001, B01001001, B00110110, B00000000, // 3
4, 8, B00011000, B00010100, B00010010, B01111111, B00000000, // 4
4, 8, B00100111, B01000101, B01000101, B00111001, B00000000, // 5
4, 8, B00111110, B01001001, B01001001, B00110000, B00000000, // 6
4, 8, B01100001, B00010001, B00001001, B00000111, B00000000, // 7
4, 8, B00110110, B01001001, B01001001, B00110110, B00000000, // 8
4, 8, B00000110, B01001001, B01001001, B00111110, B00000000, // 9
2, 8, B01010000, B00000000, B00000000, B00000000, B00000000, // :
2, 8, B10000000, B01010000, B00000000, B00000000, B00000000, // ;
3, 8, B00010000, B00101000, B01000100, B00000000, B00000000, // <
3, 8, B00010100, B00010100, B00010100, B00000000, B00000000, // =
3, 8, B01000100, B00101000, B00010000, B00000000, B00000000, // >
4, 8, B00000010, B01011001, B00001001, B00000110, B00000000, // ?
5, 8, B00111110, B01001001, B01010101, B01011101, B00001110, // @
4, 8, B01111110, B00010001, B00010001, B01111110, B00000000, // A
4, 8, B01111111, B01001001, B01001001, B00110110, B00000000, // B
4, 8, B00111110, B01000001, B01000001, B00100010, B00000000, // C
4, 8, B01111111, B01000001, B01000001, B00111110, B00000000, // D
4, 8, B01111111, B01001001, B01001001, B01000001, B00000000, // E
4, 8, B01111111, B00001001, B00001001, B00000001, B00000000, // F
4, 8, B00111110, B01000001, B01001001, B01111010, B00000000, // G
4, 8, B01111111, B00001000, B00001000, B01111111, B00000000, // H
3, 8, B01000001, B01111111, B01000001, B00000000, B00000000, // I
4, 8, B00110000, B01000000, B01000001, B00111111, B00000000, // J
4, 8, B01111111, B00001000, B00010100, B01100011, B00000000, // K
4, 8, B01111111, B01000000, B01000000, B01000000, B00000000, // L
5, 8, B01111111, B00000010, B00001100, B00000010, B01111111, // M
5, 8, B01111111, B00000100, B00001000, B00010000, B01111111, // N
4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // O
4, 8, B01111111, B00001001, B00001001, B00000110, B00000000, // P
4, 8, B00111110, B01000001, B01000001, B10111110, B00000000, // Q
4, 8, B01111111, B00001001, B00001001, B01110110, B00000000, // R
4, 8, B01000110, B01001001, B01001001, B00110010, B00000000, // S
5, 8, B00000001, B00000001, B01111111, B00000001, B00000001, // T
4, 8, B00111111, B01000000, B01000000, B00111111, B00000000, // U
5, 8, B00001111, B00110000, B01000000, B00110000, B00001111, // V
5, 8, B00111111, B01000000, B00111000, B01000000, B00111111, // W
5, 8, B01100011, B00010100, B00001000, B00010100, B01100011, // X
5, 8, B00000111, B00001000, B01110000, B00001000, B00000111, // Y
4, 8, B01100001, B01010001, B01001001, B01000111, B00000000, // Z
2, 8, B01111111, B01000001, B00000000, B00000000, B00000000, // [
4, 8, B00000001, B00000110, B00011000, B01100000, B00000000, // \ backsplash
2, 8, B01000001, B01111111, B00000000, B00000000, B00000000, // ]
3, 8, B00000010, B00000001, B00000010, B00000000, B00000000, // hat
4, 8, B01000000, B01000000, B01000000, B01000000, B00000000, // _
```

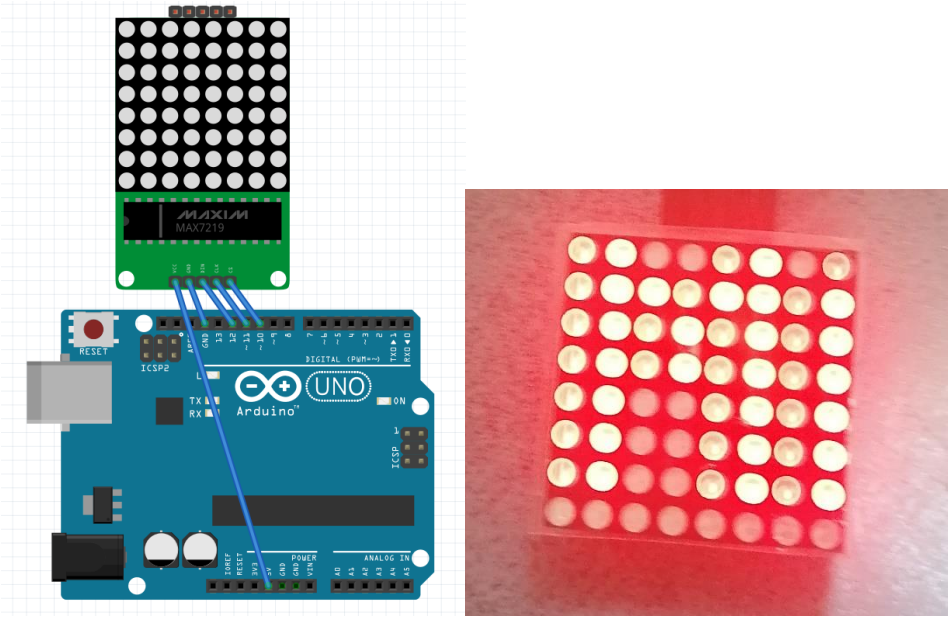
```
2, 8, B00000001, B00000010, B00000000, B00000000, B00000000, // `
4, 8, B00100000, B01010100, B01010100, B01111000, B00000000, // a
4, 8, B01111111, B01000100, B01000100, B00111000, B00000000, // b
4, 8, B00111000, B01000100, B01000100, B00101000, B00000000, // c
4, 8, B00111000, B01000100, B01000100, B01111111, B00000000, // d
4, 8, B00111000, B01010100, B01010100, B00011000, B00000000, // e
3, 8, B00000100, B01111110, B00000101, B00000000, B00000000, // f
4, 8, B10011000, B10100100, B10100100, B01111000, B00000000, // g
4, 8, B01111111, B00000100, B00000100, B01111000, B00000000, // h
3, 8, B01000100, B01111101, B01000000, B00000000, B00000000, // i
4, 8, B01000000, B10000000, B10000100, B01111101, B00000000, // j
4, 8, B01111111, B00010000, B00101000, B01000100, B00000000, // k
3, 8, B01000001, B01111111, B01000000, B00000000, B00000000, // l
5, 8, B01111100, B00000100, B01111100, B00000100, B01111000, // m
4, 8, B01111100, B00000100, B00000100, B01111000, B00000000, // n
4, 8, B00111000, B01000100, B01000100, B00111000, B00000000, // o
4, 8, B11111100, B00100100, B00100100, B00011000, B00000000, // p
4, 8, B00011000, B00100100, B00100100, B11111100, B00000000, // q
4, 8, B01111100, B00001000, B00000100, B00000100, B00000000, // r
4, 8, B01001000, B01010100, B01010100, B00100100, B00000000, // s
3, 8, B00000100, B00111111, B01000100, B00000000, B00000000, // t
4, 8, B00111100, B01000000, B01000000, B01111100, B00000000, // u
5, 8, B00011100, B00100000, B01000000, B00100000, B00011100, // v
5, 8, B00111100, B01000000, B00111100, B01000000, B00111100, // w
5, 8, B01000100, B00101000, B00010000, B00101000, B01000100, // x
4, 8, B10011100, B10100000, B10100000, B01111100, B00000000, // y
3, 8, B01100100, B01010100, B01001100, B00000000, B00000000, // z
3, 8, B00001000, B00110110, B01000001, B00000000, B00000000, // {
1, 8, B01111111, B00000000, B00000000, B00000000, B00000000, // |
3, 8, B01000001, B00110110, B00001000, B00000000, B00000000, // }
4, 8, B00001000, B00000100, B00001000, B00000100, B00000000, // ~
};
int data = 12;    // DIN pin of MAX7219 module
int load = 10;    // CS pin of MAX7219 module
int clock = 11;  // CLK pin of MAX7219 module
int maxInUse = 1;    //change this variable to set how many MAX7219's you'll use
MaxMatrix m(data, load, clock, maxInUse); // define module
byte buffer[10];
void setup(){
  m.init(); // module initialize
  m.setIntensity(2); // dot matix intensity 0-15
  Serial.begin(9600); // serial communication initialize
  Serial.println("ARDUinVET");  }
void loop(){
     printStringWithShift("ARDUinVET   ", 100);
   m.shiftLeft(false, true);  }
void printCharWithShift(char c, int shift_speed){
  if (c < 32) return;
  c -= 32;
  memcpy_P(buffer, CH + 7*c, 7);
```

```
  m.writeSprite(32, 0, buffer);
  m.setColumn(32 + buffer[0], 0);
  for (int i=0; i<buffer[0]+1; i++)
  {    delay(shift_speed);
    m.shiftLeft(false, false);  }
}
void printStringWithShift(char* s, int shift_speed){
  while (*s != 0){
    printCharWithShift(*s, shift_speed);
    s++;  }
}
void printString(char* s)
{  int col = 0;
  while (*s != 0)
  {    if (*s < 32) continue;
    char c = *s - 32;
    memcpy_P(buffer, CH + 7*c, 7);
    m.writeSprite(col, 0, buffer);
    m.setColumn(col + buffer[0], 0);
    col += buffer[0] + 1;
    s++;  }        }
```

Circuito 5:

Titolo del circuito: Visualizzare ogni lettera dell'alfabeto

Descrizione del circuito: L'interfaccia Dot Matrix visualizza ogni lettera dell'alfabeto dopo 1 secondo



1-) Breadboard view

/* Display every letter of alphabet*/

```
#include <MaxMatrix.h>//download from https://code.google.com/archive/p/arudino-maxmatrix-library/downloads
int DIN = 12;   // DIN pin of MAX7219 module
int CLK = 11;   // CLK pin of MAX7219 module
int CS = 10;    // CS pin of MAX7219 module
int maxInUse = 1;
MaxMatrix m(DIN, CS, CLK, maxInUse);
char A[] = {8, 8,
B00111000,B01000100,B01000100,B01000100,B01111100,B01000100,B01000100,B01000100};
char B[] = {8, 8,
B01111000,B01000100,B01000100,B01111000,B01000100,B01000100,B01111000,B00000000};
char c[] = {8, 8,
B00000000,B00111100,B01000000,B01000000,B01000000,B01000000,B00111100,B00000000};
char d[] = {8, 8,
B00000000,B01111000,B01000100,B01000100,B01000100,B01000100,B01111000,B00000000};
char e[] = {8, 8,
B00000000,B01111100,B01000000,B01000000,B01111100,B01000000,B01000000,B01111100};
char f[] = {8, 8,
B00000000,B01111100,B01000000,B01000000,B01111100,B01000000,B01000000,B01000000};
char g[] = {8, 8,
B00000000,B00111100,B01000000,B01000000,B01000000,B01001110,B01000010,B00111110};
char h[] = {8, 8,
B00000000,B01000100,B01000100,B01111100,B01000100,B01000100,B01000100,B00000000};
char i[] = {8, 8,
B00000000,B01111100,B00010000,B00010000,B00010000,B00010000,B01111100,B00000000};
char j[] = {8, 8,
B00000000,B00000100,B00000100,B00000100,B00000100,B01000100,B00111000,B00000000};
char k[] = {8, 8,
B00000000,B00100100,B00101000,B00110000,B00101000,B00100100,B00100010,B00000000};
char l[] = {8, 8,
B00000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01111100,B00000000};
char n[] = {8, 8,
B00000000,B01000010,B01100010,B01010010,B01001010,B01000110,B01000010,B00000000};
char o[] = {8, 8,
B00111100,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B00111100};
char p[] = {8, 8,
B00000000,B00111000,B00100100,B00100100,B00111000,B00100000,B00100000,B00000000};
char q[] = {8, 8,
B00111100,B01000010,B01000010,B01000010,B01001010,B01000110,B00111110,B00000001};
char r[] = {8, 8,
B01111000,B01000100,B01000100,B01111000,B01100000,B01010000,B01001000,B01000100};
char s[] = {8, 8,
B00111100,B01000000,B01000000,B00111000,B00000100,B00000100,B01111000,B00000000};
char t[] = {8, 8,
B00000000,B01111100,B00010000,B00010000,B00010000,B00010000,B00010000,B00000000};
char u[] = {8, 8,
B00000000,B01000010,B01000010,B01000010,B01000010,B01000010,B00111100,B00000000};
char v[] = {8, 8,
B00000000,B00100100,B00100100,B00100100,B00100100,B00100100,B00011000,B00000000};
char w[] = {8, 8,
```

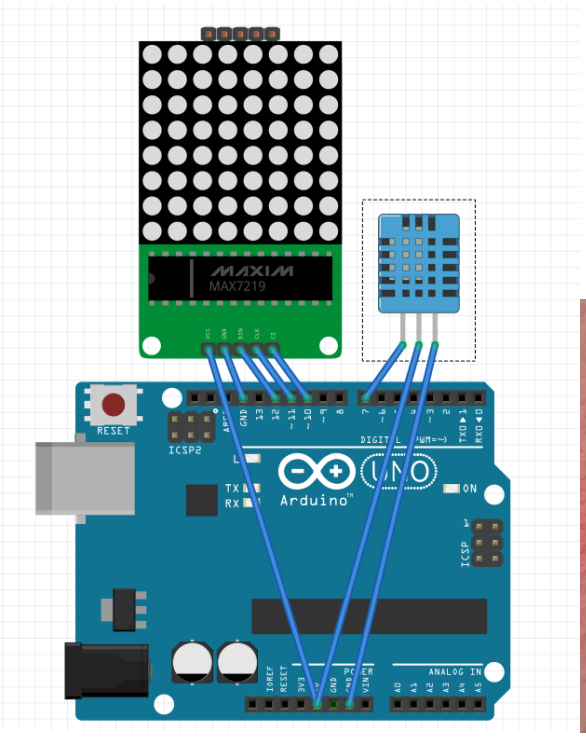
```
B00000000,B01010100,B01010100,B01010100,B01010100,B01010100,B00111000,B00000000};
char x[] = {8, 8,
B00000000,B01000100,B00101000,B00010000,B00101000,B01000100,B00000000,B00000000};
char y[] = {8, 8,
B10000001,B01000010,B00100100,B00011000,B00011000,B00011000,B00011000,B00011000};
char z[] = {8, 8,
B00000000,B01111100,B00001000,B00010000,B00100000,B01111100,B00000000,B00000000};vo
id setup() {
  m.init(); // MAX7219 initialization
  m.setIntensity(5); // initial led matrix intensity, 0-15      }
void loop() {
  // Seting the LEDs On or Off at x,y or row,column position
  m.setDot(6,2,true);
  delay(1000);
  m.setDot(6,3,true);
  delay(1000);
  m.clear(); // Clears the display
  for (int i=0; i<8; i++){
    m.setDot(i,i,true);
    delay(300);}
  m.clear();
  // Displaying the character at x,y (upper left corner of the character)
  m.writeSprite(0, 0, A);
  delay(1000);
  m.writeSprite(0, 0, B);
  delay(1000);
  m.writeSprite(0, 0, c);
  delay(1000);
  m.writeSprite(0, 0, d);
  delay(1000);
  m.writeSprite(0, 0, e);
  delay(1000);
  m.writeSprite(0, 0, f);
  delay(1000);
m.writeSprite(0, 0, g);
  delay(1000);
m.writeSprite(0, 0, h);
 delay(1000);
m.writeSprite(0, 0, i);
 delay(1000);
m.writeSprite(0, 0, j);
 delay(1000);
m.writeSprite(0, 0, k);
 delay(1000);
 m.writeSprite(0, 0, l);
 delay(1000);
m.writeSprite(0, 0, n);
 delay(1000);
m.writeSprite(0, 0, o);
 delay(1000);
```

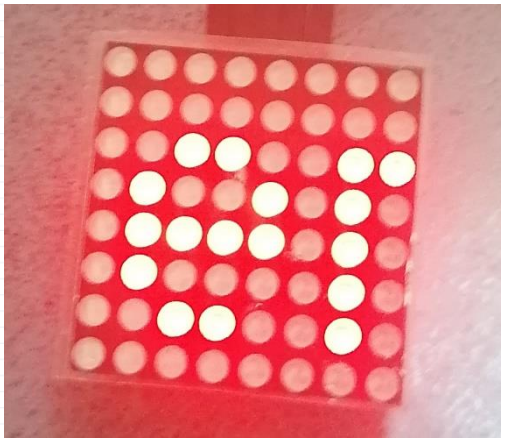
```
 m.writeSprite(0, 0, p);
delay(1000);
 m.writeSprite(0, 0, q);
delay(1000);
 m.writeSprite(0, 0, r);
delay(1000);
 m.writeSprite(0, 0, s);
delay(1000);
 m.writeSprite(0, 0, t);
delay(1000);
 m.writeSprite(0, 0, u);
delay(1000);
 m.writeSprite(0, 0, v);
delay(1000);
 m.writeSprite(0, 0, w);
delay(1000);
 m.writeSprite(0, 0, x);
delay(1000);
m.writeSprite(0, 0, y);
 delay(1000);
m.writeSprite(0, 0, z);
 delay(1000);}
```

Circuito 6:

Titolo del circuito: Interfaccia Dot Matrix con Arduino

Descrizione del circuito: L'interfaccia Dot Matrix visualizza la temperatura misurata con il sensore di temperatura DHT11





1-) Breadboard view

/* Interface Dot Matrix with Arduino */

```
//D7= temp
//D10=CS
//D11=CLK
//D12=DIN
#include <MaxMatrix.h> //include matrix library
#include <avr/pgmspace.h>
#include <stdlib.h>
#include "DHT.h" //include the temp sensor library
#define DHTPIN 7     // what pin we're connected to
#define DHTTYPE DHT11   // DHT 11 temp&humid sensor
DHT dht(DHTPIN, DHTTYPE);
PROGMEM const unsigned char CH[] = {
3, 8, B00000000, B00000000, B00000000, B00000000, B00000000, // space
1, 8, B01011111, B00000000, B00000000, B00000000, B00000000, // !
3, 8, B00000011, B00000000, B00000011, B00000000, B00000000, // "
5, 8, B00010100, B00111110, B00010100, B00111110, B00010100, // #
4, 8, B00100100, B01101010, B00101011, B00010010, B00000000, // $
5, 8, B01100011, B00010011, B00001000, B01100100, B01100011, // %
5, 8, B00110110, B01001001, B01010110, B00100000, B01010000, // &
1, 8, B00000011, B00000000, B00000000, B00000000, B00000000, // '
3, 8, B00011100, B00100010, B01000001, B00000000, B00000000, // (
3, 8, B01000001, B00100010, B00011100, B00000000, B00000000, // )
5, 8, B00101000, B00011000, B00001110, B00011000, B00101000, // *
5, 8, B00001000, B00001000, B00111110, B00001000, B00001000, // +
2, 8, B10110000, B01110000, B00000000, B00000000, B00000000, // ,
4, 8, B00001000, B00001000, B00001000, B00001000, B00000000, // -
2, 8, B01100000, B01100000, B00000000, B00000000, B00000000, // .
4, 8, B01100000, B00011000, B00000110, B00000001, B00000000, // /
4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // 0
3, 8, B01000010, B01111111, B01000000, B00000000, B00000000, // 1
4, 8, B01100010, B01010001, B01001001, B01000110, B00000000, // 2
4, 8, B00100010, B01000001, B01001001, B00110110, B00000000, // 3
4, 8, B00011000, B00010100, B00010010, B01111111, B00000000, // 4
4, 8, B00100111, B01000101, B01000101, B00111001, B00000000, // 5
4, 8, B00111110, B01001001, B01001001, B00110000, B00000000, // 6
4, 8, B01100001, B00010001, B00001001, B00000111, B00000000, // 7
4, 8, B00110110, B01001001, B01001001, B00110110, B00000000, // 8
4, 8, B00000110, B01001001, B01001001, B00111110, B00000000, // 9
2, 8, B01010000, B00000000, B00000000, B00000000, B00000000, // :
2, 8, B10000000, B01010000, B00000000, B00000000, B00000000, // ;
3, 8, B00010000, B00101000, B01000100, B00000000, B00000000, // <
3, 8, B00010100, B00010100, B00010100, B00000000, B00000000, // =
3, 8, B01000100, B00101000, B00010000, B00000000, B00000000, // >
4, 8, B00000010, B01011001, B00001001, B00000110, B00000000, // ?
5, 8, B00111110, B01001001, B01010101, B01011101, B00001110, // @
4, 8, B01111110, B00010001, B00010001, B01111110, B00000000, // A
```

```
4, 8, B01111111, B01001001, B01001001, B00110110, B00000000, // B
4, 8, B00111110, B01000001, B01000001, B00100010, B00000000, // C
4, 8, B01111111, B01000001, B01000001, B00111110, B00000000, // D
4, 8, B01111111, B01001001, B01001001, B01000001, B00000000, // E
4, 8, B01111111, B00001001, B00001001, B00000001, B00000000, // F
4, 8, B00111110, B01000001, B01001001, B01111010, B00000000, // G
4, 8, B01111111, B00001000, B00001000, B01111111, B00000000, // H
3, 8, B01000001, B01111111, B01000001, B00000000, B00000000, // I
4, 8, B00110000, B01000000, B01000001, B00111111, B00000000, // J
4, 8, B01111111, B00001000, B00010100, B01100011, B00000000, // K
4, 8, B01111111, B01000000, B01000000, B01000000, B00000000, // L
5, 8, B01111111, B00000010, B00001100, B00000010, B01111111, // M
5, 8, B01111111, B00000100, B00001000, B00010000, B01111111, // N
4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // O
4, 8, B01111111, B00001001, B00001001, B00000110, B00000000, // P
4, 8, B00111110, B01000001, B01000001, B10111110, B00000000, // Q
4, 8, B01111111, B00001001, B00001001, B01110110, B00000000, // R
4, 8, B01000110, B01001001, B01001001, B00110010, B00000000, // S
5, 8, B00000001, B00000001, B01111111, B00000001, B00000001, // T
4, 8, B00111111, B01000000, B01000000, B00111111, B00000000, // U
5, 8, B00001111, B00110000, B01000000, B00110000, B00001111, // V
5, 8, B00111111, B01000000, B00111000, B01000000, B00111111, // W
5, 8, B01100011, B00010100, B00001000, B00010100, B01100011, // X
5, 8, B00000111, B00001000, B01110000, B00001000, B00000111, // Y
4, 8, B01100001, B01010001, B01001001, B01000111, B00000000, // Z
2, 8, B01111111, B01000001, B00000000, B00000000, B00000000, // [
4, 8, B00000001, B00000110, B00011000, B01100000, B00000000, // \ backsplash
2, 8, B01000001, B01111111, B00000000, B00000000, B00000000, // ]
3, 8, B00000010, B00000001, B00000010, B00000000, B00000000, // hat
4, 8, B01000000, B01000000, B01000000, B01000000, B00000000, // _
2, 8, B00000001, B00000010, B00000000, B00000000, B00000000, // `
4, 8, B00100000, B01010100, B01010100, B01111000, B00000000, // a
4, 8, B01111111, B01000100, B01000100, B00111000, B00000000, // b
4, 8, B00111000, B01000100, B01000100, B00101000, B00000000, // c
4, 8, B00111000, B01000100, B01000100, B01111111, B00000000, // d
4, 8, B00111000, B01010100, B01010100, B00011000, B00000000, // e
3, 8, B00000100, B01111110, B00000101, B00000000, B00000000, // f
4, 8, B10011000, B10100100, B10100100, B01111000, B00000000, // g
4, 8, B01111111, B00000100, B00000100, B01111000, B00000000, // h
3, 8, B01000100, B01111101, B01000000, B00000000, B00000000, // i
4, 8, B01000000, B10000000, B10000100, B01111101, B00000000, // j
4, 8, B01111111, B00010000, B00101000, B01000100, B00000000, // k
3, 8, B01000001, B01111111, B01000000, B00000000, B00000000, // l
5, 8, B01111100, B00000100, B01111100, B00000100, B01111000, // m
4, 8, B01111100, B00000100, B00000100, B01111000, B00000000, // n
4, 8, B00111000, B01000100, B01000100, B00111000, B00000000, // o
4, 8, B11111100, B00100100, B00100100, B00011000, B00000000, // p
4, 8, B00011000, B00100100, B00100100, B11111100, B00000000, // q
4, 8, B01111100, B00001000, B00000100, B00000100, B00000000, // r
4, 8, B01001000, B01010100, B01010100, B00100100, B00000000, // s
```

```
3, 8, B00000100, B00111111, B01000100, B00000000, B00000000, // t
4, 8, B00111100, B01000000, B01000000, B01111100, B00000000, // u
5, 8, B00011100, B00100000, B01000000, B00100000, B00011100, // v
5, 8, B00111100, B01000000, B00111100, B01000000, B00111100, // w
5, 8, B01000100, B00101000, B00010000, B00101000, B01000100, // x
4, 8, B10011100, B10100000, B10100000, B01111100, B00000000, // y
3, 8, B01100100, B01010100, B01001100, B00000000, B00000000, // z
3, 8, B00001000, B00110110, B01000001, B00000000, B00000000, // {
1, 8, B01111111, B00000000, B00000000, B00000000, B00000000, // |
3, 8, B01000001, B00110110, B00001000, B00000000, B00000000, // }
4, 8, B00001000, B00000100, B00001000, B00000100, B00000000, // ~
};
int data = 12;    // DIN pin of MAX7219 module
int load = 10;    // CS pin of MAX7219 module
int clock = 11;  // CLK pin of MAX7219 module
int maxInUse = 1;    //change this variable to set how many MAX7219's you'll use
MaxMatrix m(data, load, clock, maxInUse); // define module
byte buffer[10];
void setup(){
  m.init(); // module initialize
  m.setIntensity(2); // dot matix intensity 0-15
  Serial.begin(9600); // serial communication initialize
  Serial.println("DHTxx test!");
  dht.begin();
}
void loop(){
  int t = dht.readTemperature();
  char temp[4];
  itoa(t,temp,10); //convert int to char!!!!
  Serial.println(temp);
   printStringWithShift("BRAILA ROMANIA", 100);
  printStringWithShift("  temp: ", 100);
  printStringWithShift(temp, 100);
  printStringWithShift(" C   ", 100);
  m.shiftLeft(false, true);
  }
void printCharWithShift(char c, int shift_speed){
  if (c < 32) return;
  c -= 32;
  memcpy_P(buffer, CH + 7*c, 7);
  m.writeSprite(32, 0, buffer);
  m.setColumn(32 + buffer[0], 0);
  for (int i=0; i<buffer[0]+1; i++)
  {
    delay(shift_speed);
    m.shiftLeft(false, false);
  }
}
void printStringWithShift(char* s, int shift_speed){
  while (*s != 0){
```

```
    printCharWithShift(*s, shift_speed);
    s++;
  }
}
void printString(char* s)
{
  int col = 0;
  while (*s != 0)
  {
    if (*s < 32) continue;
    char c = *s - 32;
    memcpy_P(buffer, CH + 7*c, 7);
    m.writeSprite(col, 0, buffer);
    m.setColumn(col + buffer[0], 0);
    col += buffer[0] + 1;
    s++;
  }
}
```

Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Titolo Progetto: “Teaching and Learning Arduinos in Vocational Training”

Acronimo Progetto: “ ARDUinVET ”

Progetto N: “2020-1-TR01-KA202-093762”

Motor Module and Training Kit

(CC, passo, servo)

Co-funded by the Erasmus+ Programme of the European Union

Modulo motore Arduino e KIT di formazione

Il kit di formazione per moduli motore è uno strumento didattico sviluppato nell'ambito del progetto ARDUinVET, con l'obiettivo di spiegare in modo semplice il funzionamento di diversi tipi di motori (Servo, Step e DC), controllati tramite Arduino.

Utilizzando la piattaforma di programmazione Arduino, si intende affrontare in modo semplice la programmazione degli Arduino per mettere in funzione i diversi motori, con l'utilizzo dell'hardware sviluppato.

L'obiettivo è mostrare agli studenti come si costruisce e si programma un circuito stampato (figura 1), nonché la sua applicabilità, facendo funzionare i motori ed eseguendo diversi compiti, in questo caso specifico.

Schema del circuito Scudo motore

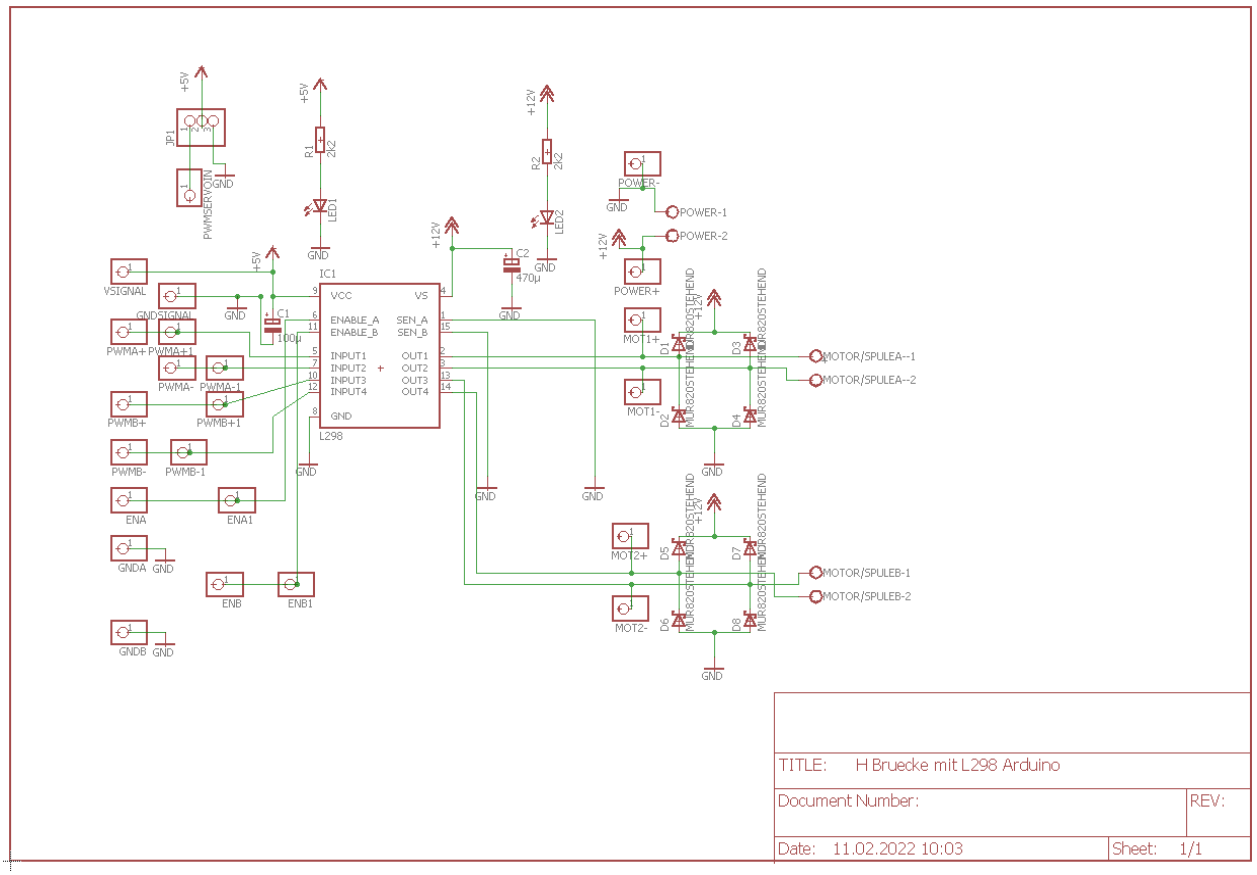


Immagine: 1

Layout Schermatura motore

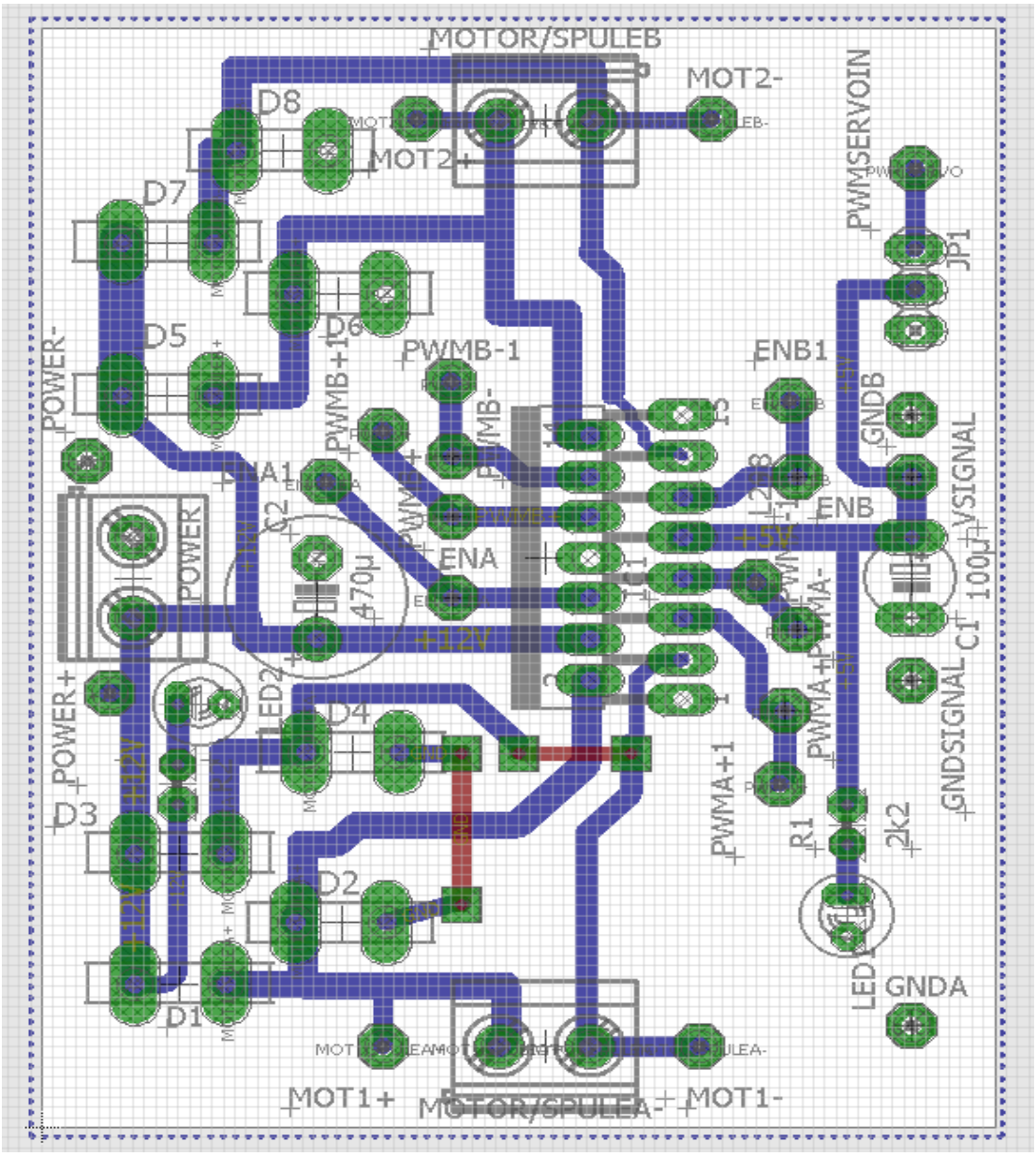


Immagine: 2

Co-funded by the Erasmus+ Programme of the European Union

Motori

Motori a corrente continua

I motori a corrente continua (motori a corrente continua, figura 3) sono dispositivi elettronici che funzionano grazie alle forze di attrazione e repulsione generate dai magneti.

Esempio:

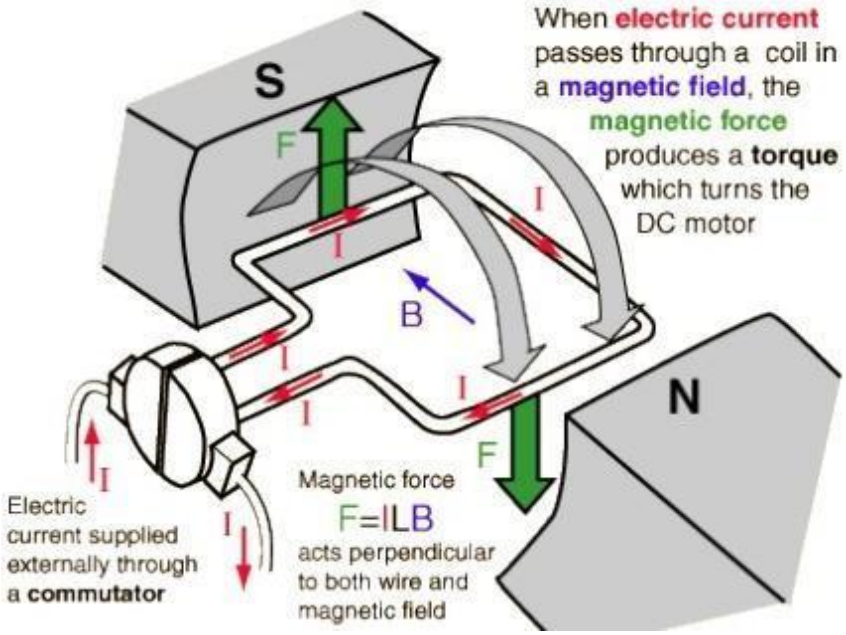
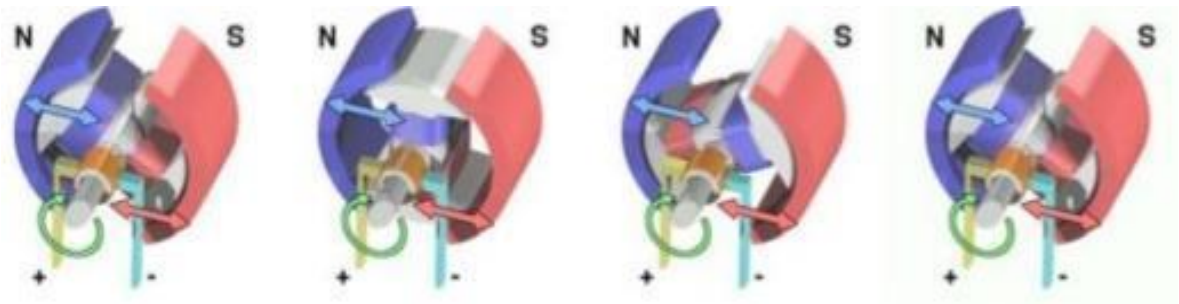


Figura 3: Funzionamento dei motori CC



1 2 3 4

1. Quando la bobina viene alimentata, si genera un campo magnetico intorno al rotore che provoca una repulsione tra i magneti, facendo girare il motore in senso orario;
2. Il rotore continua a girare;
3. Quando il rotore si allinea orizzontalmente, il campo magnetico si inverte, continuando il movimento in senso orario;
4. Il processo si ripete continuamente mentre il motore viene alimentato.

Motori a passo

Il motore passo-passo è un motore elettrico che si muove in base a un impulso ricevuto attraverso un controllore. Il numero di passi che il motore fornisce è esattamente uguale al numero di impulsi ricevuti e la velocità del motore è uguale alla frequenza degli impulsi. Si tratta quindi di un dispositivo semplice (senza spazzole, senza interruttore e senza encoder). Sono motori utilizzati in diversi settori dell'elettronica e dell'automazione, come la robotica, il movimento degli assi cartesiani, ecc.

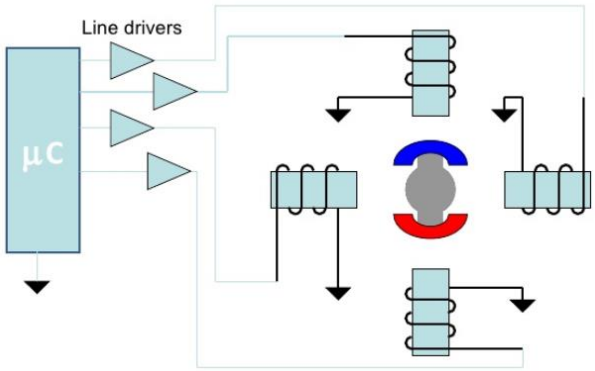


Figura 4: Schema semplice di un motore passo-passo

Figura 5: Motore a passo

Esempi di motori a passo

Motore unipolare

Un motore unipolare ha due bobine per fase, una per ogni direzione della corrente elettrica. In questa configurazione è possibile invertire un polo magnetico senza commutare la direzione della corrente elettrica; il circuito dell'interruttore può essere realizzato in modo molto semplice per ciascuna bobina.

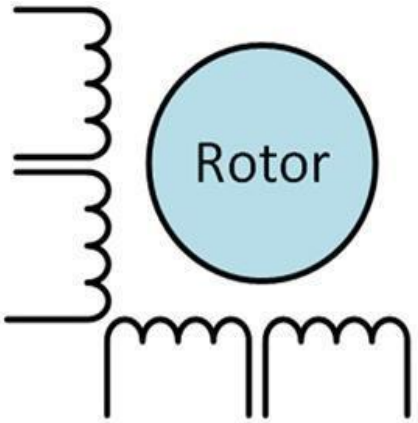


Figura 6: Motore unipolare

Cycle	C1	C2	C3	C4
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

Tabella 1: Tabella degli stati.

Motore bipolare

I motori bipolari hanno una bobina unica per fase. La corrente elettrica in una bobina deve essere invertita per invertire un polo magnetico. Il circuito elettrico è quindi un po' più complicato, richiedendo la necessità di un ponte ad H. Ci sono due collegamenti per fase e nessuno è in comune.

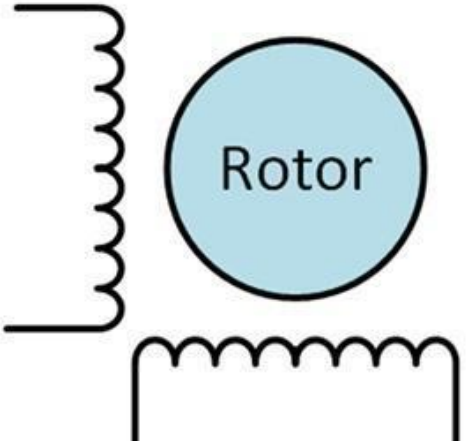


Figura 7: Motore bipolare

Cycle	C1	C2	C3	C4
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

Tabella 2: Tabella degli stati

Funzionamento motorio a gradini

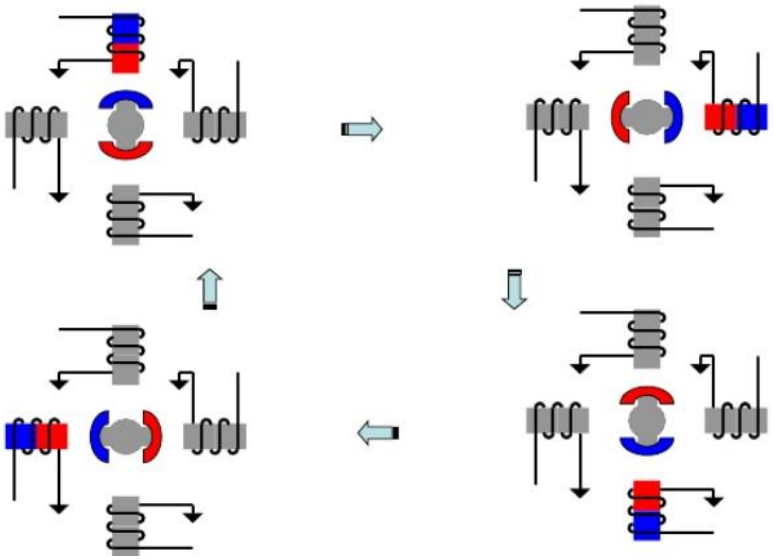


Figura 8: Schema del motore a gradini

Servomotori

Un servomotore è un dispositivo elettromeccanico che, attraverso un segnale elettrico, posiziona l'asse in una posizione angolare. Normalmente questo tipo di motore è compatto e consente una posizione precisa del proprio asse. Attualmente i servomotori sono utilizzati nella robotica e nella modellazione.

Operazione:

Un servomotore può essere suddiviso in quattro parti:

- **Circuito di controllo** - Responsabile della ricezione dei segnali PWM e dell'energia. Controlla la posizione del potenziometro e controlla il motore in base al segnale PWM ricevuto.
- **Potenziometro** - È collegato all'asse di uscita del servo, per posizionarlo.
- **Motore** - Muove l'ingranaggio e l'asse principale del servo.
- **Ingranaggi di trasmissione** - Riducono la rotazione del motore, diminuendone la forza in modo da applicare una coppia superiore all'asse principale. Muovono il potenziometro insieme all'asse.

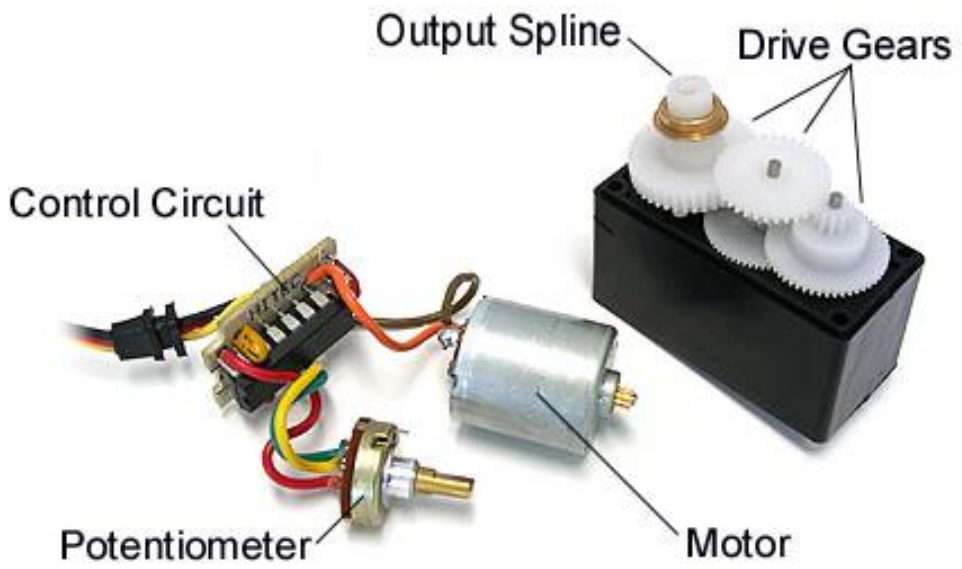


Figura 10: Parti del servomotore

Posizioni del servomotore

Il controllo di un servomotore si ottiene mediante un segnale di ingresso che presenta livelli di tensione TTL che ne specificano la posizione. Il formato di questo segnale segue la modulazione PWM (Pulse Width Modulation) come mostrato in figura 11. La codifica in PWM avviene attraverso impulsi di alto livello in relazione al periodo totale di oscillazione, che nel caso dei servomotori è di 20ms.

In questo caso specifico, se in 20 millisecondi 1 millisecondo è a livello alto, il servomotore dovrebbe trovarsi nella posizione minima, come mostrato nella figura 11.

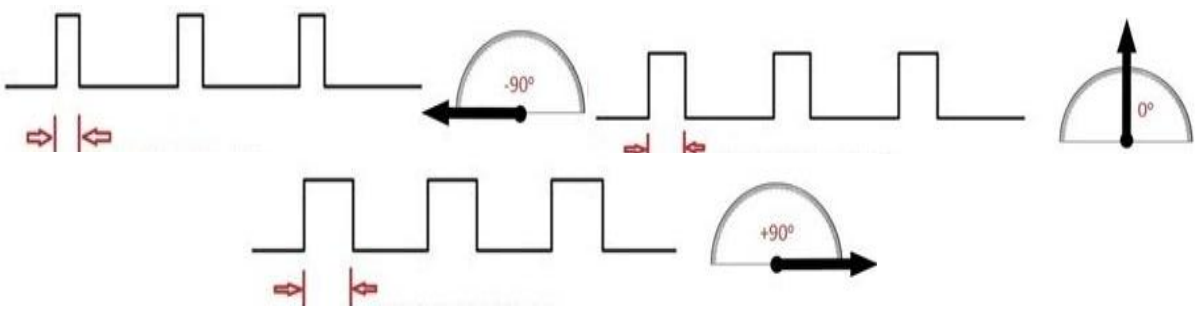


Figura 11: Servomotore PWM

PWM (modulazione di larghezza di impulso)

Il PWM può essere implementato in diversi settori dell'elettronica. Uno dei suoi impieghi è l'alimentazione, il controllo della velocità dei motori CC, il controllo della luce, il controllo dei servomotori e molte altre applicazioni. Attraverso la PWM è possibile controllare la velocità e la potenza.

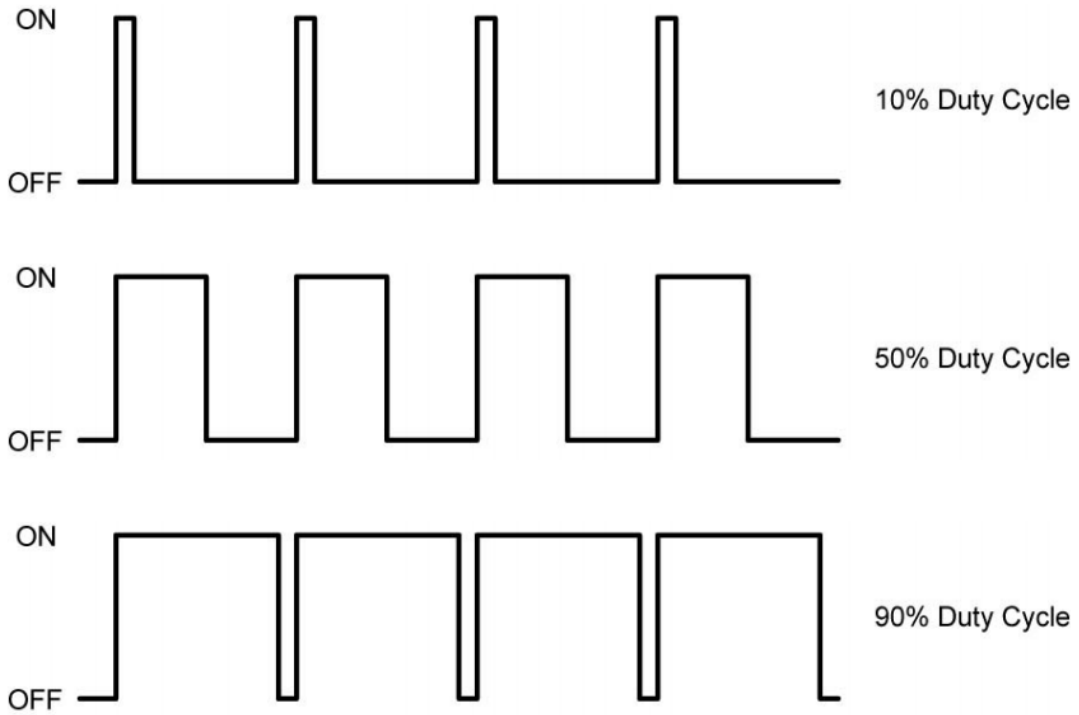


Figura 12: PWM

Funzionamento PWM

Considerando un'onda quadra, per ottenere il corretto funzionamento del PWM dobbiamo variare la larghezza dell'impulso dell'onda. Per il calcolo sono necessari il periodo e l'ampiezza dell'impulso e il risultato è chiamato duty-cycle, definito dall'equazione:

$$\text{Duty Cycle} = 100 \text{Pulse Width} \text{Period}$$

Ciclo di lavoro: valore percentuale;
Larghezza d'impulso: sequenza di tempo in cui il segnale è a livello alto;
Periodo: durata di un ciclo d'onda.

NOTA: I seguenti esperimenti saranno eseguiti utilizzando il kit motore Arduino, che gli studenti realizzeranno da soli.

Titolo del circuito_1 (con motori CC): "Un motore DC semplice"

Spiegazione del circuito: Questo programma dimostra le sequenze di accensione e spegnimento di un motore a corrente continua.

Programma:

// Un motore CC semplice

```
/*
     Connessione:
Arduino | MotorShielt
PIN | PIN
-----------------------
+5V | VSIGNAL
GND | SEGNALE GND
4 | ENA
5 | PWMA+
6 | PWMA-
*/
//-------------------------------------------------------------
// variabili
//-------------------------------------------------------------
int ena = 4; // il pin 4 di Arduino è collegato all'ENA
int right = 5; // pin 5 di Arduino collegato a PWMA+
int left = 6; // Il pin 5 di Arduino è collegato a PWMA-.

//-------------------------------------------------------------
// funzione di impostazione
//-------------------------------------------------------------
vuoto setup()
{
  pinMode(ena, OUTPUT);
  pinMode(right, OUTPUT);
  pinMode(left, OUTPUT);
  digitalWrite(ena, HIGH); /Attiva ENA
  ritardo(900);
}
//-------------------------------------------------------------
// ciclo Funzione
//-------------------------------------------------------------
vuoto loop()
{
  analogWrite(destra, 255); // gira a destra con la massima velocità
  ritardo(1000);
  analogWrite(right, 0); // spegnere
  ritardo(1000);
  analogWrite(left, 255); // gira a sinistra alla massima velocità
  ritardo(1000);
  analogWrite(left, 0); // spegnere
  ritardo(1000);
  analogWrite(right, 127); // gira a destra a metà velocità
  ritardo(1000);
  analogWrite(right, 0); // spegnere
  ritardo(1000);
  analogWrite(left, 127); // gira a sinistra a metà velocità
  ritardo(1000);
  analogWrite(left, 0); // spegnere
  ritardo(1000);
}
```

Titolo del circuito_2: "Avanzamento di un motore CC".

Spiegazione del circuito: Questo programma controlla un motore a corrente continua. I comandi di controllo vengono impartiti tramite il monitor seriale.

I comandi di controllo:

Scrivere nel monitor seriale	*Azione*
fermarsi	Il motore si ferma
andare a destra	Motore a destra
andare a sinistra	Motore a sinistra
+	Aumentare la velocità
-	Riduzione della velocità

Programma:

// Anticipo di un motore CC

```
/*
    Connessione:
Arduino | MotorShielt
PIN | PIN
-------------------------
+5V | VSIGNAL
GND | SEGNALE GND
4 | ENA
5 | PWMA+
6 | PWMA-
*/
//-------------------------------------------------------------
// variabili
//-------------------------------------------------------------
int en_a = 4; /Collegamento del pin 4 di ArduPin al pin di abilitazione
int right_a = 5; //il pin5 di ArduPin è collegato al pin PWMA+.
int left_a = 6; /Pin6 collegato al pin PWMA
String message = ""; //salva il messaggio dell'interfaccia seriale
int val = 80; //valore di defalco per la velocità
int del = 10; //valore in- e in- diminuito con i comandi "+" e "-".
int minimumVal = 70; //valore minimo della velocità / valore
int maximumVal = 250; //velocità massima / valore
bool flag_go_right = false; //diventa vero quando viene impartito il comando "vai a destra".
bool flag_go_left = false; //diventa vero quando viene impartito il comando "go left".

//-------------------------------------------------------------
// funzione di impostazione
//-------------------------------------------------------------
vuoto setup()
{
  Serial.begin(9600);
  pinMode(en_a, OUTPUT);
  pinMode(right_a, OUTPUT);
  pinMode(left_a, OUTPUT);
  digitalWrite(en_a, LOW);
```

```
  analogWrite(right_a, 0);
  analogWrite(left_a, 0);
  ritardo(900);
}
//-------------------------------------------------------------
// ciclo Funzione
//-------------------------------------------------------------
vuoto loop()
{
  se(Serial.available()>0)
  {
    messaggio = Serial.readString();
    Serial.println(" --> Messaggio in arrivo: " + messaggio ); //Vedi il messaggio in arrivo

    if(message.equals("stop\n"))
    {
      digitalWrite(en_a, LOW);
    analogWrite(right_a, 0);
      analogWrite(left_a, 0);
      flag_go_right = false;
      flag_go_left = false;
      Serial.println(" <-- Messaggio in uscita: stop \n");
    }
    else if(message.equals("vai a destra"))  //verifica il messaggio se "vai a destra"
    {
      digitalWrite(en_a, HIGH); //impostazione del pin di abilitazione del Modul su HIGH
    analogWrite(right_a, val); //scrive il valore sul pin PWM della svolta a destra
      analogWrite(left_a, 0); //scrive zero sul pin PWM della svolta a sinistra
      flag_go_left = false; //impostazione dei flag
      flag_go_right = true;
      Serial.println(" <-- Messaggio in uscita: vai a destra\n"); //feedback
    }
    else if(message.equals("vai a sinistra"))
    {
      digitalWrite(en_a, HIGH);
    analogWrite(right_a, 0);
      analogWrite(left_a, val);
      flag_go_right = false;
      flag_go_left = true;
      Serial.println(" <--Messaggio in uscita: vai a sinistra\n");
    }
    else if(message.equals("+\n"))      //controlla il messaggio se "+"
    {
      if(val >= maximumVal) // raggiunge il valore massimo
      {
        Serial.println(" <-- Messaggio in uscita: valore massimo! \n");
      }
      altrimenti se (val < maximumVal)
      {
        val = val + del; //aumenta il valore
        if(flag_go_right) // controlla che la direzione sia quella dell'incremento
        {
          analogWrite(right_a, val);
```

```
        }
        else if(flag_go_left)
        {
          analogWrite(left_a, val);
        }
        Serial.print(" <-- Messaggio in uscita: valore = "); Serial.println(val); Serial.println("");
//feedback
      }
    }
    else if(message.equals("-\n"))
    {
      if(val <= minimumVal) // raggiungere la velocità minima
      {
        Serial.println(" <-- Messaggio in uscita: valore minimo! \n");
      }
      altrimenti se (val > minimumVal)
      {
        val = val - del;
        se(flag_go_right)
        {
          analogWrite(right_a, val);
        }
        else if(flag_go_left)
        {
          analogWrite(left_a, val);
        }
        Serial.print(" <-- Messaggio in uscita: valore = "); Serial.println(val); Serial.println("");
      }
    }
    else //genera il messaggio se il messaggio è sconosciuto
    {
      Serial.println("<<-- Il messaggio in arrivo " + messaggio + " è sconosciuto");
    }
  }
}
```

Circuito title_3: "Avanzamento di due motori in c.c.".

Spiegazione del circuito: Questo programma controlla due motori a corrente continua. I comandi di controllo vengono impartiti tramite il monitor seriale.

Il comando di controllo:

Scrivere nel monitor seriale	*Azione*
fermarsi	Il motore si ferma
fermare un	Arresto del motore a
fermarsi b	Arresto del motore b
andare a destra a	Motore a destra
andare a sinistra a	Motore a sinistra
vai a destra b	Motore b svolta a destra

andare a sinistra b	Motore b girare a sinistra
+	Aumentare la velocità
-	Riduzione della velocità

Programma:

```
// due motori DC avanzano
/*
 Connessione:
Arduino | MotorShielt
PIN | PIN
-----------------------
+5V | VSIGNAL
GND | GND
4 | ENA
5 | PWMA+
6 | PWMA-
8 | ENB
9 | PWMB+
10 | PWMB-
*/

//-------------------------------------------------------------
// variabili
//-------------------------------------------------------------
String message = "";

int en_a = 4;
int right_a = 5;
int left_a = 6;
bool flag_go_right_a = false;
bool flag_go_left_a = false;

int en_b = 8;
int right_b = 9;
int left_b = 10;
bool flag_go_right_b = false;
bool flag_go_left_b = false;

int val = 255;
int del = 10;
int minimumVal = 70;
int maximumVal = 250;

//-------------------------------------------------------------
// impostazione Funzione
//-------------------------------------------------------------
vuoto setup()
{
  Serial.begin(9600);

  pinMode(en_a, OUTPUT);
  pinMode(right_a, OUTPUT);
```

```
  pinMode(left_a, OUTPUT);

  pinMode(en_b, OUTPUT);
  pinMode(right_b, OUTPUT);
  pinMode(left_b, OUTPUT);

  digitalWrite(en_a, LOW);
  analogWrite(right_a, 0);
  analogWrite(left_a, 0);

  digitalWrite(en_b, LOW);
  analogWrite(right_b, 0);
  analogWrite(left_b, 0);

  ritardo(900);
}

//-------------------------------------------------------------
// ciclo Funzione
//-------------------------------------------------------------
vuoto loop()
{
  se(Serial.available()>0)
  {
    messaggio = Serial.readString();
    Serial.println(" --> Messaggio in arrivo: " + messaggio);

    if(message.equals("stop\n"))                    // entrambi i motori si fermano
    {
      digitalWrite(en_a, LOW);
    analogWrite(right_a, 0);
      analogWrite(left_a, 0);
      flag_go_right_a = false;
      flag_go_left_a = false;

      digitalWrite(en_b, LOW);
      analogWrite(right_b, 0);
      analogWrite(left_b, 0);
      flag_go_right_b = false;
      flag_go_left_b = false;

      Serial.println(" <-- Messaggio in uscita: stop all\n");
    }
    else if(message.equals("stop a\n"))             // motore un arresto
    {
      digitalWrite(en_a, LOW);
      analogWrite(right_a, 0);
      analogWrite(left_a, 0);
      flag_go_right_a = false;
      flag_go_left_a = false;
      Serial.println(" <--Messaggio in uscita: stop a\n");
    }
    else if(message.equals("stop b\n"))             // arresto del motore b
```

```
{
 digitalWrite(en_b, LOW);
 analogWrite(right_b, 0);
 analogWrite(left_b, 0);
 flag_go_right_b = false;
 flag_go_left_b = false;
 Serial.println(" <-- Messaggio in uscita: stop b\n");
 }
else if(message.equals("vai a destra"))
{
 digitalWrite(en_a, HIGH);
analogWrite(right_a, val);
 analogWrite(left_a, 0);
 flag_go_left_a = false;
 flag_go_right_a = true;
 Serial.println(" <-- Messaggio in uscita: andare a destra");
}
else if(message.equals("vai a destra b\n"))
{
 digitalWrite(en_b, HIGH);
 analogWrite(left_b, 0);
 analogWrite(right_b, val);
 flag_go_left_b = false;
 flag_go_right_b = true;
 Serial.println(" <-- Messaggio in uscita: andare a destra");
}
else if(message.equals("vai a sinistra"))
{
 digitalWrite(en_a, HIGH);
analogWrite(right_a, 0);
 analogWrite(left_a, val);
 flag_go_right_a = false;
 flag_go_left_a = true;
 Serial.println(" <-- Messaggio in uscita: andare a sinistra");
}
else if(message.equals("vai a sinistra b\n"))
{
 digitalWrite(en_b, HIGH);
 analogWrite(right_b, 0);
 analogWrite(left_b, val);
 flag_go_right_b = false;
 flag_go_left_b = true;
 Serial.println(" <-- Messaggio in uscita: andare a sinistra");
}
else if(message.equals("+n"))
{
 if(val >= maximumVal) // raggiunge il valore massimo
 {
  Serial.println(" <-- Messaggio in uscita: valore massimo! \n");
 }
 altrimenti se (val < maximumVal)
 {
  val = val + del;
```

```
      if(flag_go_right_a) analogWrite(right_a, val);
      if(flag_go_right_b) analogWrite(right_b, val);
      if(flag_go_left_a) analogWrite(left_a, val);
      if(flag_go_left_b) analogWrite(left_b, val);
      Serial.print(" <-- Messaggio in uscita: valore = "); Serial.println(val); Serial.println("");
     }
    }
    else if(message.equals("-\n"))
    {
     if(val <= minimumVal) // raggiungere il valore minimo
     {
      Serial.println(" <-- Messaggio in uscita: valore minimo! \n");
     }
     altrimenti se (val > minimumVal)
     {
      val = val - del;
      if(flag_go_right_a) analogWrite(right_a, val);
      if(flag_go_right_b) analogWrite(right_b, val);
      if(flag_go_left_a) analogWrite(left_a, val);
      if(flag_go_left_b) analogWrite(left_b, val);

      Serial.print(" <-- Messaggio in uscita: valore = "); Serial.println(val); Serial.println("");
     }
    }
    altro
    {
     Serial.println(" <<-- Messaggio in arrivo " + messaggio + " È SCONOSCIUTO");
    }
  }
}
```

Circuito title_4: "Semplice programma motore a passi".

Spiegazione del circuito: "Un motore a gradini gira in senso orario e antiorario.

Programma:

// Semplice programma motore a passi

```
/*
   Connessione:
Arduino | MotorShielt
PIN | PIN
-------------------------
+5V | VSIGNAL
GND | SEGNALE GND
4 | ENA
5 | PWMA+
6 | PWMA-
8 | ENB
9 | PWMB+
10 | PWMB-
```

```
*/

//---------------------------------------------------------------
// biblioteche
//---------------------------------------------------------------
#include < Stepper.h>
// libretto per lo stepper
//---------------------------------------------------------------
// costanti
//---------------------------------------------------------------
#define STEPS 200
// Angolo di passo del motore passo-passo utilizzato = 1,8 gradi
// 360 / 1,8° = 200

//---------------------------------------------------------------
// classe
//---------------------------------------------------------------
Motore passo-passo (FASI, 5,6,9,10);
  //creare una nuova istanza della classe Stepper
  //Parametro:
  // PASSI -> il numero di passi in un giro del motore
  // 5,6,9,10 -> Pin utilizzati

//---------------------------------------------------------------
// variabili
//---------------------------------------------------------------
int spe = 50;

//---------------------------------------------------------------
// funzione di impostazione
//---------------------------------------------------------------
vuoto setup()
{
  Serial.begin(9600);
  pinMode(4, INPUT);
  pinMode(8, INPUT);
  digitalWrite(4 ,HIGH); //attivazione del pin ENA del MotorShielt
  digitalWrite(8 ,HIGH); //attivazione del pin ENB del MotorShielt

  Motor.setSpeed(spe); //Oggetto Motore ottenere la velocità in "rotazione al minuto".
}

vuoto loop()
{
  Motor.step(100); /Ruota il motore di un numero specifico di passi,
                   //alla velocità determinata dall'ultima chiamata a setSpeed();
                   //in questo caso 100 passi in senso orario;
  ritardo(200);

  Motor.step(-50); //50 passi in senso antiorario;
  ritardo(200);
}
```

Circuito title_5: "Programma motore a passi avanzato"

Spiegazione del circuito: Questo programma controlla un motore passo-passo. I comandi di controllo vengono impartiti tramite il monitor seriale.

Esempio: Inviare 100 attraverso il monitor seriale ad Arduino Il motore passo-passo fa 100 passi. Se si invia -100 il motorino passo-passo compie 100 passi nella direzione opposta.

Programma:

```
// Programma motore a passo avanzato
/*
   Connessione:
Arduino | MotorShielt
PIN | PIN
-------------------------
+5V | VSIGNAL
GND | SEGNALE GND
4 | ENA
5 | PWMA+
6 | PWMA-
8 | ENB
9 | PWMB+
10 | PWMB-
*/

//---------------------------------------------------------------
// biblioteche
//---------------------------------------------------------------
#include < Stepper.h>
//---------------------------------------------------------------
// costanti
//---------------------------------------------------------------
#define STEPS 200
//---------------------------------------------------------------
// classe
//---------------------------------------------------------------
Motore passo-passo (FASI, 5,6,9,10);
//---------------------------------------------------------------
// variabili
//---------------------------------------------------------------

int spe = 100;
String message = "";

//---------------------------------------------------------------
// funzione di impostazione
//---------------------------------------------------------------

vuoto setup()
```

```
{
  Serial.begin(9600);
  pinMode(4, INPUT);
  pinMode(8, INPUT);
  digitalWrite(4 ,HIGH);
  digitalWrite(8 ,HIGH);
  Motor.setSpeed(spe);
}

vuoto loop()
{
  se(Serial.available()>0)
  {
    messaggio = Serial.readString();
    Serial.println("Messaggio in arrivo: " + messaggio );

    if(message.toInt())
    {
      Serial.print(" <-- Messaggio in uscita: Steps -> " + messaggio);
      Motor.step(message.toInt());
    }
    altro
    {
      Serial.println(" <<-- Messaggio in arrivo " + messaggio + " !!! --> non è un numero <-- !!!
\´n'n");
    }
  }
}
```

Circuito title_6 (con servomotore): " Programma semplice per servomotore".

Spiegazione del circuito: In questo programma un servomotore viene controllato da un segnale pwm. Viene utilizzato il ciclo for

```
// Programmazione semplice di un servomotore
/*
   Connessione:
Arduino | MotorShielt
PIN | PIN
-----------------------
+5V | VSIGNAL
GND | SEGNALE GND
3 | PWMSERVOIN
*/
//-------------------------------------------------------------
// variabili
//-------------------------------------------------------------
int del = 100; // tempo di ritardo

//-------------------------------------------------------------
// funzione di impostazione
//-------------------------------------------------------------
vuoto setup()
{
  Serial.begin(9600);
  pinMode(3, OUTPUT);
}

vuoto loop()
{
  for(int i = 0; i<255; i++)
    {
     analogWrite(3,i);
     Serial.println(i);
     ritardo(del);
    }

   ritardo(1000);
}
```

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Titolo Progetto: “Teaching and Learning Arduinos in Vocational Training”

Acronimo Progetto: “ ARDUinVET ”

N°Progetto: “2020-1-TR01-KA202-093762”

Sensors Module and Training Kit

Sensori

Un sensore è un dispositivo, un modulo o un sottosistema dell'elettronica il cui scopo è creare un'interfaccia tra un circuito e il suo ambiente. È il sistema che riceve informazioni dall'ambiente, il mondo fisico, e le invia come valore al circuito. In realtà rileva gli eventi o i cambiamenti che avvengono nell'ambiente. Per rilevare questi eventi o cambiamenti, il sensore deve misurare determinati valori in una scala fisica. Misura una proprietà, come la pressione, la posizione, la temperatura o l'accelerazione, e risponde con un feedback. Quindi converte questi valori in un segnale elettrico (solitamente).

Pertanto, un sensore viene sempre utilizzato insieme ad altri dispositivi elettronici. I sensori si trovano in molti oggetti di uso quotidiano, come i dispositivi sensibili al tatto o al movimento, i dispositivi azionati dalla luce o dal suono, ecc. L'uso dei sensori si è esteso oltre i tradizionali campi di misurazione della temperatura, della pressione o del flusso, ad esempio i sensori GPS. I sensori analogici sono ancora ampiamente utilizzati. Le applicazioni comprendono la produzione e i macchinari, gli aeroplani e il settore aerospaziale, le automobili, la medicina, la robotica e molti altri aspetti della nostra vita quotidiana.

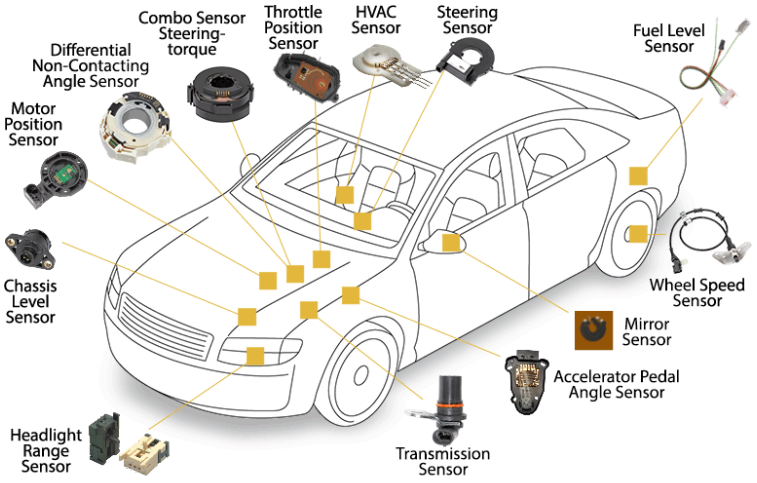


Figura 1. Sensori di un’automobile

Caratteristiche di un sensore

Un buon sensore rispetta le seguenti regole:

- deve essere sensibile alla proprietà misurata
- deve essere insensibile a qualsiasi altra proprietà che si possa incontrare nella sua applicazione
- non deve influenzare la proprietà misurata.

Le principali caratteristiche di base dei sensori sono:

- Linearità. Il sensore ha una proprietà o caratteristica il cui valore cambia. Quando la grandezza fisica che misura viene anch'essa modificata. È auspicabile che le variazioni nella misura della grandezza fisica causino cambiamenti apprezzabili nella proprietà del sensore. Questa proprietà è chiamata linearità ed è di fondamentale importanza.

- Accuratezza. La vicinanza del valore di uscita al valore di ingresso.

- Errore: La differenza tra il valore misurato e il valore reale.

- Tolleranza: L'errore massimo che il sensore può generare.

- Ingresso a fondo scala (FSI): Specifica in quali fotogrammi della dimensione fisica misurata può essere utilizzato il sensore.

- Fondo scala in uscita (FSO): Imposta i valori che la tensione o la corrente possono ricevere all'uscita del sensore.

- Sensibilità: Esprime quanto è alto il segnale di uscita del sensore per ogni unità della grandezza fisica misurata

- Risoluzione: Esprime la più piccola variazione della grandezza fisica che il sensore può rilevare e modificare di conseguenza il suo valore di uscita

- Isteresi: Un errore di isteresi fa sì che il valore di uscita vari a seconda dei valori di ingresso precedenti. Se l'uscita di un sensore è diversa a seconda che un determinato valore di ingresso sia stato raggiunto aumentando o diminuendo l'ingresso, il sensore presenta un errore di isteresi.

- Ritardo: È il ritardo della variazione del valore di uscita dopo una modifica dell'ingresso.

- Zona morta: la quantità massima di variazione del valore di ingresso che non influisce sul valore di uscita.

Categorie di sensori

Esistono diversi modi per classificare i sensori, alcuni dei quali sono elencati di seguito.

La prima categorizza i sensori in base alla forma del valore di uscita, dividendo i sensori analogici da quelli digitali.

La seconda categorizzazione riguarda ciò che un sensore può misurare, con una distinzione più significativa tra sensori naturali e chimici. I sensori naturali controllano grandezze fisiche come la posizione, la massa, la corrente, il tempo e le loro dimensioni relative, mentre i sensori chimici controllano la presenza di diversi gas in una particolare atmosfera.

Un'altra modalità si riferisce ai materiali con le cui proprietà fisiche funzionano i sensori, con categorie principali di sensori con materiali conduttivi, semiconduttori, dielettrici, magnetici e superconduttivi.

Infine, un altro metodo di classificazione si riferisce al campo di utilizzo del sensore, con categorie principali quali sensori industriali, medici, militari, ambientali, nonché sensori per applicazioni di trasporto e automazione.

Figura 2: Vari sensori

Sensori di base

Ecco alcuni sensori di base per applicazioni comuni:

Fotoresistenza LDR: si tratta di una resistenza variabile il cui valore cambia a seconda della luce che vi cade sopra.

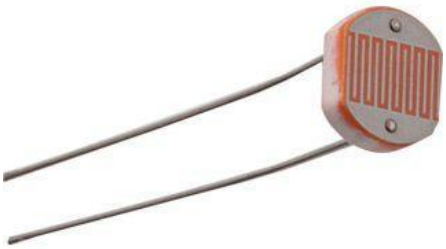


Figura 3: Fotoresistenza

Sensore digitale di luminosità / Lux / Sensore di luce: Il sensore di luminosità TSL2561 è un sensore di luce digitale avanzato, ideale per l'uso in un'ampia gamma di situazioni di luce. Questo sensore è più preciso e consente di calcolare con esattezza i lux e può essere configurato per diversi intervalli di guadagno/tempo per rilevare al volo intervalli di luce compresi tra 0,1 e 40.000+ Lux. Contiene diodi sia a infrarossi che a spettro completo! Ciò significa che è possibile misurare separatamente la luce a infrarossi, a spettro completo o visibile dall'uomo.

Figura 4: Sensore digitale di luminosità / Lux / Sensore di luce

Sensore di temperatura (con uscita analogica): L'intervallo di temperatura che misura è solitamente compreso tra -40 °C e + 100 °C con una precisione di ± 1 °C.

Figura 5: Sensore di temperatura

Sensore di umidità e temperatura: sensore digitale di temperatura e umidità altamente preciso. Presenta un intervallo di misurazione 0-100% RH con una precisione di temperatura di +/- 0,3°C a 25°C.

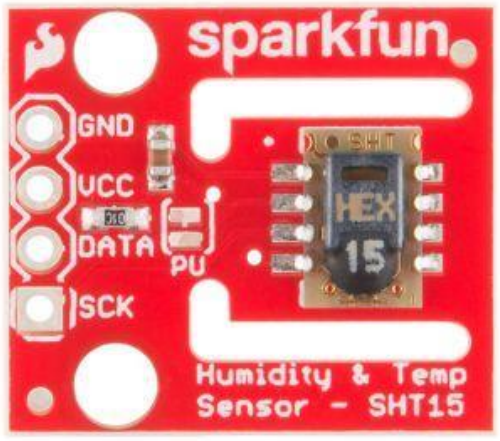


Figura 6: Sensore di umidità e temperatura

Sensore a infrarossi: utilizzato per il calcolo della distanza. La distanza calcolabile va da 2 cm. a 400 cm. con una precisione di un centimetro.

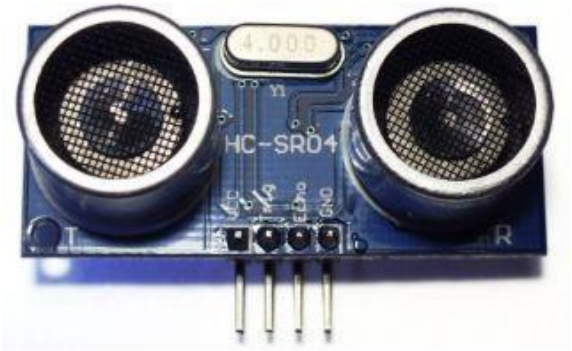


Figura 7: Sensore di distanza a ultrasuoni

Sensori di movimento: È in grado di rilevare il movimento di un essere umano o di un animale domestico all'interno di una stanza entro sei metri. Il sensore è dotato di due resistenze trimmer che consentono di regolare la sensibilità e il tempo di attivazione dal momento in cui rileva il movimento.

Figura 8: Sensore di movimento

Sensore di vibrazione: Quando il sensore rileva una vibrazione, viene generata una tensione, utilizzando una resistenza da 1Mohm.

Figura 9: Sensore di vibrazione

Sensore breakout con accelerometro e giroscopio a tre assi: Combinando un giroscopio a 3 assi e un accelerometro a 3 assi sullo stesso die di silicio con un processore di movimento digitale (DMP) a bordo in grado di elaborare complessi algoritmi di Motion Fusion a 9 assi, il sensore può restituire tutti i valori di allineamento trasversale.

Figura 10: Sensore breakout con accelerometro e giroscopio a tre assi

Gas Sensor: Sensibile per GPL, gas naturale e gas di carbone. La tensione di uscita aumenta con l'aumentare della concentrazione dei gas misurati.

Figura 11: Gas sensor

Rilevatore di suoni: Questo sensore fornisce un'uscita audio, ma anche un'indicazione binaria della presenza del suono e una rappresentazione analogica della sua ampiezza.

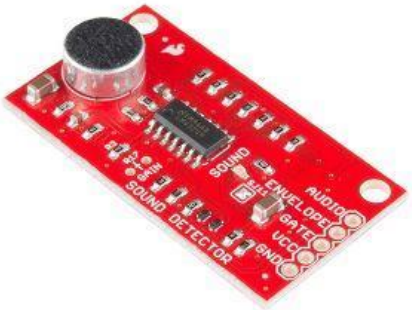


Figura 12: Rilevatore di suoni

PROGETTO 1- NOME: SENSORE AD ULTRASUONI PER SEGNALETICA STRADALE

Scopo del progetto: In questo progetto gli studenti vedranno come utilizzare un sensore a ultrasuoni in un segnale stradale utilizzando un resistore di pull-up esterno.

Attraverso questo progetto, gli studenti potranno sperimentare un sensore a ultrasuoni, un dispositivo che misura la distanza di un oggetto utilizzando le onde sonore.

Metodologia

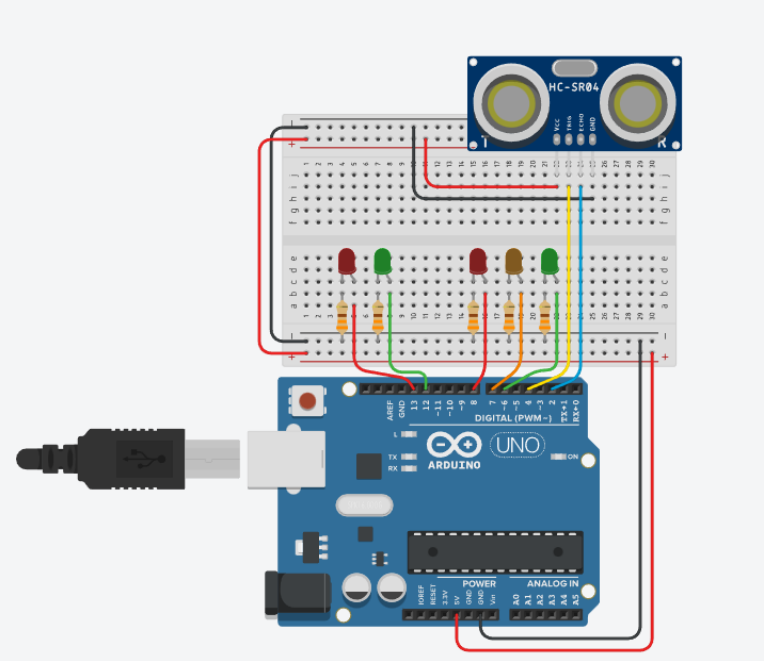
Inizialmente, descriviamo il flusso di lavoro desiderato del progetto. Poi chiediamo agli studenti di selezionare i componenti necessari e di usare Tinkercad per progettare e disegnare il circuito. Utilizzeranno lo strumento Codice di Tinkercad per scrivere il codice.

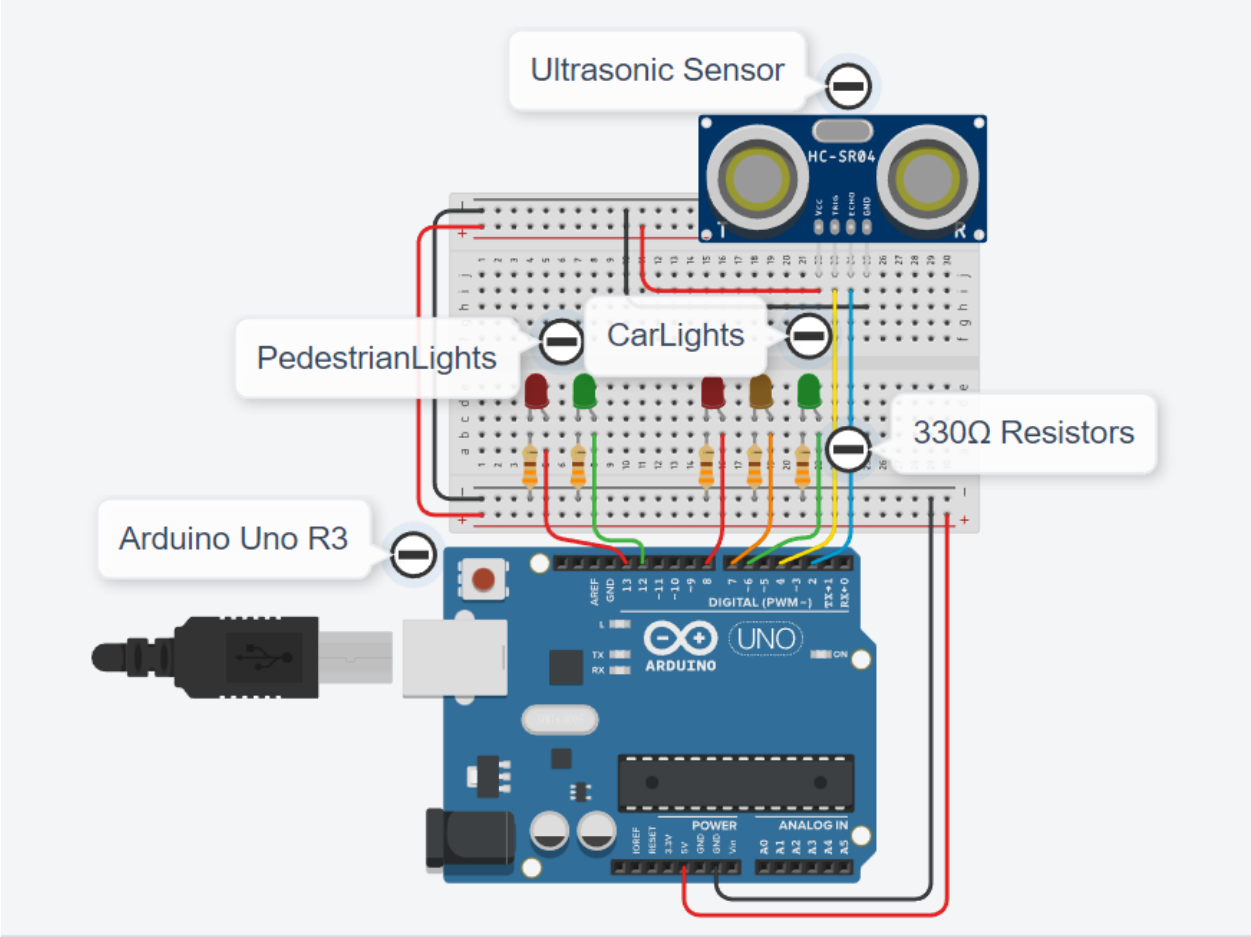
Quindi creeranno il circuito, scriveranno il codice sullo strumento software Arduino e lo caricheranno sulla scheda Arduino.

La simulazione su Tinkercad verificherà che il circuito sia stato costruito correttamente.

Questo progetto è rivolto agli studenti del livello base, che iniziano a beneficiare dei risultati e degli esiti del nostro progetto Erasmus+ KA-202 Strategic Partnerships in VET, chiamato "ArduinVET".

Circuito di progetto.





Come funziona:

Il sensore a ultrasuoni funziona inviando un'onda sonora a una frequenza ultrasonica e aspettando che rimbalzi sull'oggetto. Il ritardo tra la trasmissione e la ricezione del suono viene utilizzato per calcolare la distanza.

Gli studenti possono attivare il sensore posizionando la mano davanti al sensore e facendo diminuire il ritardo tra la trasmissione e la ricezione del suono. La distanza viene visualizzata sullo schermo seriale. Gli studenti possono vedere l'attivazione del semaforo osservando le luci.

Nel circuito, insieme al sensore a ultrasuoni, utilizziamo una resistenza di pull-up esterna.

- Il resistore di pull-up mantiene il pin 3 costantemente nello stato HIGH (+5V).
- Quando il sensore a ultrasuoni viene attivato, il pin 3 viene messo momentaneamente a terra (stato BASSO).

Tabella dei tempi

Fasi	Veicoli	Pedoni	Tempo	Descrizione
0				Il sensore non è stato attivato

1			3sec	Una volta attivato il sensore, vengono attivate le fasi da 1 a 4, quindi il dispositivo torna allo stato precedente (fase 0).
2			3sec	
3			4sec	
4			3sec	
0				Finché il sensore non viene nuovamente attivato

Elenco dei componenti:

Our components are:

- Scheda Arduino
- Sensore a ultrasuoni HC-SR04
- Breadboard e fili di collegamento
- Cavo USB
- Arduino UNO R3
- 5XLED
- 5X330 ohm

Codice Arduino del progetto:

```
//Project Name: ULTRASOVIC SENSOR IN TRAFFIC LIGHTS
// Traffic signal with pedestrian light and ultrasonic sensor
// Declaration of constants
  const byte greenCar = 6;
  const byte orangeCar = 7;
  const byte redCar = 8;
  const byte greenPedestrian = 12;
  const byte redPedestrian = 13;
//  Parameter declaration
  boolean buttonOn=false;

// Parameter declaration Ultrasonic Sensor HC-SR04 * ******
// defines pins numbers
  const int trigPin = 4;  //D4
  const int echoPin = 2;  //D3


void setup() {
 // Initialize constants Ultrasonic Sensor HC-SR04 * ******
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600); // Starts the serial communication
 // parameter initialization
```

```
  pinMode(greenCar,OUTPUT);
  pinMode(orangeCar,OUTPUT);
  pinMode(redCar,OUTPUT);

  pinMode(greenPedestrian,OUTPUT);
  pinMode(redPedestrian,OUTPUT);

  // Initialization of prices for pedestrian signaling
  digitalWrite(greenPedestrian,LOW);
  digitalWrite(redPedestrian,HIGH);

  // Initialization of prices for vehicle signaling
  digitalWrite(greenCar,HIGH);
  digitalWrite(orangeCar,LOW);
  digitalWrite(redCar,LOW);
  pinMode (2, INPUT_PULLUP);
}

void loop() {
 //****** * 3. Code Ultrasonic Sensor HC-SR04 * ******
 // Clears the trigPin
  delay(500);
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds
  const long duration = pulseIn(echoPin, HIGH);
  Serial.print("Duration: ");
  Serial.println(duration);
// Calculating the distance
  const long distance= duration/58.2;
//Prints the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.println(distance);

  delay(2000);
  if (distance<20){
     buttonOn=true;
  }
  if(buttonOn == true)
    {
     buttonOn = false;

     delay(3000);
     digitalWrite(greenCar,LOW);
```

```
    digitalWrite(orangeCar,HIGH);

    delay(3000);
    digitalWrite(orangeCar,LOW);
    digitalWrite(redCar,HIGH);

    digitalWrite(greenPedestrian,HIGH);
    digitalWrite(redPedestrian,LOW);

    delay(4000);
    digitalWrite(greenPedestrian,LOW);
    digitalWrite(redPedestrian,HIGH);

    delay(3000);
    digitalWrite(greenCar,HIGH);
    digitalWrite(redCar,LOW);

    delay(5000);
  }else{
    digitalWrite(greenPedestrian,LOW);
    digitalWrite(redPedestrian,HIGH);

    digitalWrite(greenCar,HIGH);
    digitalWrite(orangeCar,LOW);
    digitalWrite(redCar,LOW);
    }
}
```

PROGETTO 2 - NOME: SISTEMA DI ALLARME

Scopo del progetto: In questo progetto, gli studenti vedranno come funziona un sistema di allarme in grado di rilevare la presenza di gas liquido, il movimento in uno spazio chiuso e l'aumento ingiustificato della temperatura. Il sistema di allarme sarà controllato da un pulsante e da un sensore a fotoresistenza. Premendo il pulsante gli studenti possono attivare il sistema di allarme. Quando il cicalino suona, premendo il pulsante possono fermarlo. Quando associano un sensore attivato alla pressione del pulsante, possono disattivare il sistema di allarme. Il fotoresistore attiva il sistema di allarme nel caso in cui il sistema di allarme non sia attivato e l'illuminazione sia scarsa.

Attraverso questo progetto, gli studenti potrebbero sperimentare:

- un modulo sensore buzzer attivo che ha un circuito oscillatore incorporato che produce una frequenza sonora costante. Si accende e si spegne con un pin di Arduino
- un sensore a ultrasuoni, un dispositivo che misura la distanza da un oggetto utilizzando le onde sonore
- un sensore di gas e
- due indicatori luminosi e un pulsante.

- un sensore a fotoresistenza
- un sensore di temperature

Metodologia

In primo luogo, si descrive il flusso di lavoro desiderato del progetto da sviluppare in più fasi. Agli studenti viene quindi chiesto di selezionare i componenti necessari per progettare e disegnare il circuito. Utilizzeranno l'applicazione "Arduino" per scrivere il codice.

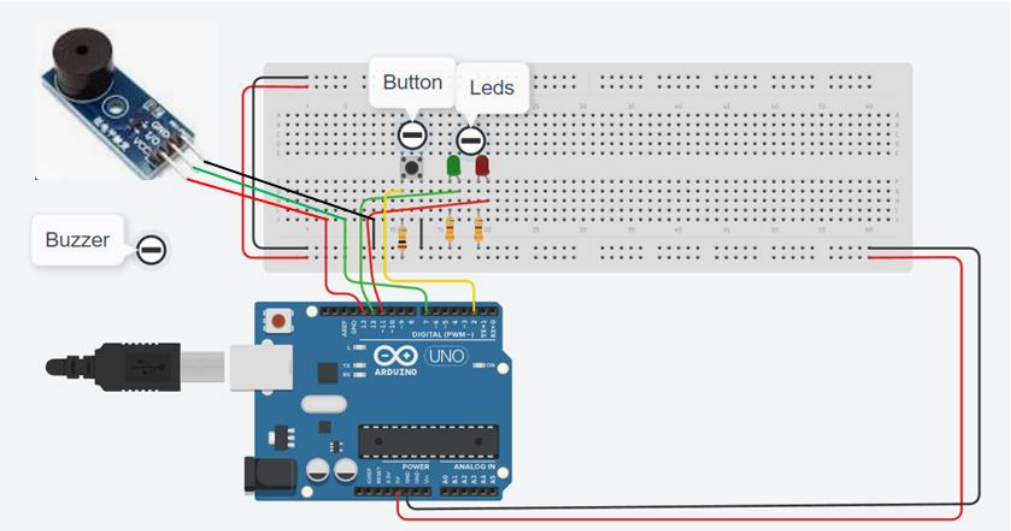
Quindi creeranno il circuito, scriveranno il codice sul software Arduino e lo caricheranno sulla scheda Arduino.

Gli studenti verificheranno che il circuito sia stato costruito correttamente.

Questo progetto è rivolto agli studenti del livello base, che iniziano a beneficiare dei risultati e degli esiti del nostro progetto Erasmus+ KA-202 Strategic Partnerships in VET, chiamato "ArduinVET".

Passo 1 (Cicalino con diverse tonalità di suono nel sistema di allarme)

Circuito di progetto.



Come funziona:

Un sensore buzzer attivo è dotato di un circuito oscillante incorporato, per cui la frequenza del suono è costante. È in grado di produrre il suono stesso.

All'interno di questo lavoro il tono del suono cessa di avere lo stesso suono attraverso il codice. Non appena il pulsante viene premuto, inizia a emettere un segnale acustico con un pin Arduino, il led rosso in attesa si spegne e il led verde si accende.

Nel circuito, insieme al sensore a ultrasuoni, utilizziamo una resistenza di pull-up esterna.

- Il resistore di pull-up mantiene il pin 3 permanentemente nello stato HIGH (+5V).
- Quando si preme il pulsante, il pin 3 viene messo momentaneamente a terra (stato BASSO).

Elenco di componenti:

I nostril component sono:

- Buzzer
- Breadboard e cavi di collegamento
- Cavo USB
- Arduino UNO R3
- 2XLED
- 2X330 ohm
- Pulsante
- 1X10Kohm

Codice Arduino del passo 1:

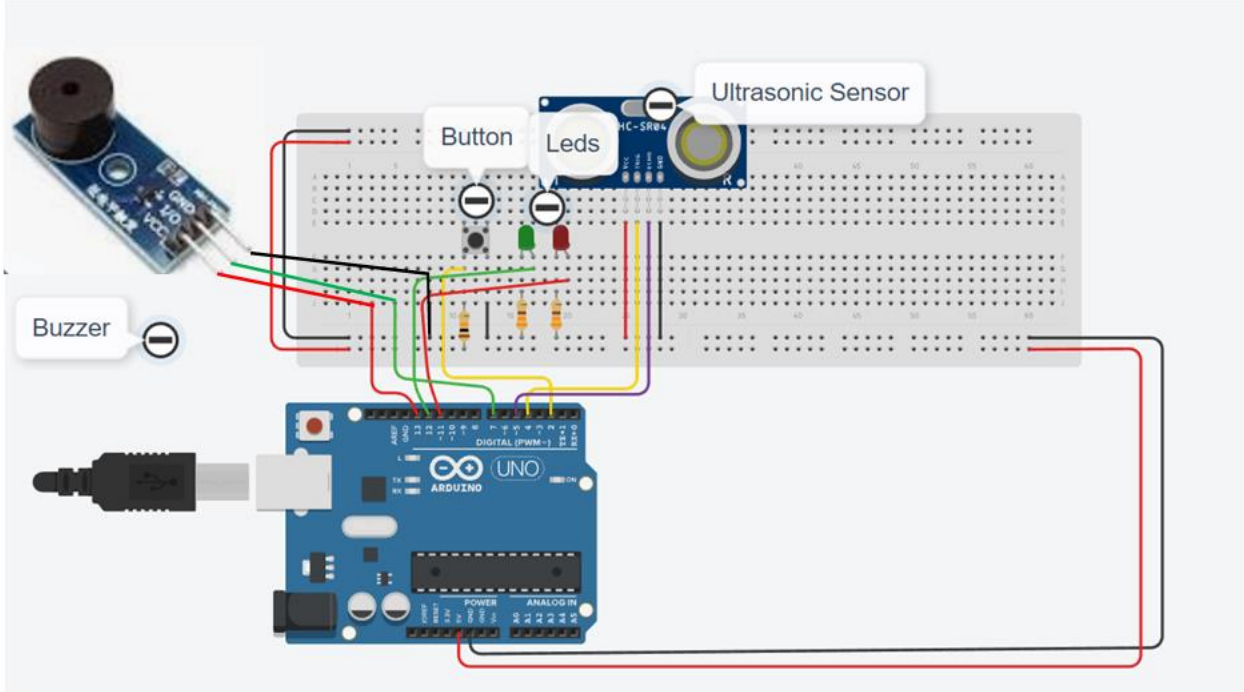
```
//Project Name: Alarm System

// Step 1  (Buzzer with different sound tone)

//Declaration-define of variables - constants
int buzz= 7; // I/O-pin from buzzer Module connects here
int buzzVcc= 13; //Vcc-pin from buzzer Module connects here
const int wpm = 20; // Morse speed in WPM
const int dotL = 1200/wpm; // Calculated dot-length
const int dashL = 3*dotL; // Dash = 3 x dot
const int sPause = dotL; // Symbol pause = 1 dot
const int lPause = dashL; // Letter pause = 3 dots
const int wPause = 7*dotL; // Word pause = 7 dots

int red_led=12;
int green_led=11;

boolean buttonOn=false;
void setup()
{
  // Initialization of variables - constants
  pinMode(buzz,OUTPUT);
```

```
  pinMode(buzzVcc,OUTPUT);

  pinMode(red_led,OUTPUT);
  pinMode(green_led,OUTPUT);

  pinMode (2, INPUT_PULLUP);
}

void loop(){
  if (digitalRead(2)== LOW){
     buttonOn=true;
     }

    //Buzzer
    if(buttonOn==true){

     digitalWrite(red_led, HIGH);
     digitalWrite(green_led, HIGH);

     digitalWrite(buzz, LOW); // Tone ON
     delay(dashL); // Tone length
     digitalWrite(buzz, HIGH); // Tone OFF
     delay(sPause); // Symbol pause
     digitalWrite(buzzVcc, HIGH);
     digitalWrite(buzz, LOW); // Tone ON
     delay(dotL); // Tone length
     digitalWrite(buzz, HIGH); // Tone OFF
     delay(sPause); // Symbol pause

     digitalWrite(buzz, LOW); // Tone ON
     delay(dashL); // Tone length
     digitalWrite(buzz, HIGH); // Tone OFF
     delay(sPause); // Symbol pause

     digitalWrite(buzz, LOW); // Tone ON
     delay(dotL); // Tone length
     digitalWrite(buzz, HIGH); // Tone OFF
     delay(sPause); // Symbol pause

     delay(lPause-sPause); // Subtracts pause already taken

     digitalWrite(buzz, LOW); // Tone ON
     delay(dashL); // Tone length
     digitalWrite(buzz, HIGH); // Tone OFF
     delay(sPause); // Symbol pause
     digitalWrite(buzz, LOW); // Tone ON
     delay(dashL); // Tone length
     digitalWrite(buzz, HIGH); // Tone OFF
     delay(sPause); // Symbol pause
```

```
    digitalWrite(buzz, LOW); // Tone ON
    delay(dotL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause

    digitalWrite(buzz, LOW); // Tone ON
    delay(dashL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause
    delay(wPause-sPause); // Subtracts pause already taken
    delay(5000);
    //Variable return to their original status
    buttonOn=false;
    digitalWrite(buzz, LOW); // Tone ON
    digitalWrite(buzzVcc, LOW);
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, LOW);
  }
}
```

Passo 2 (Aggiunto un sensore a ultrasuoni HC-SR04 nel Sistema di allarme)

Collegare il sensore a ultrasuoni HC-SR04



Come funziona:

Il sensore a ultrasuoni funziona inviando un'onda sonora a una frequenza ultrasonica e attende che rimbalzi sull'oggetto. Il ritardo tra la trasmissione del suono e la sua ricezione viene utilizzato per calcolare la distanza. Gli studenti possono attivare il sistema di allarme premendo il pulsante, quindi il led verde si accende; se posizionano la mano davanti al sensore, il ritardo tra la trasmissione e la ricezione del suono diminuisce e il led verde si spegne, si accende il led rosso e il cicalino audio inizia a suonare. La distanza verrà visualizzata sullo schermo seriale. Gli studenti possono anche disattivare il sistema di allarme premendo nuovamente il pulsante.

Codice Arduino del passo 2:

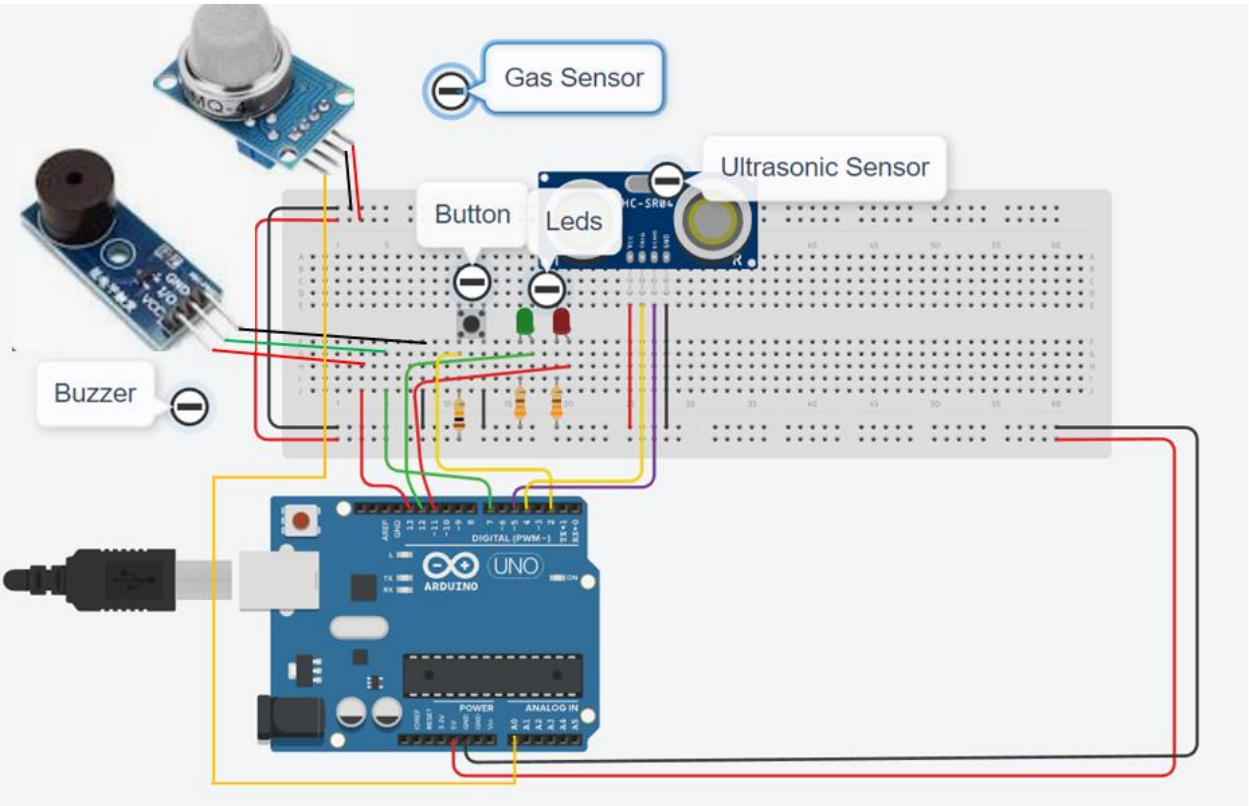
```
// code of Step 2 - Added Ultrasonic Sensor HC-SR04
int red_led=12;
int green_led=11;
int sensorThres=400;

//buzzer
int buzz= 13; // I/O-pin from buzzer connects here
int buzzVcc= 7; //Vcc-pin from buzzer connects here
const int wpm = 20; // Morse speed in WPM
const int dotL = 1200/wpm; // Calculated dot-length
const int dashL = 3*dotL; // Dash = 3 x dot
const int sPause = dotL; // Symbol pause = 1 dot
const int lPause = dashL; // Letter pause = 3 dots
const int wPause = 7*dotL; // Word pause = 7 dots

//Ultrasonic Sensor
#define trigPin 4
#define echoPin 5

//Button
boolean buttonOn=false;

void setup()
{
pinMode(red_led,OUTPUT);
pinMode(buzz,OUTPUT);
pinMode(green_led,OUTPUT);

pinMode(buzz,OUTPUT); // Set buzzer-pin as output
pinMode(buzzVcc,OUTPUT); //Vcc Buzzer

//Ultrasonic Sensor
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);

pinMode (2, INPUT_PULLUP);

Serial.begin(9600);
}
```

```
void loop()
{
//Button
 if (digitalRead(2)== LOW){
     buttonOn=true;
     digitalWrite(green_led, HIGH);
     }

//Ultrasoc Sensor code
long duration, distance;
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = (duration/2) / 29.1;

digitalWrite(buzzVcc, LOW);

if (distance < 20)
  {
  while(buttonOn==true){

   if (digitalRead(2)== LOW){
     buttonOn=false;
     digitalWrite(red_led, LOW);
     digitalWrite(green_led, LOW);
     digitalWrite( buzz, LOW);
  }else{
  digitalWrite(red_led, HIGH);
  digitalWrite(green_led, LOW);
  digitalWrite( buzz, HIGH);



  //buzzer
  digitalWrite(buzzVcc, HIGH);
  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  digitalWrite(buzz, LOW); // Tone ON
  delay(dotL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause
```

```
  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  digitalWrite(buzz, LOW); // Tone ON
  delay(dotL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  delay(lPause-sPause); // Subtracts pause already taken

  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause
  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  digitalWrite(buzz, LOW); // Tone ON
  delay(dotL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause
  delay(wPause-sPause); // Subtracts pause already taken

  }  //end else
  } //end while
} //end if

delay(100);
}
```

Passo 3 (Aggiunto un sensore di gas nel sistema di allarme)

Collegare il sensore di gas



Come funziona:

Quando il rilevatore di gas rileva del gas e il sistema di allarme è già attivato, la luce verde si spegne, la luce rossa si accende e il cicalino suona.

Codice Arduino del passo 3:

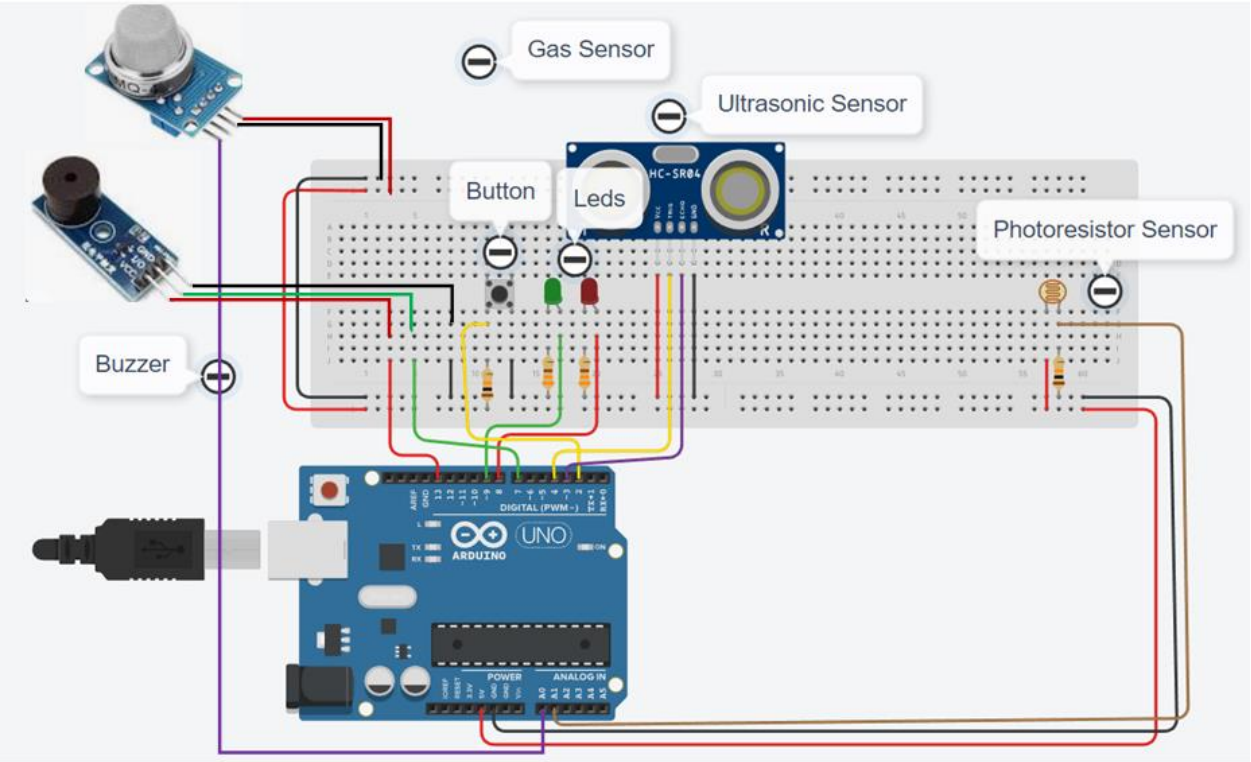
```
//WORK : ALARM SYSTEM
// ADDED AN GAS SENSOR IN SYSTEM
int red_led=11;
int green_led=12;
```

```
int gas_value;
int sensorThres=400;

//buzzer
int buzz= 13; // I/O-pin from buzzer connects here
int buzzVcc= 7; //Vcc-pin from buzzer connects here
const int wpm = 20; // Morse speed in WPM
const int dotL = 1200/wpm; // Calculated dot-length
const int dashL = 3*dotL; // Dash = 3 x dot
const int sPause = dotL; // Symbol pause = 1 dot
const int lPause = dashL; // Letter pause = 3 dots
const int wPause = 7*dotL; // Word pause = 7 dots

//Ultrasonic Sensor
#define trigPin 4
#define echoPin 5

//Button
boolean buttonOn=false;

void setup()
{
pinMode(red_led,OUTPUT);
pinMode(green_led,OUTPUT);
pinMode(A0,INPUT); //A0 Gass

pinMode(buzz,OUTPUT); // Set buzzer-pin as output
pinMode(buzzVcc,OUTPUT); //Vcc Buzzer

//Ultrasonic Sensor
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);

pinMode (2, INPUT_PULLUP);

Serial.begin(9600);
}

void loop()
{
//Button
 if (digitalRead(2)== LOW){
     buttonOn=true;
     digitalWrite(green_led, HIGH);
     Serial.println("The alarm system is enabled!");
     }

    //Ultrasoc Sensor code
long duration, distance;
digitalWrite(trigPin, LOW);
```

```
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = (duration/2) / 29.1;


digitalWrite(buzzVcc, LOW);

gas_value=analogRead(A0);

if (gas_value > sensorThres or distance < 20)
 {
  while(buttonOn==true){
  if (digitalRead(2)== LOW){
     buttonOn=false;
     digitalWrite(red_led, LOW);
     digitalWrite(green_led, LOW);
     digitalWrite( buzz, LOW);
     Serial.println("The alarm system is disabled!");
 }else{
 digitalWrite(red_led, HIGH);
 digitalWrite(green_led, LOW);
 digitalWrite( buzz, HIGH);
 Serial.println("DANGER!!!!");
 Serial.println(gas_value);

 //buzzer
 digitalWrite(buzzVcc, HIGH);
 digitalWrite(buzz, LOW); // Tone ON
 delay(dashL); // Tone length
 digitalWrite(buzz, HIGH); // Tone OFF
 delay(sPause); // Symbol pause

 digitalWrite(buzz, LOW); // Tone ON
 delay(dotL); // Tone length
 digitalWrite(buzz, HIGH); // Tone OFF
 delay(sPause); // Symbol pause

 digitalWrite(buzz, LOW); // Tone ON
 delay(dashL); // Tone length
 digitalWrite(buzz, HIGH); // Tone OFF
 delay(sPause); // Symbol pause

 digitalWrite(buzz, LOW); // Tone ON
 delay(dotL); // Tone length
 digitalWrite(buzz, HIGH); // Tone OFF
 delay(sPause); // Symbol pause
```

```
  delay(lPause-sPause); // Subtracts pause already taken

  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause
  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  digitalWrite(buzz, LOW); // Tone ON
  delay(dotL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause
  delay(wPause-sPause); // Subtracts pause already taken
        } //end else
   }//end while
  } //end if

delay(100);
}
```

Passo 4 (Aggiunto un sensore a fotoresistenza nel sistema di allarme)



Collegare il sensore a fotoresistenza

Come funziona:

Il sistema di allarme si attiva quando la luminosità diminuisce.

Codice Arduino del passo 4:

```
//WORK : ALARM SYSTEM
//ADDED A PHOTORESISTOR SENSOR IN SYSTEM
int red_led=8;
int green_led=9;
int gas_value;
int sensorThres=400;

//------------------buzzer-----------------------
int buzz= 13; // I/O-pin from buzzer connects here
int buzzVcc= 7; //Vcc-pin from buzzer connects here
const int wpm = 20; // Morse speed in WPM
const int dotL = 1200/wpm; // Calculated dot-length
const int dashL = 3*dotL; // Dash = 3 x dot
const int sPause = dotL; // Symbol pause = 1 dot
const int lPause = dashL; // Letter pause = 3 dots
const int wPause = 7*dotL; // Word pause = 7 dots
```

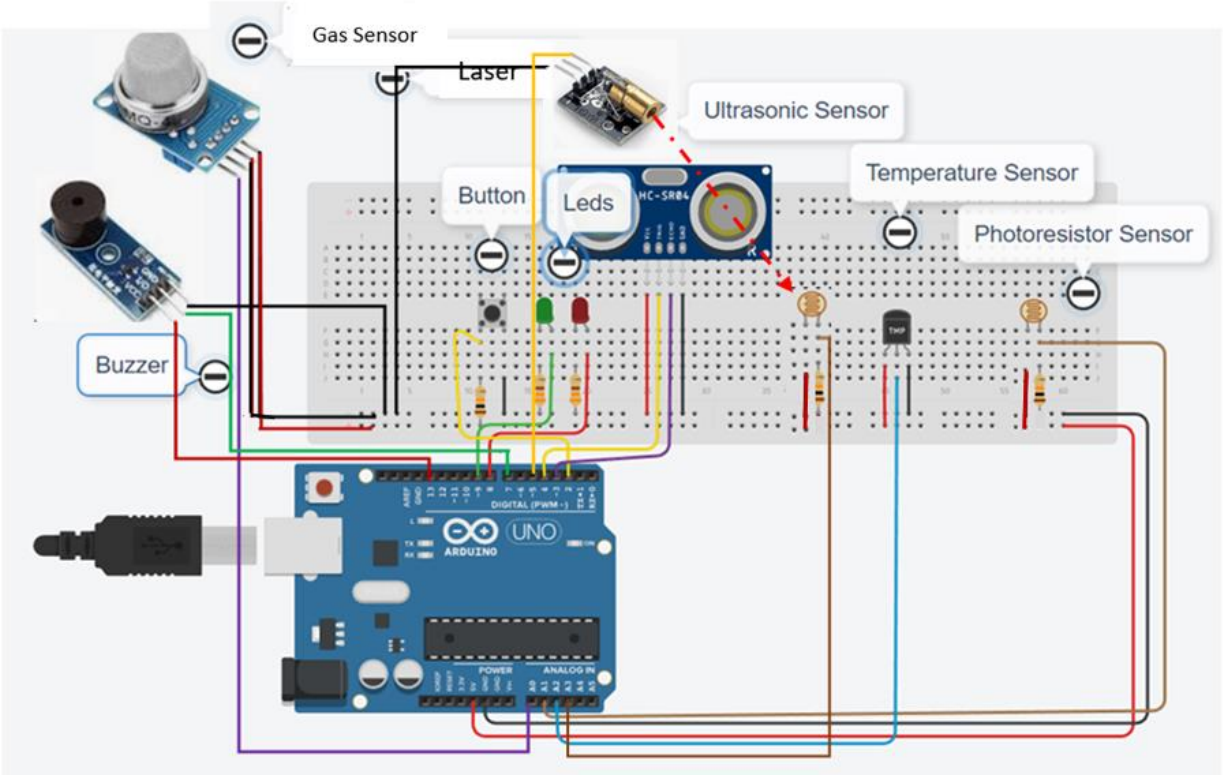
```
//------------Ultrasonic Sensor--------------------
#define trigPin 4
#define echoPin 5

//------------------Button-------------------------
boolean buttonOn=false;

void setup()
{
pinMode(red_led,OUTPUT);
pinMode(green_led,OUTPUT);
digitalWrite(red_led, LOW);
digitalWrite(green_led, LOW);
pinMode(A0,INPUT); //A0 Gass

pinMode(buzz,OUTPUT); // Set buzzer-pin as output
pinMode(buzzVcc,OUTPUT); //Vcc Buzzer

//Ultrasonic Sensor
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);

pinMode (2, INPUT_PULLUP);

Serial.begin(9600);
}
void loop()
{
  //-------------photoresistor----------------
  int valuePhotor = analogRead(A1);
   Serial.println("Analog valuePhotor : ");
   Serial.println(valuePhotor);
   delay(250);

  //------------------Button------------------
   if (digitalRead(2)== LOW or valuePhotor<110){
       buttonOn=true;
       digitalWrite(red_led, LOW);
       digitalWrite(green_led, HIGH);
       }

  //-------------Ultrasoc Sensor--------------
  long duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
```

```
distance = (duration/2) / 29.1;
Serial.println("Distance");
Serial.println(distance);

digitalWrite(buzzVcc, LOW);

gas_value=analogRead(A0);

if (gas_value > sensorThres or distance < 20){
  while(buttonOn==true){
    if (digitalRead(2)== LOW){
      buttonOn=false;
      digitalWrite(red_led, LOW);
      digitalWrite(green_led, LOW);
      digitalWrite( buzz, LOW);
      Serial.println("NO LEAKAGE");
      Serial.println(gas_value);
   }else{
   digitalWrite(red_led, HIGH);
   digitalWrite(green_led, LOW);
   digitalWrite( buzz, HIGH);
   Serial.println("DANGER!!!!");
   Serial.print("Gas Value: ");
   Serial.println(gas_value);
   Serial.print("Distance: ");
   Serial.println(distance);

   //buzzer
   digitalWrite(buzzVcc, HIGH);
   digitalWrite(buzz, LOW); // Tone ON
   delay(dashL); // Tone length
   digitalWrite(buzz, HIGH); // Tone OFF
   delay(sPause); // Symbol pause

   digitalWrite(buzz, LOW); // Tone ON
   delay(dotL); // Tone length
   digitalWrite(buzz, HIGH); // Tone OFF
   delay(sPause); // Symbol pause

   digitalWrite(buzz, LOW); // Tone ON
   delay(dashL); // Tone length
   digitalWrite(buzz, HIGH); // Tone OFF
   delay(sPause); // Symbol pause

   digitalWrite(buzz, LOW); // Tone ON
   delay(dotL); // Tone length
   digitalWrite(buzz, HIGH); // Tone OFF
   delay(sPause); // Symbol pause

   delay(lPause-sPause); // Subtracts pause already taken
```

```
    digitalWrite(buzz, LOW); // Tone ON
    delay(dashL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause
    digitalWrite(buzz, LOW); // Tone ON
    delay(dashL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause

    digitalWrite(buzz, LOW); // Tone ON
    delay(dotL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause

    digitalWrite(buzz, LOW); // Tone ON
    delay(dashL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause
    delay(wPause-sPause); // Subtracts pause already taken
    } //end else
  } //end while
 } //end if
 delay(100);
}
```

Passo 5 (Aggiunta di un sensore di temperatura e di un modulo trasmettitore laser nel sistema d'allarme.)

Collegare il sensore di temperatura



In questa fase avremo bisogno di un trasmettitore laser, di un fotoresistore per il ricevitore e di un resistore da 1KOhm per proteggerlo.

Come funziona:

Il trasmettitore laser, insieme al sensore a fotoresistenza che funziona come ricevitore, istruisce Arduino ad accendere il sistema quando per qualche motivo la ricezione viene interrotta. Inoltre, il sistema di allarme si attiva quando il valore del sensore di temperatura supera un certo valore.

Codice Arduino del Passo 5:

```
//WORK : ALARM SYSTEM
//ADDED A TEMPERATURE SENSOR IN SYSTEM
int red_led=8;
int green_led=9;
int gas_value;
int sensorThres=400;

//----------------buzzer--------------------------
int buzz= 13; // I/O-pin from buzzer connects here
int buzzVcc= 7; //Vcc-pin from buzzer connects here
```

```
const int wpm = 20; // Morse speed in WPM
const int dotL = 1200/wpm; // Calculated dot-length
const int dashL = 3*dotL; // Dash = 3 x dot
const int sPause = dotL; // Symbol pause = 1 dot
const int lPause = dashL; // Letter pause = 3 dots
const int wPause = 7*dotL; // Word pause = 7 dots

//-------------Ultrasonic Sensor------------------
#define trigPin 3
#define echoPin 4

//----------------Button-------------------------
boolean buttonOn=false;

//-----------temperature sensor-------------------
float tempf;
int tempPin = A2;

//-----------Laser--------------------
int Laser = 5; // creating a variable named Laser which is assigned to digital pin 5
int valueLaserPh=0;// creating a variable named valueLaserPh and setting is value to zero
float valueLaserPhF=0;// creating a variable named valueLaserPhF and setting is value to zero

// room temperature in Fa
const float baselineTemp = 100.0;

void setup()
{
  pinMode(red_led,OUTPUT);
  pinMode(buzz,OUTPUT);
  pinMode(green_led,OUTPUT);
  pinMode(A0,INPUT); //A0 Gas

  pinMode(buzz,OUTPUT); // Set buzzer-pin as output
  pinMode(buzzVcc,OUTPUT); //Vcc Buzzer

  //Ultrasonic Sensor
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

//-----------Laser--------------------
  pinMode (Laser,OUTPUT); // designating digital pin 5 for output
  digitalWrite(Laser,LOW); // just making sure the laser is off at startup or reset


  pinMode (2, INPUT_PULLUP);

  //temperature sensor
  pinMode(tempPin,INPUT);
```

```
  Serial.begin(9600);
}


void loop()
{
  digitalWrite(red_led, LOW);
  digitalWrite(Laser,HIGH);
  //-----------------Laser-------------------------------
  valueLaserPh=analogRead(A4); //reading the voltage on A4 and storing the value received in
"voltage"
  valueLaserPhF = valueLaserPh * (5.0 / 1023.0); // transforming the value stored in "voltage"
to readable information

  //----------temperature sensor---------------------------
  tempf = analogRead(tempPin);
  tempf=(tempf*5)/10;
  // convert the analog volt to its temperature equivalent
  Serial.print("TEMPERATURE = ");
  Serial.print(tempf); // display temperature value
  Serial.print("*F");
  Serial.println();
  delay(1000); // update sensor reading each one second

 //------------photoresistor------------------
  int valuePhotor = analogRead(A1);
  Serial.print("Photoresistor value : ");
  Serial.println(valuePhotor);
  delay(250);

//-----------------Button----------------------
 if (digitalRead(2)== LOW or valuePhotor<20){
     buttonOn=true;
     digitalWrite(red_led, LOW);
     digitalWrite(green_led, HIGH);
     digitalWrite(Laser,HIGH); // turning the laser on
     Serial.println("The alarm system is enabled!");
     }

//------------Ultrasoc Sensor -----------
long duration, distance;
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = (duration/2) / 29.1;
Serial.print("distance : ");
Serial.println(distance);
```

```
digitalWrite(buzzVcc, LOW);
gas_value=analogRead(A0);
Serial.print("Gas Value : ");
Serial.println(gas_value);
Serial.print("Laser value : ");
Serial.println(valueLaserPhF);

if (gas_value > sensorThres or distance < 10 or tempf>baselineTemp or valueLaserPhF<1)
 {
  while(buttonOn==true){
   if (digitalRead(2)== LOW){
     buttonOn=false;
     //digitalWrite(red_led, LOW);
     digitalWrite(green_led, LOW);
     digitalWrite( buzz, LOW);
     Serial.println("The alarm system is disabled!");
     distance=100;
     tempf=0.0;
  }else{
       digitalWrite(red_led, HIGH);
       digitalWrite(green_led, LOW);
       digitalWrite( buzz, HIGH);
       Serial.println("DANGER!!!!");
       Serial.print("Gas Value: ");
       Serial.println(gas_value);
       Serial.print("Distance: ");
       Serial.println(distance);
       Serial.print("Temperature: ");
       Serial.println(tempf);
       Serial.print("Photoresistor value : ");
       Serial.println(valuePhotor);
       Serial.print("Laser value : ");
       Serial.println(valueLaserPhF);

       //--------------buzzer--------------
       digitalWrite(buzzVcc, HIGH);
       digitalWrite(buzz, LOW); // Tone ON
       delay(dashL); // Tone length
       digitalWrite(buzz, HIGH); // Tone OFF
       delay(sPause); // Symbol pause

       digitalWrite(buzz, LOW); // Tone ON
       delay(dotL); // Tone length
       digitalWrite(buzz, HIGH); // Tone OFF
       delay(sPause); // Symbol pause

       digitalWrite(buzz, LOW); // Tone ON
       delay(dashL); // Tone length
       digitalWrite(buzz, HIGH); // Tone OFF
       delay(sPause); // Symbol pause
```

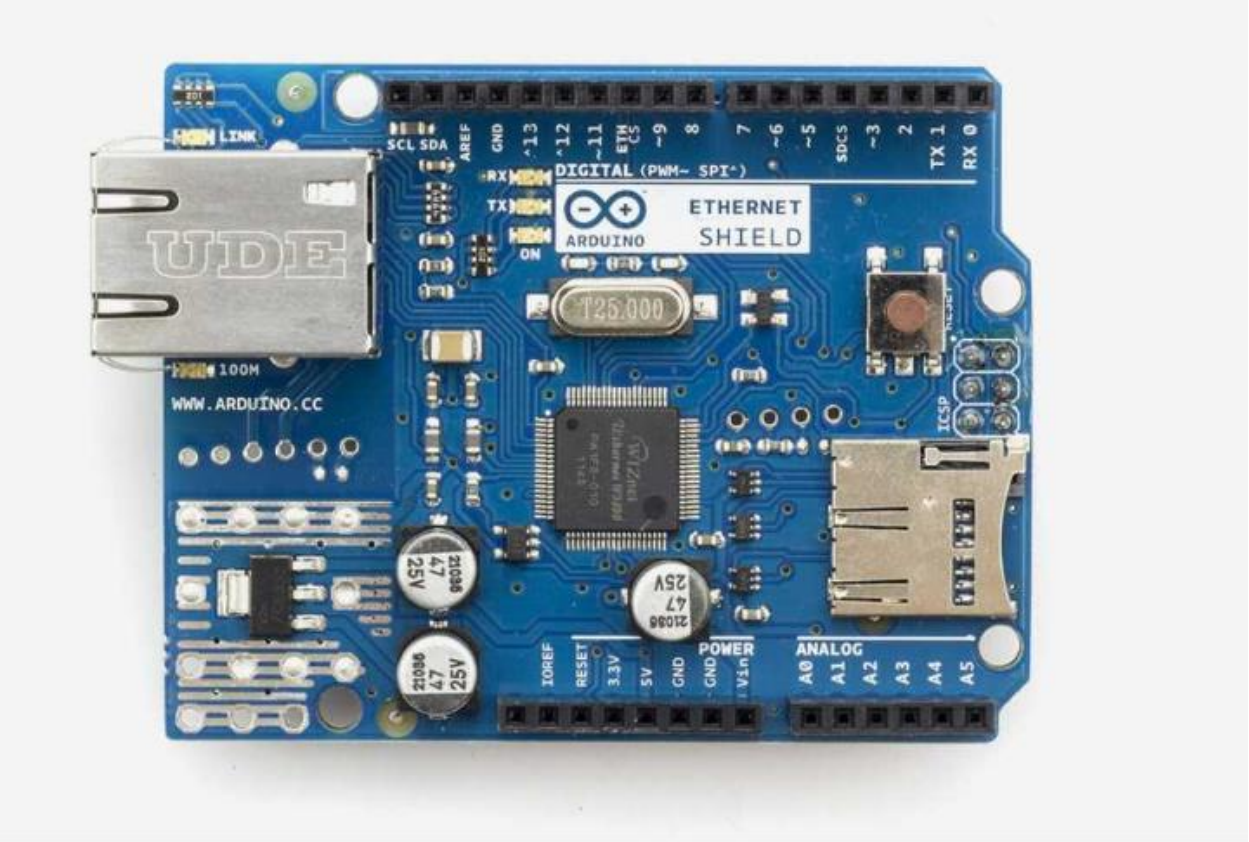
```
      digitalWrite(buzz, LOW); // Tone ON
      delay(dotL); // Tone length
      digitalWrite(buzz, HIGH); // Tone OFF
      delay(sPause); // Symbol pause

      delay(lPause-sPause); // Subtracts pause already taken

      digitalWrite(buzz, LOW); // Tone ON
      delay(dashL); // Tone length
      digitalWrite(buzz, HIGH); // Tone OFF
      delay(sPause); // Symbol pause
      digitalWrite(buzz, LOW); // Tone ON
      delay(dashL); // Tone length
      digitalWrite(buzz, HIGH); // Tone OFF
      delay(sPause); // Symbol pause

      digitalWrite(buzz, LOW); // Tone ON
      delay(dotL); // Tone length
      digitalWrite(buzz, HIGH); // Tone OFF
      delay(sPause); // Symbol pause

      digitalWrite(buzz, LOW); // Tone ON
      delay(dashL); // Tone length
      digitalWrite(buzz, HIGH); // Tone OFF
      delay(sPause); // Symbol pause
      delay(wPause-sPause); // Subtracts pause already taken
     }//end else
   }//end while
  } //end if
delay(100);
}
```

PROGETTO 3- NOME: WebServer

Scopo del progetto: In questo progetto, gli studenti vedranno come collegare un arduino a una rete locale utilizzando uno shield ethernet. Sperimenteranno con i sensori e potranno vedere i valori ricevuti sullo schermo del computer attraverso la rete locale.

Attraverso questo progetto, gli studenti hanno potuto sperimentare:

- Un sensore di gas
- Un sensore a fotoresistenza
- Un sensore di temperatura



Questo shield Ethernet consente a una scheda Arduino di connettersi a Internet.

Elenco dei componenti:

- Shield Ethernet
- Cavo Ethernet
- Breadboard e fili di collegamento
- Cavo USB
- Arduino UNO R3
- un sensore a fotoresistenza
- un sensore di gas
- un sensore di temperatura

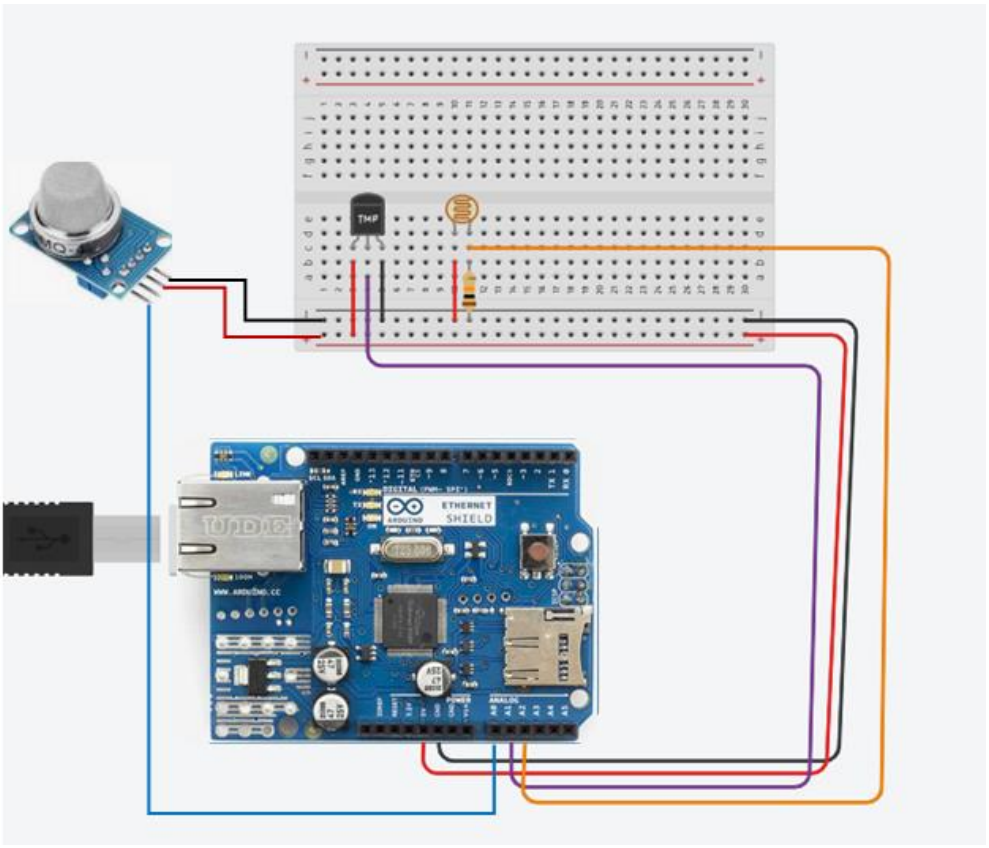
- un resistore 10Kohm

Come funziona:

L'Arduino Ethernet Shield 2 consente a una scheda Arduino di collegarsi a Internet. Si basa sul chip Ethernet Wiznet W5500. Wiznet W5500 fornisce uno stack di rete (IP) in grado di gestire sia TCP che UDP. Supporta fino a otto connessioni socket simultanee.

In questo esempio gli studenti cercheranno l'IP dello shield Ethernet in un browser di rete locale e potranno vedere i valori dei sensori che hanno collegato allo shield Ethernet in una pagina amichevole. La pagina si aggiorna automaticamente ogni 5 secondi.

Circuito di progetto



Metodologia

In primo luogo, si descrive il flusso di lavoro desiderato del progetto da sviluppare in più fasi. Agli studenti viene quindi chiesto di selezionare i componenti necessari per progettare e disegnare il circuito. Utilizzeranno l'applicazione "Arduino" per scrivere il codice.

Quindi creeranno il circuito, scriveranno il codice sul software Arduino e lo caricheranno sulla scheda Arduino.

Gli studenti verificheranno che il circuito sia stato costruito correttamente.

Questo progetto è rivolto agli studenti del livello base, che iniziano a beneficiare dei risultati e degli esiti del nostro progetto Erasmus+ KA-202 Strategic Partnerships in VET, chiamato "ArduInVet".

Codice Arduino del Progetto:

```
/*
  Web Server

 A simple web server that shows the value of the analog input pins.
 using an Arduino Wiznet Ethernet shield.

 Circuit:
 * Ethernet shield attached to pins 10, 11, 12, 13
 * Analog inputs attached to pins A0 through A5 (optional)

 */

#include <SPI.h>
#include <Ethernet.h>
unsigned long refreshCounter  = 0;
// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xA8, 0x61, 0x0A, 0xAE, 0x63, 0xA8
};
IPAddress ip(169, 254, 122, 108);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

EthernetClient client;
void setup() {
  // You can use Ethernet.init(pin) to configure the CS pin
  //Ethernet.init(10);  // Most Arduino shields
  //Ethernet.init(5);   // MKR ETH shield
  //Ethernet.init(0);   // Teensy 2.0
```

```
  //Ethernet.init(20);  // Teensy++ 2.0
  //Ethernet.init(15);  // ESP8266 with Adafruit Featherwing Ethernet
  //Ethernet.init(33);  // ESP32 with Adafruit Featherwing Ethernet

  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
        ; // wait for serial port to connect. Needed for native USB port only
  }
  Serial.println("ARDUinVET in Ethernet WebServer ");

  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);

  // Check for Ethernet hardware present
  if (Ethernet.hardwareStatus() == EthernetNoHardware) {
    Serial.println("Ethernet shield was not found.  Sorry, can't run without hardware. :(");
        while (true) {
              delay(1); // do nothing, no point running without Ethernet hardware
        }
  }
  if (Ethernet.linkStatus() == LinkOFF) {
        Serial.println("Ethernet cable is not connected.");
  }
  // start the server
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}

void loop() {
     // listen for incoming clients
    client = server.available();
    if (client) {
         Serial.println("ARDUinVET in client");
        // an http request ends with a blank line
        boolean currentLineIsBlank = true;
        while (client.connected()) {
            if (client.available()) {
                char c = client.read();
               Serial.write(c);
                // if you've gotten to the end of the line (received a newline
               // character) and the line is blank, the http request has ended,
               // so you can send a reply
                   if (c == '\n' && currentLineIsBlank) {
                        // send a standard http response header
                        client.println("HTTP/1.1 200 OK");
                        client.println("ARDinVET - HTTP");
                        client.println("Content-Type: text/html");
```

```
                client.println("Connection: close");  // the connection will be closed
after       //completion of the response
                client.println("Refresh: 5");  // refresh the page automatically every 5 sec
                client.println();
                webpage(client);
                break;
                }
                if (c == '\n') {
                      // you're starting a new line
                      currentLineIsBlank = true;
                      } else if (c != '\r') {
                            // you've gotten a character on the current line
                            currentLineIsBlank = false;
                        }
                }
            }
            // give the web browser time to receive the data
            delay(1);
            // close the connection:
            client.stop();
            Serial.println("client disconnected");
      }
}

void webpage(EthernetClient client) { /* function webpage */
////Send wbepage to client
client.println("ARDUinVET - HTTP");
//client.println("Content-Type: text/html");
client.println(""); //  do not forget this one
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<head>");
client.println("<title>ARDUinVET</title>");
//client.println("<script src='https://smtpjs.com/v3/smtp.js'></script>");
//client.println("<script>");
//client.println("function sendEmail() {");
/*client.println("Email.send({");
client.println("  Host: 'smtp.gmail.com',");
client.println("  Username : '**********@gmail.com',");
client.println("  Password : '******************',");
client.println("  To : '**********@gmail.com',");
client.println("  From : '**********@gmail.com',");
client.println("  Subject : 'Data from Arduino',");
client.println("  Body : 'Sensors value....',");
client.println("  }).then(");
client.println("    message => alert('mail sent successfully')");
client.println("  );");*/
//client.println("}");
//client.println("</script>");
client.println("</head>");
```

```
client.println("<body bgcolor = '#cccc00'>");
client.println("<hr/><hr>");
client.println("<h1 style='color : #0000cc;'><center>  Alarm System </center></h1>");
client.println("<hr/><hr>");
client.println("<br><br>");
client.println("<br><br>");
client.println("<center>");
client.println("<br>Sensors Output<br>");
client.println("Gas.....:");
client.println("  <input value=" + String(analogRead(A0)) + " readonly></input>");
client.println("<br>");
client.println("  Photoresistor..:");
client.println("  <input value=" + String(analogRead(A1)) + " readonly></input>");
client.println("<br>");
client.println("  Temperature....:");
client.println("  <input value=" + String(analogRead(A2)) + " readonly></input>");
client.println("<br>");
client.println("  </center>");
client.println("<br><br>");
client.println("<center>");
client.println("<a style='color : #0000cc;'
href='https://www.arduinvet.com/'>www.arduinvet.com</a>");
client.println("</center>");
client.println("<br><br>");
client.println("</body></html>");
client.println();
delay(1);
}
```

una schermata della pagina di progetto

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Titolo Progetto: "Teaching and Learning Arduinos in Vocational Training"

Acronimo Progetto: " ARDUinVET "

N°Progetto: "2020-1-TR01-KA202-093762"

Progetti FreeArduino

(Questi progetti sono stati realizzati dagli studenti delle scuole partner nell'ambito del partenariato ARDUinVET).

NO:	NOME DEL PROGETTO	Paese
1	CAPPELLO DI ALLONTANAMENTO SOCIALE	TURCHIA
2	MISURATORE DI TEMPERATURA (GRECIA)	GRECIA
3	LAMPEGGIATORE PER FRENI DI EMERGENZA - LUCE FRENO ADATTIVA	ROMANIA
4	LINEA DI SORVEGLIANZA	AUSTRIA
5	SERRA AUTOMATIZZATA (ITALIA)	ITALIA

1_NOME DEL PROGETTO: CAPPELLO DI DISTANZIAMENTO SOCIALE (TURCHIA)

Scopo del progetto: In questi giorni, in cui la pandemia di Covid-19 è in atto, seguire la regola della "DISTANZA SOCIALE" è una delle condizioni fondamentali per non contrarre la malattia.

Per attirare l'attenzione sul tema del "social distaning", decidiamo di realizzare un progetto Arduino sull'argomento.

La regola viene ricordata con il "cappello di allontanamento sociale", che prepareremo. Il cappello emette un segnale acustico e contemporaneamente un avviso luminoso.

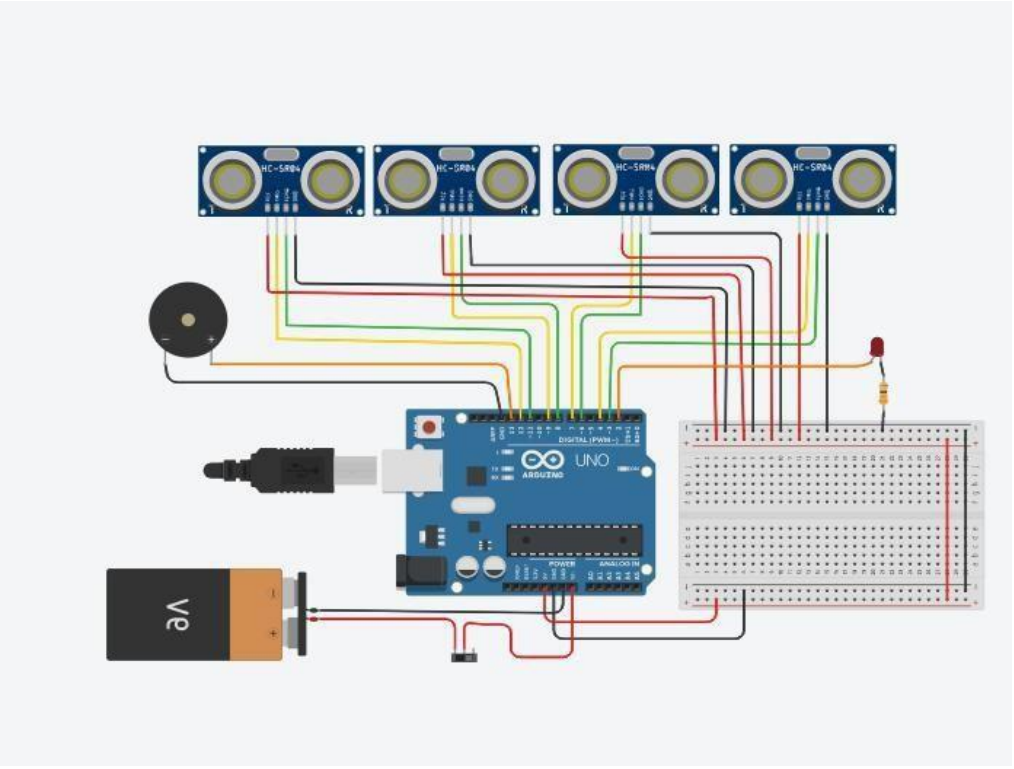
Il nostro metodo di progetto:

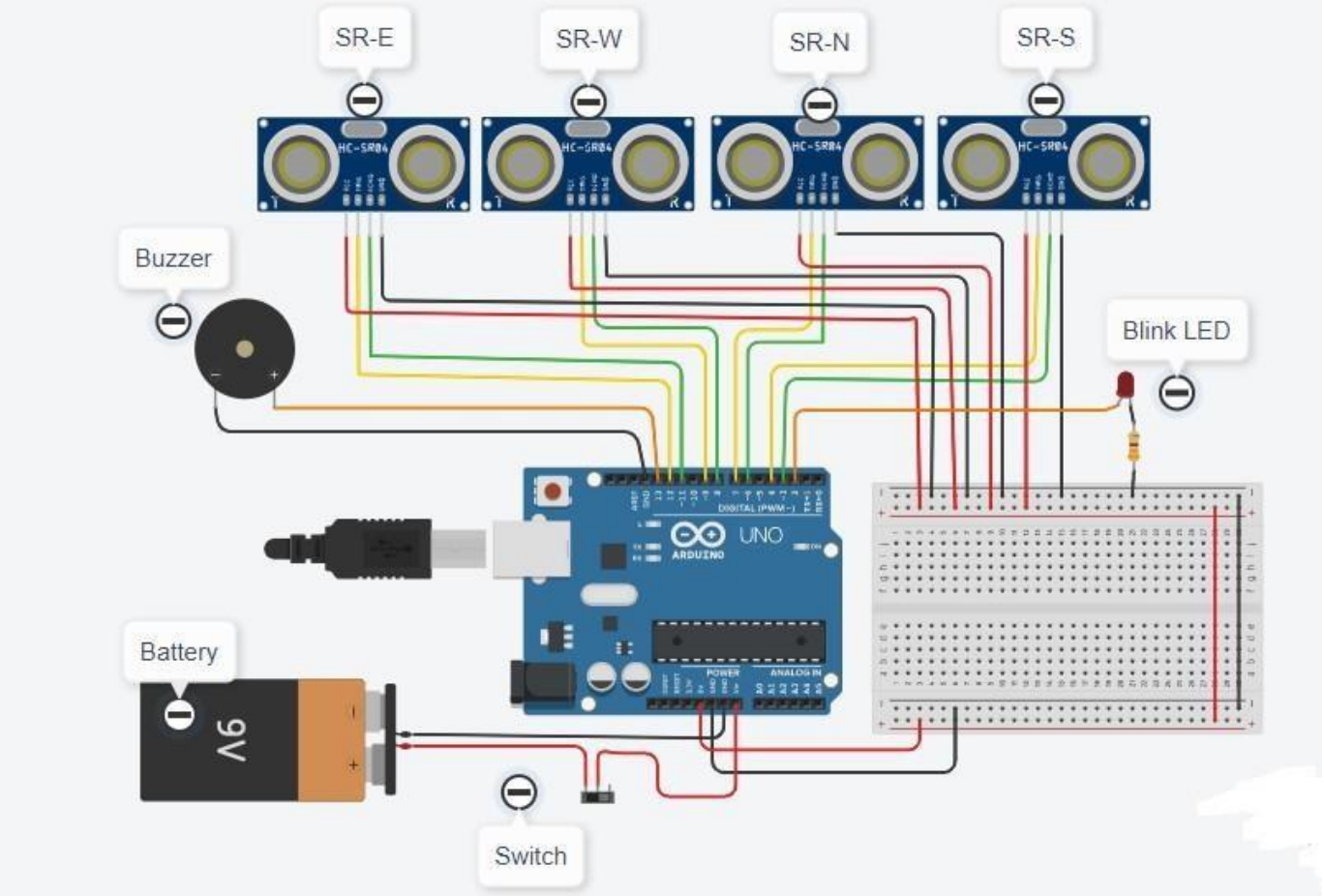
In primo luogo, viene preparato l'algoritmo del progetto. In base all'algoritmo preparato, si realizza il progetto logico e si scrive il programma Arduino. Arduino viene utilizzato per la progettazione del circuito. Durante la realizzazione di questi progetti, i nostri studenti beneficiano di tutti i risultati e i prodotti del progetto Erasmus+ KA-202 VET, denominato "ArduinVET".

Il nostro circuito di progettazione definitivo viene messo a punto dopo aver fornito l'attrezzatura necessaria e aver programmato Arduino. Il nostro circuito di progettazione finale viene posizionato visivamente sul tappo.

Questo progetto viene esposto alle fiere e ricorda ai visitatori la regola del "Social Distancing" e la sua importanza.

Circuito di progetto.





Come funziona:

4 sensori di distanza a ultrasuoni sono posizionati in 4 direzioni del cappello. Quando i sensori rilevano la presenza di una persona in avvicinamento o in prossimità di 100 cm o meno, il cicalino suona e il led lampeggia. Il circuito sarà alimentato da una batteria da 9V.

Elenco dei materiali:

Un cappello, un Arduino Uno R3, 4 sensori di distanza a ultrasuoni, un cicalino, una resistenza da 330 Ω, un LED rosso lampeggiante, una batteria da 9 V, un interruttore a scorrimento.

Programma Arduino del progetto:

//Nome del progetto: CAPPELLO DI DISTANZIAMENTO SOCIALE:

```
int trigPin=12;
int echoPin=11;
int trigPin2=9;
int echoPin2=8;
int trigPin3=7;
int echoPin3=6;
int trigPin4=4;
int echoPin4=3;
void setup()
{
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(trigPin2, OUTPUT);
pinMode(echoPin2, INPUT);
pinMode(trigPin3, OUTPUT);
pinMode(echoPin3, INPUT);
pinMode(trigPin4, OUTPUT);
pinMode(echoPin4, INPUT);
pinMode(13,OUTPUT);
pinMode(2,OUTPUT);
Serial.begin(9600);
}
void loop() {
 digitalWrite(trigPin, LOW);
```

```
delay(3);
digitalWrite(trigPin, HIGH);
delay(3);
digitalWrite(trigPin, LOW);
int time1 = pulseIn(echoPin, HIGH);
int distanza1 = (tempo1/2) / 28,97;      //La distanza      sociale è uguale e inferiore a 100 cm.
//.....
 digitalWrite(trigPin2, LOW);
delay(3);
digitalWrite(trigPin2, HIGH);
delay(3);
digitalWrite(trigPin2, LOW);

int time2 = pulseIn(echoPin2, HIGH);
int distanza2 = (tempo2/2) / 28,97;   //La distanza          sociale è uguale e inferiore a 100 cm.
//.....
 digitalWrite(trigPin3, LOW);
delay(3);
digitalWrite(trigPin3, HIGH);
delay(3);
digitalWrite(trigPin3, LOW);
int time3 = pulseIn(echoPin3, HIGH);
int distanza3 = (tempo3/2) / 28,97;      //La distanza      sociale è uguale e inferiore a 100 cm.
//.....
digitalWrite(trigPin4, LOW);
delay(3);
```

```
digitalWrite(trigPin4, HIGH);
delay(3);
digitalWrite(trigPin4, LOW);
int time4 = pulseIn(echoPin4, HIGH);
int distanza4 = (tempo4/2) / 28,97;       //La distanza     sociale è uguale e inferiore a 100 cm.
//.....
if(distanza1<75)
 {
  digitalWrite(13,HIGH);
  digitalWrite(2,HIGH);
  }
 altrimenti se(distanza2<75)
 {
  digitalWrite(13,HIGH);
   digitalWrite(2,HIGH);
  }
 altrimenti se(distanza3<75)
 {
  digitalWrite(13,HIGH);
   digitalWrite(2,HIGH);
  }
 altrimenti se(distanza4<75)
 {
  digitalWrite(13,HIGH);
   digitalWrite(2,HIGH);
  }
```

```
altro
{
digitalWrite(13,LOW);
digitalWrite(2,LOW);
}
 Serial.println(distanza1);      // Per vedere quale sensore ultrosonc è attivato sul monitor seriale
 Serial.println(distanza2);
 Serial.println(distanza3);
 Serial.println(distanza4);
 delay(500);
}
```

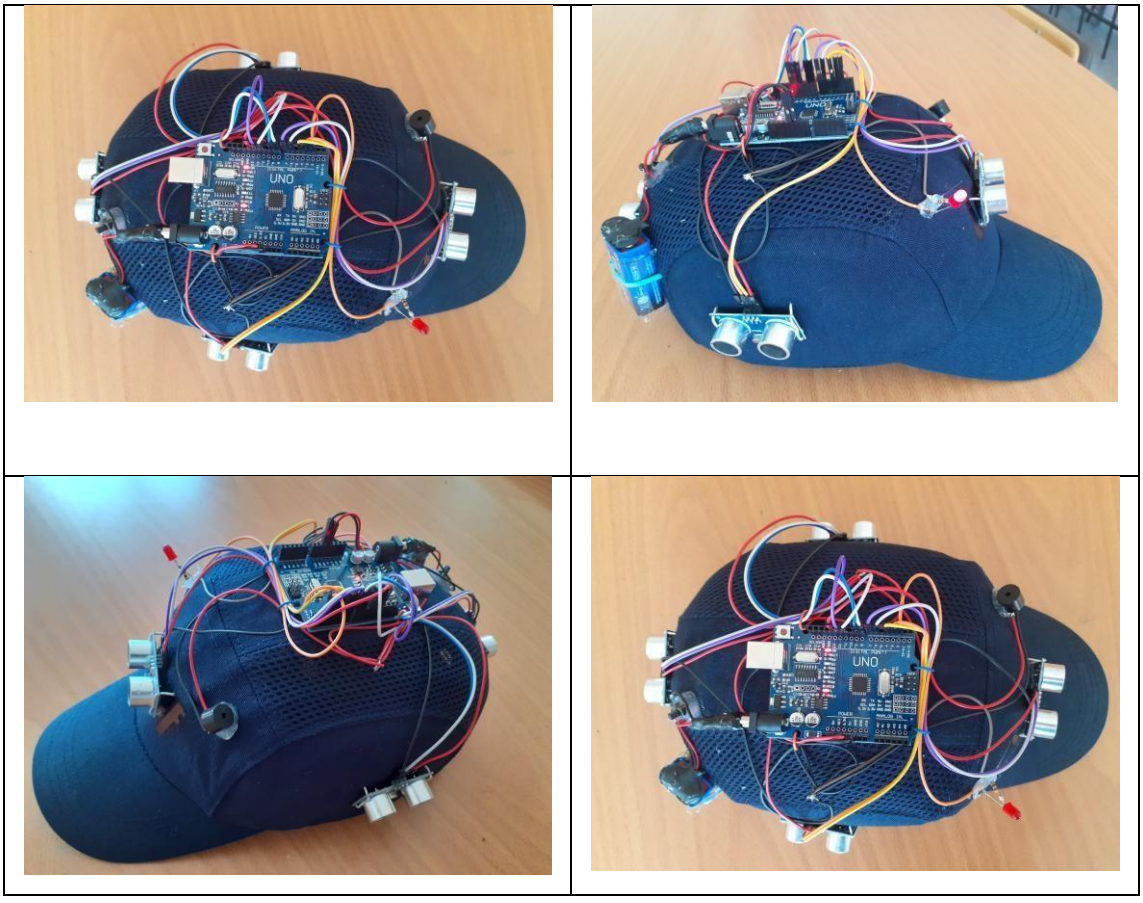


Foto: Circuito di progettazione finale sul tappo

2_NOME DEL PROGETTO: MISURATORE DI TEMPERATURA (GRECIA)

Scopo del progetto: Questo progetto è una versione estesa del semplice progetto di base Love Meter che si trova nel manuale originale di Arduino. In questa versione estesa, gli studenti potranno gestire un sensore (sensore di temperatura), sperimentare con il potenziometro, accendere un LED RGB e conoscere il display LCD (se non disponibile, altrimenti il display sarà solo il Serial Monitor del software Arduino).

È un progetto che permette agli studenti di sperimentare con molti componenti di tipo diverso e di familiarizzare con i valori di input e i metodi di visualizzazione.

Metodologia

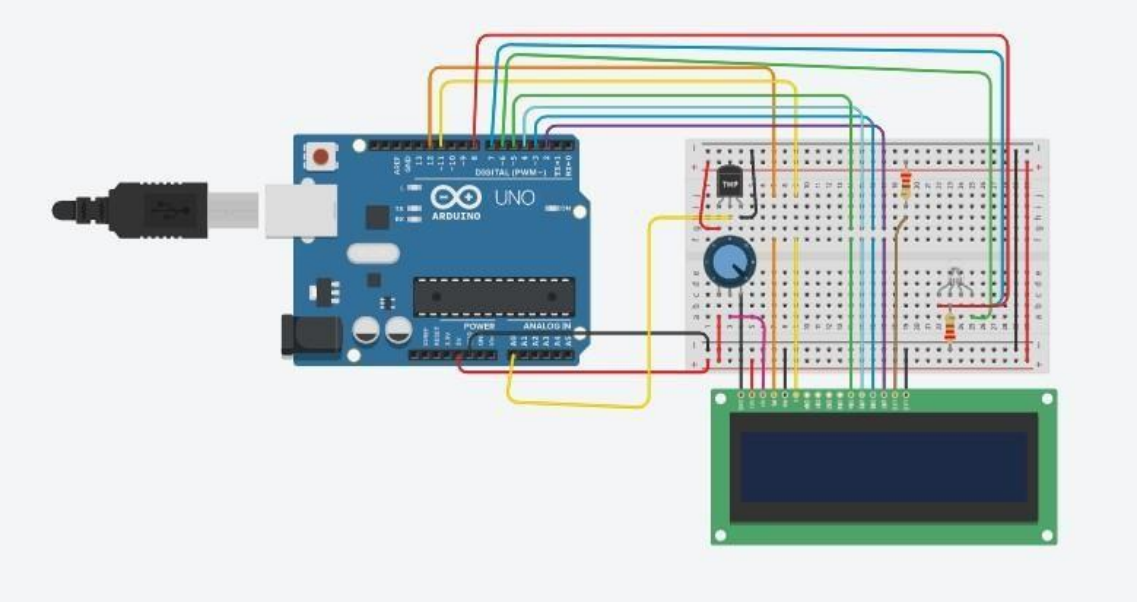
Inizialmente, descriviamo il flusso di lavoro desiderato del progetto. Poi chiediamo agli studenti di selezionare i componenti necessari e di usare Tinkercad per progettare e disegnare il circuito. Per scrivere il codice utilizzeranno lo strumento Codice di Tinkercad.

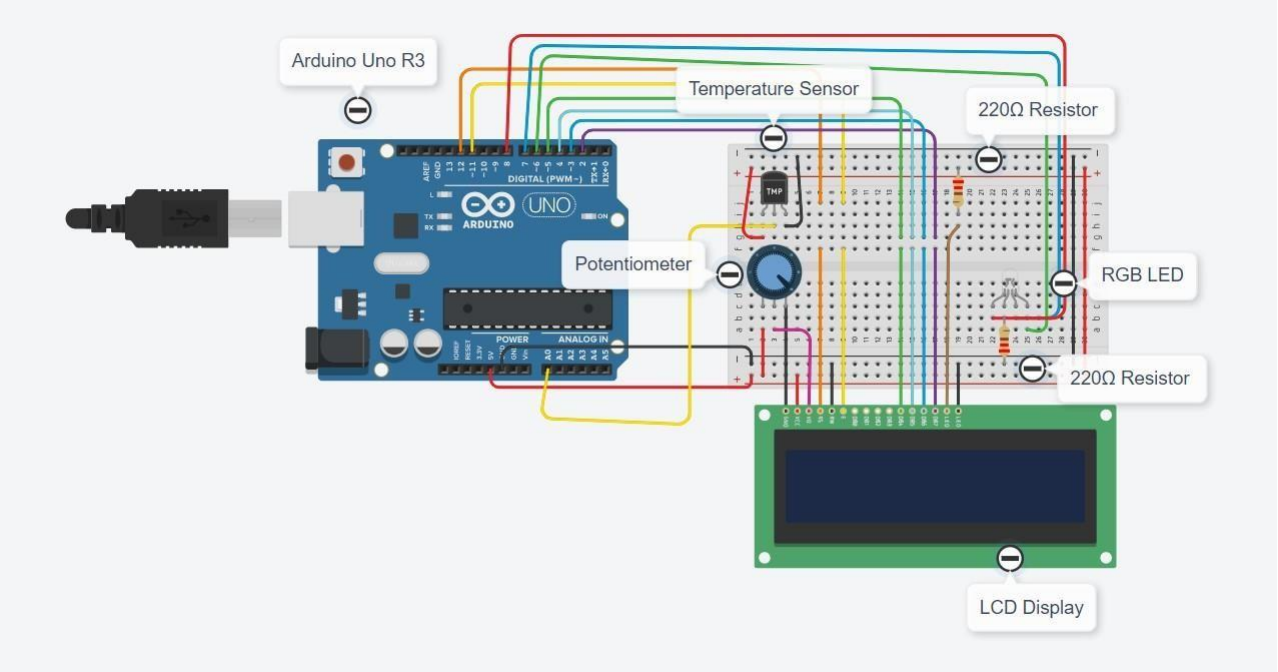
Poi creeranno il circuito, scriveranno il codice sul software Arduino e lo caricheranno sulla scheda Arduino.

La simulazione su Tinkercad verificherà che il circuito sia stato costruito correttamente.

Questo progetto è rivolto agli studenti del livello base, che iniziano a beneficiare dei risultati e degli esiti del nostro progetto Erasmus+ KA-202 Partenariati strategici nell'IFP, chiamato "ArduinVET".

Circuito di progetto.





Come funziona:

Il sensore di temperatura immette i valori della temperatura ambiente. Naturalmente, gli studenti possono modificare la temperatura con le dita. La temperatura viene visualizzata sul display LCD (se disponibile) e sul monitor seriale. Il LED RGB avrà colori diversi (spento, verde, blu, rosso) a seconda del valore della temperatura. Il potenziometro sarà utilizzato come interruttore per il display LCD. Il circuito sarà alimentato dal cavo USB (se necessario, è possibile utilizzare una batteria).

Elenco dei componenti:

I nostri componenti sono: un Arduino Uno R3, un sensore di temperatura, due (2) resistenze da 220Ω, un LED RGB e (opzionale) un potenziometro e un display LCD (16*2).

Codice Arduino del progetto:

//Nome del progetto: MISURATORE DI TEMPERATURA

// includere il codice della libreria:

```
#include <LiquidCrystal.h>
int t=0;
int sensore =
A0; float temp;
float tempc;
float tempf;
// inizializzare la libreria con i numeri dei pin dell'interfaccia
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
//**********RGB**********
const int sensorPin = A0;
// temperatura ambiente in gradi
Celsius const float baselineTemp
= 20.0;
//**************************
void setup() {
  // impostare il numero di colonne e righe
dell'LCD: pinMode(sensor,INPUT);
lcd.setCursore (0,0);
lcd.begin(16, 2);
  // Stampa un messaggio
sull'LCD. lcd.print ("
                    ARDUINO     ");
```

```
lcd.setCursore (0,1);

lcd.print ("TEMPERATURE METER");
```

```
ritardo (3000);
Serial.begin(9600);
for(int pinNumber = 6; pinNumber<8; pinNumber++){
    pinMode(pinNumber,OUTPUT); // pin6:Blu pin7:Verde pin8:Rosso
    digitalWrite(pinNumber, LOW);
   }
}
void loop() {
delay(2000);
t=t+2;
temp=letturaalogica(sensore);
//tempc=(temp*5)/10;
tempc=map(((temp-20)*3,04), 0, 1023, -40, 125);
tempf=(tempc*1,8)+32;
Serial.println("_______");
Serial.println("Logger temperatura");
Serial.print("Tempo in secondi= ");
Serial.println(t);
Serial.print("Temperatura in gradi Celsius = ");
Serial.println(tempc);
Serial.print("Temperatura in gradi Fahrenheit
= "); Serial.println(tempf);
lcd.setCursore(0,0);
lcd.print("Temp in C = ");
lcd.println(tempc);
lcd.setCursore(0,1);
```

```
lcd.print("Temp in F = ");
lcd.println(tempf);


//*************code RGB****************
  // leggere il valore sul pin AnalogIn 0
  // e memorizzarlo in una variabile
  int sensorVal = analogRead(sensorPin);
  // inviare il valore del sensore a 10 bit dalla porta
  seriale Serial.println("valore del sensore: ");
  Serial.println(sensorVal);

  // convertire la lettura ADC in tensione
  float voltage = (sensorVal/1024.0) * 5.0;
  // Inviare il livello di tensione dalla porta
  seriale Serial.println(", Volts: ");
  Serial.println(voltage);
    if(tempc < 20){
    digitalWrite(6, LOW);
    digitalWrite(7, LOW);
    digitalWrite(8, LOW);
    }
    else if(tempc >= 20 && tempc < 80){
          digitalWrite(6, HIGH);
    digitalWrite(7, LOW);
    digitalWrite(8, LOW);
  }
```

```
 else if(tempc >= 80 && tempc < 140){
        digitalWrite(6, LOW);
  digitalWrite(7, HIGH);
  digitalWrite(8, LOW);
}
 else if(tempc >= 140){
        digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, HIGH);
}
ritardo(100);
}
```

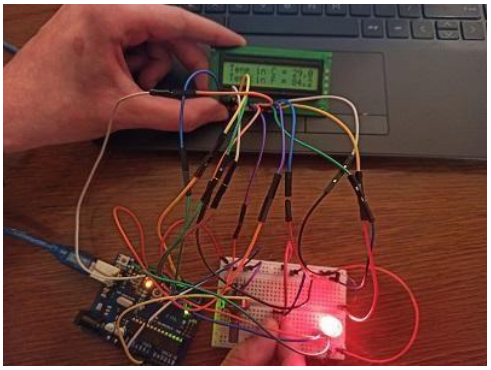
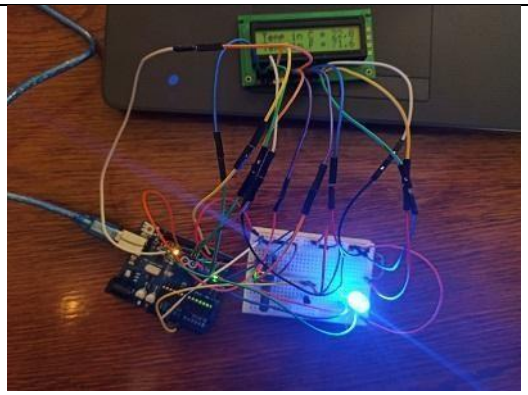





Foto: Kit di formazione del progetto e studenti che lavorano al progetto.

3_NOME DEL PROGETTO: LAMPEGGIATORE DI EMERGENZA - LUCE FRENO ADATTIVA

(ROMANIA)

Obiettivo del progetto: In caso di frenata brusca, le luci di frenata adattive possono avvertire il traffico in arrivo e contribuire così a prevenire i tamponamenti. Si attivano se, durante la frenata e a seconda della pressione di frenata e della velocità, viene identificata una situazione di frenata critica.

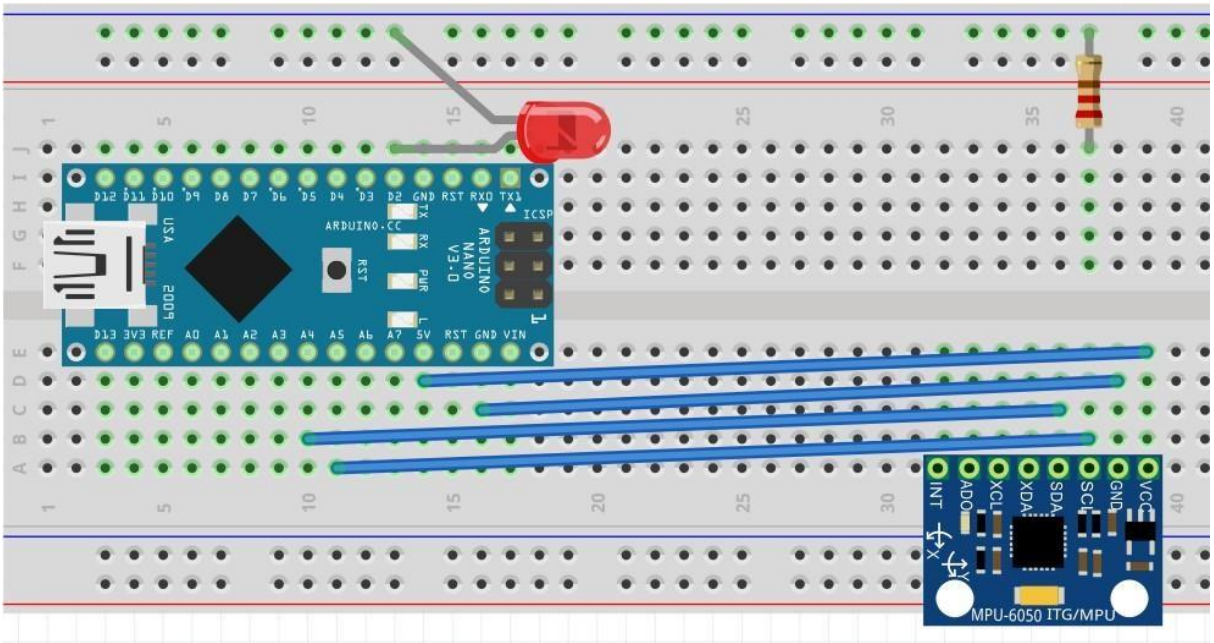
Il nostro metodo di progetto:

In primo luogo, viene preparato l'algoritmo del progetto. In base all'algoritmo preparato, si realizza il progetto logico e si scrive il programma Arduino. Arduino viene utilizzato per la progettazione del circuito. Durante la realizzazione di questi progetti, i nostri studenti beneficiano di tutti i risultati e i prodotti del progetto Erasmus+ KA-202 VET, denominato "ArduinVET".

Il nostro circuito di progettazione definitivo viene messo a punto dopo aver fornito l'attrezzatura necessaria e aver programmato Arduino. Il nostro circuito di progettazione finale viene collocato in un'automobile.

Questo progetto viene esposto alle fiere e ricorda ai visitatori quanto sia importante la sicurezza stradale.

Circuito di progetto.



Come funziona:

Le luci di emergenza si **attivano per avvisare i veicoli che si trovano dietro di loro in caso di frenata brusca**. Questa funzione fa sì che la luce dei freni lampeggi invece di brillare in modo costante come nella frenata normale. Le luci di emergenza si attivano a velocità superiori a 50 km/h in caso di brusche frenate.

Si basa su un accelerometro elettronico (MPU6050) assistito da un microcontrollore e rileva le frenate improvvise in base alla forza d'inerzia e fa lampeggiare rapidamente il terzo stop, rendendo l'auto molto più facile da notare rispetto a quella posteriore.

MPU6050 dispone di un giroscopio a 3 assi, di un accelerometro a 3 assi e di un processore di movimento digitale integrati in un unico chip. Funziona con un'alimentazione di 3V-5V. MPU6050 **utilizza il protocollo I2C per la comunicazione e il trasferimento dei dati**. Questo modulo dispone di un ADC a 16 bit integrato che garantisce una grande precisione.

Il filtro di Kalman è un efficiente filtro ricorsivo che valuta lo stato di un sistema dinamico sulla base di una serie di misure sensibili al rumore. Grazie alle sue caratteristiche intrinseche, è un filtro eccellente per il rumore e i disturbi che agiscono su sistemi gaussiani a media zero.

Elenco dei materiali:

Un Arduino Nano, un MPU6050, un resistore da 180 Ohm, un LED rosso.

Programma Arduino del progetto:

//Nome del progetto: LAMPEGGIATORE PER FRENI DI EMERGENZA - LUCE FRENO ADATTIVA

```
#include<Wire.h>;
//https://www.codetd.com/en/article/11749279 questo è un articolo che spiega molto bene come
comunicare più velocemente con MPU6050
//in condizioni automobilistiche c'è molto rumore dovuto alle condizioni della strada, alle
vibrazioni ecc. e userò un ottimo filtro software (Kalman).
const int MPU_addr = 0x68;
float kalman_old = 0;
float cov_old = 1;
float val1, val2, val3, val4, val5;
int med1_sort, med2_sort, med3_sort, med4_sort;
float avg;
int med;
```

```
int m;
int med_sort[5];
int c_avg = 1;
int c_med = 1;
int d = 0;
float old_x = 0;
float real_angle = 0;
float prev_angle = 0;
unsigned long previousMillis = 0;
const long interval = 50;
void setup() {
  Serial.begin(9600); // per il monitor USB
  Serial.println ("Premi un tasto per iniziare"); //segnala
  l'inizializzazione effettuata while (Serial.available() == 1);
  /Collegamento al sensore G
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B); // Registro PWR_MGMT_1
  Wire.write(0x00); // imposta a zero (sveglia la MPU-6050)
  Wire.endTransmission(true);
  delay(50);
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x1C);
  Wire.write(0x00);
  Wire.endTransmission(true);
  pinMode(2, OUTPUT); //segnale di uscita per il primo e il secondo stadio.
digitalWrite (2, LOW);
//pinMode(7, OUTPUT);//segnale di uscita per il SECONDO stadio. Attivare questa e la riga
successiva se si desidera una funzionalità più espressiva.
//digitalWrite (7, LOW); Se si attiva questa opzione, è necessario abilitare il pin di uscita digitale 7 in tutti gli sketch!!!
//questo segnale può essere utilizzato con un transistor PNP o un MOSFET P-Gate per completare il circuito
```

//nel circuito reale uso un led blu da 5v. Nel circuito di frittura uso un normale led rosso da 1,8 con una resistenza aggiuntiva di 180 ohm}.

```
void loop() {
  unsigned long currentMillis = millis();
  se (currentMillis - previousMillis >= intervallo) {
    previousMillis = currentMillis;
    Wire.beginTransmission(MPU_addr);
    Wire.write(0x3D);
// 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
// 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
// 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
    Wire.endTransmission(false);
    Wire.requestFrom(MPU_addr, 2, true);
    int16_t YAxisFull = (Wire.read() << 8 | Wire.read());
    float YAxisFinal = (float) YAxisFull / 16384,0;
    int YAxis_Filtered = general_filter(c_avg, c_med, YAxisFull);
    int YAxis_kalman = kalman_filter(YAxisFull);
    int YAxis_movg = mov_avg(c_avg, YAxisFull); int
    YAxis_median = mediana(c_med, YAxisFull);
    Serial.print("YAxis_Filtered");Serial.print(YAxis_Filtered/16384.0);
    Serial.print("\t");
   // Serial.print("YAxis_kalman");Serial.print(YAxis_kalman);
  // Serial.print("\t");
  // Serial.print("YAxis_movavg");Serial.print(YAxis_movavg);
   // Serial.print("\t");
  // Serial.print("YAxis_median");Serial.print(YAxis_median);
  // Serial.print("\t");
    Serial.print("YAxisFull");Serial.println(YAxisFull/16384.0);
   // Serial.print("\t");
   // Serial.print("YAxis_Final");Serial.println(YAxis_Final);
//FIRST stage
se (YAxis_Filtrato/16384.0 > 0,4 e YAxis_Filtrato/16384.0 <= 0,7){ for
  (int i = 1; i <= 4; i++) {
      digitalWrite (2, HIGH);
     // digitalWrite (7, HIGH);
```

```
      ritardo (75);
      digitalWrite (2, LOW);
     // digitalWrite (7, LOW);
      delay (50);}}
else {
  //Secondo stadio - freno molto
duro if (YAxis_Filtered/16384.0 >
0.7){
   for (int i = 1; i <= 7; i++) {
      digitalWrite (2, HIGH);
     // digitalWrite (7, HIGH);
      delay (75);
      digitalWrite (2, LOW);
      //digitalWrite (7, LOW);
      delay (50);}}}
    c_avg = c_avg + 1;
    c_med = c_med + 1;
    se (c_avg > 5 && c_med > 5)
                          }}
float kalman_filter (float input)
{ float kalman_new = kalman_old;
  float cov_new = cov_old + 0,50;
  float kalman_gain = cov_new / (cov_new + 0,9);
  float kalman_calculated = kalman_new + (kalman_gain * (input - kalman_new));
  cov_new = (1 - kalman_gain) * cov_old;
  cov_old = cov_new;
  kalman_old = kalman_calcolato;
  return kalman_calcolato;}
float mov_avg (int counter, float input)
{ avg = 0;
  m = 0;
  if (counter == 1) {
    val1 = input;
```

```
  avg = val1; }
else if (counter == 2) {
  val2 = input;
  avg = val2; }
else if (counter == 3) {
  val3 = input;
  avg = val3; }
else if (counter == 4) {
  val4 = input;
  avg = val4; }
else if (counter == 5) {
  val5 = input;
  avg = val5; }
else if (counter > 5 ) {
  counter = 6;
  se (val1 == 0) {
    m = m + 1;    }
  se (val2 == 0) {
    m = m + 1;    }
  se (val3 == 0) {
    m = m + 1;    }
  se (val4 == 0) {
    m = m + 1;    }
  se (val5 == 0) {
    m = m + 1;    }
  if (input == 0) {
    m = m + 1;    }
  d = 6 - m;
  se (d ==
  0)
  {  avg  =  input;
  counter  =  1;  }
  altrimenti
```

```
{ avg = (val1 + val2 + val3 + val4 + val5 + input) / d; }
```

```
  val1 = val2;
  val2 = val3;
  val3 = val4;
  val4 = val5;
  val5 = input; }
  return avg;}
mediana (int counter, int input)
{ if (counter == 1) {
  med1_sort = input;
  med_sort[0] = med1_sort; }
  else if (counter == 2) {
  med2_sort = input;
  med_sort[1] = med2_sort; }
  else if (counter == 3) {
  med3_sort = input;
  med_sort[2] = med3_sort; }
  else if (counter == 4) {
  med4_sort = input;
  med_sort[3] = med4_sort; }
  else if (counter >= 5) {
  counter = 6;
    med_sort[4] = input;
    sort(med_sort, 5);
    med = med_sort[2];
    med1_sort = med2_sort;
  med2_sort = med3_sort;
  med3_sort = med4_sort;
  med4_sort = input;
  med_sort[0] = med1_sort;
  med_sort[1] = med2_sort;
  med_sort[2] = med3_sort;
  med_sort[3] = med4_sort; }
  return med;}
```

```
void sort(int a[], int size) {
  per (int i = 0; i < (size - 1); i++) {
    for (int o = 0; o < (size - (i + 1)); o++) {
      if (a[o] > a[o + 1]) {
        int t = a[o];
        a[o] = a[o + 1];
        a[o + 1] = t;      }} } }
/funzione con tutti i filtri applicati
float general_filter (int counter_avg, int counter_med, float input) {
float input_movavg = mov_avg(counter_avg, input);    //applicazione
movingavgfloat input_med = median(counter_med, input_movavg);
//applicazione mediana float input_filtered = kalman_filter(input_med);
                                                          //applicazione Kalman
  return input_filtered;}
```

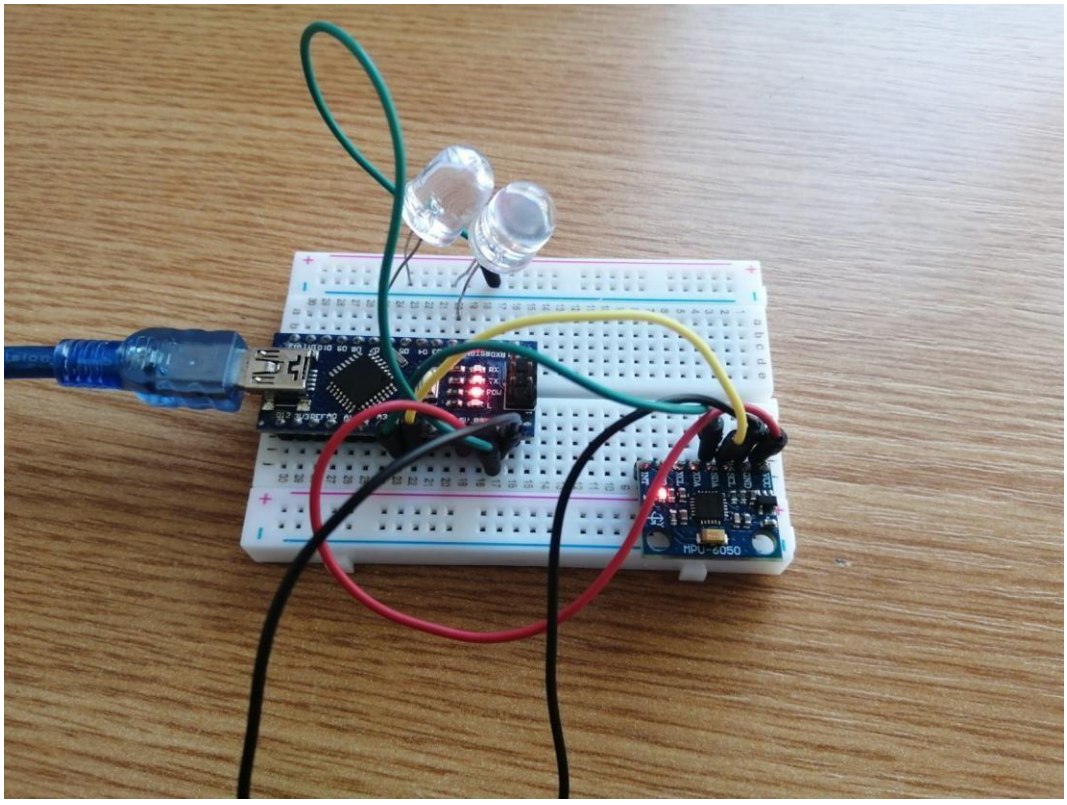


Foto: Circuito di progettazione finale

4_NOME DEL PROGETTO: LINEFOLLOWER (AUSTRIA)

Che cos'è un Linefollower:

Il robot segui-linea è una macchina mobile in grado di rilevare e seguire la linea tracciata sul pavimento. In genere, il percorso è predefinito e può essere visibile come una linea nera su una superficie bianca ad alto contrasto di colore. Sicuramente, questo tipo di robot deve rilevare la linea con i suoi sensori a raggi infrarossi (IR) installati sotto il robot. Successivamente, i dati vengono trasmessi al processore. Il processore deciderà quindi i comandi appropriati e li invierà al conducente, in modo che il robot segua il percorso.

Il punto importante nella costruzione di un robot inseguitore di linee è un buon controllo, sufficiente a seguire il percorso il più velocemente possibile.

Funzionamento del robot inseguitore di linee

Il concetto di robot inseguitore di linee è legato alla luce. In questo caso, utilizziamo il comportamento della luce su una superficie bianca e nera. Il colore bianco riflette tutta la luce che cade su di esso, mentre il colore nero assorbe la luce.

In questo robot inseguitore di linee, utilizziamo trasmettitori e ricevitori IR (fotodiodi). Vengono utilizzati per inviare e ricevere le luci. Quando i raggi IR cadono su una superficie bianca, vengono riflessi verso il ricevitore IR, generando alcune variazioni di tensione.

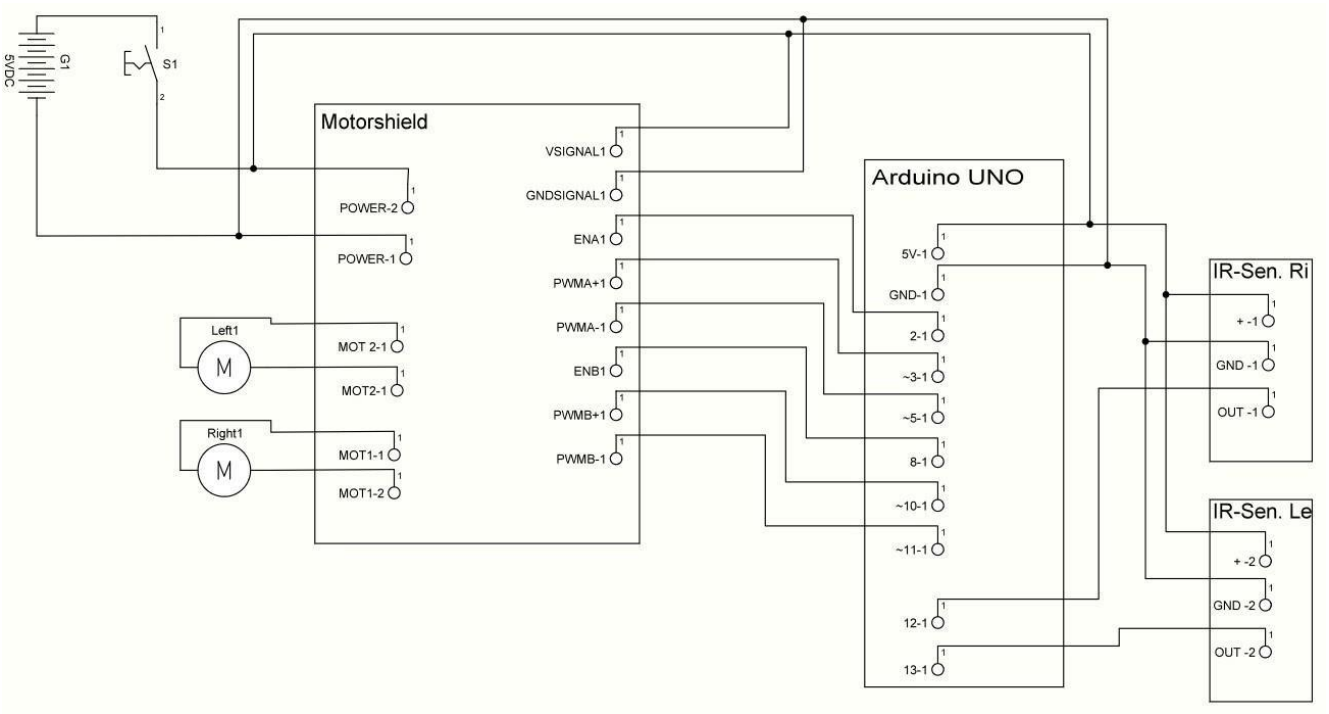
Quando i raggi IR cadono su una superficie nera, vengono assorbiti dalla superficie nera e non vengono riflessi; di conseguenza, il ricevitore IR non riceve alcun raggio.

In questo progetto, quando il sensore IR rileva una superficie bianca, Arduino riceve come ingresso 1 (HIGH), mentre quando rileva una linea nera, Arduino riceve come ingresso 0 (LOW). In base a questi ingressi, Arduino Uno fornisce l'uscita corretta per controllare il robot.

Lista di candidati:

parti	tipo	pezzi
Arduino	Uno	1
Paraurti motore	da HTL Wolfsberg	1
Motore CC	COM-Motore01, 3-9V DC, joy-it	2
Banca di energia	Flip12 Caricabatterie telefonico, 3350mAh	1
Sensori IR	SEN-KY032IR, joy-it	2
Interruttore principale		1
Cavo di collegamento	RB-CB3-025, joy-it	1
Cavo USB	Digitus, AK-300102-010-S, Tipo A-B	s

Schema del circuito:



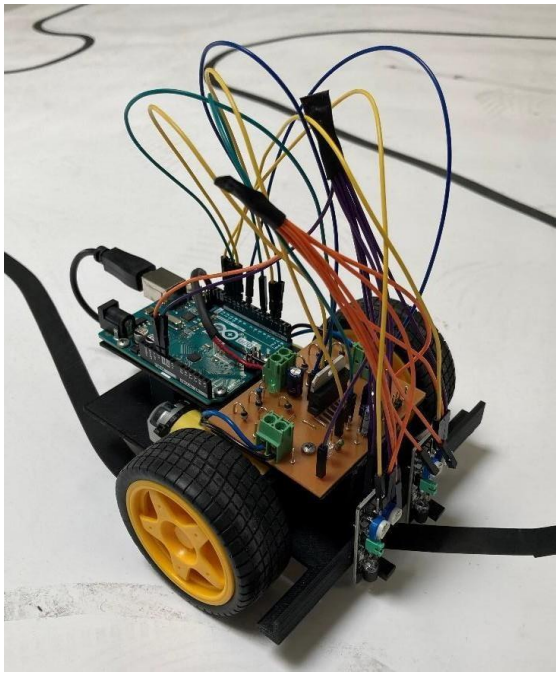


Foto: Circuito di progettazione finale

Programma:

```
// Linefollower
const byte sr = 12;
const byte sl = 13;
const byte enr = 2;
const byte enl = 8;
const byte mrf = 3;
const byte mrb = 5;
const byte mlf = 10;
const byte mlb = 11;

vuoto setup()
{
  Serial.begin(9600);

  pinMode(sr,INPUT);
  pinMode(sl,INPUT);

  pinMode(enr,OUTPUT);
  pinMode(enl,OUTPUT);
  digitalWrite(enr,HIGH);
  digitalWrite(enl,HIGH);

  pinMode(mrf,OUTPUT);
  pinMode(mrb,OUTPUT);
  pinMode(mlf,OUTPUT);
  pinMode(mlb,OUTPUT);

  analogWrite(mrf,0);
  analogWrite(mrb,0);
  analogWrite(mlf,0);
  analogWrite(mlb,0);
}
  bool senR;
  bool senL;

vuoto loop()
{
  senR = digitalRead(sr);
  senL = digitalRead(sl);

  if(!senR && !senL) //staight
  {
    analogWrite(mrf,220);
    analogWrite(mrb,0);
    analogWrite(mlf,200);
    analogWrite(mlb,0);
```

```
 }

 // sensore destro in linea
 /Guida a destra
 altrimenti se(senR && !senL)
 {
  analogWrite(mrf,0);
  analogWrite(mrb,100);
  analogWrite(mlf,255);//255
  analogWrite(mlb,0);
 }

 //sinistro del sensore sulla linea
 /Guida a sinistra
 se(!senR && senL)
 {
  analogWrite(mrf,255);//255
  analogWrite(mrb,0);
  analogWrite(mlf,0);
  analogWrite(mlb,100);
 }

 // entrambi i sensori sul nero
 // stop
 altrimenti se(senR && senL)
 {
  analogWrite(mrf,0);
  analogWrite(mrb,0);
  analogWrite(mlf,0);
  analogWrite(mlb,0);
  delay(3000);
 }
}
```

5_NOME DEL PROGETTO: SERRA AUTOMATIZZATA (ITALIA)

Lo scopo del progetto: Perché un progetto di serra automatizzata a scuola?

Gli studenti, a scuola, hanno seguito diversi percorsi sulla cittadinanza attiva e sull'Agenda 2030 dell'ONU per lo Sviluppo Sostenibile, sulla sostenibilità energetica, sull'importanza di evitare gli sprechi d'acqua e sulla necessità di sviluppare metodi di coltivazione innovativi e rispettosi dell'ambiente, considerando i profondi cambiamenti climatici in corso. La coscienza ecologica dei ragazzi si esprime nella ricerca di soluzioni tecniche e idee innovative, partendo dalle conoscenze teoriche e di laboratorio acquisite a scuola e dalle opportunità applicative di Arduino.

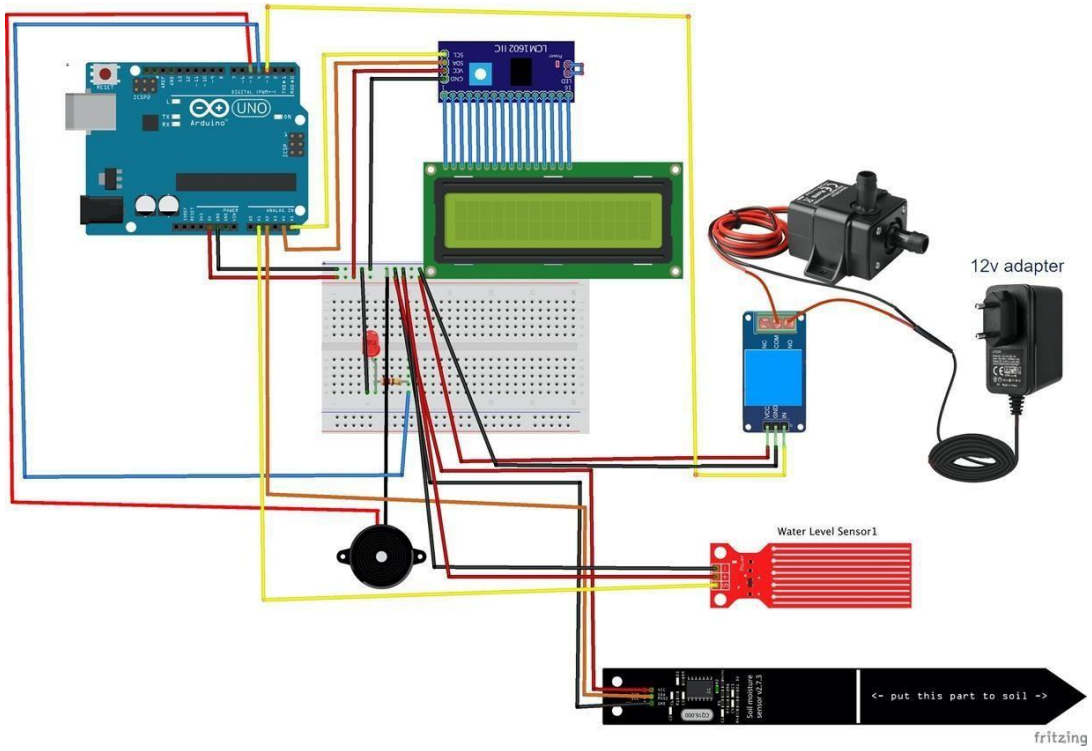
Alcuni studenti, supportati dagli insegnanti, hanno progettato un sistema di rilevamento della temperatura del suolo e dell'aria, dell'umidità del terreno, per gestire l'irrigazione e la ventilazione di un prototipo di serra adatto alla coltivazione di frutta e verdura locale.

La sperimentazione ha dato i risultati sperati e contribuirà all'effettiva implementazione nell'area nelle vicinanze della scuola destinata a ospitare il futuro orto scolastico.

Il nostro metodo di progetto

In primo luogo, è stato preparato un elenco di tutti i parametri che vogliamo utilizzare per il funzionamento della serra. A partire da questo è stata preparata una prima versione dell'algoritmo del progetto. In base a questo algoritmo, sono stati acquistati i sensori necessari ed è stato realizzato il progetto logico. Per la progettazione dei circuiti è stato utilizzato Arduino. Infine, è stato implementato lo script finale. Dopo aver verificato la corretta impostazione dei valori di soglia dei vari sensori, questi sono stati posizionati all'interno della serra.

Progetto Circuito



Come funziona:

Un display LCD fornisce i valori della temperatura e dell'umidità dell'aria, dell'umidità del terreno e della quantità d'acqua presente nel contenitore di irrigazione. quando i valori sono al di sotto di una certa soglia, una pompa motorizzata controllata da un relè avvia l'irrigazione del terreno. quando il livello dell'acqua è al di sotto di una certa soglia, il display mostra un avviso di riempimento accompagnato da un allarme acustico.

Elenco dei materiali:

Una serra, un Arduino Uno R3, un sensore di temperatura/umidità DHT11, un cicalino, un sensore di umidità del terreno, un sensore di livello dell'acqua, un led rosso, un alimentatore da 12 V, una pompa da 12 V, una resistenza da 220 ohm, un relè, un display LCD.

Programma Arduino del progetto:

//Nome del progetto: SERRA AUTOMATIZZATA:

```
#includere
<LiquidCrystal_I2C.h>
#includere <Wire.h>
#include <DHT.h>
#define DHTPIN 11
#define DHTTYPE DHT11 // DHT 11
DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27,20,4); // imposta l'indirizzo LCD su 0x27 per un display a 16 caratteri e 2 righe
int t, h, Lumen, lumin, water, igro, umdtr, wlevel;

const int pin_acqua =
A1; const int pin_igro =
A2; const int pin_pump =
3; const int pin_led = 4;
const int pin_buzzer = 5;

void setup() {
   Serial.begin(9600);

   dht.begin();
   lcd.init(); lcd.backlight();
   lcd.setCursor(5,0);
   lcd.print("Scuola");
   lcd.setCursor(3,1);
   lcd.print("Serra");
   ritardo (5000);
```

```
    lcd.clear();
    lcd.setCursor(2,0);
    lcd.print("IIS Einstein");
    lcd.setCursor(3,1);
    lcd.print("De Lorenzo");
    delay (5000); lcd.clear();

    pinMode(pin_pump,OUTPUT);
    pinMode(pin_led, OUTPUT);
    pinMode(pin_buzzer, OUTPUT);
    digitalWrite(pin_pump, HIGH);
}

void loop() {
    // livello dell'acqua
    water = analogRead (pin_water);
    wlevel = map (water,0, 1023, 0, 100);
    if(wlevel < 35){
        lcd.clear();
        tone(pin_buzzer, 800);
        digitalWrite(pin_led, HIGH);
        delay(300);
        noTone(pin_buzzer);
        digitalWrite(pin_led, LOW);
        delay(300);
        lcd.clear();
        //lcd.begin(16, 2);
```

```
    lcd.setCursore(2,0);
    lcd.print("Riempimento
    acqua"); ritardo (1000);
    lcd.clear();
} else {
    noTone(pin_buzzer);
    digitalWrite(pin_led, LOW);
}

// Igrometro
igro = analogRead(pin_igro);
umdtr = mappa (igro, 0, 1023, 0, 100);

if(umdtr < 45){
    digitalWrite(pin_pump, LOW);
    delay(10000);
    digitalWrite(pin_pump, HIGH);
}

// Lettura umidità e temperatura del sensore
DHT11 h = dht.readHumidity();
t = dht.readTemperature();

// posiziona il cursore in colonna 0 e linea 1
// (nota: la linea 1 e la seconda linea, poichè si conta incominciando da 0):
lcd.setCursor(0, 0);
lcd.print("T:");
lcd.print(t);
```

```
    lcd.print( (char) 223 );
    lcd.print("C");

    lcd.setCursore(10,
    0); lcd.print("H:");
    lcd.print(h);
    lcd.print("%");

    lcd.setCursor(0, 1);
    lcd.print("SH:");
    lcd.print(umdtr);
    lcd.print("%");

    lcd.setCursore(10,
    1); lcd.print("WL:");
    lcd.print(wlevel);
    lcd.print("%");
}
```

Foto: Progetto definitivo

Allegati

La missione di Arduino è consentire a chiunque di migliorare la propria vita attraverso l'elettronica e le tecnologie digitali accessibili. Un tempo esisteva una barriera tra il mondo dell'elettronica, del design e della programmazione e il resto del mondo. Arduino ha abbattuto questa barriera. Per ulteriori informazioni su Arduino, è possibile visitare il sito Web...

www.arduino.cc/

Per maggiori informazioni sul nostro progetto ARDinVET, potete visitare il sito web, qui sotto...

www.arduinvet.com

"Questo progetto è finanziato dal Programma Erasmus+ dell'Unione Europea. Tuttavia, la Commissione europea e l'Agenzia nazionale turca non possono essere ritenute responsabili per l'uso che può essere fatto delle informazioni in esso contenute".