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

Erasmus+ KA-202

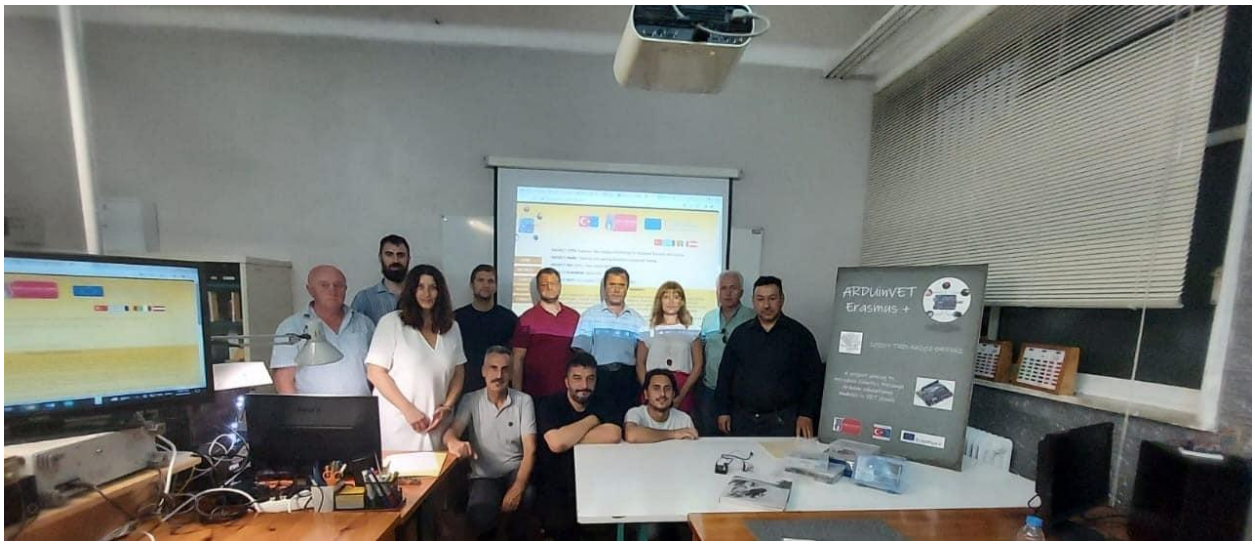
Proiecte de parteneriate strategice pentru educație și formare profesională

Titlul proiectului: "Predarea și învățarea ARDUINO în formarea profesională"

Acronimul proiectului: " ARDUinVET "

Numărul proiectului: "2020-1-TR01-KA202-093762"

GHID ARDUinVET



“Acest proiect este finanțat de programul Erasmus + al Uniunii Europene. Cu toate acestea, Comisia Europeană și Agenția Națională Turcă nu pot fi trase la răspundere pentru orice utilizare a informațiilor conținute în acestea”

“This project is Funded by the Erasmus+ Program of the European Union. However, European Commission and Turkish National Agency cannot be held responsi-ble for any use which may be made of the information contained therein”

COORDINATOR:

Gölbaşı Mesleki ve Teknik Anadolu Lisesi (Ankara / TURKEY)

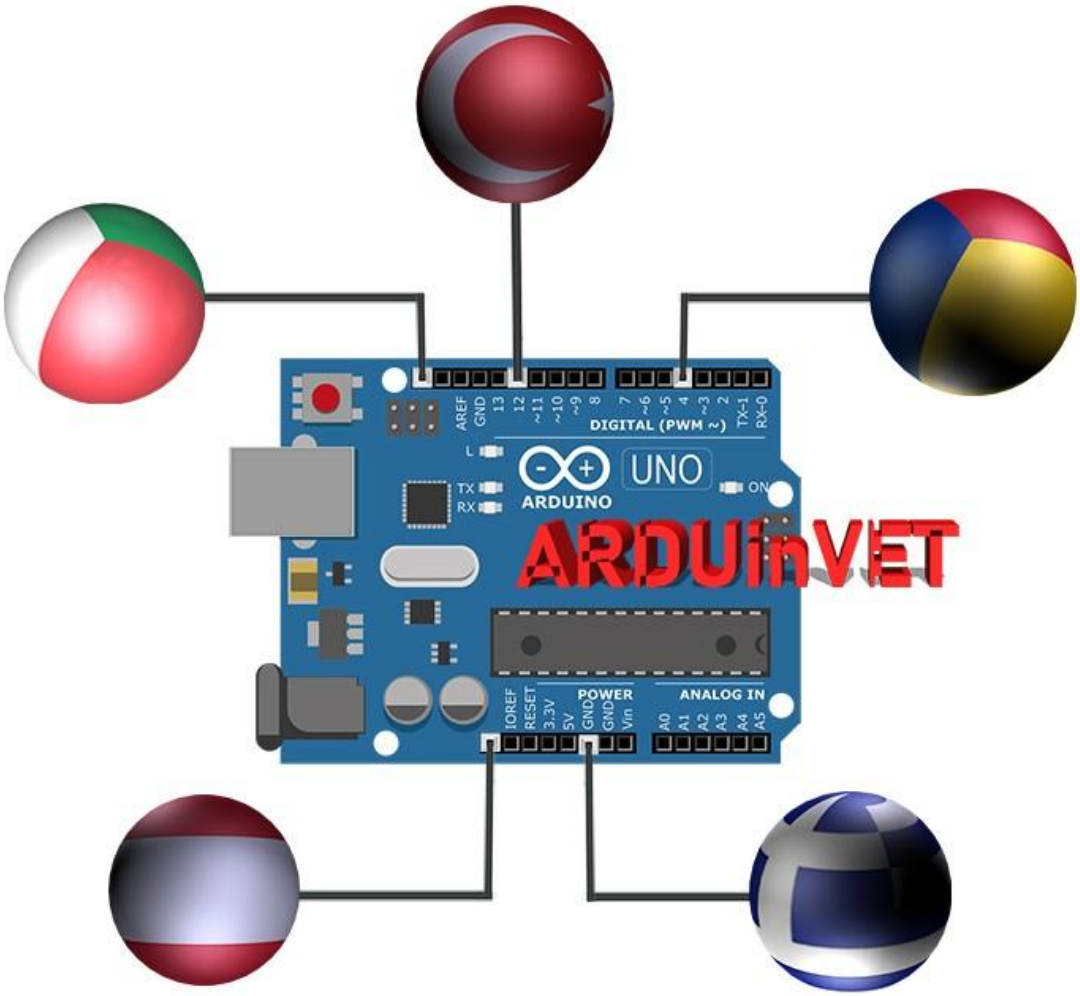
PARTNERS:

2 EK Peiraia (Piraeus / Greece)

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)



Sumarul proiectului

Tehnologia se dezvoltă foarte rapid zi de zi. Este un fapt că evoluțiile tehnologice obligă un om să se dezvolte continuu în profesia sa. Mai mult decât orice alt domeniu, tehnologia electronică și a informației și comunicațiilor se dezvoltă mai rapid. Aceste evoluții au condus la producerea unor sisteme de control mai complexe. Arduino-urile sunt acum folosite pentru a controla aceste sisteme complexe.

Deoarece suntem instituții de învățământ care predăm tehnologia, trebuie să urmărim evoluțiile din lume. Proiectul „Predarea și învățarea ARDUINO în formarea profesională" își propune să adapteze aplicațiile ARDUINO la formarea profesională și să dezvolte un set de formare mai eficient și un ghid pentru laboratoare și atelierele elevilor din învățământul profesional și tehnic.

Obiectivul principal al acestui proiect este de a dezvolta un ghid de bune practici și un set de instruire Arduino, de a introduce modele de instruire Arduino altor participanți în timpul vizitelor lor în țara gazdă, de a compara diferite sisteme educaționale și metode de instruire cu alte școli participante și de a împărtăși cele mai bune practici. Participanții la proiect; profesori electrici, electronici, TIC, automatizare VET. Participanții predau Arduino în școlile VET. Există un coordonator și un total de 4 parteneri în proiect, iar partenerii sunt școli VET din Turcia, Italia, România, Austria și Grecia. Numărul total de mobilități din proiect este de 40. Un număr total de 5 TPM vor fi realizate pe tot parcursul proiectului și fiecare partener va participa la fiecare TPM cu 2 participanți. Un TPM va avea loc în fiecare țară.

Cu activitățile proiectului, se va asigura că cele mai bune practici vor fi împărtășite pentru predarea Arduino și partenerii le vor adapta la mediile lor educaționale. Partenerii proiectului vor produce kit-uri experimentale Arduino și un ghid de bune practici. Activitățile proiectului nostru își propun să ofere un mediu de învățare și predare de succes pentru toți profesorii și elevii VET. Ca o condiție de învățare reușită, profesorii trebuie să își consolideze rolul în facilitarea învățării. Profesorii au nevoie de noi kit-uri experimentale și module pentru inovație, lucru în echipă, feedback și evaluare. Profesorilor ar trebui să li se ofere această oportunitate pentru dezvoltarea profesională continuă.

Principalele rezultate ale proiectului sunt; un kit de instruire pentru lecțiile Arduino în învățământul profesional și tehnic și un Ghid pentru acest kit, un site web al proiectului, un DVD al proiectului. Metodologia care va fi utilizată în execuția proiectului este „Make-Develop-Share". În proiectul nostru, bunele practici vor fi făcute mai întâi, apoi dezvoltate și, în final, împărtășite. Totul este deschis și transparent în proiectul nostru. Fiecare sarcină și responsabilitate vor fi înregistrate sau scrise în proiect. Produsele proiectului nu sunt doar produse scrise sau documentare, ci și un kit de antrenament și kit-uri practice Arduino.

Pe termen lung, ne propunem să diseminăm proiectul către profesori, elevi și școli de învățământ profesional și tehnic, instituții de învățământ locale, piața muncii electronice și TIC. Credem că rezultatele și produsele proiectelor noastre vor avea un impact pe termen scurt și lung asupra educației profesionale și a pieței muncii prin activitățile de diseminare. Bugetul total al proiectului cu 5 parteneri este de 59.000 de euro.

CONTENTS:

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

PROIECTELE MODULELOR and KITURILOR din ARDUinVET

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

Erasmus+ KA-202

Proiecte de parteneriate strategice pentru educație și formare profesională

Titlul proiectului: "Predarea și învățarea ARDUINO în formarea profesională"

Acronimul proiectului: " ARDUinVET "

Numărul proiectului: "2020-1-TR01-KA202-093762"

INTRODUCERE ÎN ARDUINO

(BAZELE ARDUINO)

INTRODUCERE ÎN ARDUINO

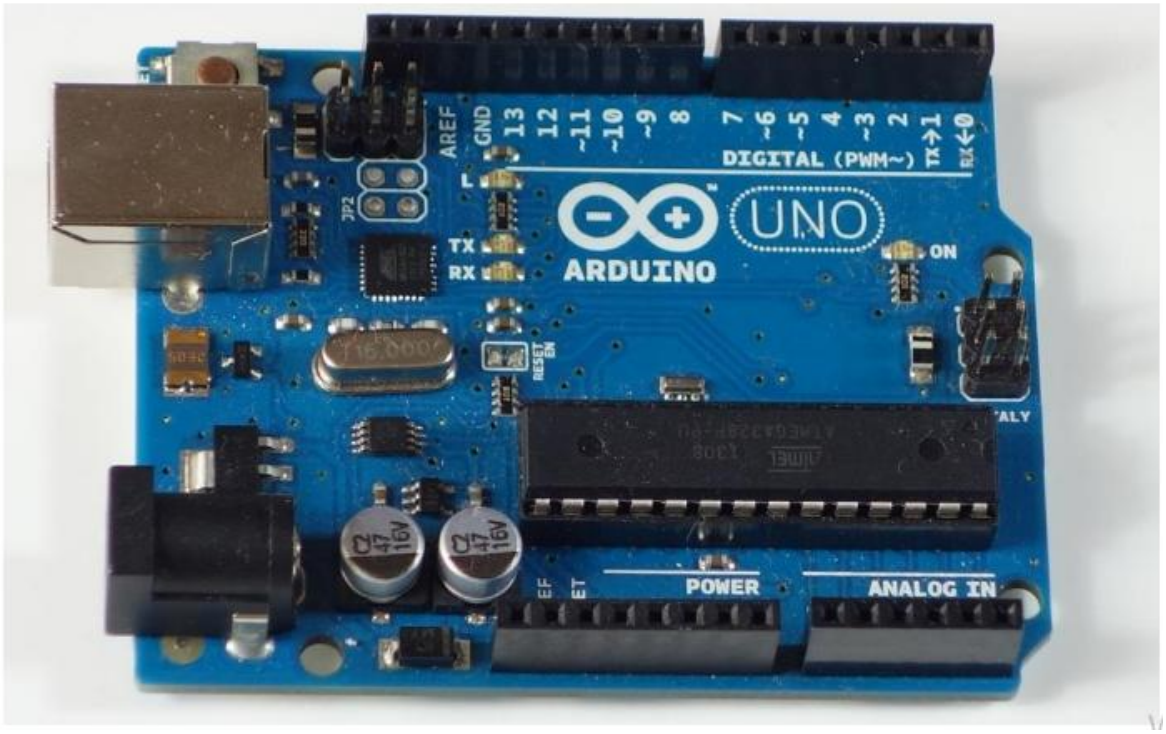
Arduino este o platformă prototip (open-source) bazată pe un hardware și software ușor de utilizat. Se compune dintr-o placă de circuit, care poate fi programată (denumită microcontroler) și un software numit Arduino IDE (Integrated Development Environment), care este folosit pentru a scrie și încărca codul pe placa fizică.

Cu alte cuvinte, Arduino este un mic computer pe care îl puteți programa pentru a citi informațiile din lumea înconjurătoare și a trimite comenzi către lumea exterioară. Toate acestea sunt posibile, deoarece puteți conecta mai multe dispozitive și componente la Arduino pentru a face ceea ce doriți. Puteți face proiecte uimitoare cu el, nu există nicio limită pentru ceea ce puteți face și, folosindu-vă imaginația, totul este posibil!

În termeni simpli, Arduino este un sistem de calcul mic, care poate fi programat cu instrucțiunile dumneavoastră pentru a interacționa cu diferite semnale de intrare și ieșire. Modelul actual de placă Arduino Uno, are dimensiuni destul de mici în comparație cu o mâna medie umană.

Ce este ARDUINO?

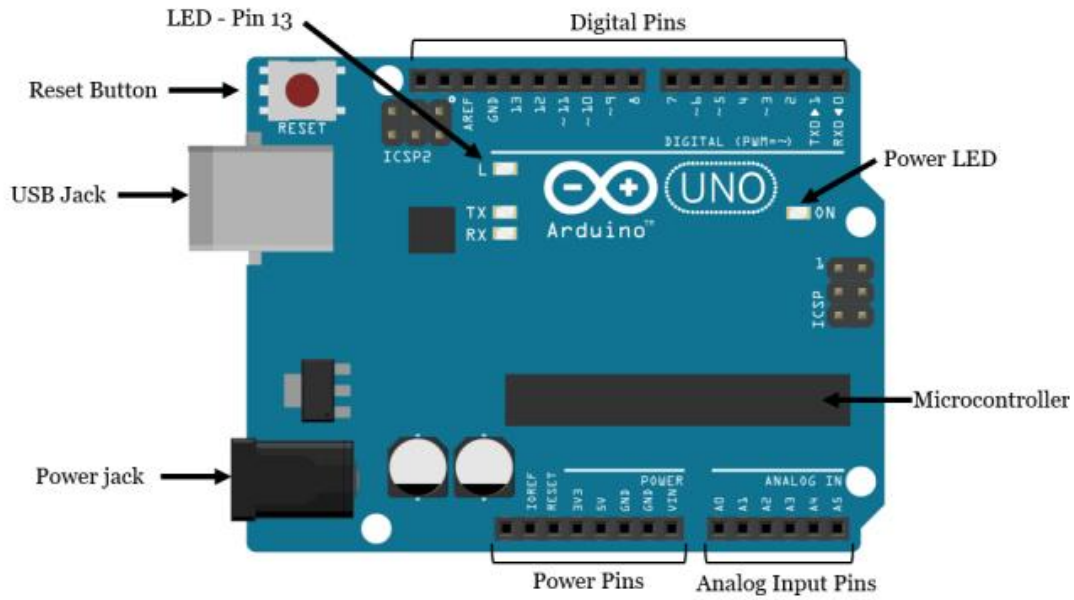
Arduino este placa prezentată în figura de mai jos.



Practic, este o placă mică de dezvoltare cu un creier (cunoscut și sub numele de microcontroler) pe care îl puteți conecta la circuite electrice. Acest lucru facilitează citirea intrărilor - citirea datelor din exterior - și controlul ieșirilor - trimiterea unei comenzi către exterior. Creierul acestei plăci (Arduino Uno) este un cip ATmega328p unde vă puteți stoca programele care îi va spune lui Arduino ce trebuie să facă.

Explorarea plăcii Arduino Uno

În figura de mai jos, puteți vedea o placă Arduino etichetată. Să vedem ce face fiecare parte.



- **Microcontrolerul:** ATmega328p este creierul Arduino. Totul de pe placa Arduino este menit să susțină acest microcontroler. Aici se stochează programele care îi transmit lui Arduino ce trebuie să facă.
- **Pinii digitali:** Arduino are 14 pini digitali, etichetați de la 0 la 13, care pot acționa ca intrări sau ieșiri.

✧ Când sunt setați ca intrări, acești pini pot citi tensiunea. Pot citi doar două stări: HIGH sau LOW.

✧ Când sunt setați ca ieșiri, acești pini pot aplica tensiune. Ele pot aplica doar 5V (HIGH) sau 0V (LOW).

- **Pinii PWM:** aceștia sunt pinii digitali marcați cu a ~ (pinii 11, 10, 9, 6, 5 și 3). PWM înseamnă „modularea lățimii impulsurilor” și permite pinilor digitali să dea „fals” pentru valori variabile de tensiune. Veți afla mai multe despre PWM mai târziu.
- **Pinii TX și RX:** pinii digitali 0 și 1. T reprezintă „transmit” și R pentru „recepție”. Arduino folosește acești pini pentru a comunica cu alte electronice prin Serial. De asemenea, Arduino

folosește acești pini pentru a comunica cu computerul dumneavoastră. atunci când încărcați un cod nou. Evitați să utilizați acești pini pentru alte sarcini, în afară de comunicarea serială, cu excepția cazului în care vă epuizați pinii.

- **LED atașat pinului digital 13:** Acesta este util pentru o depanare ușoară a programelo (sketches) Arduino.
- **LED-uri TX și RX:** aceste LED-uri clipesc când sunt trimise informații între computer și Arduino.
- **Pinii analogici:** pinii analogici sunt etichetați de la A0 la A5 și sunt adesea folosiți pentru citirea senzorilor analogici. Pot citi diferite valori de tensiune între 0 și 5V. În plus, pot fi folosiți și ca pini de ieșire/intrare digitali, precum pinii digitali.
- **Pinii de alimentare:** Arduino furnizează 3,3V sau 5V prin acești pini. Acest lucru este foarte util, deoarece majoritatea componentelor necesită 3,3V sau 5V pentru a funcționa. Pinii etichetați ca „GND” sunt pinii de masă.
- **Buton Reset:** când apăsați butonul respectiv, programul care se execută în prezent în Arduino se repornește. De asemenea, aveți un pin Reset lângă pinii de alimentare care acționează ca buton de resetare. Când aplicați o tensiune mică la acel pin, acesta va reseta Arduino.
- **LED de pornire:** va fi aprins când Arduino este alimentat.
- **Mufă USB:** aveți nevoie de un cablu USB A tată, USB B tată (prezentat în figura de mai jos) pentru a încărca programe de pe computer pe placa Arduino. Acest cablu alimentează și Arduino.

- **Mufă de alimentare:** puteți alimenta Arduino prin mufa de alimentare. Tensiunea de intrare recomandată este de 7V la 12V. Există mai multe moduri de a vă alimenta Arduino: de exemplu; acumulatori reîncărcabili, baterii de unică folosință, adaptor sau panou solar.

Caracteristicile Arduino

Arduino Uno; are microcontroler Atmel Atmega 328P și conexiune de intrare USB, intrare jack de alimentare, buton reset. Arduino are tot ce ar trebui să aibă un microcontroler.

Microcontroller	Atmega328P
Tensiune de lucru	5V
Tensiune de intrare (recomandată)	7-12V
Tensiune de intrare (maximă)	6-20V
Pini digitali intrare/ ieșire	14
Pini intrare/ ieșire PWM	6
Pini intrare analogici	6
Pini intrare/ ieșire DC	20mA
Pini DC de 3.3V	50mA
Memorie flash	32 KB
SRAM	2KB
EEPROM	1 KB
Clock Speed	16 MHz
Lungime	68.6 mm
Lățime	53.4 mm
Greutate	25 g

Figură: Caracteristici Arduino Uno

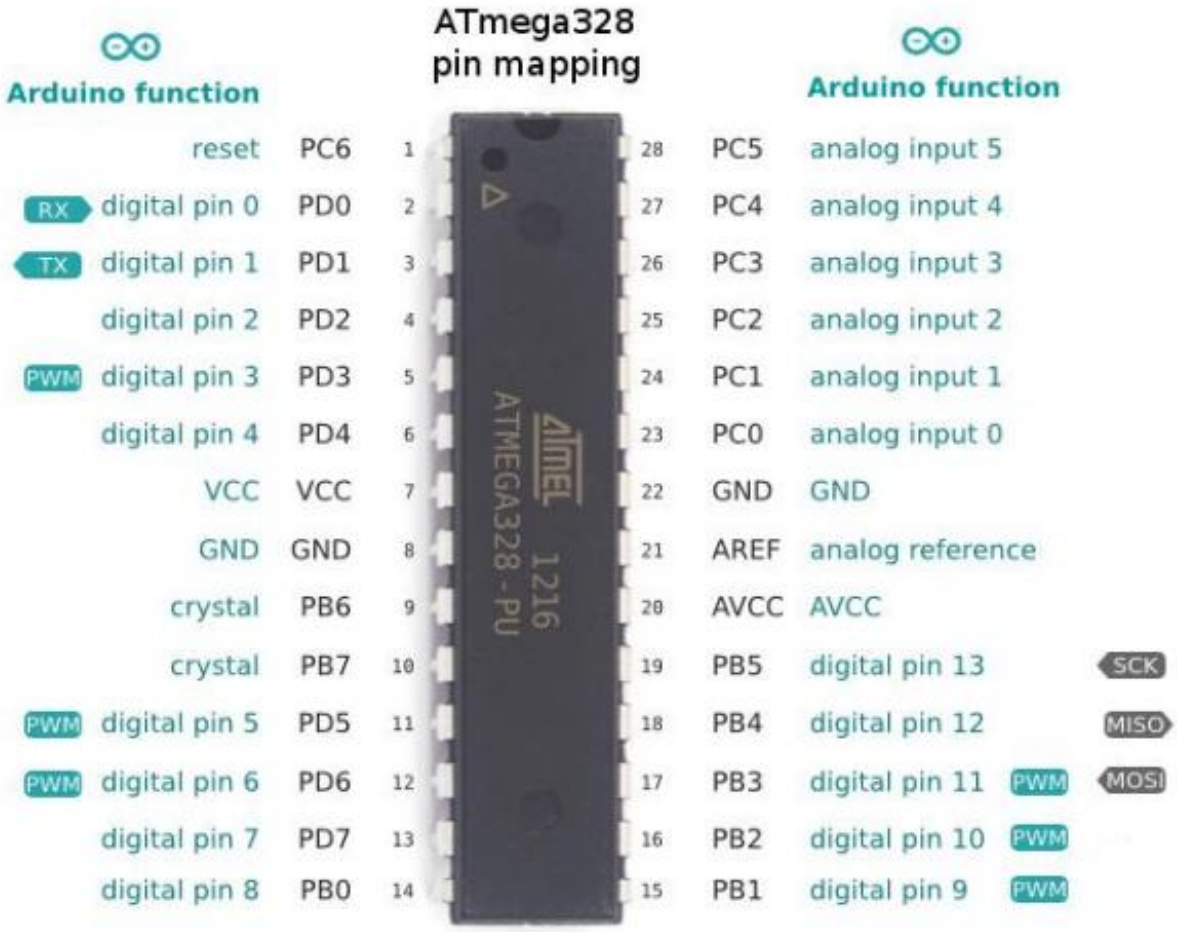


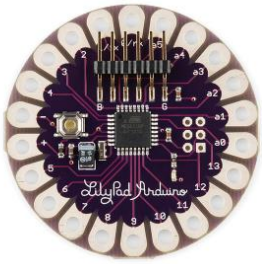
Figura: Atmega 328P Pins

Alte tipuri de ARDUINO

ARDUINO MEGA

Are microcontroler Atmega 2560. Are 54 pini de intrare-ieșire digitali, 16 intrări analogice, 4 porturi seriale hardware și un oscilator de cristal de 16 MHz. Este alimentat atât de adaptor USB sau de curent continuu. În general, cardul, care are aceleași caracteristici ca și ARDUINO UNO, este preferat în proiectele mai mari, deoarece are mai mulți pini.	

ARDUINO LILYPAD

Lilypad este conceput pentru a fi cusut pe rochii și țesături. În acest fel, poate fi utilizat în proiecte interesante care pot fi proiectate pentru a fi purtabile. Are un microcontroler Atmega 168V.	

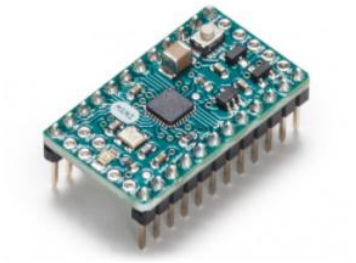
ARDUINO ETHERNET

Are un cip Ethernet și un port Ethernet pentru realizarea de proiecte conectate la internet. Există, de asemenea, un slot pentru card SD, care are modelul Atmega 328 ca microcontroler.	

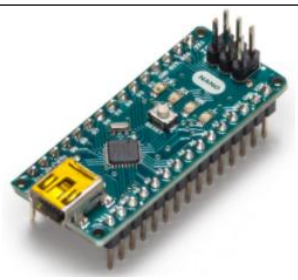
ARDUINO BLUETOOTH

Există un modul Bluetooth pe ARDUINO BT, ideal pentru realizarea aplicațiilor care comunică cu protocolul Bluetooth. Acest modul poate fi utilizat și pentru programarea ARDUINO prin Bluetooth.	

ARDUINO MİNİ

Este un model Arduino conceput pentru a fi acționat pe placă de circuit sau integrat într-un alt design. Pe acesta există microcontrolerul Atmega 168 sau Atmega 328. Este ideal pentru aplicații în care dimensiunile mici sunt deosebit de importante.	

ARDUINO NANO

Este un model foarte mic și proiectat pentru aplicații pe placa de circuit și are un microcontroler Atmega 328 sau Atmega 168, regulator de tensiune, cip convertor serial la USB, port de intrare de tensiune DC și port mini USB.	

ARDUINO LEONARDO

Este una dintre plăcile Arduino, care conține un microcontroler Atmega 32u4 și nu necesită un cip suplimentar pentru conexiunea USB. Cu 20 de intrări / ieșiri digitale și 12 intrări analogice, microcontrolerul de pe placă are un capac de montare la suprafață. Datorită capacităților sale de conectare USB, Leonardo poate fi conectat la computer ca mouse sau tastatură.	

ARDUINO ESPLORA

Esplora este o placă Arduino care conține diverși senzori, spre deosebire de ceilalți. Datorită senzorilor de pe placă de bază, este posibil să efectuați multe aplicații fără a fi nevoie de alte adăugiri și cunoștințe electronice excesive. Esplora este echipat cu un potențiometru glisant, senzor de lumină și sunet, senzor de temperatură, generator de sunet, mini joystick analogic pe 2 axe, LED cu 3 culori și un accelerometru. Esplora este, de asemenea, echipat cu microcontroler Atmega 32U4 AVR ca Leonarda. Aplicațiile care pot acționa ca mouse sau tastatură pot fi dezvoltate atunci când sunt conectate la un computer cu conexiunea sa micro USB.	

Descărcarea IDE-ului Arduino

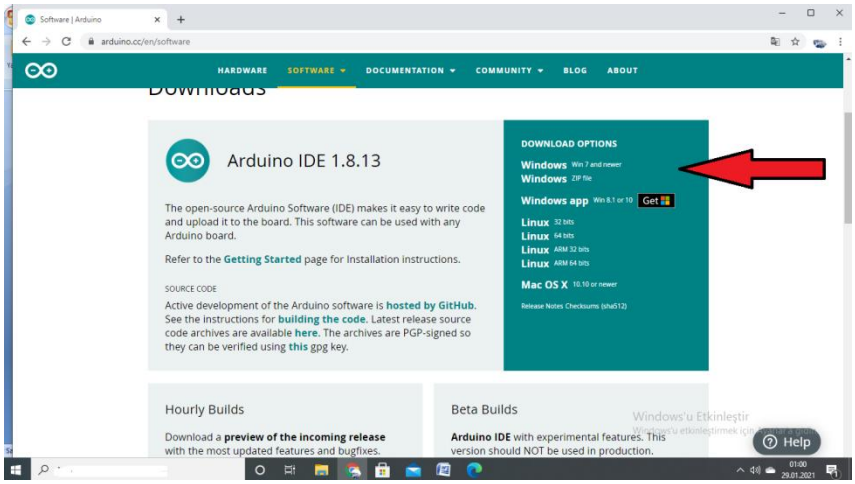
Arduino IDE (Integrated Development Environment) este locul în care se dezvoltă programele care îi vor spune lui Arduino ce trebuie să facă.

Pentru a instala Arduino IDE pentru Windows, trebuie să urmăm instrucțiunile.

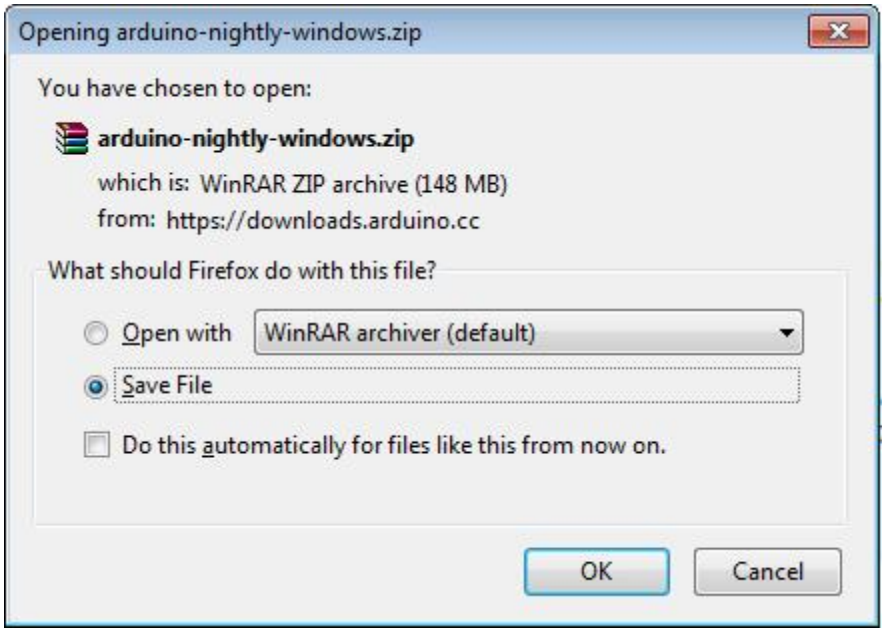
Puteți încărca programe noi pe cipul principal, ATmega328p, prin USB folosind Arduino IDE.

Pentru a descărca ID-ul Arduino, faceți clic pe următorul link:

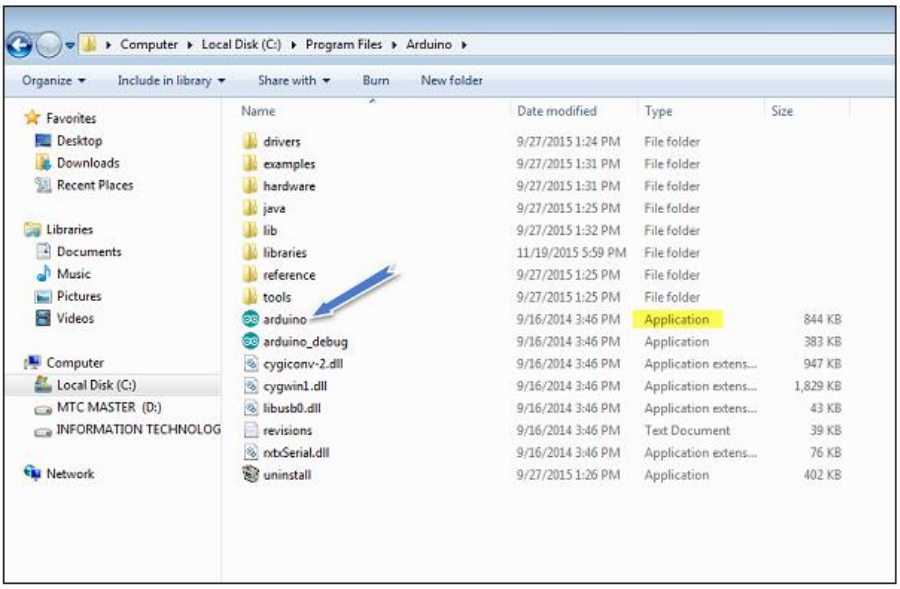
https://www.arduino.cc/en/Main/Software.



Selectați sistemul de operare pe care îl utilizați și descărcați-l. După ce software-ul nostru Arduino IDE este descărcat, trebuie să dezarhivați folderul.

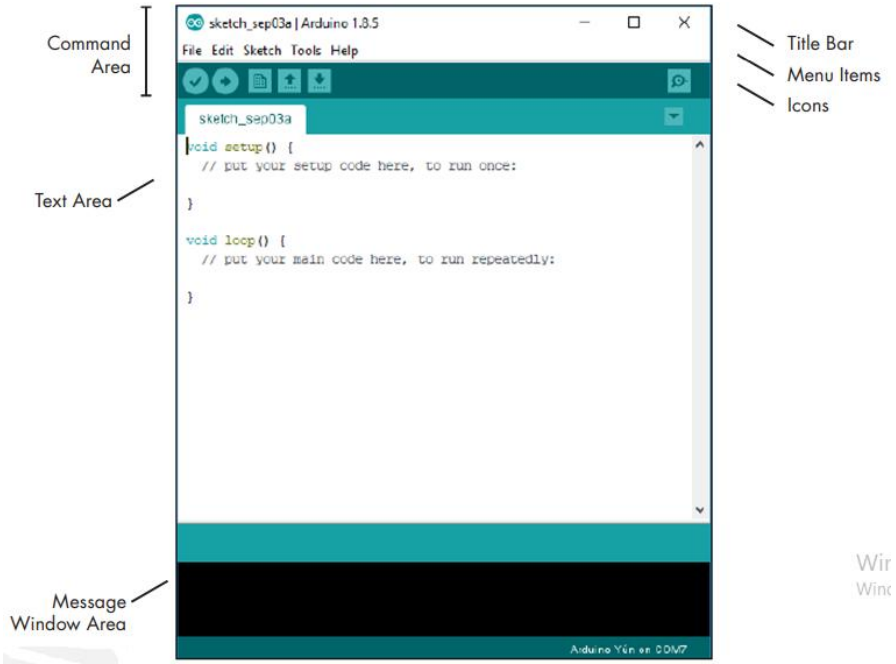


În interiorul folderului, putem găsi pictograma aplicației cu o etichetă infinită (application.exe). Faceți dublu clic pe pictogramă pentru a porni IDE-ul. Apoi, pur și simplu urmați expertul de instalare pentru a instala Arduino IDE.



Fereastra Arduino IDE pentru scrierea programelor

Când deschideți prima dată Arduino IDE, ar trebui să vedeți ceva similar cu figura de mai jos. Așa cum se arată în figura de mai jos, Arduino IDE seamănă cu un procesor de text simplu. IDE este împărțit în trei zone principale: zona de comandă, zona de text și zona ferestrei de mesaje.



Elemente de meniu

Ca și în cazul oricărui procesor de text sau editor de text, puteți face clic pe unul dintre elementele de meniu pentru a afișa diferite opțiuni.

File: Conține opțiuni pentru salvarea, încărcarea și tipărirea programelor (sketch); un set complet de exemple de programe (sketch) de deschis; precum și submeniul Preferences.

Edit: Conține funcțiile obișnuite de copiere, lipire și căutare comune oricărui procesor de text

Sketch: Conține funcția de verificare a programului (sketch) înainte de încărcare pe o placă, precum și câteva dosare de programe (sketch) și opțiuni de import.

Tools: Conține o varietate de funcții, precum și comenzile pentru a selecta tipul de placă Arduino și portul USB.

Help: Conține linkuri către diferite subiecte de interes și versiunea IDE.

Ce este programul (sketch)

Un program (sketch) Arduino este un set de instrucțiuni pe care le creați pentru a îndeplini o anumită sarcină; cu alte cuvinte, un sketch este un program.

Programul (sketch) nu este altceva decât un set de instrucțiuni pentru Arduino. Programele create folosind Arduino IDE sunt salvate ca fișiere .pde. Pentru a crea un program (sketch) trebuie să faceți cele trei părți principale: declarația variabilă, funcția de configurare și funcția principală de buclă.

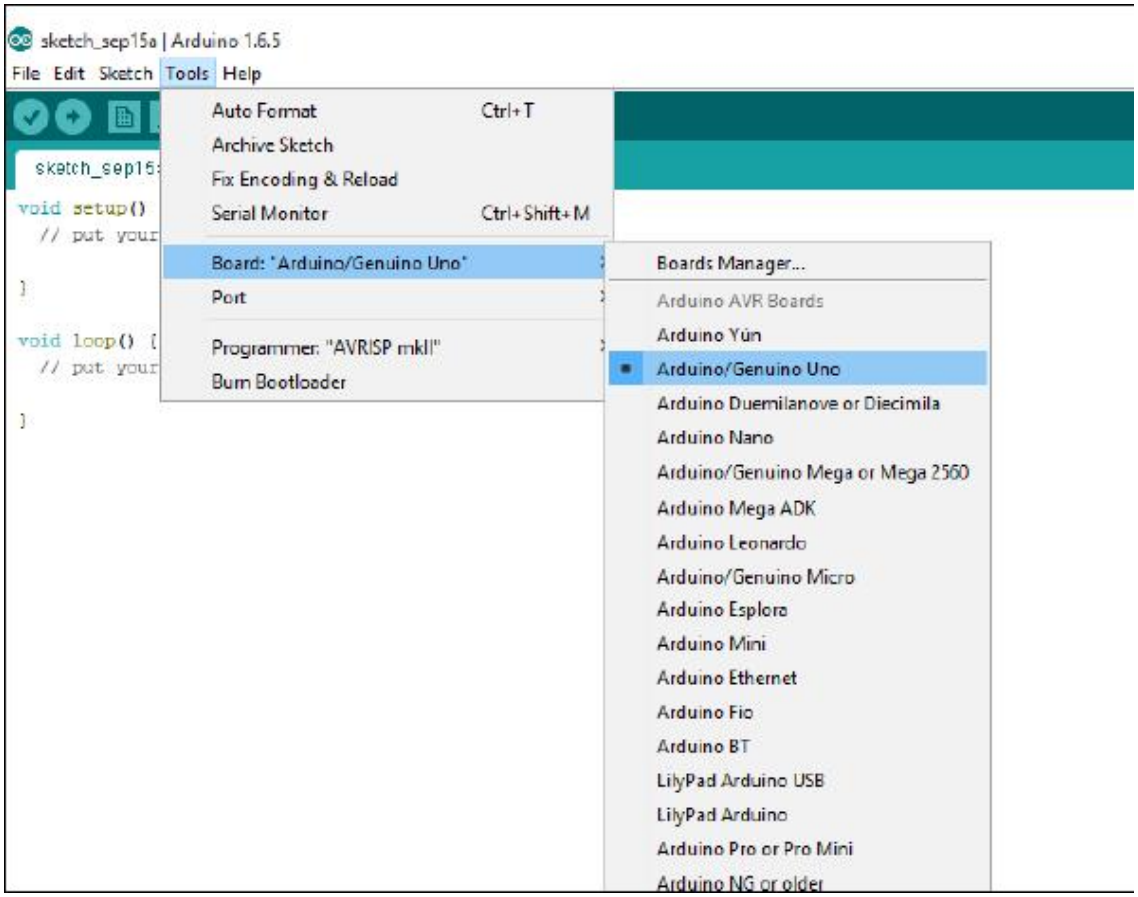
Butoane pentru bara de instrumente IDE Arduino

Sub bara de instrumente a meniului sunt șase pictograme. Treceți cu mouse-ul peste fiecare pictogramă pentru a-i afișa numele. Pictogramele, de la stânga la dreapta, sunt după cum urmează:

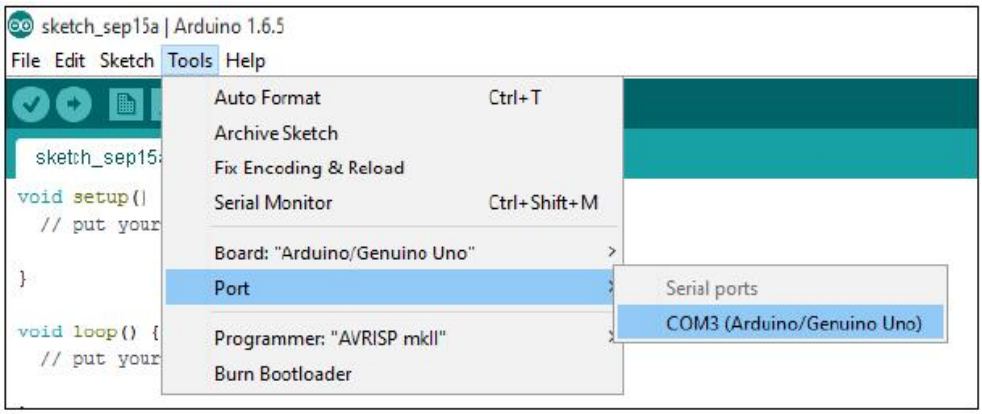
	**Verify (Compile):** Faceți clic pe acesta pentru a verifica dacă programul (sketch) Arduino este valid și nu conține nicio greșeală de programare.
	**New:** Faceți clic pe acesta pentru a deschide un nou program (sketch) gol într-o fereastră nouă.
	**Open:** Faceți clic pe aceasta pentru a deschide un program (sketch) salvat.
	**Save:** Faceți clic pe acesta pentru a salva programul (sketch) deschisă. Dacă nu are un nume, vi se va solicita să creați unul.
	**Upload:** Faceți clic pe acesta pentru a verifica și apoi încărcați programul (sketch) pe placa Arduino.
	**Serial Monitor:** Faceți clic pe acesta pentru a deschide o fereastră nouă pentru utilizare la trimiterea și primirea de date între Arduino și IDE.

Conectarea Arduino

Conectați-vă Arduino UNO la computer prin USB. După conectarea Arduino cu un cablu USB, trebuie să vă asigurați că IDE-ul Arduino a selectat placa potrivită. În cazul nostru, folosim Arduino Uno, deci ar trebui să mergem la **Tools ▸ Board: ▸ Arduino/Genuino Uno.**



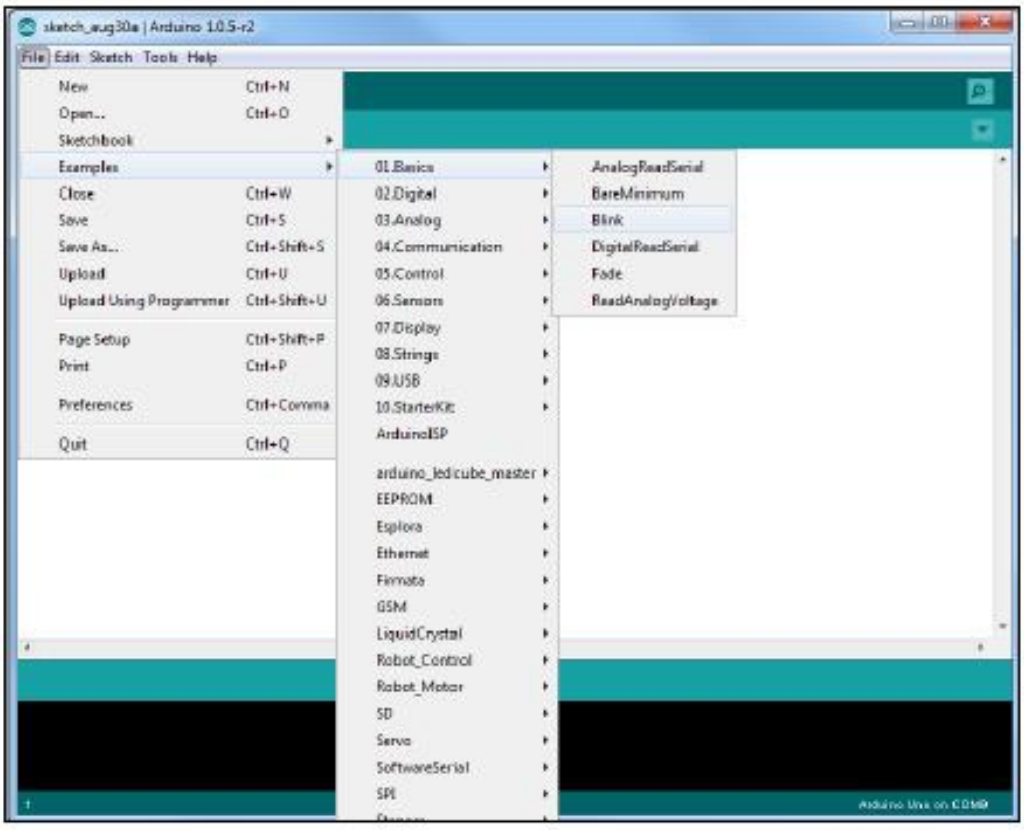
Apoi, ar trebui să selectați portul serial la care este conectat Arduino. Accesați **Tools ▸ Port** și selectați portul potrivit.



Încărcarea unui program Arduino (Sketch)

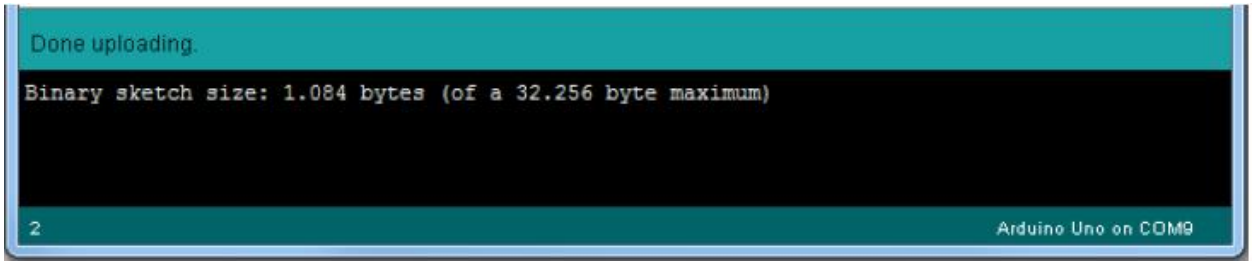
Pentru a vă arăta cum să încărcați codul pe placa Arduino, vă vom arăta un simplu exemplu. Acesta este unul dintre cele mai simple exemple - constă în clipirea LED-ul integrat sau pinul digital 13 în fiecare secundă.

1. Deschideți ID-ul Arduino.
2. Accesați **File ▸ Examples ▸ 01.Basics ▸ Blink**



În mod implicit, Arduino IDE este pre-configurat pentru Arduino UNO. Faceți clic pe butonul **Upload** și așteptați câteva secunde.

După câteva secunde, ar trebui să vedeți un mesaj **Done uploading.**



Acest cod clipește pur și simplu LED-ul integrat de pe Arduino UNO (evidențiat cu culoare roșie). Ar trebui să vedeți micul LED aprins timp de o secundă și oprit pentru încă o secundă în mod repetat.

Controlarea unei ieșiri și citirea unei intrări

O placă Arduino conține pini digitali, pini analogici și pini PWM.

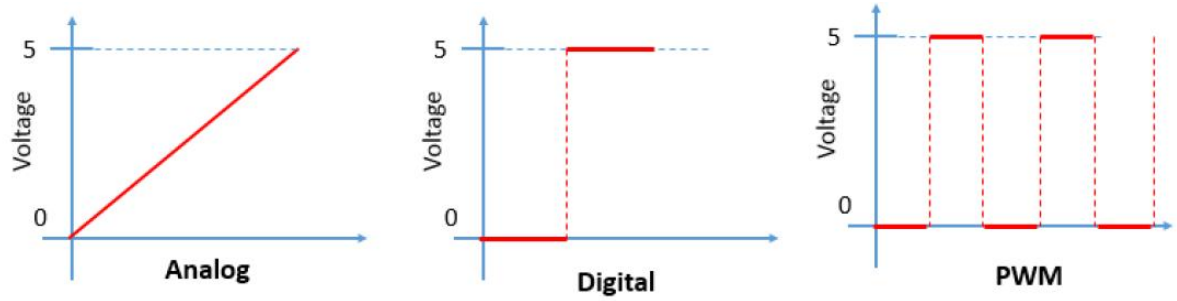
Diferența dintre digital, analog și PWM

În **pinii digitali,** aveți doar două stări posibile, care sunt ON sau OFF. Acestea pot fi, de asemenea, denumite High sau Low, 1 sau 0 și 5V sau 0V.

De exemplu, dacă un LED este aprins, atunci starea sa este High sau 1 sau 5V. Dacă este dezactivat, veți avea Low, sau 0 sau 0V.

În **pinii analogici**, aveți stări posibile nelimitate între 0 și 1023. Acest lucru vă permite să citiți valorile senzorului. De exemplu, cu un senzor de lumină, dacă este foarte întunecat, veți citi 1023, dacă este foarte luminos, veți citi 0, dacă există o luminozitate între întuneric și foarte luminos, veți citi o valoare între 0 și 1023.

Pinii PWM sunt pini digitali, deci produc fie 0, fie 5V. Cu toate acestea, acești pini pot emite valori de tensiune intermediare „false" între 0 și 5V, deoarece pot efectua „Modularea lățimii pulsului" (PWM). PWM permite „simularea" diferitelor niveluri de putere prin oscilarea tensiunii de ieșire a Arduino.



Controlarea unei ieșiri

Pentru a controla o ieșire digitală utilizați funcția digitalWrite () și între paranteze scrieți pinul pe care doriți să îl controlați, apoi HIGH sau LOW.

Pentru a controla un pin PWM utilizați funcția analogWrite () și între paranteze scrieți pinul pe care doriți să îl controlați și un număr între 0 și 255.

Citirea unei intrări

Pentru a citi o intrare analogică utilizați funcția analogRead () și pentru o intrare digitală utilizați digitalRead ().

Notă: Cea mai bună modalitate de a învăța Arduino este practicarea. Deci, faceți multe proiecte și începeți să construiți ceva.

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

Erasmus+ KA-202

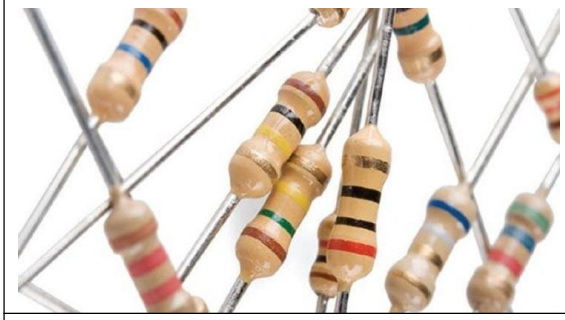
Strategic Partnerships Project for Vocational Education and Training

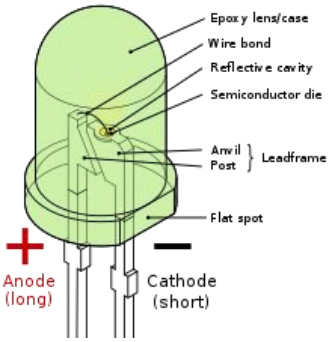
Project Title: "Teaching and Learning Arduinos in Vocational Training"

Project Acronym: " ARDUinVET "

Project No: "2020-1-TR01-KA202-093762"

KİT DE İNSTRUİRE PENTRU MODUL INTRĂRI/IEȘIRI ARDUINO





Planificarea proiectelor noastre

Când începeți primele voastre proiecte, ați putea fi tentați să vă scrieți programul (sketch) imediat după ce ați venit cu o idee nouă. Dar înainte de a începe să scrieți, câțiva pași de pregătire de bază sunt necesari. La urma urmei, placa dvs. Arduino nu este un cititor de minte; are nevoie de instrucțiuni precise și chiar dacă aceste instrucțiuni pot fi executate de Arduino, este posibil ca rezultatele să nu fie ceea ce vă așteptați dacă ați trecut cu vederea chiar și un detaliu minor.

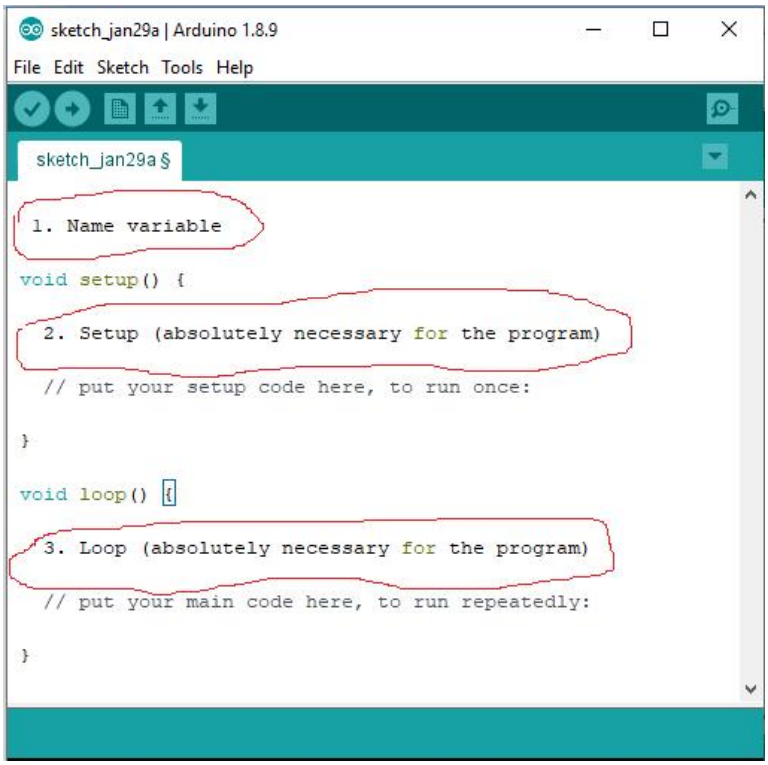
Dacă creați un proiect care să clipească pur și simplu o lumină sau un model automat de semnal feroviar, un plan detaliat este baza succesului. Când vă proiectați proiectele Arduino, urmați acești pași de bază:

1. Definiți-vă obiectivul. Stabiliți ce doriți să realizați.
2. Scrieți algoritmul. Un algoritm este un set de instrucțiuni care descrie cum să vă realizați proiectul. Algoritmul dvs. va enumera pașii necesari pentru a vă atinge obiectivul proiectului.
3. Selectați hardware-ul. Determinați cum se va conecta la Arduino.
4. Scrieți programul (sketch). Creați-vă programul inițial care va spune lui Arduino ce trebuie să facă.
5. Conectați-l. Conectați-vă hardware-ul, circuitele și alte articole la placa Arduino.
6. Testați și depanați. Funcționează? În această etapă, identificați erorile și găsiți cauzele acestora, fie în program (sketch), hardware sau algoritm.

Cu cât petreci mai mult timp planificându-ți proiectul, cu atât vei avea mai ușor timp în etapa de testare și depanare.

Structura de bază a unui program (sketch)

Programul Arduino este numit sketch. Un sketch poate fi împărțită în trei părți.



1. Denumirea variabilelor

În prima parte sunt denumite elementele programului. Această parte nu este absolut necesară.

2. Configurare (absolut necesară pentru program):

Configurarea va fi efectuată o singură dată. Aici îi spuneți programului, de exemplu, ce pin (slot pentru cabluri) ar trebui să fie o intrare și ceea ce ar trebui să fie o ieșire pe placă.

Definit ca ieșire: pinul ar trebui să scoată o tensiune. De exemplu: Cu acest pin un LED este menit să se aprindă.

Definit ca intrare: Placa ar trebui să citească o tensiune. De exemplu: Un comutator este acționat. Placa recunoaște acest lucru, deoarece primește o tensiune pe pinul de intrare.

3. Buclă (absolut necesară pentru program):

Această buclă va fi repetată continuu de către placă. Asimilează programul (sketch) de la început până la sfârșit și începe din nou de la început și așa mai departe.

Alte reguli de sintaxă

Este necesar să acordați atenție acestor reguli în timp ce scrieți programele Arduino. În caz contrar, programul vostru va eșua.

; (Punct şi virgulă)

; (Punct și virgulă) este folosit pentru a termina o afirmație. Dacă o linie nu se termină în punct și virgulă va duce la o eroare a compilatorului.

Exemplu: int a=13;

{} Acolade

{} Acoladele sunt o parte importantă a limbajului de programare Arduino. Acestea sunt utilizate în mai multe construcții diferite, iar acest lucru poate fi uneori confuz pentru începători. O paranteză „{” trebuie să fie întotdeauna urmată de o paranteză „}”.

Principalele utilizări ale acoladelor: funcții, bucle, declarații condiționale

Exemplu:

```
void myfunction(datatype argument)
{           statements(s)                    }
```

// (Comentariu cu o singură linie) şi // (Comentariu pe mai multe linii)**

Acestea sunt liniile din program care sunt folosite pentru a vă informa pe voi sau pe ceilalți despre modul în care funcționează programul. Acestea sunt ignorate de compilator și nu sunt exportate în procesor, deci nu ocupă spațiu pe cipul Atmega. Scopul comentariilor este de a vă ajuta să înțelegeți (sau să vă amintiți) cum funcționează programul dumneavoastră sau să îi informați pe ceilalți cum funcționează programul dumneavoastră. Există două moduri diferite de a marca o linie ca un comentariu:

Exemple:

```
x = 5;          // Acesta este un comentariu cu o singură linie. Orice după linii oblice este un comentariu
//  sfârșitul liniei
```

x = 5; /*Acesta este un comentariu multi linie ……
Sfârșitul comentariului multi linie */

 # define:

 # define permite programatorului să dea un nume unei valori constante înainte ca programul să fie compilat. Constantele definite în Arduino nu ocupă spațiul de memorie al programului pe cip. În general, cuvântul cheie **const** este preferat pentru definirea constantelor și trebuie utilizat în locul #define.

Exemplu:

#define ledPin 3 // Compilatorul va înlocui orice mențiune a ledPin cu valoarea 3 la momentul compilării.

 # include:

#include este folosit pentru a include bibliotecile externe în programul (sketch - ul) dvs. Acest lucru oferă programatorului acces la un grup mare de biblioteci C standard (grupuri de funcții prestabilite), precum și biblioteci scrise special pentru Arduino.

Exemplu:

#include <servo.h>

Arduino - Tipuri de date

Tipurile de date sunt utilizate pentru declararea variabilelor sau funcțiilor de diferite tipuri. Tipul unei variabile determină cât spațiu ocupă în stocare și modul în care modelul de biți stocat este interpretat.

Expresiile care sunt folosite pentru a stoca orice informație în memorie și care pot modifica valoarea în timpul fluxului programului se numesc variabile. Variabilele pot fi numere, caractere sau expresii logice. Tipul de date adecvat trebuie selectat în funcție de tipul variabilei. O anumită zonă este alocată în funcție de tipul de date, utilizate în definirea variabilei din memorie.

Următorul tabel oferă toate tipurile de date pe care le veți utiliza în timpul programării Arduino.

Tip	Conținut
boolean	poate fi adevărat sau fals
char	-128 până la 127
byte	0 până la 255
unsigned char	0 până la 255
int	-32,768 până la 32,767
unsigned int	0 până la 65,535
word	(la fel ca unsigned int)
long (or long int)	-2,147,483,648 până la 2,147,483,647
unsigned long	0 până la 4,294,967,295
float	-3.4028235E+38 până la 3.4028235E+38
double	(la fel ca float)

Arduino - Variabile și Constante

La definirea variabilei, trebuie determinate numele variabilei, valoarea variabilei și tipul de date adecvat pentru variabilă.

Definiția este făcută așa cum se vede în exemplul de mai jos:

int LED =12;

Aici: **int** = tipul de dată **LED** = numele variabilei **12** = valoarea variabilei

Variabilele, pe care le folosește Arduino, au o proprietate numită scope. Scope este o regiune a programului și acolo există trei locuri în care variabilele pot fi declarate. Acestea sunt:

- In interiorul unei funcții sau a unui bloc, care se numește variabile locale.
- În definiția parametrilor funcției, care se numește parametri formali.
- În afara tuturor funcțiilor, care se numește variabile globale.

O constantă este un calificativ de variabilă care modifică comportamentul variabilei, făcând o variabilă „numai în citire”. Aceasta înseamnă că variabila poate fi utilizată la fel ca orice altă variabilă de tipul ei, dar valoarea ei nu poate fi modificată. Veți primi o eroare de compilator dacă încercați să atribuiți o valoare unei variabile const.

Constantele definite cu cuvântul cheie **const** respectă regulile de amplificare a variabilelor care guvernează alte variabile. Acest lucru și capcanele utilizării #define, fac din cuvântul cheie const o metodă superioară pentru definirea constantelor și este preferat în locul utilizării #define.

Exemplu:

```
const float pi = 3.14;
```

Note:

#define **or** const: Puteți utiliza fie const, fie #define pentru a crea constante numerice sau de șir. Pentru tablouri, va trebui să utilizați const. În general, const este preferată în locul #define pentru definirea constantelor.

Arduino – Operatori

Un operator este un simbol care îi spune compilatorului să îndeplinească funcții matematice sau logice specifice. În Arduino, pot fi utilizate următoarele tipuri de operatori.

Operatori aritmetici:

Să presupunem că variabila A este 10 și variabila B este 20 atunci:

Denumire **operator**	**Simbol operator**	**Exemplu**
egal	=	A = B
adunare	+	A + B egal 30
scădere	-	A - B egal -10
înmulțire	*	A * B egal 200
împărțire	/	B / A egal 2
restul diviziunii între numere	%	B % A egal 0

Operatori de comparație

Să presupunem că variabila A este 10 și variabila B este 20 atunci:

Denumire **operator**	**Simbol operator**	**Exemplu**
egal cu	==	(A == B) nu este adevărat
diferit	!=	(A != B) este adevărat
mai mic decât	<	(A < B) este adevărat
mai mare decât	>	(A > B) nu este adevărat
mai mic sau egal cu	<=	(A <= B) este adevărat
mai mare sau egal cu	>=	(A >= B) nu este adevărat

Operatori Booleni

Să presupunem că variabila A este 10 și variabila B este 20 atunci:

Denumire **operator**	**Simbol operator**	**Exemplu**
și	&&	(A && B) este adevărat
sau	||	(A || B) este adevărat
nu	!	!(A && B) este fals

Operatori Bitwise

Să presupunem că variabila A este 60 și variabila B este 13 atunci:

Denumire **operator**	**Simbol operator**	**Exemplu**
și	&	(A & B) will give 12 which is 0000 1100
sau	|	(A | B) will give 61 which is 0011 1101
xor	^	(A ^ B) will give 49 which is 0011 0001
nu	~	(~A) will give -60 which is 1100 0011
deplasare la stânga	<<	A << 2 will give 240 which is 1111 0000
deplasare la dreapta	>>	A >> 2 will give 15 which is 0000 1111

Arduino - Funcții I / O (comenzi)

Funcțiile permit structurarea programelor pentru îndeplinirea sarcinilor individuale. Cazul tipic pentru crearea unei funcții este atunci când trebuie să efectuați aceeași acțiune de mai multe ori într-un program. Acestea vor deveni mai clare atunci când vom arăta exemple de programe reale în circuite și programe Arduino. Pentru a explica diversele funcții de comandă, le-am împărțit în comenzi separate

Pinii de pe placa Arduino pot fi configurați ca intrări sau ieșiri. Vom explica funcționarea pinilor în aceste moduri. Este important să rețineți că majoritatea pinilor analogici Arduino pot fi configurați și utilizați, exact în același mod ca pinii digitali.

Funcția pinMode():

Funcția pinMode() este folosită pentru a configura un pin specific pentru a se comporta fie ca intrare, fie ca ieșire. Este posibil să activați rezistențele interne pull - up cu modul INPUT_PULLUP.

Sintaxa: pinMode(pin, mode)

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

Parametri:
pin: pin -ul ARDUINO: numărul pinului căruia îi setăm modul
mode: INPUT, OUTPUT, or INPUT_PULLUP.

Exemple:

```
pinMode(13, OUTPUT);      // setează pinul 13 digital ca ieșire
pinMode(5, INPUT);        // setează pinul 5 digital ca intrare
```

Funcția digitalWrite():

Funcția **digitalWrite()** este utilizată pentru a scrie o valoare HIGH sau LOW pe un pin digital. Dacă pinul a fost configurat ca o IEȘIRE cu pinMode(), tensiunea acestuia va fi setată la valoarea corespunzătoare: 5V pentru HIGH, 0V (masă) pentru LOW. Dacă pinul este configurat ca INPUT, digitalWrite() va activa (HIGH) sau va dezactiva (LOW) pull-up-ul intern de pe pinul de intrare. Se recomandă setarea pinMode() la INPUT_PULLUP pentru a permite pull-up-ul intern.

Dacă nu setați pinMode() ca OUTPUT și conectați un LED la un pin, atunci când apelați digitalWrite(HIGH), LED-ul poate lumina slab. Fără a seta în mod explicit pinMode(),

digitalWrite() va fi activat rezistorul de pull-up intern, care acționează ca un rezistor mare de limitare al curentului.

Sintaxă:

digitalWrite (pin ,value);

pin: numărul pinului al cărui mod dorești să-l setezi

value: HIGH (1), sau LOW(0).

Exemplu:

```
digitalWrite(LED, HIGH);          // aprinde LED
digitalWrite(LED, LOW);           // stinge LED
```

Funcția digitalRead():

Funcția **digitalRead()** citește valoarea de la un pin digital specificat, fie HIGH, fie LOW.
Sintaxa: digitalRead(pin)
pin: numărul pinului ARDUINO pe care vrei să-l citească
Exemplu:

```
val = digitalRead(inPin);   // citește pinul de intrare
```

Notă: Pinii analogici de intrare pot fi folosiți ca pini digitali, menționați ca A0, A1, etc.

Funcția delay():

Funcția **delay()** întrerupe programul pentru perioada de timp (în milisecunde) specificată ca parametru (Există 1000 de milisecunde într-o secundă).
Sintaxa: delay(ms):

ms: numărul de milisecunde de întrerupere.

Tipul de dată permis: unsigned long.

Exemple:

```
delay(1000);                // așteaptă 1 secundă
delay(2000);                // așteaptă 2 secunde
```

Funcția analogWrite():

Funcția **analogWrite()** scrie o valoare analogică (undă PWM) pe un pin. Poate fi folosit pentru a aprinde un LED la diferite luminozități sau pentru a conduce un motor la diferite viteze. După un apel către analogWrite(), pinul va genera o undă dreptunghiulară constantă a ciclului de funcționare specificat până la următorul apel către analogWrite() (sau un apel către digitalRead() sau digitalWrite()) pe același pin.

Exemplu:

Setează ieșirea LED - ului proporțională cu valoarea citită la potențiometru

```
val = analogRead(analogPin);   // citește pinul de ieșire
```

```
analogWrite(ledPin, val / 4);   // analogRead are valoare de la 0 la 1023, analogWrite are valoare de la 0 la 255
```

Funcția analogRead():

Arduino este capabil să detecteze dacă există o tensiune aplicată la unul dintre pinii săi și să o raporteze prin intermediul funcției digitalRead(). Există o diferență între un senzor de pornire/oprire (care detectează prezența unui obiect) și un senzor analogic, a cărui valoare se modifică continuu. Pentru a citi acest tip de senzor, avem nevoie de un alt tip de pin.

În partea din dreapta jos a plăcii Arduino, veți vedea șase pini marcați „Analog In". Acești pini speciali nu numai că spun dacă există o tensiune aplicată acestora, ci și valoarea acesteia. Folosind funcția analogRead (), putem citi tensiunea aplicată unuia dintre pini.

Această funcție returnează un număr între 0 și 1023, care reprezintă tensiuni între 0 și 5 volți. De exemplu, dacă există o tensiune de 2,5 V aplicată pinului 0, analogRead (0) returnează 512.

Sintaxa: analogRead(pin);

pin: numărul pinului analog de citit de la 0 la 5

Exemplu:

```
val = analogRead(analogPin);        // citește pinul de intrare
Serial.println(val);                // valoare de depanare
```

Funcția if:

Instrucțiunea **if** verifică dacă există o condiție și execută următoarea instrucțiune sau set de instrucțiuni dacă condiția este „adevărată".

Sintaxa:

```
if (condition)
{                                                    //instrucțiune/instrucțiuni
                                                                                    }
```

condiție: o expresie booleană (i.e., poate fi Adevărat sau Fals).

Exemplu: Parantezele pot fi omise după o instrucțiune if. Dacă se face acest lucru, următoarea linie (definită de punct și virgulă) devine singura afirmație condițională.

```
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);
}
            // toate sunt corecte
```

Comanda if-else:

Comanda if .....else permite un control mai mare asupra fluxului de cod decât instrucțiunea de bază if, permițând gruparea mai multor teste. O clauză else (dacă există) va fi executată dacă condiția din instrucțiunea if are ca rezultat Fals. Comada else poate continua un alt if test, astfel încât mai multe teste care se exclud reciproc să poată fi executate în același timp.

Fiecare test va trece la următorul până când se va întâlni un test adevărat. Când se găsește un test adevărat, se execută blocul său de cod asociat, iar programul sare apoi la linie urmând întreaga construcție if/else. Dacă niciun test nu se dovedește a fi adevărat, blocul implicit else este executat, dacă există unul, și setează comportamentul implicit. Un număr nelimitat de asemenea, else dacă sunt permise ramurile.

Sintaxa:

```
if (condition is TRUE) {
  // face lucrul A
}
else
  //dacă NOT,  face lucrul B
}
```

```
if (condition1) {
  // face lucrul A
}
else if (condition2) {
  // face lucrul B
}
else {
  // face lucurl C
}
```

Exemple: (Mai jos este un extras dintr-un cod pentru sistemul senzorului de temperatură)

```
if (temperature >= 70) {
  // Pericol! Închide sistemul.
}
else if (temperature >= 60) { // 60 <= temperature < 70
  // Avertizare! Necesită atenție.
}
else { // temperature < 60
  // Sigur! Continuă sarcinile obișnuite.
}
```

Comanda for:

Instrucțiunea **for** este utilizată pentru a repeta un bloc de enunțuri închise între acolade. Un contor de incrementare este de obicei folosit pentru a crește și termina bucla. Instrucțiunea for este utilă pentru orice operație repetitivă și este adesea utilizată în combinație cu tablouri pentru a opera pe colecții de date/pini.

Sintaxa:

```
for (initialization; condition; increment) {
 //declarație/ declarații;
}
```

Parametrii:

inițializarea: se face prima și exact o dată

condiția: de fiecare dată prin buclă, starea este testată; dacă este adevărat, blocul de instrucțiuni și incrementul sunt executate, atunci condiția este testată din nou. Când condiția devine falsă, bucla se termină.

incrementul: se execută după fiecare buclă dacă condiția este adevărată

Examples:

```
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);       }
```

Comanda switch ..... case:

Ca și instrucțiunea **if**, comanda **switch/case** controlează fluxul de programe permițând programatorului să specifice coduri diferite care ar trebui executate în diferite condiții. În special, o declarație switch compară valoarea unei variabile cu valorile specificate în instrucțiunile de caz. Când se găsește o instrucțiune de caz a cărei valoare se potrivește cu cea a variabilei, se execută codul din instrucțiunea de caz.

Cuvântul cheie **break** părăsește instrucțiunea switch și este de obicei utilizat la sfârșitul fiecărui caz. Fără o instrucțiune de pauză, instrucțiunea switch va continua să execute următoarele expresii („cădere”) până la o pauză sau până la sfârșitul instrucțiunii switch.

Sintaxa:

```
switch (var) {
  case label1:
    // declarație
    break;
  case label2:
    // declarație
    break;
  default:
    // declarație
    break;
}
```

Exemplu de cod:

```
switch (var) {
  case 1:
    //face ceva când variabila este 1
    break;
  case 2:
    //face ceva când variabila este 2
    break;
  default:
    // dacă nu se potrivește nimic, face mod implicit
    // modul implicit este opțional
    break;
}
```

var: o variabilă a cărei valoare să fie comparată cu diferite cazuri.

Tipuri de date permise: int, char.

label1, label2: constantele

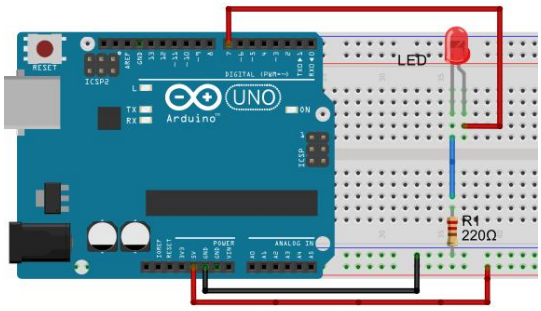
Tipuri de date permise: int, char.

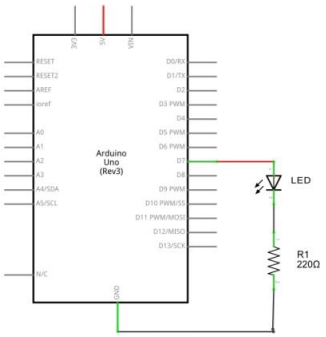
Notă:

Circuitele Arduino pot fi prezentate în două moduri;

1-) vizualizare Breadboard

2-) vizualizare Schematică





1-) Vedere Breadboard

2-) Vedere Schematică

Vom utiliza vederea Breadboard care este mai folosită

Destul cu vorbăria! Haideți să facem ceva!

Circuite Arduino cu Butoane și LED - uri

Majoritatea pinilor de pe Arduino pot fi configurați ca intrare sau ieșire. Setul de butoane și module LED este primul kit de formare al ARDUinVET pentru a face elevii să învețe sistemele I/O ale Arduino. Deci, poate fi numit kit de antrenament I/O Arduino. În acest kit de antrenament, așa cum se arată mai jos, 6 butoane sunt conectate la Arduino ca intrări. Și un buzzer, un conector cu 2 pini și 6 LED-uri sunt conectate ca ieșiri.

Deoarece elevii pot folosi acest kit de formare pentru a învăța sistemele I/O ale Arduino, acest kit de formare le poate face să testeze circuitul Arduino mai ușor. De asemenea, pot folosi o placă de testare pentru a testa circuitele și experimentele Arduino.

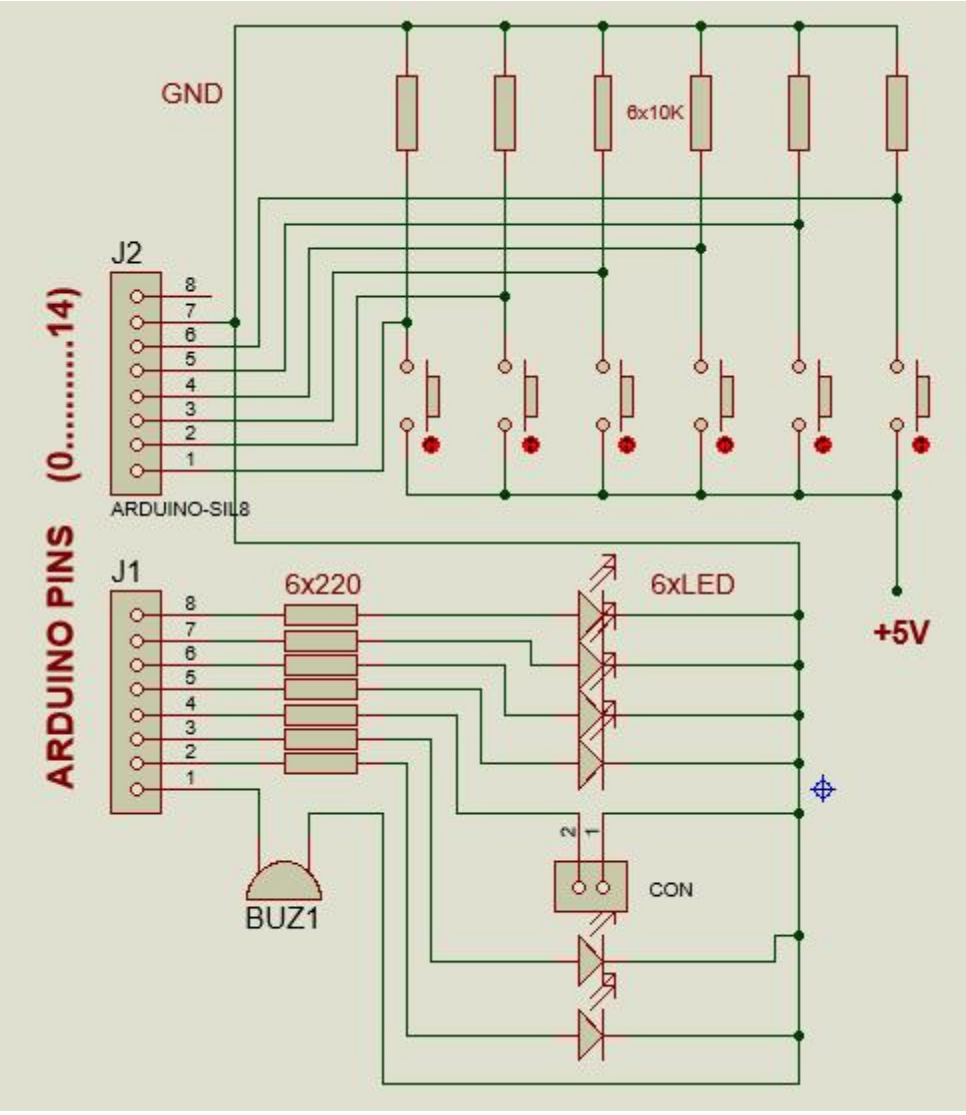


Figura: Circuitul kit -ului modulului Butoane și LED - uri

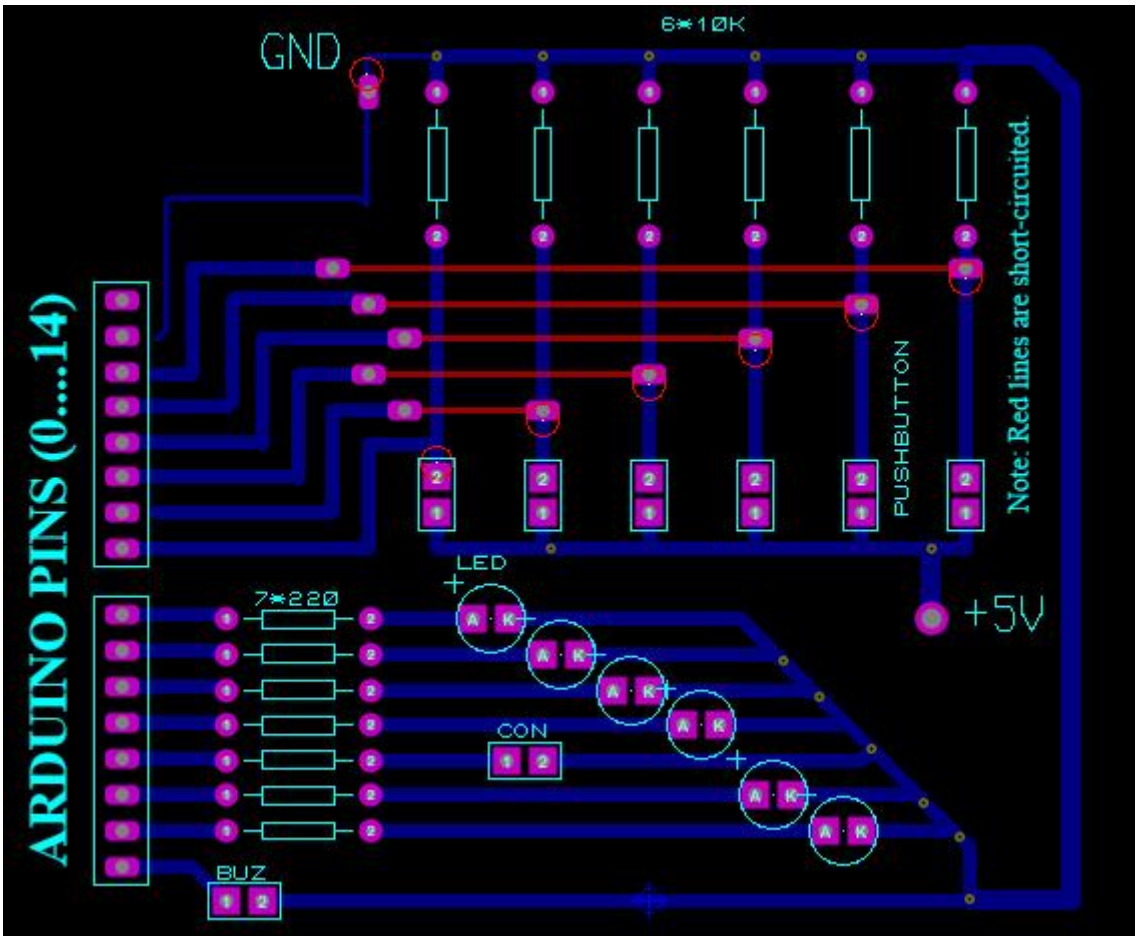


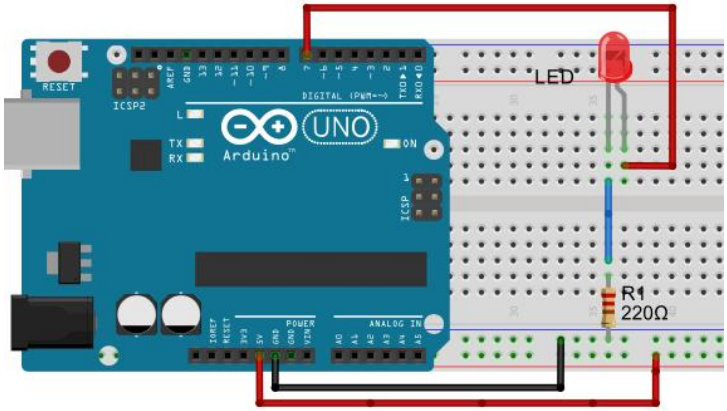
Figura: Schema PCB kit-ului modulului Butoane și LED - uri

Exemple de circuite și programe

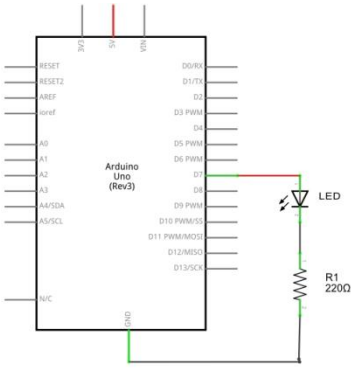
Circuit 1:

Numele circuitului: Program LED intermitent

Descrierea circuitului: Un LED conectat la pinul 13 ARDUINO care va lumina intermitent la interval de 1 secundă.



1-) Vedere Breadboard



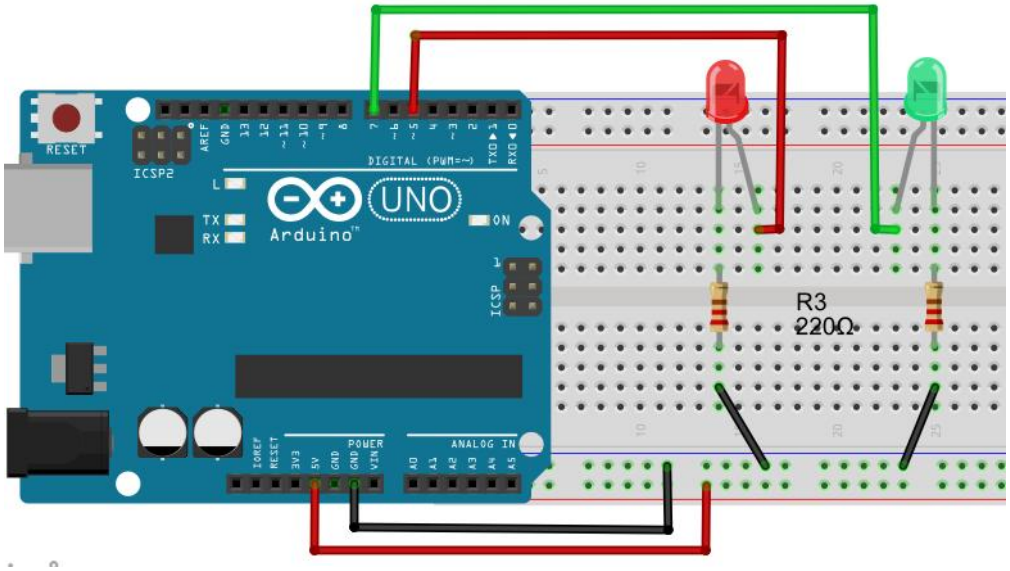
2-) Vedere Schematică

```
/*    Program aprindere intermitentă LED    */

int led = 7;                    // declarare variabila LED întreagă

void setup() {                  // metoda setup() este executată o singură dată
pinMode(led, OUTPUT);           // pinul LED - ului este declarat ca ieșire digitală
}

void loop() {                   // metoda loop() se repetă
digitalWrite(led, HIGH);        // aprinde LED
delay(1000);                    // opreșet program pentru 1000 millisecunde
digitalWrite(led, LOW);         // stinge LED
delay(1000);                    // oprește program pentru 1000 millisecunde
}
```

Circuitul 2:

Titlul circuitului: Program 2 LED-uri care clipesc intermitent

Descrierea circuitului: Flip Flop; 2 LED-uri luminează alternativ la inteval de 2 secunde.

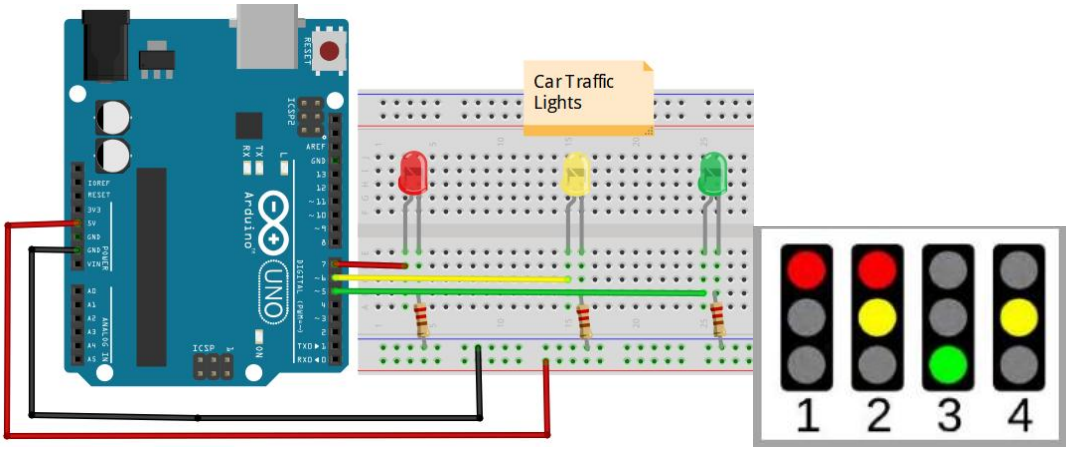



```
/*    Flip Flop     */
int greenLED=5;                    // Pinul la care este conectat LED-ul verde
int redLED=7;                      // Pinul la care este conectat LED-ul roșu
void setup()  {
pinMode(greenLED, OUTPUT);         // pinul LED-ului verde este inițializat ca ieșire
pinMode(redLED, OUTPUT);           //pinul LED-ului roșu este inițializat ca ieșire
                }
void loop(){
digitalWrite(greenLED,  HIGH);          // aprindere LED verde
digitalWrite(redLED,  LOW);             // stingere LED roșu
pause(2000);
digitalWrite(greenLED,  LOW);           // stingere LED verde
digitalWrite(redLED,  HIGH);            // aprindere LED roșu
pause(2000);
                }
```

Circuitul 3:

Titlul circuitului: Lumini semafor

Descrierea circuitului: Funcționarea unui semnafor cu 3 LED - uri (verde, galben și roșu)

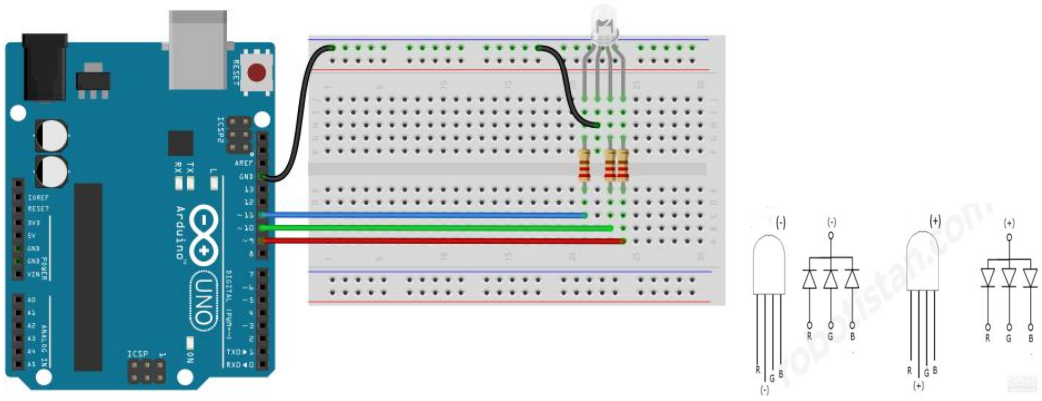



```
/*      Semafor    */

int redLED = 7;
int yellowLED =6;
int greenLED = 5;
void setup() {                          //  inițializăm pinii ca Ieșiri
pinMode(redLED,  OUTPUT);
pinMode(yellowLED,  OUTPUT);
pinMode(greenLED,  OUTPUT);
}
void loop() {
digitalWrite(redLED,  HIGH);            // LED-ul roșu este aprins pentru 9 secunde
digitalWrite(yellowLED,  LOW);
digitalWrite(greenLED,  LOW);
delay(9000);
digitalWrite(redLED,  HIGH);          // LED-urile roșu și galben sunt aprinse pentru 2 secunde
digitalWrite(yellowLED,  HIGH);
digitalWrite(greenLED,  LOW);
delay(2000);
digitalWrite(redLED,  LOW);             // LED-ul verde este aprins pentru 9 secunde
digitalWrite(yellowLED,  LOW);
digitalWrite(greenLED,  HIGH);
delay(9000);
digitalWrite(redLED,  LOW);             // LED-ul galben este aprins pentru 2 secunde
digitalWrite(yellowLED,  HIGH);
digitalWrite(greenLED,  LOW);
delay(2000);
/* Bucla se repetă    */
 }
```

Circuitul 4:

Numele circuitului: Aplicație cu LED RGB

Descrierea circuitului: Circuitul este alcătuit dintr-un LED RGB cu 3 LED-uri, conectate comun și plasate într-o singură casetă, iar ntensitatea luminii pentru cele 3 culori este controlată digital. În plus, putem obține culoarea dorită folosind tehnica PWM.



/* Se va aprinde fiecare culoare a LED-ului RGB la un interval de 1 secundă. Dacă vrem să apindem culoarea albă trebuie să aprindem toate cele trei culori ale LED-ului*/

```
const int BlueLed=11;        // se conectează LED -ul albastru la pinul 11
const int GreenLed=10;       //  se conectează LED -ul verde la pinul 10
const int RedLed=9;          // se conectează LED -ul roșu la pinul 9
        // declarăm pinii la care LED-ul este conectat ca Ieșiri.
void setup()  {
pinMode(BlueLed,OUTPUT);
pinMode(GreenLed,OUTPUT);
pinMode(RedLed,OUTPUT);     }
        // aici ăncepe bucla
void loop() {
digitalWrite(BlueLed, LOW);        // LED-ul roșu este aprins.
digitalWrite(GreenLed, LOW);
digitalWrite(RedLed, HIGH);
delay(1000);
digitalWrite(BlueLed,  LOW );      // LED-ul verde este aprins
digitalWrite(GreenLed, HIGH);
```

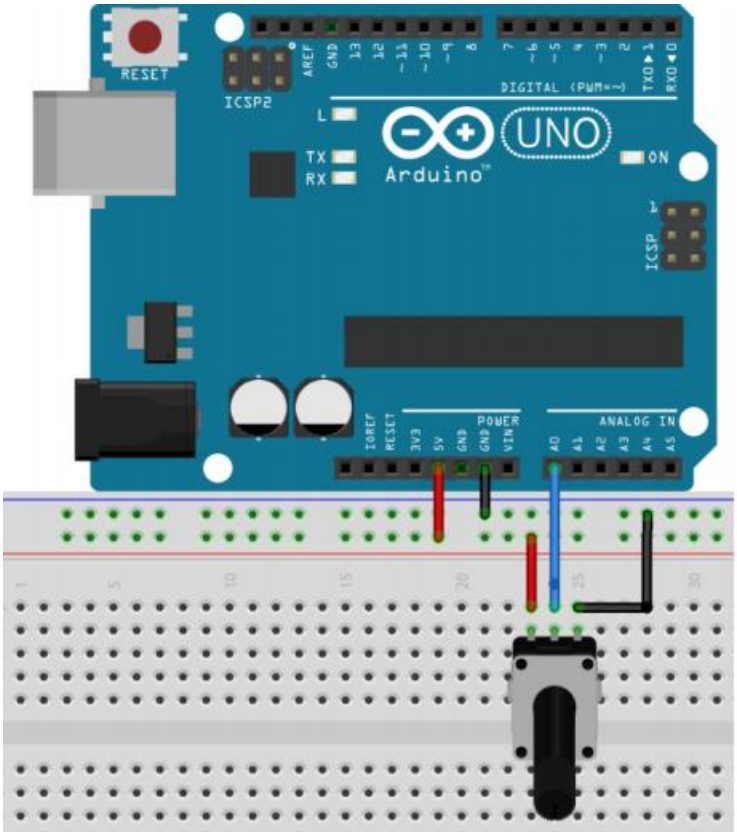
```
digitalWrite(RedLed, LOW );
delay(1000);
digitalWrite(BlueLed,  HIGH);          // LED-ul albastru este aprins
digitalWrite(GreenLed,  LOW);
digitalWrite(RedLed, LOW);
delay(1000);
        // afișăm culoarea alb prin aprinderea celor 3 LED -uri
digitalWrite(BlueLed,  HIGH);
digitalWrite(GreenLed,  HIGH);
digitalWrite(RedLed,  HIGH);
delay(1000);
      }
```

Circuitul 5:

Titlul circuitului: Citirea tensiunii analogice, Citirea valorii de la un potențiometru

Explicarea circuitului: Circuitul ne învață cum să citim un semnal de intrare analogic de la pinul 0. Semnalul de intrarea este convertit de analogRead() în tensiune și afișat pe un monitor serial Arduino IDE.

Potențiometrul: Un potențiometru este un traductor simplu electromecanic, care convertește mișcarea rotativă sau liniară de la operatorul de intrare într-o schimbare de rezistență. Această schimbare este (sau poate fi) utilizată pentru a controla orice, de la volumul unui sistem hi-fi la direcția unei nave imense de containere.	



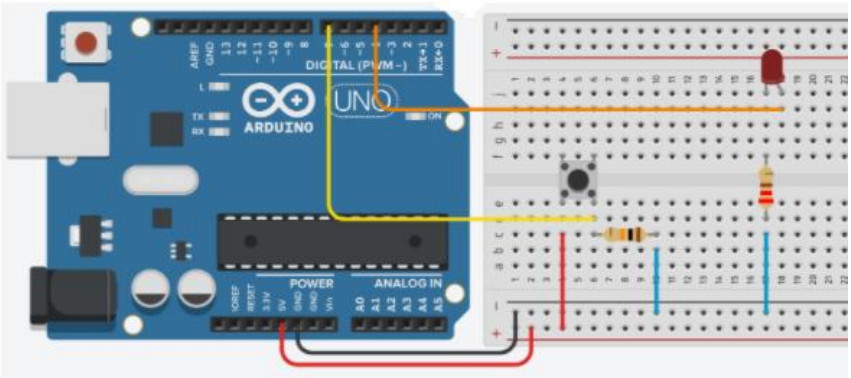
/* ReadAnalogVoltage: Citirea tensiunii de intrare analogice de la pinul 0, convertirea ei în tensiune și afișarea rezultatului pe un monitor serial.
Reprezentarea grafică este valabilă folosind un ploter serial (Tools > Serial Plotter menu).
Se atașează pinul central al potențiometrului la pinul A0 și pinii exteriori la +5V și masă. */

```
        // programul va rula o dată la apăsarea butonului Reset
void setup()        {
         // inițializăm comunicația serială la  9600 bits pe secundă:
  Serial.begin(9600); }
        // bucla se repetă continuu:
void loop() {
         // citește ca Intrare pinul A0:
  int sensorValue = analogRead(A0);
        // afișează valoarea citiă:
  Serial.println(sensorValue);
  delay(1000);        // întârzirea dintre citiri pentru stabilitate
}
```

Circuitul 6:

Titlul circuitului: Folosirea butoanelor în Arduino

Descrierea circitului: Când apăsăm butonul atașat pinului 7, butonul se închide și aprinde un LED conectat la pinul digital 4.




```
/* Buton conectat la pinul 7 care aprinde prin apăsare un LED conectat la pinul digital 4 */
void setup()
{
  pinMode(4, OUTPUT);  // pin-4 declarat Ieșire
  pinMode(7, INPUT);     //  pin7 declarat Intrare
}
void loop()
{
   if (digitalRead(7) == HIGH)              //  dacă pinul 7 este High,
    digitalWrite(4, HIGH);                  // LED -ul se aprinde,
    if (digitalRead(7) == LOW)              // dacă pinul 7 este LOW,
     digitalWrite(4, LOW);                  // LED -ul se stinge
}
```

NOTĂ: Comanda IF-THEN poate fi folosită deasemenea pentru conetarea butonului la piniide intrare ARDUINO. Această conectare poate fi făcută în două moduri diferite, conectare Pull_Up sau Pull-down.

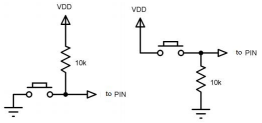
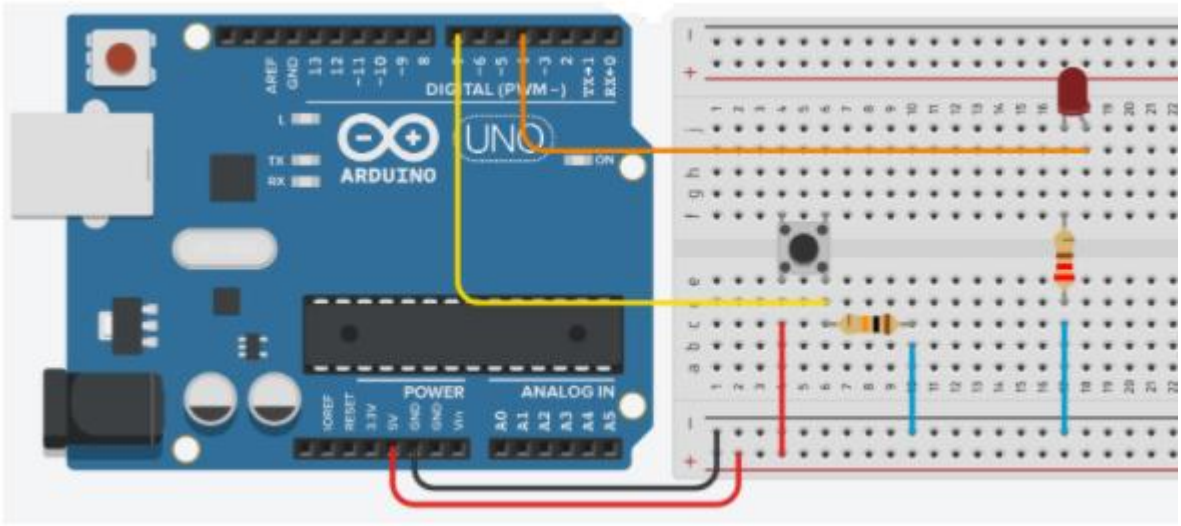


Figura: Comutatoare sau butoane care pot fi folosite pentru comanda IF…THEN

Circuitul 7:

Titlul circuitului: Folosirea comenzii ELSE în Arduino

Descrierea circuitului: Când apăsăm butonul atașat pinului 7, the circuitul se închide și aprinde LED-ul conectat la pinul digital 4. De asemenea, vom învăța cum determinăm pinii cu variabila "int".

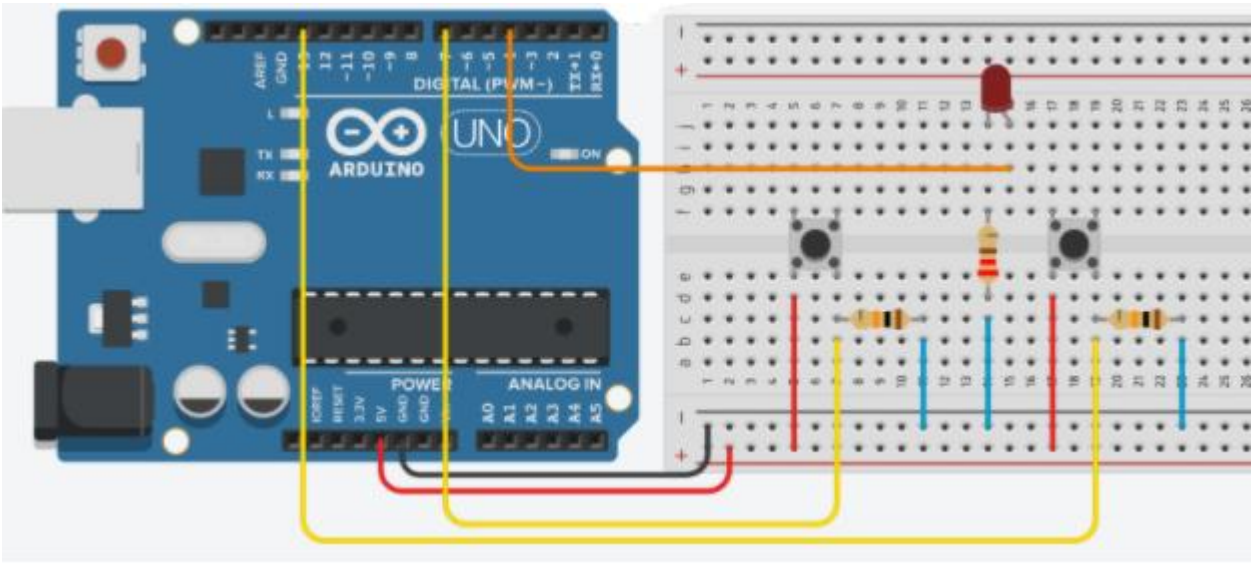



```
int led = 4;
int buton =7;
void setup()
{
  pinMode(led, OUTPUT);
  pinMode(buton, INPUT);
}
void loop()
{
    if (digitalRead(buton) == HIGH)          // Butonul este Închis
       digitalWrite(led, HIGH);              // LED-ul este aprins
     else                                    // Dacă nu,
       digitalWrite(led, LOW);               // LED-ul este stins
}
```

Circuitul 8:

Titlul circuitului: Două butoane și 1 LED

Descrierea circuitului: Un buton aprinde LED-ul, celălalt stinge LED-ul




```
/*      2 butoane și 1 LED
Butonul este conectat cu o rezistență de 10K în serie.        */

int led = 4;                    //  pinul 4 este declarat LED
int button1 = 7;                // pinul 7 este declarat Buton 1
int button2 = 13;               // pinul 13 este declarat Buton 2
void setup()
{
  pinMode(led, OUTPUT);                 // LED declarat Ieșire
  pinMode(button1, INPUT);               //  Butonul 1 declarat Intrare
  pinMode(button2, INPUT);               // Butonul 2 declarat Intrare
}
void loop()
{
  if (digitalRead(button1) == HIGH)         // dacă Butonul 1 este apăsat
     digitalWrite(led, HIGH);                // LED-ul se aprinde
    if (digitalRead(button2) == HIGH)       // dacă Butonul 2 este apăsat
     digitalWrite(led, LOW);                // LED-ul se stinge
}
```

Cum se utilizează SERIAL MONITOR în ARDUINO

ID-ul Arduino are o caracteristică care poate fi de mare ajutor în depanarea programelor sau controlul Arduino de pe tastatura computerului.

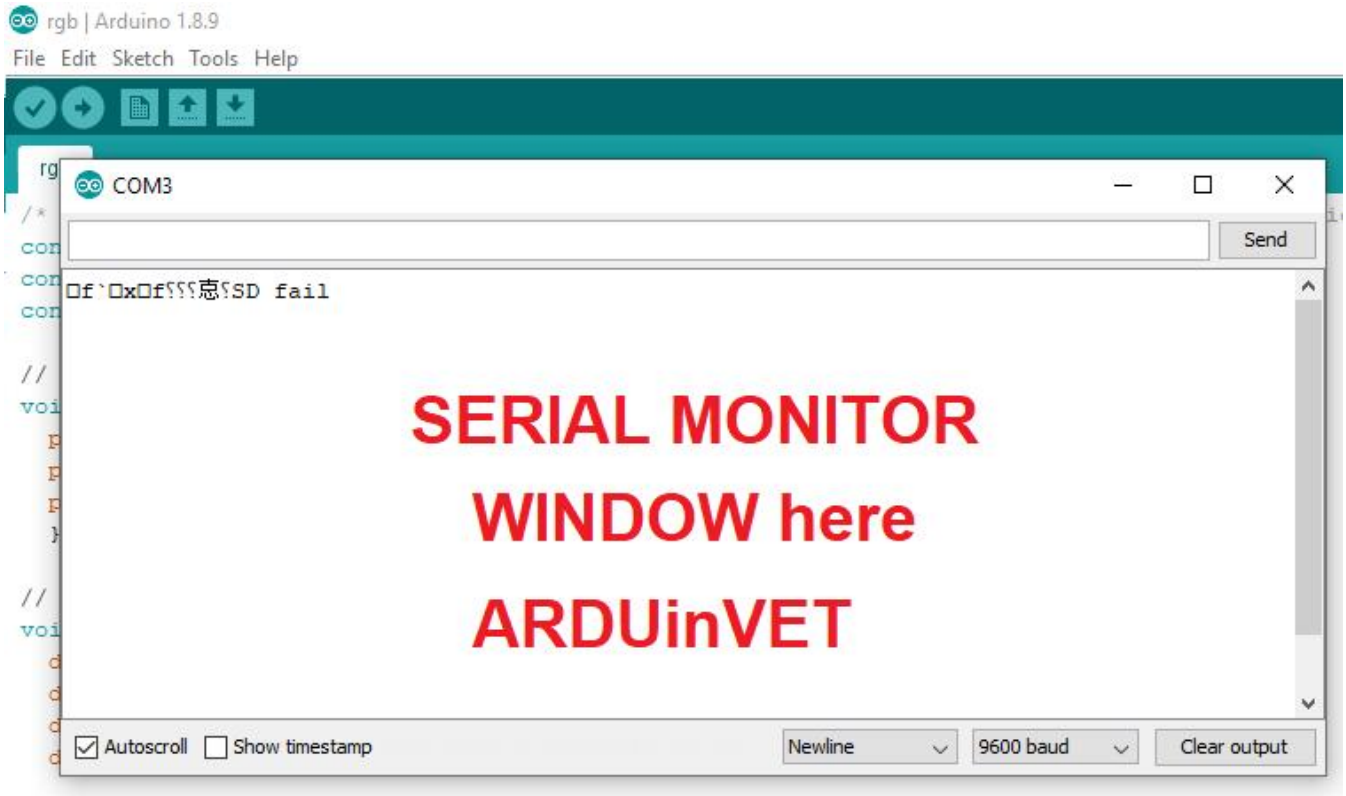
The Serial Monitor este o fereastră pop-up separată care acționează ca un terminal separat care comunică prin primirea și trimiterea datelor seriale. Vedeți pictograma din extrema dreaptă a imaginii aici 

.

Datele seriale sunt trimise printr-un singur fir (dar de obicei se deplasează prin USB în cazul nostru) și constă dintr-o serie de 1 și 0 trimise prin fir. Datele pot fi trimise în ambele direcții (în cazul nostru pe două fire).

Veți utiliza Serial Monitor pentru a depana Arduino Software Sketches sau pentru a vizualiza datele trimise de un program (sketch) funcțional. Trebuie să aveți un Arduino conectat prin USB la computer pentru a putea activa monitorul serial.

Deschideți Serial Monitor făcând clic pe caseta Serial Monitor din IDE. Ar trebui să arate ca imaginea de mai jos. Asigurați-vă că baud-ul (viteza) este setat la 9600. Este situat în colțul din dreapta jos. (Important este că este setat la fel în programul nostru și aici. Deoarece valoarea implicită aici este 9600, noi le setăm în program)



Comenzile principale de utilizare pentru Serial Monitor

Serial.begin(); Setează rata de date în biți pe secundă (baud) pentru transmisia de date în serie. Pentru a comunica cu Serial Monitor, asigurați-vă că utilizați una dintre ratele de transmisie enumerate în meniul din colțul din dreapta jos al ecranului său. Cu toate acestea, puteți specifica alte rate - de exemplu, pentru a comunica prin pinii 0 și 1 cu o componentă care necesită o anumită rată de transmisie.

Sintaxa: Serial.begin(speed);

Exemplu: Serial.begin(9600);

Serial.print(); Tipărește date pe portul serial ca text ASCII care poate fi citit de om. Această comandă poate lua mai multe forme. Numerele sunt tipărite utilizând un caracter ASCII pentru fiecare cifră. Flotările sunt tipărite în mod similar cu cifre ASCII, implicit la două zecimale. Octeții sunt trimiși ca un singur caracter. Caracterele și șirurile sunt trimise ca atare. De exemplu:

Serial.print(78;) gives "78"

Serial.print('N'); gives "N"

Serial.print("Hello ArduinVet."); gives "Hello ArduinVet."

Serial.println(); Tipărește datele în portul serial ca text ASCII citibil de către om, urmat de un caracter de returnare a caracterului (ASCII 13 sau „\ r") și un caracter de linie nouă (ASCII 10 sau „\ n"). Această comandă ia aceeași formă ca Serial.print().

Serial.println(val);

Serial.println(val, format);

Serial.println(13, BIN); dă "1101", valoarea în binar a numărului 13 pe un Serial Monitor.

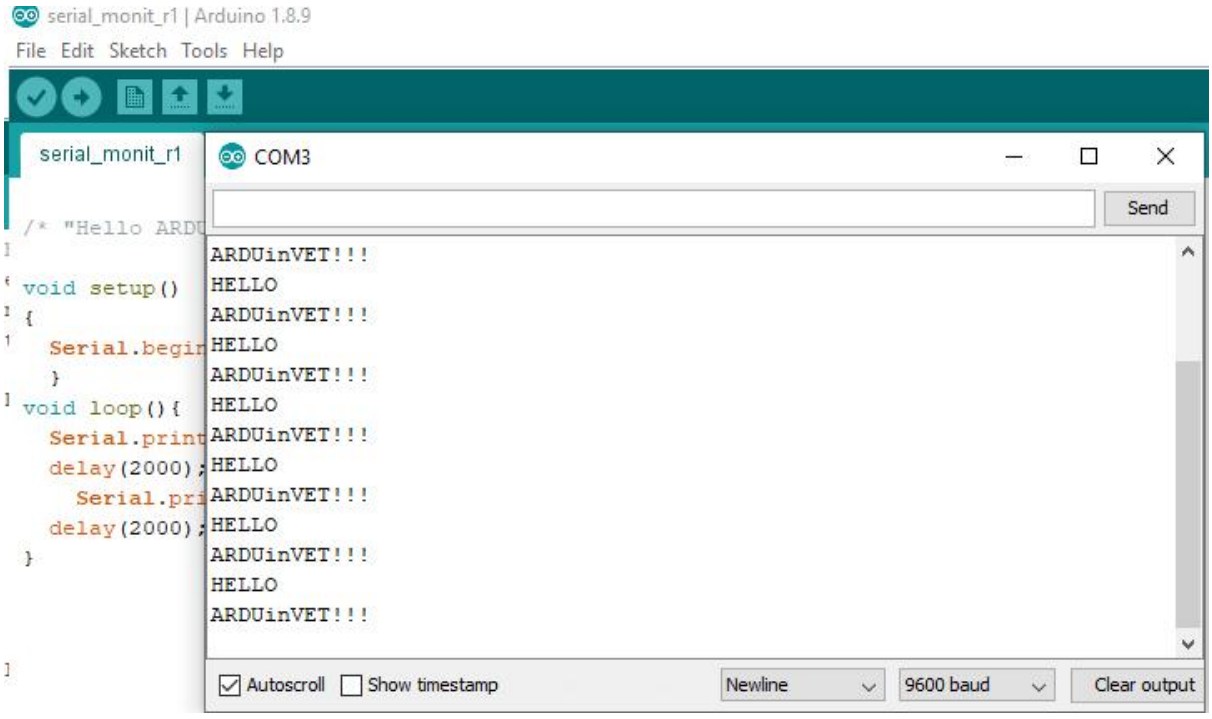
NOTĂ: Singura diferență între Serial.print și Serial.println este că Serial.println înseamnă că următorul lucru trimis portului serial după acesta va începe pe următoarea linie. Este posibil să fi observat un al treilea lucru nou. Există ceva între ghilimele („). Aceasta se numește șir.

Circuitul 9:

Titlul circuitului: "Hello ARDUinVET" aplicat pe Serial Monitor”

Descrierea circuitului: ”Hello ARDUinVET” se va vedea pe Serial Monitor.

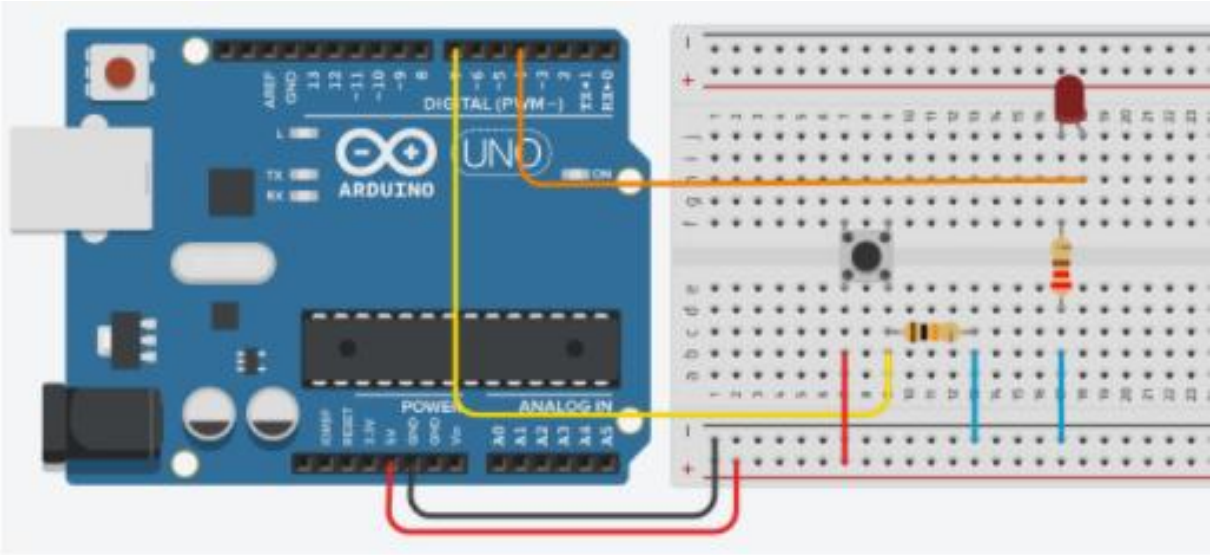
```
/*       "Hello ARDUinVET" aplicat pe Serial Monitor  */
void setup()
{
  Serial.begin(9600);     // viteza de comunicare serială
}
void loop()
{
  Serial.println("  HELLO ");
  delay(2000);
    Serial.println("ARDUinVET!!!");
  delay(2000);
}
```



Circuitul 10:

Titlul circuitului: " Analiza stării butoanelor pe Serial Monitor"

Descrierea circuitului: Dacă butonul este apăsat LED -ul luminează și apare mesajul "Led is ligthing, LED=ON" pe Serial Monitor. Dacă butonul nu este apăsat LED -ul luminează și apare mesajul "Button is not pressed" pe Serial Monitor.




```
/*      Analiza stării butonului pe Serial Monitor   */
void setup()                  {
 pinMode(4, OUTPUT);
 pinMode(7, INPUT);
   Serial.begin(9600);        }
void loop()                    {
 if (digitalRead(7) == HIGH)        // citește semnalul du=igital din pinul 7
   {
   digitalWrite(4, HIGH);
   Serial.println("Led is ligthing, LED=ON");
   delay(250);
   }
  else                        // dacă condiția anterioară nu este îndeplinită
   {
   digitalWrite(4, LOW);
```

```
  Serial.println("Button is not pressed");
  delay(1000);
  }
}                                   //pe Serial Monito apare textul de mai jos
```

```
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
```

Circuitul 11:

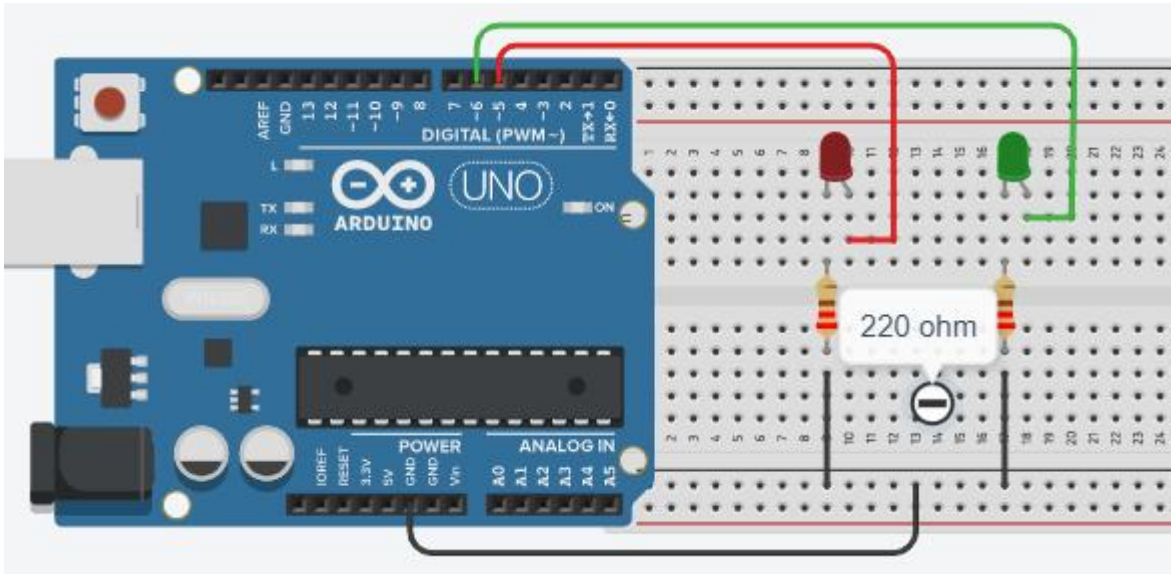
Titlul circuitului: " Trimiterea datelor de la Serial Monitor."

Descrierea circuitului:

Dacă caracterul "A" de pe tastatură este apăsat LED 1 este aprins

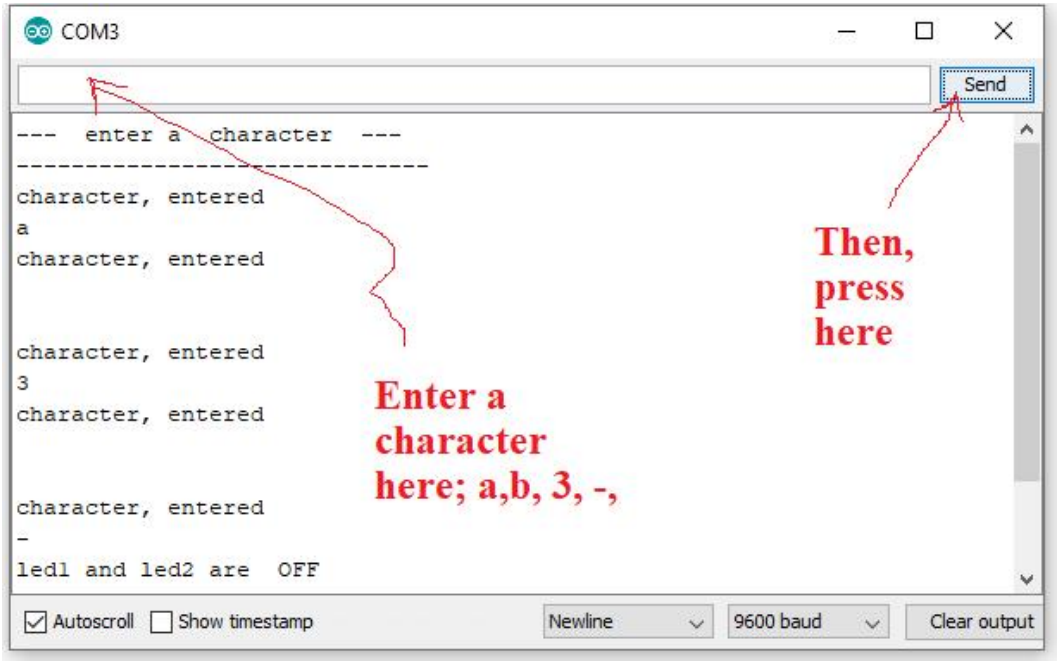
Dacă caracterul "3" de pe tastatură este apăsat LED 2 este aprins

Dacă caracterul "-" de pe tastatură este apăsat LED 1 și LED 2 sunt stinse




```
/*     Trimiterea datelor de la Serial Monitor  */
char character;
#define  led1 5
#define  led2 6
 void setup()   {
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
Serial.begin(9600);
Serial.println ("---  enter a  character  --- ");
Serial.println ("----------------------------
");                                                    }
void loop()
{                                                                  "
```

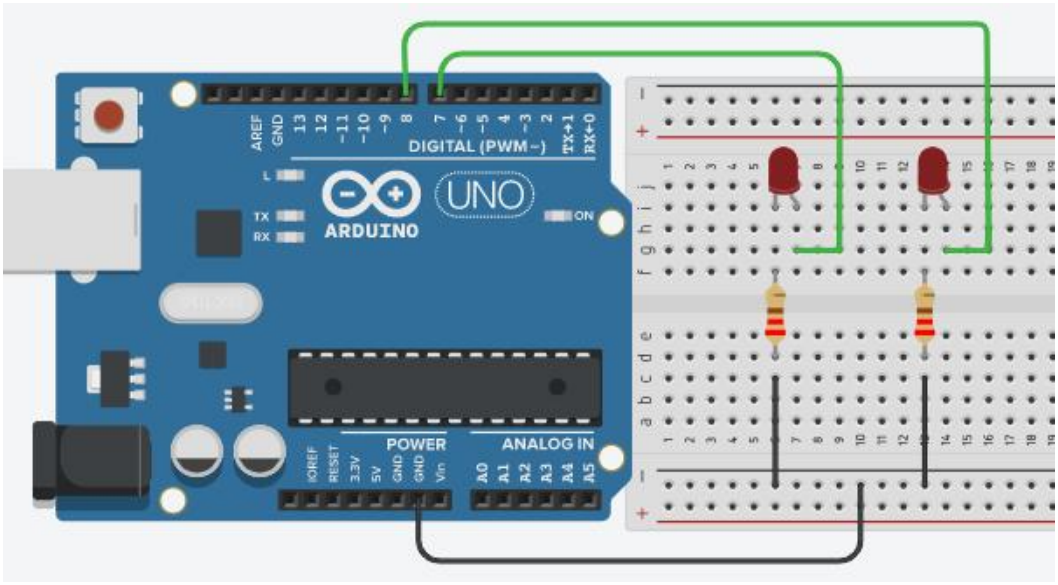
```
    if (Serial.available()>0)
{
character=Serial.read();
Serial.println ("character, entered ");
Serial.println (character);
if (character=='A') digitalWrite (led1, 1);
if (character=='a') digitalWrite (led1, 1);
if (character=='3') digitalWrite (led2, 1);
if (character=='-')
{
digitalWrite(led1,0);
digitalWrite(led2,0);
Serial.println ("led1 and led2 are  OFF
");                                                    }
                                                                  }
                                                                        }
```



Circuitul 12:

Titlul circuitului: "Învățarea comenzii FOR"

Descrierea circuitului:



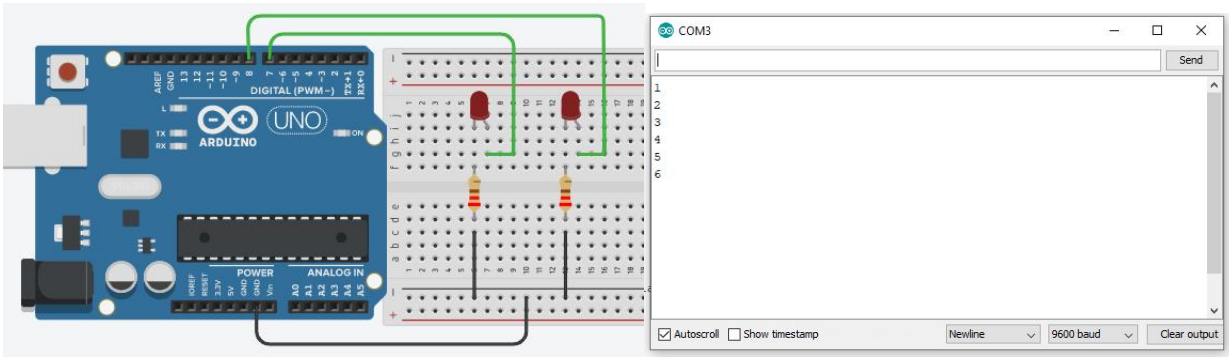

```
/*    "Învățarea comenzii FOR"   */
int   led1 = 7;
int   led2 = 8;
void setup() {
Serial.begin(9600);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);
                }
void loop()
{
 for(int i=1; i<6; i++)
{
Serial.println(i);
digitalWrite(led1, 1);
digitalWrite(led2, 1);
delay(1000);
digitalWrite(led1, 0);
digitalWrite(led2, 0);
delay(1000);
        }
           }
```

Circuitul 13:

Titlul circuitului: "Comanda FOR versus Serial Monitor"

Descrierea circuitului: În această aplicație, LED-ul clipește de 6 ori. Numerele 1, 2, 3, 4, 5, 6 vor apărea pe Serial Monitor. Apoi va fi introdus în bucla while ieșind din bucla for. Și programul se va opri aici. Pentru a reporni, trebuie apăsat butonul de resetare.

Aici, folosim același circuit ca cel anterior



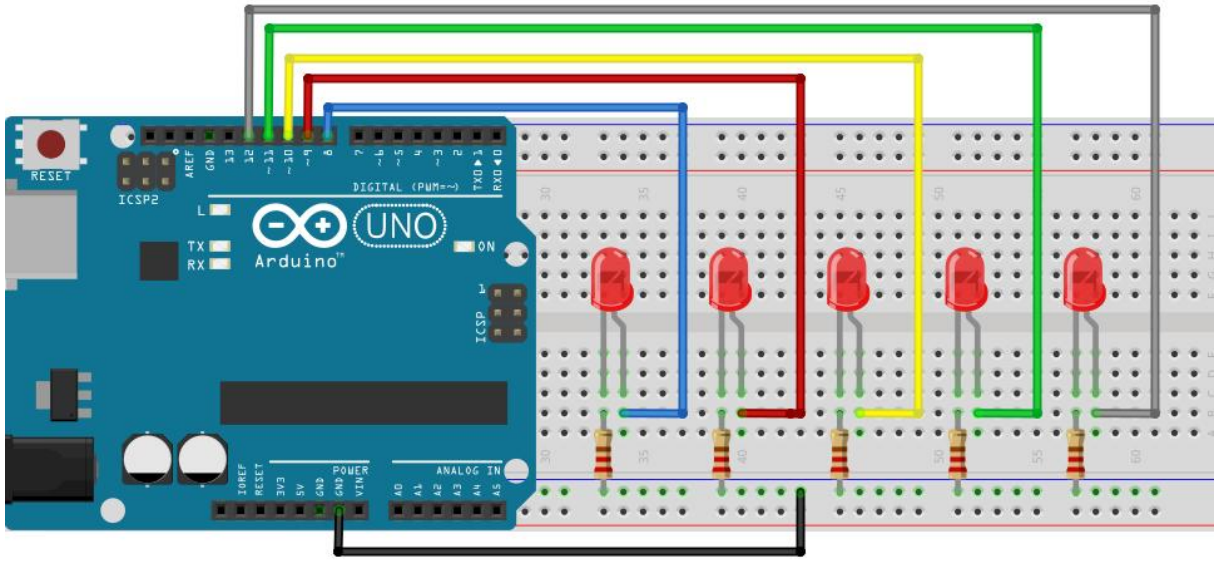

```
/*     "Comanda FOR versus Serial Monitor"   */

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++)               // valoarile lui i = 1, 2,3, 4,5, 6
  {
Serial.println(i);                      // scrie valoarea lui i pe Serial Monitor
digitalWrite(led1, 1);
digitalWrite(led2, 1);
delay(1000);
digitalWrite(led1, 0);
digitalWrite(led2, 0);
delay(1000);
      }
while(1);                   //  bucla for nu intră într-o buclă infinită
}
```

NOTĂ: Pentru restart se apasă butonul Reset

Circuitul 14:

Titlul circuitului: "Knight Rider cu 5 LED - uri folosind comanda FOR"

Descrierea circuitului:




```
/*     Knight Rider cu 5 LED - uri folosind comanda 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++)                    // numărătorul în sus începe aici
  {
digitalWrite(b, HIGH);
delay(150);
digitalWrite (b, LOW);
delay(150);
  }
  for (int b=12; b>=8; b--)                   // numărătoarea în jos începe aici
{
digitalWrite (b, HIGH);
delay(150);
digitalWrite (b, LOW);
delay(150);
  }
                }
```

Circuitul 15:

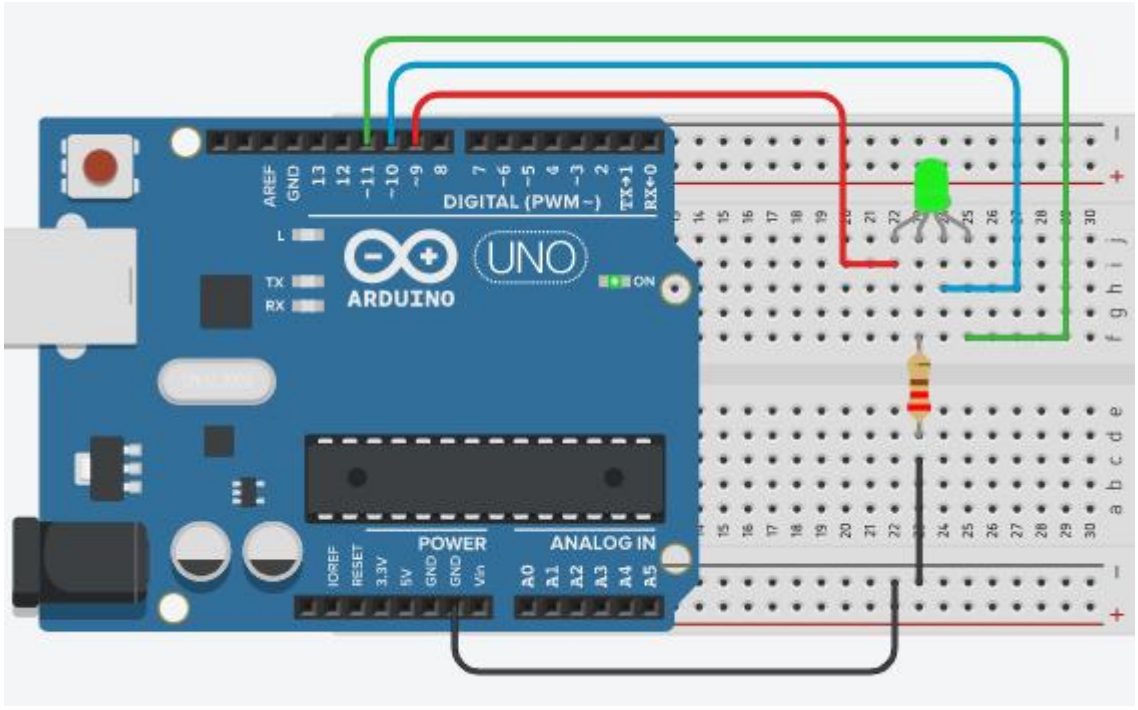
Titlul circuitului: "Producrea aleatoare a culorilor cu LED RGB, folosind comanda Switch/Case"

Descrierea circuitului: Aleator comanda Switch/Case este folosită în această aplicație. În acest circuit aleaotor se vor obține culori cu roșu, verde și albastru.

NOTĂ:

random() : Funcția random aleator un număr.
Sintaxa: random(max), random(min, max)

min: valoarea aleatoare cea mai mică, (opțional).
max: valoarea aleatoare cea mai mare



/* Producerea aleatoare a culorilor cu LED RGB, folosind comanda Switch/Case */

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

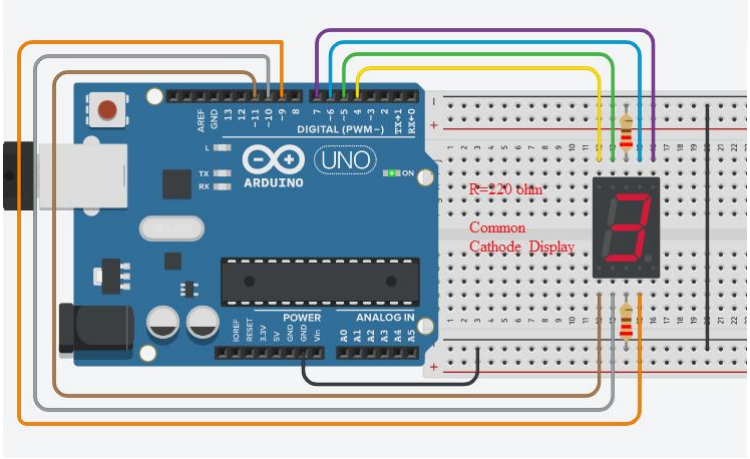
```
void setup() {

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

void loop()
{

colour=random(4);
Serial.println(colour);

switch(colour)  {

 case 0:
digitalWrite (R, 1);              //LED -ul roșu aprins
digitalWrite (G, 0);
digitalWrite (B, 0);

delay(dly);
break;

 case 1:
digitalWrite (R, 0);            //LED -ul verde aprins
digitalWrite (G, 1);
digitalWrite (B, 0);
delay(dly);
break;

 case 2:                          //LED -ul albastru aprins
digitalWrite (R, 0);
digitalWrite (G, 0);
digitalWrite (B, 1);
delay(dly);
break;

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

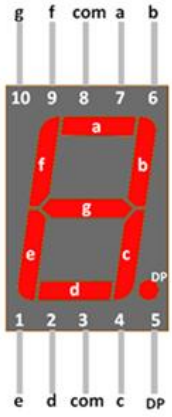
Circuitul 16:

Titlul circuitului: "Numărător de la 0 la 9 cu display cu 7 segmente cu anod comun"

Descrierea circuitului: Pentru a vedea un număr pe afișaj, LED-ul corespunzător este aprins din cele 7 LED-uri, reprezentate de literele a, b, c, d, e, f, g.

NOTă: Un afișaj cu șapte segmente este o formă de dispozitiv de afișare electronică pentru afișarea numerelor zecimale.






```
/*    "Numărător de la 0 la 9 cu display cu 7 segmente cu anod comun"    */
int      a=6,  b=7,  c=9,  d=10,    e=11,  f=5,  g=4;
int  number;
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(number=0;  number< =9;  number++)
{                                          delay(1000);
switch(number)  {
 case 0:                                   digitalWrite (f, HIGH);
digitalWrite (a, HIGH);                    digitalWrite (g, LOW);
digitalWrite (b, HIGH);                    break;
digitalWrite (c, HIGH);                    case 1:
digitalWrite (d, HIGH);                    digitalWrite (a, LOW);
digitalWrite (e, HIGH);                    digitalWrite (b, HIGH);
```

```
digitalWrite (c, HIGH);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
break;
case 2:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, LOW);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, LOW);
digitalWrite (g, HIGH);
break;
case 3:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, HIGH);
break;
case 4:
digitalWrite (a,LOW );
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;
case 5:
digitalWrite (a, HIGH);
digitalWrite (b, LOW);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, LOW);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;
case 6:
digitalWrite (a, HIGH);
digitalWrite (b, LOW);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;
case 7:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
break;
case 8:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;
case 9:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, LOW);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;

                }

                        }

                                }
```

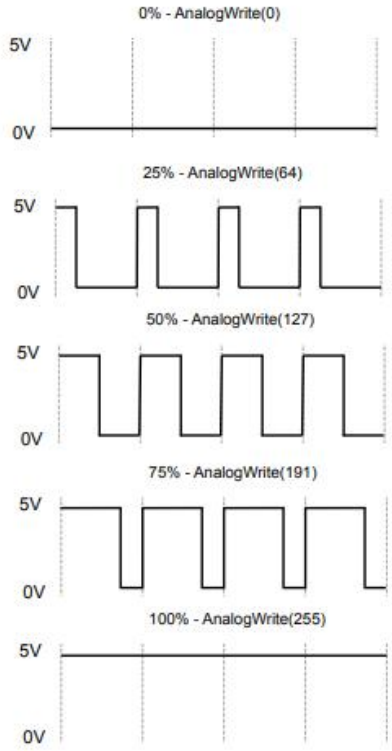
Circuitul 17:

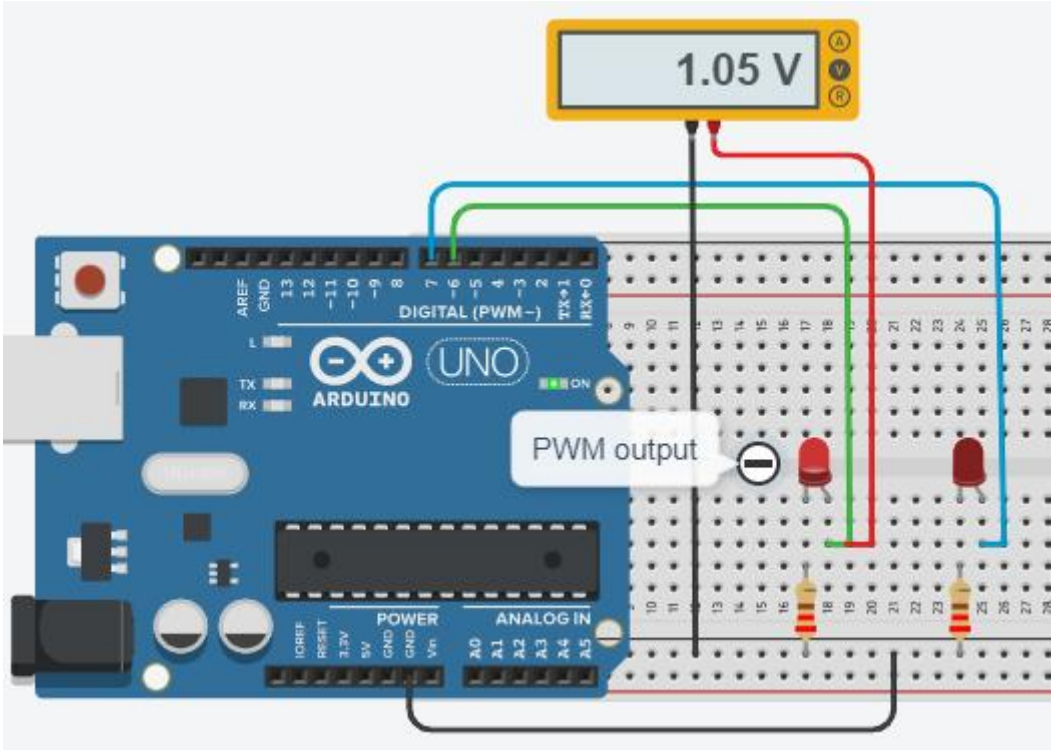
Titlul circuitului: " Utilizarea tehnicii PWN"

Descrierea circuitului: În practică, sunt utilizate pin-6 ca ieșire PWM și pin-7 ca ieșire digitală. Astfel, se urmărește observarea diferenței dintre ele. Aplicații precum reglarea luminozității cu LED-uri, controlul vitezei motorului pot fi realizate cu metoda PWM.

NOTĂ: Arduino acceptă PWM (pe anumiți pini marcați cu o tildă (~) pe placa dvs. Arduino - pini 3, 4,5,9,10 și 11) la 500Hz. (De 500 de ori pe secundă.) Puteți da o valoare între 0 și 255. 0 înseamnă că nu este niciodată 5V. 255 înseamnă că este întotdeauna 5V. Pentru a face acest lucru, faceți un apel către analogWrite () cu valoarea. Raportul dintre timpul „ON" și timpul total se numește „ciclu de funcționare". Se spune că o ieșire PWM care este pornită la jumătate din timp are un ciclu de funcționare de 50%.

Vă puteți gândi la PWM ca fiind activat pentru x / 255 unde x este valoarea pe care o trimiteți cu analogWrite (). Mai jos este un exemplu care arată cum arată impulsurile:





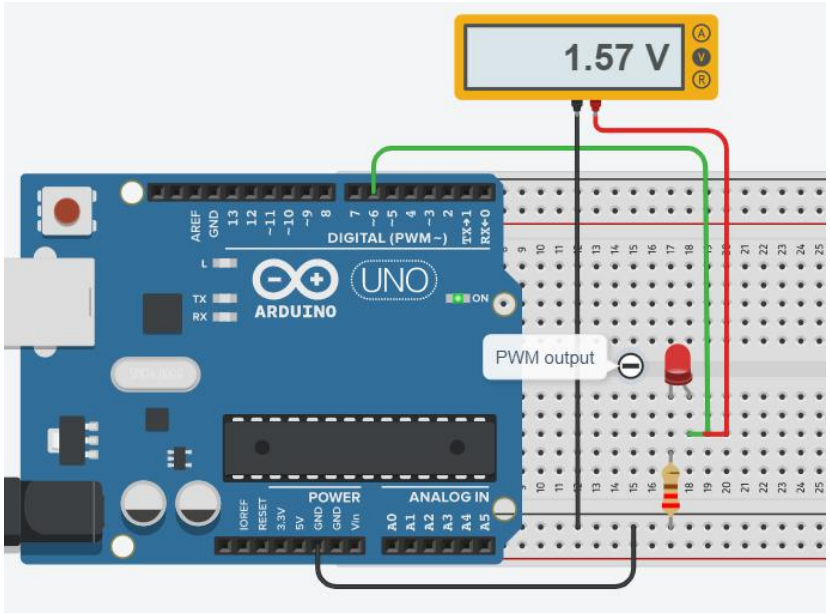
/* Learnning the PWN Technique by using analogWrite() */

```
#define   led1  6
#define   led2  7

void setup()  {

pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);      }

void loop()                 {

analogWrite(led1, 60);        //  valoarea de 60 este trimisă la LED1 pin, i.e 1.05 V.

analogWrite(led2,60);         //  valoarea de 60 este trimisă la LED2 pin, i.e 1.05 V.

delay (100);                  // doar LED1 luminează

.                                                    }
```

Circuitul 18:

Titlul circuitului: " Schimbarea luminozității unui LED de la minim la maxim."

Descrierea circuitului: Utilizând tehnica PWM, poate fi modificată luminozitatea LED-ului de la minim la maxim. Aici, **analogWrite ()** și comanda **for** vor fi folosite împreună.



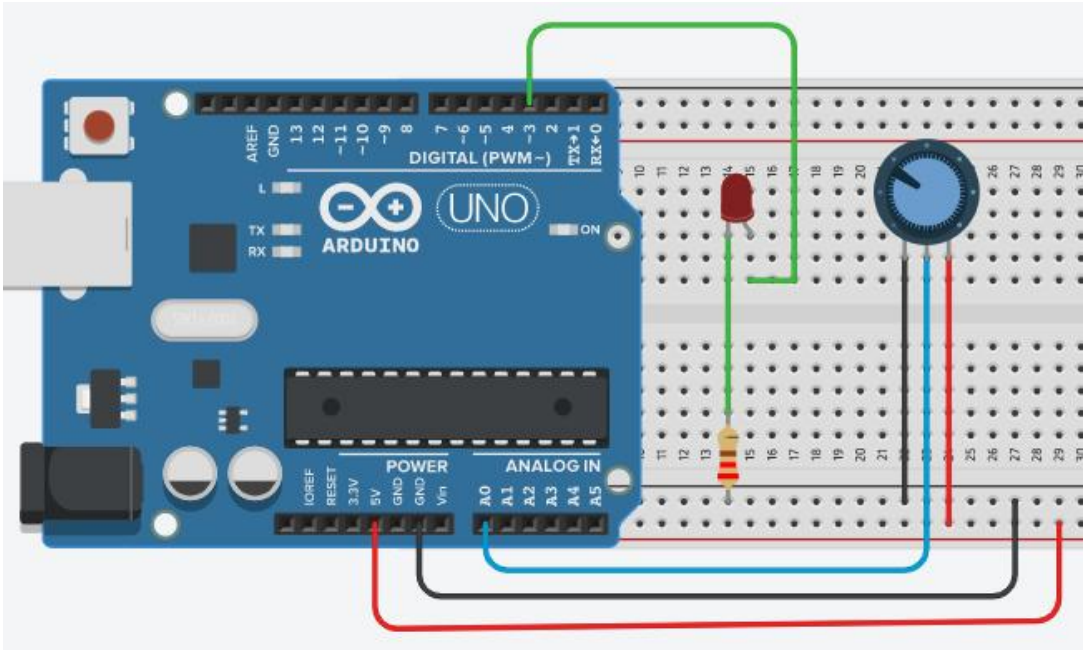

```
/* Schimbarea luminozității unui LED de la minim la maxim.  */
#define led1   6
int a;
void setup()        {
Serial.begin(9600);
pinMode(led1, OUTPUT);   }
void loop() {
for (a=0; a<=255; a++) {
Serial.println(a);
analogWrite(led1, a);
delay (100);
          }            }
```

NOTĂ: Valori cuprinse între 0 și 5 volți sunt afișate pe voltmetru. Numerel de la 0 la 255 sunt afișate pe monitorul serial.

Circuiult 19:

Titlul circuitului: "Potențiometrul care reglează luminozitatea LED-ului"

Circuit Explanation: Folosind un potențiometru de (10K), se poate regla strălucirea LED-ului de la minim la maxim. Aici, se folosește functia map().



Notă:

Funcția map():

Re-mapează un număr de la o gamă la alta. Adică, o valoare din **fromLow** ar fi mapată la **toLow**, o valoare from **fromHigh** to **toHigh**, valori intermediare la valori intermediare etc.

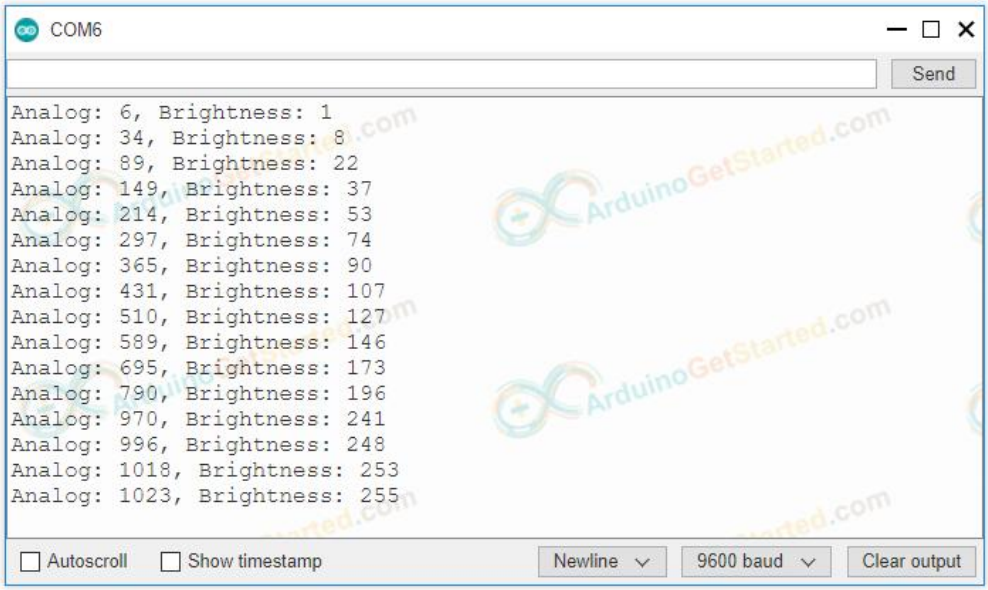
Nu constrânge valorile în interval, deoarece valorile în afara intervalului sunt uneori intenționate și utile. Funcția **constrain()** poate fi utilizată înainte sau după această funcție, dacă se doresc limite la intervale.

Funcția **map ()** folosește matematică întegrată, deci nu va genera fracții, atunci când matematica ar putea indica faptul că ar trebui să o facă. Resturile fracționare sunt trunchiate și nu sunt rotunjite sau mediate.

Syntaxa: map(value, fromLow, fromHigh, toLow, toHigh);

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

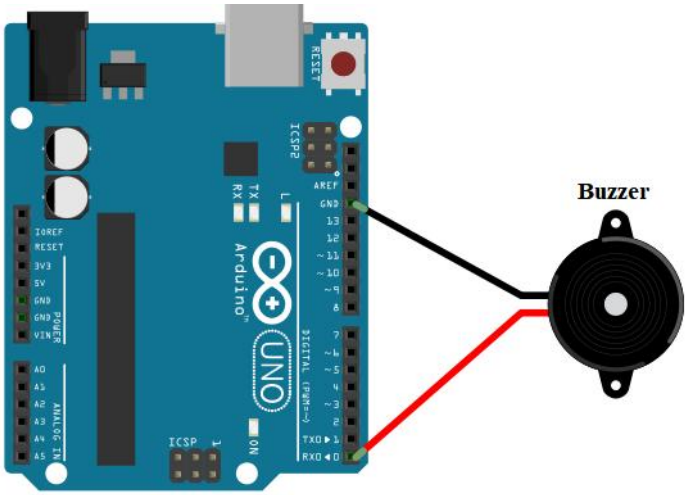
```
/*     Potentiometrul care reglează luminozitatea LED-ului    */
int LED_PIN = 3;
void setup() {
Serial.begin(9600);
pinMode(LED_PIN, OUTPUT);      }
void loop() {
  // citește intrările de la pinul analog A0 (valori între 0 și 1023)
 int analogValue = analogRead(A0);
  // modifică strălucirea (valori între 0 și 255)
 int brightness = map(analogValue, 0, 1023, 0, 255);
  // modifică strălucirea LED- ului conectat la pinul 3
 analogWrite(LED_PIN, brightness);
  // afișează valoarea
  Serial.print("Analog: ");
  Serial.print(analogValue);
  Serial.print(" Brightness: ");
  Serial.println(brightness);
 delay(100);
// Serial Monitor afișează ca mai jos
                                        }
```



Circuitul 20:

Titlul circuitului: "Folosirea unui buzzer"

Descrierea circuitului: Un buzzer este un dispozitiv de semnal audio, care poate fi mecanic, electromecanic sau piezoelectric (piezo pe scurt). Utilizările tipice ale buzzerelor în industrie sunt ca dispozitive de alarmă, care fac un zgomot sau un bip în timp ce au nevoie de buzzing.

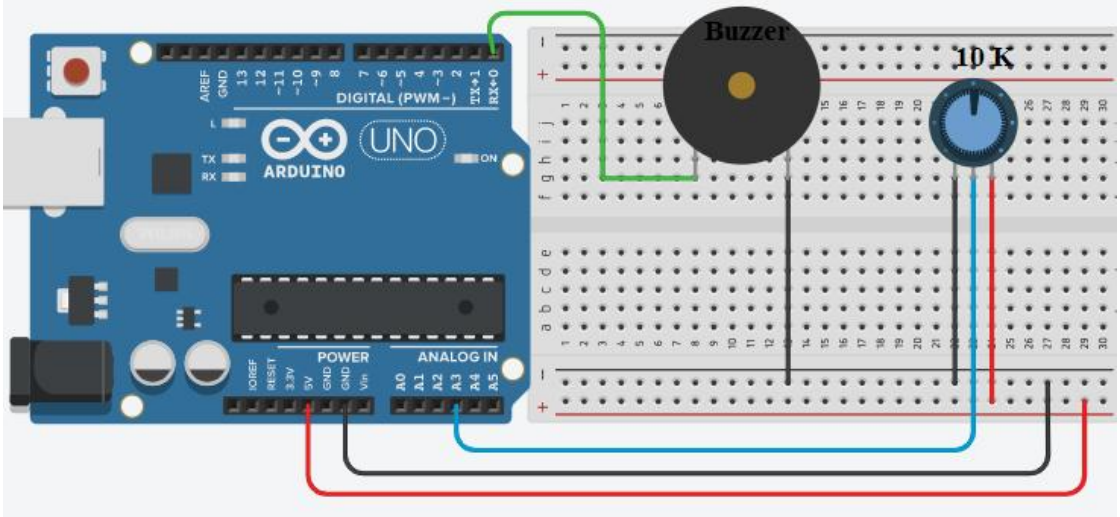



```
/* Folosirea unui buzzer*/
#define buzzer_pin  0
 void setup()  {
 pinMode(buzzer_pin, OUTPUT);        }
void loop()       {
 digitalWrite(buzzer_pin, HIGH);
 delay(500);
 digitalWrite(buzzer_pin, LOW);
 delay(500);     }
```

Circuitul 21:

Titlul circuitului: "Controlarea unui buzzer cu Potențiometrul"

Descrierea circuitului: Când valoarea potențiometrului este mai mare de 500, buzzerul începe să sune




```
/* To control a  buzzer with Potentiometer  */

const int POT_PIN = A3;              // Pin - ul Arduino conecta la pinul Potențiometrului
const int BUZZER_PIN  = 0;           // Pin - ul Arduino conectat la pinul Buzzerului
const int ANALOG_THRESHOLD  = 500;
int analogValue;

void setup() {

  pinMode(BUZZER_PIN, OUTPUT);      // setarea pin - ului ARDUINO în modul Ieșire

}

void loop() {

 analogValue = analogRead(POT_PIN);    //citește semnalul de intrare de la pin-ul analog

if(analogValue > ANALOG_THRESHOLD)
digitalWrite(BUZZER_PIN, HIGH);                  // Buzzer - ul sună

else
digitalWrite(BUZZER_PIN, LOW);                   // Buzzer - ul se oprește

}
```

Erasmus+ KA-202

Proiecte de parteneriate strategice pentru educație și formare profesională

Titlul proiectului: "Predarea și învățarea ARDUINO în formarea profesională"

Acronimul proiectului: " ARDUinVET "

Numărul proiectului: "2020-1-TR01-KA202-093762"

LCD

Modul și Kit de instruire

LCD Modul și Kit de instruire

În acest modul, dorim să explicăm cum să afișați mesajele de stare sau citirile senzorilor Arduino pe ecranele LCD. Sunt extrem de comune și o modalitate rapidă de a adăuga o interfață lizibilă la proiectul dvs.

Un LCD este prescurtarea de la Liquid Crystal Display. Este o unitate de afișare care utilizează cristale lichide pentru a produce o imagine vizibilă. Când se aplică curent acestui tip special de cristal, acesta devine opac blocând lumina de fundal care trăiește în spatele ecranului. Ca rezultat, acea zonă va deveni întunecată în comparație cu altele. Și așa sunt afișate caracterele pe ecran.

Modul cu LCD cu interfață 16×2 caractere cu Arduino

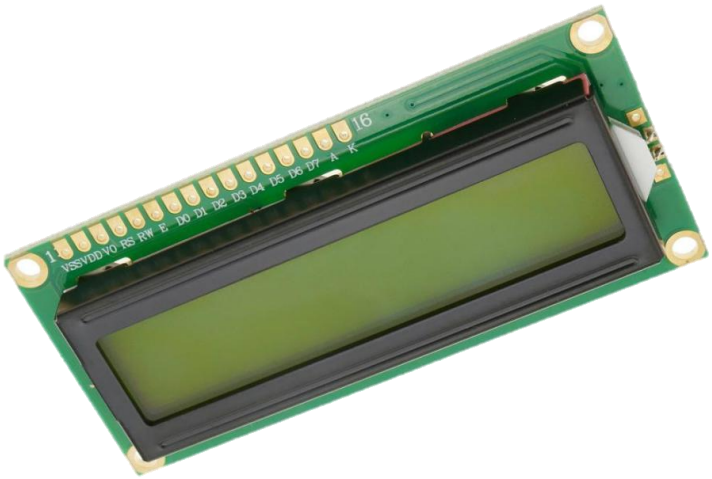
Există diferite tipuri de afișaje LCD pe care le puteți conecta la Arduino. Cel mai comun se bazează pe un cip de controler LCD cu interfață paralelă de la Hitachi numit HD44780.

Aceste ecrane LCD sunt ideale pentru afișarea doar a textului/caracterelor, de unde și numele „LCD caractere”. Ecranul are o lumină de fundal LED și poate afișa 32 de caractere ASCII pe două rânduri cu 16 caractere pe fiecare rând.

Există câteva dreptunghiuri pentru fiecare caracter de pe afișaj, fiecare dintre aceste dreptunghiuri este o grilă de 5×8 pixeli.

Deși afișează doar text, vin în multe dimensiuni și culori: de exemplu, 16×1, 16×4, 20×4, cu text alb pe fundal albastru, cu text negru pe verde și multe altele.

Comunitatea Arduino a dezvoltat deja o bibliotecă pentru a gestiona LCD-urile HD44780 (LiquidCrystal Library); așa că le vom avea interfațate în puțin timp.



Mai jos sunt pinii de ieșire a LCD:

- **VSS:** este pinul de masă și trebuie conectata la masa plăcii Arduino;
- **VDD:** conectat 5 V;
- **VD:** pentru ajustarea contrastului (connectat la pinul central al potențiometrului) ;
- **RS:** controlează în care zonă de memorie a LCD sunt stocate datele trimise;
- **R/W:** pin pentru selectarea modului citit/ scris;
- **E:** dacă este activat, permite modulului LCD să efectueze instrucțiuni speciale;
- **D0 to D7:** transmitere de date;
- **A and K:** anod și catod pentru a oferi iluminare de fundal la modulul LCD.

Arduino - LCD funcții (comenzi)
LiquidCrystal lcd() - Creează o variabilă de tip LiquidCrystal. Afișajul poate fi controlat folosind 4 sau 8 linii de date. Dacă este primul, omiteți numerele de pin de la d0 la d3 și lăsați acele linii neconectate. Pinul RW poate fi legat la masă în loc să fie conectat la un pin de pe Arduino; dacă da, omiteți-l din parametrii acestei funcție.

Sintaxă:

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)

Parametrii:

rs: numărul pinului Arduino care este conectat la pinul RS de pe LCD

rw: numărul pinului Arduino care este conectat la pinul RW de pe LCD (opțional)

activare: numărul pinului Arduino care este conectat la pinul de activare de pe LCD

d0, d1, d2, d3, d4, d5, d6, d7: numerele pinilor Arduino care sunt conectați la pinii de date corespunzători de pe LCD. d0, d1, d2 și d3 sunt opționale; dacă este omis, LCD-ul va fi controlat folosind doar cele patru linii de date (d4, d5, d6, d7).

lcd.begin() - Inițializează interfața cu ecranul LCD și specifică dimensiunile (lățimea și înălțimea) afișajului. begin() trebuie apelat înaintea oricărei alte comenzi de bibliotecă LCD.

Sintaxă: lcd.begin(cols, rows)

Parametrii:

lcd: o variabilă de tip LiquidCrystal

cols: numărul de coloane pe care le are afișajul

rows: numărul de linii pe care le are afișajul

lcd.print() - Imprimă text pe LCD.

Syntax:

lcd.print(data)

lcd.print(data, BASE)

Parametrii:

lcd: o variabilă de tip LiquidCrystal

date: datele de imprimat (char, byte, int, long sau string)

BASE (optional): baza în care se imprimă numerele: BIN pentru binar (bază 2), DEC pentru zecimal (bază 10), OCT pentru octal (bază 8), HEX pentru hexazecimal (bază 16).

Întoarce

byte print() va returna numărul de octeți scriși, deși citirea acelui număr este opțională

lcd.setCursor() - Poziționați cursorul LCD; adică setați locația în care va fi afișat textul ulterior scris pe LCD.

Sintaxă:

lcd.setCursor(col, row)

Parametrii:

lcd: o variabilă de tip LiquidCrystal

col: coloana în care să poziționați cursorul (cu 0 fiind prima coloană)

rows: rândul pe care să poziționați cursorul (cu 0 fiind primul rând)

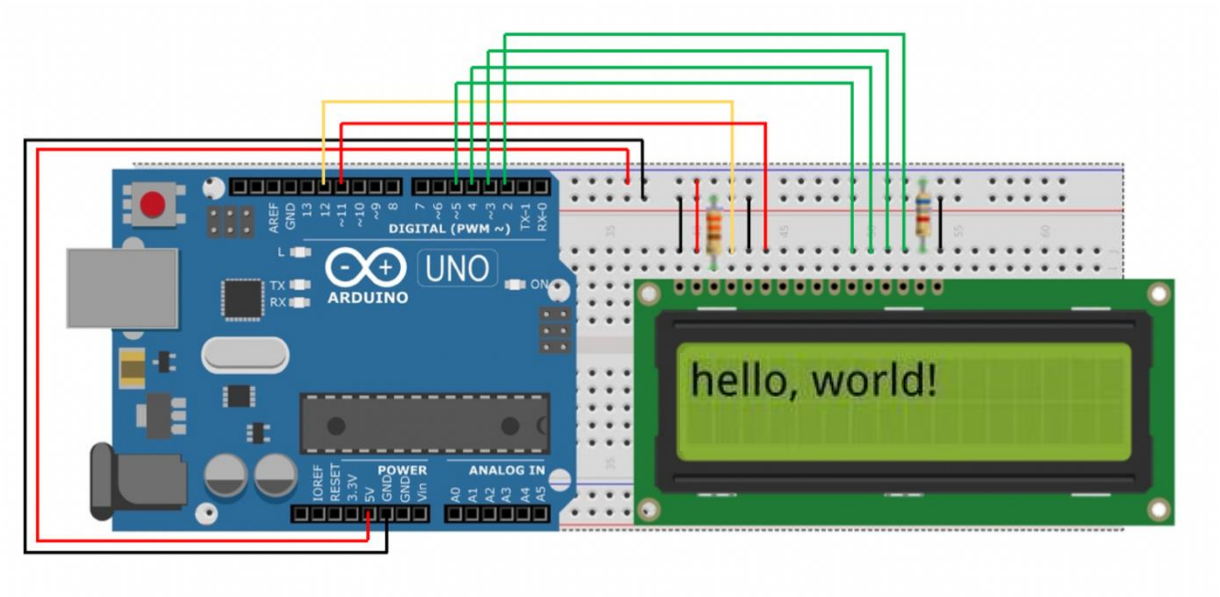
lcd.clear(); - Șterge ecranul LCD și poziționează cursorul în colțul din stânga sus. Sintaxă:

lcd.clear()

Parametrii:

lcd: o variabilă de tip LiquidCrystal

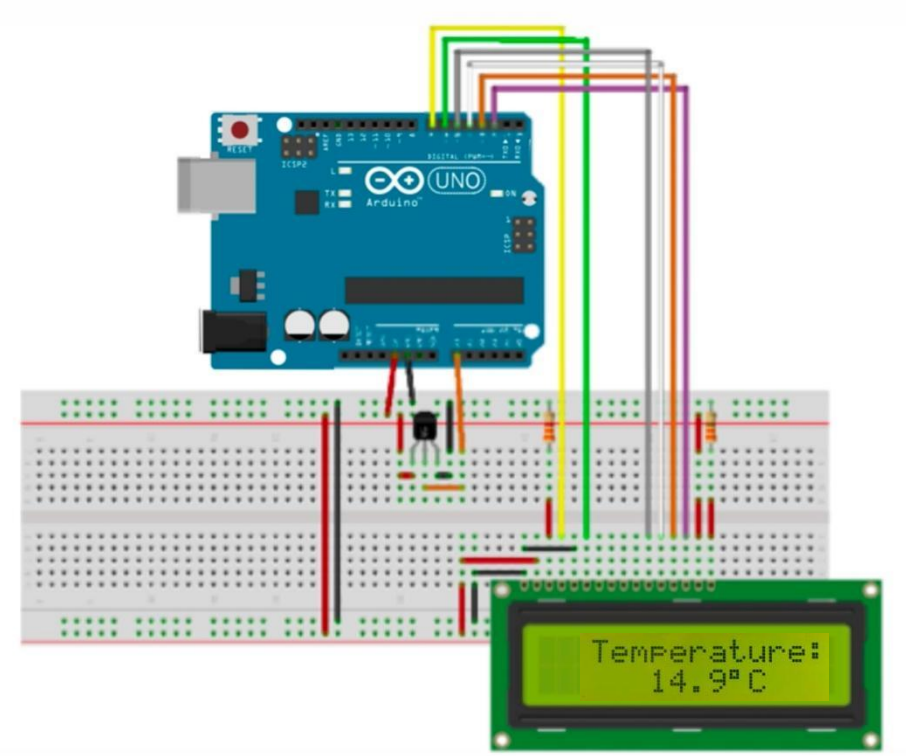
Circuit 1:
Numele circuitului: "Afișarea unui mesaj pe LCD"
Explicarea circuitului: dacă este posibil să conectați un afișaj LCD la microcontroler și să imprimați pe ecran mesajele dorite.
Notă: trebuie să includeți librăria LiquidCrystal.h pentru a folosi funcțiile LCD descrise mai sus.




```
/*  Afișarea unui mesaj  */

#include <LiquidCrystal.h> // include librăria cod

//constantele pentru numărul de rânduri și coloane în LCD

const int numRows = 2;

const int numCols = 16;

// inițializează librăria cu numărul pinilor interfeței

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

void setup()

{

lcd.begin(numCols, numRows);

lcd.print("hello, world!"); // Afișează un mesaj pe LCD.

}
```

Circuit 2:

Numele circuitului: "Termometru digital"
Explicarea circuitului: crează un termometru digital cu Arduino. Temperatura va fi luată cu senzorul de temperatură LM35 și trebuie afișată pe un afișaj LCD și actualizată în fiecare secundă.
Pinul Vo al afișajului LCD reglează contrastul caracterelor afișate pe afișaj. După cum sa menționat mai sus, trebuie să fie conectat la o rezistență de 3,3 kΩ conectată la GND. Cu toate acestea, în unele afișaje, poate fi necesar să conectați pinul Vo direct la GND
Notă: senzorul LM35 are 3 borne: unul pentru alimentare, unul pentru masă și unul pentru ieșirea tensiunii proporționale cu temperatura detectată, care este egală cu 10 mV pentru fiecare grad centigrad, și este calibrat în grade Celsius.



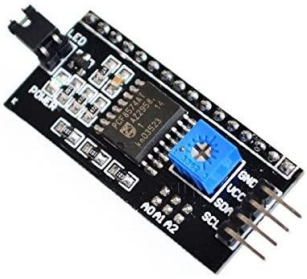
```
/*  Termometru digital  */
#include <LiquidCrystal.h> // Bibliotecă pentru afișajul LCD
#define pin_temp A0 // Vout este pinul de conectare a piciorului la senzorul de temperatură
float temp = 0; // Variabilă în care va fi stocată temperatura detectată
LiquidCrystal lcd(7, 6, 5, 4, 3, 2); // Inițializarea bibliotecii cu pini pentru afișaj LCD
 void setup()
{
  lcd.begin(16, 2); //Setează numărul de rânduri și de coloane ale LCD
  lcd.setCursor(0, 0); //Trece cursorul peste primul rând (row 0) și prima coloană
  lcd.print ("Temperature:"); Afișează mesajul 'Temperature:' în prima linie
   //*Imposed ADC Vref at 1.1V
 (pentru o mai mare acuratețe în calculul temperaturii)
   IMPORTANT: If you use Arduino Mega replace INTERNAL with INTERNAL1V1 */
  analogReference(INTERNAL);
}
void loop()
{
  /*calculează temperatura ==========================================*/
  temp = 0;
   for (int i = 0; i < 5; i++) { // Execută următoarea instrucțiune de 5 ori
     temp += (analogRead(pin_temp) / 9.31); // Calculează temperatura și suma la variabilă 'temp'
      }
  temp /= 5; // Calculează media matematică a valorilor temperaturii
 /*=================================================================*/
  /*I see the temperature on the LCD display
  ==========================================*/
  lcd.setCursor(0, 1); //lcd.print(temp); Mută cursorul peste prima coloană și pe al doilea rând lcd.print(temp);
  // Mold on LCD display temperature
   lcd.print(" C"); // Mold a space and the 'C' font on the display
 /*=========================================*/
  delay(1000); //Întârziere de o secundă (poate fi modificată)
   }
```

Interfața I2C LCD cu Arduino

Dacă doriți să conectați un afișaj LCD cu Arduino, trebuie să consumați o mulțime de pini pe Arduino. Chiar și în modul pe 4 biți, Arduino necesită încă un total de șapte conexiuni, adică jumătate din pinii I/O digitali disponibili.

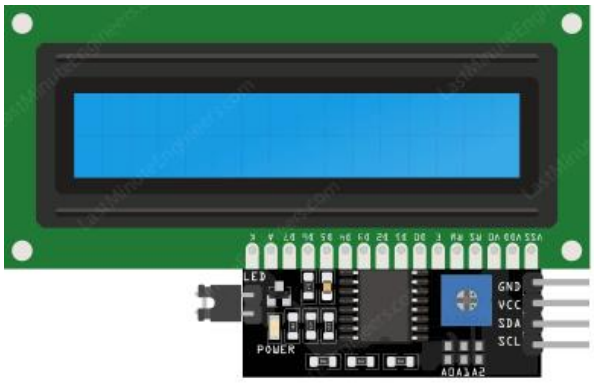
Soluția este utilizarea unui afișaj LCD care interfață cu protocolul I2C. Consumă doar doi pini I/O care nici măcar nu pot face parte dintr-un set de pini I/O digitale și pot fi partajați și cu alte dispozitive I2C.

Cel mai difuz afișaj LCD I2C este format dintr-un afișaj LCD cu caractere HD44780 și un adaptor LCD I2C.



Cea mai importantă parte a adaptorului este cipul I/O Expander de 8 biți – PCF8574. Acest cip convertește datele I2C de la un Arduino în datele paralele cerute de afișajul LCD. Placa vine și cu un potențiometru mic pentru a face ajustări fine la contrastul afișajului. Dacă utilizați mai mult decât un dispozitiv pe aceeași magistrală I2C, poate fi necesar să setați o altă adresă I2C pentru placă, astfel încât să nu intre în conflict cu un alt dispozitiv I2C. Din acest motiv, placa are trei jumperi de lipit (A0, A1 și A2).

Un afișaj I2C LCD are doar 4 pini cu interfața Arduino.



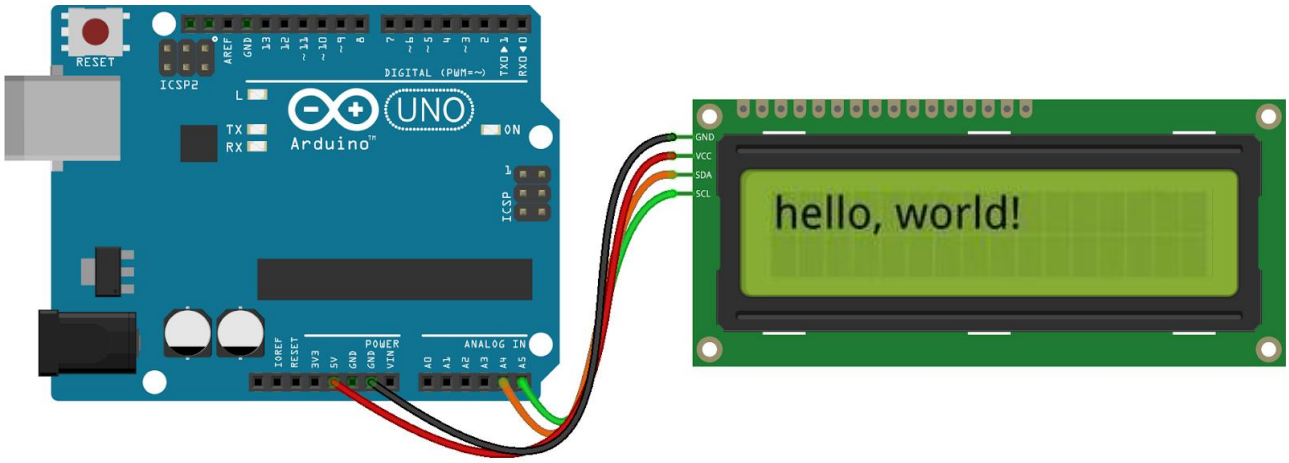
Pinii de ieșire sunt următorii:

- **GND:** este un pin de masă și ar trebui să fie conectat la pământul Arduino;
- **VCC:** furnizează energie modulului și LCD-ului. Conectați-l la ieșirea de 5V a Arduino sau la o sursă de alimentare separată;
- **SDA:** este un pin de date seriale. Această linie este utilizată atât pentru transmitere, cât și pentru recepție. Conectați-vă la pinul SDA de pe Arduino;
- **SCL:** este un pin Serial Clock. Acesta este un semnal de temporizare furnizat de dispozitivul Bus Master. Conectați-vă la pinul SCL de pe Arduino.

Pe plăcile Arduino cu aspectul R3, SDA (linia de date) și SCL (linia de ceas) sunt pe anteturile pinului aproape de pinul AREF. Sunt cunoscute și ca A5 (SCL) și A4 (SDA). Consultați tabelul de mai jos pentru a identifica pinii corecti în funcție de modelul Arduino.

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

Circuit 3:
Numele circuitului: "Afișarea unui mesaj prin I2C LCD"
Explicarea circuitului: este posibil sa conectati un display LCD la microcontroler cu doar 4 pini si sa imprimati pe ecran mesajele dorite.
Notă: trebuie să instalați o bibliotecă numită LiquidCrystal_I2C. Această bibliotecă este o versiune îmbunătățită a bibliotecii LiquidCrystal, care vine împreună cu IDE-ul dvs. Arduino.




```
/*  Afișarea unui mesaj prin I2C Display */
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F,16,2);  // setați adresa LCD la 0x3F pentru un afișaj cu 16 caractere
și 2 linii
void setup() {
  lcd.init();
  lcd.clear();
  lcd.backlight();      // Asigurați-vă că lumina de fundal este aprinsă
  // Afișează un mesaj pe ambele linii ale  LCD.
  lcd.setCursor(2,0);   //Setează cursorul la caracterul 2 linia 0
  lcd.print("Hello world!");
  lcd.setCursor(2,1);   //Mută cursorul pe caracterul 2 linia 1
  lcd.print("with I2C protocol!");
}
void loop() {
}
```

Interfața OLED Graphic Display Module cu Arduino

O altă posibilitate de a utiliza protocolul I2C este alegerea unui afișaj OLED (Organic Light-Emitting Diode). Sunt super ușoare și subțiri și produc o imagine mai luminoasă și mai clară.

Un afișaj OLED funcționează fără lumină de fundal. Acesta este motivul pentru care afișajul are un contrast atât de mare, un unghi de vizualizare extrem de larg și poate afișa niveluri de negru profunde. Absența luminii de fundal reduce semnificativ puterea necesară pentru a rula OLED-ul.

După cum putem vedea din figură, pinout-ul este interfața clasică I2C cu patru pini, văzută deja mai sus.

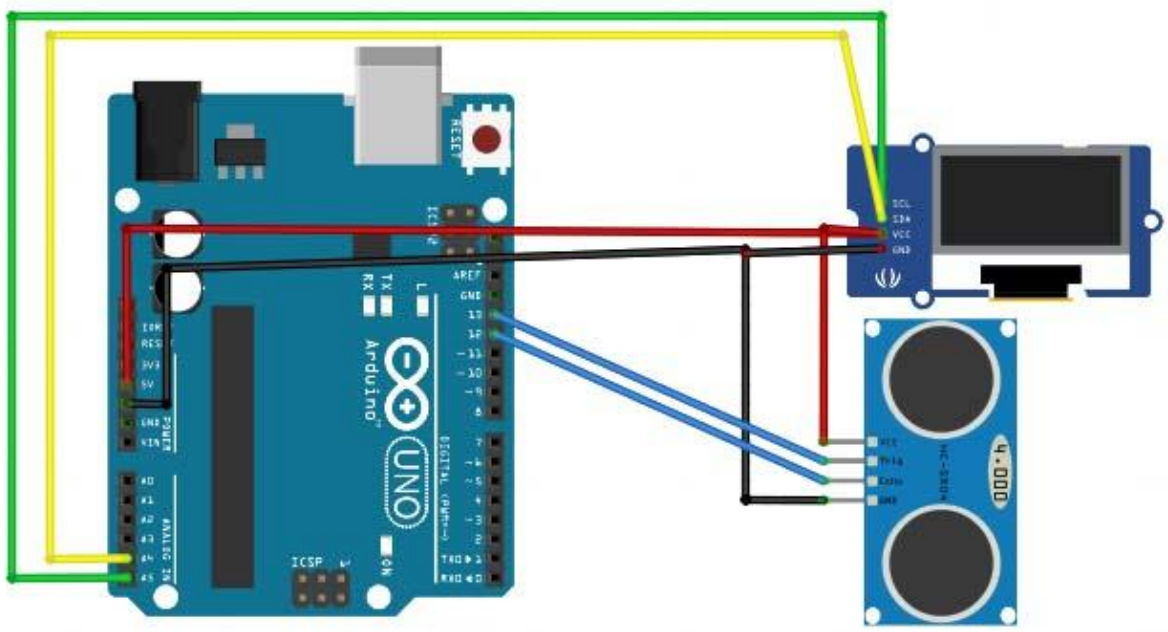
Comunitatea Arduino a dezvoltat deja câteva biblioteci pentru a gestiona aceste afișaje OLED, cum ar fi biblioteca SSD1306 a Adafruit. Pentru a instala biblioteca, navigați la Schița > Includeți biblioteca > Gestionați bibliotecile... Așteptați ca Managerul bibliotecii să descarce indexul bibliotecilor și să actualizeze lista bibliotecilor instalate.

Funcționare Arduino - OLED Graphic Display Module (Comenzi)
pinMode();

display.begin()

display.clearDisplay();

Circuit 4:
Numele circuitului: "Senzor de distanță"
Explicația circuitului: Distanța va fi luată cu senzorul ultrasonic HC-SR04 și va fi afișată pe un afișaj OLED.
Notă: Există doar patru pini de care trebuie să vă faceți griji pe HC-SR04: VCC (Power), Trig (Trigger), Echo (Receive) și GND (Ground).



```
/*  Senzor de distanță  */
#include <SPI.h>        // această bibliotecă vă permite să comunicați cu dispozitivele SPI, cu
                        Arduino ca dispozitiv principal.
#include <Wire.h>      // această bibliotecă vă permite să comunicați cu dispozitive I2C / TWI
#include <Adafruit_GFX.h>            // librăria afișajului OLED
#include <Adafruit_SSD1306.h>
#define CommonSenseMetricSystem
#define trigPin 13       // definește pinii senzorului
#define echoPin 12

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

```
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //initialize with the I2C addr 0x3C (128x64)
 display.clearDisplay();
}
void loop() {
 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;
 display.setCursor(22,20);    // cursor de setare a afișajului oled
 display.setTextSize(3);        //mărimea textului
 display.setTextColor(WHITE);       // dacă scrii negru șterge lucrurile
 display.println(distance);     // tipăriți variabila noastră
 display.setCursor(85,20);
 display.setTextSize(3);
display.println("cm");
 display.display();
 delay(500);
 display.clearDisplay();
  Serial.println(distance);//debug
  }
```

Interfața Nokia 5110 Graphic LCD Display cu Arduino

Puteți interfața Arduino cu mici LCD-uri similare cu cele folosite de Nokia în telefoanele mobile 3310 și 5110. aceste afișaje sunt mici (doar aproximativ 1,5 inchi), ieftine, ușor de utilizat, o putere destul de scăzută (doar 6 până la 7 mA) și pot afișa text, precum și hărți de biți.

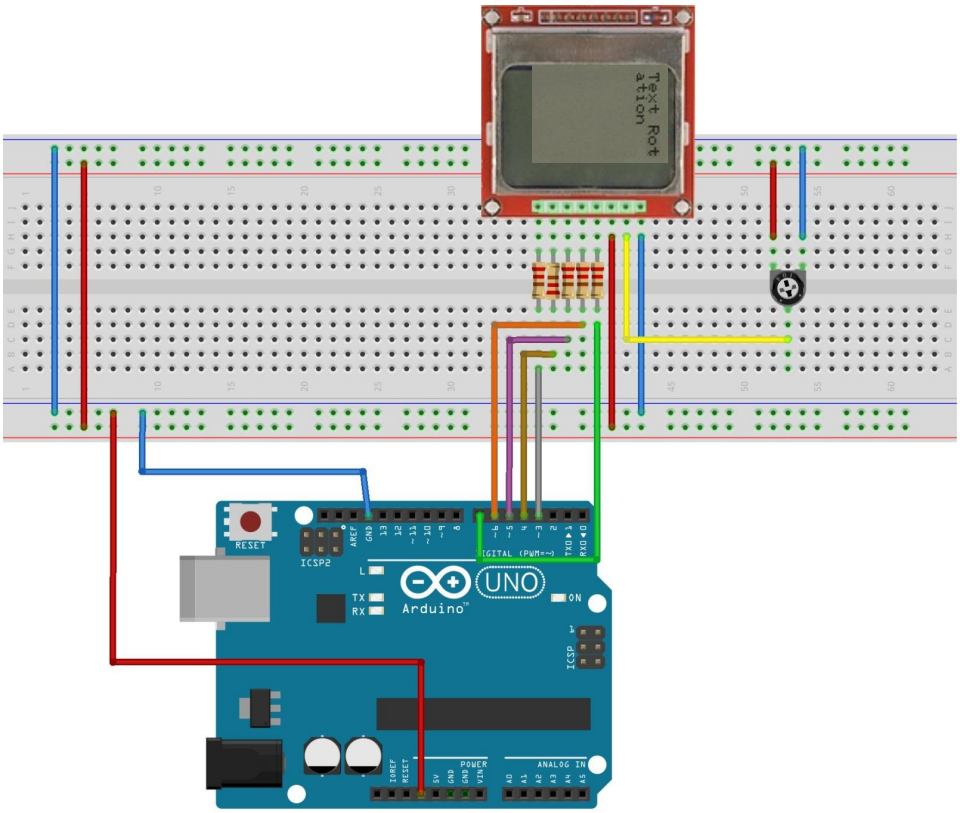
Acestea sunt afișaje grafice de 84×48 pixeli. Acestea se conectează la microcontrolere printr-o interfață magistrală serială similară cu SPI. LCD-ul vine și cu o lumină de fundal în diferite culori, cum ar fi roșu, verde, albastru și alb. Iluminarea de fundal nu este altceva decât patru LED-uri distanțate în jurul marginilor afișajului.
Are 8 pini care îl interfață cu Arduino, pinout-ul este după cum urmează:

- **RST:** resetează afișajul. Este o semnificație activă a pinului scăzut; puteți reseta afișajul trăgând-l jos. De asemenea, puteți conecta acest pin la resetarea Arduino, astfel încât să resetați ecranul automat;
- **CE(Chip Enable**): este folosit pentru a selecta unul dintre multele dispozitive conectate care partajează aceeași magistrală SPI;
- **D/C(Data/Command):** PIN-ul spune afișajului dacă datele pe care le primește sunt o comandă sau date afișabile;
- **DIN:** este un pin de date serial pentru interfața SPI;
- **CLK:** este un pin de ceas serial pentru interfața SPI;
- **VCC:** furnizează energie pentru LCD-ul pe care îl conectăm la pinul de 3,3V volți de pe Arduino;
- **BL(Backlight):** controlează iluminarea de fundal a afișajului. Pentru a-i controla luminozitatea, puteți adăuga un potențiometru sau puteți conecta acest pin la orice pin Arduino compatibil PWM;
- **GND:** este un pin de masă și ar trebui să fie conectat la pământul Arduino.

Puteți conecta pinii de transmisie a datelor la orice pin I/O digital. LCD-ul are niveluri de comunicare de 3v, așa că nu putem conecta direct acești pini la Arduino. O modalitate este să adăugați rezistențe în linie cu fiecare pin de transmisie a datelor. Doar adăugați rezistențe de 10kΩ între pinii CLK, DIN, D/C și RST și un rezistor de 1kΩ între CE. Pinul de iluminare de fundal (BL) este conectat la 3,3 V printr-un rezistor de limitare a curentului de 330Ω. Puteți adăuga un potențiometru sau conecta acest pin la orice pin Arduino compatibil PWM, dacă doriți să-i controlați luminozitatea.
Comunitatea Arduino a dezvoltat deja câteva biblioteci pentru a gestiona aceste afișaje NOKIA, cum ar fi biblioteca Adafruit PCD8544 Nokia 5110 LCD. Pentru a instala biblioteca, navigați la Schița > Includeți biblioteca > Gestionați bibliotecile... Așteptați ca Managerul bibliotecii să descarce indexul bibliotecilor și să actualizeze lista bibliotecilor instalate.

Circuit 5:
Numele circuitului: "Text rotit"
Explicația circuitului: Puteți roti conținutul afișajului apelând funcția setRotation(). Vă permite să vizualizați afișajul în modul portret sau să îl întoarceți cu susul în jos.
Notă: Funcția acceptă un singur parametru care corespunde la 4 rotații cardinale. Această valoare poate fi orice număr întreg nenegativ începând de la 0. De fiecare dată când creșteți valoarea, conținutul afișajului este rotit cu 90 de grade în sens invers acelor de ceasornic. De exemplu:0 – Keeps the screen to the standard landscape orientation.

- 1 – Rotește ecranul 90° la dreapta.
- 2 – Întoarce ecranul cu susul în jos.
- 3 – Royește ecranul 90° la stânga.




```
/*  Text rotit */
#include <SPI.h>        // această bibliotecă vă permite să comunicați cu dispozitivele SPI, cu
                        Arduino ca dispozitiv principal.
#include <Adafruit_GFX.h>          // bibliotecile afișajului OLED
#include <Adafruit_PCD8544.h>      // Declare LCD object for software
SPIAdafruit_PCD8544(CLK,DIN,D/C,CE,RST);
Adafruit_PCD8544 display = Adafruit_PCD8544 (7, 6, 5, 4, 3);
```

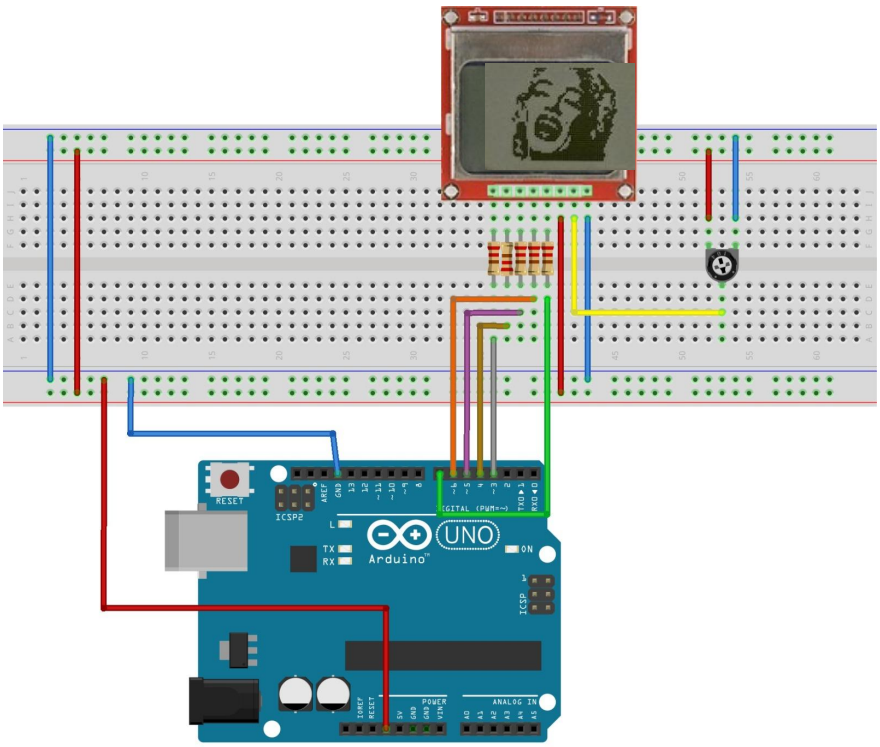
```
void setup()  {
        Serial.begin(9600);
        //Inițializează afișajul
        display.begin();
        display.setContrast(57);        // puteți modifica contrastul pentru a adapta afișajul
        display.clearDisplay();         // golește tamponul
// Text Rotation
while(1)
{
        display.clearDisplay();
        display.setRotation(rotatetext);
        display.setTextSize(1);
        display.setTextColor(BLACK);
        display.setCursor(0,0);
        display.println("Text Rotation");
        display.display();
        delay(1000);
        display.clearDisplay();
        rotatetext++;
}
}
void loop() {}
```

Circuit 6:

Numele circuitului: "Imaginea lui Marilyn Bmp"

Explicația circuitului: cum să desenați imagini bitmap pe ecranul LCD Nokia 5110. În acest exemplu există un portret al lui Marilyn Monroe.

Notă: rezoluția ecranului ecranului LCD Nokia 5110 este de 84×48 pixeli, așa că imaginile mai mari decât acestea nu vor fi afișate corect. Pentru a afișa imaginea bitmap pe ecranul LCD al Nokia 5110, trebuie să apelăm funcția drawBitmap(). Este nevoie de șase parametri:. colțul din stânga sus coordona X, colțul din stânga sus coordonata Y, matrice de octeți a bitmap-ului monocrom, lățimea bitmap-ului în pixeli, înălțimea bitmap-ului în pixeli și Culoare. În exemplul nostru, imaginea bitmap are dimensiunea de 84×48. Deci, coordonatele X și Y sunt setate la 0, în timp ce lățimea și înălțimea sunt setate la 84 și 48.




```
/*  Imaginea lui Marilyn Bmp */

#include <SPI.h>

#include <Adafruit_GFX.h>

#include <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, 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, 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, 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,
        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,
        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, 0xf0, 0x0f,
        0xfe, 0x09, 0xff, 0xf7, 0xfc, 0x00, 0x00, 0x00, 0x08, 0xf8, 0x01, 0xfc, 0x19, 0xff, 0xff, 0xf8,
        0x00, 0x00, 0x00, 0x0c, 0x78, 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, 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();
        // Afișează desenul
        display.drawBitmap(0, 0,  MarilynMonroe, 84, 48, BLACK);
        display.display();
        // Rotește desenul
        //display.invertDisplay(1);
}
void loop() {}
```

Erasmus+ KA-202

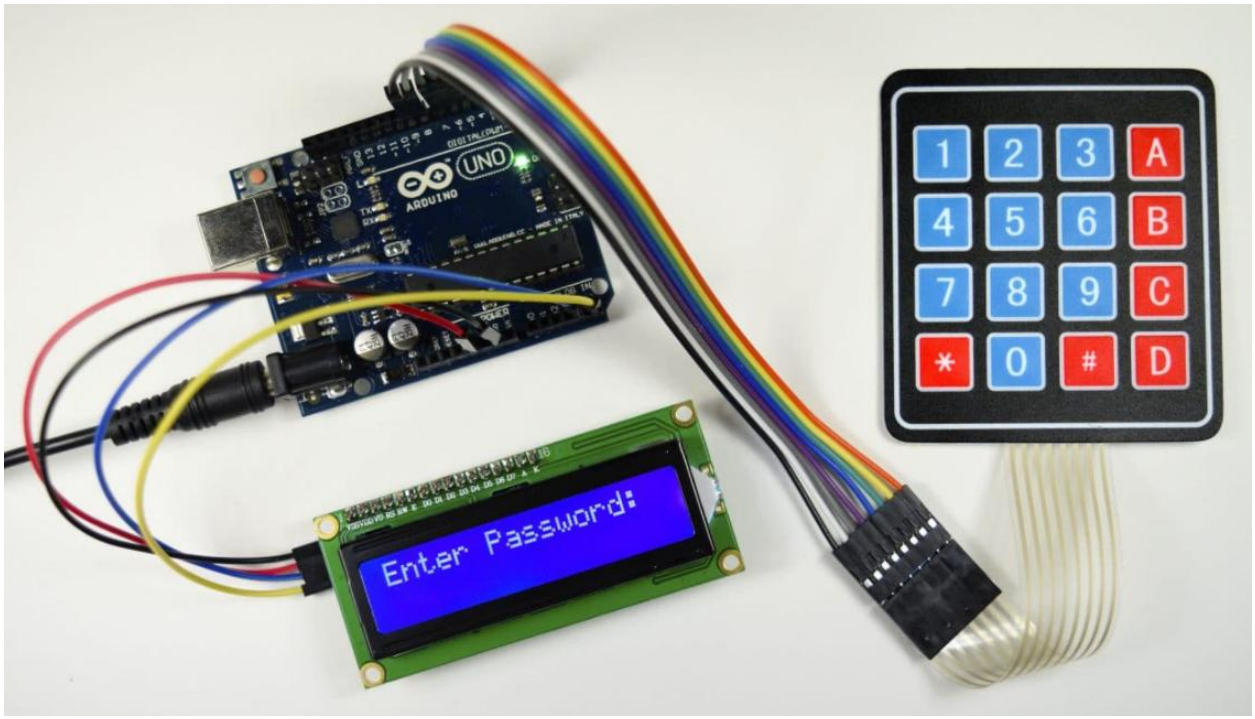
Proiecte de parteneriate strategice pentru educație și formare profesională

Titlul proiectului: "Predarea și învățarea ARDUINO în formarea profesională"

Acronimul proiectului: " ARDUinVET "

Numărul proiectului: "2020-1-TR01-KA202-093762"

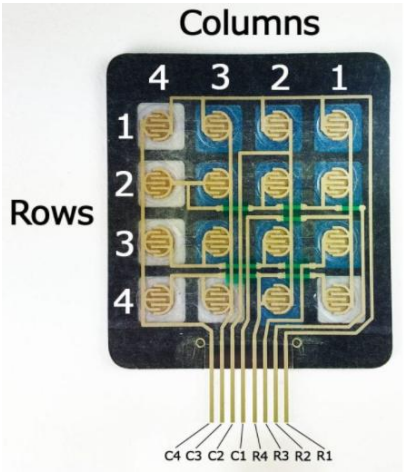
Tastatură modul și kit de instruire



Ce este o tastatură?

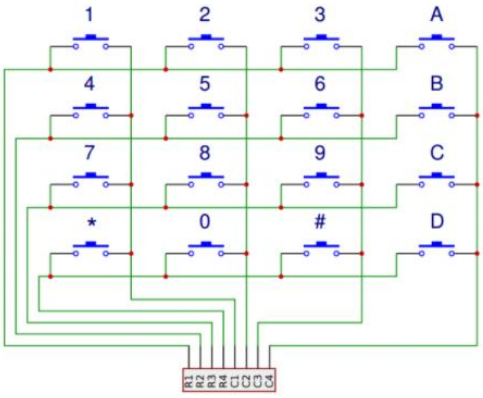
O tastatură este un sistem de butoane aranjate într-o matrice care funcționează ca un dispozitiv de comutare pentru a asigura o conexiune între o linie și o coloană.

Vom folosi o tastatură cu membrană cu matrice 4X4, deoarece este subțire și are suport adeziv, astfel încât să poată fi lipită pe majoritatea suprafețelor plane. Sub fiecare cheie este un comutator cu membrană. Fiecare întrerupător dintr-un rând este conectat la celelalte întrerupătoare din rând printr-o urmă conductivă de sub placă. Fiecare întrerupător dintr-o coloană este conectat în același mod - o parte a comutatorului este conectată la toate celelalte comutatoare din acea coloană printr-o urmă conductivă. Fiecare rând și coloană este scos la un singur pin, pentru un total de 8 pini pe o tastatură 4X4.	



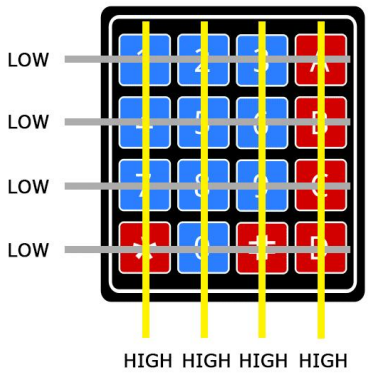
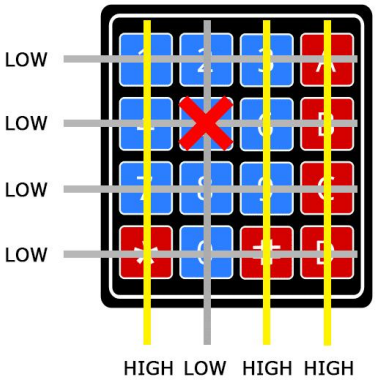
Apăsarea unui buton închide comutatorul dintre o coloană și o urmă de rând, permițând curentului să curgă între un pin de coloană și un pin de rând.

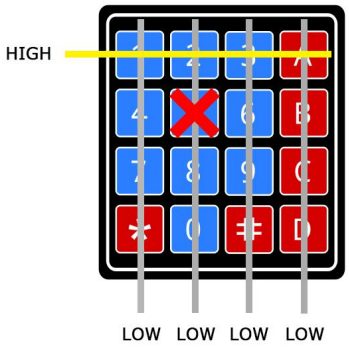
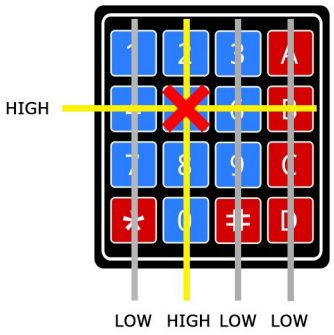
Schema pentru o tastatură 4X4 arată cum sunt conectate rândurile și coloanele:



Arduino detectează ce buton este apăsat detectând pinul de rând și coloană care este conectat la buton.

Acest lucru se întâmplă în patru pași:

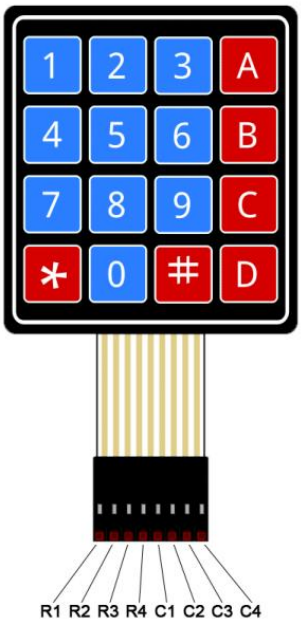
1. În primul rând, atunci când nu este apăsat niciun buton, toți pinii coloanei sunt ținuți ÎNALT, iar toți pinii rândului sunt ținuți JOS	
2. Când este apăsat un buton, pinul coloanei este tras la JOS deoarece curentul din coloana ÎNALT trece la pinul rândului JOS:	

3. Arduino știe acum în ce coloană se află butonul, așa că acum trebuie doar să găsească rândul în care se află butonul. Face acest lucru schimbând fiecare dintre pinii rândului în HIGH și, în același timp, citind toți pinii coloanei la detectați ce pin de coloană revine la HIGH	
4. Când pinul coloanei devine din nou HIGH, Arduino a găsit pinul de rând care este conectat la buton:	

Din diagrama de mai sus, puteți vedea că combinația dintre rândul 2 și coloana 2 ar putea însemna doar că a fost apăsat butonul numărul 5.

CONECTAȚI TASTATURA LA ARDUINO

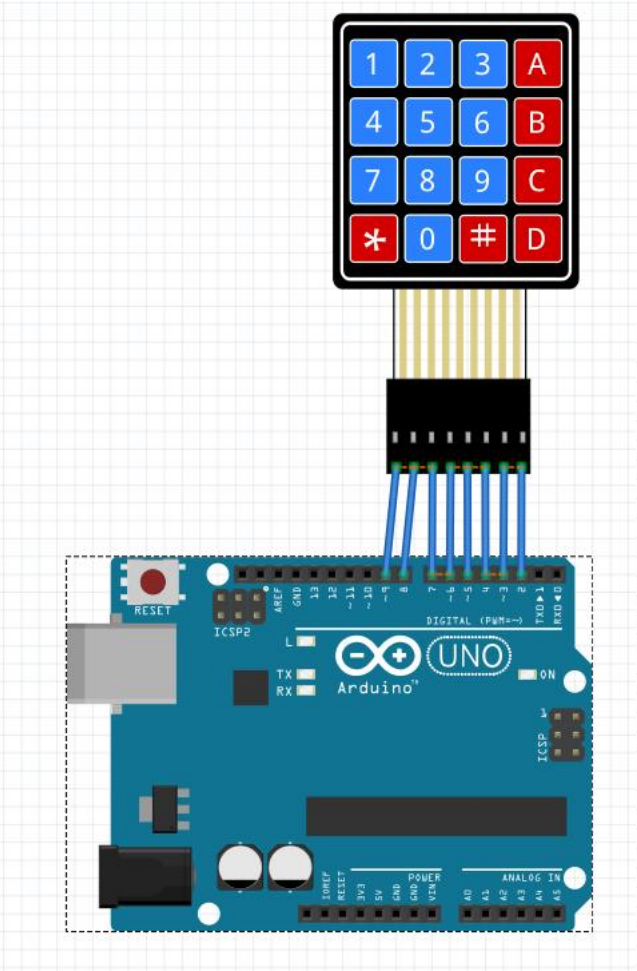
Dispunerea pinilor pentru majoritatea tastaturilor cu membrană va arăta astfel:



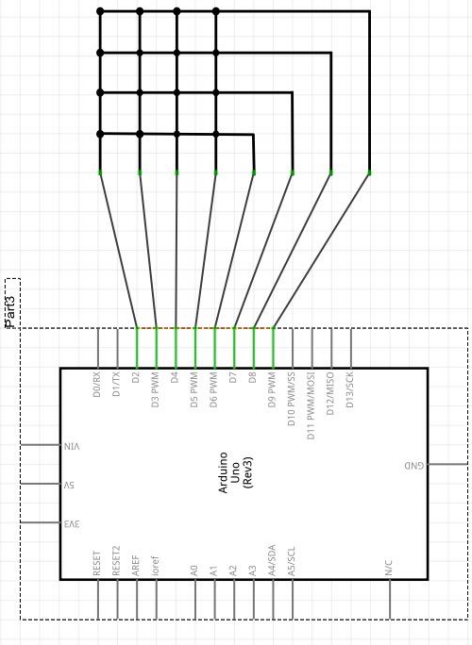
Circuit 1

Circuit title: Monitorul serial afișează tasta apăsată

Circuit description: Programul ne va arăta cum să afișăm fiecare apăsare de tastă pe monitorul serial.



1-) Vedere panou

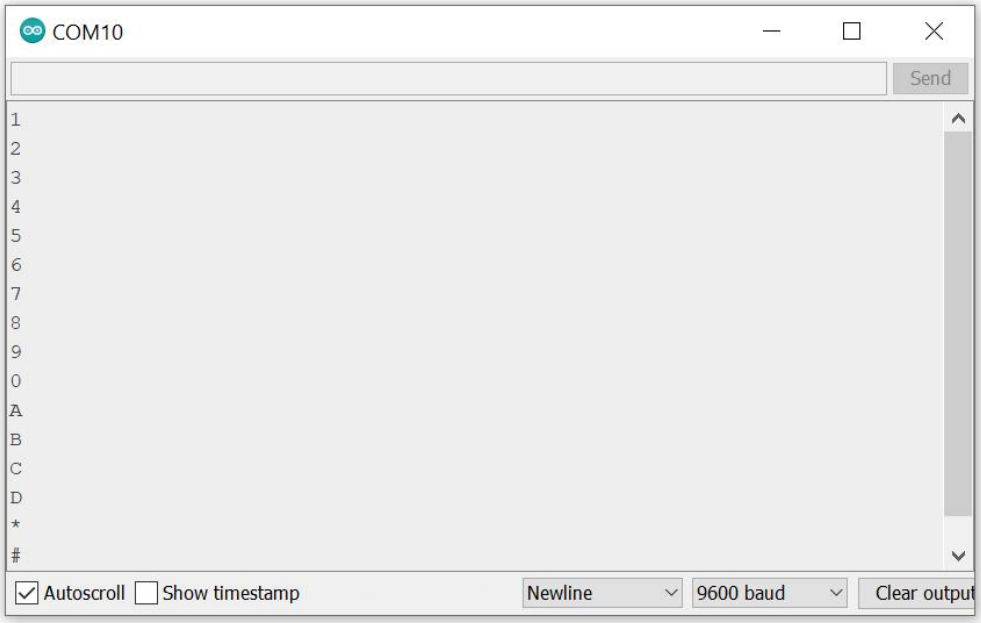


2-) Vedere schematică

/* “Monitorul serial afișează tasta apăsată” */

```
#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'},
```

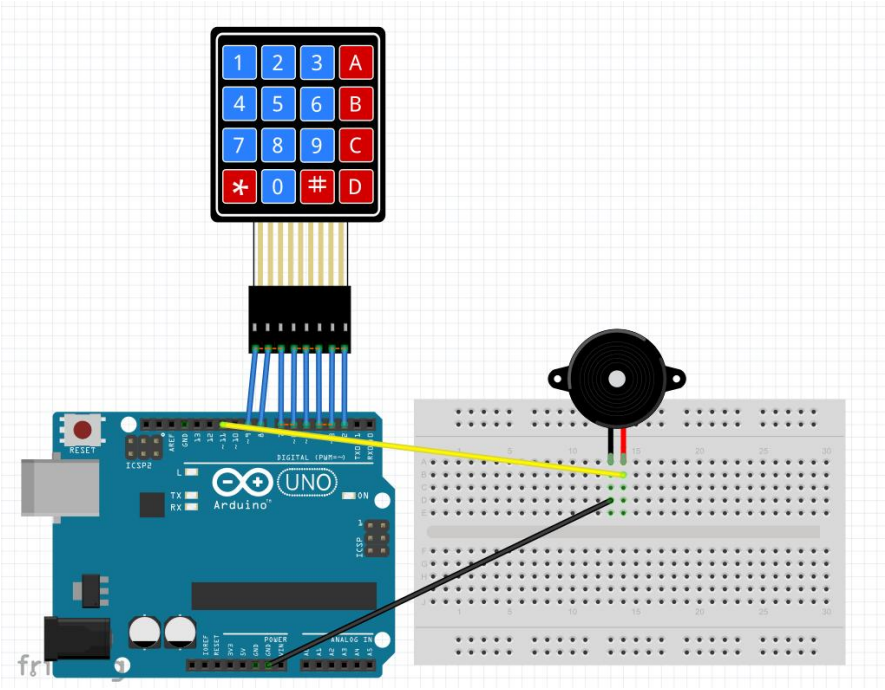
```
  {'*', '0', '#', 'D'}
};
byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS,
COLS);
void setup(){
  Serial.begin(9600);
}
void loop(){
  char customKey = customKeypad.getKey();
  if (customKey){
    Serial.println(customKey);
  }
}
```



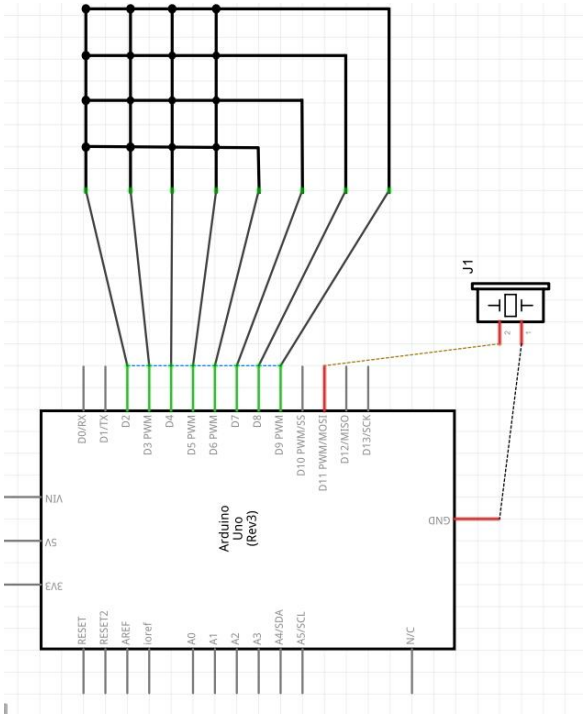
Circuit 2

Circuit title: Bip la apăsarea tastaturii

Circuit description: Când o tastă de pe tastatură este apăsată, soneria piezo emite un bip



1-) Vedere panou



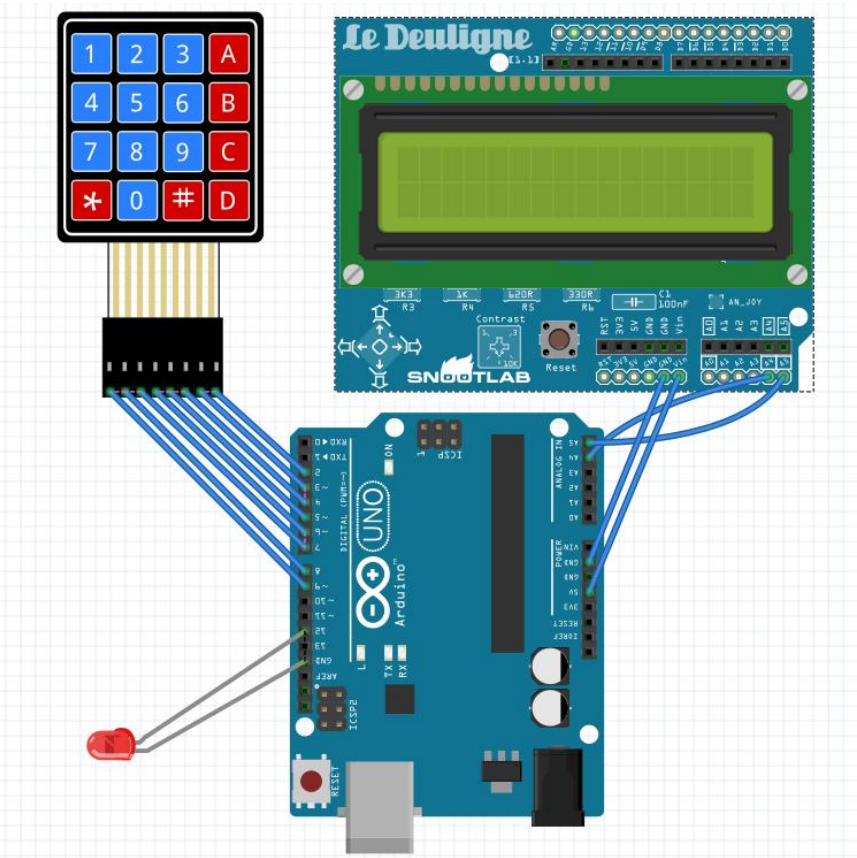
2-) Vedere schematică

```
/*    “Bip la apăsarea tastaturii”   */

#include <Keypad.h>
#include <ezBuzzer.h>
const int BUZZER_PIN = 11;
const int ROW_NUM    = 4; // patru rânduri
const int COLUMN_NUM = 4; // patru coloane
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};      // conectarea pinurilor rândului de pe
tastatură
byte pin_column[COLUMN_NUM] = {5, 4, 3, 2}; // conetarea pinurilor coloanelor de pe
tastatură
Keypad keypad = Keypad(makeKeymap(keys), pin_rows, pin_column, ROW_NUM,
COLUMN_NUM );
ezBuzzer buzzer(BUZZER_PIN); // obiectulBuzzer se atașează la un 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); // afișează tasta de Monitorul serial
    buzzer.beep(100);  // generează un sunet de 100 ms
  }
}
```

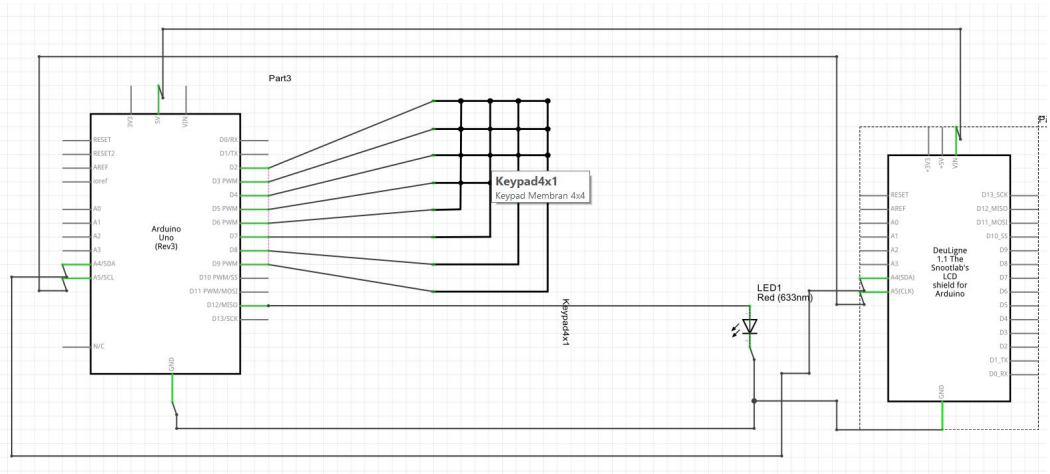
Circuit 3

Circuit title: Cod de deblocare Arduino

Circuit description: Program care setează o tastatură pe Arduino. Un LED se va aprinde când introduceți codul corect



1-) Vedere panou



2-) Vedere schematică

```
/* Cod de deblocare Arduino */
#include <Keypad.h> //Bibliotecă care poate fi descărcată de pe Arduino IDE
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#define Solenoid 12          // Poarta tranzistorului care controlează solenoidul
//se folosește un simplu LED
#include <liquidcrystal_i2c.h></liquidcrystal_i2c.h></lcd.h></wire.h></keypad.h>
#define I2C_ADDR 0x27 // LCD i2c adresa pinilor
#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; //numărul de rânduri de pe tastatură
const byte numCols= 4; //numărul de coloane de pe tastatură
int code = 1234;  //acesta este codul
int tot,i1,i2,i3,i4;
char c1,c2,c3,c4;
//definește tasta apăsată în funcție de rând și coloane așa cum apare pe keymap
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}; //Râmdurile 0 to 3
byte colPins[numCols]= {5,4,3,2}; //Coloanele 0 to 3
//inițializează o instanță a clasei Keypad
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();  //Funcția getKey menține programul să ruleze,
atâta timp cât nu ați apăsat „*”, totul de mai jos nu va fi declanșat
      if (keypressed == '*')          // puteți folosi restul codului dvs. pur și simplu
         {
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Enter Code");              //când tasta „*” este apăsată, puteți introduce
parola
              keypressed = myKeypad.waitForKey();  // aici toate programele sunt oprite
până când introduceți cele patru cifre, apoi se compară cu codul de mai sus
              if (keypressed != NO_KEY)
               {
                c1 = keypressed;
```

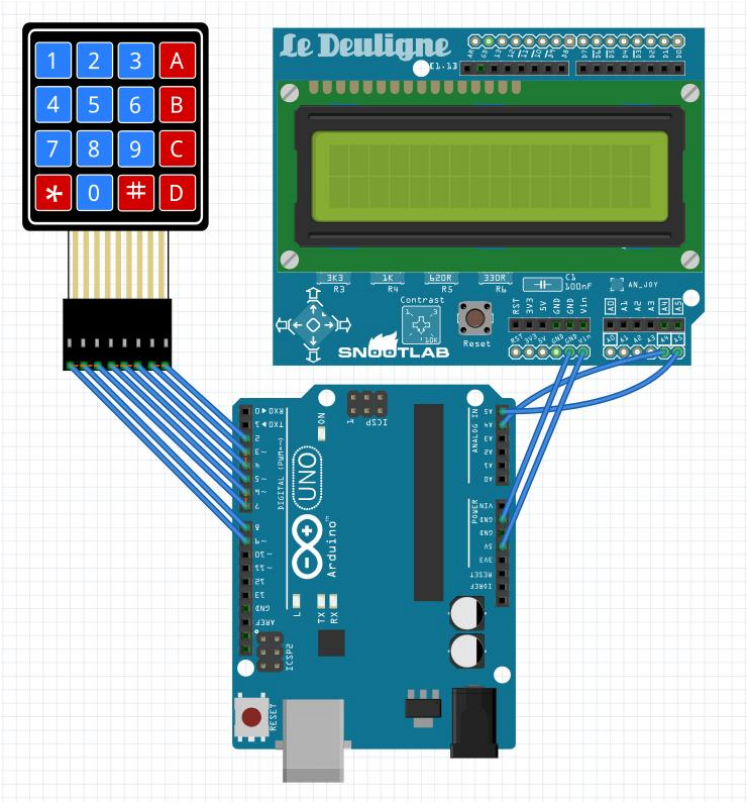
```
                lcd.setCursor(0, 1);
                lcd.print("*");
                }
              keypressed = myKeypad.waitForKey();
              if (keypressed != NO_KEY)
               {
                c2 = keypressed;
                lcd.setCursor(1, 1);
                lcd.print("*");
                }
               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;       //tastele apăsate sunt stocate în caractere, le convertesc în int, apoi am făcut o multiplicare pentru a obține codul ca int de xxxx
               i2=(c2-48)*100;
               i3=(c3-48)*10;
               i4=c4-48;
               tot=i1+i2+i3+i4;
           if (tot == code) //dacă codul este corect, declanșați orice doriți aici, imprimați doar un mesaj pe ecran
      {
```

```
        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 //daca codul este gresit primesti altceva
       {
        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");
        }
          }
}
```

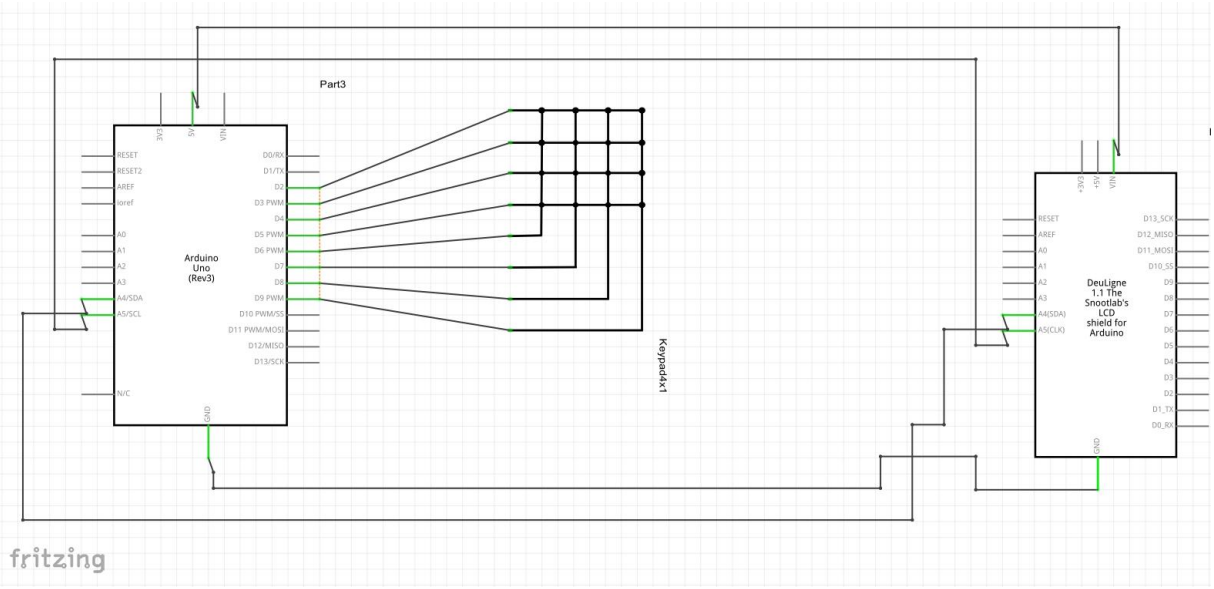
Circuit 4

Circuit title: Arduino calculator

Circuit description: Program face calcule matematice simple



1-) Vedere placă



2-) Vedere schematică

```
/* Arduino calculator */
#include <Keypad.h>
#include <EEPROM.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#define I2C_ADDR 0x27          //I2C adresă
#define BACKLIGHT_PIN 3        // Declararea pinior LCD
#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; // patru rânduri
const byte COLS = 4; // patru coloane
// definirea 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}; //Rândurile 0 to 3
byte colPins[COLS]= {5,4,3,2}; //Coloanele 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); //Iluminare de fundal
        lcd.print("Calculator Ready"); //Afișează un mesaj introductiv
  lcd.setCursor(0, 1);   // setarea cursorului la coloana 0, linia 1
  lcd.print("A=+ B=- C=* D=/"); //Afișează un mesaj introductiv
   delay(6000); //Așteptați ca afișajul să afișeze informații
    lcd.clear(); //apoi șterge tot
                    //          for(i=0 ; i<sizeof(code);i++){       //Când încărcați codul prima dată,
păstrați-l comentat
//          EEPROM.get(i, code[i]);           //Încărcați codul și schimbați-l pentru a-l stoca în
EEPROM
//          }       //Apoi anulați comentariul pentru bucla și reîncărcați codul (se face o
singură dată)
        }
void loop() {
  key = kpd.getKey(); // stocarea valorii tastei apăsate într-un caracter
if (key!=NO_KEY)
DetectButtons();
if (result==true)
CalculateResult();
DisplayResult();
}
void DetectButtons()
{
    lcd.clear(); // apoi șterge tot
   if (key=='*') // Dacă anulează tasta apăsată
   {Serial.println ("Button Cancel"); Number=Num1=Num2=0; result=false;}
       if (key == '1') // dacă tasta 1 este apăsată
   {Serial.println ("Button 1");
   if (Number==0)
   Number=1;
```

```
else
Number = (Number*10) + 1; //apasă de două ori
}
    if (key == '4') // dacă tasta 4 este apăsată
{Serial.println ("Button 4");
if (Number==0)
Number=4;
else
Number = (Number*10) + 4; //apasă de două ori
}
    if (key == '7') // dacă tasta 7 este apăsată
{Serial.println ("Button 7");
if (Number==0)
Number=7;
else
Number = (Number*10) + 7; //apasă de două ori
}
  if (key == '0')
{Serial.println ("Button 0"); //tasta 0 este apăsată
if (Number==0)
Number=0;
else
Number = (Number*10) + 0; //apasă de două ori
}
    if (key == '2') //tasta 2 este apăsată
{Serial.println ("Button 2");
 if (Number==0)
Number=2;
else
Number = (Number*10) + 2; //apasă de două ori
}
    if (key == '5')
{Serial.println ("Button 5");
```

```
 if (Number==0)
Number=5;
else
Number = (Number*10) + 5; //apasă de două ori
}
    if (key == '8')
{Serial.println ("Button 8");
 if (Number==0)
Number=8;
else
Number = (Number*10) + 8; //apasă de două ori
}
  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; //apasă de două ori
}
    if (key == '6')
{Serial.println ("Button 6");
if (Number==0)
Number=6;
else
Number = (Number*10) + 6; //apasă de două ori
}
    if (key == '9')
{Serial.println ("Button 9");
```

```
   if (Number==0)
   Number=9;
   else
   Number = (Number*10) + 9; //apasă de două ori
   }
    if (key == 'A' || key == 'B' || key == 'C' || key == 'D') //Detectarea butoanelor pe coloana
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);   // setați cursorul pe coloana 0, linia 1
 lcd.print(Num1); lcd.print(action); lcd.print(Num2);
   if (result==true)
 {lcd.print(" ="); lcd.print(Number);} //Afișați rezultatul
   lcd.setCursor(0, 1);   // setați cursorul pe coloana 0, linia 1
     lcd.print(Number); //Afișați rezultatul
}
```

Erasmus+ KA-202

Proiecte de parteneriate strategice pentru educație și formare profesională

Titlul proiectului: "Predarea și învățarea ARDUINO în formarea profesională"

Acronimul proiectului: " ARDUinVET "

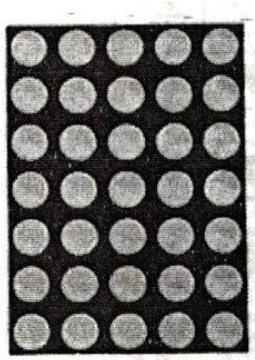
Numărul proiectului: "2020-1-TR01-KA202-093762"

Afișaj cu matrice în puncte

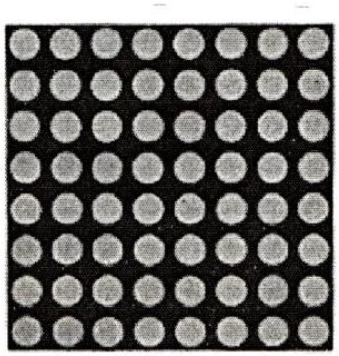
modul și kit de instruire

Ce este un afișaj cu matrice în puncte?

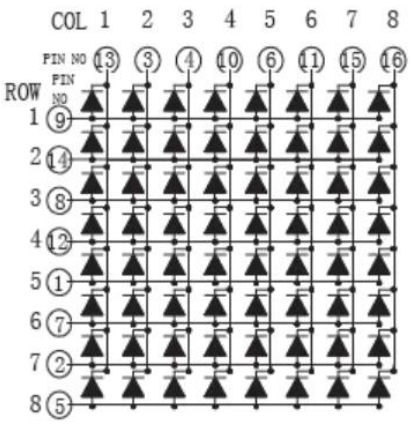
Afișajul cu matrice în puncte este un grup de LED-uri aranjate într-un sistem specific. O matrice LED standard este alcătuită din 5x7 sau 8x8 LED – uri dispuse în rânduri și coloane așa cum se arată în figura de mai jos. O matrice în puncte 5x7 are 5 rânduri și 7 coloane și o matrice în puncte 8x8 are 8 rânduri și 8 coloane, care sunt conectate împreună.

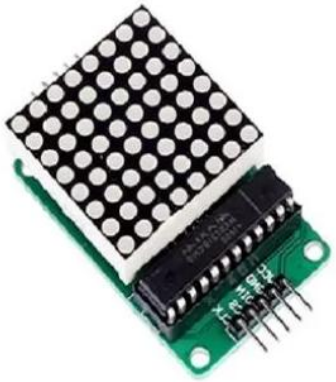


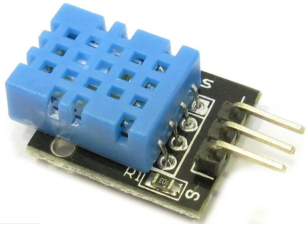
Afișaj matrice în puncte 5x7



Afișaj matrice în puncte 8x8

Dacă ne uităm la un afișaj cu matrice de puncte 8x8, aceasta conține 16 pini din care 8 pini folosiți pentru rânduri și 8 pentru coloane. Asta înseamnă, în rânduri și coloane, un total de 64 de LED-uri. Începem de la Pinul # 1 la Pinul # 8. Pinul numărul 1 este R5 (Rândul-5) și Pinul numărul 8 este R3 (Rândul-3) în partea de jos. În partea superioară de la Pinul 9 (Rândul-1) la Pinul 16 (coloana-1). Un începător încurcă întotdeauna și începe de la zero, pentru că știm poza/diagrama. de multe ori primim de la o sursă, de asemenea, trebuie să alegem care dintre ele este +VE și -VE. ar putea fi un expert care poate înțelege tipul de catod/anod comun.	

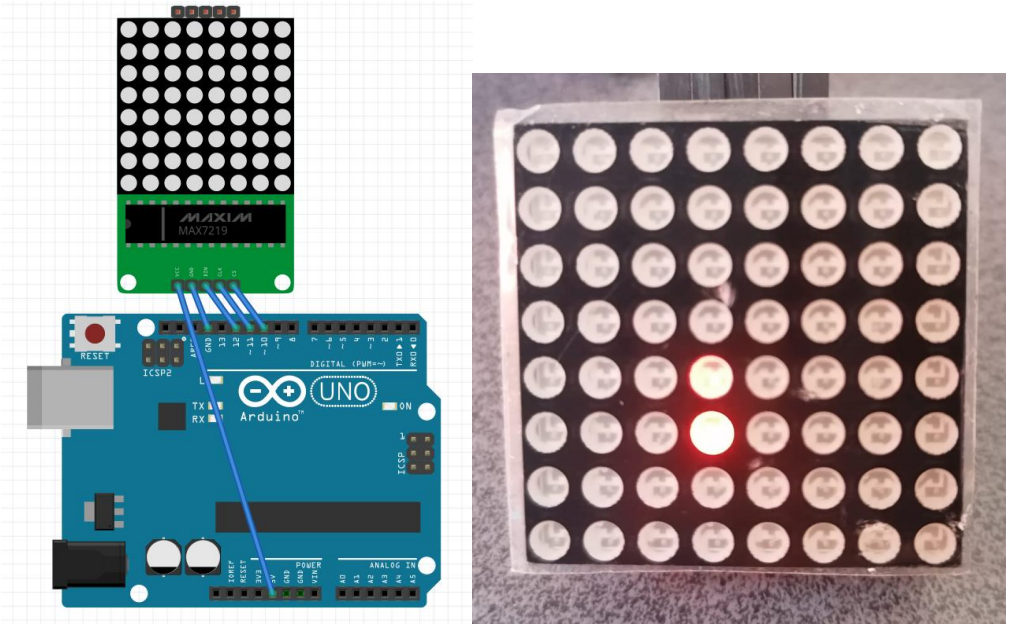
Afișajul cu matrice de puncte MAX7219 LED este unul dintre cele populare disponibile pe piață și utilizat de studenți, pasionații de electronică și în special în aplicațiile de afișare industrială. Există două tipuri de module disponibile în general. Acestea sunt modulul generic. Fiecare modul este format din două unități. Unul este matricea de puncte LED 8X8, iar celălalt este MAX7219 IC. MAX7219 este un driver de afișaj cu catod comun, cu intrări seriale și pini de ieșire. Are o capacitate de curent reglabilă care poate fi setată folosind doar un rezistor extern. În plus, are o interfață serială cu patru fire care poate fi conectată cu ușurință la toate microprocesoarele. Poate conduce 64 de LED-uri individuale conectate la pinii săi de ieșire folosind doar 4 fire folosind Arduino. În plus, poate conduce afișări cu matrice de puncte, afișări cu 7 segmente și grafice cu bare. În plus, MAX7219 are un decodor BCD încorporat care îl face ușor de utilizat cu afișaje numerice cu șapte segmente. În plus, are o memorie RAM statică de 8×8 pe care o putem folosi pentru a stoca numere. Este unul dintre cele mai populare drivere de afișare IC.	

	DHT11 este un senzor digital de temperatură și umiditate. Folosește un senzor capacitiv de umiditate și un termistor pentru a măsura aerul din jur și scoate un semnal digital pe pinul de date (nu sunt necesari pini de intrare analogice).

Circuit 1:

Circuit title: Aprinderea tuturor LED - urilor unul câte unul

Circuit description: Afișajul aprinde unul câte unul toate LED - urile



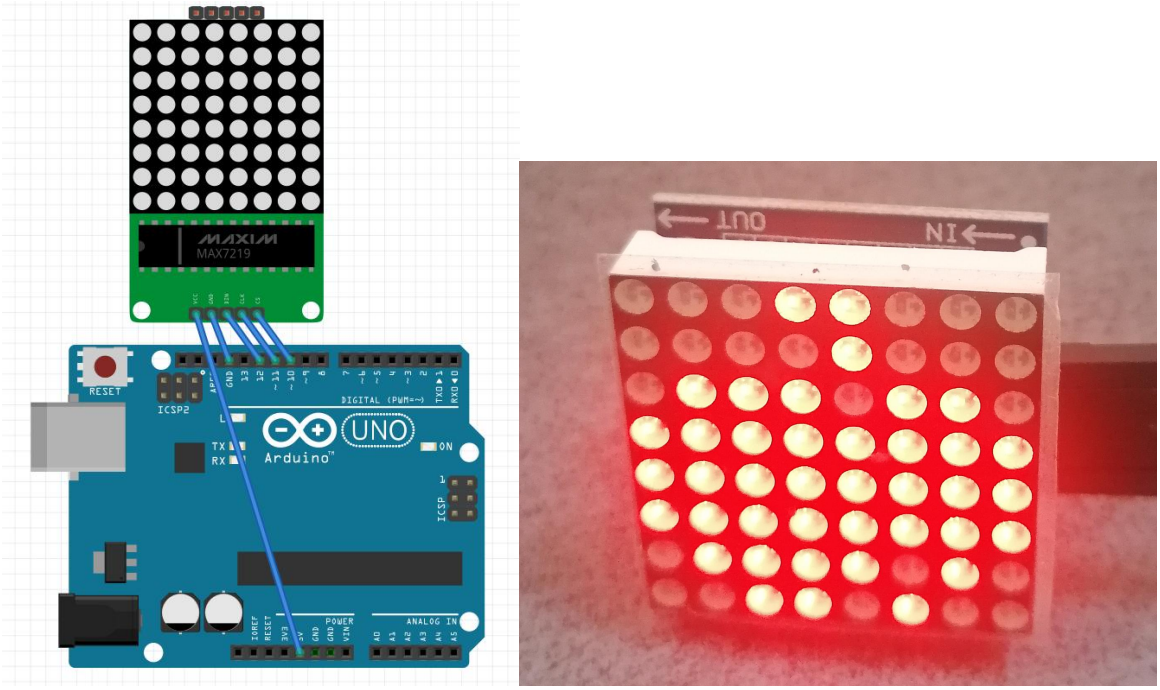
1-) Vedere panou

```
/*   Aprinderea tuturor LED - urilor unul câte unul  */
#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); } }}
```

Circuit 2:

Circuit title: Afișarea pictogramei Apple

Circuit description: Afișorul aprinde pictograma Apple



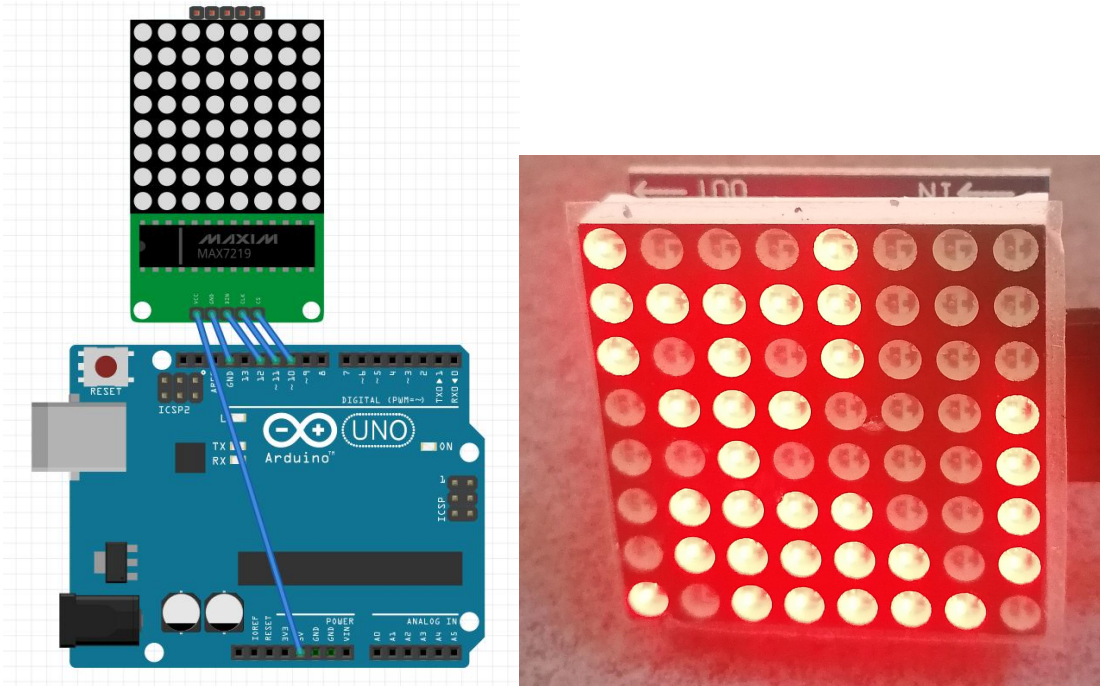
1-) Vedere panou

```
/*  Afișorul aprinde pictograma Apple*/
#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,B001
10100};
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]);   }
```

Circuit 3:

Circuit title: Afișarea pictogramei pisică

Circuit description: Afișorul aprinde pictograma pisică



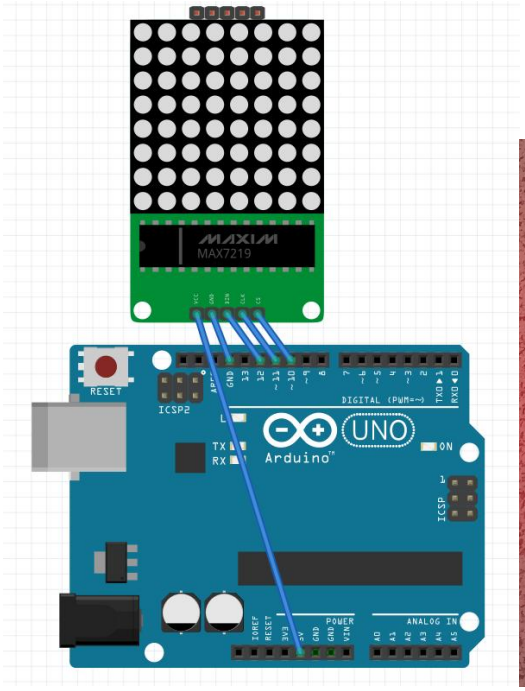
1-) Vedere panou

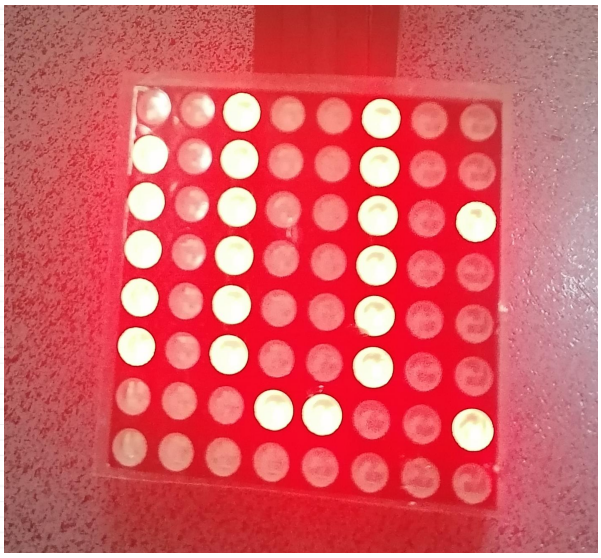
```
/*  Afișorul aprinde pictograma pisică*/
#include <LedControl.h>
int DIN = 12;
int CS = 10;
int CLK = 11;
LedControl lc=LedControl(DIN, CLK, CS,0);
int Cat[8]
={B10001000,B11111000,B10101000,B01110001,B00100001,B01111001,B01111101,B10111
110 };
void setup() {
  lc.shutdown(0,false);
  lc.setIntensity(0,0);
  lc.clearDisplay(0);  }
  void loop(){
    for(int i=0;i<8;i++) lc.setRow(0,i,Cat[i]); }
```

Circuit 4:

Circuit title: Afișarea textului curgător ARDUINVET

Circuit description: Afișorul rulează textul ARDUINVET





1-) Vedere panou

/* Afișorul rulează textul 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, // spațiu

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, // \ backspace

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

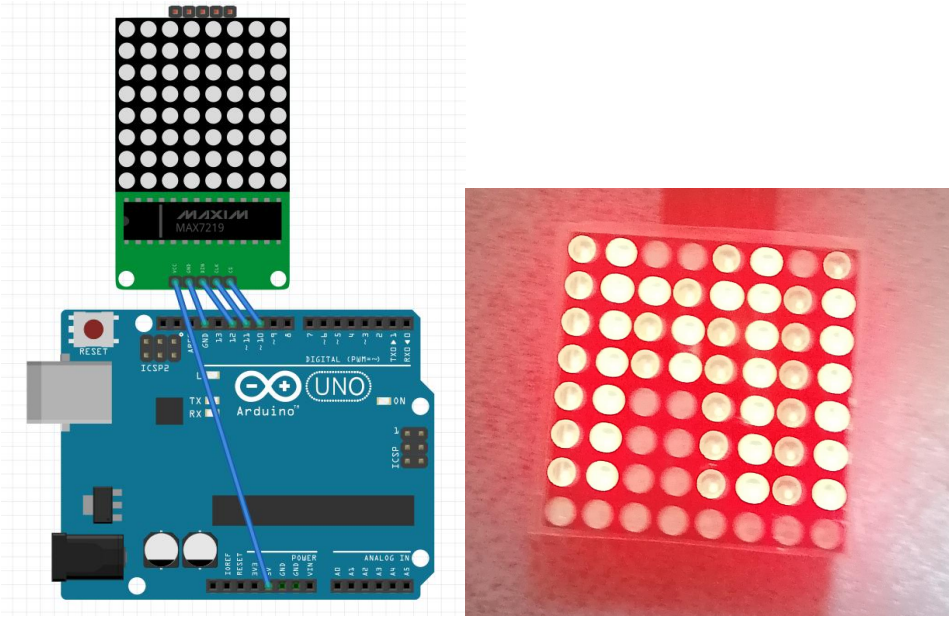
```
3, 8, B01000100, B01111101, B01000000, B00000000, B00000000, // i
4, 8, B01000000, B10000000, B10000100, B01111101, B00000000, // j
4, 8, B01111111, B00010000, B00101000, B01000100, B00000000, // k
3, 8, B01000001, B01111111, B01000000, B00000000, B00000000, // l
5, 8, B01111100, B00000100, B01111100, 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;    // Pinul DIN al modulului MAX7219
int load = 10;    // Pinul CS al modulului MAX7219
int clock = 11;  // PIN CLK al modulului MAX7219
int maxInUse = 1;    //modificați această variabilă pentru a seta câte MAX7219 veți folosi
MaxMatrix m(data, load, clock, maxInUse); // definiți modulul
byte buffer[10];
void setup(){
  m.init(); // inițializarea modulului
  m.setIntensity(2); // intensitate dot matrix 0-15
  Serial.begin(9600); // inițializarea comunicației seriale
```

```
 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++;  }       }
```

Circuit 5:

Circuit title: Afișarea tuturor literelor din alfabet

Circuit description: Afișorul afișează fiecare literă a alfabetului la interval de 1 secundă



1-) Vedere panou

```
/*  Afișarea tuturor literelor din alfabet*/
#include <MaxMatrix.h>//descărcați de pe https://code.google.com/archive/p/arudino-maxmatrix-library/downloads
int DIN = 12;   // Pinul DIN al modulului MAX7219
int CLK = 11;   // PIN CLK al modulului MAX7219
int CS = 10;    // Pinul CS al modulului MAX7219
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,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,B0011110
0};
char p[] = {8, 8,
B00000000,B00111000,B00100100,B00100100,B00111000,B00100000,B00100000,B0000000
0};
char q[] = {8, 8,
B00111100,B01000010,B01000010,B01000010,B01001010,B01000110,B00111110,B0000000
1};
char r[] = {8, 8,
B01111000,B01000100,B01000100,B01111000,B01100000,B01010000,B01001000,B0100010
0};
char s[] = {8, 8,
B00111100,B01000000,B01000000,B00111000,B00000100,B00000100,B01111000,B0000000
0};
char t[] = {8, 8,
B00000000,B01111100,B00010000,B00010000,B00010000,B00010000,B00010000,B0000000
0};
char u[] = {8, 8,
B00000000,B01000010,B01000010,B01000010,B01000010,B01000010,B00111100,B0000000
0};
char v[] = {8, 8,
B00000000,B00100100,B00100100,B00100100,B00100100,B00100100,B00011000,B0000000
0};
char w[] = {8, 8,
B00000000,B01010100,B01010100,B01010100,B01010100,B01010100,B00111000,B0000000
0};
char x[] = {8, 8,
B00000000,B01000100,B00101000,B00010000,B00101000,B01000100,B00000000,B0000000
0};
char y[] = {8, 8,
B10000001,B01000010,B00100100,B00011000,B00011000,B00011000,B00011000,B0001100
0};
char z[] = {8, 8,
```

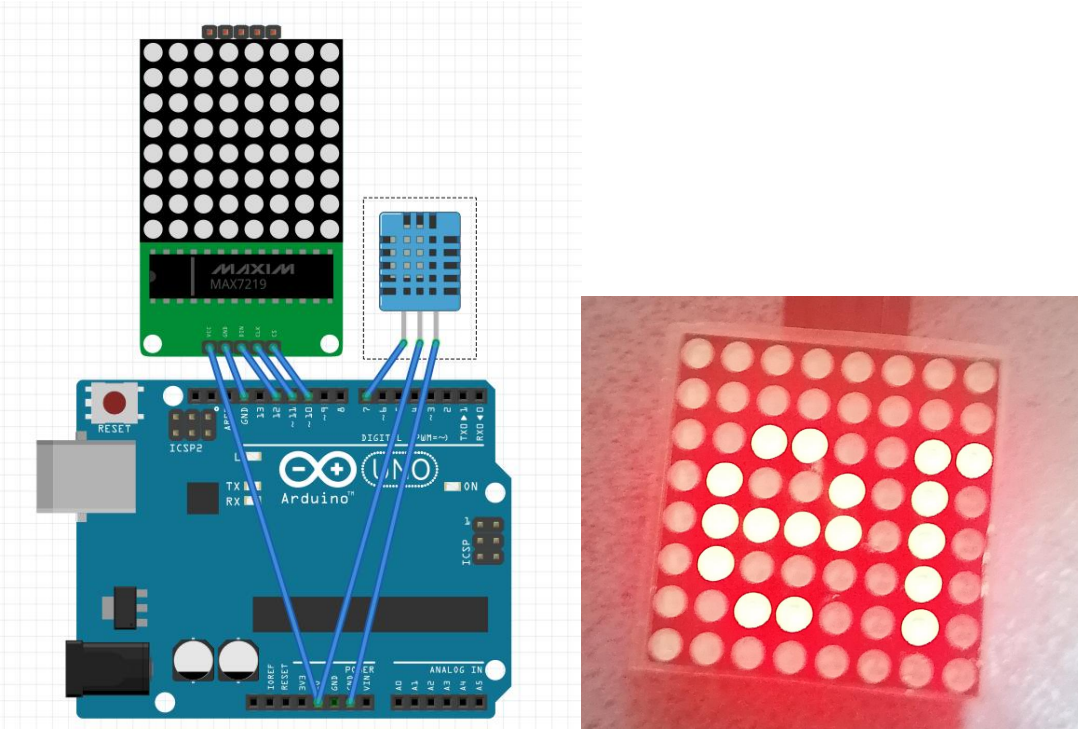
```
B00000000,B01111100,B00001000,B00010000,B00100000,B01111100,B00000000,B0000000
0};void setup() {
  m.init(); // Inițializare MAX7219
  m.setIntensity(5); // intensitatea inițială a matricei led, 0-15   }
void loop() {
  // Activarea sau oprirea LED-urilor la poziția x,y sau rând,coloană
  m.setDot(6,2,true);
  delay(1000);
  m.setDot(6,3,true);
  delay(1000);
  m.clear(); // Șterge afișajul
  for (int i=0; i<8; i++){
    m.setDot(i,i,true);
    delay(300);}
  m.clear();
  // Afișarea caracterului la x,y (colțul din stânga sus al caracterului)
  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);}
```

Circuit 6:

Circuit title: Interfață Dot Matrix cu Arduino

Circuit description: Afișorul afișează temperatura măsurată de senzorul DHT11



1-) Vedere panou

```
/*     Interfață Dot Matrix cu Arduino   */
//D7= temp
//D10=CS
//D11=CLK
//D12=DIN
#include <MaxMatrix.h> //include biblioteca de matrice
#include <avr/pgmspace.h>
#include <stdlib.h>
#include "DHT.h" //include biblioteca de senzori de temperatură
#define DHTPIN 7     // la ce pin suntem conectați
#define DHTTYPE DHT11   // Senzor de temperatură și umiditate DHT 11
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;    // Pinul DIN al modulului MAX7219
int load = 10;    // Pinul CS al modulului MAX7219
int clock = 11;  // PIN CLK al modulului MAX7219
int maxInUse = 1;    //modificați această variabilă pentru a seta câte MAX7219 veți folosi
```

```
MaxMatrix m(data, load, clock, maxInUse); // definiți modulul
byte buffer[10];
void setup(){
  m.init(); // inițializarea modulului
  m.setIntensity(2); // intensitate dot matice 0-15
  Serial.begin(9600); // inițializarea comunicației seriale
  Serial.println("DHTxx test!");
  dht.begin();
}
void loop(){
  int t = dht.readTemperature();
  char temp[4];
  itoa(t,temp,10); //convertiți int in 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++;
  }
}
```

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

Proiecte de parteneriate strategice pentru educație și formare profesională

Titlul proiectului: "Predarea și învățarea ARDUINO în formarea profesională"

Acronimul proiectului: " ARDUinVET "

Numărul proiectului: "2020-1-TR01-KA202-093762"

ARDUINO MOTOR

MODUL ȘI KIT DE INSTRUIRE

(DC, Step, Servo)

Modul și kit de instruire Arduino Motor

Kitul de antrenament al modulului motor este un instrument didactic dezvoltat în cadrul proiectului ARDUinVET, cu scopul de a explica într-un mod simplu funcționarea diferitelor tipuri de motoare (Servo, Step și DC), controlate prin Arduino.

Folosind platforma de programare a Arduino, ne propunem să abordăm într-o manieră simplă programarea Arduino-urilor pentru a pune în funcțiune diferitele motoare, cu utilizarea hardware-ului dezvoltat.

Scopul este de a arăta elevilor cum să construiască și să programeze o placă de circuit imprimat (figura 1), precum și aplicabilitatea acesteia, făcând motoarele să funcționeze și să execute diferite sarcini, în acest caz specific.

Schema de circuit motor shield

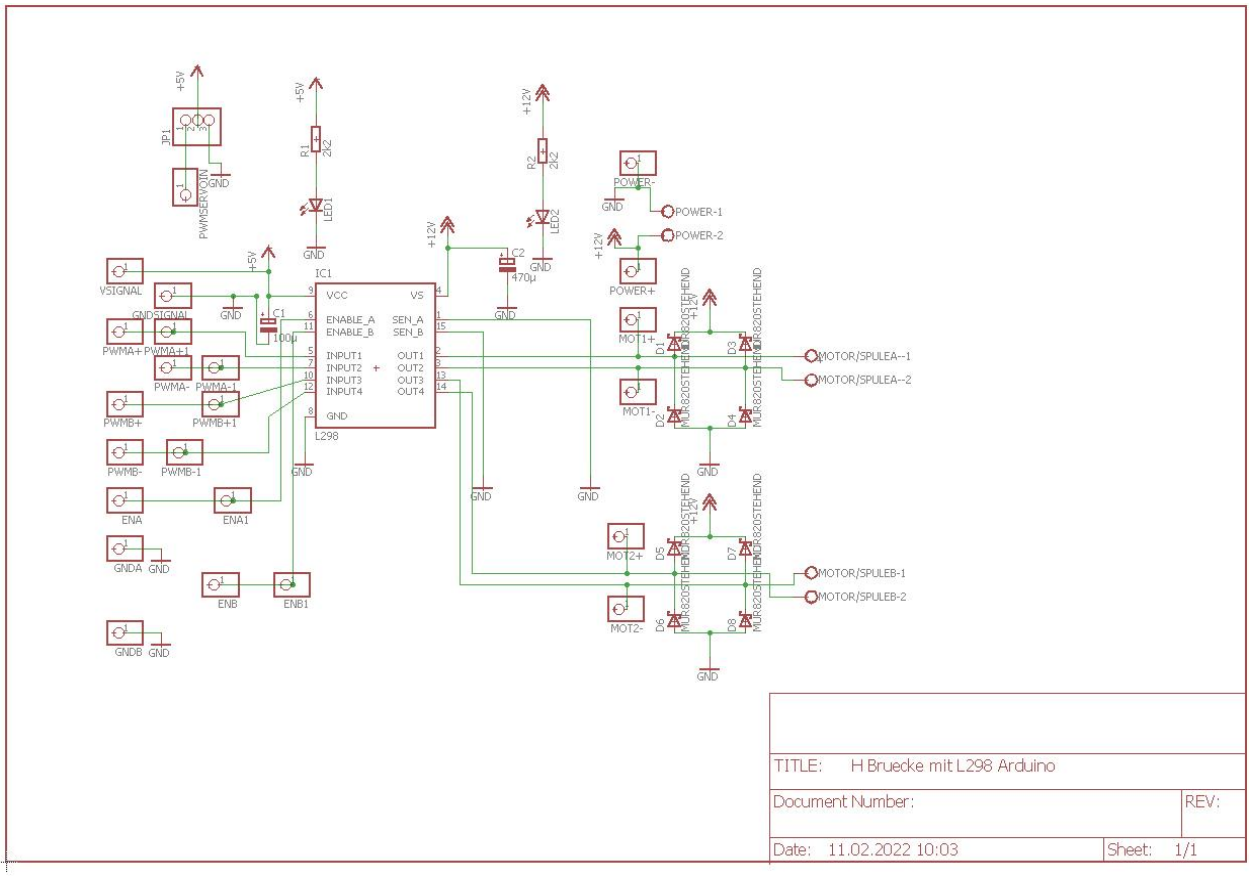


Figura: 1

Aspect motor shield

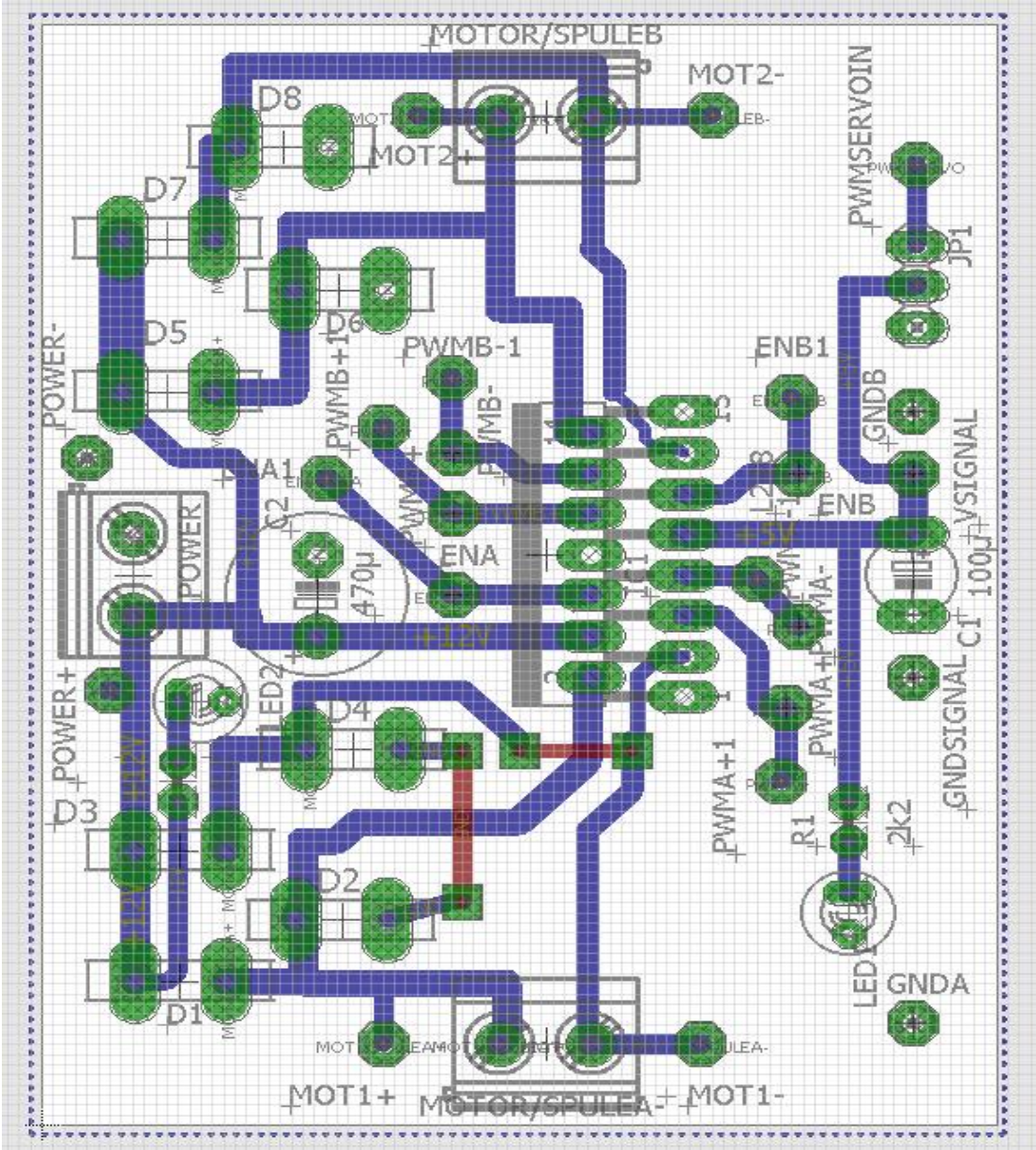


Figura: 2

Motoare

Motor de CC

Motoarele cu curent continuu, (motoarele cu curent continuu, imaginea 3) sunt dispozitive electronice care funcționează prin forțele de atracție și repulsie generate de magneți.

Exemple:

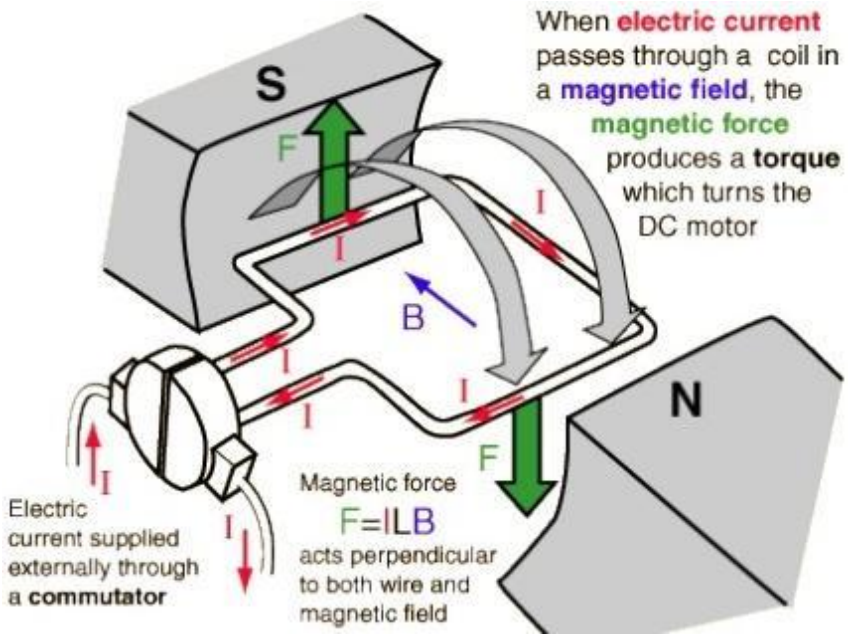
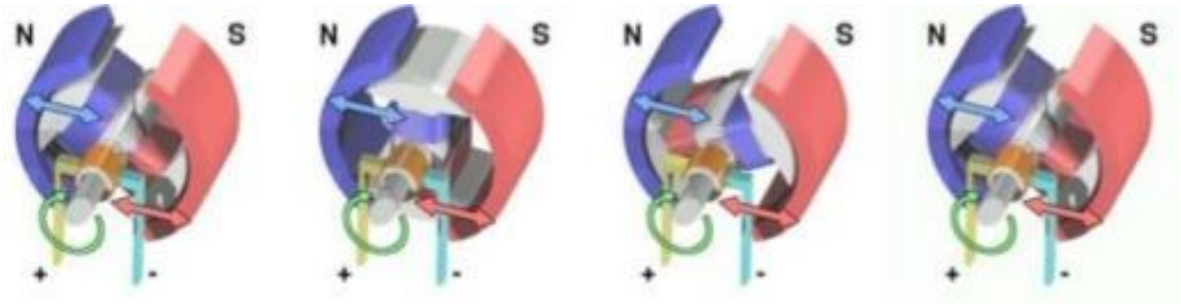


Figura 3: Funcționarea motorului de cc



1 **2** **3** **4**

1. Când bobina este alimentată, în jurul rotorului se generează un câmp magnetic, provocând o respingere între magneți, făcând motorul să se rotească în sensul acelor de ceasornic;

2. Rotorul continuă să se rotească;

3. Când rotorul se aliniază orizontal câmpul magnetic este inversat, continuând mișcarea în sensul acelor de ceasornic;

4. Procesul se repetă continuu în timp ce motorul este alimentat.

Motor pas cu pas

Motorul pas cu pas este un motor electric care se mișcă în funcție de un impuls primit printr-un controler. Numărul de pași pe care îi dă motorul este exact același cu numărul de impulsuri primite, iar viteza motorului este aceeași cu frecvența impulsurilor. Prin urmare, este un dispozitiv simplu (fără perii, fără comutatoare și fără codificatoare). Sunt motoare utilizate în mai multe domenii electronice și automatizări, cum ar fi robotica, mișcarea axelor carteziene, printre altele.

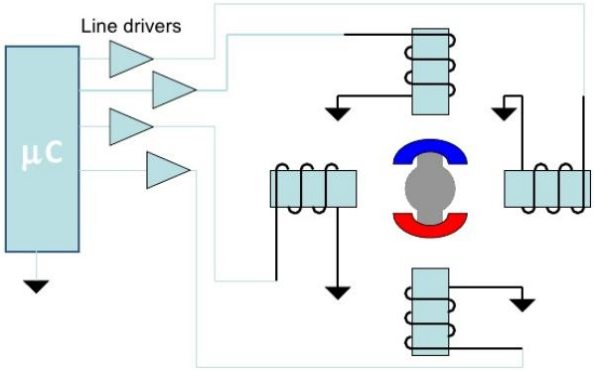


Figura 4: Diagrama simplă a motorului pas cu pas

Figura 5: Moto pas cu pas

Exemple de motoare pas cu pas

Motor unipolar

Un motor unipolar are două bobine pe fază, câte una pentru fiecare direcție a curentului electric. In aceasta configuratie un pol magnetic poate fi inversat fara a comuta directia curentului electric, circuitul comutatorului se poate face foarte simplu pentru fiecare bobina.

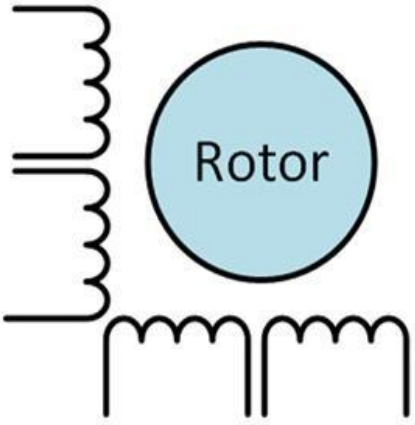


Figura 6: Motor unipolar

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

Tabelul 1: Tabel de stare

Motor Bipolar

Motoarele bipolare au o bobină unică după fază. Curentul electric dintr-o bobină trebuie inversat pentru a inversa un pol magnetic. Deci circuitul electric este puțin mai complicat, solicitând necesitatea unei punți în H. Există două conexiuni pe fază și niciuna nu este în comun.

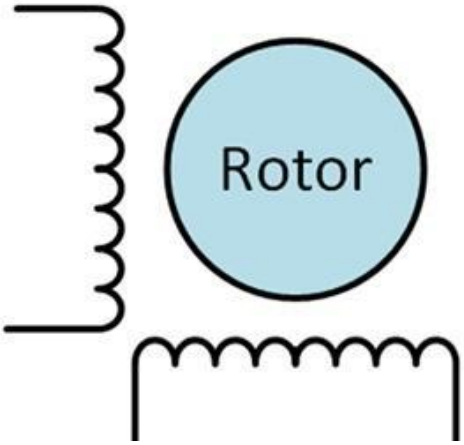


Figura 7: Motor Bipolar

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

Tabelul 2: Tabel de stare

Funcționarea motorului pas cu pas

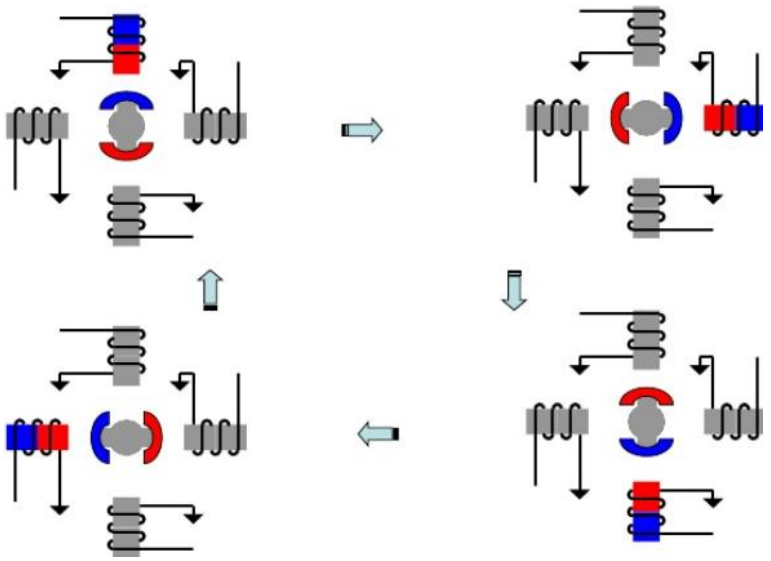


Figura 8: Schema motorului pas cu pas

Servomotor

Un servomotor este un dispozitiv electromecanic care, printr-un semnal electric, plasează axa într-o poziție unghiulară. În mod normal, acest tip de motor este compact, permițând o poziție precisă a propriei axe. În prezent, servomotoarele sunt utilizate în robotică și modelare.

Figura 9: Servomotor

Operații:

Un servomotor poate fi împărțit în patru părți:

● Circuit de control – Responsabil pentru recepția semnalelor PWM și a energiei. Controlează poziția potențiometrului și controlează motorul în funcție de semnalul PWM primit.
● **Potențiometru** – Este conectat la axa de ieșire servo, poziționându-l.
● **Motor** – Mișcă angrenajul și axa principală a servo-ului.
● **Roți dințate de antrenare** – Reduce rotația motorului, scăzând rezistența astfel încât acesta să aplice un cuplu superior pe axa principală. Ele mișcă potențiometrul împreună cu axa.

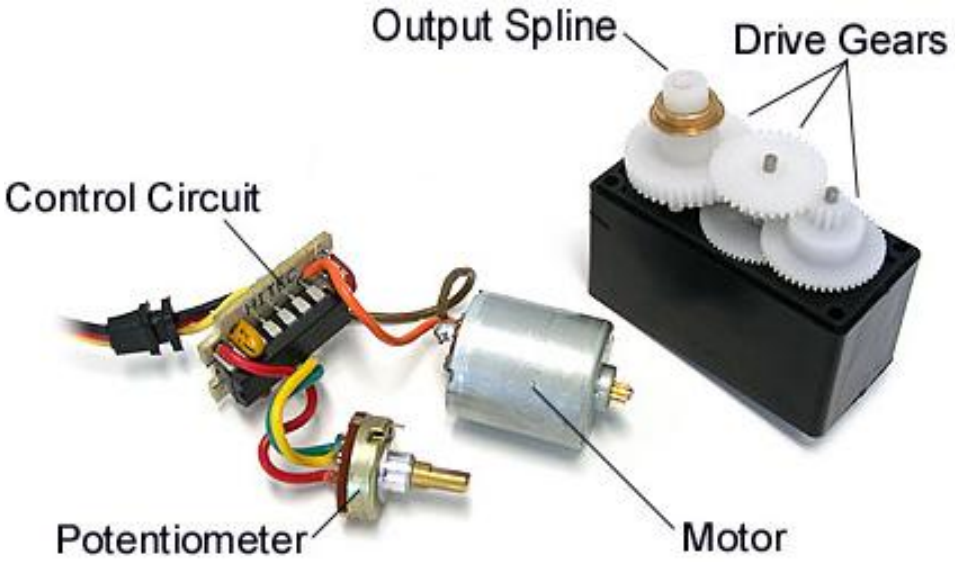


Figura 10: Părțile servomotorului

Poziția servomotorului

Controlul unui servomotor se obține printr-un semnal de intrare care prezintă niveluri de tensiune TTL care specifică poziția acestuia. Acest format de semnal urmează modulația PWM (Pulse Width Modulation) așa cum este prezentată în imaginea 11. Codificarea în PWM se face prin impuls de nivel înalt în raport cu perioada totală de oscilație, care în cazul servomotoarelor este de 20ms.

În acest caz specific, dacă în 20 de milisecunde 1 milisecundă este la nivel înalt, servomotorul ar trebui să fie în poziția minimă, așa cum se arată în figura 11.

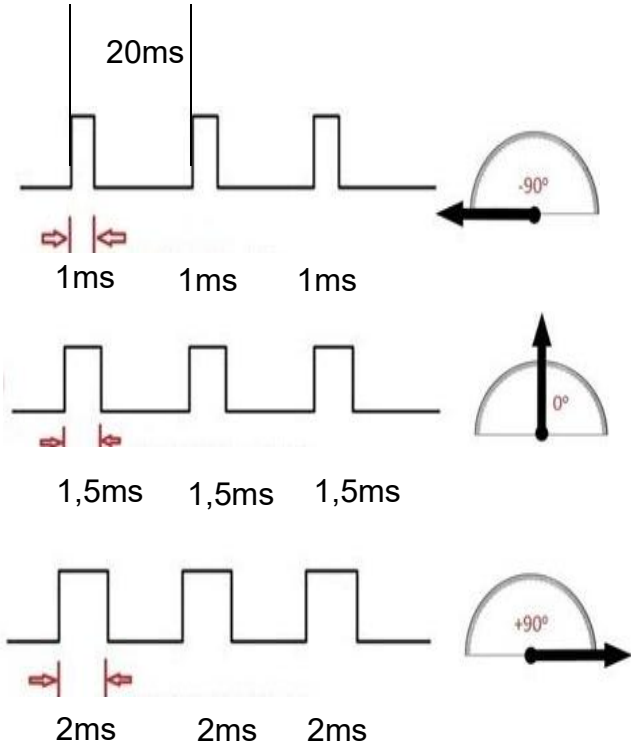


Figura 11: Servomotor PWM

PWM (Modularea lățimii pulsului)

PWM poate fi implementat în mai multe domenii ale electronicii. Una dintre utilizările sale este în alimentarea cu energie, controlul vitezei motoarelor DC, controlul luminii, controlul servomotoarelor și alte câteva aplicații. Prin PWM putem controla viteza și puterea.

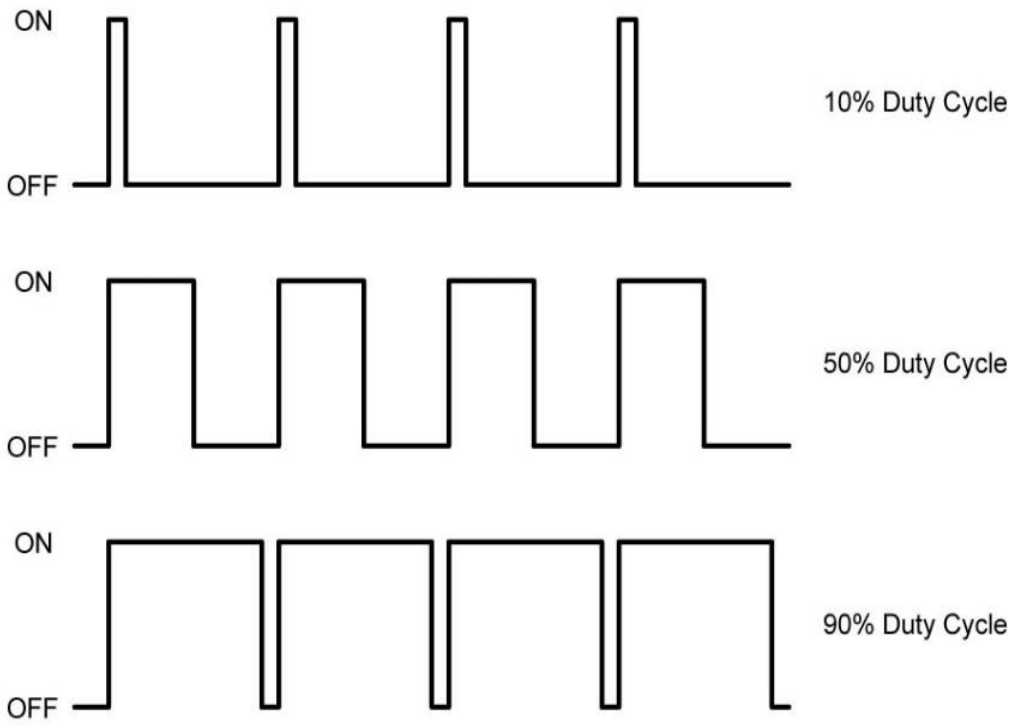


Figura 12: PWM

Operațiune PWM

Având în vedere o undă pătrată, pentru a obține funcționarea corectă a PWM trebuie să variem lățimea impulsului undei. Pentru a calcula, avem nevoie de perioada și lățimea pulsului, iar rezultatul acestuia se numește duty-cycle, așa cum este definit de ecuația:

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

Duty cycle: valoare procentuală;

Pulse Width: secvență de timp în care semnalul este la nivel înalt;

Perioada: durata unui ciclu de val.

NOTĂ: Următoarele experimente vor fi făcute folosind kitul de motor Arduino, pe care elevii îl vor realiza singuri.

Numele circuitului_1 (cu DC Motor): " Motor de CC simplu"
Explicarea circuitului: Acest program demonstrează secvențele de pornire și oprire ale unui motor de curent continuu
Program:

```
// Motor de CC simplu
/*
     Connection:
Arduino   | MotorShielt
PIN       | PIN
-------------------------
+5V       | VSIGNAL
GND       | GNDSIGNAL
4         | ENA
5         | PWMA+
6         | PWMA-
*/
//  variabile
int ena = 4;    // Arduino pin 4 connected to ENA
int right = 5;  // Arduino pin 5 connected to PWMA+
int left = 6;   // Arduino pin 5 connected to PWMA-
//  setarea funcționării
void setup()
{
 pinMode(ena, OUTPUT);
 pinMode(right, OUTPUT);
 pinMode(left, OUTPUT);
 digitalWrite(ena, HIGH);  //activ ENA
 delay(900);
}
// funcționarea buclei
void loop()
{
 analogWrite(right, 255);  // modifcă sensul la dreapta cu viteză maximă
 delay(1000);
 analogWrite(right, 0);    // oprit
 delay(1000);
 analogWrite(left, 255);   // modifică sensul la stânga cu viteză maximă
 delay(1000);
 analogWrite(left, 0);     // oprit
 delay(1000);
 analogWrite(right, 127);  //modifică sensul la dreapta cu jumătate de viteză
 delay(1000);
 analogWrite(right, 0);    // oprit
 delay(1000);
 analogWrite(left, 127);   // modifică sensul la stânga cu jumătate de viteză
 delay(1000);
 analogWrite(left, 0);     // oprit
 delay(1000);
}
```

Numele circuitului_2: "Avans cu motor de CC"

Explicarea ciruitului: Acest program controlează un motor de curent continuu. Comenzile de control sunt emise prin intermediul monitorului serial.

Comenzile de control

Scrie în Monitorul Serial	*Acțiune*
stop	Motorul se oprește
go right	Motorul modifică sensul la dreapta
go left	Motorul modifică sensul la stânga
+	Crește viteza
-	Descrește viteza

Program:

// Avans cu motor de CC

```
/*
    Connection:
Arduino   | MotorShielt
PIN       | PIN
-------------------------
+5V       | VSIGNAL
GND       | GNDSIGNAL
4         | ENA
5         | PWMA+
6         | PWMA-
*/
//-------------------------------------------------------------
//  variables
//-------------------------------------------------------------
int en_a = 4;              //ArduPin4 conectat la activ Pin
int right_a = 5;           //ArduPin5 conectat la PWMA+ Pin
int left_a = 6;            //ArduPin6 conectat la PWMA- Pin
String message = "";       //păstrează mesajul intefeței seriale
int val = 80;              //definește valoarea pentru viteză
int del = 10;              //valoarea crește și scade la comanda "+" and "-"
int minimumVal = 70;       //viteza minimă / valoare
int maximumVal = 250;      //viteza maximă / valoare
bool flag_go_right = false; //devine adevărată când "go right" este emisă
bool flag_go_left = false;  //devine adevărată când comanda "go left" este emisă

//-------------------------------------------------------------
//  setare funcționare
//-------------------------------------------------------------
void 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);
 delay(900);
}
//-------------------------------------------------------------
// loop Function
//-------------------------------------------------------------
void loop()
{
 if(Serial.available()>0)
  {
   message = Serial.readString();
   Serial.println(" --> Incoming message: " + message ); //Vedeți mesajul primit

   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(" <-- Outgoing message: stop \n");
   }
   else if(message.equals("go right\n"))  //verifică mesajul dacă "go right"
   {
    digitalWrite(en_a, HIGH);         //setează activ Pin of the Modul to HIGH
          analogWrite(right_a, val);        //scrie valoarea lui PWM Pin la rotirea spre dreapta
    analogWrite(left_a, 0);           //scrie zero la PWM Pin la rotirea spre stânga
    flag_go_left = false;             //set the flags
    flag_go_right = true;
    Serial.println(" <-- Outgoing message: go right\n");  //feedback
   }
   else if(message.equals("go left\n"))
   {
    digitalWrite(en_a, HIGH);
          analogWrite(right_a, 0);
    analogWrite(left_a, val);
    flag_go_right = false;
    flag_go_left = true;
    Serial.println(" <-- Outgoing message: go left\n");
   }
   else if(message.equals("+\n"))      //verifică mesajul dacă "+"
   {
    if(val >= maximumVal)          // atinge valoarea maximă
```

```
    {
      Serial.println(" <-- Outgoing message: Maximum value! \n");
    }
    else if (val < maximumVal)
    {
      val = val + del;    //crește valoarea
      if(flag_go_right)   // verificați în ce direcție ar trebui să creșteți
      {
        analogWrite(right_a, val);
      }
      else if(flag_go_left)
      {
        analogWrite(left_a, val);
      }
      Serial.print(" <-- Outgoing message: value = "); Serial.println(val); Serial.println("");
//feedback
    }
  }
  else if(message.equals("-\n"))
  {
    if(val <= minimumVal)                    // atinge viteza minimă
    {
      Serial.println(" <-- Outgoing message: Minimum value! \n");
    }
    else if (val > minimumVal)
    {
      val = val - del;
      if(flag_go_right)
      {
        analogWrite(right_a, val);
      }
      else if(flag_go_left)
      {
        analogWrite(left_a, val);
      }
      Serial.print(" <-- Outgoing message: value = "); Serial.println(val); Serial.println("");
    }
  }
  else //generează mesajul dacă este necunoscut
  {
    Serial.println("<<-- Incoming message " + message + "      IS UNKNOWN \n\n");
  }
 }
}
```

Numele circuitului_3: "Două motoare de CC înainte"
Explicația circuitului: Acest program controlează două motoare de curent continuu. Comenzile de control sunt emise prin intermediul monitorului serial.

Comezile de control:

Scrie în Monitorul Serial	*Acțiune*
stop	Motorul se oprește
stop a	Oprește motorul a
stop b	Oprește motorul b
go right a	Motorul a se rotește în dreapta
go left a	Motorul a se rotește în stânga
go right b	Motorul b se rotește în dreapta
go left b	Motorul b se rotește în stânga
+	Crește viteza
-	Scade viteza

Program:

```
// Două motoare de CC înainte /*
Conexiuni:
Arduino  | MotorShielt
PIN      | PIN
-------------------------
+5V      | VSIGNAL
GND      | GND
4        | ENA
5        | PWMA+
6        | PWMA-
8        | ENB
9        | PWMB+
10       | PWMB-
*/
//-------------------------------------------------------------
// variabile
//-------------------------------------------------------------
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;

//-------------------------------------------------------------
// setare funcții
//-------------------------------------------------------------
void 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);

 delay(900);
}
//-------------------------------------------------------------
// funcționare buclă
//-------------------------------------------------------------
void loop()
{
 if(Serial.available()>0)
 {
  message = Serial.readString();
  Serial.println(" --> Incoming message: " + message);

  if(message.equals("stop\n"))                    // amândouă motoarele se opresc
  {
   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(" <-- Outgoing message: stop all\n");
}
else if(message.equals("stop a\n"))                    // motorul a se oprește
{
  digitalWrite(en_a, LOW);
  analogWrite(right_a, 0);
  analogWrite(left_a, 0);
  flag_go_right_a = false;
  flag_go_left_a = false;
  Serial.println(" <-- Outgoing message: stop a\n");
}
else if(message.equals("stop b\n"))                    // motorul b se oprește
{
  digitalWrite(en_b, LOW);
  analogWrite(right_b, 0);
  analogWrite(left_b, 0);
  flag_go_right_b = false;
  flag_go_left_b = false;
  Serial.println(" <-- Outgoing message: stop b\n");
 }
else if(message.equals("go right a\n"))
{
  digitalWrite(en_a, HIGH);
        analogWrite(right_a, val);
  analogWrite(left_a, 0);
  flag_go_left_a = false;
  flag_go_right_a = true;
  Serial.println(" <-- Outgoing message: go right a\n");
}
else if(message.equals("go right 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(" <-- Outgoing message: go right b\n");
}
else if(message.equals("go left a\n"))
{
  digitalWrite(en_a, HIGH);
        analogWrite(right_a, 0);
  analogWrite(left_a, val);
  flag_go_right_a = false;
  flag_go_left_a = true;
```

```
    Serial.println(" <-- Outgoing message: go left a\n");
   }
   else if(message.equals("go left 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(" <-- Outgoing message: go left b \n");
   }
   else if(message.equals("+\n"))
   {
    if(val >= maximumVal)                    // atinge valoarea maximă
    {
     Serial.println(" <-- Outgoing message: Maximum value! \n");
    }
    else if (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(" <-- Outgoing message: value = "); Serial.println(val); Serial.println("");
    }
   }
   else if(message.equals("-\n"))
   {
    if(val <= minimumVal)                    // atinge valoarea minimă
    {
     Serial.println(" <-- Outgoing message: Minimum value! \n");
    }
    else if (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(" <-- Outgoing message: value = "); Serial.println(val); Serial.println("");
    }
   }
   else
   {
    Serial.println("  <<-- Incoming message " + message + "      IS UNKNOWN \n\n");
   }
  }
}
```

Numele circuitului_4: "Program motor pas simplu"
Explicarea circuitului: "Un pas Motorul se rotește în sensul acelor de ceasornic și în sens invers acelor de ceasornic
Program:

```
// Program motor pas simplu /*
   Conexiuni:
Arduino  | MotorShielt
PIN      | PIN
------------------------
+5V      | VSIGNAL
GND      | GNDSIGNAL
4        | ENA
5        | PWMA+
6        | PWMA-
8        | ENB
9        | PWMB+
10       | PWMB-
*/
//-------------------------------------------------------------
//  librarys
//-------------------------------------------------------------
#include <Stepper.h>
// libraty for the stepper
//-------------------------------------------------------------
//  constants
//-------------------------------------------------------------
#define STEPS 200
// Step Angle of used stepmotor = 1,8 degrees
// 360 / 1,8° = 200
//-------------------------------------------------------------
//  class
//-------------------------------------------------------------
Stepper Motor(STEPS, 5,6,9,10);
  //create a new instance of the Stepper class
  //Parameter:
  //  STEPS -> the number fo steps in one revolution of the Motor
  //  5,6,9,10 -> used Pins
//-------------------------------------------------------------
// variables
//-------------------------------------------------------------
int spe = 50;

//-------------------------------------------------------------
// setup function
//-------------------------------------------------------------
void setup()
{
  Serial.begin(9600);
  pinMode(4, INPUT);
```

```
 pinMode(8, INPUT);
 digitalWrite(4 ,HIGH);    //activate ENA Pin of the MotorShielt
 digitalWrite(8 ,HIGH);    //activate ENB Pin of the MotorShielt

 Motor.setSpeed(spe);  //Object Motor get the speed in "rotation per minute"
}

void loop()
{
 Motor.step(100);    //Motorul se întoarce unu anumit număr de pași,
                     //la o viteză determinate de cel mai recent apel către setSpeed();
                     //în acest caz 100 de pași în sensul acelor de ceasornic;
 delay(200);

 Motor.step(-50);    //50 de pași în sens invers acelor de ceasornic
 delay(200);
}
```

Numele circuitului_5: "Program avansat motor pas cu pas"

Explicația circuitului: Acest program controlează un motoare pas. Comenzile de control sunt emise prin intermediul monitorului serial.

Exemplu: Trimiteți 100 pe monitorul serial către Arduino Motorul pas cu pas face 100 de pași. Dacă trimiteți -100 Motorul pas cu pas face 100 de pași în altă direcție.

Program:

```
// Program avansat motor pas cu pas
/*
   Conexiuni:
Arduino  | MotorShielt
PIN      | PIN
-------------------------
+5V      | VSIGNAL
GND      | GNDSIGNAL
4        | ENA
5        | PWMA+
6        | PWMA-
8        | ENB
9        | PWMB+
10       | PWMB-
*/

//-------------------------------------------------------------
//  librarys
//-------------------------------------------------------------
#include <Stepper.h>
//-------------------------------------------------------------
//  constants
//-------------------------------------------------------------
#define STEPS 200
//-------------------------------------------------------------
//  class
//-------------------------------------------------------------
Stepper Motor(STEPS, 5,6,9,10);
//-------------------------------------------------------------
// variables
//-------------------------------------------------------------

int spe = 100;
String message = "";

//-------------------------------------------------------------
// setup function
//-------------------------------------------------------------

void setup()
```

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

void loop()
{
 if(Serial.available()>0)
  {
   message = Serial.readString();
   Serial.println("\n\n --> Incoming message: " + message );

   if(message.toInt())
   {
    Serial.print(" <-- Outgoing message:  Steps -> " + message);
    Motor.step(message.toInt());
   }
   else
   {
    Serial.println("  <<-- Incoming message " + message + "  !!! -->  is not a number <-- !!!
\n\n");
   }
  }

}
```

Numele circuitului_6 (cu Servo Motor): ” Program simplu Servomotor ”

Explicația circuitului: În acest program servomotorul este controlat de semnale date de PWM. Bucla for este folosită

// Program simplu Srvomotor

```
/*
   Conexiuni:
Arduino  | MotorShielt
PIN      | PIN
-------------------------
+5V      | VSIGNAL
GND      | GNDSIGNAL
3        | PWMSERVOIN
*/
//-----------------------------------------------------------
// variables
//-----------------------------------------------------------
int del = 100;  // time for delay

//-----------------------------------------------------------
// setup function
//-----------------------------------------------------------
void setup()
{
 Serial.begin(9600);
 pinMode(3, OUTPUT);
}

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

  delay(1000);
}
```

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

Erasmus+ KA-202

Proiecte de parteneriate strategice pentru educație și formare profesională

Titlul proiectului: "Predarea și învățarea ARDUINO în formarea profesională"

Acronimul proiectului: " ARDUinVET "

Numărul proiectului: "2020-1-TR01-KA202-093762"

Senzori

Modul și Kit de instruire

Senzori

Un senzor este un dispozitiv, modul sau subsistem electronic, al cărui scop este de a crea o interfață între un circuit și mediul său. Este sistemul care primește informații din mediul înconjurător, lumea fizică și trimite aceste informații ca valoare către circuit. De fapt, detectează evenimente sau schimbări care au loc în mediu. Pentru a detecta aceste evenimente sau modificări, senzorul trebuie să măsoare anumite valori pe o scară fizică. Măsoară o proprietate, cum ar fi presiunea, poziția, temperatura sau accelerația și răspunde cu feedback. Apoi convertește aceste valori într-un semnal electric – de obicei.

Deci, un senzor este întotdeauna folosit cu alte electronice. Senzorii pot fi găsiți în multe obiecte de zi cu zi, cum ar fi dispozitive sensibile la atingere sau la mișcare, dispozitive operate de lumină sau sunet etc. Utilizarea senzorilor s-a extins dincolo de câmpurile tradiționale de măsurare a temperaturii, presiunii sau debitului, de ex. Senzori GPS. Senzorii analogici sunt încă folosiți pe scară largă. Aplicațiile includ producție și mașini, avioane și aerospațiale, mașini, medicină, robotică și multe alte aspecte ale vieții noastre de zi cu zi.

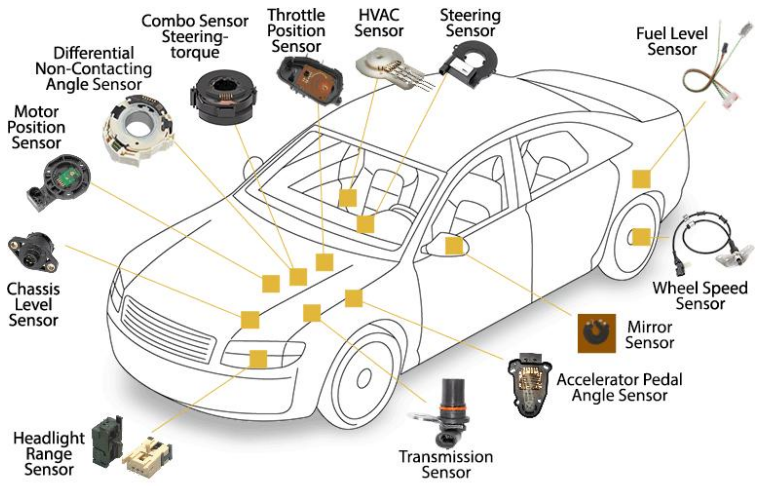


Figura1. Senzorii unui automobil

Caracteristicile senzorilor

Un senzor bun respectă următoarele reguli:

-trebuie sa fie sensibil la proprietatea masurata

-trebuie să fie insensibil la orice altă proprietate susceptibilă de a fi întâlnită în aplicarea sa

-nu trebuie sa influenteze proprietatea masurata.

Principalele caracteristici de bază ale senzorilor sunt:

-**Liniaritate.** Senzorul are o proprietate sau o caracteristică, a cărei valoare se modifică. Când se modifică și mărimea fizică pe care o măsoară. Este de dorit ca variațiile sale în măsurarea mărimii fizice să provoace modificări apreciabile în proprietatea senzorului. Această proprietate se numește liniaritate și are o importanță principală.

-**Precizie.** Apropierea valorii de ieșire de valoarea de intrare.

-**Eroare**: Diferența dintre valoarea măsurată și valoarea reală.

-**Toleranță**: eroarea maximă pe care o poate genera senzorul.

-Full - Scale Input (FSI): Specifică în ce cadre cu dimensiunea fizică măsurată poate fi utilizat senzorul

-Full - Scale Output (FSO): Setează valorile pe care tensiunea sau curentul le poate primi la ieșirea senzorului

-Sensibilitate: Exprimă cât de mare iese semnalul de ieșire la senzor pentru fiecare unitate de dimensiune fizică măsurată

-**Rezoluție**: Exprimă cea mai mică modificare a dimensiunii fizice pe care o poate detecta senzorul și, în consecință, își modifică valoarea de ieșire

-Histerezis: O eroare de histerezis face ca valoarea de ieșire să varieze în funcție de valorile anterioare de intrare. Dacă ieșirea unui senzor este diferită, în funcție de faptul dacă o anumită valoare de intrare a fost atinsă prin creșterea sau scăderea intrării, atunci senzorul are o eroare de histerezis

- Întârziere: Este întârzierea modificării valorii de ieșire după o schimbare de intrare.

-Zona moartă: cantitatea maximă de modificare a valorii de intrare care nu afectează valoarea de ieșire.

Clasificarea senzorilor

Există mai multe moduri de clasificare a senzorilor, dintre care unele sunt enumerate mai jos.

Primul clasifică senzorii în funcție de forma valorii de ieșire, împărțind senzorii analogici și digitali.

A doua categorizare se referă la ceea ce poate măsura un senzor, cu o distincție mai semnificativă între senzorii naturali și cei chimici. Senzorii naturali controlează dimensiunile fizice, cum ar fi locația, masa, curentul, timpul și dimensiunile lor relative, în timp ce senzorii chimici controlează prezența diferitelor gaze într-o anumită atmosferă.

O altă modalitate se referă la materialele ale căror proprietăți fizice funcționează senzorul, cu principalele categorii de senzori cu materiale conductoare, semiconductoare, dielectrice, magnetice și supraconductoare.

În sfârșit, o altă metodă de clasificare se referă la domeniul de utilizare al senzorului, cu categorii majore precum senzori industriali, medicali, militari, de mediu, precum și senzori pentru aplicații de transport și automatizare.

Figura 2: Tipuri de senzori

Senzori de bază

Iată câțiva senzori de bază pentru aplicații comune:

Foto Rezistor LDR: Rezistorul este un rezistor variabil a cărui valoare se modifică în funcție de lumina care cade asupra lui.

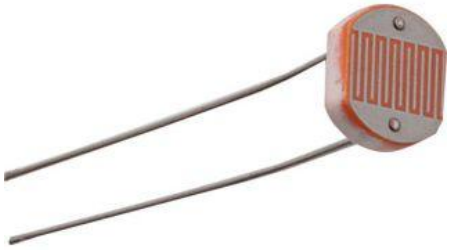


Figura 3: Fotorezistor

Senzor digital de luminozitate / lux / lumină: Senzorul de luminozitate TSL2561 este un senzor de lumină digital avansat, ideal pentru utilizare într-o gamă largă de situații de lumină. Acest senzor este mai precis, permițând calcule exacte de lux și poate fi configurat pentru diferite intervale de câștig/temporizare pentru a detecta intervale de lumină de la 0,1 la 40.000+ Lux din mers. Conține atât diode în infraroșu, cât și diode cu spectru complet! Aceasta înseamnă că puteți măsura separat lumina infraroșu, cu spectru complet sau lumina vizibilă pentru om.

Figura3: Senzor digital de luminozitate / lux / lumină

Senzor de temperatură (cu ieșire analogică): intervalul de temperatură pe care îl măsoară este de obicei de la -40 ° C la + 100 ° C, cu o precizie de ± 1 ° C.

Figura 4: Senzor de temperatură

Senzor de umiditate și temperatură: senzor digital de temperatură și umiditate foarte precis. Dispune de un interval de măsurare 0-100% RH, cu o precizie a temperaturii de +/- 0,3°C @ 25°C.

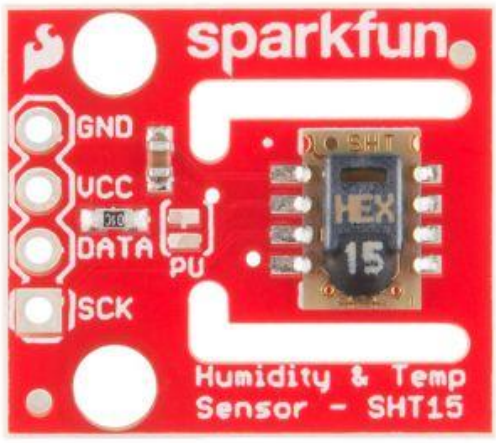


Figura 5: Senzor de umiditate și temperatură

Senzor infraroșu: Folosit pentru calcularea distanței. Distanța pe care o puteți calcula este de la 2 cm. pana la 400 cm. cu o precizie de un centimetru.

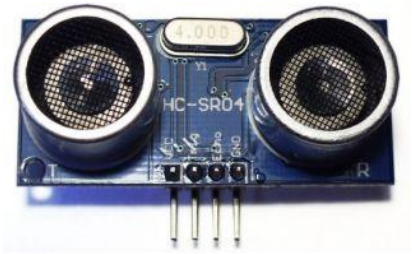


Figura 6: Senzor cu infraroșu

Senzor de mișcare: are capacitatea de a detecta mișcarea unui om sau a unui animal de companie într-o cameră pe o rază de șase metri. Senzorul are două rezistențe trimmer unde se pot regla sensibilitatea și timpul de activare din momentul în care detectează mișcare.

Figura 7: Senzor de mișcare

Senzor de vibrații: Când senzorul detectează vibrații, se generează o tensiune, folosind o rezistență de 1 Mohm

Figura 8: Senzor de vibrații

Accelerometru cu trei axe și senzor de deblocare a giroscopului: prin combinarea unui giroscop cu 3 axe și a unui accelerometru cu 3 axe pe aceeași matriță de siliciu împreună cu un procesor digital de mișcare (DMP) integrat, capabil să proceseze algoritmi complexi de fuziune de mișcare pe 9 axe, senzorul poate returnează toate valorile de aliniere a axelor transversale.

Figura 9: Accelerometru cu trei axe și senzor de deblocare a giroscopului

Senzor de gaz: sensibil pentru GPL, gaz natural, gaz de cărbune. Tensiunea de ieșire crește odată cu creșterea concentrației gazelor măsurate.

Figura 11: Senzor de gaz

Detector de sunet: Acest senzor oferă o ieșire audio, dar și o indicație binară a prezenței sunetului și o reprezentare analogică a amplitudinii acestuia.

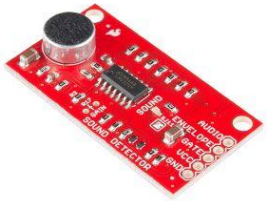


Figura 12: Detectopr de sunet

PRIECT1- NUME: SENSOR ULTRASONIC PENTRU LUMINILE DE TRAFIC

Scopul proiectului: În acest proiect, elevii vor vedea cum putem folosi un senzor ultrasonic în semnalul de trafic folosind o rezistență externă de tragere.

Prin acest proiect, elevii vor putea experimenta cu un senzor ultrasonic, un dispozitiv care măsoară distanța până la un obiect folosind unde sonore.

Metodologia

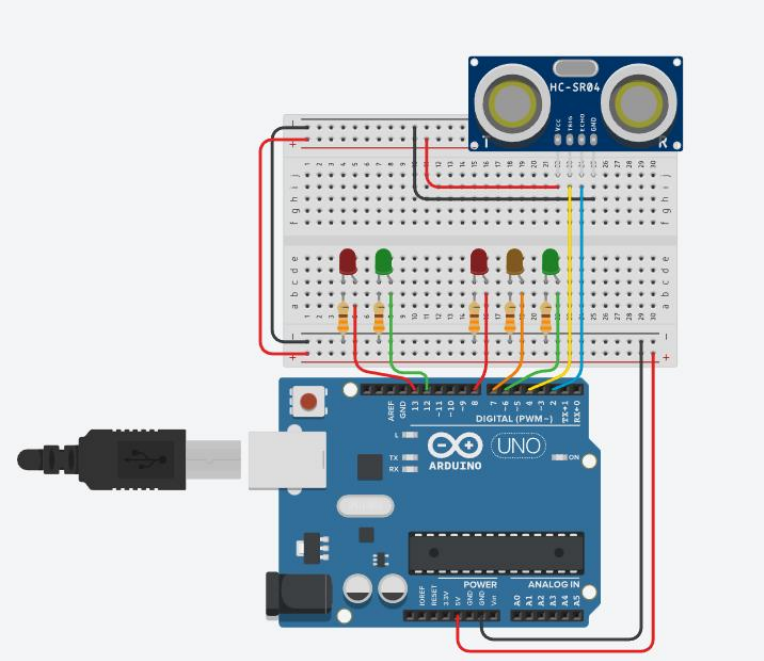
Inițial, descriem fluxul de lucru dorit al proiectului. Apoi le cerem elevilor să selecteze componentele necesare și să utilizeze Tinkercad pentru a proiecta și a desena circuitul. Ei vor folosi Tinkercad Code Tool pentru a scrie codul.

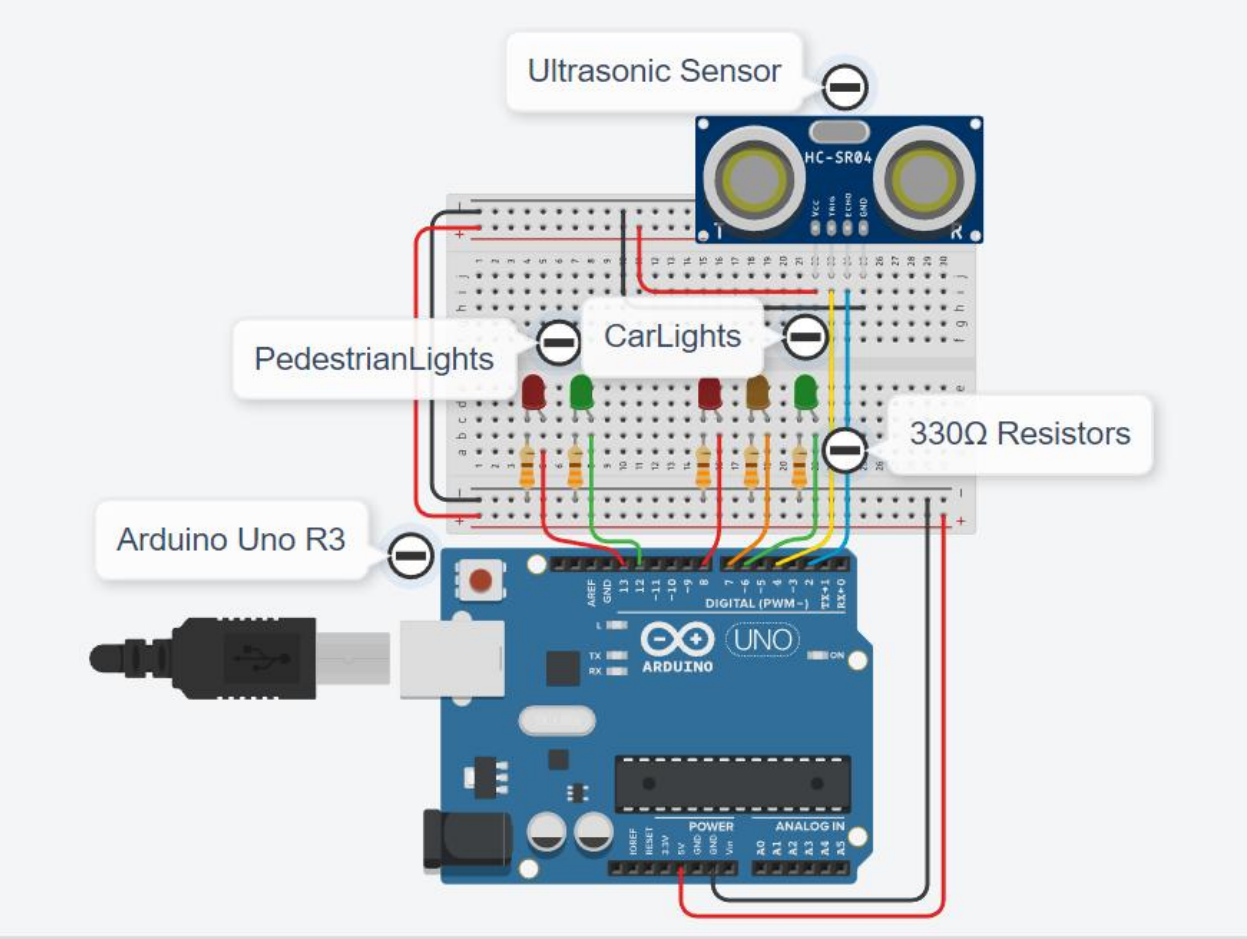
Apoi vor crea circuitul, vor scrie codul pe instrumentul software Arduino și l-au încărcat pe placa Arduino.

Simularea pe Tinkercad va verifica dacă circuitul a fost construit corect.

Acest proiect este pentru studenții de nivel de bază, care încep să beneficieze de rezultatele și rezultatele din proiectul nostru Erasmus+ KA-202 Parteneriate strategice în VET, numit „ArduinVET”.

Circuitul proiectului





Cum lucrează:

Senzorul cu ultrasunete funcționează trimițând o undă sonoră la o frecvență ultrasonică și așteptând ca aceasta să revină de pe obiect. Timpul de întârziere dintre transmisia sunetului și recepția sunetului este apoi utilizat pentru a calcula distanța.

Elevii pot activa senzorul punând mâna în fața senzorului, ceea ce duce la scăderea întârzierii între transmisia sunetului și recepția sunetului. Distanța va fi afișată pe ecranul serial. Elevii pot vedea activarea semafoarelor observând luminile.

În circuit, împreună cu senzorul cu ultrasunete, folosim un rezistor extern pull-up.

- Rezistorul pull-up menține pinul 3 permanent în stare HIGH (+5V).
- Când senzorul cu ultrasunete este activat, pinul 3 este împământat (stare LOW) momentan.

Tabelul de timpi

Faza	Vehicule	Pietoni	Timp	Descriere
0				Senzorul nu este activat
1			3sec	Odată activat senzorul, se activează fazele de la 1 la 4 și apoi dispozitivul revine la starea anterioară (Faza 0)
2			3sec	
3			4sec	
4			3sec	
0				Până când senzorul este activat din nou

Lista componenetelor:

Componentele noastre sunt:

- Placă Arduino
- Senzor ultrasonic HC-SR04
- Breadboard și Jump Wire
- Cablu USB
- Arduino UNO R3
- 5 LED uri
- 5 rezistențe de 330 ohm

Programul:

```
//Nume proiect: SENSOR ULTRASONIC PENTRU LUMINILE DE TRAFIC
// Semnal rutier cu lumină pentru pietoni și senzor ultrasonic
// Declararea constantelor
  const byte greenCar = 6;
  const byte orangeCar = 7;
  const byte redCar = 8;
  const byte greenPedestrian = 12;
  const byte redPedestrian = 13;
//  Declararea parametrilor
  boolean buttonOn=false;
// Parameter declaration Ultrasonic Sensor HC-SR04 * ******
// definirea numărului pinilor
  const int trigPin = 4;  //D4
  const int echoPin = 2;  //D3


void setup() {
```

```
 // Initialize constants Ultrasonic Sensor HC-SR04 * ******
  pinMode(trigPin, OUTPUT); // Setare trigPin ca ieșire
  pinMode(echoPin, INPUT); // Setare echoPin ca intrare
  Serial.begin(9600); // Începe comunicarea serială
 // inițializarea parametrilor
  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);
// Calcularea distanței
  const long distance= duration/58.2;
//Afișarea distanței pe monitorul serial
  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);
    }
}
```

PROIECT 2 - NUME: SISTEM DE ALARMĂ

Scopul proiectului: În acest proiect, elevii vor vedea cum funcționează un sistem de alarmă care poate detecta gazul lichid, mișcarea într-un spațiu închis, precum și creșterea nejustificată a temperaturii. Sistemul de alarma va fi controlat de un buton si un senzor fotorezistor. Prin apăsarea butonului elevii pot activa sistemul de alarmă. Când soneria sună prin apăsarea butonului, ei îl pot opri. Atunci când asociază un senzor activat cu apăsarea butonului, pot dezactiva sistemul de alarmă. Fotorezistorul activează sistemul de alarmă în cazul în care sistemul de alarmă nu este activat și iluminarea este scăzută.
Prin acest proiect, elevii ar putea experimenta cu:

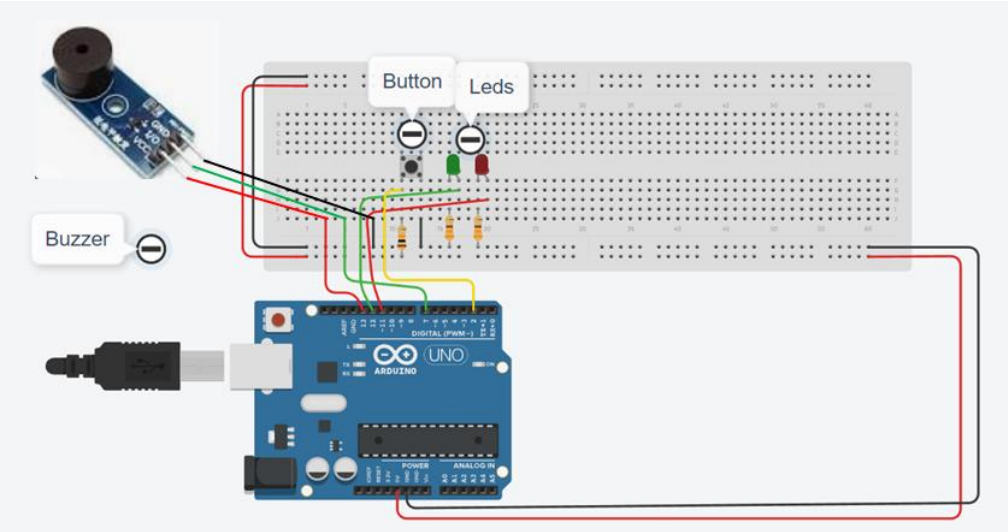
- un modul senzor de sonerie activă care are încorporat un circuit oscilator care produce o frecvență a sunetului constantă. Se pornește și se oprește cu un pin Arduino
- un senzor ultrasonic, un dispozitiv care măsoară distanța până la un obiect folosind unde sonore
- un senzor de gaz și două lumini indicatoare și un buton.
- un senzor fotorezistor
- un senzor de temperature

Metoda

În primul rând, descriem fluxul de lucru dorit al proiectului care urmează să fie dezvoltat în pași. Elevii sunt rugați apoi să selecteze componentele necesare pentru proiectarea și desenarea circuitului. Ei vor folosi aplicația „Arduino” pentru a scrie codul.
Apoi vor crea circuitul, vor scrie codul pe instrumentul software Arduino și l-au încărcat pe placa Arduino.
Elevii vor verifica dacă circuitul a fost construit corect.
Acest proiect este pentru studenții de nivel de bază, care încep să beneficieze de rezultatele și rezultatele din proiectul nostru Erasmus+ KA-202 Parteneriate strategice în VET, numit „ArduinVET”.

Pasul 1 (Buzzer cu diferite tonuri în sistemul de alarmă)

Circuitul proietului



Cum funcționează:

O unitate de senzor activă a soneriei are un circuit oscilant încorporat, astfel încât frecvența sunetului este constantă. Este capabil să producă sunetul în sine.
În cadrul acestei lucrări, tonul sunetului încetează să mai aibă același sunet prin cod. Imediat ce butonul este apăsat, începe să sune cu un pin Arduino, led-ul roșu care așteaptă se stinge și led-ul verde se aprinde

În circuit, împreună cu senzorul cu ultrasunete, folosim un rezistor extern pull-up.

- Rezistorul pull-up menține pinul 3 permanent în stare HIGH (+5V).
- Când butonul este apăsat, pinul 3 este împământat (stare LOW) momentan.

Lista componentelor:

- Alarmă
- Breadboard and Jump Wires
- Cablu USB
- Arduino UNO R3
- 2 LED-uri
- 2 rezistențe de 330 ohm
- Buton
- 1 rezistență de 10Kohm

Programul Arduino pentru pasul 1:

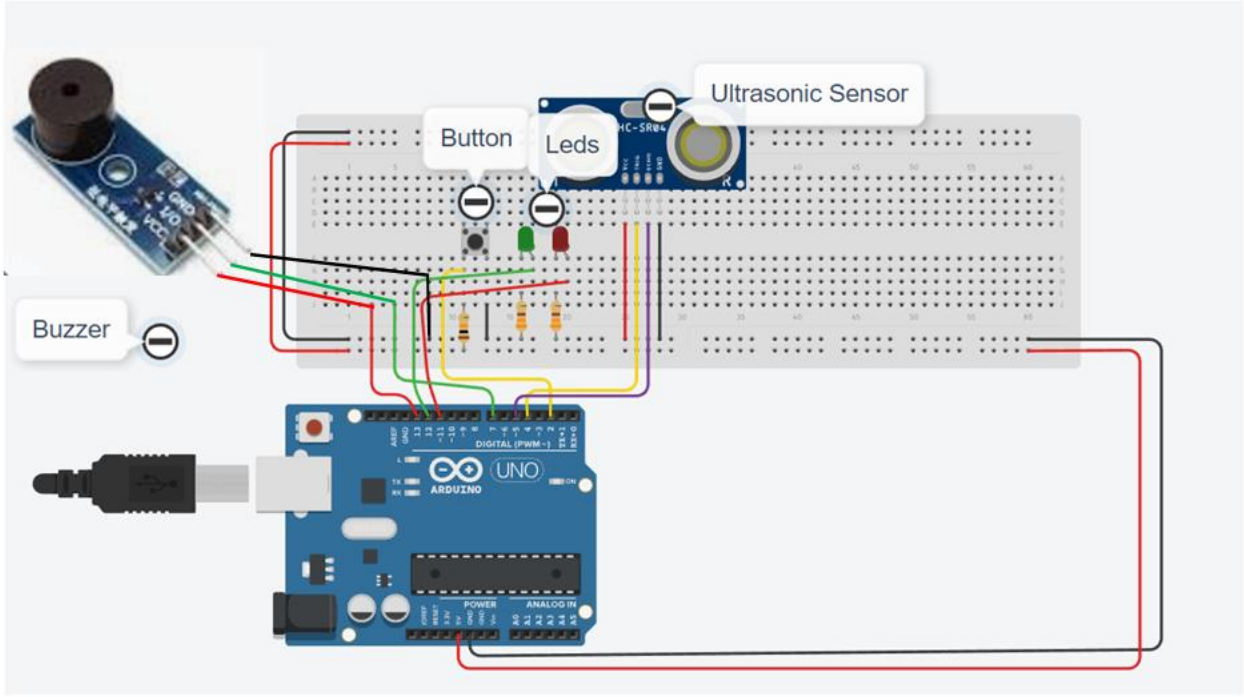
```
//Nume proiect: Sistem de alarmă

// Pasul 1  (Alarmă cu diferite sunete)

//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);
  }
}
```

Pasul 2 (A fost adăugat un sensor ultrasonic HC-SR04 în sistemul de alarmă)

Conectarea senzorului ultrasonic HC-SR04



Cum funcționează:

Senzorul cu ultrasunete funcționează trimițând o undă sonoră la o frecvență ultrasonică și așteaptă să sară de pe obiect. Timpul de întârziere dintre transmisia sunetului și recepția sunetului este apoi utilizat pentru a calcula distanța. Elevii pot activa sistemul de alarmă apăsând butonul, apoi ledul verde se va aprinde, iar dacă își pun mâna în fața senzorului, întârzierea dintre transmisia sunetului și recepția sunetului va scădea și ledul verde se va stinge, aprinde ledul roșu și soneria audio va începe să sune. Distanța va fi afișată pe ecranul serial. De asemenea, studenții pot dezactiva sistemul de alarmă apăsând din nou butonul.

Programul Arduino pentru pasul 2:

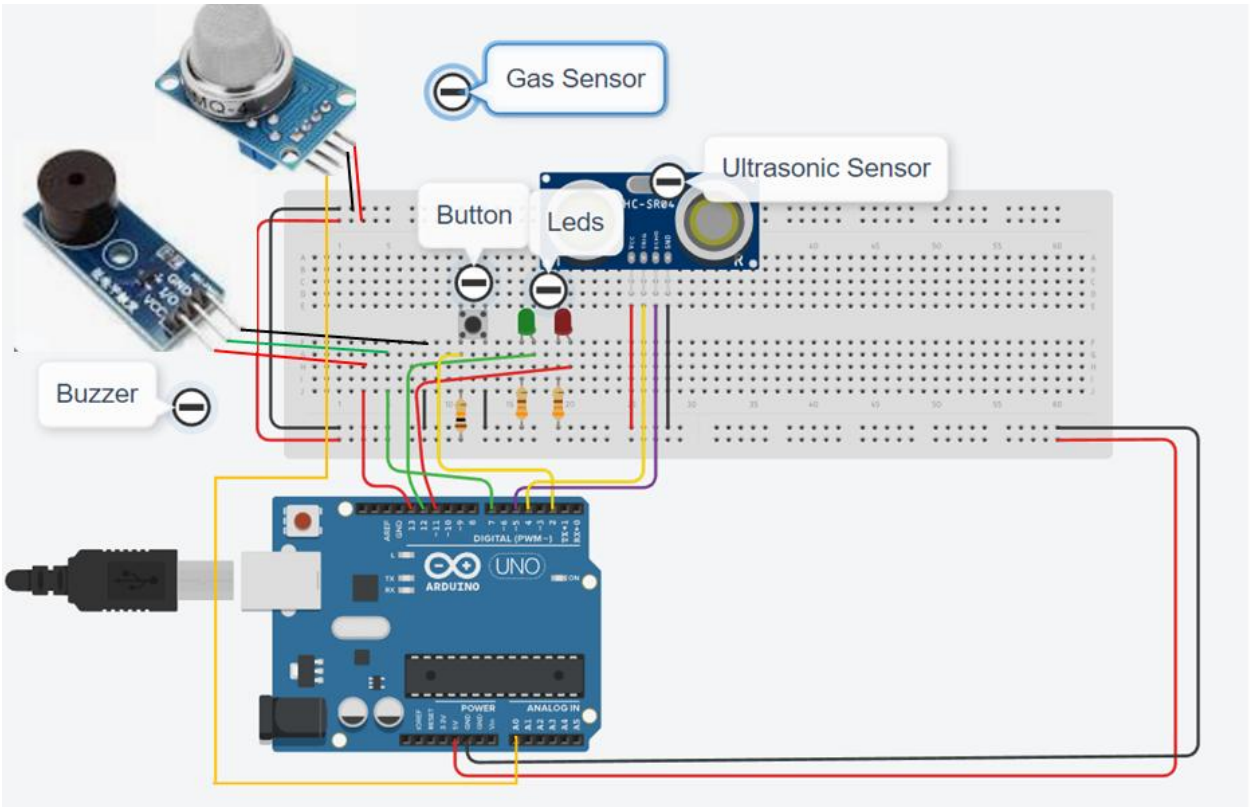
```
// codul pentru pasul 2 – adăugare sensor ultrasonic HC-SR04
int red_led=12;
int green_led=11;
int sensorThres=400;

//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);
}
```

Pasul 3 (Adăugăm un sensor de gaz la sistemul de alarmă)

Conectarea senzorului de gaze



Cum funcționează:

Când detectorul de gaz detectează gaz și sistemul de alarmă este deja activat, lumina verde se stinge, lumina roșie se aprinde și sunetul sonor.

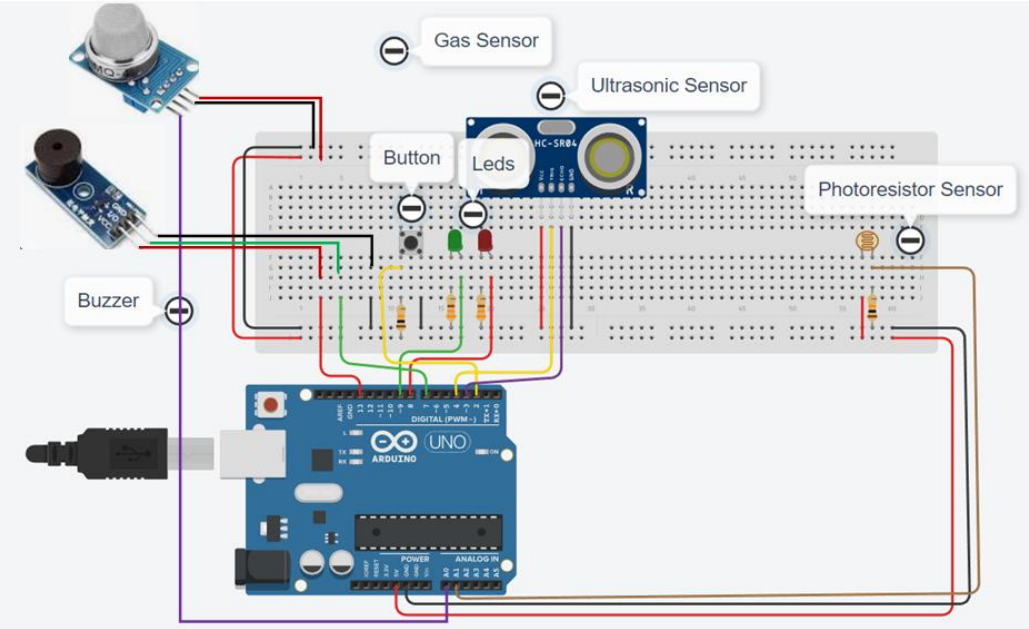
Programul Arduino pentru pasul 3:

```
//NUME: SISTEM DE ALARMĂ
// Adăugarea unui sensor de gaze în alarmă
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);
}
```

Pasul 4 (Adăugarea unui sensor cu fotorezistențe în sistemul de alarmă)
Conectarea unui sensor cu fotorezistență



Cum funcționează:

Sistemul de alarmă este activat când luminozitatea scade.

Programul Arduino pentru pasul 4:

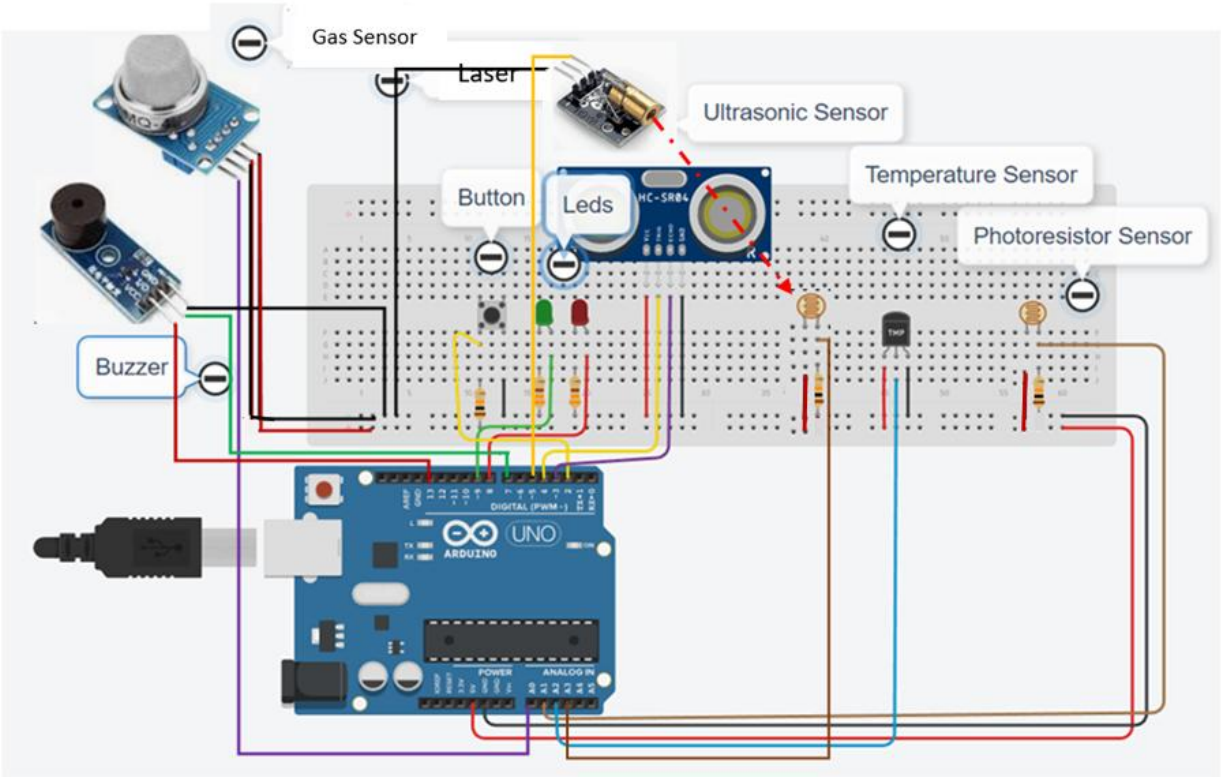
```
//NUME: SISTEM DE ALARMĂ
//ADĂUGAREA UNUI SENZOR CU FOTOREZISTENȚĂ
int red_led=8;
int green_led=9;
int gas_value;
int sensorThres=400;
//-----------------buzzer-----------------------
int buzz= 13; // I/O-pin from buzzer connects here
int buzzVcc= 7; //Vcc-pin from buzzer connects here
const int 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);}
```

Pasul 5 (Adăugarea unui sensor de temperature și a unui modul de transmitere cu laser în sistemul de alarmă)

Conectarea senzorului de temperatură



În acest pas vom avea nevoie de un transmițător laser, un fotorezistor pentru receptor și un rezistor de 1KOhm pentru a-l proteja.

Cum funcționează:

Transmițătorul laser împreună cu senzorul fotorezistor care funcționează ca un receptor instruiește Arduino să pornească sistemul atunci când din anumite motive recepția este întreruptă. De asemenea, sistemul de alarma este activat atunci cand valoarea senzorului de temperatura depaseste o anumita valoare.

Programul Arduino pentru pasul 5:

```
//NUME : SISTEM DE ALARMĂ
//ADĂUGAREA UNUI SENZOR DE TEMPERATURĂ
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
```

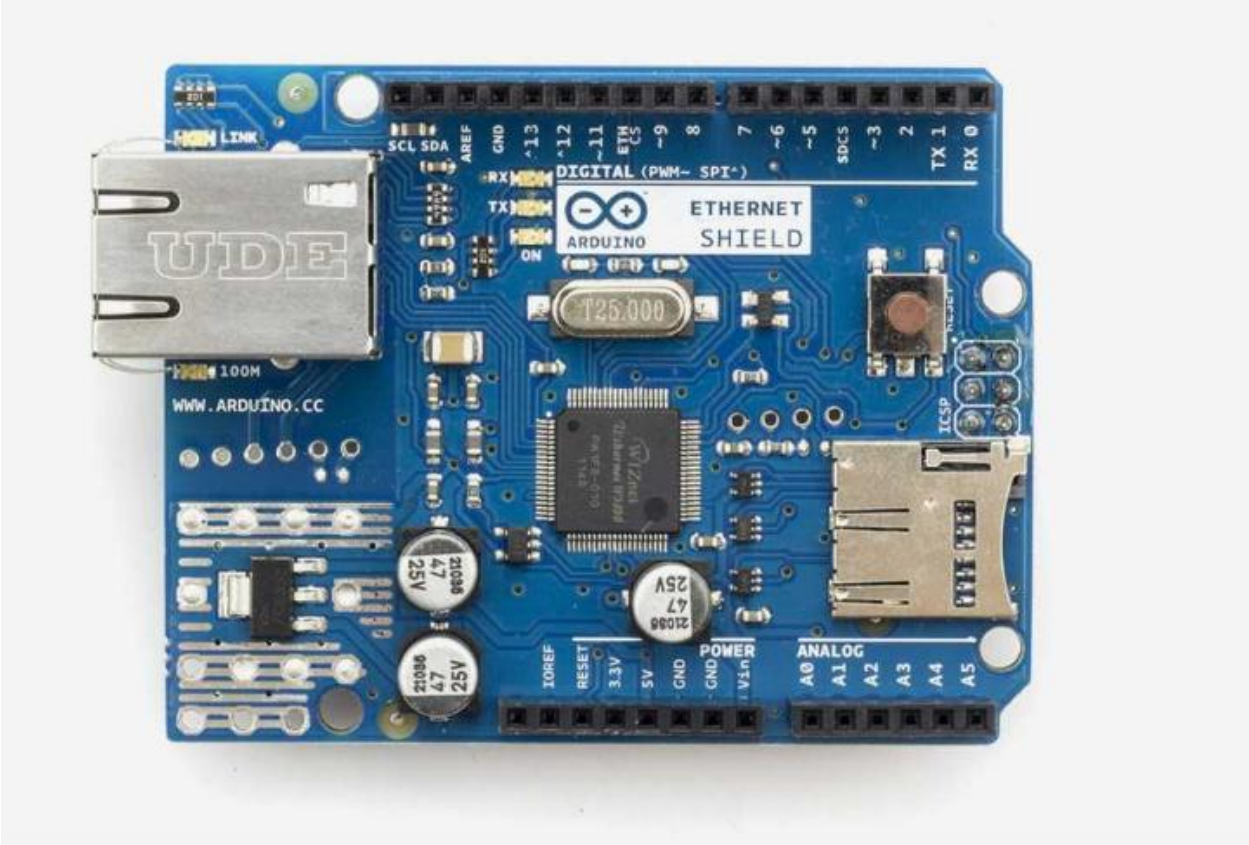
```
        digitalWrite(buzz, HIGH); // Tone OFF
        delay(sPause); // Symbol pause

        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);
}
```

PROIECT 3- NUME: WebServer

Scopul proiectului: În acest proiect, studenții vor vedea cum conectăm un arduino la o rețea locală folosind un scut Ethernet. Ei vor experimenta cu senzori și vor putea vedea valorile primite pe ecranul computerului prin intermediul rețelei locale.
Prin acest proiect, elevii ar putea experimenta cu:

- un senzor de gaz şi
- un senzor fotorezistor
- un senzor de temperatură

Acest scut Ethernet permite unei plăci Arduino să se conecteze la internet.



Lista componenetelor

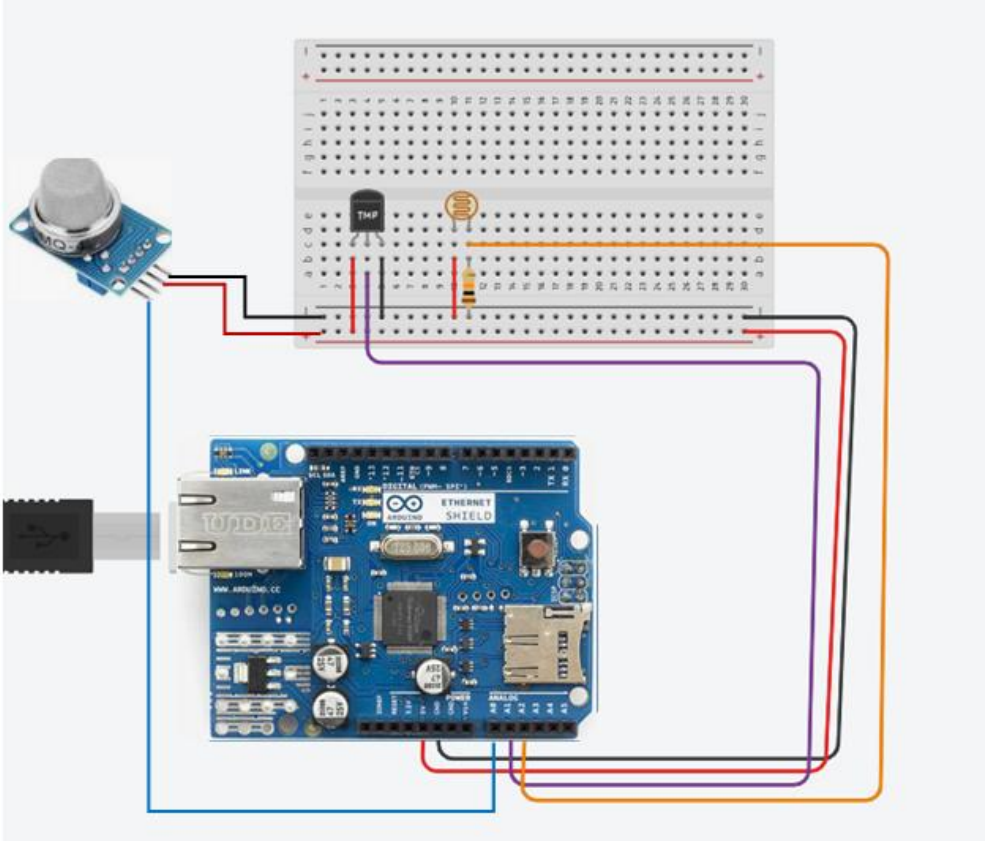
- Ethernet Shield
- Ethernet cablu
- Breadboard and Jump Wires
- Cablu USB
- Arduino UNO R3
- Un sensor cu fotorezistență
- Sensor de gaz
- Sensor de temperatură
- Un resistor de 10Kohm

Cum funcționează:

Arduino Ethernet Shield 2 permite unei plăci Arduino să se conecteze la internet. Se bazează pe (cipul Wiznet W5500 Ethernet). Wiznet W5500 oferă o stivă de rețea (IP) capabilă atât de TCP, cât și de UDP. Suportă până la opt conexiuni de priză simultane.

În acest exemplu, elevii vor căuta IP-ul scutului ethernet într-un browser de rețea locală și vor putea vedea valorile senzorilor pe care i-au conectat la scutul ethernet într-o pagină prietenoasă. Pagina se reîmprospătează automat la fiecare 5 secunde.

Circuitul proiectului:



Metodologia

În primul rând, descriem fluxul de lucru dorit al proiectului care urmează să fie dezvoltat în pași. Elevii sunt rugați apoi să selecteze componentele necesare pentru proiectarea și desenarea circuitului. Ei vor folosi aplicația „Arduino” pentru a scrie codul.
Apoi vor crea circuitul, vor scrie codul pe instrumentul software Arduino și l-au încărcat pe placa Arduino.
Elevii vor verifica dacă circuitul a fost construit corect.
Acest proiect se adresează studenților de nivel de bază, care încep să beneficieze de rezultatele și rezultatele proiectului nostru Erasmus+ KA-202 Parteneriate strategice în VET, numit „ArduInVet”.

Programul Arduino al proiectului:

```
/*
  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);
}
```

un screenshot al paginii proiectului

Erasmus+ KA-202

Proiecte de parteneriate strategice pentru educație și formare profesională

Titlul proiectului: "Predarea și învățarea ARDUINO în formarea profesională"

Acronimul proiectului: " ARDUinVET "

Numărul proiectului: "2020-1-TR01-KA202-093762"

Proiecte Arduino gratuite

(Aceste proiecte au fost realizate de elevii școlilor partenere din parteneriatul ARDUinVET)

Nr	Numele proiectului	Țara
1	PĂLĂRIE DE DISTANȚARE SOCIALĂ	TURCIA
2	CONTOR DE TEMPERATURĂ	GRECIA
3	LAMPĂ DE FRÂNĂ DE URGENȚĂ - LAMPĂ DE FRANĂ ADAPTIVĂ	ROMÂNIA
4	URMĂRITOR DE LINIE	AUSTRIA
5	SERĂ AUTOMATIZATĂ	ITALIA

1_NUMELE PROIECTULUI: PĂLĂRIE DE DISTANȚARE SOCIALĂ (TURCIA)

Scopul Proiectului: În aceste zile, când pandemia de Covid-19 este efectivă, respectarea regulii „DISTAȚIEI SOCIALE” este una dintre condițiile de bază pentru a nu te îmbolnăvi.

Pentru a atrage atenția asupra problemei „distanțării sociale”, decidem să proiectăm un proiect Arduino pe această temă.

Regula este reamintită cu „Pălăria de Distanțare Socială”, pe care o vom pregăti. Va emite un sunet sonor sonor și, în același timp, o avertizare ușoară.

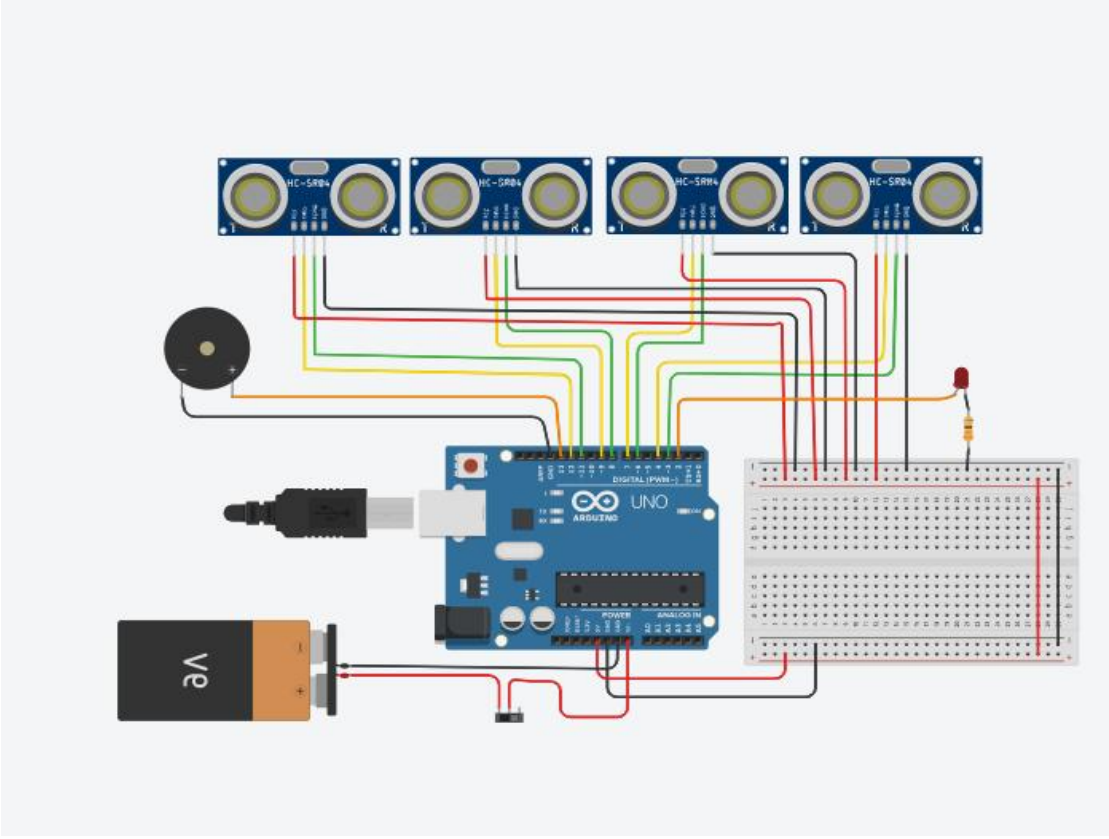
Metoda noastră de proiect:

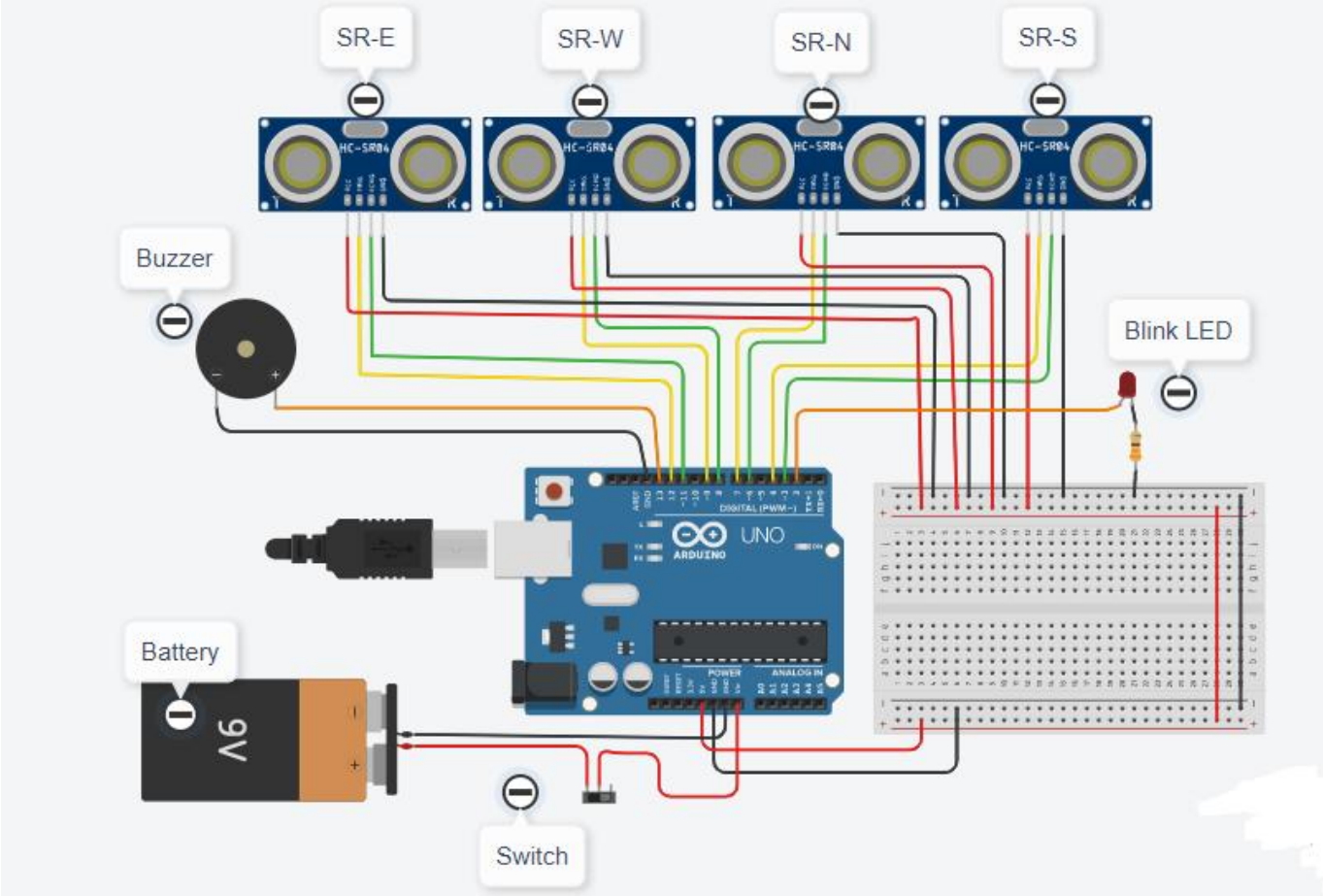
Mai întâi, este pregătit algoritmul proiectului. Conform acestui algoritm pregătit, se realizează designul logic și se scrie programul Arduino. Arduino este folosit în proiectarea circuitului. În timpul realizării acestor modele, elevii noștri beneficiază de toate rezultatele și produsele proiectului Erasmus+ KA-202 VET, denumit „ArduinVET”.

Circuitul nostru de proiectare finalizat este configurat după furnizarea echipamentelor necesare și programarea Arduino. Circuitul nostru final de proiectare este plasat vizual pe pălărie.

Acest proiect este expus la târguri și reamintește vizitatorilor regula „Distanțare socială” și importanța acesteia.

Circuitul proiectului





Cum funcționează:

4 senzori ultrasonici de distanță sunt plasați în 4 direcții ale pălăriei. Când senzorii detectează pe cineva, care se apropie sau aproape de 100 cm sau mai jos, soneria va suna și LED - ul va clipi. Circuitul va fi alimentat de o baterie de 9V.

Lista de materiale:

O pălărie, un Arduino Uno R3, 4 senzori ultrasonici de distanță, un Buzzer, un resistor de 330Ω, un LED roșu, o baterie de 9V, un comutator glisant.

Programul Arduino al proiectului:

```
//Numele proiectului: Pălăria de distanțare socială
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 distance1 = (time1/2) / 28.97;         //Distanțarea socială este mai mică sau egală cu 100cm
//-----------------------------------------------------
 digitalWrite(trigPin2, LOW);
delay(3);
digitalWrite(trigPin2, HIGH);
delay(3);
digitalWrite(trigPin2, LOW);

int time2 = pulseIn(echoPin2, HIGH);
int distance2 = (time2/2)  / 28.97;    // Distanțarea socială este mai mică sau egală cu 100cm
//-----------------------------------------------------
 digitalWrite(trigPin3, LOW);
delay(3);
digitalWrite(trigPin3, HIGH);
delay(3);
digitalWrite(trigPin3, LOW);
int time3 = pulseIn(echoPin3, HIGH);
int distance3 = (time3/2)  / 28.97;       // Distanțarea socială este mai mică sau egală cu 100cm
//-----------------------------------------------------
digitalWrite(trigPin4, LOW);
delay(3);
```

```
digitalWrite(trigPin4, HIGH);
delay(3);
digitalWrite(trigPin4, LOW);
int time4 = pulseIn(echoPin4, HIGH);
int distance4 = (time4/2)  / 28.97;         // Distanțarea socială este mai mică sau egală cu 100cm.
//---------------------------------------------------------
if(distance1<75)
 {
  digitalWrite(13,HIGH);
  digitalWrite(2,HIGH);
  }
 else if(distance2<75)
 {
  digitalWrite(13,HIGH);
   digitalWrite(2,HIGH);
  }
 else if(distance3<75)
 {
  digitalWrite(13,HIGH);
   digitalWrite(2,HIGH);
  }
 else if(distance4<75)
 {
  digitalWrite(13,HIGH);
   digitalWrite(2,HIGH);
  }
```

```
else
{
digitalWrite(13,LOW);
digitalWrite(2,LOW);
}
 Serial.println(distance1);      // pentru a vedea care senzor este activat pe monitorul serial
 Serial.println(distance2);
 Serial.println(distance3);
 Serial.println(distance4);
 delay(500);
}
```

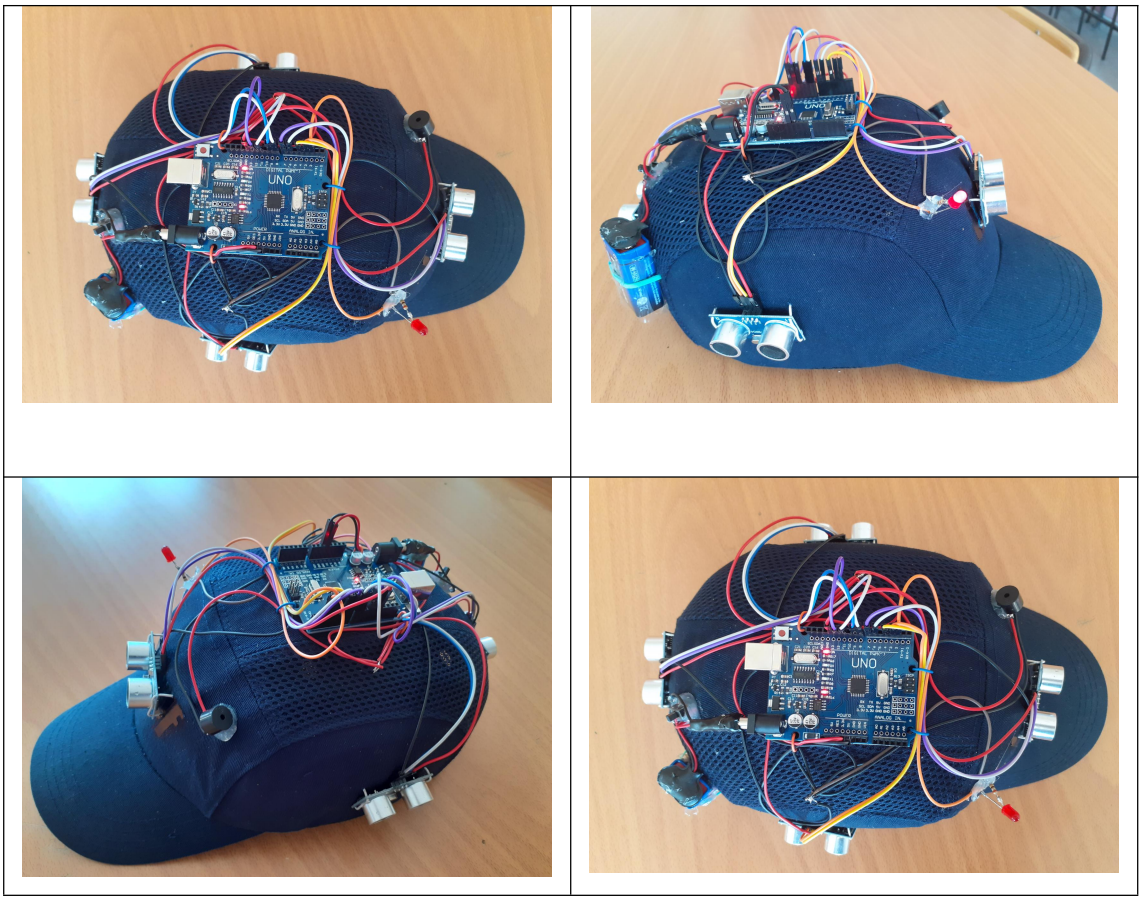


Foto: Designul final al circuitului pe șapcă

2_NUMELE PROIECTULUI: CONTOR DE TEMPERATURĂ (GRECIA)

Scopul proiectului: Acest proiect este o versiune extinsă a proiectului de bază Love Meter pe care oricine îl poate găsi în manualul original Arduino. În această versiune extinsă, elevii vor putea să lucreze cu un senzor (senzor de temperatură), să experimenteze cu potențiometrul, să aprindă un LED RGB și să cunoască afișajul LCD (dacă nu este disponibil, altfel afișajul va fi doar Monitorul Serial al instrumentul software Arduino).

Este un proiect care permite elevilor să experimenteze cu multe componente de diferite tipuri și să se familiarizeze cu valorile de intrare și metodele de afișare.

Metoda de lucru:

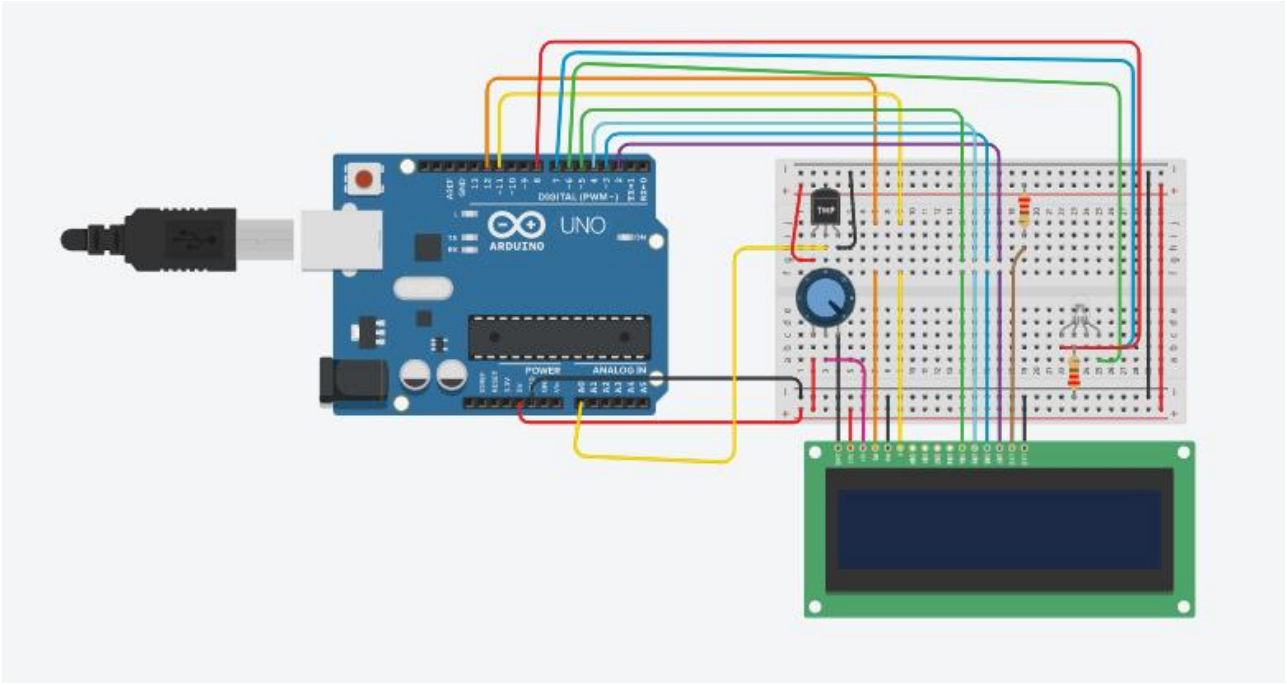
Inițial, descriem fluxul de lucru dorit al proiectului. Apoi le cerem elevilor să selecteze componentele necesare și să utilizeze Tinkercad pentru a proiecta și a desena circuitul. Ei vor folosi Tinkercad Code Tool pentru a scrie codul.

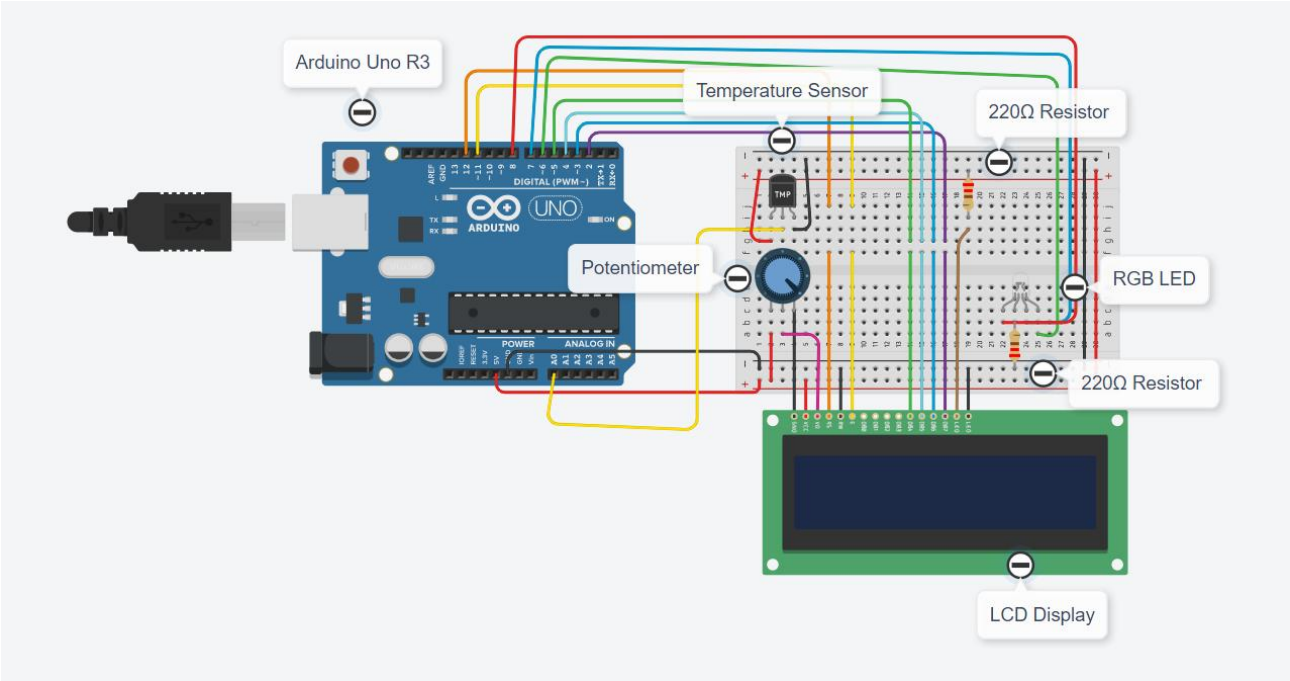
Apoi vor crea circuitul, vor scrie codul pe instrumentul software Arduino și îl încarcă pe placa Arduino.

Simularea pe Tinkercad va verifica dacă circuitul a fost construit corect.

Acest proiect este pentru elevii de nivel de bază, care încep să beneficieze de rezultatele din proiectul nostru Erasmus+ KA-202 Parteneriate strategice în VET, numit „ArduinVET”.

Circuitul proiectului:





Cum funcționează:

Senzorul de temperatură introduce valorile temperaturii camerei. Desigur, elevii pot schimba temperatura folosind degetele. Temperatura va fi afișată pe afișajul LCD (dacă este disponibil) și pe monitorul serial. LED-ul RGB va avea culori diferite (oprit, verde, albastru, roșu) în funcție de valoarea temperaturii. Potențiometrul va fi folosit ca comutator pentru afișajul LCD. Circuitul va fi alimentat de la cablul USB (dacă este necesar, puteți utiliza o cutie de baterii).

Lista de componente:

Componentele necesare sunt: un Arduino Uno R3, un sensor de temperatură, două (2) rezistențe de 220Ω, un RGB LED și (opțional) un potențiometru și un afișaj LCD (16*2)

Codul Arduino al proiectului:

//Numele proiectului: CONTOR DE TEMPERATURĂ

// include codul bibliotecii:

```
#include <LiquidCrystal.h>
int t=0;
int sensor = A0;
float temp;
float tempc;
float tempf;
// inițializați biblioteca cu numerele pinilor de interfață
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
//**********RGB**********
const int sensorPin = A0;
// temperature camerei în Celsius
const float baselineTemp = 20.0;
//*************************
void setup() {
  // setează numărul de coloane și rânduri ale LCD ului
pinMode(sensor,INPUT);
lcd.setCursor (0,0);
lcd.begin(16, 2);
  // Afișează mesajul pe LCD.
lcd.print ("    ARDUINO     ");
lcd.setCursor (0,1);
lcd.print ("TEMPERATURE METER");
```

```
delay (3000);
Serial.begin(9600);
for(int pinNumber = 6; pinNumber<8; pinNumber++){
   pinMode(pinNumber,OUTPUT); // pin6:Albastru pin7:Verde  pin8:Roșu
   digitalWrite(pinNumber, LOW);
  }
}
void loop() {
delay(2000);
t=t+2;
temp=analogRead(sensor);
//tempc=(temp*5)/10;
tempc=map(((temp-20)*3.04), 0, 1023, -40, 125);
tempf=(tempc*1.8)+32;
Serial.println("_______");
Serial.println("Temperature Logger");
Serial.print("Time in Seconds= ");
Serial.println(t);
Serial.print("Temp in deg Celcius = ");
Serial.println(tempc);
Serial.print("Temp in deg Fahrenheit = ");
Serial.println(tempf);
lcd.setCursor(0,0);
lcd.print("Temp in C = ");
lcd.println(tempc);
lcd.setCursor(0,1);
```

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


//************code RGB***************
  // read the value on AnalogIn pin 0
  // and store it in a variable
  int sensorVal = analogRead(sensorPin);
  // send the 10-bit sensor value out the serial port
  Serial.println("sensor Value: ");
  Serial.println(sensorVal);

  // convert the ADC reading to voltage
  float voltage = (sensorVal/1024.0) * 5.0;
  // Trimite tensiunea la portul serial
  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);
}
delay(100);
}
```

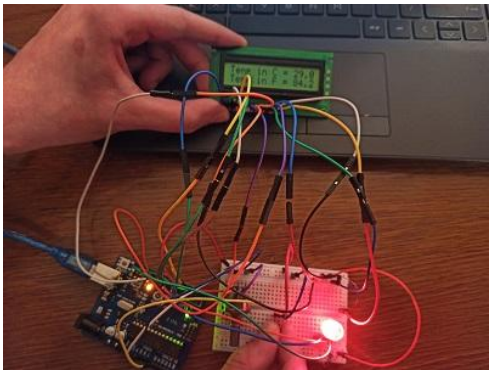
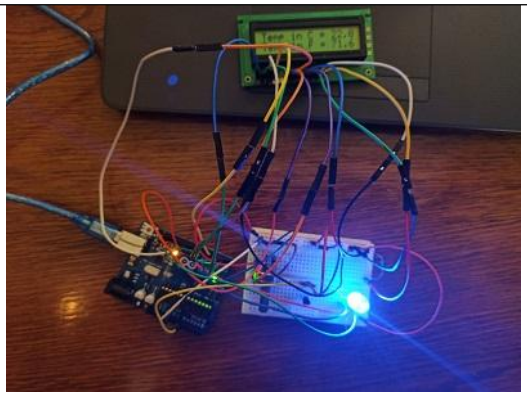
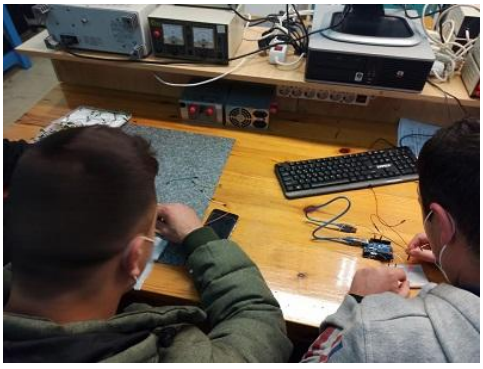





Foto: Kit ul proiectului și elevii lucrând la realizarea acestuia

3_NUMELE PROIECTULUI: LAMPĂ DE FRÂNĂ DE URGENȚĂ - LAMPĂ DE FRANĂ ADAPTIVĂ (ROMÂNIA)

Scopul proiectului: În cazul unei frânări puternice, lumina de frână adaptivă poate avertiza traficul care se apropie și, astfel, poate ajuta la prevenirea coliziunilor din spate. Acestea sunt activate dacă, în timpul frânării, și în funcție de presiunea și viteza de frânare, se identifică o situație critică de frânare.

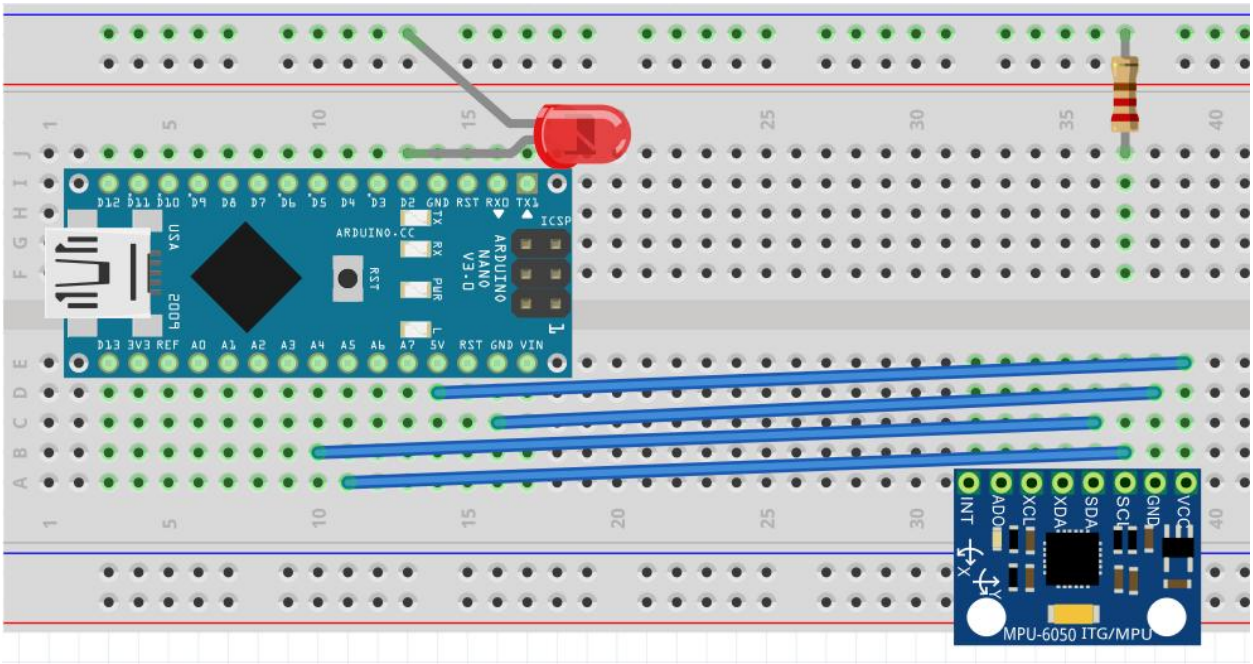
Metoda proiectului:

Mai întâi, este pregătit algoritmul proiectului. Conform acestui algoritm pregătit, se realizează designul logic și se scrie programul Arduino. Arduino este folosit în proiectarea circuitului. În timpul realizării acestor modele, elevii noștri beneficiază de toate rezultatele și produsele proiectului Erasmus+ KA-202 VET, denumit „ArduinVET”.

Circuitul nostru de proiectare finalizat este configurat după furnizarea echipamentelor necesare și programarea Arduino. Circuitul nostru final de proiectare este plasat într-o mașină.

Acest proiect este expus la târguri și reamintește vizitatorilor cât de importantă este siguranța în trafic.

Circuitul proiectului:



Cum funcționează:

Luminile de frână de urgență sunt activate pentru a alerta vehiculele din spate despre frânarea puternică. Funcția înseamnă că lumina de frână clipește în loc să strălucească cu o strălucire constantă, ca în frânarea normală. Luminile de frână de urgență sunt activate la viteze de peste 50 km/h în cazul frânării puternice.

Se bazează pe un accelerometru electronic (MPU6050) asistat de un microcontroler și detectează frânarea bruscă pe baza forței de inerție și clipește rapid a 3-a oprire a frânei, făcând mașina mult mai ușor de observat decât spatele.

MPU6050 are un giroscop cu 3 axe, un accelerometru cu 3 axe și un procesor digital de mișcare integrat pe un singur cip. Funcționează la sursa de alimentare de 3V-5V. MPU6050 utilizează protocolul I2C pentru comunicare și transfer de date. Acest modul are un ADC de 16 biți încorporat care oferă o precizie deosebită.

Filtrul Kalman este un filtru recursiv eficient care evaluează starea unui sistem dinamic pe baza unei serii de măsurători sensibile la zgomot. Datorită caracteristicilor sale intrinseci, este un filtru

Lista de materiale:

un Arduino Nano, un MPU6050, o rezistență de 180 Ohm, un LED roșu.

Programul Arduino al proiectului:

//Numele proiectului: LAMPĂ DE FRÂNĂ DE URGENȚĂ - LAMPĂ DE FRANĂ ADAPTIVĂ

```
#include<Wire.h>;
//https://www.codetd.com/en/article/11749279 acesta este un articol care explică foarte bine cum să comunici mai rapid cu MPU6050
//deoarece în automobil este mult zgomot de la condițiile de drum, vibrații etc.se va folosi un filtru software foarte bun (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); // for USB monitor
  Serial.println ("Hit a key to start");     // inițializarea semnalului efectuată
  while (Serial.available() == 1);
  //Connect to G-Sensor
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0x00); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  delay(50);
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x1C);
  Wire.write(0x00);
  Wire.endTransmission(true);
  pinMode(2, OUTPUT); // semnal de ieșire pentru prima și a doua etapă.
digitalWrite (2, LOW);
//pinMode(7, OUTPUT);// semnal de ieșire pentru a doua etapă. Activați această linie și următoarea dacă doriți o funcționalitate mai expresivă.
//digitalWrite (7, LOW); Dacă activați această opțiune, trebuie să activați pinul 7 de ieșire digitală în toate schițele!!
// acest semnal poate fi utilizat cu un tranzistor PNP sau un MOSFET P-Gate pentru a finaliza circuitul
```

```
// in circuit real folosesc un LED albastru de 5v. În circuitul de fritizare utilizați un LED roșu obișnuit de 1,8 cu rezistor suplimentar - 180 ohm}
void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    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_movavg = mov_avg(c_avg, YAxisFull);
    int YAxis_median = median(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
if (YAxis_Filtered/16384.0 > 0.4 and YAxis_Filtered/16384.0 <= 0.7){
  for (int i = 1; i <= 4; i++) {
```

```
      digitalWrite (2, HIGH);
     // digitalWrite (7, HIGH);
      delay (75);
      digitalWrite (2, LOW);
     // digitalWrite (7, LOW);
      delay (50);}}
else {
  //SECOND stage - very hard brake
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;
    if (c_avg > 5 && c_med > 5)
    {      c_avg = 6;
      c_med = 6;    }  }}
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_calculated;
  return kalman_calculated;}
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;
  if (val1 == 0) {
    m = m + 1;    }
  if (val2 == 0) {
    m = m + 1;    }
  if (val3 == 0) {
    m = m + 1;    }
  if (val4 == 0) {
    m = m + 1;    }
  if (val5 == 0) {
    m = m + 1;    }
  if (input == 0) {
    m = m + 1;    }
  d = 6 - m;
  if (d == 0)
  {      avg = input;
    counter = 1;    }
```

```
    else
    {      avg = (val1 + val2 + val3 + val4 + val5 + input) / d;    }
    val1 = val2;
    val2 = val3;
    val3 = val4;
    val4 = val5;
    val5 = input;  }
  return avg;}
float median (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) {
  for (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;      }    }  }  }
//functionează cu filtru aplicat
float general_filter (int counter_avg, int counter_med, float input) {
float input_movavg = mov_avg(counter_avg, input);    //Mutarea aplicațieiAvg
float input_med = median(counter_med, input_movavg); //Aplicația Median
float input_filtered = kalman_filter(input_med);     // Aplicația Kalman
  return input_filtered;}
```

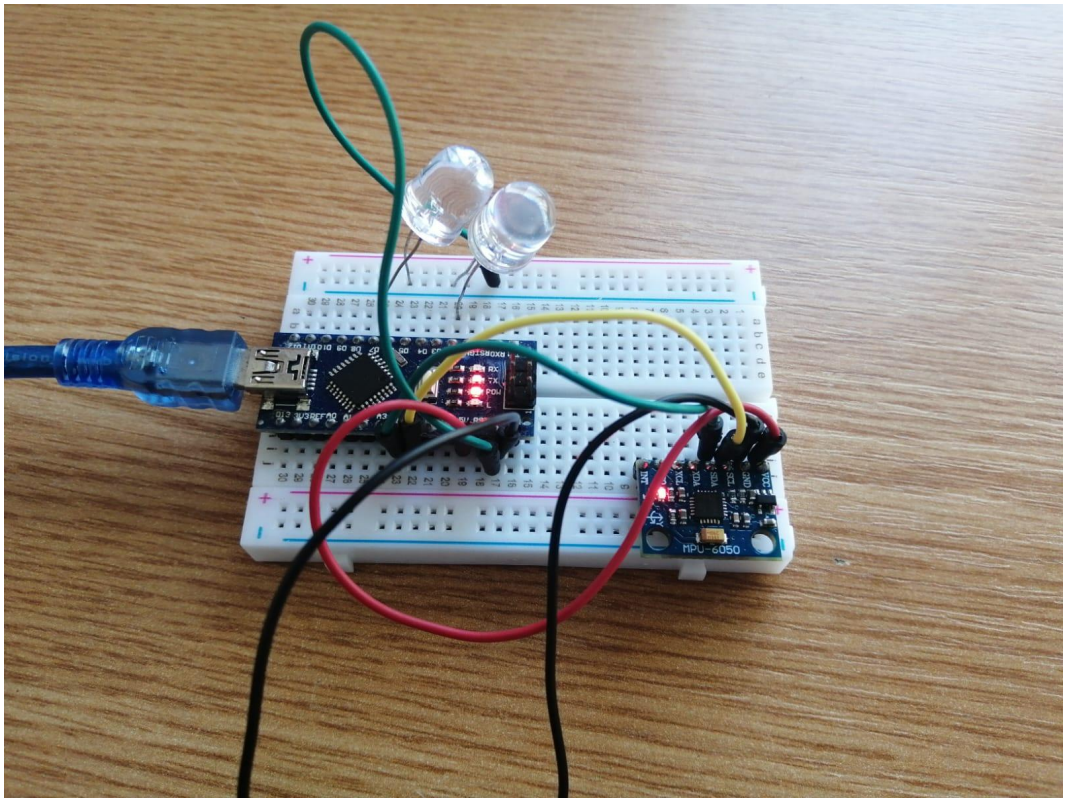


Foto: Circuitul final

4_NUMELE PROIECTULUI: URMĂRITOR DE LINIE (AUSTRIA)

Ce este un urmăritor de linie:

Robotul de urmărire a liniilor este o mașină mobilă care poate detecta și urmări linia trasată pe podea. În general, calea este predefinită și poate fi fie vizibilă ca o linie neagră pe o suprafață albă cu o culoare foarte contrastată. Cu siguranță, acest tip de robot ar trebui să detecteze linia cu senzorii de raze infraroșii (IR) instalați sub robot. După aceea, datele sunt transmise procesorului. Prin urmare, procesorul va decide recomandările potrivite și apoi le trimite șoferului și astfel calea va fi urmată de robotul urmăritor de linie.

Punctul important al construirii unui robot de urmărire a liniilor este un control bun care este suficient pentru a urma calea cât mai repede posibil.

Cum funcționează un robot urmăritor de linie

Conceptul de robot de urmărire a liniilor este legat de lumină. Aici, folosim comportamentul luminii pe suprafața alb-negru. Culoarea albă reflectă toată lumina care cade pe ea, în timp ce culoarea neagră absoarbe lumina.

În acest robot de urmărire a liniilor, folosim transmițătoare și receptoare IR (fotodiode). Sunt folosite pentru a trimite și primi lumini. Când razele IR cad pe o suprafață albă, acestea sunt reflectate către receptorul IR, generând unele modificări de tensiune.

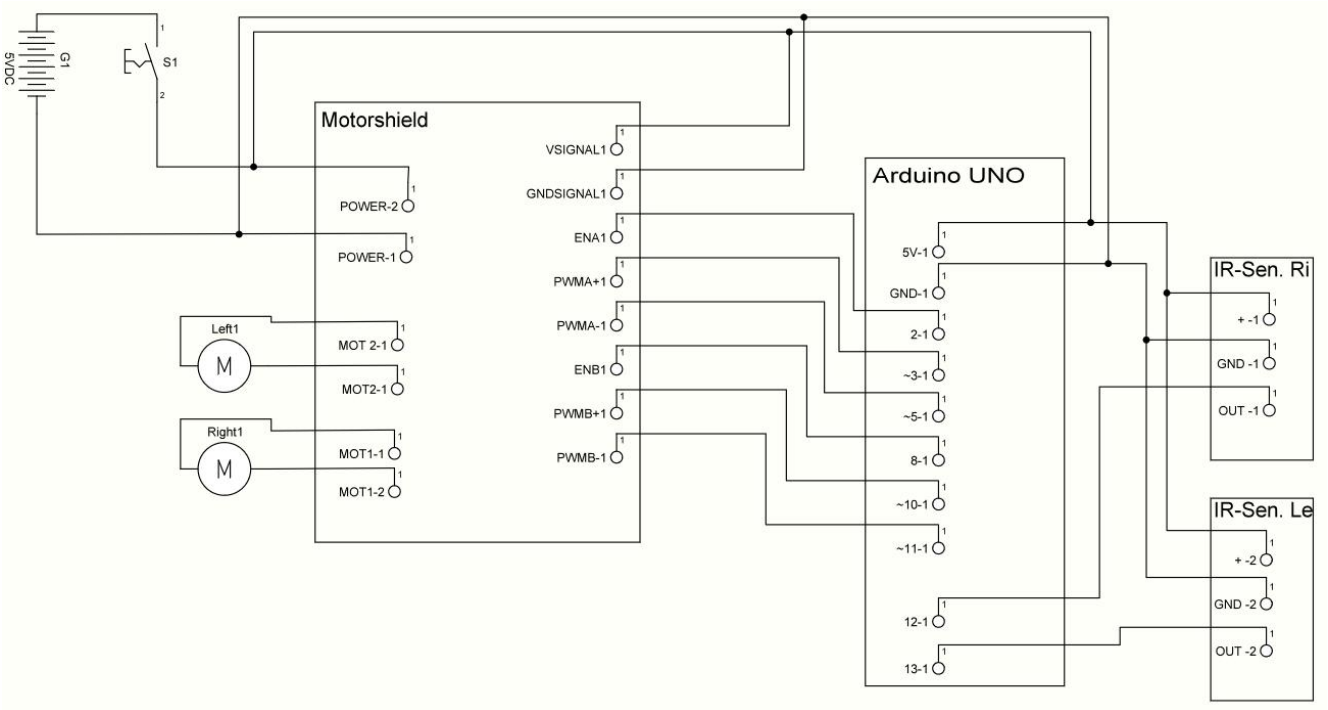
Când razele IR cad pe o suprafață neagră, acestea sunt absorbite de suprafața neagră și nicio rază nu este reflectată; astfel, receptorul IR nu primește nicio rază.

În acest proiect, când senzorul IR detectează o suprafață albă, un Arduino primește 1 (HIGH) ca intrare, iar când detectează o linie neagră, un Arduino primește 0 (LOW) ca intrare. Pe baza acestor intrări, un Arduino Uno oferă ieșirea adecvată pentru a controla robotul.

Lista de materiale:

Materiale	**Tip**	**număr**
Arduino	Uno	1
Scutmotor	de la HTL Wolfsberg	1
DC-Motor	COM-Motor01, 3-9Volt DC, joy-it	2
Sursă	Flip12 PhonePowerBank, 3350mAh	1
IR-Senzor	SEN-KY032IR, joy-it	2
Intrerupător principal		1
Jumperwire	RB-CB3-025, joy-it	1
USB Cablu	Digitus,AK-300102-010-S, Typ A-B	s

Schema circuitului:



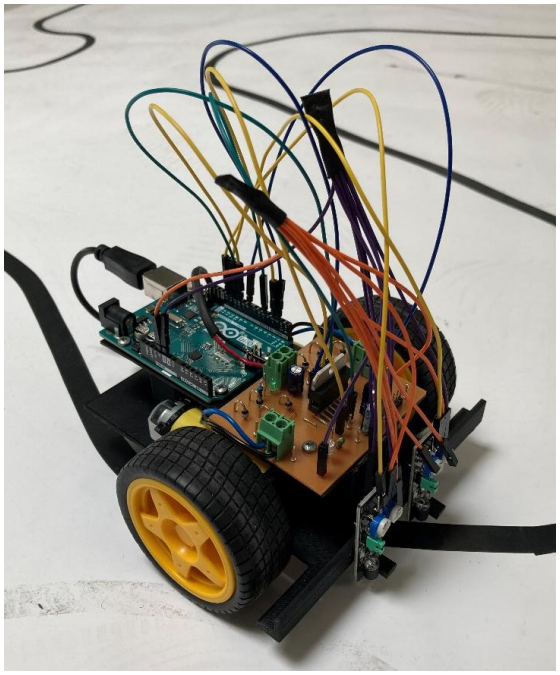


Foto: Produsul final

Program:

```
// Urmăritor de linie
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;

void 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;

void loop()
{
 senR = digitalRead(sr);
 senL = digitalRead(sl);

 if(!senR && !senL) //staight
 {
  analogWrite(mrf,220);
  analogWrite(mrb,0);
  analogWrite(mlf,200);
  analogWrite(mlb,0);
```

```
  }

  //right sensor on line
  //drive to right
  else if(senR && !senL)
  {
    analogWrite(mrf,0);
    analogWrite(mrb,100);
    analogWrite(mlf,255);//255
    analogWrite(mlb,0);
  }

  //left sensor on line
  //drive to left
  else if(!senR && senL)
  {
    analogWrite(mrf,255);//255
    analogWrite(mrb,0);
    analogWrite(mlf,0);
    analogWrite(mlb,100);
  }

  // both sensors on black
  // stop
  else if(senR && senL)
  {
    analogWrite(mrf,0);
    analogWrite(mrb,0);
    analogWrite(mlf,0);
    analogWrite(mlb,0);
    delay(3000);
  }
}
```

5_NUME PROIECT: SERĂ AUTOMATIZATĂ (ITALIA)

Scopul proiectului: De ce un proiect de seră automatizat la școală?

Elevii de la școală au urmat diverse căi pe cetățenia activă și pe Agenda ONU 2030 pentru Dezvoltare Durabilă, despre durabilitatea energetică, importanța evitării risipei de apă și necesitatea dezvoltării unor metode de cultivare inovatoare și prietenoase cu mediul, având în vedere schimbările climatice profunde care au loc. loc. Conștientizarea ecologică a tinerilor se exprimă în căutarea de soluții tehnice și idei inovatoare, pornind de la cunoștințele teoretice și de laborator dobândite la școală inerente oportunităților de aplicare ale Arduino.

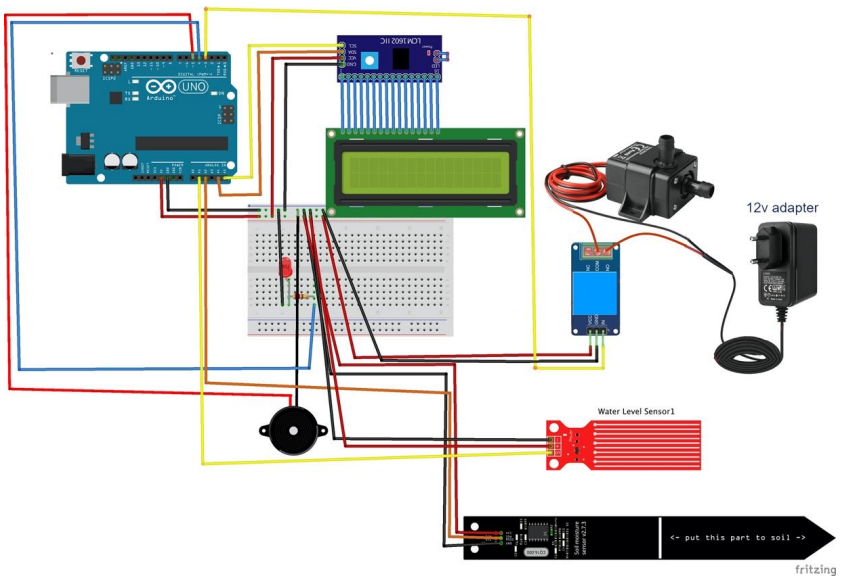
Unii elevi, sprijiniți de profesori, au proiectat un sistem de detectare a temperaturii solului și a aerului, a umidității solului, pentru a gestiona irigarea și ventilația unui prototip de seră potrivit pentru cultivarea fructelor și legumelor locale.

Experimentarea a dat rezultatele scontate și va contribui la implementarea efectivă în zona din vecinătatea școlii destinate să găzduiască viitoarea grădină de legume școlare.

Metoda noastră de proiect:

În primul rând, a fost pregătită o listă cu toți parametrii pe care dorim să-i folosim pentru a opera sera. Pornind de la aceasta a fost pregătită o primă versiune a algoritmului proiectului. Conform acestui algoritm, au fost achiziționați senzorii necesari și s-a realizat proiectul logic. Arduino este folosit în proiectarea circuitelor. În cele din urmă, scriptul final a fost implementat. După verificarea setării corecte a valorilor de prag ale diferiților senzori, aceștia au fost poziționați în interiorul serei.

Circuitul proiectului



Cum funcționează:

Un display LCD oferă valorile temperaturii și umidității aerului, umidității solului și cantității de apă prezentă în recipientul de irigare. cand valorile sunt sub un anumit prag, o pompa motorizata comandata de un releu porneste irigarea solului. când nivelul apei este sub un anumit prag, afișajul afișează o avertizare de reumplere însoțită de o alarmă sonoră.

Lista de materiale:

O seră, un Arduino Uno R3, un sensor de temperature/uniditate DHT11, un buzzer, un sensor de umiditate a solului, un sensor de nivel al apei, un LED roșu, un adaptor de 12V, o pompă de 12V, un resistor de 220 ohm, un releu, un afișaj LCD.

Programul Arduino al proiectului:

```
//Nume proiect: SERĂ AUTOMATIZATĂ
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#include <DHT.h>
#define DHTPIN 11
#define DHTTYPE DHT11 // DHT 11
DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27,20,4);  // setează adresa afișajului LCD la 0x27 pentru 16 caractere și 2 linii
int t, h, Lumen, lumin, water, igro, umdtr, wlevel;
const int pin_water = 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("School");
  lcd.setCursor(3,1);
  lcd.print("Greenhouse");
  delay (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() {
  // water level
  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.setCursor(2,0);
    lcd.print("Refill Water");
    delay (1000);
    lcd.clear();
} else {
    noTone(pin_buzzer);
    digitalWrite(pin_led, LOW);
}
    // Igrometer
igro = analogRead(pin_igro);
umdtr = map (igro, 0, 1023, 0, 100);

if(umdtr < 45){
    digitalWrite(pin_pump, LOW);
    delay(10000);
    digitalWrite(pin_pump, HIGH);
}
    // citește umiditatea și temperatura cu senzorul DHT11
h = dht.readHumidity();
t = dht.readTemperature();
 // poziția cursorului în coloana 0 și linia 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.setCursor(10, 0);
  lcd.print("H:");
  lcd.print(h);
  lcd.print("%");

  lcd.setCursor(0, 1);
  lcd.print("SH:");
  lcd.print(umdtr);
  lcd.print("%");

  lcd.setCursor(10, 1);
  lcd.print("WL:");
  lcd.print(wlevel);
  lcd.print("%");
}
```

Photo: Final design

Anexe

Misiunea lui Arduino este de a permite oricui să-şi îmbunătăţească viaţa prin electronice accesibile şi tehnologii digitale. A existat odată o barieră între lumea electronică, design şi programare şi restul lumii. Arduino a spart această barieră. Pentru mai multe informaţii despre Arduinos, puteţi vizita site-ul, mai jos...

www.arduino.cc/

Pentru mai multe informaţii despre proiectele noastre ARDinVET, puteţi vizita site -ul de mai jos…

www.arduinvet.com

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

„Acest proiect este finanțat de Programul Erasmus+ al Uniunii Europene. Cu toate acestea, Comisia Europeană și Agenția Națională Turcă nu pot fi făcute responsabile pentru orice utilizare care poate fi făcută a informațiilor conținute în acestea.”

“This project is Funded by the Erasmus+ Program of the European Union. However, European Commission and Turkish National Agency cannot be held responsible for any use which may be made of the information contained therein”