

Universitatea
Transilvania
din Brașov
FACULTATEA DE INGINERIE ELECTRICĂ
ȘI ȘTIINȚA CALCULATOARELOR

SISTEM DE SEMNALIZARE PENTRU TRANSPORT ALTERNATIV

Lucrare de laborator

Studenti:
MITU Mariana-Luciana
CÎNIPĂ Alexandru
ENESCU Raymond
MARINESCU Mihai

BRAȘOV, 2024

Cuprins

Cuprins

1	Introducere	2
1.1	Descrierea proiectului.....	2
2	Obiectivele lucrării de laborator	4
2.1	Rezultate așteptate	5
3	Introducere în lucrul cu Arduino	6
3.1	Componenta Hardware	6
3.2	ATmega328P.....	8
3.3	Componenta Software: Arduino IDE	9
3.4	Instalare și utilizare Arduino.....	9
4	Descrierea componentelor	11
5	Structura proiectului.....	15
5.1	Schema logică.....	15
5.2	Schema electrică	17
5.3	Etape de conectare a componentelor sistemului	19
6	Resursele interne din microcontroller (timer, regiștri, întreruperi, cad)	23
7	Realizarea lucrării de laborator.....	24
7.1	Buzzer.....	24
7.2	Matrice de 8x8 led-uri	26
7.3	Fotorezistență.....	29
7.4	Modulul Joystick.....	31
7.5	Ecran OLED.....	34
7.6	Senzor Ultrasonic	36
8	Rezolvarea proiectului	39
9	Foi de calatalog.....	40

1 INTRODUCERE

Într-o lume în care traficul reprezintă un impediment major în realizarea obiectivelor zilnice și distrugе starea de spirit încă de la primele ore ale dimineții, mijloacele de transport alternative au devenit soluții ideale pentru a ușura viațа locuitorilor din zonele urbane. Mobilitatea urbană diversificată salvează timp și energie, încurajând un stil de viață sănătos, dar, însă, necesită asumarea anumitor riscuri în ceea ce privește siguranțа.

Aproape 60% dintre accidentele rutiere implică bicicliști și utilizatori de trotinete, iar majoritatea acestor incidente se datorează lipsei de semnalizare adecvată. De exemplu, în 2022, au fost raportate 8 decese și peste 130 de răniți grav în accidente de trotinete electrice, multe dintre aceste cazuri fiind cauzate de neatentia utilizatorilor și de absența unor semnalizări corespunzătoare.

Astfel, introducerea unui sistem de semnalizare pentru mijloacele de transport alternative este nu doar relevantă, ci și esențială pentru a reduce aceste statistici alarmante. Prin integrarea tehnologiilor moderne în utilizarea acestor mijloace de transport, se vizează creșterea vizibilității și, implicit, a siguranței utilizatorilor în trafic. Această inițiativă nu doar că va contribui la prevenirea accidentelor, dar va și încuraja mai multe persoane să adopte transportul alternativ, având un impact pozitiv asupra mediului și sănătății publice.

1.1 DESCRIEREA PROIECTULUI

Proiectul de laborator constă în dezvoltarea unui sistem care să permită schimbarea direcției de mers sau apariția obstacolelor pentru un mijloc de transport alternativ, format din următoarele componente: placă Arduino Uno, modul Joystick, matrice de LED-uri 8x8, senzor ultrasonic, Buzzer, fotorezistență, ecran OLED și breadboard.

Primul pas constă în adăugarea unui **modul Joystick** pe unul dintre mânerele vehiculului, pentru a asigura accesibilitate rapidă în indicarea direcției de mers. Indicatorul de direcție este format dintr-o **matrice de LED-uri 8x8** va fi poziționat în partea din spate a mijlocului de transport

(spre exemplu, pentru o bicicletă ar fi poziționat în spatele șeii) pentru a putea fi ușor de remarcat de către ceilalți participanți la trafic. Placa **Arduino Uno** va fi protejată într-o cutie în spatele indicatorului de direcție.

În partea din față a vehiculului (de exemplu, pe ghidon) se va monta un **senzor ultrasonic**, care măsoară distanța față de obiectele din față, utilizând unde sonore reflectate. Lângă acesta se va afla un ecran OLED care va indica, în funcție de valoarea citită de senzorul ultrasonic, distanța bicicletei față de un eventual obstacol. În cazul în care obstacolul se află la o distanță mai mică decât distanța de prag (setată în prealabil), atunci se va activa **Buzzer-ul** și, totodată, se vor aprinde LED-urile. Adițional, se poate monta un senzor ultrasonic și în partea din spate a vehiculului pentru a verifica dacă un alt participant la trafic se apropie prea mult de biciclist, ceea ce va duce la aprinderea LED-urilor și activarea buzzer-ului pentru a alerta utilizatorul. Implementarea adițională nu va fi implementată în acest proiect, deoarece este suficient să folosim un singur senzor ultrasonic pentru a înțelege cum se folosește și care sunt funcționalitățile acestuia.

În plus, montând **un senzor de luminozitate**, sistemul va putea ajusta automat intensitatea LED-urilor de semnalizare în funcție de condițiile de iluminare (de exemplu, în timpul nopții, LED-urile vor luceafără mai intens). Aceasta este un detaliu important pentru siguranța pe timp de noapte sau în condiții de vizibilitate redusă.

2 OBIECTIVELE LUCRĂRII DE LABORATOR

Scopul lucrării de laborator este de a-i ajuta pe studenți, majoritatea nefamiliarizați cu mediul de dezvoltare Arduino, să învețe să utilizeze placa Arduino Uno, familiarizându-se cu limbajul de programare Arduino C++. Vor învăța cum să scrie cod pentru a controla joystick-ul, matricea de LED-uri, senzorii, Buzzer-ul și ecranul OLED, gestionând astfel intrările și ieșirile sistemului. Aceasta va include concepte de bază precum funcțiile, variabilele și structurile de control (if-else, loop).

- Familiarizarea cu platforma Arduino IDE prin instalarea software-ului, configurarea acestuia și configurarea plăcii Arduino Uno.
- Înțelegerea structurii hardware a plăcii Arduino Uno, explicând pinii digitali, analogici, pinul de alimentare, și portul de comunicație USB.
- Introducerea microcontrolerului ATmega328P, subliniind rolul regiszrelor interne, precum registrul de date (Data Direction Register) și registrul de control (Control Register).
- Utilizarea regiszrelor pentru configurarea și citirea pinilor de intrare/ieșire.
- Explicarea conceptului de întreruperi și modul în care acestea sunt folosite pentru a reacționa la evenimente externe (de exemplu, schimbarea stării joystick-ului sau detectarea unui obstacol).
- Implementarea întreruperilor pentru a activa LED-urile sau buzzer-ul automat, fără a suprasolicita bucla principală (loop).
- Explicarea funcțiilor de bază din Arduino (de ex. digitalWrite, digitalRead, analogRead, analogWrite) și aplicarea lor în cadrul proiectului.
- Înțelegerea și includerea librăriilor externe în IDE-ul Arduino și integrarea lor în codul proiectului.
- Interfațarea și Programarea Componentelor Electronice:
 - **Joystick:** învățarea modului de citire a valorilor analogice și interpretarea poziției joystick-ului pentru a indica direcția.
 - **Matricea de LED-uri 8x8:** configurarea și controlul matricei pentru afișarea direcției de mers; utilizarea librăriilor pentru control.

- **Senzorul Ultrasonic**: înțelegerea principiului de funcționare și integrarea senzorului pentru măsurarea distanței față de obstacole.
- **Buzzer**: programarea unui buzzer pentru alerte sonore atunci când un obstacol este detectat la o distanță critică.
- **Senzorul de Luminozitate**: configurarea unui foterezistor pentru ajustarea automată a intensității LED-urilor în funcție de lumina ambientală.
- **Ecran OLED**: afișarea informațiilor despre distanță față de obstacole pe ecranul OLED.
- Introducerea unor tehnici de optimizare a codului Arduino, cum ar fi reducerea delay-urilor, utilizarea variabilelor globale pentru înregistrarea datelor critice și minimizarea încărcării procesorului.

2.1 REZULTATE AȘTEPTATE

La finalul laboratorului, studenții ar trebui să fie capabili să:

- Instaleze și configureze corect mediul de dezvoltare Arduino și să scrie cod pentru controlul componentelor electronice.
- Înțeleagă structura plăcii Arduino Uno și funcționalitatea regiștrilor și intreruperilor.
- Să programeze și să utilizeze senzori pentru a crea un sistem funcțional de semnalizare și detectare a obstacolelor pentru un vehicul alternativ.
- Să utilizeze eficient librăriile și să interpreteze informațiile de la senzori pentru a ajusta semnalizarea și intensitatea LED-urilor.
- Să testeze și să optimizeze performanța sistemului pentru a funcționa în condiții reale.

Fiecare pas de realizare a acestui proiect le va oferi studenților o experiență concretă de aplicare a cunoștințelor tehnice în crearea unui sistem complex și funcțional.

3 INTRODUCERE ÎN LUCRUL CU ARDUINO

Conform site-ului oficial, Arduino este o platformă open-source dedicată proiectelor de electronică și programare, destinată atât începătorilor, cât și profesioniștilor.

3.1 COMPONENTA HARDWARE

Componenta hardware este o placă fizică de mici dimensiuni (precum Arduino Uno folosită în această lucrare de laborator), echipată cu un microcontroller. Acest microcontroller este, practic, un mic calculator capabil să execute instrucțiuni simple, gestionând semnalele primite sau transmise la diversi senzori și actuatori. Arduino are o varietate de pini digitali și analogici, care permit conectarea componentelor externe, cum ar fi senzori, LED-uri, motoare și ecrane. Există mai multe modele de plăci Arduino, fiecare având caracteristici și capabilități diferite, potrivite pentru diverse proiecte:

- **Arduino Uno** – cea mai populară și ușor de folosit placă, potrivită pentru aplicații generale.
- **Arduino Nano** – o versiune miniaturizată a lui Uno, ideală pentru spații restrânse.
- **Arduino Mega** – are mai mulți pini și memorie, potrivit pentru proiecte mai complexe.

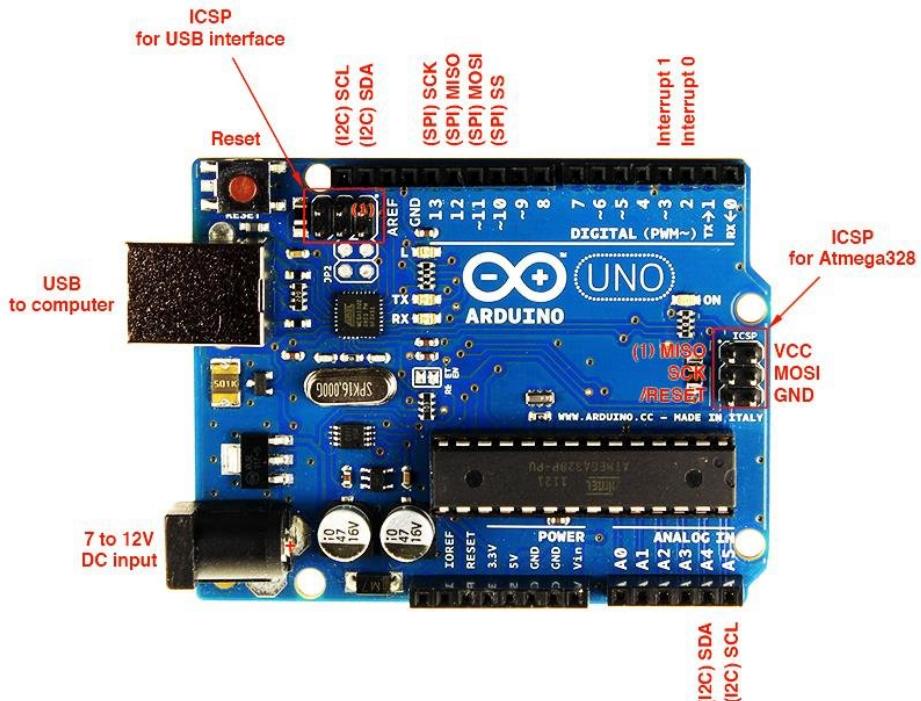
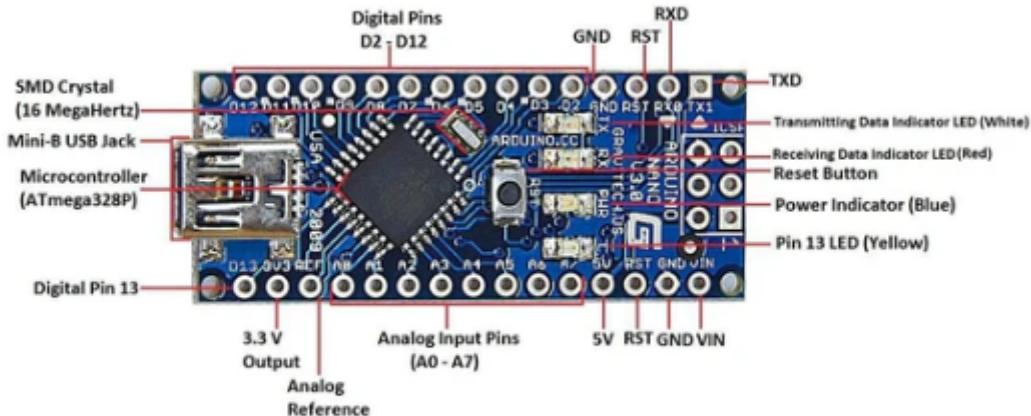


Figure 1. Placa de dezvoltare Arduino Uno



Arduino Nano V3.0 Pinout

www.Robotbanao.com

Figure 2. Placa de dezvoltare Arduino Nano

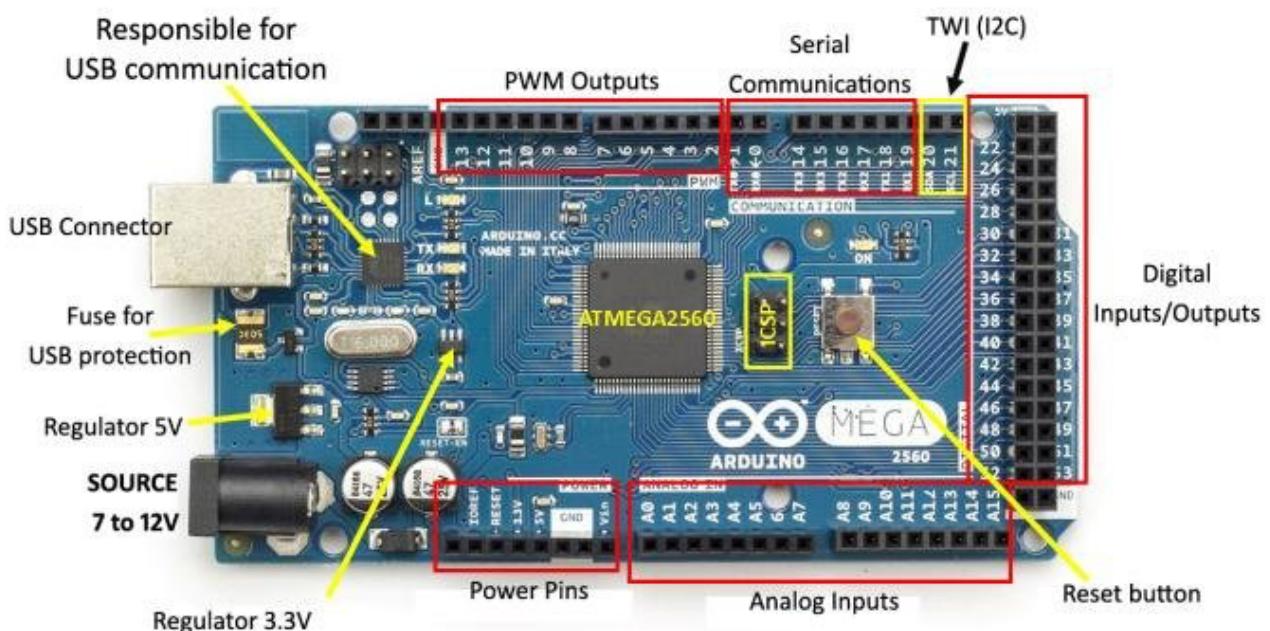


Figure 3. Placa de dezvoltare Arduino Mega

Pentru realizarea acestui proiect se va folosi placa de dezvoltare Arduino Uno deoarece este o placă ușor de configurat, cu o documentație cuprinzătoare, compatibilă cu senzorii și modulele utilizate și, totodată, are suficienți pini digitali și analogici pentru a controla componentele necesare.

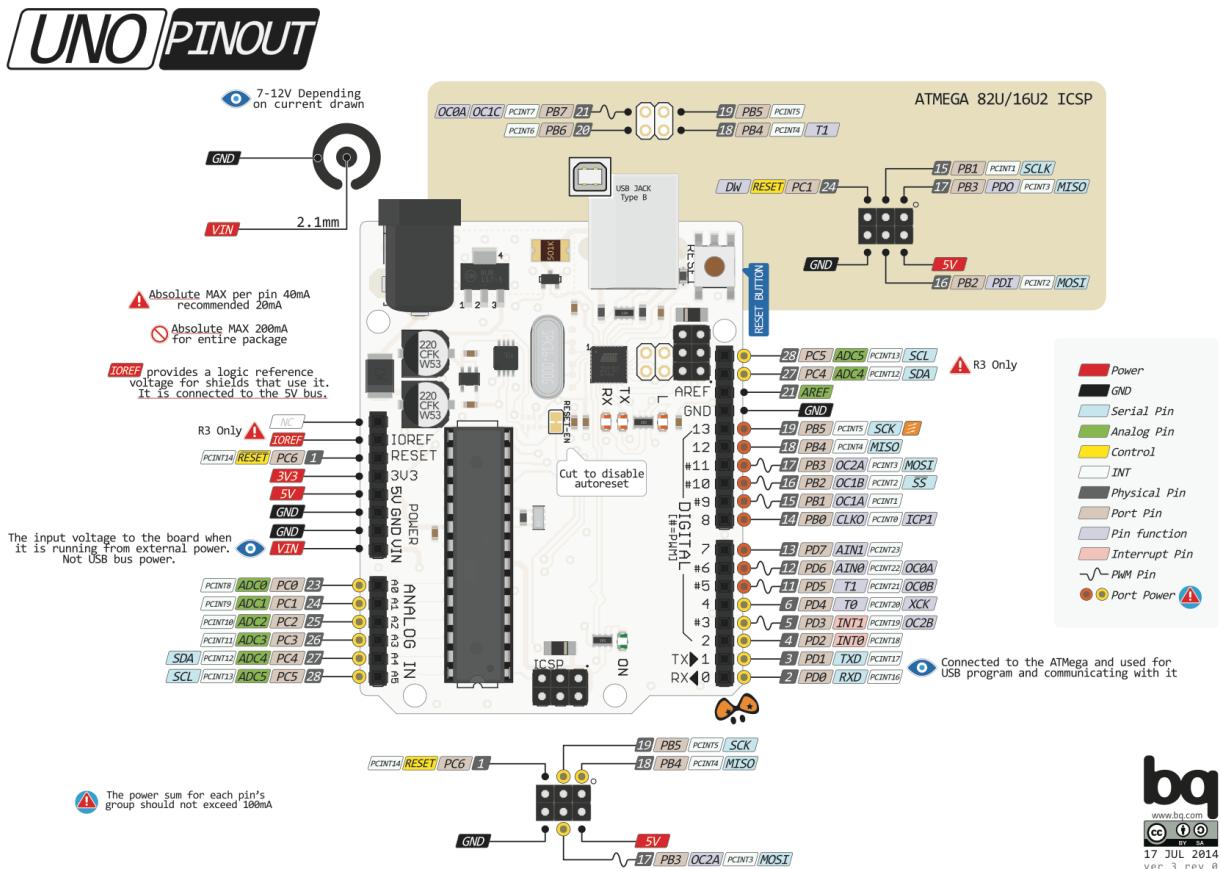


Figure 4. Descrierea pinilor placii Arduino Uno

3.2 ATMEGA328P

Microcontroller-ul ATmega328P de pe placă Arduino Uno este un cip cu arhitectură pe 8 biți, bazat pe familia de microcontrolere AVR de la Atmel, optimizat pentru aplicații embedded.

ATmega328P dispune de trei porturi de intrare/iesire digitale:

- **Port B**: corespunzător pinii 8-13 de pe placă Arduino (digitali).
 - **Port C**: corespunzător pinii analogici A0-A5 (intrări analogice).
 - **Port D**: corespunzător pinii digitali 0-7.

Fiecare port are trei registri principali pentru configurare și control:

- 1) **DDR (Data Direction Register)**: Controlează direcția fiecărui pin (intrare sau ieșire). În Arduino, este echivalentul funcției `pinMode()`. Setarea unui bit la 1 configura pinul ca ieșire. Setarea unui bit la 0 configura pinul ca intrare.
 - 2) **PORT**: Controlează starea pinilor setați ca ieșiri și activează rezistențelor de pull-up pentru pini de intrare. La ieșire, setarea unui bit la 1 înregistrează un nivel HIGH, iar 0 un nivel LOW. La intrare, activarea unui bit în PORT permite activarea rezistențelor interne de pull-up.
 - 3) **PIN**: Folosit pentru a citi starea pinii de intrare. Fiecare bit al acestui registru corespunde stării de pe un anumit pin și poate fi doar citit.

Registri specifici porturilor:

- **PORTD**: include registrele DDRD, PORTD, si PIND pentru controlul pinilor 0-7.

- **PORTB**: include registrele DDRB, PORTB, și PINB pentru controlul pinilor 8-13.
- **PORTC**: include registrele DDRC, PORTC, și PINC pentru controlul intrărilor analogice.

Fiecare bit al acestor registri reprezintă un pin specific al portului, oferind un control precis la nivel de hardware, util pentru aplicațiile care necesită tempi de răspuns rapid și optimizări la nivel de consum de resurse.

Microcontroller-ul ATmega328P include suport pentru întârzieri externe, comunicare serială, PWM (Pulse Width Modulation) și un timer hardware, toate esențiale pentru implementarea aplicațiilor embedded de timp real, cum ar fi cele din acest proiect.

3.3 COMPOENTA SOFTWARE: ARDUINO IDE

Platforma software, cunoscută sub numele de Arduino IDE (Integrated Development Environment), este utilizată pentru a scrie codul care va rula pe placa Arduino. Arduino IDE folosește un limbaj de programare derivat din C și C++ și este dotat cu numeroase funcții predefinite care simplifică accesul la componentele hardware. După ce codul este scris și compilat în IDE, acesta poate fi încărcat pe placa Arduino prin intermediul unui cablu USB.

Un proiect Arduino funcționează prin procesul de interacțiune între microcontroller și componentele conectate. Codul scris în Arduino IDE este trimis către microcontroller, care rulează instrucțiunile pentru a controla componentele externe. În esență, microcontroller-ul interpretează semnalele pe care le primește de la senzori sau le trimit către actuatori pentru a realiza funcționalitatea dorită.

3.4 INSTALARE ȘI UTILIZARE ARDUINO

Se poate folosi atât versiunea online, cât și versiunea de desktop a platformei Arduino IDE.

- **Arduino Web Editor**
 - La link-ul următor : <https://www.arduino.cc/en/software> se selectează butonul pentru navigarea către editorul web
 - Se creează un cont (dacă nu există deja)
 - Instalare Arduino Plugin (Pentru a permite IDE-ului online să comunice cu placa Arduino prin USB, va trebui să instalezi un mic plugin. Instrucțiunile apar pe ecran, iar instalarea este simplă și rapidă)
 - Conectează Placa Arduino prin USB
 - Scrie codul în interfața online, apasă pe „Verify” pentru a compila, și apoi pe „Upload” pentru a încărca codul pe placă.

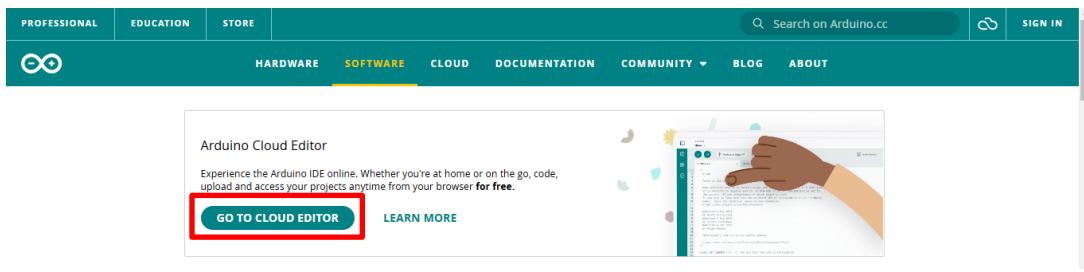


Figure 5. Accesarea platformei Arduino Web Editor

- **Arduino IDE pentru Desktop**

- De la link-ul de mai sus se descarcă versiunea compatibilă cu sistemul de operare utilizat
- Pe Windows: rulează fișierul .exe și urmează instrucțiunile de instalare. Acceptă toate permisiunile necesare și asigură-te că selectezi opțiunea de a instala driverele pentru Arduino.
- După instalare, deschide Arduino IDE. Vei vedea o interfață simplă, cu o zonă de editare pentru cod și butoane de bază pentru compilare și încărcare.

Atât versiunea desktop cât și cea online au funcționalități similare:

- **Verify (Verificare)** – verifică sintaxa codului pentru eventualele erori.
- **Upload (Încărcare)** – încarcă codul pe placa Arduino conectată.
- **Serial Monitor** – permite vizualizarea datelor transmise și recepționate de placa Arduino (foarte util pentru debug).

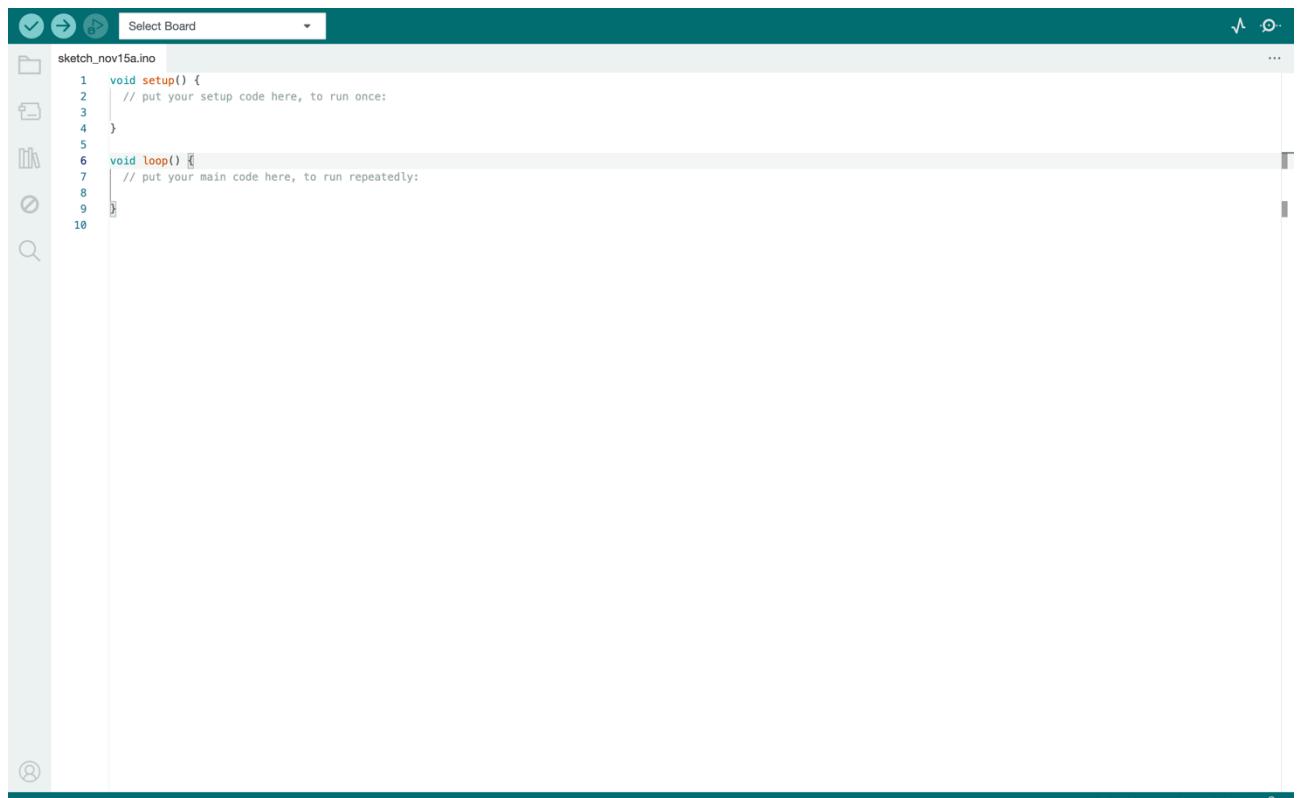


Figure 6. Interfața Arduino IDE pentru Desktop

4 DESCRIEREA COMPONENTELOR

În acest capitol, vom prezenta principalele componente utilizate în proiectul nostru, explicând funcționalitatea și specificațiile tehnice ale fiecărei piese. Aceste detalii sunt esențiale pentru a înțelege cum interacționează componentele în ansamblu și contribuie la funcționarea corectă a proiectului. Fiecare componentă are un rol specific în sistem, fie că este responsabilă pentru interacțiunea utilizatorului, măsurarea unor parametri, sau afișarea informațiilor.

Arduino Uno (cu microcontroller ATmega328P)

- **Tensiune de alimentare:** 5V prin portul USB sau 7-12V prin pinul de alimentare extern.
- **Tip semnal:** Acceptă atât semnale digitale (pe pinii 0-13), cât și analogice (pe pinii A0-A5).
- **Pini I/O:** 14 pini digitali (dintre care 6 pot fi folosiți pentru PWM) și 6 intrări analogice.
- **Frecvență de operare:** 16 MHz, ce oferă performanță suficientă pentru majoritatea aplicațiilor embedded.
- **Memorie:** 32KB Flash, 2KB SRAM, și 1KB EEPROM.
- **Dimensiuni:** Aproximativ 68.6mm x 53.4mmz.

Modul Joystick

- **Tensiune de alimentare:** 5V.
- **Tip semnal:** Analogic.
- **Pini:**
 - Rx: Semnal analogic pentru mișcarea pe axa X.
 - Ry: Semnal analogic pentru mișcarea pe axa Y.
 - SW: Buton de apăsare (de obicei utilizat pentru a detecta o apăsare a joystick-ului, activând un semnal digital).
 - GND: Pinul de masă.
 - VCC: Pinul de alimentare.
- **Dimensiuni:** Compact, variază între 2-4 cm în diametru.
- **Rază de mișcare:** Mișcările joystick-ului sunt limitate într-un domeniu de aproximativ $\pm 45^\circ$ pe fiecare axă, oferind o gamă largă de valori de la 0 la 1023 (pentru o rezoluție de 10 biți pe fiecare axă).



Figure 7. Modul Joystick

Matrice de LED-uri 8x8

- **Tensiune de alimentare:** 5V.
- **Tip semnal:** Digital.
- **Control digital:** Utilizează un circuit de control MAX7219, care simplifică interfațarea prin reducerea numărului de pini necesari pentru control. Modulul are următorii pini:
 - VCC (se conectează la sursa de alimentare),
 - GND (se conectează la masa),
 - DIN (pinul prin care se transmite informația către modulul MAX7219),
 - CS (Folosit pentru a selecta dispozitivul, atunci când mai multe dispozitive sunt conectate la aceeași magistrală SPI),
 - CLK (Semnal de sincronizare pentru transferul de date).
- **Interfață:** Comunică printr-un protocol de tip SPI, facilitând integrarea rapidă cu plăci de dezvoltare precum Arduino.
- **Frecvență de actualizare:** Controlată de Arduino prin semnal PWM pentru fiecare LED; viteza de actualizare este de obicei de câțiva kilohertzi pentru a preveni efectul de pâlpâire.
- **Dimensiuni:** În general, aproximativ 3x3 cm pentru matricea de bază 8x8.



Figure 8. Matrice de LED-uri 8x8

Senzor Ultrasonic (HC-SR04)

- **Tensiune de alimentare:** 5V.
- **Tip semnal:** Digital.
- **Pini:** Are 4 pini – VCC, Trig, Echo și GND.
- **Frecvență de operare:** 40 kHz pentru unda sonoră.
- **Nivel de sensibilitate:** Detectează obstacole între 2 cm și 4 m cu o precizie de aproximativ 3 mm.
- **Dimensiuni:** Aproximativ 4x2 cm.



Figure 9. Senzor ultrasonic

Buzzer

- **Tensiune de alimentare:** 3.3V - 5V.
- **Tip semnal:** Digital.
- **Pini:** VCC, GND, I/O (pin de semnal pentru controlul buzzer-ului).
- **Curent consumat:** Consumă un curent redus, între 5-30 mA, în funcție de tensiunea aplicată.
- **Funcționalitate:** Sunetul generat are o frecvență fixă, în jur de 2-4 kHz, suficient de puternic pentru a atrage atenția.
- Este un buzzer activ, ceea ce înseamnă că produce sunet atunci când primește un semnal logic (HIGH).



Figure 10. Buzzer

Fotorezistență (LDR - Light Dependent Resistor)

- **Tensiune de alimentare:** 3.3V - 5V.
- **Tip semnal:** Modulul trimite un semnal analogic care variază în funcție de intensitatea luminii ambientale.
- **Pini:** VCC, GND, S (ieșire analogică, conectată la un pin analogic pe Arduino pentru citirea variațiilor de lumină).
- **Interval de măsurare:** Detectează o gamă largă de intensități de lumină, de la întuneric complet până la lumina solară puternică.
- **Nivel de sensibilitate:** Fotorezistența schimbă rezistența electrică în funcție de cantitatea de lumină incidentă. Cu cât lumina este mai intensă, cu atât rezistența este mai mică, iar semnalul de ieșire crește.

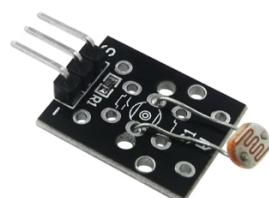


Figure 11. Modul cu fotorezistență

Ecran OLED

- **Tensiune de alimentare:** 3.3V - 5V.
- **Interfață de comunicație:** I2C .
- **Rezoluție:** 128 x 64 pixeli.
- **Dimensiunea ecranului:** 0.96"
- **Pini:** VCC, GND, SCL (semnal de ceas pentru comunicația I2C), SDA (semnal de date pentru comunicația I2C).
- **Putere consumată:** 0.08W.

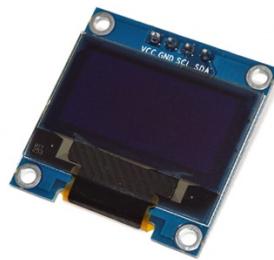


Figure 12. Ecran OLED

5 STRUCTURA PROIECTULUI

5.1 SCHEMA LOGICĂ

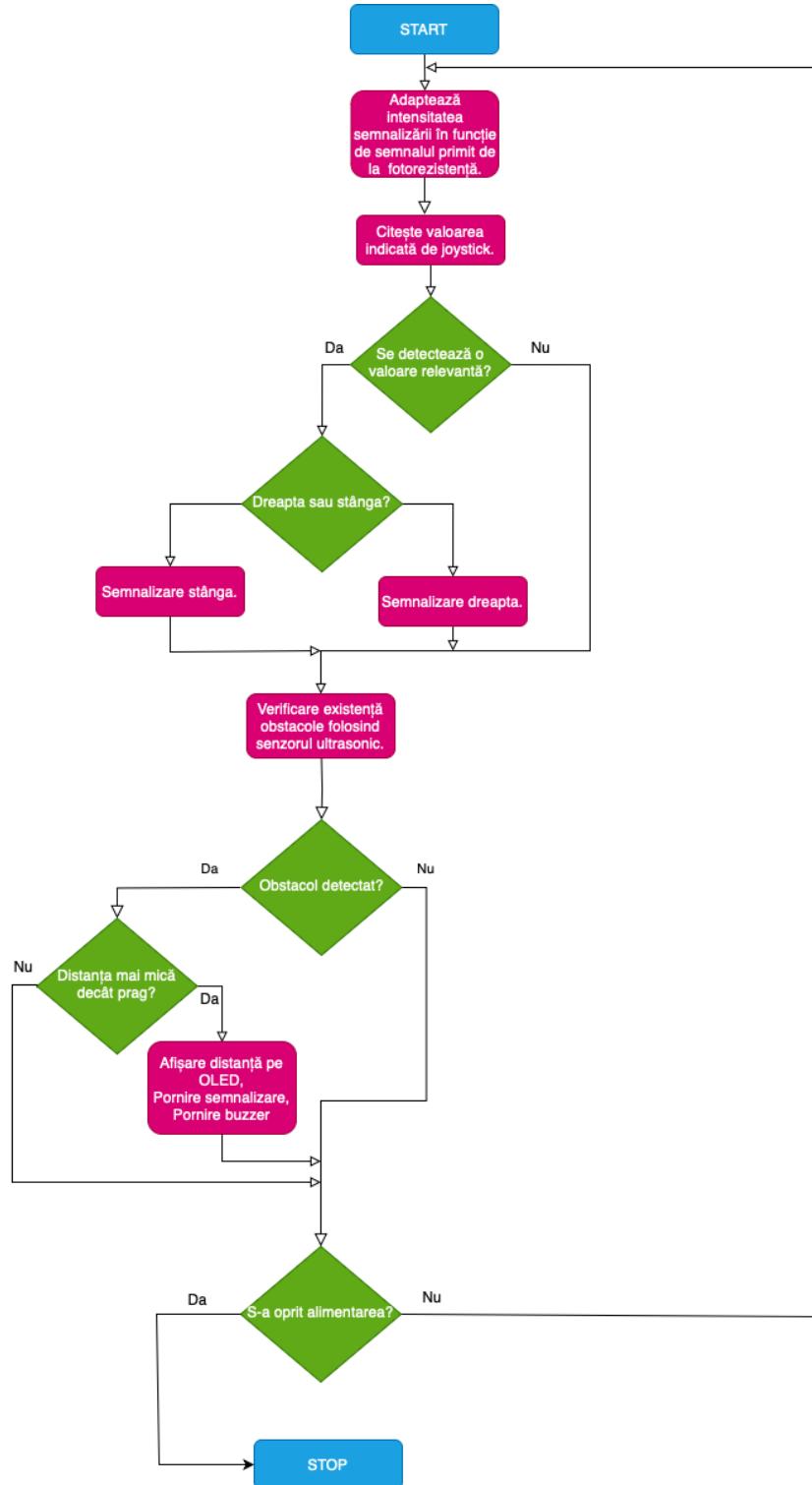


Figure 13. Schema logică a proiectului

Sistemul propus reprezintă o soluție intelligentă pentru îmbunătățirea siguranței și accesibilității unui mijloc de transport alternativ (precum o bicicletă). Acesta integrează mai multe componente electronice pentru a facilita semnalizarea direcției de mers, detectarea obstacolelor și adaptarea semnalelor vizuale în funcție de condițiile de iluminare. Funcționarea sistemului urmează pașii logici detaliați în schemă:

1. **Pornirea sistemului** - Sistemul este inițiat prin alimentarea cu energie electrică de la o sursă adecvată. După pornire, toate componentele devin funcționale și gata să proceseze informațiile primite de la senzori.
2. **Adaptarea intensității semnalelor luminoase** - Un senzor de luminositate (fotorezistență) măsoară condițiile de iluminare ambientală. Semnalul analogic obținut este utilizat pentru a regla automat intensitatea LED-urilor de semnalizare, asigurând vizibilitatea optimă pe timp de noapte sau în condiții de vizibilitate redusă.
3. **Intervenția joystick-ului** – Sistemul citește poziția joystick-ului amplasat pe ghidon. Aceasta permite utilizatorului să indice manual direcția de mers.
4. **Detectarea obstacolelor prin senzor ultrasonic** - Sistemul utilizează un senzor ultrasonic amplasat în partea frontală pentru a detecta obiectele din proximitate. Undele sonore sunt emise și măsurate la revenire, calculând astfel distanța față de un obstacol. Dacă nu există obstacol, sistemul continuă să monitorizeze. Dacă se detectează un obstacol, distanța calculată este comparată cu un prag prestabilit.
5. **Reacția la obstacole** - Dacă distanța față de obstacol este mai mică decât pragul setat, sistemul reacționează prin:
 - Aprinderea unui mesaj vizual pe ecranul OLED, indicând distanța până la obstacol.
 - Activarea unui buzzer pentru a alerta utilizatorul.
 - Aprinderea LED-urilor pentru a atrage atenția asupra situației.
6. **Încheierea procesului** - După executarea unei acțiuni, sistemul revine în modul de monitorizare continuă, asigurând funcționarea repetitivă și în timp real. Acest proces are loc atât timp cât sistemul este alimentat.

5.2 SCHEMA ELECTRICĂ

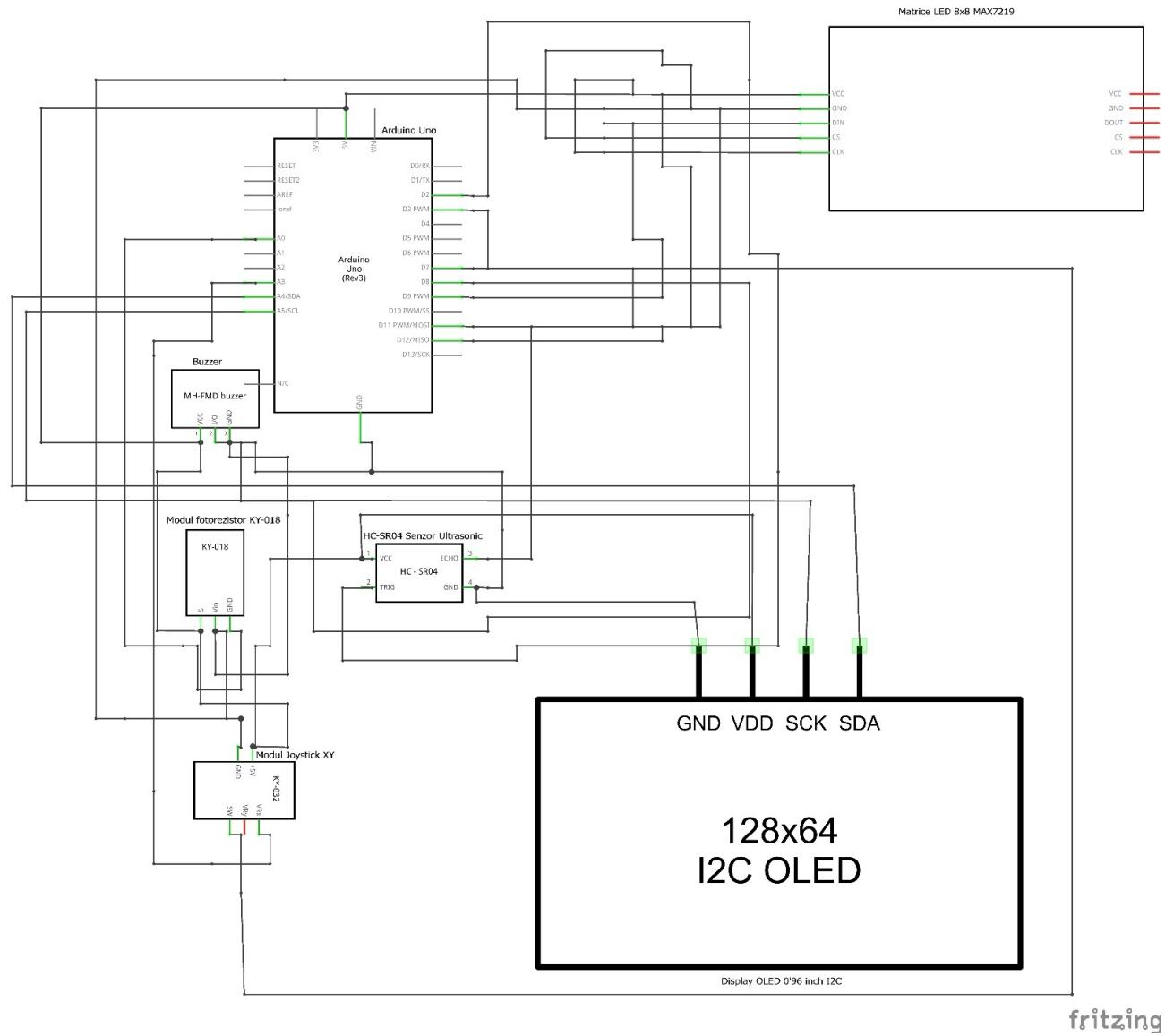


Figure 14. Schema electrică

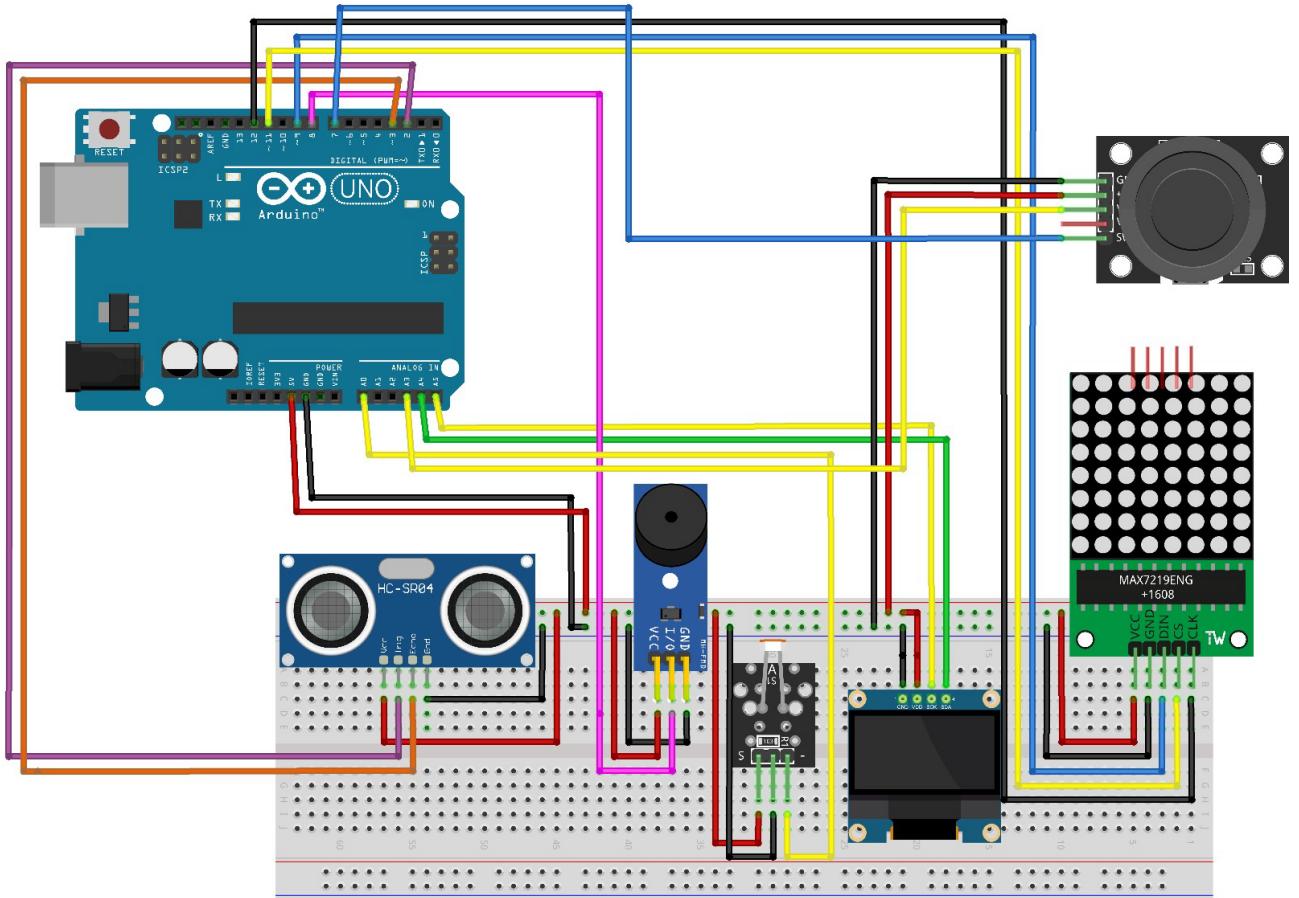


Figure 15. Simularea interconectării componentelor sistemului

Senzor Ultrasonic

Trig - purple

Echo - Orange

Senzor fotorezistor

Pin Analogic - yellow

OLED Display

SCL - Yellow (Color Standard, dar pe proiectul fizic este Green)

SDA - Green (Color Standard, dar pe proiectul fizic este Black)

Buzzer

Pin analogic - pink

Joystick

SW - Blue

VRX - Yellow

Matrice

DIN - Blue

CS - Yellow

CLK - Black

Figure 16. Legenda pentru firele de conectare

5.3 ETAPE DE CONECTARE A COMPONENTELOR SISTEMULUI

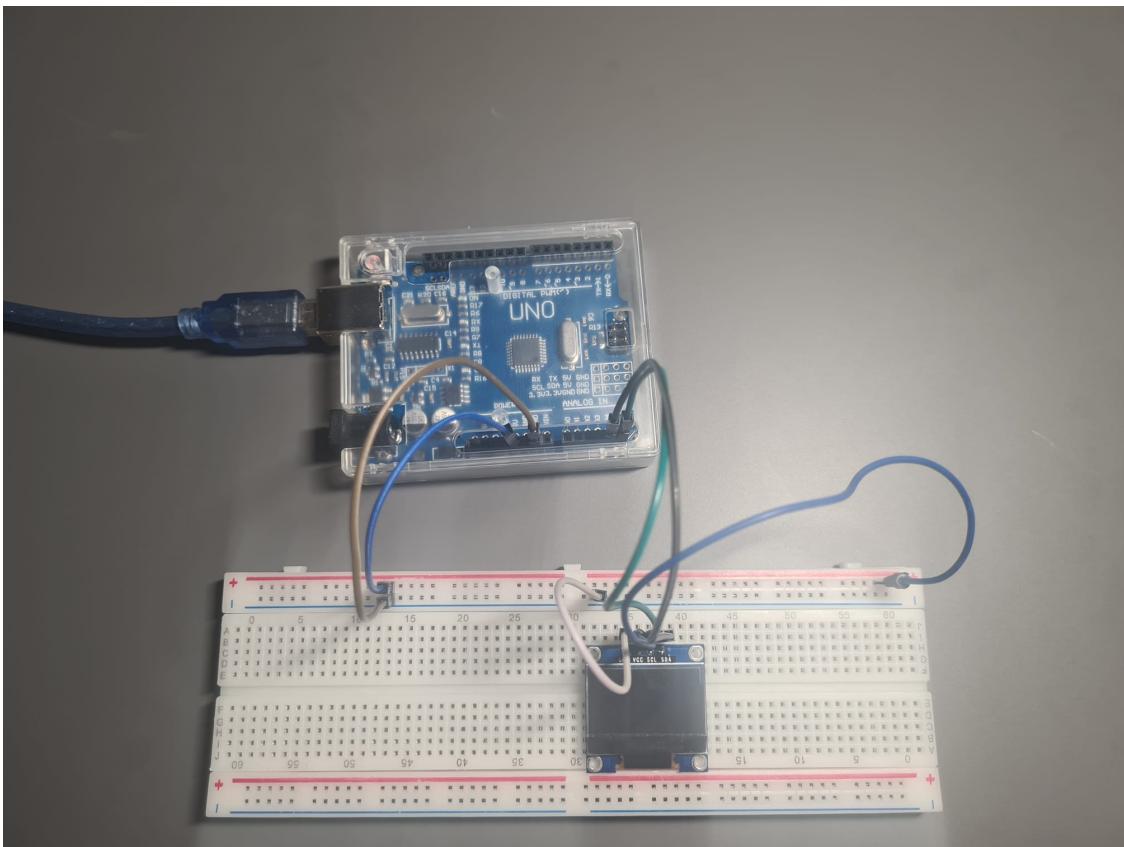


Figure 17. Etapa 1 - conectarea ecranului OLED

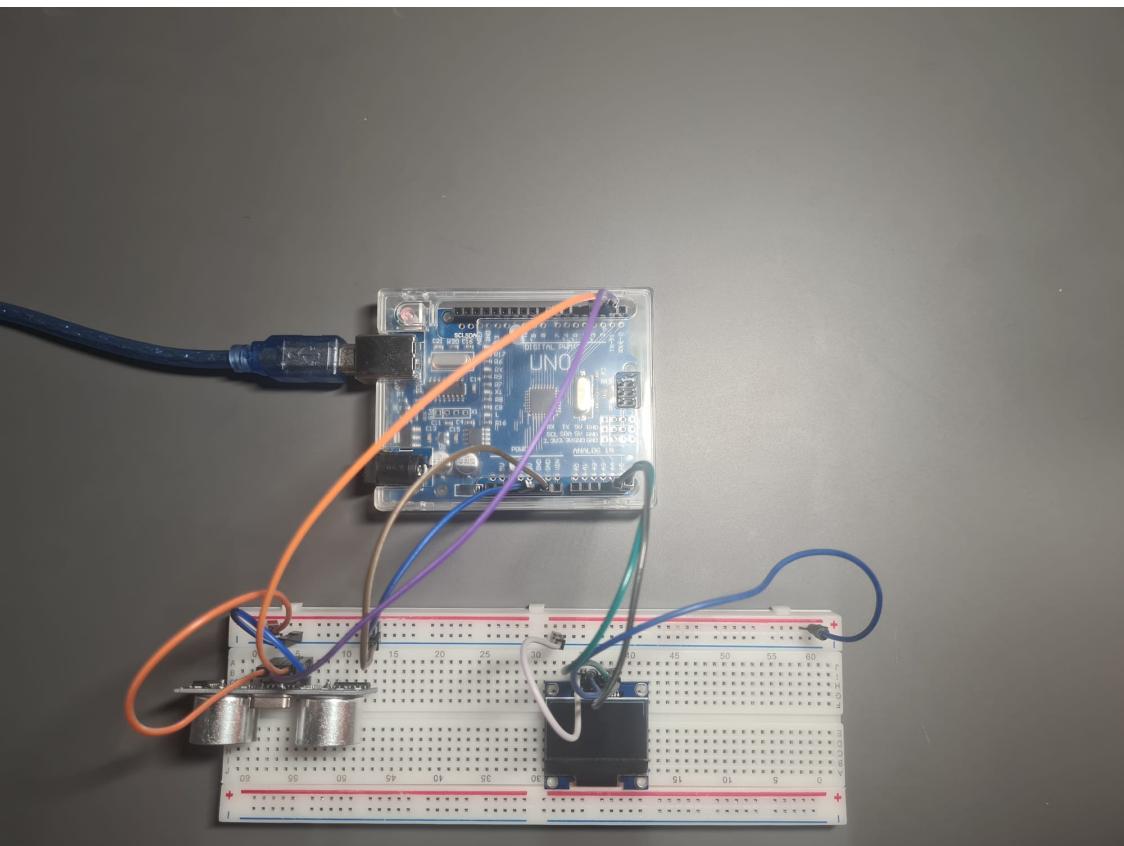


Figure 18. Etapa 2 - conectarea senzorului ultrasonic

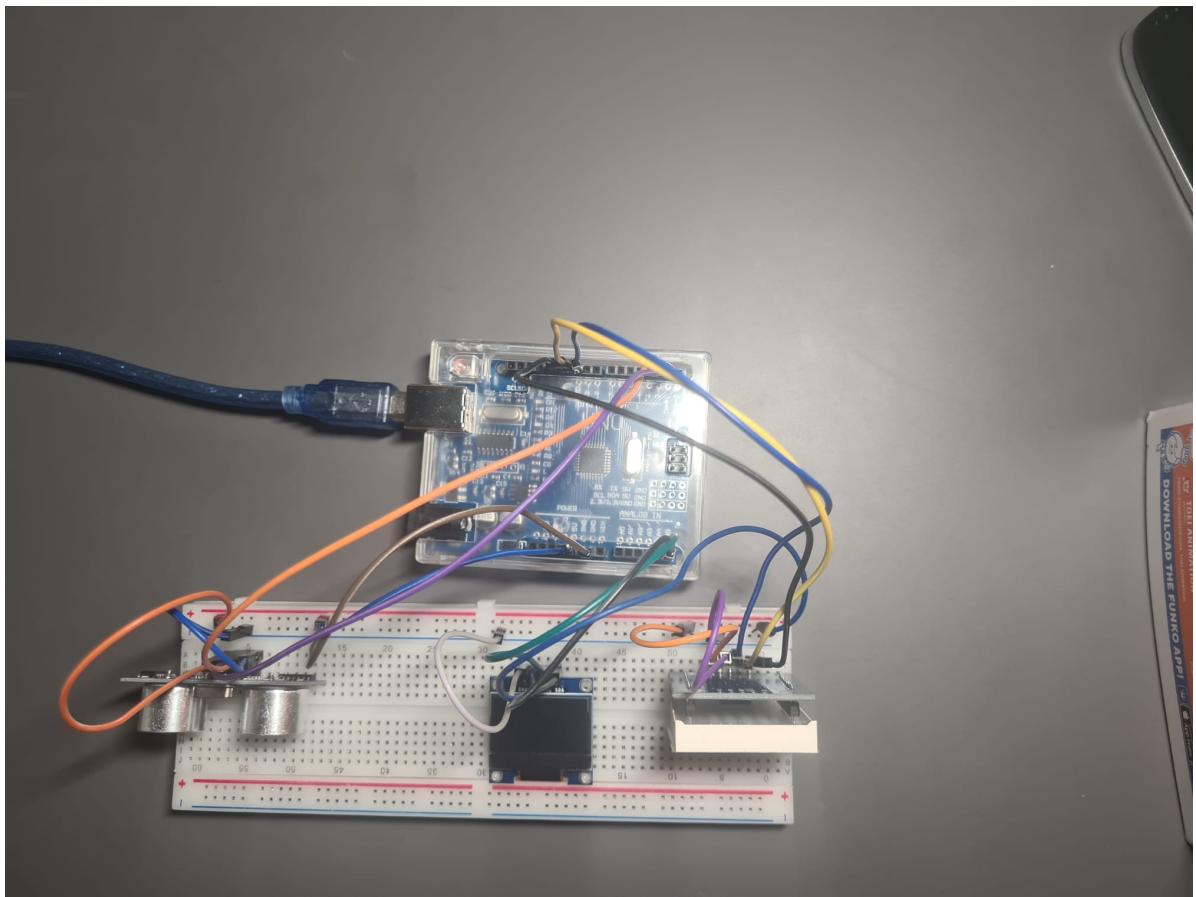


Figure 19. Etapa 3 - conectarea matricii de 8x8 led-uri

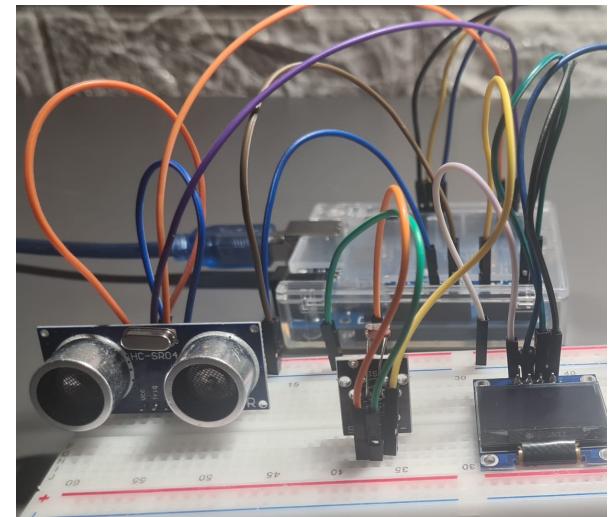
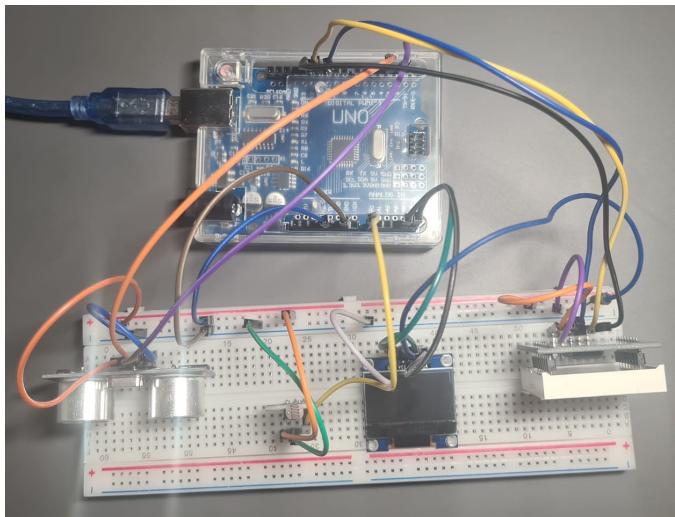


Figure 20. Etapa 4 - conectarea modulului cu fotorezistență

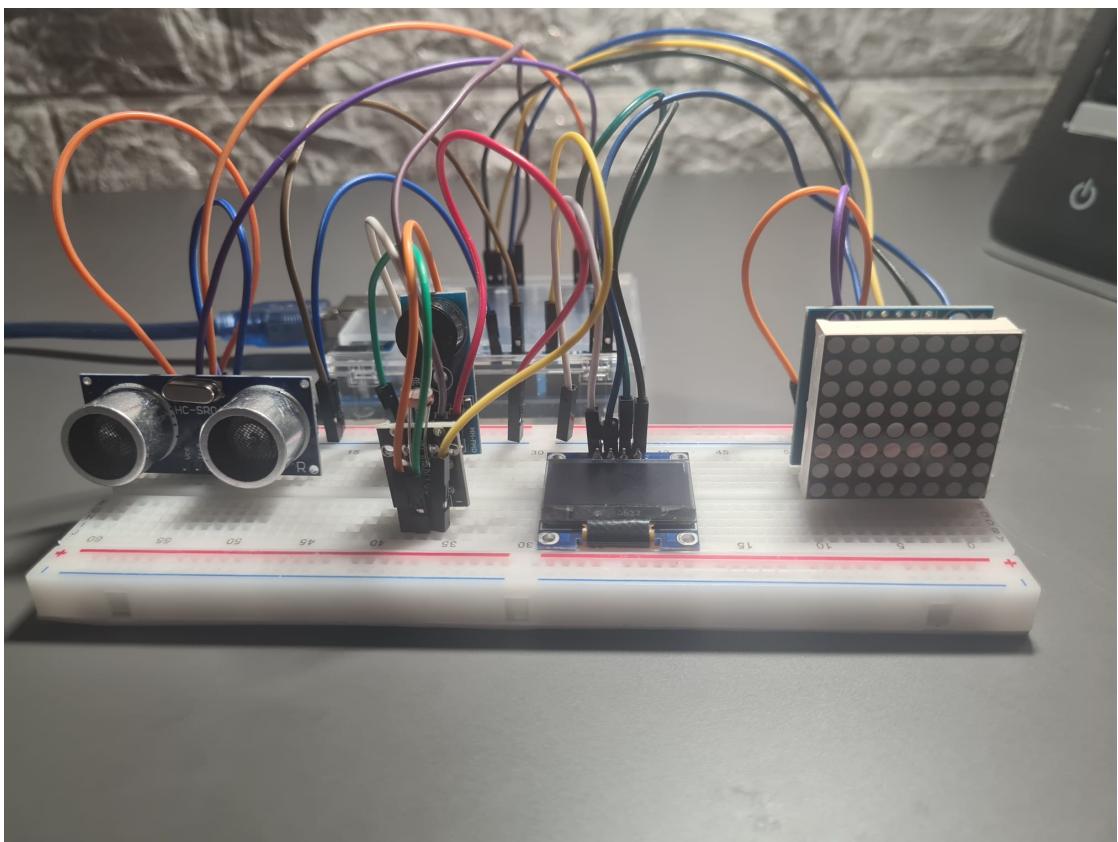


Figure 21. Etapa 5 - conectarea buzzer-ului

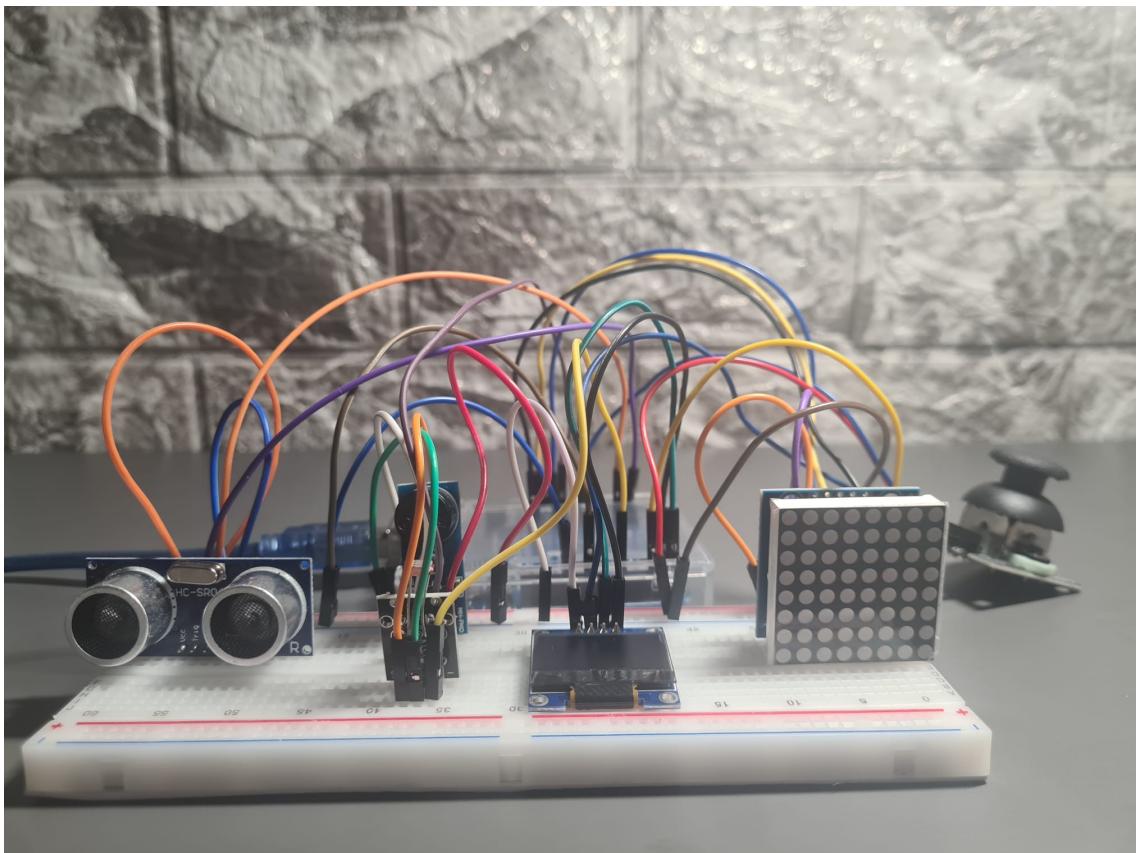


Figure 22. Etapa 6 - conectarea joystick-ului

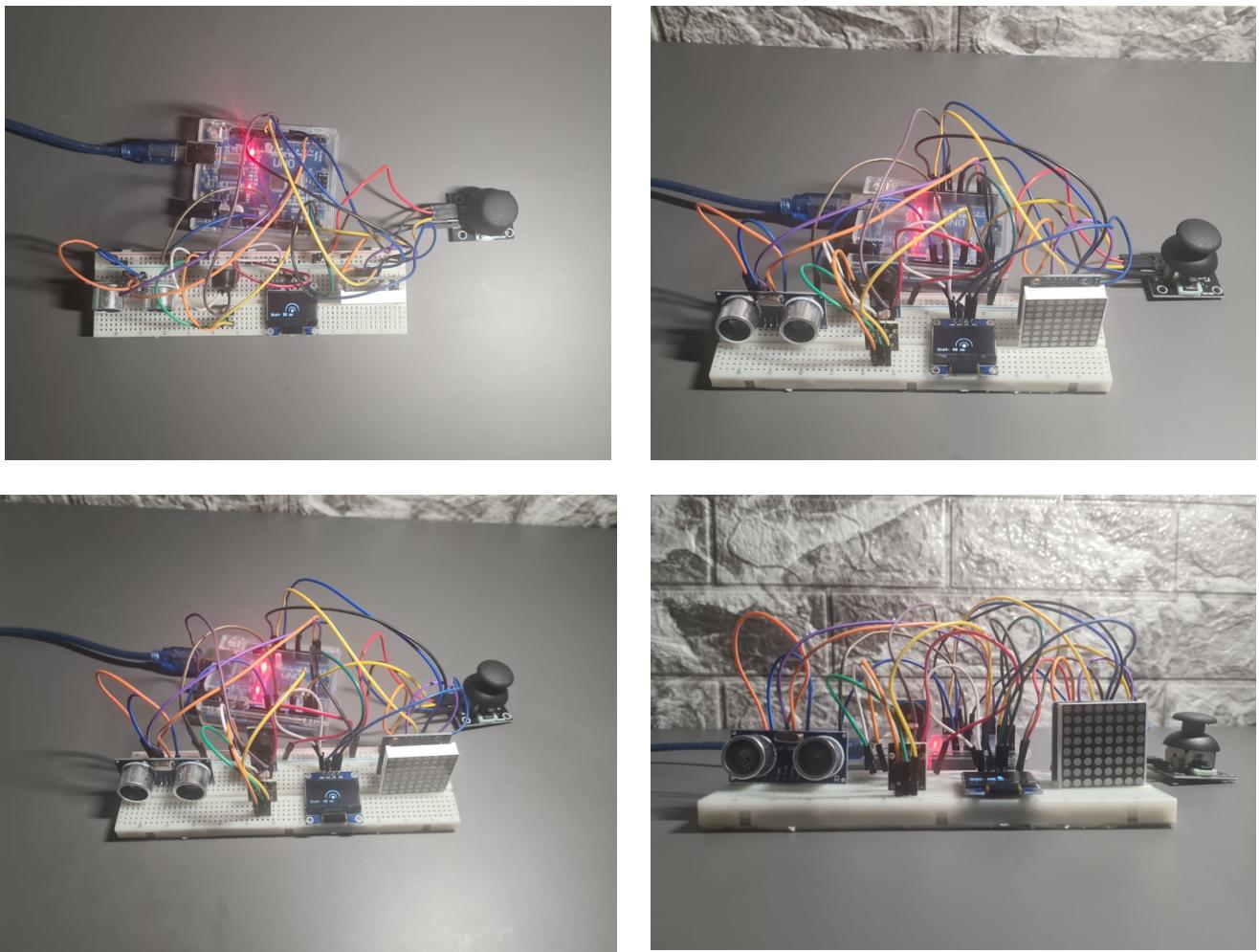


Figure 24. Reprezentarea fizică a sistemului

Pentru a putea vedea mai în detaliu modul se conectare al componentelor și funcționalitatea proiectului, se poate accesa următorul link: <https://youtu.be/JKuHA6htxtk>

De asemenea, în capitolul [Realizarea lucrării de laborator](#) se propune o serie de exerciții din care ulterior va rezulta structura finală a proiectului și se va explica pas cu pas modul de conectare a fiecărei componente.

6 RESURSELE INTERNE DIN MICROCONTROLLER (TIMER, REGIȘTRII, ÎNTRERUPERI, CAD)

În cadrul acestui proiect, am utilizat regiștrii hardware ai microcontrolerului ATmega328P pentru a controla funcționarea unui buzzer conectat la pinul digital 8 al plăcii Arduino Uno. Prin acces direct la registrele microcontrolerului și configurarea Timer-ului 1 în modul CTC (Clear Timer on Compare Match), am generat un semnal PWM necesar activării buzzer-ului.

Acest tip de abordare oferă o înțelegere profundă a funcționării hardware-ului, permitând o gestionare eficientă a resurselor microcontrolerului fără a depinde de funcții abstractizate din biblioteci predefinite.

Descrierea codului cu regiștrii al proiectului se află în capitolul următor, la secțiunea [Buzzer](#).

7 REALIZAREA LUCRĂRII DE LABORATOR

Scopul acestui capitol este de a realiza în format fizic sistemul de semnalizare pentru mijloacele de transport alternativ, pornind de la înțelegerea fiecărei componente în parte și prezentarea unei secțiuni de cod pentru fiecare, urmată de implementarea propriu-zisă și realizarea unor sarcini de lucru pentru a facilita învățarea și segmentarea informațiilor. Ulterior, prin rezolvarea sarcinilor de lucru și unirea secțiunilor de cod, se va obține un sistem funcțional.

7.1 BUZZER

În această lucrare de laborator, vom învăța cum să controlăm un buzzer utilizând regiștrii microcontrolerului. Buzzer-ul este un dispozitiv simplu care produce un sunet de frecvență fixă sau variabilă, fiind frecvent utilizat pentru alerte sau notificări sonore în diverse aplicații. Scopul acestui exercițiu este de a înțelege cum se configurează și se controlează hardware-ul prin acces direct la regiștrii microcontrolerului, fără a folosi funcții abstractizate din biblioteci predefinite.

Codul prezentat controlează un buzzer prin configurarea timer-ului 1 al microcontrolerului ATmega328P.

Cerință:

Configurați și controlați un buzzer utilizând **Timer-ul 1** al microcontrolerului ATmega328P, accesând direct registrele hardware.

- Pinul I/O al buzzer-ului va fi conectat la pinul digital 8 al placii Arduino Uno (PB0 conform structurii de pini a microcontroler-ului), care va fi setat ca ieșire.
- Setați timer-ul 1 în modul CTC (Clear Timer on Compare Match) pentru a genera un semnal PWM și folosiți prescalar 1.
- Urechea umană poate percepse sunete cu frecvențe cuprinse între 20Hz și 20MHz. Setați valoarea registrului OCR1A pentru a genera un semnal cu o frecvență de 300Hz.

Pentru rezolvarea acestui exercițiu se va folosi documentația microcontroler-ului ATmega328P, pagina 108.

Componente necesare:

- Buzzer
- Arduino Uno

- Breadboard
- Fir de conectare

Mod de conectare:

Buzzer-ul se conectează la breadboard pe linia C, pozițile 38 (VCC), 37 (I/O), 36 (GND).

Se conectează un fir de la D 38 la linia de alimentare (+).

Se conectează un fir de la D 37 la pinul digital 8 al plăcii Arduino Uno.

Se conectează un fir de la D 36 la linia de ground (-).

Pentru ușurință urmăriți [figura 16](#).

Cod:

```

1. // funcția de configurare a microcontroler-ului la pornirea sistemului
2. void setup() {
3.   setup_timer();
4. }
5.
6. // inițializează timer-ul pentru a genera semnalul necesar buzzer-ului
7. void setup_timer()
8. {
9.   DDRB |= 1<<0;           // configurează pinul PB0 ca ieșire pentru a genera semnalul PWM
10.  TCCR1A = 1 << 6;        // OCR1A își va schimba valoare când timer-ul ajunge la valoarea registrului
OCR1A
11. TCCR1B = 1 << 3;        // se activează modul CTC TCCR1B = 00001000;
12. TCCR1B|= 1 << 0;        // se pornește timer-ul cu prescaler = 1
13. OCR1A = 26560;          // o iteratie de numărare va dura 1,66 ms, cu prescaler de 1
14. }
15.
16. void loop() {
17.   TCCR1B &= ~(1 << 0); // TCCR1B = TCCR1B & 11111110, dezactivează temporizatorul oprind
contorizarea
18.   delay(1000);          // așteaptă 1 secundă
19.   TCCR1B |= 1 << 0;     // reactivează temporizatorul
20.   delay(1000);          // așteaptă 1 secundă
21. }
22.

```

Explicație:

Conform structurii celor doi registrii de control ai Timer-ului 1, pentru a se seta modul CTC se va seta bitul WGM12 (pe poziția 3) al registrului TCCR1B în 1, care resetează temporizatorul la fiecare comparație egală cu OCR1A. De asemenea, pentru a seta prescalatorul 1 se va seta bitul CS10 (aflat pe poziția 0) al registrului TCCR1B în 1, ceea ce înseamnă că temporizatorul va număra fiecare impuls de ceas.

15.11.1 TCCR1A – Timer/Counter1 Control Register A

Bit (0x80)	7	6	5	4	3	2	1	0	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 23. Structura registrului TCCR1A

15.11.2 TCCR1B – Timer/Counter1 Control Register B

Bit (0x81)	7	6	5	4	3	2	1	0	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 24. Structura registrului TCCR1B

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1X at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, phase correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, phase correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, phase correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX

Figure 25. Tabel ajutător pentru setarea modului CTC

Table 15-6. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk _{I/O} /1 (no prescaling)

Figure 26. Tabel ajutător pentru setarea prescalatorului

Pentru a calcula valoarea de comparație pentru care se va genera un semnal de ieșire de 300Hz, se realizează următoarele calcule (16MHz – frecvența microcontrolerului):

$$f = 300 \text{ Hz} \Rightarrow T = \frac{1}{f} = \frac{1}{300 \text{ Hz}} = \frac{1000 * 10^{-3}}{300} \text{ ms} = 3,33 \text{ ms}$$

$$\begin{aligned} \text{semiperioada} &= \frac{T}{2} = 1,66 \text{ ms} \\ 16 * 10^6 \text{ tacte} \dots & 1s = 1000ms \\ x \text{ tacte} \dots & 1,66ms \\ x &= \frac{16 * 10^6 * 1,66}{1000} = 26560 \end{aligned}$$

Sarcini de lucru:

- Modificați valoarea registrului OCR1A pentru a schimba frecvența sunetului produs de buzzer în 500Hz.
- Modificați codul pentru a produce o secvență de sunete diferite, de exemplu, simulând o mică melodie.

7.2 MATRICE DE 8X8 LED-URI

În această lucrare de laborator, vom învăța cum să controlăm o matrice LED de 8x8 utilizând modulul MAX7219 și microcontrolerul ATmega328P. Matricele LED sunt utilizate frecvent în aplicații de afișare, precum ceasuri digitale, panouri publicitare și jocuri. Scopul acestui

exercițiu este de a înțelege cum să configurăm și să controlăm afișajul prin utilizarea funcțiilor dedicate, lucrând cu matrici de date și bucle.

Codul prezentat controlează o matrice LED de 8x8 pentru a afișa diverse forme, inclusiv un triunghi, prin manipularea individuală a LED-urilor.

Explicarea funcțiilor din biblioteca LedControl.h:

```
1. #include <LedControl.h> // Biblioteca care ajută la controlul modulului MAX7219
2.
3. // Definim pinii Arduino conectați la modulul MAX7219
4. #define DATA_PIN 9 // Pinul digital conectat la pinul DIN al modulului
5. #define CLK_PIN 12 // Pinul digital conectat la pinul CLK al modulului
6. #define CS_PIN 11 // Pinul digital conectat la pinul CS al modulului
7.
8. // Creăm un obiect LedControl pentru a controla matricea LED
9. // Parametrii:
10. // - DATA_PIN: pin pentru date
11. // - CLK_PIN: pin pentru ceas (clock)
12. // - CS_PIN: pin pentru selectarea modulului
13. // - 1: numărul de matrice LED conectate (în acest caz, doar una)
14. LedControl matrix = LedControl(DATA_PIN, CLK_PIN, CS_PIN, 1);
15.
16. void setup() {
17.     // Pornim matricea LED. Funcția "shutdown()" controlează dacă afișajul este pornit sau opri.
18.     // shutdown(0, false): pornim matricea cu indexul 0 (prima matrice).
19.     matrix.shutdown(0, false);
20.
21.     // Setăm luminozitatea matricei LED.
22.     // setIntensity(0, 8): setăm luminozitatea matricei 0 (prima matrice) la nivelul 8 (intervalul este 0-15).
23.     matrix.setIntensity(0, 8);
24.
25.     // Stergem toate LED-urile de pe matrice pentru a începe cu un afișaj curat.
26.     // clearDisplay(0): stergem afișajul matricei 0 (prima matrice).
27.     matrix.clearDisplay(0);
28. }
29.
30. void loop() {
31.     // Acest loop va aprinde un singur LED, care se va mișca de-a lungul matricei
32.     // LED-ul va parcurge toate rândurile (de sus în jos) și coloanele (de la stânga la dreapta)
33.
34.     // Iterăm prin fiecare rând al matricei
35.     for (int row = 0; row < 8; row++) { // rândurile sunt numerotate de la 0 la 7
36.         // Iterăm prin fiecare coloană a rândului curent
37.         for (int col = 0; col < 8; col++) { // coloanele sunt numerotate de la 0 la 7
38.             // Stergem matricea înainte de a aprinde un nou LED.
39.             // Acest pas asigură că doar un singur LED este aprins la un moment dat.
40.             matrix.clearDisplay(0);
41.
42.             // Aprindem LED-ul de la poziția (row, col).
43.             // setLed(0, row, col, true):
44.             // - 0: indică matricea pe care vrem să lucrăm (în cazul nostru, prima matrice).
45.             // - row: rândul unde se află LED-ul.
46.             // - col: coloana unde se află LED-ul.
47.             // - true: aprinde LED-ul (dacă am pune "false", l-ar stinge).
48.             matrix.setLed(0, row, col, true);
49.
50.             // Pauză de 200 milisecunde pentru ca mișcarea LED-ului să fie vizibilă.
51.             delay(200);
52.         }
53.     }
54. }
```

Cerință:

Configurați și controlați o matrice LED 8x8 utilizând modulul MAX7219 pentru afișarea unui triunghi. Folosiți biblioteca [LedControl.h](#) pentru a configura matricea de led-uri.

- a) Definiți funcția displayTriangle() care să permită afișarea unui triunghi pe matricea de led-uri.
- b) Definiți funcția blinkTriangle() care se permite afișarea triunghiului timp de 200ms și apoi curățarea matricii. Repetați acest proces de 3 ori.

Componente necesare:

- Matrice 8x8 led-uri (modulul MAX7219)
- Arduino Uno
- Breadboard
- Fir de conectare

Mod de conectare:

Modulul se conectează la breadboard pe linia B, pozițiile 5 (VCC), 4 (GND), 3 (DIN), 2(CS), 1(CLK).

Se conectează un fir de la C 5 la linia de alimentare (+).

Se conectează un fir de la C 4 la linia de ground (-).

Se conectează un fir de la C 3 la pinul 9 al plăcii Arduino Uno.

Se conectează un fir de la C 2 la pinul 11 al plăcii Arduino Uno.

Se conectează un fir de la C 1 la pinul 12 al plăcii Arduino Uno.

Pentru ușurință urmăriți [figura 16](#).

Cod:

```
1. #include <LedControl.h> // Biblioteca pentru controlul modulului MAX7219
2.
3. // Definim pinii conectați la modulul MAX7219
4. #define DATA_PIN 9 // Pinul digital conectat la DIN
5. #define CLK_PIN 12 // Pinul digital conectat la CLK
6. #define CS_PIN 11 // Pinul digital conectat la CS
7.
8. // Inițializăm obiectul LedControl pentru controlul matricei
9. LedControl matrix = LedControl(DATA_PIN, CLK_PIN, CS_PIN, 1); // Avem doar o matrice
10.
11. // Definim matricea pentru triunghi
12. // Fiecare linie reprezintă un rând din matrice, iar fiecare bit controlează un LED (1 = aprins, 0 = stins)
13. int triangleMatrix[8] = {
14.     B00011000, // Rând 0: aprinde LED-urile din mijloc
15.     B00111100, // Rând 1: extinde rândul
16.     B01111110, // Rând 2: mai multe LED-uri aprinse
17.     B11111111, // Rând 3: rând complet aprins
18.     B00000000, // Rând 4: restul rândurilor sunt stinse
19.     B00000000,
20.     B00000000,
21.     B00000000
22. };
23.
24. void setup() {
25.     // Pornim matricea și o configurăm
```

```

26. matrix.shutdown(0, false); // Pornim afişajul matricei 0
27. matrix.setIntensity(0, 8); // Setăm luminozitatea la nivel mediu (0-15)
28. matrix.clearDisplay(0); // Curăţăm matricea înainte de a afişa ceva
29. }
30.
31. void loop() {
32.   // Clipim triunghiul
33.   blinkTriangle();
34.
35.   // Pauză de 1 secundă înainte de următoarea clipire
36.   delay(1000);
37. }
38.
39. // Funcția pentru afișarea triunghiului pe matrice
40. void displayTriangle() {
41.   // -----Va fi completată de către studenți-----
42. }
43.
44. // Funcția pentru ștergerea matricei
45. void clearMatrix() {
46.   matrix.clearDisplay(0); // Ștergem toate LED-urile de pe matrice
47. }
48.
49. // Funcția pentru clipirea triunghiului
50. void blinkTriangle() {
51.   // -----Va fi completată de către studenți-----
52. }
53.
54. // Funcția pentru clipirea triunghiului
55. void blinkTriangle() {
56.   // -----Va fi completată de către studenți-----
57. }

```

Indicații pentru rezolvare:

Pentru realizarea funcției displayTriangle() se va curăța mai întâi afișajul pentru a preveni interefrență cu o formă deja afișată, iar apoi se va itera prin fiecare rând al matricei pentru a seta valorile led-urilor folosind variabila globală triangleMatrix.

Sarcini de lucru:

- Creați o animație pentru o săgeată care se mișcă spre dreapta. Folosiți o matrice pentru săgeată și creați o funcție moveArrowRight() care mută săgeata cu un pas la dreapta până dispare de pe ecran. (Indicații : Începeți afișând săgeata pe prima poziție. Folosiți un loop pentru a muta săgeata spre dreapta. La fiecare pas, mutați biții săgelei spre dreapta (>> 1) și actualizați matricea. Folosiți delay(200) între pași pentru a face mișcarea vizibilă.)
- Creați o animație pentru o săgeată care se mișcă spre stânga. Folosiți o matrice pentru săgeată și creați o funcție moveArrowLeft() care mută săgeata cu un pas la stânga până dispare de pe ecran.

7.3 FOTOREZISTENȚĂ

În această lucrare de laborator, vom învăța cum să utilizăm un fotorezistor pentru a măsura nivelul de luminozitate ambientală și pentru a ajusta automat intensitatea unei matrice LED de 8x8 controlată prin modulul MAX7219 și microcontrolerul ATmega328P.

Fotorezistoarele sunt utilizate frecvent în aplicații care implică măsurători de lumină, precum

sistemele de iluminat automat, detectoarele de mișcare și dispozitivele IoT pentru case inteligente.

Cerință:

Citiți valoarea analogică a fotorezistorului, convertiți valoare într-un procentaj și afișați-o pe ecranul serial.

Componente necesare:

- Matrice 8x8 led-uri (modulul MAX7219)
- Modul cu fotorezistență
- Arduino Uno
- Breadboard
- Fire de conectare

Mod de conectare:

Modulul se conectează la breadboard pe linia H, pozițiile 31 (VCC), 30 (GND), 29 (R1).

Se conectează un fir de la I 31 la linia de alimentare (+).

Se conectează un fir de la I 30 la linia de ground (-).

Se conectează un fir de la I 29 la pinul analogic A0 al plăcii Arduino Uno.

Pentru ușurință urmăriți [figura 16](#).

Cod:

```
1. // Definim pinul analogic la care este conectat fotorezistorul
2. #define PHOTORESISTOR_PIN A0
3.
4. // Declarăm o variabilă pentru a stoca valoarea citită de la fotorezistor
5. int lightValue = 0;
6.
7. void setup() {
8.     // Configurăm comunicația serială pentru a putea vedea datele pe monitorul serial
9.     // 9600 este viteza de transmisie a datelor (baud rate)
10.    Serial.begin(9600);
11. }
12.
13. void loop() {
14.     // Citim valoarea analogică de la fotorezistor (între 0 și 1023)
15.     lightValue = analogRead(PHOTORESISTOR_PIN);
16.
17.     // Convertim valoarea într-un procentaj (între 0% și 100%) pentru a o înțelege mai ușor
18.     int lightPercentage = map(lightValue, 0, 1023, 0, 100);
19.
20.     // Afisăm pe monitorul serial valoarea brută și valoarea procentuală a luminozității
21.     Serial.print("Luminozitate (valoare brută): ");
22.     Serial.print(lightValue);                                // Afisează valoarea brută
23.     Serial.print("    Luminozitate (%): ");
24.     Serial.println(lightPercentage);                      // Afisează valoarea în procente
25.
26.     // Adăugăm o pauză de 500 milisecunde pentru a evita afișarea continuă
27.     delay(500);
28. }
29.
```

Sarcini de lucru:

Folosind codul furnizat la exercițiul de la matricea de 8x8 LED-uri, creați un proiect care ajustează luminozitatea unei matrici de LED-uri în funcție de valoarea luminii ambientale citite de fotorezistență. Definiți funcția `adjustMatrixIntensity()`:

- a) Citiți valoarea de la fotorezistență folosind `analogRead()`.
- b) Convertiți valoarea citită într-un procentaj folosind funcția `map()` - permite transformarea unei valori dintr-un interval inițial într-un alt interval dorit.
- c) Ajustați intensitatea matricei LED astfel: dacă procentajul este sub 50%, setați intensitatea matricei la un nivel mai mare (mai luminoasă), iar dacă procentajul este peste 50%, reduceti intensitatea matricei.

```
1. // Biblioteca pentru controlul matricei LED
2. #include <LedControl.h>
3.
4. // Definim pinii pentru matricea LED
5. #define DATA_PIN 9
6. #define CLK_PIN 12
7. #define CS_PIN 11
8.
9. // Inițializăm obiectul pentru controlul matricei LED
10. LedControl matrix = LedControl(DATA_PIN, CLK_PIN, CS_PIN, 1);
11.
12. // Definim pinul pentru fotorezistor
13. #define PHOTORESISTOR_PIN A0
14.
15. // Variabile pentru valori de lumină
16. int lightValue = 0;
17. int lightPercentage = 0;
18.
19. void setup() {
20.     // Configurăm matricea LED
21.     matrix.shutdown(0, false); // Pornim matricea
22.     matrix.setIntensity(0, 8); // Setăm o intensitate inițială medie
23.     matrix.clearDisplay(0); // Ștergem display-ul
24.
25.     // Pornim comunicația serială
26.     Serial.begin(9600);
27. }
28.
29. void loop() {
30.     adjustMatrixIntensity(); // Apelăm funcția care ajustează intensitatea matricei
31.     delay(500); // Pauză pentru stabilitate
32. }
33.
34. void adjustMatrixIntensity() {
35.     // -----Va fi completată de către studenți-----
36. }
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
```

7.4 MODULUL JOYSTICK

În această lucrare de laborator, vom învăța cum să utilizăm un modul joystick pentru a controla o matrice LED de 8x8 conectată la microcontrolerul ATmega328P prin intermediul modulului MAX7219. Joystick-urile sunt utilizate frecvent în aplicații de control al mișcării, cum ar fi jocurile video, roboții teleghidați sau dispozitivele electronice care necesită o interfață de

control intuitivă. Scopul acestui exercițiu este de a înțelege cum să citim și să interpretăm valorile analogice furnizate de joystick, să mapăm aceste valori pentru a controla poziția elementelor afișate pe matricea LED și să implementăm funcții dedicate pentru o interacțiune fluidă.

Cerință:

Conectați modulul Joystick la pinii analogici A2 – Ry, A3 – Rx ai plăcii Arduino Uno și modificați funcția loop() astfel încât să citească valorile axelor și să le afișeze pe ecranul serial sub formatul X: [valoare], Y: [valoare].

Componente necesare:

- Matrice 8x8 led-uri (modulul MAX7219)
- Modul cu Joystick
- Arduino Uno
- Breadboard
- Fir de conectare

Mod de conectare:

Modulul se conectează direct la pinii plăcii Arduino Uno, exceptând VCC și GND.

Se conectează un fir de la VCC la linia de alimentare (+).

Se conectează un fir de la GND la linia de ground (-).

Se conectează un fir de la Ry la pinul analogic A2 al plăcii Arduino Uno.

Se conectează un fir de la Rx la pinul analogic A3 al plăcii Arduino Uno.

Se conectează un fir de la SW la pinul digital 7 al plăcii Arduino Uno.

Pentru ușurință urmăriți [figura 16](#).

Cod:

```
1. void setup() {
2.   Serial.begin(9600); // Pornire comunicare serială
3. }
4.
5. void loop() {
6.   int xValue = analogRead(A0); // Citire axa X
7.   int yValue = analogRead(A1); // Citire axa Y
8.
9.   Serial.print("X: ");
10.  Serial.print(xValue);
11.  Serial.print(", Y: ");
12.  Serial.println(yValue);
13.
14.  delay(200); // Pauză pentru citire
15. }
16.
```

Sarcini de lucru:

Folosind codul furnizat la exercițiul de la matricea de 8x8 LED-uri, creați un proiect care afișează săgeți pentru indicarea direcției în funcție de valoarea citită de la modulul Joystick pe axa X.

- a) Citiți valoarea axei X de la Joystick folosind analogRead().
- b) Afişați săgeată stânga (arrowLeft) sau săgeată dreapta (arrowRight) în funcție de poziția Joystick-ului. Ștergeți afișajul dacă Joystick-ul este centrat. Folosiți pragul de 300 pentru a decide direcția indicată de Joystick.

```
1. #include <LedControl.h>
2.
3. // Pini pentru joystick
4. #define JOY_X_PIN           // Pinul pentru axa X a joystick-ului
5. const int threshold = _____; // Prag pentru mișările joystick-ului
6.
7. // Pini pentru matrice
8. int DIN = 10;             // Pinul pentru date
9. int CS = 11;              // Pinul pentru Chip Select
10. int CLK = 12;             // Pinul pentru Clock
11.
12. // Obiect pentru controlul matricei MAX7219
13. LedControl matrix = LedControl(DIN, CLK, CS, 1);
14.
15. // Săgeți pentru afișare
16. int arrowLeft[8] = {
17.     B00011100,
18.     B00011100,
19.     B00011100,
20.     B11111111,
21.     B01111111,
22.     B00111110,
23.     B00011100,
24.     B00001000
25. };
26.
27. int arrowRight[8] = {
28.     B00001000,
29.     B00011100,
30.     B00111110,
31.     B01111111,
32.     B11111111,
33.     B00011100,
34.     B00011100,
35.     B00011100
36. };
37.
38. void setup() {
39.     // Configurare serială pentru debugging
40.     Serial.begin(9600);
41.
42.     // Inițializare matrice
43.     matrix.shutdown(0, false); // Activează matricea
44.     matrix.clearDisplay(0);   // Șterge afișajul matricei
45.
46.     Serial.println("Setup complet!");
47. }
48.
49. void loop() {
50.     // -----Va fi completată de către studenți-----
51. }
52.
```

7.5 ECRAN OLED

În această lucrare de laborator, vom învăța cum să utilizăm un ecran OLED pentru afișarea de informații text și grafice controlate de un microcontroler ATmega328P. Ecranele OLED sunt componente versatile, utilizate frecvent în proiecte de electronică și IoT pentru afișarea clară și eficientă a datelor, având aplicații în dispozitive portabile, panouri de control și interfețe grafice. Scopul acestui exercițiu este de a ne familiariza cu funcțiile oferite de biblioteca [Adafruit SSD1306](#), cum ar fi afișarea de mesaje statice, poziționarea textului și generarea de forme grafice, precum și de a înțelege structura logică pentru gestionarea afișajului. Prin finalizarea acestui laborator, vom explora conceptul de afișare dinamică bazată pe valori numerice.

Cerință:

Conectați modulul cu ecran OLED la placa Arduino Uno și utilizați funcțiile setCursor(), setTextSize(), setTextColor() și display() pentru a afișa pe două linii mesajul:

Hello

World!

Componente necesare:

- Modul cu ecran OLED
- Arduino Uno
- Breadboard
- Fir de conectare

Mod de conectare:

Modulul se conectează la breadboard pe linia C, pozițiile 21 (GND), 20 (VDD), 19 (BCK), 18(BDA).

Se conectează un fir de la B 20 la linia de alimentare (+).

Se conectează un fir de la B 21 la linia de ground (-).

Se conectează un fir de la B 19 la pinul analogic A5 al plăcii Arduino Uno.

Se conectează un fir de la B 18 la pinul analogic A4 al plăcii Arduino Uno.

Pentru ușurință urmăriți [figura 16](#).

Cod:

```
1. #include <Adafruit_GFX.h>          // Biblioteca Adafruit_GFX oferă funcții grafice generale pentru
desene (linii, cercuri, text) pe ecrane.
2. #include <Adafruit_SSD1306.h>        // Biblioteca specifică pentru controlul display-urilor OLED bazate
pe controlerul SSD1306.
3. #include <Wire.h>                  // Biblioteca Wire este utilizată pentru comunicația I2C, care
permite controlul display-ului OLED conectat la Arduino.
4.
5. // Dimensiuni ecran OLED
```

```

6. #define SCREEN_WIDTH 128          // Definește lățimea display-ului OLED în pixeli. (latimea maxima)
7. #define SCREEN_HEIGHT 64         // Definește înălțimea display-ului OLED în pixeli. (inaltimea
maxima)
8.
9. // Obiect pentru OLED
10. // Creează un obiect display de tip Adafruit_SSD1306.
11. // Parametrii:
12. // - SCREEN_WIDTH și SCREEN_HEIGHT: Specifică dimensiunile display-ului.
13. // - &Wire: Specifică utilizarea comunicației I2C.
14. // - -1: Indică că nu este utilizat un pin pentru reset (optional).
15. Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
16.
17. void setup() {
18.   Serial.begin(9600);
19.
20.   // Parametrii: - SSD1306_SWITCHCAPVCC: Indică utilizarea sursei de tensiune internă pentru OLED.
21.   //               - 0x3C: Adresa I2C a display-ului OLED (specifică pentru majoritatea SSD1306).
22.   if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
23.     Serial.println(F("Eroare la inițializarea OLED!"));
24.     for ();}
25. }
26.
27. display.clearDisplay();           // Șterge orice conținut afișat anterior pe display.
28. display.setTextSize(2);          // Setăm dimensiunea textului
29. display.setTextColor(SSD1306_WHITE); // Setăm culoarea textului
30. display.setCursor(10, 20);        // Setăm poziția textului
31. display.print("Hello,");         // Afișăm prima linie
32. display.setCursor(10, 40);        // Afișăm a doua linie
33. display.print("World!");         // Actualizăm ecranul
34. display.display();
35. }
36.
37. void loop() {
38.   // Nu facem nimic în loop
39. }
40.

```

Sarcini de lucru:

Completați spațiile libere în codul de mai jos pentru a realiza o aplicație care afișează un simbol pe baza unui semnal numeric. În funcție de nivelul de semnal signalLevel afișați pe ecranul OLED una, două sau trei arce.

```

1. #include <Adafruit_GFX.h>
2. #include <Adafruit_SSD1306.h>
3. #include <Wire.h>
4.
5. // Dimensiuni ecran OLED
6. #define SCREEN_WIDTH 128
7. #define SCREEN_HEIGHT 64
8.
9. Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
10.
11. void setup() {
12.   Serial.begin(9600);
13.
14.   if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
15.     Serial.println(F("Eroare la inițializarea OLED!"));
16.     for ();}
17. }
18.
19. //----- Completați codul pentru a curăța display-ul-----
20.
21. }
22.
23. void loop() {
24.   int signalLevel = 2; // Puteți modifica între 1, 2, 3 pentru testare
25.
26.   display.clearDisplay();
27.   display.fillCircle(64, 48, 4, SSD1306_WHITE); // Baza simbolului

```

```

28.
29. // ----- Completați codul pentru a desena arcele pe baza lui signalLevel -----
31. display.setTextSize(1);
32. display.setTextColor(SSD1306_WHITE);
33. display.setCursor(0, 56);
34. display.print("Signal: ");
35. display.print(signalLevel);
36.
37. display.display(); // Actualizăm ecranul
38.
39. delay(1000);
40. }
41.

```

7.6 SENZOR ULTRASONIC

În această lucrare de laborator, vom învăța cum să utilizăm senzorul ultrasonic HC-SR04 pentru măsurarea distanțelor și cum să afișăm aceste informații pe un ecran OLED utilizând un microcontroler ATmega328P. Senzorul ultrasonic HC-SR04 este folosit frecvent în aplicații precum roboți autonomi, sisteme de siguranță și monitorizare a distanței, oferind o modalitate eficientă de a detecta obstacole și măsura distanțe. Ecranul OLED completează această funcționalitate prin afișarea clară și personalizabilă a datelor.

Cerință:

Conectați senzorul ultrasonic la placa Arduino Uno și utilizați funcțiile digitalWrite() și pulseIn() pentru a alcătui un program Arduino care să îndeplinească următoarele sarcini:

- Să configureze pinul **TRIG** ca ieșire digitală și pinul **ECHO** ca intrare digitală.
- Să trimită un impuls ultrasonic de 10 microsecunde prin pinul **TRIG**.
- Să măsoare timpul (în microsecunde) cât semnalul reflectat rămâne pe **HIGH** folosind funcția pulseIn().
- Să calculeze distanța până la un obiect folosind formula și să se afișeze rezultatul pe monitorul serial. (0.034 reprezintă viteza sunetului în aer ($340 \text{ m/s} = 0.034 \text{ cm}/\mu\text{s}$), iar împărțirea la 2 este necesară deoarece semnalul ultrasonic parcurge distanța dus-întors)

$$\text{Distanța (cm)} = \frac{\text{Durata } (\mu\text{s}) * 0.034}{2}$$

Componente necesare:

- Senzor ultrasonic
- Modul cu ecran OLED
- Arduino Uno
- Breadboard
- Fire de conectare

Mod de conectare:

Modulul se conectează la breadboard pe linia B, pozițiile 57 (VCC), 56 (TRIG), 55 (ECHO), 54 (GND).

Se conectează un fir de la C 57 la linia de alimentare (+).

Se conectează un fir de la C 54 la linia de ground (-).

Se conectează un fir de la C 56 la pinul digital 2 al plăcii Arduino Uno.

Se conectează un fir de la C 55 la pinul digital 3 al plăcii Arduino Uno.

Pentru ușurință urmăriți [figura 16](#).

Cod:

```
1. #define TRIG_PIN 2 // TRIG_PIN (Pinul 2) este utilizat pentru a genera un semnal ultrasonic.
2. #define ECHO_PIN 3 // ECHO_PIN (Pinul 3) este utilizat pentru a primi semnalul ultrasonic
reflectat.
3.
4. void setup() {
5.   Serial.begin(9600);          // Inițializare Serial Monitor
6.   pinMode(TRIG_PIN, OUTPUT);  // Configurează pinul TRIG ca ieșire digitală, astfel încât Arduino
să poată genera impulsuri către senzor.
7.   pinMode(ECHO_PIN, INPUT);   // Configurează pinul ECHO ca intrare digitală, astfel încât Arduino
să poată citi impulsurile reflectate.
8.
9.
10. void loop() {
11.   // Trimite un impuls de 10µs pe pinul TRIG
12.   digitalWrite(TRIG_PIN, LOW); // Asigură că pinul TRIG este setat la LOW pentru a reseta senzorul
înainte de a trimite un nou semnal.
13.   delayMicroseconds(2);      // Creează o pauză scurtă de 2 microsecunde înainte de a genera
impulsul.
14.   digitalWrite(TRIG_PIN, HIGH); // Activează senzorul, trimițând un impuls ultrasonic prin setarea
pinului TRIG la HIGH.
15.   delayMicroseconds(10);     // Durata impulsului emis este de 10 microsecunde.
16.   digitalWrite(TRIG_PIN, LOW); // Oprește semnalul ultrasonic, setând pinul TRIG înapoi la LOW.
17.
18.   // Măsoară durata impulsului returnat
19.   long duration = pulseIn(ECHO_PIN, HIGH); // Măsoară timpul (în microsecunde) cât pinul ECHO
rămâne HIGH
20.
21.   // Calculează distanța (durata * viteza sunet / 2)
22.   // duration: Timpul măsurat (în microsecunde).
23.   // 0.034: Viteză sunetului în aer (340 m/s sau 0.034 cm/µs).
24.   // Împărțim la 2 deoarece semnalul ultrasonic parcurge distanța dus-întors.
25.   float distance = duration * 0.034 / 2;
26.
27.   // Afisează distanța în cm
28.   Serial.print("Distanța: ");
29.   Serial.print(distance);
30.   Serial.println(" cm");
31.
32.   delay(500);
33. }
34.
```

Sarcini de lucru:

Completați spațiile libere în codul de mai jos pentru a realiza o aplicație care afișează pe ecranul OLED distanța măsurată de senzorul ultrasonic (în cm). Dacă distanța este:

- <10 cm: Desenați 3 arce de cerc pe ecran.
- Între 10 cm și 20 cm: Desenați 2 arce de cerc.

- Între 20 cm și 30 cm: Desenați 1 arc de cerc.
- Mai mare de 30 cm: Afisați doar distanța măsurată.

```

1. #include <Adafruit_GFX.h>
2. #include <Adafruit_SSD1306.h>
3.
4. #define SCREEN_WIDTH 128
5. #define SCREEN_HEIGHT 64
6. #define TRIG_PIN 2
7. #define ECHO_PIN 3
8.
9. Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
10.
11. void setup() {
12.   Serial.begin(9600);
13.
14.   // Configurare pinii senzorului ultrasonic
15.   _____; // <- Completați cu configurarea pinului TRIG ca OUTPUT
16.   _____; // <- Completați cu configurarea pinului ECHO ca INPUT
17.
18.   // Inițializare OLED
19.   if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
20.     Serial.println(F("Eroare la initializarea OLED!"));
21.     for (;;) // Blochează programul
22.   }
23.
24.   // Curățăm ecranul OLED
25.   _____; // <- Completați pentru a curăța display-ul
26. }
27.
28. void loop() {
29.   // Trimite un impuls pe pinul TRIG
30.   digitalWrite(TRIG_PIN, LOW);
31.   delayMicroseconds(2);
32.   _____; // <- Completați pentru a trimite un impuls HIGH de 10 microsecunde
33.   digitalWrite(TRIG_PIN, LOW);
34.
35.   // Măsoară durata impulsului returnat
36.   _____; // <- Completați codul pentru a citi impulsul de pe pinul ECHO
37.
38.   // Calculează distanța (durata * viteză sunet / 2)
39.   float distance = _____; // <- Completați formula de calcul pentru distanță
40.
41.   // Afisare pe OLED
42.   display.clearDisplay();
43.
44.   // Configurare text
45.   display.setTextSize(2);
46.   display.setTextColor(SSD1306_WHITE);
47.   _____; // <- Completați pentru a seta poziția cursorului
48.
49.   // Afisare distanta pe ecran
50.   display.print("Dist: ");
51.   display.print(______); // <- Completați pentru a afișa valoarea distanței
52.   display.println(" cm");
53.   // -----Completați codul pentru afișarea arcelor-----
54.   // Actualizare ecran OLED
55.   _____; // <- Completați pentru a actualiza ecranul OLED
56.
57.   delay(500);
58. }
59.

```

Prin finalizarea acestor exerciții de laborator, studenții vor dobândi cunoștințe fundamentale despre utilizarea perifericelor. Prin concatenarea părților de cod din sarcinile de lucru se va obține sistemul de semnalizare pentru mijloacele de transport alternativ.

8 REZOLVAREA PROIECTULUI

Pentru a sprijini procesul de învățare și a facilita accesul la soluțiile prezentate în cadrul acestui laborator, toate rezolvările exercițiilor propuse au fost încărcate pe o platformă de versionare. Studenții pot consulta și descărca aceste soluții direct de pe GitHub, la următorul link: [Sistem-de-semnalizare-pentru-transportul-alternativ.](#)

Acest repository include codurile sursă bine structurate și documentate, alături de explicații detaliate pentru fiecare exercițiu. Scopul acestei inițiative este de a oferi un suport tehnic ușor accesibil, care să contribuie la aprofundarea cunoștințelor și la aplicarea lor practică în dezvoltarea de sisteme embedded.

9 FOI DE CALATLOG

Componenta	Link foaie de catalog	Pret
Matrice 8x8 Led - MAX7219	https://www.analog.com/media/en/technical-documentation/data-sheets/MAX7219-MAX7221.pdf	10,54 RON
Senzor Ultrasonic – HC-SR04	https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf	6,96 RON
Buzzer – MH-FMD	https://egizmo.net/oc/kits%20documents/High%20Quality%20Passive%20Buzzer/High%20quality%20passive%20buzzer.pdf	8,33 RON
Modul Joystick XY – KY-023	https://www.atrinelec.com/download/datasheet/joystick_module(Atrinelec.com).pdf	5,36 RON
Modul fotorezistență – KY-018	https://atom.ubbcluj.ro/alpar/datasheets/sensors_actuators/KY%C2%AD018%20Photo%20resistor%20module.pdf	2,28 RON
128x64 I2C OLED	https://www.vishay.com/docs/37902/oled128x64dbpp3n00000.pdf	17,83 RON

Placa Arduino Uno		<u>29,89</u> RON
Breadboard		<u>10,98</u> RON
Cabluri		<u>5,90</u> RON
Total		98,07 RON