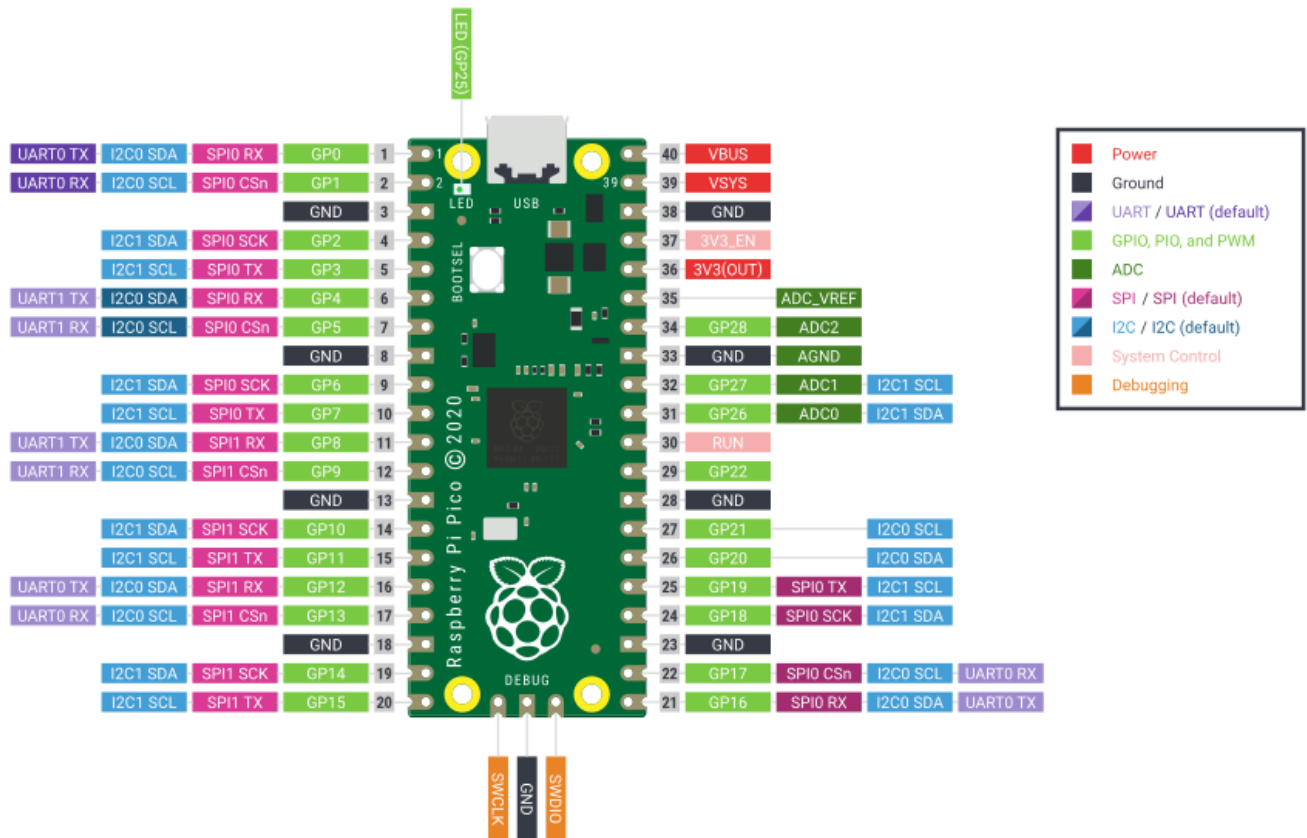


Descriere detaliată

Programarea in MicroPython a unui raspberry pi 2020 utilizand Thonny:



Daca modificam valoarea parametrului utime.sleep(2) de la inceput, cu utime.sleep(5), ledul va fi aprins pentru 5 secunde.

In cea de-a doua problema,vom analiza intensitatea luminii led-ului:

```
from machine import Pin,PWM
from time import sleep
```

```
pwm=PWM(Pin(15))
```

```
pwm.freq(1000)
```

```
while True:
```

```
    for duty in range(65025):
        pwm.duty_u16(duty)
        sleep(0.0001)
```

```
    for duty in range(65025, 0, -1):
        pwm.duty_u16(duty)
        sleep(0.0001)
```

```
from machine import Pin, PWM
from time import sleep
```

```
# Configurăm PWM pe pinul 15 (unde este conectat LED-ul)
```

```
pwm = PWM(Pin(15))
```

```
pwm.freq(1000) # Setăm frecvența PWM la 1000 Hz
```

```
while True:
```

```
    # Creștem intensitatea LED-ului
```

```
    for duty in range(0, 65025, 64): # Ajustăm pasul pentru a controla viteza de variație
```

```
        pwm.duty_u16(duty)
```

```
        sleep(0.01) # Pauză de 10 milisecunde pentru a vizualiza schimbarea intensității
```

```
    # Scădem intensitatea LED-ului
```

```
    for duty in range(65024, -1, -64):
```

```
        pwm.duty_u16(duty)
```

```
        sleep(0.01) # Pauză de 10 milisecunde pentru a vizualiza schimbarea intensității
```

Scaderea intensitatii LED-ului folosind pwm:

```
from machine import Pin, PWM
from time import sleep
```

```
# Configurăm PWM pe pinul 15 (unde este conectat LED-ul)
```

```
pwm = PWM(Pin(25))
```

```
pwm.freq(1000) # Setăm frecvența PWM la 1000 Hz
```

```
# LED-ul pornește la intensitate maximă și scade treptat până se stinge
```

```
while True:
```

```
for duty in range(65025, -1, -64): # Scădem valoarea ciclului de lucru de la 65025 la 0 în pași de 64
```

```
    pwm.duty_u16(duty)
```

```
    sleep(0.001) # Pauză de 10 milisecunde pentru a vizualiza schimbarea intensității
```

```
    # Pauză după ce LED-ul se stinge complet, înainte de a reporni ciclul
```

```
    sleep(0.5) # Pauză de 1 secundă (poate fi ajustată după preferințe)
```

→ pentru ca led-ul să scadă cu o intensitate mai mare, putem mari pasul

→ în loc de -64, vom pune o valoare mai mare, de exemplu, -512

→ primul **sleep** modifică durata pauzei dintre două aprinderi ale led-ului (pentru a vizualiza schimbarea intensității)

→ cel de-al doilea **sleep** modifică durata pauzei până vom aprinde din nou led-ul

REALIZAREA UNUI TERMOMETRU DIGITAL FOLOSIND RASPBERRY PI PICO



SENZORUL DE TEMPERATURA FOLOSIT: DS18B20

Pinul 36 de pe raspberry este pentru alimentare

Pinul 38-ground

Pinul 34-alimentare(VDD)

Codul programului:

```
from machine import Pin
```

```
import time
```

```
import onewire
```

```
import ds18x20
```

```
ow = onewire.OneWire(Pin(28))
```

```
ds = ds18x20.DS18X20(ow)
```

```
devices = ds.scan()
```

```
print('found devices:', devices)
```

```
while True:
```

```
ds.convert_temp()
time.sleep_ms(750)
for device in devices:
    print("Device: {}".format(device))
    print("Temperatura din camera este= {}".format(ds.read_temp(device)))
```

Modificam problema anterioara in felul urmatoar:

- daca temperatura citita de senzor este mai mica de 30 de grade, avem LED-ul aprins
- daca temperatura citita de senzor este mai mare sau egala cu 30 de grade, avem LED-ul stins

```
from machine import Pin
import time
import onewire
import ds18x20
```

```
ow = onewire.OneWire(Pin(28))
ds = ds18x20.DS18X20(ow)
led_pin=Pin(25,Pin.OUT) #pinul pentru LED
```

```
devices = ds.scan()
print('found devices:', devices)
```

```
while True:
    ds.convert_temp()
    time.sleep_ms(750)
    for device in devices:
        temp=ds.read_temp(device)
        print("Device: {}".format(device))
        print("Temperatura din camera este= {}".format(temp))
```

```
    if temp<=30:
        led_pin.value(1)
    else:
        led_pin.value(0)
```

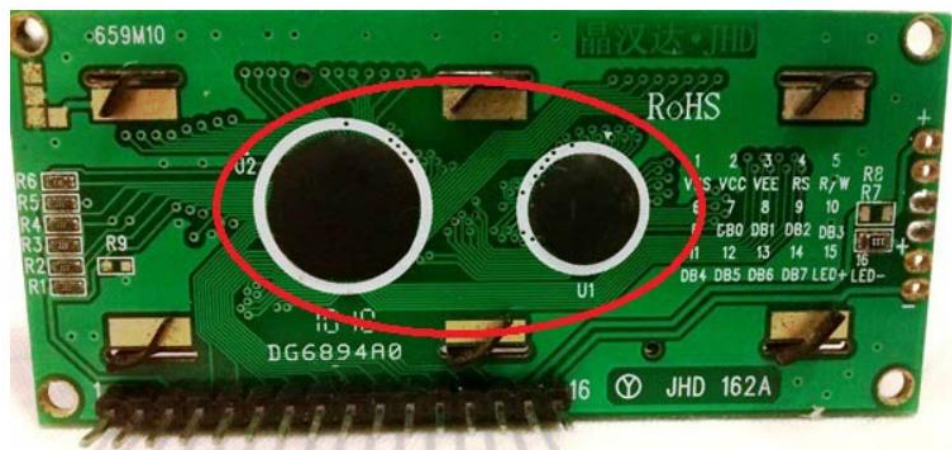
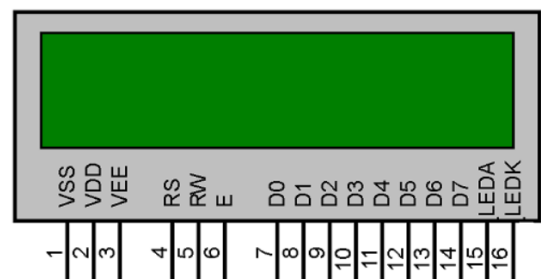
```
    time.sleep(1)
```

INTRODUCEREA UNUI ECRAN LCD PENTRU AFISAREA TEMPERATURII

-notiuni teoretice-

Display ul folosit este lcd 16x2 qapass
Acest lcd permite afisarea pe 2 linii si 16 coloane
Acesta are urmatoarea configuratie a pinilor:

PIN NO	Symbol	Fuction
1	VSS	GND
2	VDD	+5V
3	V0	Contrast adjustment
4	RS	H/L Register select signal
5	RW	H/L Read/Write signal
6	E	H/L Enable signal
7	DB0	H/L Data bus line
8	DB1	H/L Data bus line
9	DB2	H/L Data bus line
10	DB3	H/L Data bus line
11	DB4	H/L Data bus line
12	DB5	H/L Data bus line
13	DB6	H/L Data bus line
14	DB7	H/L Data bus line
15	A	+4.2V for LED
16	K	Power supply for BKL(0V)



Daca intoarcem LCD-ul, vom vedea doua puncte negre, ca in imaginea de mai sus.
Acele puncte negre fac parte dintr-o interfata IC, a carei componente asociate ne ajuta sa folosim acest LCD impreuna cu MCU(Main Control Unit).
Datorita matricii 16x2, LCD-ul nostru are in total 32 caractere in total, fiecare caracter avand o dimensiune de 5*8 pixeli.
Mai jos avem o ilustrare a spatiului ocupat de fiecare caracter:



Având în vedere cele enunțate mai sus, ne rezulta că avem o dimensiune de 40 pixeli (5*8) pentru fiecare caracter, iar noi având un maxim de 32 de caractere posibil afisabile, avem o dimensiune totală de 32*40 pixeli, adică 1280px.

MODURILE DE LUCRU POSIBILE: 4BITI/8 BITI

LCD-ul poate funcționa în 2 moduri diferite de lucru, adică cele menționate mai sus.

MODUL PE 4 BITI:

- în acest mod de funcționare, transmitem datele prin nibble-uri, prima dată cel superior, iar mai apoi cel inferior

- nibble-ul superior este accesat de bitii: D4-D7

- nibble-ul inferior este accesat de bitii: D3-D0

Pentru programarea LCD-ului, acesta are implementat pe suprafața lui un expander pentru bus-ul I2C, **PCF8574**, pentru a utiliza doar 4 biți, în loc de 16

Pinii **SCL (Serial Clock Line)** și **SDA (Serial Data Line)** ale unui expander PCF8574AT sunt utilizați pentru comunicarea prin protocolul I²C (Inter-Integrated Circuit). Acesta este un protocol de comunicare serială de mare viteză dezvoltat de Philips, care permite comunicarea între un microcontroler și mai multe periferice folosind doar două fire.

• **SCL (Serial Clock Line):**

- Acest pin transportă semnalul de ceas generat de master (de obicei un microcontroler) care coordonează sincronizarea transferului de date. Toate dispozitivele conectate pe magistrala I²C folosesc acest semnal pentru a sincroniza transferul de date.

• **SDA (Serial Data Line):**

- Acest pin transportă datele bidirecțional între master și slave. Datele sunt transmise în formă serială, un bit la un moment dat, sincronizate de semnalul de ceas de pe linia SCL.

Utilizarea Practică a Pinilor SCL și SDA

- **Conectarea:** Pentru a folosi PCF8574AT într-un circuit I²C, conectezi pinul SCL al PCF8574AT la pinul SCL al microcontrolerului și pinul SDA al PCF8574AT la pinul SDA al microcontrolerului.
- **Rezistențe de Tragere în Sus:** Deoarece liniile I²C sunt de tip open-drain, este necesar să adaugi rezistențe de tragere în sus (pull-up resistors) pe ambele linii SCL și SDA pentru a asigura nivelurile logice corecte. De obicei, valorile acestor rezistențe sunt între 4.7kΩ și 10kΩ.
- **Adresarea:** Fiecare dispozitiv pe magistrala I²C are o adresă unică. PCF8574AT are 3 pini de adresare (A0, A1, A2) care permit configurarea a 8 adrese diferite, permițând conectarea mai multor PCF8574AT pe aceeași magistrală I²C fără conflicte.

Configurația tipică a unei conexiuni I²C cu un PCF8574AT arată astfel:

1. **Microcontroler:**

- SCL (ceas) conectat la SCL al PCF8574AT
- SDA (date) conectat la SDA al PCF8574AT
- Rezistențe de tragere în sus pe liniile SCL și SDA

2. **PCF8574AT:**

- Pini SCL și SDA conectați la magistrala I²C
- Pini A0, A1, A2 conectați pentru configurarea adresei

- Pinii de intrare/ieșire conectați la dispozitivele periferice pe care dorești să le controlezi.

Prin urmare, SCL și SDA sunt esențiali pentru transferul sincronizat de date pe magistrala I²C, permițând extinderea funcționalității microcontrolerului prin utilizarea PCF8574AT ca expander de intrări/ieșiri.

CONFIGURATIA PINILOR:

5 PINNING

SYMBOL	PIN		DESCRIPTION
	DIP16; SO16	SSOP20	
A0	1	6	address input 0
A1	2	7	address input 1
A2	3	9	address input 2
P0	4	10	quasi-bidirectional I/O 0
P1	5	11	quasi-bidirectional I/O 1
P2	6	12	quasi-bidirectional I/O 2
P3	7	14	quasi-bidirectional I/O 3
V _{SS}	8	15	supply ground
P4	9	16	quasi-bidirectional I/O 4
P5	10	17	quasi-bidirectional I/O 5
P6	11	19	quasi-bidirectional I/O 6
P7	12	20	quasi-bidirectional I/O 7
$\overline{\text{INT}}$	13	1	interrupt output (active LOW)
SCL	14	2	serial clock line
SDA	15	4	serial data line
V _{DD}	16	5	supply voltage
n.c.	–	3	not connected
n.c.	–	8	not connected
n.c.	–	13	not connected
n.c.	–	18	not connected

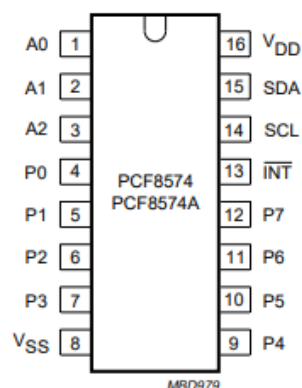


Fig.2 Pin configuration (DIP16; SO16).

COMENZI PENTRU LCD:

Hex Code	Command to LCD Instruction Register
0F	LCD ON, cursor ON
01	Clear display screen
02	Return home
04	Decrement cursor (shift cursor to left)
06	Increment cursor (shift cursor to right)
05	Shift display right
07	Shift display left
0E	Display ON, cursor blinking
80	Force cursor to beginning of first line
C0	Force cursor to beginning of second line
38	2 lines and 5×7 matrix
83	Cursor line 1 position 3
3C	Activate second line
08	Display OFF, cursor OFF
C1	Jump to second line, position 1

C2	Jump to second line, position 2
OC	Display ON, cursor OFF

Dupa ce am identificat pinii componentelor, facem conexiunile necesare intre expander si microcontroler:

- pinul 1 de pe microcontroler(SDA) va fi conectat la pinul SDA de pe expander
- pinul 2 de pe microcontroler(SCL) va fi conectat la pinul SCL de pe expander
- alimentarea o vom face de la microcontroler, iar mai apoi conectam pinul VCC al expanderului la alimentarea microcontrolerului(pinul 36)
- in continuare, vom gasi adresa i2c a modulului LCD, pentru a putea configura corect LCD-ul

PIN LAYOUT



Folosirea unui arduino afisare text pe ambele linii ale LCD-ului:

```
#include<LiquidCrystal.h>
```

```
const int rs=8, en=9, d4=4, d5=5, d6=6, d7=7;
LiquidCrystal lcd(rs ,en ,d4 ,d5, d6, d7);
void setup() {
  // setam numaru de linii/coloane ale LCD-ului:
  lcd.begin(16,2);
  //afisam un mesaj pe lcd:
  lcd.print("Data de azi:");
}
```

```
}

void loop() {
  // setam cursorul pe linia 2, coloana 1
  lcd.setCursor(0,1);
  //numarul de secunde de la resetare
  lcd.print("03/07/2024");
}
```

FOLOSIND ARDUINO UNO SI LCD keypad shield, vom implementa un joc care identifica ce buton este apasat de pe keypad.

Pe prima linie avem afisat urmatorul text:”Apasa un buton:”. In timp ce butonul respective este apasat, vom avea afisata pe linia a doua denumirea butonului actionat.

In cazul in care am apasat butonul de **reset**, contorul se opreste pentru o secunda, iar mai apoi incepe din nou contorizarea.

Nu avem o limita impusa pentru contor.

Codul scris in limbajul c:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

int lcd_key = 0;
int adc_key_in = 0;
#define btnRIGHT 0
#define btnUP 1
#define btnDOWN 2
#define btnLEFT 3
#define btnSELECT 4
#define btnNONE 5
int read_LCD_buttons(){
  adc_key_in = analogRead(0);

  if (adc_key_in > 1000) return btnNONE;

  if (adc_key_in < 50) return btnRIGHT;
  if (adc_key_in < 250) return btnUP;
  if (adc_key_in < 450) return btnDOWN;
  if (adc_key_in < 650) return btnLEFT;
  if (adc_key_in < 850) return btnSELECT;

  return btnNONE;
}

void setup(){
  lcd.begin(16, 2);
  lcd.setCursor(0,0);
  lcd.print("Apasa un buton");
}

void loop(){
  lcd.setCursor(9,1);
  lcd.print(millis()/1000);
  lcd.setCursor(0,1);
  lcd_key = read_LCD_buttons();
  switch (lcd_key){
    case btnRIGHT:{
      lcd.print("RIGHT ");
      break;
    }
    case btnLEFT:{
```

```

lcd.print("LEFT ");
break;
}
case btnUP:{
lcd.print("UP ");
break;
}
case btnDOWN:{
lcd.print("DOWN ");
break;
}
case btnSELECT:{
lcd.print("SELECT");

break;
}
case btnNONE:{
lcd.print("NICIUNUL ");
break;
}
}
}
}

```

REALIZAREA SENZORULUI DE TEMPERATURA FOLOSIND :

- microcontrolerul raspberry pico pi
- lcd 16x02+expander inclus
- breadboard

```

from machine import Pin, I2C
from pico_i2c_lcd import I2cLcd
import time
import onewire
import ds18x20

# Configurare pentru senzor DS18B20
ow = onewire.OneWire(Pin(28))
ds = ds18x20.DS18X20(ow)
led_pin = Pin(25, Pin.OUT) # Pinul pentru LED

# Scanare pentru dispozitive DS18B20
ds_devices = ds.scan()
if not ds_devices:
    print("Nu au fost găsite dispozitive DS18B20.")
else:

```

```

print("S-au găsit dispozitive DS18B20:", ds_devices)

# Configurare I2C
i2c = I2C(0, sda=Pin(4), scl=Pin(5), freq=100000)
lcd = I2cLcd(i2c, 0x27, 2, 16)

# Scanare dispozitive I2C
i2c_devices = i2c.scan()
if len(i2c_devices) == 0:
    print("Nu există dispozitive conectate.")
else:
    print("Nr. dispozitive I2C găsite:", len(i2c_devices))
    for device in i2c_devices:
        print("Adresa zecimală:", device, "| Adresa în hexazecimal:", hex(device))

# Afișare mesaj pe LCD
lcd.move_to(0, 0)
lcd.putstr("Temp.:")

while True:
    ds.convert_temp()
    time.sleep_ms(750)
    for device in ds_devices:
        temp = ds.read_temp(device)
        print("Device: {}".format(device))
        print("Temperatura din cameră este = {}".format(temp))

    lcd.move_to(2, 1)
    lcd.putstr("Temp: {:.2f}C".format(temp))

    if temp >= 30:
        led_pin.value(1)
    else:
        led_pin.value(0)

    time.sleep(1)

```

Am utilizat si alt lcd, a carui expander are adresa: 0x3f

```

from machine import Pin, I2C
from pico_i2c_lcd import I2cLcd
import time
import onewire
import ds18x20

# Configurare pentru senzor DS18B20
ow = onewire.OneWire(Pin(28))
ds = ds18x20.DS18X20(ow)
led_pin = Pin(25, Pin.OUT) # Pinul pentru LED

# Scanare pentru dispozitive DS18B20
ds_devices = ds.scan()

```

```

if not ds_devices:
    print("Nu au fost găsite dispozitive DS18B20.")
else:
    print("S-au găsit dispozitive DS18B20:", ds_devices)

# Configurare I2C
i2c = I2C(0, sda=Pin(4), scl=Pin(5), freq=100000)
lcd = I2cLcd(i2c, 0x3f, 2, 16)

# Scanare dispozitive I2C
i2c_devices = i2c.scan()
if len(i2c_devices) == 0:
    print("Nu există dispozitive conectate.")
else:
    print("Nr. dispozitive I2C găsite:", len(i2c_devices))
    for device in i2c_devices:
        print("Adresa zecimală:", device, "| Adresa în hexazecimal:", hex(device))

# Afişare mesaj pe LCD
lcd.move_to(0, 0)
lcd.putstr("Temp.:")

while True:
    ds.convert_temp()
    time.sleep_ms(750)
    for device in ds_devices:
        temp = ds.read_temp(device)
        print("Device: {}".format(device))
        print("Temperatura din cameră este = {}".format(temp))

    lcd.move_to(2, 1)
    lcd.putstr("Temp: {:.2f}C".format(temp))

    if temp >= 30:
        led_pin.value(1)
    else:
        led_pin.value(0)

    time.sleep(1)

```

UTILIZAREA UNUI SENZOR ULTRASONIC DE DISTANTA-HC-SR04

L-am folosit pentru a testa detectia obiectului din fata acestuia la o distanta maxima de 200 cm(2m)
Pentru a realiza acest lucru, am avut nevoie de un arduino uno, cateva fire si senzorul ultrasonic.

Acesta este codul utilizat:

```
#include <NewPing.h>

#define TRIGGER_PIN 10 // Pinul Trig
#define ECHO_PIN 13 // Pinul Echo
#define MAX_DISTANCE 200 // Distanța maximă (în centimetri)

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

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

  delay(100); // Mărim delay-ul pentru inițializare
}

void loop() {
  delay(50); // Delay scurt pentru a evita ping-uri prea dese
  unsigned int uS = sonar.ping(); // Obținem timpul de răspuns în microsecunde
  unsigned int cm = sonar.convert_cm(uS); // Convertim timpul de răspuns în centimetri
  Serial.print("Distance: ");
  Serial.print(cm); // Afișăm distanța în centimetri
  Serial.println("cm");
  delay(1000); // Delay de 1 secundă între măsurători
}
```

Codul pentru gasirea adresei device-ului:

```
#include <Wire.h>

void setup() {
  Wire.begin();
  Serial.begin(9600);
  while (!Serial); // Așteaptă deschiderea serial monitorului
  Serial.println("\nI2C Scanner");
  Serial.println("Scanning...");
  byte count = 0;

  for (byte i = 1; i < 127; i++) {
    Wire.beginTransmission(i);
    if (Wire.endTransmission() == 0) {
      Serial.print("Found address: ");
      Serial.print(i, DEC);
      Serial.print(" (0x");
      Serial.print(i, HEX);
      Serial.println(")");
    }
  }
}
```

```

        count++;
    }
}

Serial.print("Found ");
Serial.print(count, DEC);
Serial.println(" device(s).");
}

void loop() {}

```

Afisare simpla text pe lcd:

```

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Adresă I2C a modulului LCD 1602 cu expander (PCF8574)
// Schimbă adresa dacă este diferită (poate fi 0x27 sau 0x3F)
#define I2C_ADDRESS 0x27

// Dimensiunea LCD-ului (16 coloane x 2 rânduri pentru LCD 1602)
LiquidCrystal_I2C lcd(I2C_ADDRESS, 16, 2);

void setup() {
    // Inițializarea comunicării I2C
    Wire.begin();

    // Inițializarea LCD-ului
    lcd.init();

    // Porneste iluminarea de fundal a LCD-ului
    lcd.backlight();

    // Setează cursorul la începutul primului rând
    lcd.setCursor(0, 0);

    // Afişează un mesaj pe LCD
    lcd.print("Distanța măsurată:");
}

void loop() {
    // Aici poți adăuga alte instrucțiuni, dar în acest exemplu loop-ul este gol
}

```

Utilizarea unui neopixel pe 16 biti:

CJMU-2812-16

Pentru testarea acestuia, vom conecta led-ul la un arduino uno.

Pentru simularea culorii verzi am realizat urmatoarea combinatie de culori, codul rezultat fiind:

```
#include <Adafruit_NeoPixel.h>

// the data pin for the NeoPixels
int neoPixelPin = 6;

// How many NeoPixels we will be using, change accordingly
int numPixels = 60;

// Instantiate the NeoPixel from the library
Adafruit_NeoPixel strip = Adafruit_NeoPixel(numPixels, neoPixelPin, NEO_GRB +
NEO_KHZ800);

// Color set #1
int r1 = 0;
int g1 = 255;
int b1 = 0;

// Color set #2
int r2 = 0;
int g2 = 0;
int b2 = 255;

// Color set #3
int r3 = 255;
int g3 = 0;
int b3 = 0;

// our button
int switchPin = 4;

// a pre-processor macro
#define DELAY_TIME (10)

void setup() {
  strip.begin(); // initialize the strip
  strip.show(); // make sure it is visible
  strip.clear(); // Initialize all pixels to 'off'
}

void loop() {
  // Check to see if switch is on/off and call the proper function
  if( digitalRead(switchPin) == true )
    alloff();
  else
    activate();

  // delay for the purposes of debouncing the switch
```



```

    delay(DELAY_TIME);
}

// Turns all the NeoPixels off
void allOff() {
    strip.clear();
    strip.show();
}

// turn 1/3 of strip different colors. this is a crude, but effective way to do this.
void activate() {
    // first 20 pixels = color set #1
    for( int i = 0; i < 20; i++ ) {
        strip.setPixelColor(i, r1, g1, b1 );
    }

    // next 20 pixels = color set #2
    for( int i = 20; i < 40 ; i++ ) {
        strip.setPixelColor(i, r2, g2, b2 );
    }

    // last 20 pixels = color set #3
    for( int i = 40; i < 60; i++ ) {
        strip.setPixelColor(i, r3, g3, b3 );
    }

    strip.show();
}

```

Codul urmator a fost utilizat pentru a aprinde led-rile treptat folosind un delay de 10 ms.

```

#include <Adafruit_NeoPixel.h>

// the data pin for the NeoPixels
int neoPixelPin = 6;

// How many NeoPixels we will be using, charge accordingly
int numPixels = 60;

// Instantiate the NeoPixel from the library
Adafruit_NeoPixel strip = Adafruit_NeoPixel(numPixels, neoPixelPin, NEO_GRB +
NEO_KHZ800);

```

```

// our button
int switchPin = 4;

// initial colors, you can change these
int red = 100;
int green = 255;

// a pre-processor macro
#define DELAY_TIME (1)

// stores the start time, use unsigned long to prevent overflow
unsigned long startTime;

int startPixel = 0;
boolean bSwitchDown = false;

void setup() {
    strip.begin(); // initialize the strip
    strip.show(); // make sure it is visible
    strip.clear(); // Initialize all pixels to 'off'

    startTime = millis();
    activate();
}

// use the button asynchronously
void loop() {
    if( digitalRead(switchPin) == true ) {
        bSwitchDown = true;
        allOff();
        startTime = 0;
    }
    else {
        if( bSwitchDown )
            bSwitchDown = false;

        if( startTime + DELAY_TIME < millis() ) {
            activate();
            startTime = millis();
        }
    }
}

void allOff() {
    strip.clear();
    strip.show();
}

void activate() {
    int sp = startPixel;

    for( int i = 0; i < numPixels; i++ )
    {

```

```

        if(sp%2==0)
            strip.setPixelColor(sp,i*3,i*2,i);
        else
            strip.setPixelColor(sp, i, i*2, i*3 );
        strip.setBrightness(i);

        if( sp == numPixels )
            sp = 0;
        else
            sp++;

    }

    strip.show();

    startPixel++;
    if( startPixel == 60 )
        startPixel = 0;

}

```

Am adus cateva modificari codului, astfel incat atunci cand introducem in comunicatia seriala urmatoarele taste avem modificarile:

- apasarea tastei ,p' pentru a avea aprinse doar LED-urile pare
- apasarea tastei ,i' pentru a avea aprinse doar LED-urile impare
- apasarea tastei ,a' pentru a avea aprinse toate LED-urile

```
#include <Adafruit_NeoPixel.h>
```

```
// the data pin for the NeoPixels
int neoPixelPin = 6;
```

```
// How many NeoPixels we will be using, change accordingly
int numPixels = 60;
```

```
// Instantiate the NeoPixel from the library
Adafruit_NeoPixel strip = Adafruit_NeoPixel(numPixels, neoPixelPin, NEO_GRB +
NEO_KHZ800);
char mode='a';
// our button
int switchPin = 4;
```

```
// initial colors, you can change these
int red = 100;
```

```

int green = 255;

// a pre-processor macro
#define DELAY_TIME (1)

// stores the start time, use unsigned long to prevent overflow
unsigned long startTime;

int startPixel = 0;
boolean bSwitchDown = false;

void setup() {
  strip.begin(); // initialize the strip
  strip.show(); // make sure it is visible
  strip.clear(); // Initialize all pixels to 'off'
  Serial.begin(9600);
  startTime = millis();
  activate();
}

// use the button asynchronously
void loop() {

  if(Serial.available()>0){
    char input=Serial.read();
    if(input=='p'){
      mode='p';
      Serial.println("Mod: LED-urile pare sunt aprinse.");
    }
    else if(input=='i'){
      mode='i';
      Serial.println("Mod: LED-urile impare sunt aprinse.");
    }
    else if(input=='a'){
      mode='a';
      Serial.println("Mod: Toate LED-urile sunt aprinse.");
    }
  }

  if( digitalRead(switchPin) == true ) {
    bSwitchDown = true;
    allOff();
    startTime = 0;
  }
  else {
    if( bSwitchDown )
      bSwitchDown = false;

    if( startTime + DELAY_TIME < millis() ) {
      activate();
      startTime = millis();
    }
  }
}

```

```

    }
}

void allOff() {
    strip.clear();
    strip.show();
}

void activate() {
    int sp = startPixel;
    char c;
    for( int i = 0; i < numPixels; i++ )
    {
        if(sp%2==0 && mode=='p')
            strip.setPixelColor(sp,i*3,i*2,i);
        else if(sp%2!=0 && mode=='i')
            strip.setPixelColor(sp, i, i*2, i*3 );
        else if(mode=='a')
        {

            strip.setPixelColor(sp, strip.Color(i * 3, i * 2, i));

        }

        strip.setBrightness(i);

        if( sp == numPixels/2 )
            sp = 0;
        else
            sp++;
    }

    strip.show();

    startPixel++;
    if( startPixel == 30 )
        startPixel = 0;

}

```

Alta varianta a codului, unde modificam culorile led-urilor de diferita paritate:

Pentru cele pare:

-daca numarul led-ului este divizibil cu 4, o sa fie de o culoare diferita fata de celelalte led-uri pare

-la fel si pentru cele impare, doar ca de data asta vom schimba culoarea led-urilor divizibile cu 3

```
#include <Adafruit_NeoPixel.h>
```

```
// the data pin for the NeoPixels
```

```
int neoPixelPin = 6;
```

```
// How many NeoPixels we will be using, change accordingly
```

```
int numPixels = 60;
```

```
// Instantiate the NeoPixel from the library
```

```
Adafruit_NeoPixel strip = Adafruit_NeoPixel(numPixels, neoPixelPin, NEO_GRB +  
NEO_KHZ800);
```

```
char mode='a';
```

```
// our button
```

```
int switchPin = 4;
```

```
// initial colors, you can change these
```

```
int red = 100;
```

```
int green = 255;
```

```
// a pre-processor macro
```

```
#define DELAY_TIME (1)
```

```
// stores the start time, use unsigned long to prevent overflow
```

```
unsigned long startTime;
```

```
int startPixel = 0;
```

```
boolean bSwitchDown = false;
```

```
void setup() {
```

```
    strip.begin(); // initialize the strip
```

```
    strip.show(); // make sure it is visible
```

```
    strip.clear(); // Initialize all pixels to 'off'
```

```
    Serial.begin(9600);
```

```
    startTime = millis();
```

```
    activate();
```

```
}
```

```
// use the button asynchronously
```

```
void loop() {
```

```
    if(Serial.available()>0){
```

```
        char input=Serial.read();
```

```
        if(input=='p'){
```

```
            mode='p';
```

```
            Serial.println("Mod: LED-urile pare sunt aprinse.");
```

```
        }
```

```
        else if(input=='i'){
```

```
            mode='i';
```

```

    Serial.println("Mod: LED-urile impare sunt aprinse.");
}
else if(input=='a'){
    mode='a';
    Serial.println("Mod: Toate LED-urile sunt aprinse.");
}
}

```

```

if( digitalRead(switchPin) == true ) {
    bSwitchDown = true;
    allOff();
    startTime = 0;
}
else {
    if( bSwitchDown )
        bSwitchDown = false;

    if( startTime + DELAY_TIME < millis() ) {
        activate();
        startTime = millis();
    }
}
}

```

```

void allOff() {
    strip.clear();
    strip.show();
}

```

```

void activate() {
    int sp = startPixel;
    char c;
    for( int i = 0; i < numPixels; i++ )
    {
        if(sp%2==0 && mode=='p'){
            if(sp%4==0)
                strip.setPixelColor(sp, i*2, i*3 );
            else
                strip.setPixelColor(sp,i*3,i*2,i);}
        else if(sp%2!=0 && mode=='i'){
            if(sp%3==0)
                strip.setPixelColor(sp, i, i*2, i*3 );
            else
                strip.setPixelColor(sp,i*2,0,0);}
        else if(mode=='a')
        {

            strip.setPixelColor(sp, strip.Color(i * 3, i * 2, i));

        }

        strip.setBrightness(i);
    }
}

```

```

        if( sp == numPixels/2 )
            sp = 0;
        else
            sp++;

    }

    strip.show();

    startPixel++;
    if( startPixel == 30 )
        startPixel = 0;

}

```

AVEM 3 MODURI DE FUNCTIONARE:

Modul “a”:

-sunt aprinse toate LED-urile

Modul “p”:

-sunt aprinse LED-urile pare

Modul “i”:

-sunt aprinse LED-urile impare

Modul “n”:

-sunt aprinse primele n LED-uri; numar citit de la monitorul serial

Primul cod, utilizand if-uri:

```
#include <Adafruit_NeoPixel.h>
```

```
// the data pin for the NeoPixels
```

```
int neoPixelPin = 6;
```

```
// How many NeoPixels we will be using, change accordingly
```

```
int numPixels = 60;
```

```
// Instantiate the NeoPixel from the library
```

```
Adafruit_NeoPixel strip = Adafruit_NeoPixel(numPixels, neoPixelPin, NEO_GRB +
NEO_KHZ800);
```

```
char mode='a';
```

```
// our button
```

```
int switchPin = 4;
```

```
// initial colors, you can change these
```

```
int red = 100;
```

```
int green = 255;
```

```
// a pre-processor macro
```

```
#define DELAY_TIME (1)
```

```
// stores the start time, use unsigned long to prevent overflow
```

```
unsigned long startTime;
```



```
int startPixel = 0;
boolean bSwitchDown = false;

void setup() {
  strip.begin(); // initialize the strip
  strip.show(); // make sure it is visible
  strip.clear(); // Initialize all pixels to 'off'
  Serial.begin(9600);
  startTime = millis();
  activate();
}

// use the button asynchronously
void loop() {

if(Serial.available()>0){
  char input=Serial.read();
  if(input=='p'){
    mode='p';
    Serial.println("Mod: LED-urile pare sunt aprinse.");
  }
  else if(input=='i'){
    mode='i';
    Serial.println("Mod: LED-urile impare sunt aprinse.");
  }
  else if(input=='a'){
    mode='a';
    Serial.println("Mod: Toate LED-urile sunt aprinse.");
  }
  else if(input=='1'){
    mode='1';
    Serial.println("Mod: Numarul LED-urilor: 1.");
  }

  }

  else if(input=='2'){
    mode='2';
    Serial.println("Mod: Numarul LED-urilor: 2.");
  }

  }

  else if(input=='3'){
    mode='3';
    Serial.println("Mod: Numarul LED-urilor: 3.");
  }

  }

  else if(input=='4'){
    mode='4';
    Serial.println("Mod: Numarul LED-urilor: 4.");
  }

  }
```

```
else if(input=='5'){
    mode='5';
    Serial.println("Mod: Numarul LED-urilor: 5.");
}

else if(input=='6'){
    mode='6';
    Serial.println("Mod: Numarul LED-urilor: 6.");
}

else if(input=='7'){
    mode='7';
    Serial.println("Mod: Numarul LED-urilor: 7.");
}

else if(input=='8'){
    mode='8';
    Serial.println("Mod: Numarul LED-urilor: 8.");
}

else if(input=='9'){
    mode='9';
    Serial.println("Mod: Numarul LED-urilor: 9.");
}

else if(input=='10'){
    mode='10';
    Serial.println("Mod: Numarul LED-urilor: 10.");
}

else if(input=='11'){
    mode='11';
    Serial.println("Mod: Numarul LED-urilor: 11.");
}

else if(input=='12'){
    mode='12';
    Serial.println("Mod: Numarul LED-urilor: 12.");
}

else if(input=='13'){
    mode='13';
    Serial.println("Mod: Numarul LED-urilor: 13.");
}
```

```

else if(input=='14'){
    mode='14';
    Serial.println("Mod: Numarul LED-urilor: 14.");
}

else if(input=='15'){
    mode='15';
    Serial.println("Mod: Numarul LED-urilor: 15.");
}

else if(input=='16'){
    mode='16';
    Serial.println("Mod: Numarul LED-urilor: 16.");
}
}

```

```

if( digitalRead(switchPin) == true ) {
    bSwitchDown = true;
    allOff();
    startTime = 0;
}
else {
    if( bSwitchDown )
        bSwitchDown = false;

    if( startTime + DELAY_TIME < millis() ) {
        activate();
        startTime = millis();
    }
}
}

```

```

void allOff() {
    strip.clear();
    strip.show();
}

```

```

void activate() {
    int sp = startPixel;
    char c;
    for( int i = 0; i < numPixels; i++ )
    {
        if(sp%2==0 && mode=='p'){
            if(sp%4==0)
                strip.setPixelColor(sp, i, i*2, i*3 );
            else
                strip.setPixelColor(sp,i*3,i*2,i);
        }
        else if(sp%2!=0 && mode=='i'){
            if(sp%3==0)

```

```

strip.setPixelColor(sp, i, i*2, i*3 );
else
strip.setPixelColor(sp,i*2,0,0);}
else if(mode=='a')
{ strip.setPixelColor(sp, strip.Color(i * 3, i * 2, i));
}
else if (mode=='1'){

    strip.setPixelColor(0,strip.Color(i,0,0));
}
else if (mode=='2'){
for(int m=0;m<2;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
}

else if (mode=='3'){
for(int m=0;m<3;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
}

else if (mode=='4'){
for(int m=0;m<4;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
}

else if (mode=='5'){
for(int m=0;m<5;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
}

else if (mode=='6'){
for(int m=0;m<6;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
}

else if (mode=='7'){
for(int m=0;m<7;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
}

else if (mode=='8'){
for(int m=0;m<8;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
}

else if (mode=='9'){
for(int m=0;m<9;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
}

else if (mode=='10'){
for(int m=0;m<10;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
}

```

```

    }

    else if (mode=='11'){
    for(int m=0;m<11;m++)
        strip.setPixelColor(m,strip.Color(i,0,0));
    }

    else if (mode=='12'){
    for(int m=0;m<12;m++)
        strip.setPixelColor(m,strip.Color(i,0,0));
    }

    else if (mode=='13'){
    for(int m=0;m<13;m++)
        strip.setPixelColor(m,strip.Color(i,0,0));
    }

    else if (mode=='14'){
    for(int m=0;m<14;m++)
        strip.setPixelColor(m,strip.Color(i,0,0));
    }

    else if (mode=='15'){
    for(int m=0;m<15;m++)
        strip.setPixelColor(m,strip.Color(i,0,0));
    }

    else if (mode=='16'){
    for(int m=0;m<16;m++)
        strip.setPixelColor(m,strip.Color(i,0,0));
    }

    strip.setBrightness(i);

    if( sp == numPixels/2 )
        sp = 0;
    else
        sp++;

}

strip.show();

startPixel++;
if( startPixel == 30 )
    startPixel = 0;

}

```

Varianta anterioara nu mergea pentru cazurile "10","11","12","13","14","15",
"16", asa ca, in loc de introducerea numarului in zecimal, am optat pentru scrierea acestuia in hexa, inafara numarului "10", unde am optat pentru litera "z", deoarece litera "a" era deja folosita pentru cazul "all leds on".

```
#include <Adafruit_NeoPixel.h>

// the data pin for the NeoPixels
int neoPixelPin = 6;

// How many NeoPixels we will be using, change accordingly
int numPixels = 60;

// Instantiate the NeoPixel from the library
Adafruit_NeoPixel strip = Adafruit_NeoPixel(numPixels, neoPixelPin, NEO_GRB +
NEO_KHZ800);
char mode='a';
// our button
int switchPin = 4;

// initial colors, you can change these
int red = 100;
int green = 255;

// a pre-processor macro
#define DELAY_TIME (1)

// stores the start time, use unsigned long to prevent overflow
unsigned long startTime;

int startPixel = 0;
boolean bSwitchDown = false;

void setup() {
  strip.begin(); // initialize the strip
  strip.show(); // make sure it is visible
  strip.clear(); // Initialize all pixels to 'off'
  Serial.begin(9600);
  startTime = millis();
  activate();
}

// use the button asynchronously
void loop() {

if(Serial.available()>0){
  char input=Serial.read();
  if(input=='p'){
    mode='p';
    Serial.println("Mod: LED-urile pare sunt aprinse.");
  }
  else if(input=='i'){
    mode='i';
    Serial.println("Mod: LED-urile impare sunt aprinse.");
  }
}
```

```
}  
else if(input=='a'){  
    mode='a';  
    Serial.println("Mod: Toate LED-urile sunt aprinse.");  
}  
else if(input=='1'){  
    mode='1';  
    Serial.println("Mod: Numarul LED-urilor: 1.");  
  
}  
  
else if(input=='2'){  
    mode='2';  
    Serial.println("Mod: Numarul LED-urilor: 2.");  
  
}  
  
else if(input=='3'){  
    mode='3';  
    Serial.println("Mod: Numarul LED-urilor: 3.");  
  
}  
  
else if(input=='4'){  
    mode='4';  
    Serial.println("Mod: Numarul LED-urilor: 4.");  
  
}  
  
else if(input=='5'){  
    mode='5';  
    Serial.println("Mod: Numarul LED-urilor: 5.");  
  
}  
  
else if(input=='6'){  
    mode='6';  
    Serial.println("Mod: Numarul LED-urilor: 6.");  
  
}  
  
else if(input=='7'){  
    mode='7';  
    Serial.println("Mod: Numarul LED-urilor: 7.");  
  
}  
  
else if(input=='8'){  
    mode='8';  
    Serial.println("Mod: Numarul LED-urilor: 8.");  
  
}  
  
else if(input=='9'){
```

```

    mode='9';
    Serial.println("Mod: Numarul LED-urilor: 9.");

}

else if(input=='10'){
    mode='10';
    Serial.println("Mod: Numarul LED-urilor: 10.");

}

else if(input=='b'){
    mode='b';
    Serial.println("Mod: Numarul LED-urilor: 11.");

}

else if(input=='c'){
    mode='c';
    Serial.println("Mod: Numarul LED-urilor: 12.");

}

else if(input=='d'){
    mode='d';
    Serial.println("Mod: Numarul LED-urilor: 13.");

}

else if(input=='e'){
    mode='e';
    Serial.println("Mod: Numarul LED-urilor: 14.");

}

else if(input=='f'){
    mode='f';
    Serial.println("Mod: Numarul LED-urilor: 15.");

}

}

if( digitalRead(switchPin) == true ) {
    bSwitchDown = true;
    allOff();
    startTime = 0;
}
else {
    if( bSwitchDown )
        bSwitchDown = false;

    if( startTime + DELAY_TIME < millis() ) {

```



```

        activate();
        startTime = millis();
    }
}

void allOff() {
    strip.clear();
    strip.show();
}

void activate() {
    int sp = startPixel;
    char c;
    for( int i = 0; i < numPixels; i++ )
    {
        if(sp%2==0 && mode=='p'){
            if(sp%4==0)
                strip.setPixelColor(sp, i, i*2, i*3 );
            else
                strip.setPixelColor(sp,i*3,i*2,i);
        }
        else if(sp%2!=0 && mode=='i'){
            if(sp%3==0)
                strip.setPixelColor(sp, i, i*2, i*3 );
            else
                strip.setPixelColor(sp,i*2,0,0);
        }
        else if(mode=='a')
        { strip.setPixelColor(sp, strip.Color(i * 3, i * 2, i));
        }
        else if (mode=='1'){

            strip.setPixelColor(0,strip.Color(i,0,0));
        }
        else if (mode=='2'){
            for(int m=0;m<2;m++)
                strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='3'){
            for(int m=0;m<3;m++)
                strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='4'){
            for(int m=0;m<4;m++)
                strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='5'){
            for(int m=0;m<5;m++)
                strip.setPixelColor(m,strip.Color(i,0,0));
        }
    }
}

```

```

        else if (mode=='6'){
        for(int m=0;m<6;m++)
            strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='7'){
        for(int m=0;m<7;m++)
            strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='8'){
        for(int m=0;m<8;m++)
            strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='9'){
        for(int m=0;m<9;m++)
            strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='10'){
        for(int m=0;m<10;m++)
            strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='b'){
        for(int m=0;m<11;m++)
            strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='c'){
        for(int m=0;m<12;m++)
            strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='d'){
        for(int m=0;m<13;m++)
            strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='e'){
        for(int m=0;m<14;m++)
            strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='f'){
        for(int m=0;m<15;m++)
            strip.setPixelColor(m,strip.Color(i,0,0));
        }

        strip.setBrightness(i);

    if( sp == numPixels/2 )

```

```
        sp = 0;
    else
        sp++;

}

strip.show();

startPixel++;
if( startPixel == 30 )
    startPixel = 0;

}
```

CITIREA SI AFISAREA DISTANTEI CITITE PE LCD:

```
#include <Adafruit_NeoPixel.h>
#include <LiquidCrystal_I2C.h>
#include <NewPing.h>
#include <Wire.h>
```

```
#define TRIGGER_PIN 10 // Pinul Trig
#define ECHO_PIN 13 // Pinul Echo
#define MAX_DISTANCE 16 // Distanța maximă (în centimetri)
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2); // Actualizează 0x27 cu adresa corectă dacă este necesar
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
// the data pin for the NeoPixels
int neoPixelPin = 6;
```

```
// How many NeoPixels we will be using, charge accordingly
int numPixels = 60;
```

```
// Instantiate the NeoPixel from the library
Adafruit_NeoPixel strip = Adafruit_NeoPixel(numPixels, neoPixelPin, NEO_GRB +
NEO_KHZ800);
char mode='a';
// our button
int switchPin = 4;
```

```
// initial colors, you can change these
int red = 100;
int green = 255;
```

```
// a pre-processor macro
#define DELAY_TIME (1)
```

```
// stores the start time, use unsigned long to prevent overflow
unsigned long startTime;
```

```
int startPixel = 0;
boolean bSwitchDown = false;
```

```
void setup() {
```

```
    lcd.init();
    lcd.begin(16,2);
    strip.begin(); // initialize the strip
    strip.show(); // make sure it is visible
    strip.clear(); // Initialize all pixels to 'off'
    Serial.begin(9600);
    startTime = millis();
```

```
}
```

```
void loop(){
    delay(1000);
    unsigned int uS=sonar.ping();
```

```

    unsigned int cm=sonar.convert_cm(uS);
    Serial.print("distanța este:");
    Serial.print(cm);
    Serial.print(" cm\n");

    lcd.init();
    lcd.setCursor(0,0);
    lcd.print("distanța citită:");
    lcd.print(cm);
}

#include <Adafruit_NeoPixel.h>
#include <LiquidCrystal_I2C.h>
#include <NewPing.h>
#include <Wire.h>

#define TRIGGER_PIN 10 // Pinul Trig
#define ECHO_PIN 13 // Pinul Echo
#define MAX_DISTANCE 16 // Distanța maximă (în centimetri)

LiquidCrystal_I2C lcd(0x27, 16, 2); // Actualizează 0x27 cu adresa corectă dacă este necesar
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
// the data pin for the NeoPixels
int neoPixelPin = 6;

// How many NeoPixels we will be using, change accordingly
int numPixels = 60;

// Instantiate the NeoPixel from the library
Adafruit_NeoPixel strip = Adafruit_NeoPixel(numPixels, neoPixelPin, NEO_GRB +
NEO_KHZ800);
char mode='a';
// our button
int switchPin = 4;

// initial colors, you can change these
int red = 100;
int green = 255;

// a pre-processor macro
#define DELAY_TIME (1)

// stores the start time, use unsigned long to prevent overflow
unsigned long startTime;

int startPixel = 0;
boolean bSwitchDown = false;

void setup() {

//lcd.begin(16,2);

```

```
lcd.init();
lcd.begin(16,2);
strip.begin(); // initialize the strip
strip.show(); // make sure it is visible
strip.clear(); // Initialize all pixels to 'off'
Serial.begin(9600);
startTime = millis();
activate();
}
```

```
// use the button asynchronously
void loop() {
```

```
    delay(1000);
    unsigned int uS=sonar.ping();
    unsigned int cm=sonar.convert_cm(uS);
    Serial.print("Distanta este:");
    Serial.print(cm);
    Serial.print(" cm\n");
```

```
    lcd.init();
    lcd.backlight();
    lcd.setCursor(0,0);
    lcd.print("distanta citita:");
    lcd.print(cm);
```

```
    if(Serial.available()>0){
        char input=Serial.read();
        if(input=='p'){
            mode='p';
            Serial.println("Mod: LED-urile pare sunt aprinse.");
        }
        else if(input=='i'){
            mode='i';
            Serial.println("Mod: LED-urile impare sunt aprinse.");
        }
        else if(input=='a'){
            mode='a';
            Serial.println("Mod: Toate LED-urile sunt aprinse.");
        }
        else if(input=='1'){
            mode='1';
            Serial.println("Mod: Numarul LED-urilor: 1.");
        }

        else if(input=='2'){
            mode='2';
            Serial.println("Mod: Numarul LED-urilor: 2.");
        }
    }
```

```
else if(input=='3'){
    mode='3';
    Serial.println("Mod: Numarul LED-urilor: 3.");
}

else if(input=='4'){
    mode='4';
    Serial.println("Mod: Numarul LED-urilor: 4.");
}

else if(input=='5'){
    mode='5';
    Serial.println("Mod: Numarul LED-urilor: 5.");
}

else if(input=='6'){
    mode='6';
    Serial.println("Mod: Numarul LED-urilor: 6.");
}

else if(input=='7'){
    mode='7';
    Serial.println("Mod: Numarul LED-urilor: 7.");
}

else if(input=='8'){
    mode='8';
    Serial.println("Mod: Numarul LED-urilor: 8.");
}

else if(input=='9'){
    mode='9';
    Serial.println("Mod: Numarul LED-urilor: 9.");
}

else if(input=='10'){
    mode='10';
    Serial.println("Mod: Numarul LED-urilor: 10.");
}

else if(input=='b'){
    mode='b';
    Serial.println("Mod: Numarul LED-urilor: 11.");
}
```

```

else if(input=='c'){
    mode='c';
    Serial.println("Mod: Numarul LED-urilor: 12.");
}

else if(input=='d'){
    mode='d';
    Serial.println("Mod: Numarul LED-urilor: 13.");
}

else if(input=='e'){
    mode='e';
    Serial.println("Mod: Numarul LED-urilor: 14.");
}

else if(input=='f'){
    mode='f';
    Serial.println("Mod: Numarul LED-urilor: 15.");
}
else if(input=='l')
{
    mode='l';
    Serial.println("Mod: distanta citita.");
    for(int m=0;m<cm;m++)
        strip.setPixelColor(m,strip.Color(255,0,0));
}

}

if( digitalRead(switchPin) == true ) {
    bSwitchDown = true;
    allOff();
    startTime = 0;
}
else {
    if( bSwitchDown )
        bSwitchDown = false;

    if( startTime + DELAY_TIME < millis() ) {
        activate();
        startTime = millis();
    }
}

}

void allOff() {
    strip.clear();
    strip.show();
}

```



```

}

void activate() {
    int sp = startPixel;
    char c;
    for( int i = 0; i < numPixels; i++ )
    {
        if(sp%2==0 && mode=='p'){
            if(sp%4==0)
                strip.setPixelColor(sp, i, i*2, i*3 );
            else
                strip.setPixelColor(sp,i*3,i*2,i);
        }
        else if(sp%2!=0 && mode=='i'){
            if(sp%3==0)
                strip.setPixelColor(sp, i, i*2, i*3 );
            else
                strip.setPixelColor(sp,i*2,0,0);
        }
        else if(mode=='a')
        { strip.setPixelColor(sp, strip.Color(i * 3, i * 2, i));
        }
        else if (mode=='1'){

            strip.setPixelColor(0,strip.Color(i,0,0));
        }
        else if (mode=='2'){
            for(int m=0;m<2;m++)
                strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='3'){
            for(int m=0;m<3;m++)
                strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='4'){
            for(int m=0;m<4;m++)
                strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='5'){
            for(int m=0;m<5;m++)
                strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='6'){
            for(int m=0;m<6;m++)
                strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='7'){
            for(int m=0;m<7;m++)
                strip.setPixelColor(m,strip.Color(i,0,0));
        }
    }
}

```

```

        else if (mode=='8'){
for(int m=0;m<8;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='9'){
for(int m=0;m<9;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='10'){
for(int m=0;m<10;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='b'){
for(int m=0;m<11;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='c'){
for(int m=0;m<12;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='d'){
for(int m=0;m<13;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='e'){
for(int m=0;m<14;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
        }

        else if (mode=='f'){
for(int m=0;m<15;m++)
    strip.setPixelColor(m,strip.Color(i,0,0));
        }

        strip.setBrightness(i);

        if( sp == numPixels/2 )
            sp = 0;
        else
            sp++;

    }

    strip.show();

```

```
startPixel++;  
if( startPixel == 30 )  
    startPixel = 0;
```

```
}
```

CODUL URMATOR DETECTEAZA DISTANTA DINTRE SENZOR SI OBIECT, SI IN FUNCTIE DE DISTANTA DINTRE ACESTEA, VA AVEA UN NUMAR DE LED-URI APRINSE:

DESCRIEREA DISPOZITIVULUI(pentru o distanta de maxim 16 cm, deoarece avem un numar de 16 led-uri)

-cu cat avem obiectul mai aproape, cu atat avem mai putine LED-uri albastre aprinse

-luminozitatea led-urilor albastre este mai slaba decat a celor rosii

-aceasta informatie este citita odata la o secunda

```
#include <Adafruit_NeoPixel.h>
#include <LiquidCrystal_I2C.h>
#include <NewPing.h>
#include <Wire.h>

#define TRIGGER_PIN 10 // Pinul Trig
#define ECHO_PIN 13 // Pinul Echo
#define MAX_DISTANCE 200 // Distanța maximă (în centimetri)

LiquidCrystal_I2C lcd(0x27, 16, 2); // Actualizează 0x27 cu adresa corectă dacă este
necesar
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

// Pinul de date pentru NeoPixels
int neoPixelPin = 6;

// Numărul de NeoPixels
int numPixels = 16; // Numai primii 16 pixeli pentru exemplul tău

// Instanțierea NeoPixel-ului din librărie
Adafruit_NeoPixel strip = Adafruit_NeoPixel(numPixels, neoPixelPin, NEO_GRB +
NEO_KHZ800);

void setup() {
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();
  lcd.begin(16, 2);
  strip.begin(); // inițializează strip-ul NeoPixel
  strip.show(); // asigură-te că strip-ul este vizibil
  strip.clear(); // inițializează toate pixelii cu culoarea implicită (negru)
}

void loop() {
  // Citirea distanței de la senzorul ultrasonic
  unsigned int uS = sonar.ping();
  unsigned int cm = sonar.convert_cm(uS);

  // Afișarea distanței pe LCD
  lcd.clear();
```

```

    lcd.setCursor(0, 0);
    lcd.print("Distanța:");
    lcd.print(cm);
    lcd.print(" cm");

    if (cm <= 16) {

        //int brightness = map(cm, 0, 16, 50, 255);
        //aceasta linie de cod putea fi utilizata in cazul in care se dorea o //luminozitate
        variabila, in functie de nivelul de apropiere al obiectului fata //de senzor

        for (int i = 0; i < numPixels; i++) {
            if (i < cm) {

                strip.setPixelColor(i, strip.Color(0, 0, 255));
                strip.setBrightness(50);
            } else {

                strip.setPixelColor(i, strip.Color(255, 0, 0));
                strip.setBrightness(100);
            }
            // strip.setBrightness(brightness);
        }
        strip.show();
    }

    delay(100);
}

```

Am adus cateva modificari codului astfel:

Sa avem lumina rosie doar in cazul in care obiectul e foarte aproape de senzor

Implementarea unei parole pentru activarea accesului la Lcd:

```
#include <LiquidCrystal.h>
```

```
LiquidCrystal lcd(8, 9, 4, 5, 6, 7); // Conexiunea LCD
```

```
const int ButtonPin = A0; // Pinul pentru butoane
```

```
int adc_tasta_intrare = 0;
```

```
int tasta_anterioara = 0;
```

```
int numar_apasari = 0;
```

```
int numar-incercari=3;
```

```
String parola_corecta = "1234"; // Parola corectă
```

```
String parola_introdusa = ""; // Parola introdusă de utilizator
```

```
bool parola_activata = false; // Starea parola_activata
```

```
void setup() {
```

```
    Serial.begin(9600);
```

```
    lcd.begin(16, 2); // Inițializare LCD 16x2
```

```
    lcd.print("Introduceti");
```

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

```
    lcd.print("parola: ");
```

```
}
```

```
void loop() {
```

```
    adc_tasta_intrare = analogRead(ButtonPin);
```

```
    int tasta = getKey(adc_tasta_intrare);
```

```
    if (tasta != tasta_anterioara) {
```

```
        if (tasta >= 1 && tasta <= 4) {
```

```
            parola_introdusa += (char)(tasta + '0'); // Conversie din int în char
```

```
            lcd.setCursor(8 + parola_introdusa.length() - 1, 1);
```

```
            lcd.print("*"); // Afișează caracterul "*" pentru fiecare cifră introdusă
```

```
            if (parola_introdusa.length() == 4 && numar-incercari>0) {
```

```
                // Verifică parola introdusă
```

```
                if (parola_introdusa == parola_corecta) {
```

```
                    parola_activata = true;
```

```
                    lcd.clear();
```

```
                    lcd.print("Acces permis");
```

```
                } else {
```

```
                    lcd.clear();
```

```
                    lcd.print("Parola incorecta");
```

```
                    delay(2000);
```

```
                    lcd.clear();
```

```
                    reseteazaParola();
```

```
                    numar-incercari=numar-incercari-1;
```

```
                    lcd.setCursor(0,0);
```

```
                    lcd.print("Incerari:");
```

```

        lcd.setCursor(0,1);
        lcd.print(numar_incercari);
        delay(2000);
        lcd.clear();
        if(numar_incercari==0)
        {
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("ACCES BLOCAT");
        }
    }
}

tasta_anterioara = tasta;

// Poți adăuga aici alte acțiuni în funcție de starea parola_activata
if (parola_activata) {
    // LCD-ul este activat, poți afișa alte mesaje sau efectua alte acțiuni
}

int getKey(int intrare) {
    if (intrare > 1000) return 0;
    if (intrare < 50)    return 1;
    if (intrare < 195)  return 2;
    if (intrare < 380)  return 3;
    if (intrare < 555)  return 4;
    if (intrare < 790)  return 5;
    if (intrare < 900)  return 6;
    return 0;
}

void reseteazaParola() {
    numar_apasari = 0;
    parola_introdusa = ""; // Resetează parola introdusă
    lcd.setCursor(8, 1);
    lcd.print("    "); // Șterge caracterele introduse pentru parolă
}

```

IMPLEMENTAREA UNUI CALCULATOR FOLOSIND ARDUINO UNO

Dispozitivele folosite:

- arduino uno
- lcd 1602A
- lcd keypad shield

CONTORIZARE NUMAR APASARI BUTON SELECT

```
#include <LiquidCrystal.h>
```

```
#include <Arduino.h>
```

```
LiquidCrystal lcd(8,9,4,5,6,7);
```

```
const int ButtonPin=A0;
```

```
int adc_intrare=0;
```

```
int tasta;
```

```
int tasta_anterioara;
```

```
int numar_apasari=0;
```

```
void setup()
```

```
{
```

```
  Serial.begin(9600);
```

```
  lcd.begin(16,2);
```

```
  lcd.print("Numarator");
```

```
  delay(2000);
```

```
  lcd.clear();
```

```
  lcd.print("Numar apasari:");
```

```
}
```

```
void loop()
```

```
{
```

```
  adc_intrare=analogRead(ButtonPin);
```

```
  tasta=getKey(adc_intrare);
```

```
  Serial.print("Butonul apasat este:");
```

```
  Serial.println(tasta);
```

```
  if(tasta==5 && tasta!=tasta_anterioara)
```

```
  {
```

```
    numar_apasari++;
```

```
    lcd.setCursor(1,1);
```

```
    lcd.print(numar_apasari);
```

```
  }
```

```
  tasta_anterioara=tasta;
```

```
}
```



```

int getKey(int intrare)
{
    if(intrare>1000) return 0;
    if(intrare<50) return 1;
    if(intrare<195) return 2;
    if(intrare<380) return 3;
    if(intrare<555) return 4;
    if(intrare<790) return 5;
    if(intrare<900) return 6;
    return 0;
}
#include <LiquidCrystal.h>
#include <Arduino.h>

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

const int ButtonPin = A0;

int adc_tasta_intrare = 0;
int tasta = 0;
int tastaAnterioara;

String input = "";

float nr1, nr2, rezultat;
char operatie;

void setup() {
    Serial.begin(9600);
    lcd.begin(16, 2);
    lcd.print("Calculator");
}

void loop() {
    adc_tasta_intrare = analogRead(ButtonPin);
    tasta = getKey(adc_tasta_intrare);

    Serial.print("Butonul apasat este:");
    Serial.println(tasta);

    if (tasta != tastaAnterioara) {
        if (tasta == 1) {
            addDigit('1');
        } else if (tasta == 2) {
            addDigit('2');
        } else if (tasta == 3) {
            addDigit('3');
        } else if (tasta == 4) {
            addDigit('4');
        } else if (tasta == 5) {
            selectOperation('+');
        } else if (tasta == 6) {
            resetCalculator();
        }
    }
}

```

```

        tastaAnterioara = tasta;
    }
}

int getKey(int input) {
    if (input > 1000) return 0;
    if (input < 50) return 1;
    if (input < 195) return 2;
    if (input < 380) return 3;
    if (input < 555) return 4;
    if (input < 790) return 5;
    if (input < 900) return 6;
    return 0;
}

void addDigit(char digit) {
    input += digit;
    lcd.print(digit);
}

void selectOperation(char op) {
    if (input.length() > 0) {
        if (operatie == '\0') {
            nr1 = input.toFloat();
            input = "";
            operatie = op;
            lcd.clear();
            lcd.print(nr1);
            lcd.print(operatie);
        } else {
            nr2 = input.toFloat();
            switch (operatie) {
                case '+': rezultat = addition(nr1, nr2); break;
                case '-': rezultat = subtraction(nr1, nr2); break;
                case '*': rezultat = multiplication(nr1, nr2); break;
                case '/': rezultat = division(nr1, nr2); break;
                default: rezultat = 0;
            }
            lcd.clear();
            lcd.print(rezultat);
            input = String(rezultat);
            operatie = '\0';
        }
    }
}

float addition(float num1, float num2) {
    return num1 + num2;
}

float subtraction(float num1, float num2) {
    return num1 - num2;
}

```

```
float multiplication(float num1, float num2) {  
    return num1 * num2;  
}
```

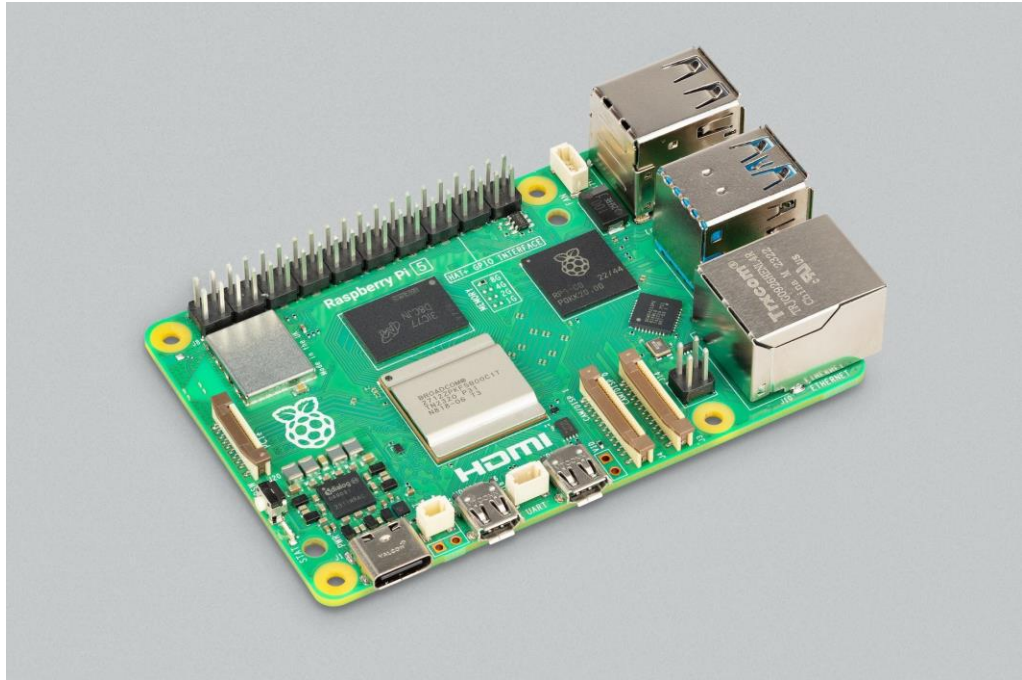
```
float division(float num1, float num2) {  
    if (num2 != 0) {  
        return num1 / num2;  
    } else {  
        return 0; // Tratare caz de împărțire la zero (poți adăuga un mesaj de eroare sau altă logica  
        dorită)  
    }  
}
```

```
void resetCalculator() {  
    lcd.clear();  
    input = "";  
    operatie = '\0';  
}
```

REALIZAREA UNUI SISTEM DE SECURITATE FOLOSIND RASPERRY PI 5

Acest sistem este unul simplu, cu ajutorul caruia putem vedea in timp real cine se apropie de casa, numarul de introduceri corecte/gresite ale parolei la usa de la intrare. Codul se actualizeaza odata la 30 de minute, astfel incat sa fie mai greu de ghicit. Datele vor fi stocate intr-o baza de date, acces la aplicatie poate avea doar proprietarul.

(Aplicatia are nevoie de imbunatatiri, inca nu a ajuns in stagiul final)



Configuratia pinilor:



3V3 power	1	2	5V power
GPIO 2 (SDA)	3	4	5V power
GPIO 3 (SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD)
Ground	9	10	GPIO 15 (RXD)
GPIO 17	11	12	GPIO 18 (PCM_CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3V3 power	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	Ground
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
Ground	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM_FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM_DIN)
Ground	39	40	GPIO 21 (PCM_DOUT)

40 GPIO Pins Description of Raspberry Pi 5

Un prim program de testare a functionalitatii lui raspberry pi 5 este:

-Blink LED:

-Conectam un led si o rezistenta in serie pe breadbord

-pinul de ground va fi conectat la masa de pe raspberry pi 5

-pinul de alimentare va fi conectat la GPIO17 de pe raspberry

Codul utilizat:

```
import gpiod
```

```
import time
```

```
import gpiod
```

```
import time
```

```
LED_PIN=17
```

```
chip=gpiod.Chip('gpiochip4')
```

```
led_line=chip.get_line(LED_PIN)
```

```
led_line.request(consumer="LED", type=gpiod.LINE_REQ_DIR_OUT)
```

```
try:
```

```
    while True:
```

```
        led_line.set_value(1)
```

```
        time.sleep(1)
```

```
        led_line.set_value(0)
```

```
        time.sleep(1) #sleep for one second
```

```
except KeyboardInterrupt:
```

```
    print("Program intrerupt de la tastatura")
```

```
finally:
```

```
    led_line.release()
```

LED-ul va sta aprins pentru o secunda, iar mai apoi stins pentru inca o secunda.

Am realizat un alt program de testare, in care vom instala librariile corespunzatoare pentru afisarea mesajelor pe LCD prin i2c.

```
from signal import signal, SIGTERM, SIGHUP, pause
```

```
from rpi_lcd import LCD
```

```
lcd = LCD()
```

```
def safe_exit(signum, frame):
```

```
    exit(1)
```

```
try:
```

```
signal(SIGTERM, safe_exit)
signal(SIGHUP, safe_exit)
```

```
lcd.text("Hello,", 1)
lcd.text("Raspberry Pi!", 2)
```

```
pause()
```

```
except KeyboardInterrupt:
    pass
```

```
finally:
    lcd.clear()
```

REALIZAREA UNUI MASURATOR DE DISTANTA PENTRU SISTEMUL DE SECURITATE:

```
#from signal import signal, SIGTERM, SIGHUP, pause
from rpi_lcd import LCD
from gpiozero import DistanceSensor
import time
```

```
#definirea pinilor senzorului
TRIG_PIN=23
ECHO_PIN=24
```

```
lcd=LCD()
```

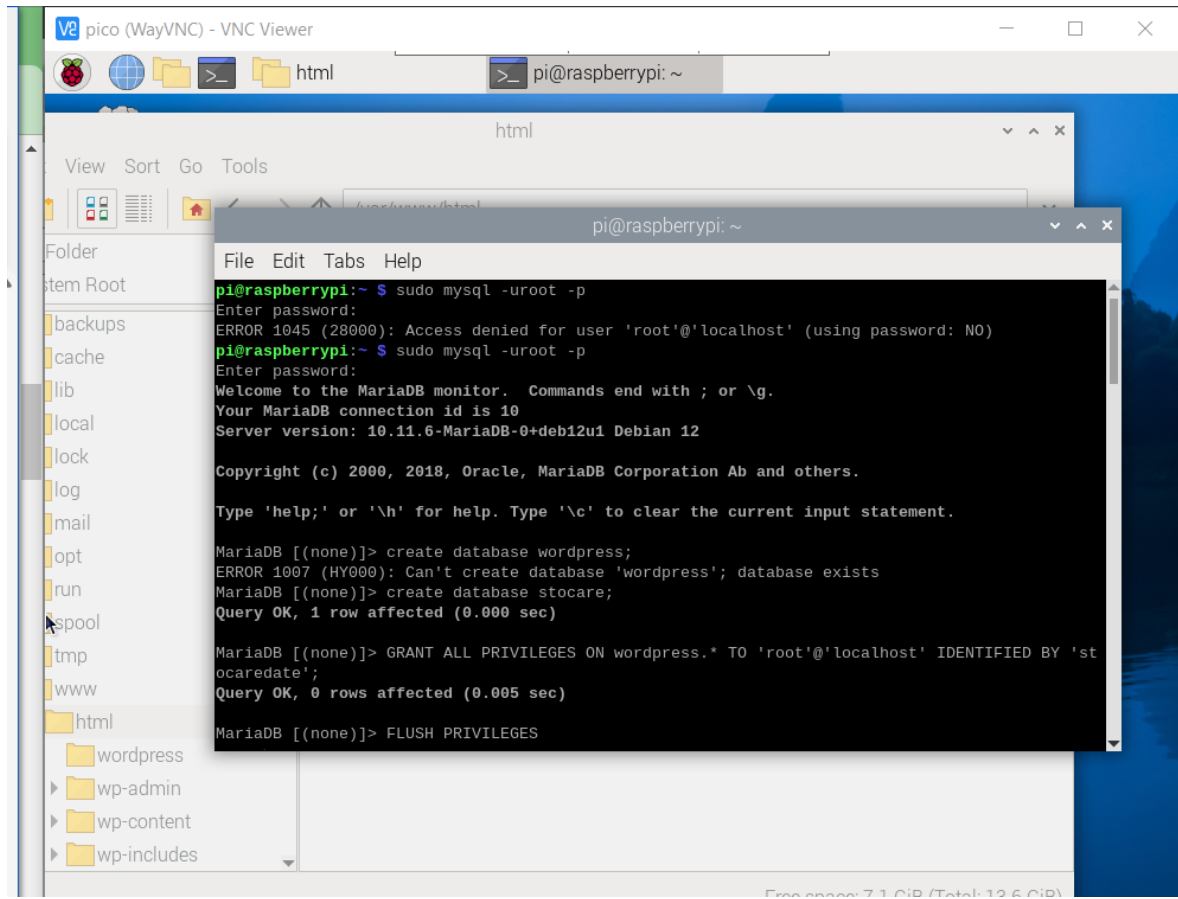
```
#crearea senzorului de distanta
sensor=DistanceSensor(echo=ECHO_PIN,trigger=TRIG_PIN)
```

```
try:
    lcd.text("Salut,",1)
    lcd.text("Raspberry Pi5!",2)
    while True:
        distance=sensor.distance*100 #conversie la cm
        lcd.text(f"Distanța: {distance:.2f} cm",3)
        time.sleep(1)
```

```
except KeyboardInterrupt:
    pass
```

```
finally:
    lcd.clear()
    sensor.close()
```

Pentru stocarea datelor am realizat o baza de date, unde am realizat un tabel in care am stocat datele citite de senzor,etc.



```
pi@raspberrypi:~$ sudo mysql -u root -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
pi@raspberrypi:~$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 10
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12

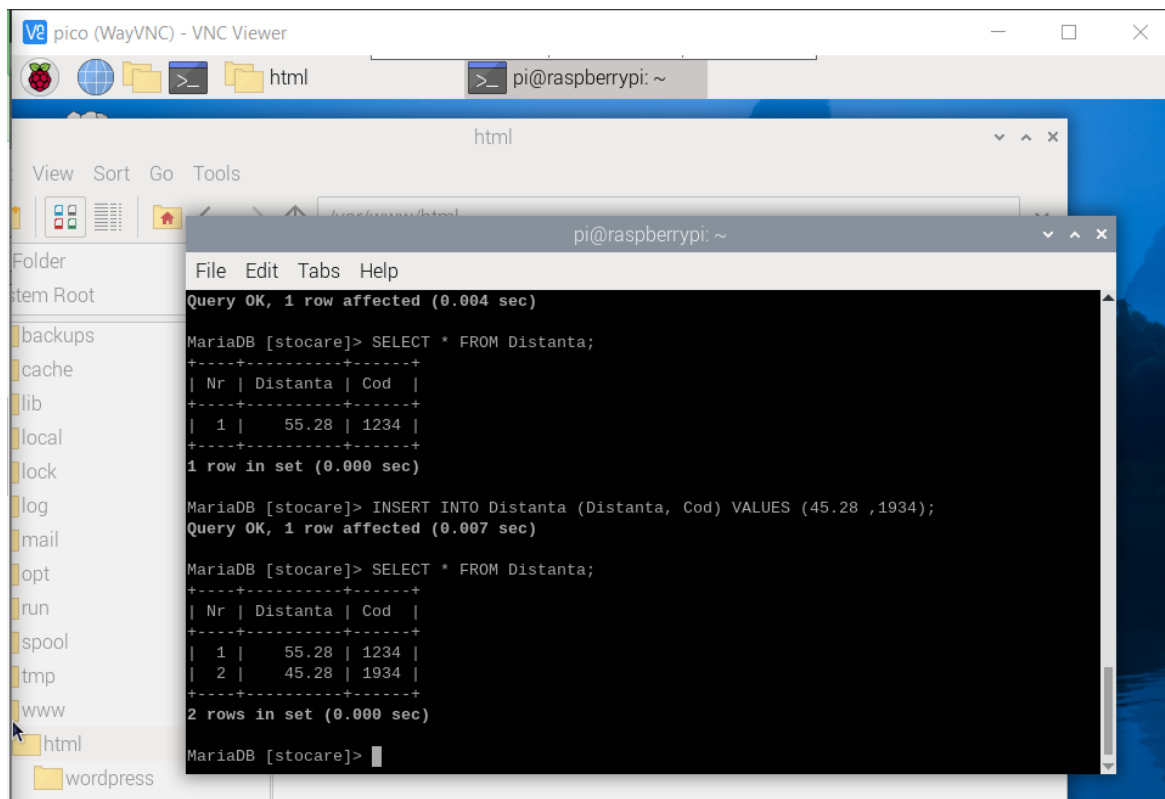
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database wordpress;
ERROR 1007 (HY000): Can't create database 'wordpress'; database exists
MariaDB [(none)]> create database stocare;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON wordpress.* TO 'root'@'localhost' IDENTIFIED BY 'stocaredate';
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]> FLUSH PRIVILEGES
```

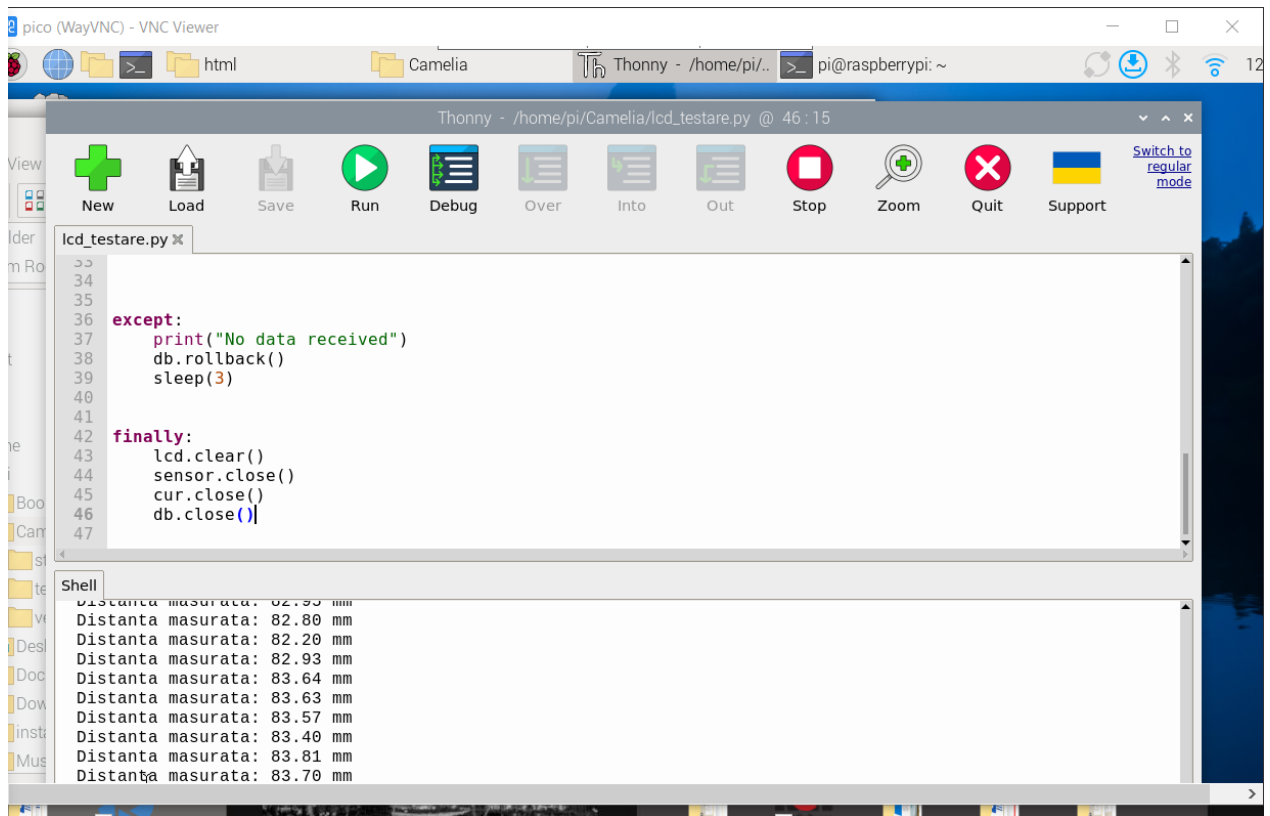


```
MariaDB [(none)]> SELECT * FROM Distanta;
+-----+-----+
| Nr | Distanta | Cod |
+-----+-----+
| 1 | 55.28 | 1234 |
+-----+-----+
1 row in set (0.000 sec)

MariaDB [(none)]> INSERT INTO Distanta (Distanta, Cod) VALUES (45.28 ,1934);
Query OK, 1 row affected (0.007 sec)

MariaDB [(none)]> SELECT * FROM Distanta;
+-----+-----+
| Nr | Distanta | Cod |
+-----+-----+
| 1 | 55.28 | 1234 |
| 2 | 45.28 | 1934 |
+-----+-----+
2 rows in set (0.000 sec)

MariaDB [(none)]>
```



Codul de mai jos l-am folosit pentru a testa daca datele sunt stocate corect in tabelul din baza de date:

```

from rpi_lcd import LCD
from gpiozero import DistanceSensor
import time
import MySQLdb

#definirea pinilor senzorului
ECHO_PIN=23
TRIG_PIN=24

#variabile sql:
db=MySQLdb.connect(host="localhost",user="root",password="stocaredate",cur=db.cursor())

lcd=LCD()

#crearea senzorului de distanta
sensor=DistanceSensor(echo=ECHO_PIN,trigger=TRIG_PIN)

try:
    lcd.text("Salut,",1)
    lcd.text("Raspberry Pi5!",2)
    while True:
        distance=sensor.distance*1000 #conversie la mm
        print(f"Distanța măsurată: {distance:.2f} mm")
        lcd.text(f"Distanța: {distance:.2f} mm",3)
        sql=("""Insert into distancelog(distance,1) Values (%s,%s)""",(distance,1))

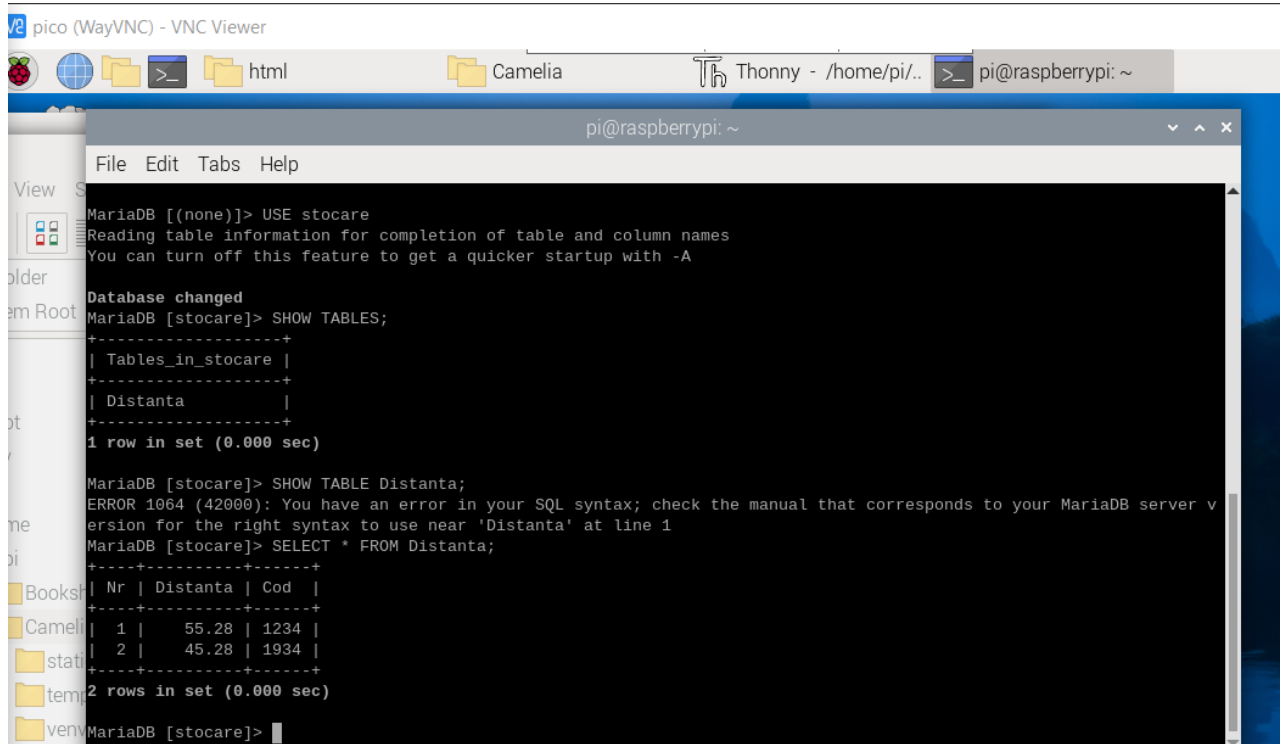
```



```
try:
    print("scrie in baza de date")
    cur.execute(*sql)
    db.commit()
    time.sleep(1)
```

```
except:
    print("No data received")
    db.rollback()
    sleep(3)
```

```
finally:
    lcd.clear()
    sensor.close()
    cur.close()
    db.close()
```



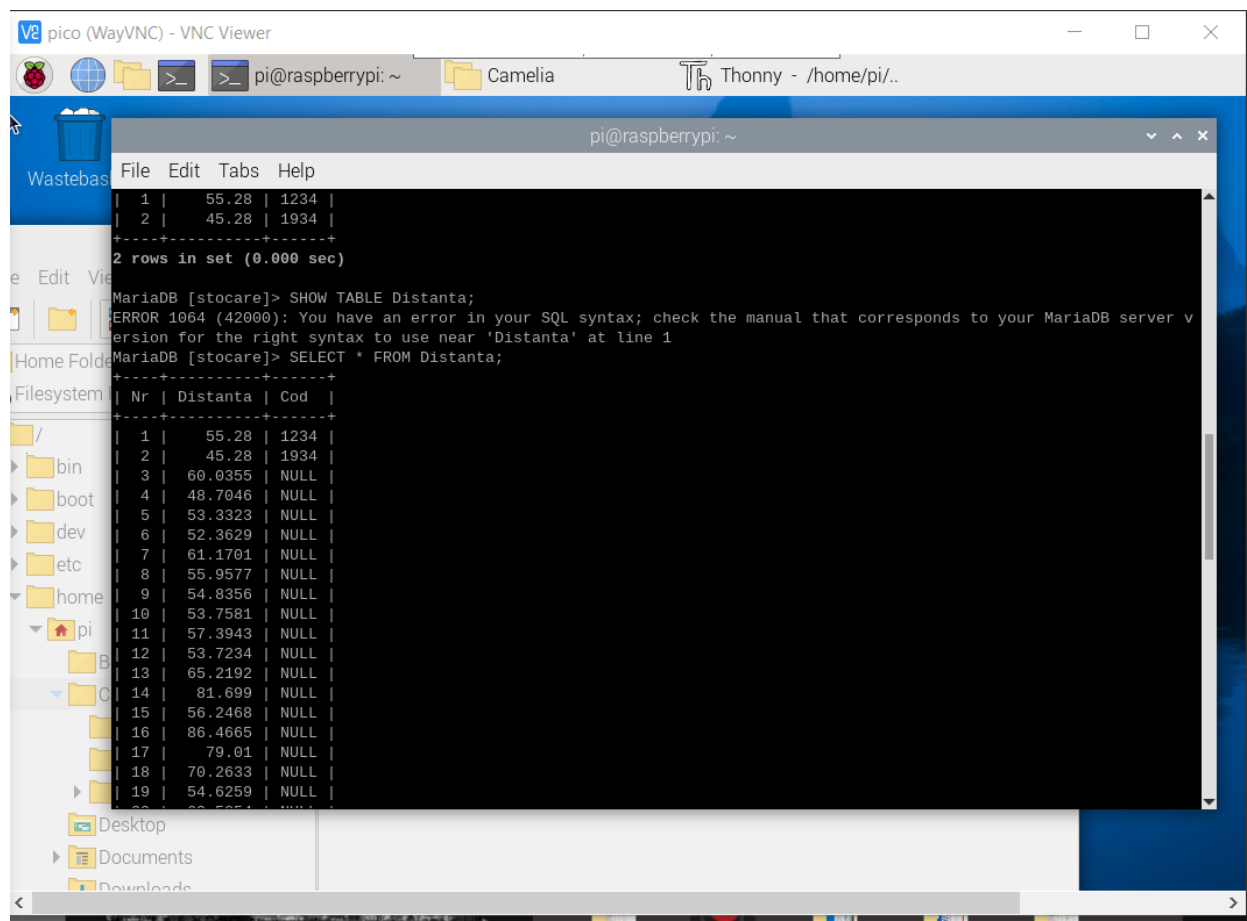
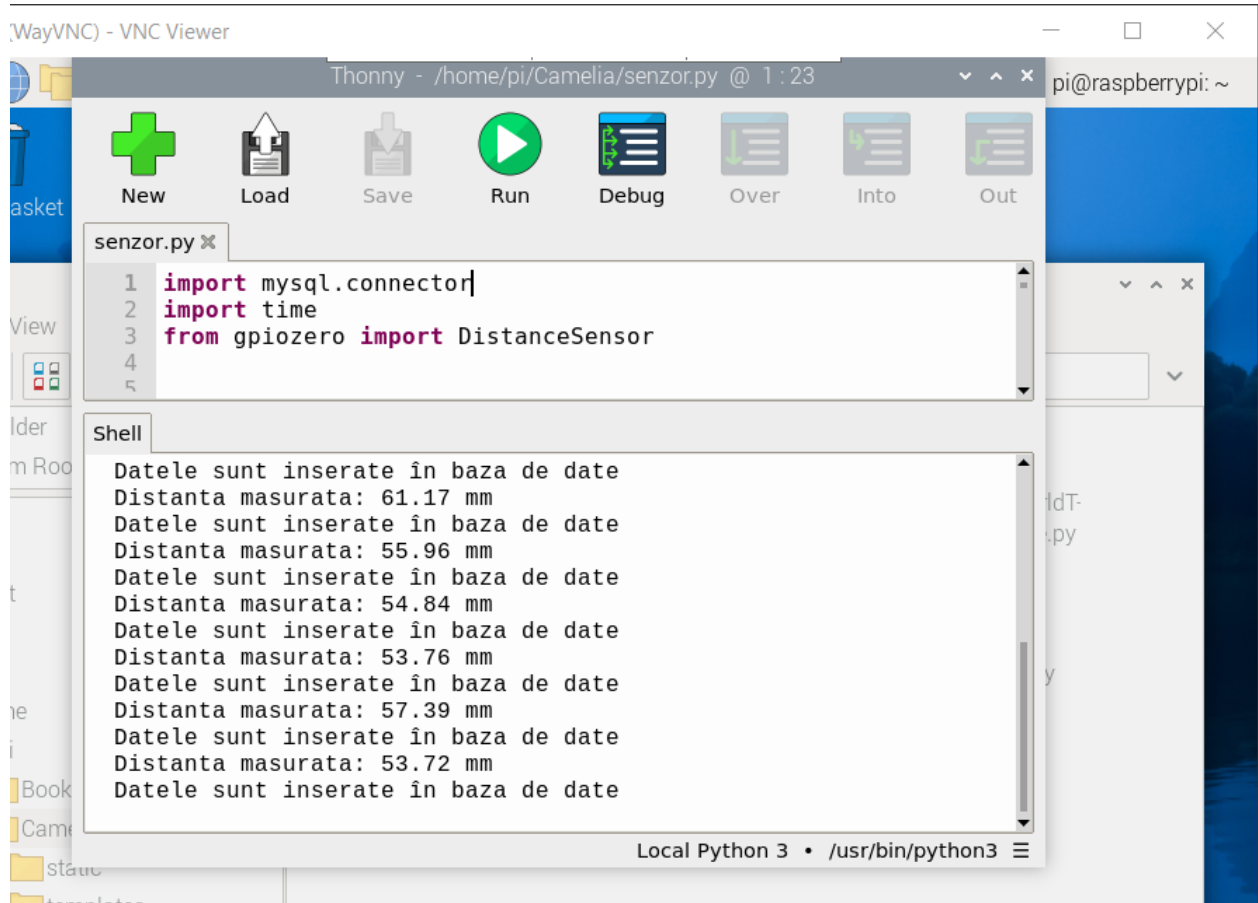
```
pico (WayVNC) - VNC Viewer
Thonny - /home/pi/..
pi@raspberrypi: ~

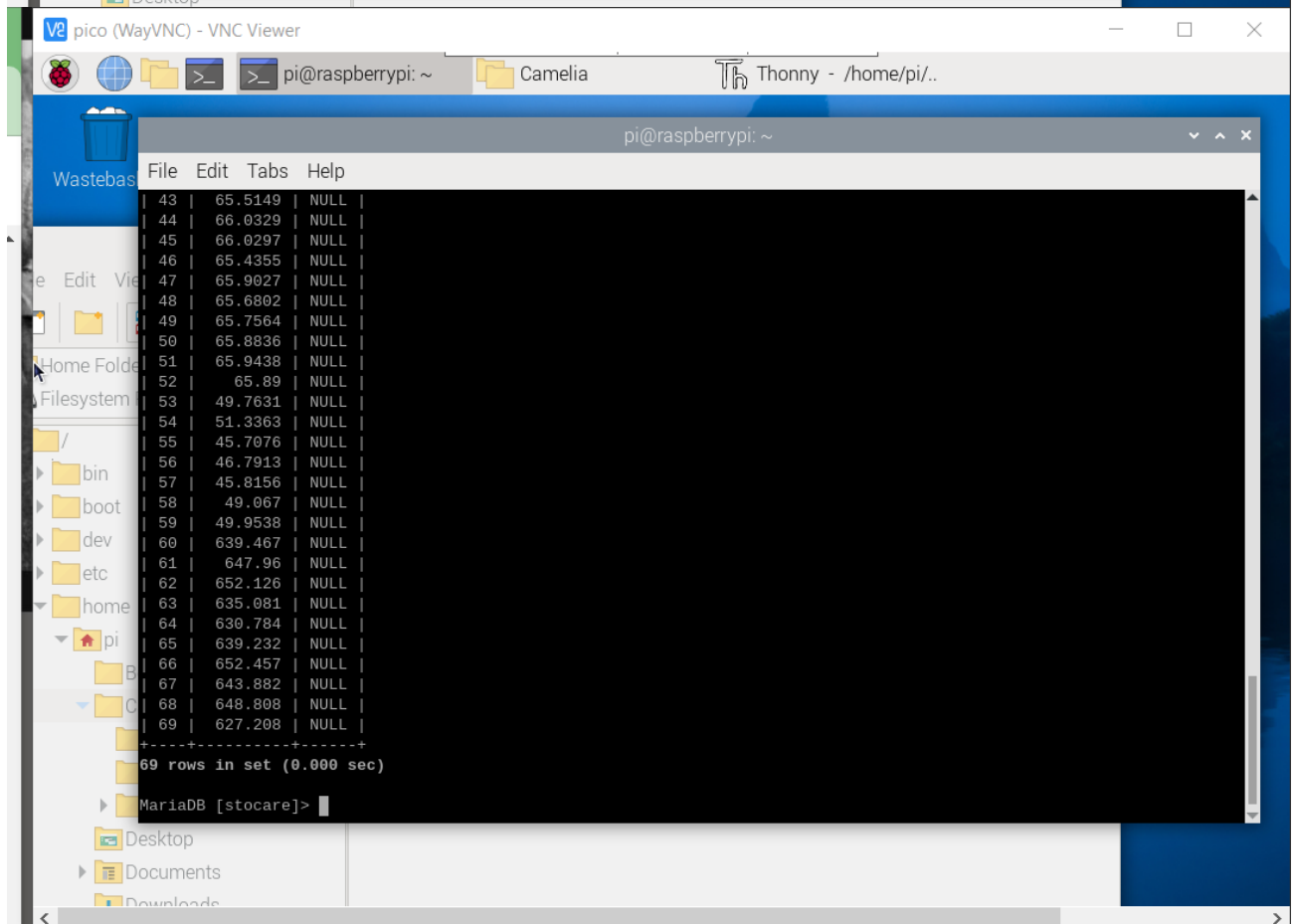
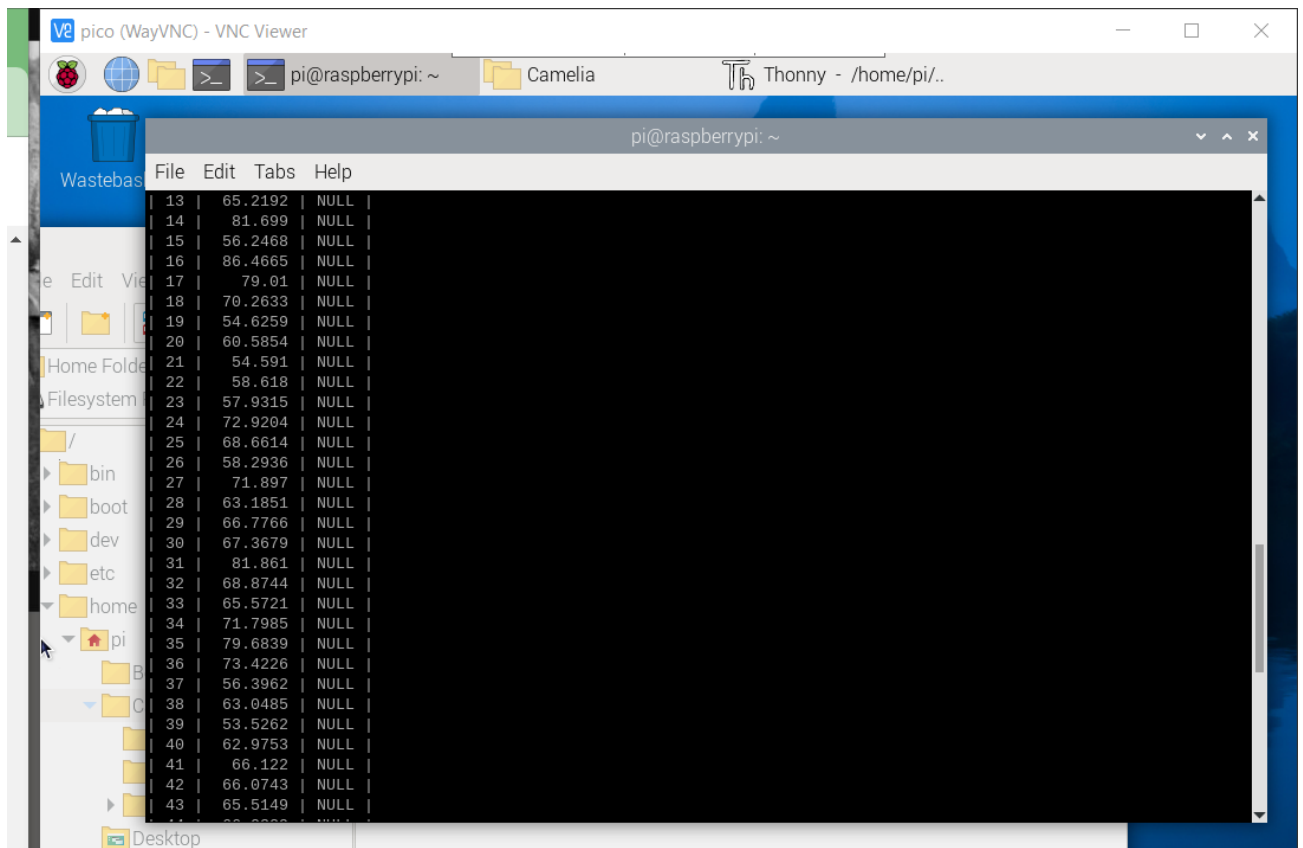
pi@raspberrypi: ~
File Edit Tabs Help
MariaDB [(none)]> USE stocare
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [stocare]> SHOW TABLES;
+-----+
| Tables_in_stocare |
+-----+
| Distanta           |
+-----+
1 row in set (0.000 sec)

MariaDB [stocare]> SHOW TABLE Distanta;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'Distanta' at line 1
MariaDB [stocare]> SELECT * FROM Distanta;
+-----+
| Nr | Distanta | Cod |
+-----+
| 1  | 55.28    | 1234 |
| 2  | 45.28    | 1934 |
+-----+
2 rows in set (0.000 sec)

MariaDB [stocare]>
```





Codul de mai jos este folosit pentru introducerea valorii citite de senzor in tabelul din baza de date:

```
import mysql.connector
import time
from gpiozero import DistanceSensor
```

```
ECHO_PIN = 23
```

```
TRIG_PIN = 24
```

```
sensor = DistanceSensor(echo=ECHO_PIN, trigger=TRIG_PIN)
```

```
try:
```

```
    con = mysql.connector.connect(
        host="localhost",
        user="root",
        password="stocaredate",
        database="stocare"
    )
```

```
    cursor = con.cursor()
```

```
    while True:
```

```
        distance = sensor.distance * 100
        print(f"Distanța măsurată: {distance:.2f} cm")
```

```
        cursor.execute("INSERT INTO Distanța (distanța) VALUES (%s)", (distance,))
        con.commit()
        print("Datele sunt inserate în baza de date")
```

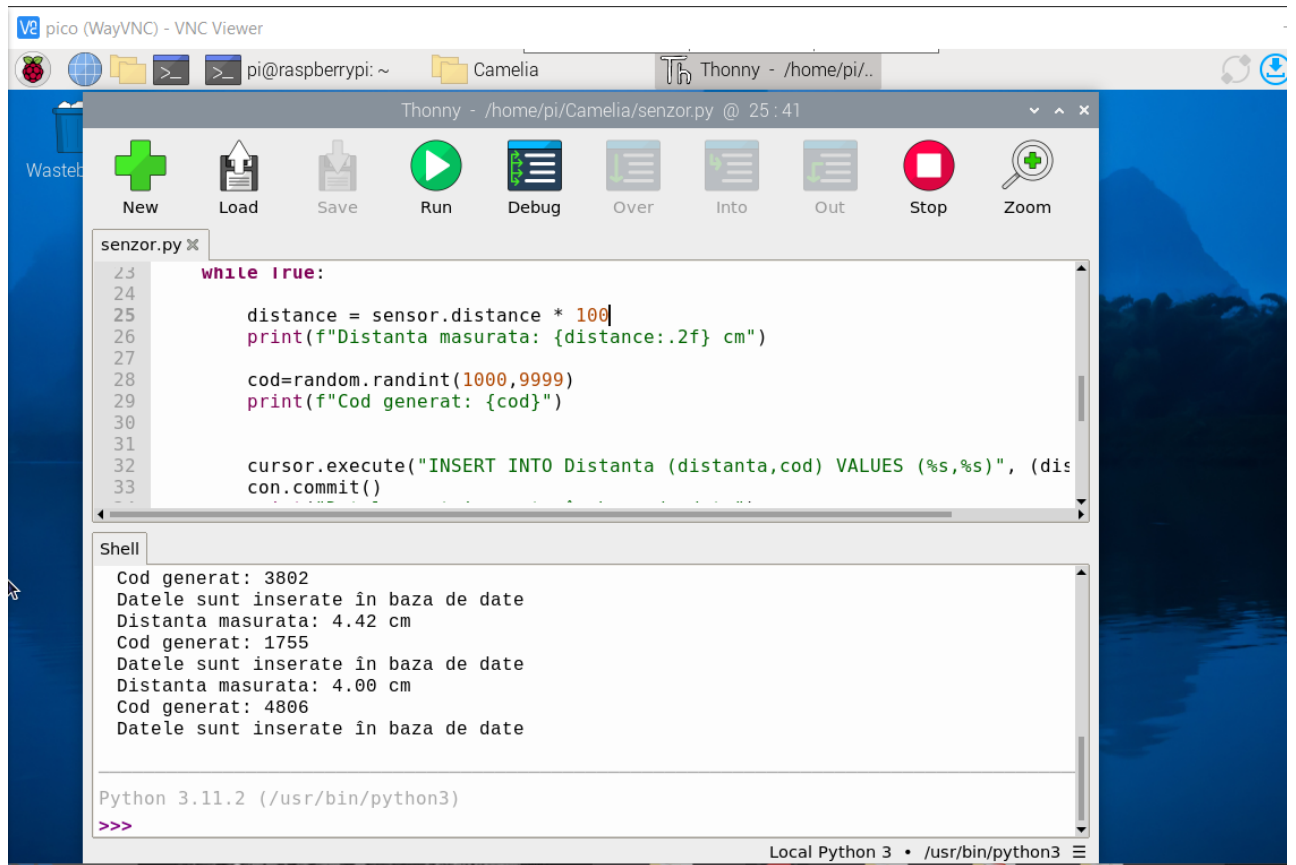
```
        time.sleep(1)
```

```
except mysql.connector.Error as err:
    print(f"Eroare de conexiune: {err}")
```

```
except KeyboardInterrupt:
    print("Program oprit manual")
```

```
finally:
```

```
    if con.is_connected():
        cursor.close()
        con.close()
    sensor.close()
```



Thonny - /home/pi/Camelia/senzor.py @ 25:41

```
senzor.py x
23 while True:
24
25     distance = sensor.distance * 100
26     print(f"Distanța măsurată: {distance:.2f} cm")
27
28     cod=random.randint(1000,9999)
29     print(f"Cod generat: {cod}")
30
31
32     cursor.execute("INSERT INTO Distanța (distanța,cod) VALUES (%s,%s)", (dis
33     con.commit()
```

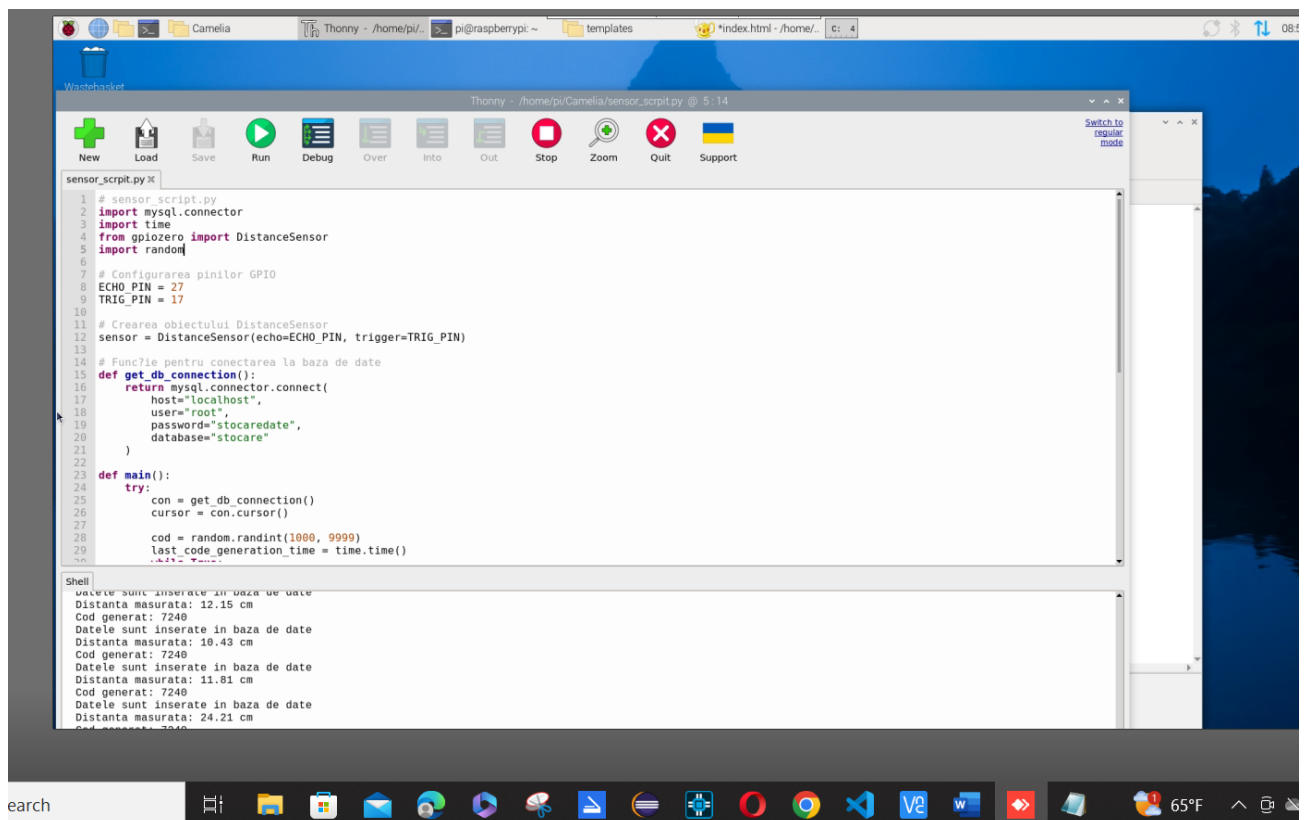
Shell

```
Cod generat: 3802
Datele sunt inserate în baza de date
Distanța măsurată: 4.42 cm
Cod generat: 1755
Datele sunt inserate în baza de date
Distanța măsurată: 4.00 cm
Cod generat: 4806
Datele sunt inserate în baza de date
```

Python 3.11.2 (/usr/bin/python3)

>>>

Local Python 3 • /usr/bin/python3



Thonny - /home/pi/Camelia/senzor_script.py @ 5:14

```
senzor_script.py x
1 # senzor_script.py
2 import mysql.connector
3 import time
4 from gpiozero import DistanceSensor
5 import random
6
7 # Configurarea pinilor GPIO
8 ECHO_PIN = 27
9 TRIG_PIN = 17
10
11 # Crearea obiectului DistanceSensor
12 sensor = DistanceSensor(echo=ECHO_PIN, trigger=TRIG_PIN)
13
14 # Funcție pentru conectarea la baza de date
15 def get_db_connection():
16     return mysql.connector.connect(
17         host="localhost",
18         user="root",
19         password="stocaredate",
20         database="stocare"
21     )
22
23 def main():
24     try:
25         con = get_db_connection()
26         cursor = con.cursor()
27
28         cod = random.randint(1000, 9999)
29         last_code_generation_time = time.time()
```

Shell

```
Datele sunt inserate în baza de date
Distanța măsurată: 12.15 cm
Cod generat: 7240
Datele sunt inserate în baza de date
Distanța măsurată: 10.43 cm
Cod generat: 7240
Datele sunt inserate în baza de date
Distanța măsurată: 11.81 cm
Cod generat: 7240
Datele sunt inserate în baza de date
Distanța măsurată: 24.21 cm
Cod generat: 7240
```

Implementarea script-ului pentru senzor:

```
import mysql.connector
```

```
import time
```

```
from gpiozero import DistanceSensor
```

```
import random
```

```
from rpi_lcd import LCD
```

```

# Configurarea pinilor GPIO
ECHO_PIN = 27
TRIG_PIN = 17
lcd=LCD()

# Crearea obiectului DistanceSensor
sensor = DistanceSensor(echo=ECHO_PIN, trigger=TRIG_PIN)

# Func?ie pentru conectarea la baza de date
def get_db_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="stocaredate",
        database="stocare"
    )

def main():
    try:
        lcd.text("Bine ati venit!",1)
        lcd.text("Introduceti codul:",2)
        time.sleep(3)
        con = get_db_connection()
        cursor = con.cursor()

        cod = random.randint(1000, 9999)
        last_code_generation_time = time.time()
        while True:
            distance = sensor.distance * 100
            print(f"Distan?a masurata: {distance:.2f} cm")
            print(f"Cod generat: {cod}")
            cursor.execute("INSERT INTO Masuratori (coloana1, coloana2) VALUES (%s, %s)",
                (distance, cod))
            con.commit()
            print("Datele sunt inserate in baza de date")

            current_time = time.time()
            if current_time - last_code_generation_time > 900:
                cod = random.randint(1000, 9999)
                last_code_generation_time = current_time
                time.sleep(5)
            else:
                time.sleep(1)

    except mysql.connector.Error as err:
        print(f"Eroare de conexiune: {err}")

    except KeyboardInterrupt:
        print("Program oprit manual")

    finally:
        if con.is_connected():
            cursor.close()

```

```
        con.close()
        sensor.close()

if __name__ == '__main__':
    main()
```

Implementarea paginii web:

```
<!DOCTYPE html>
<html lang="ro">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SISTEM DE SECURITATE</title>
    <link rel="stylesheet" href="styles.css">
    <style>
/* Resetare stiluri implicite */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: Arial, sans-serif;
    background-color: #87CEEB; /* Albastru deschis */
    color: #333;
    text-align: center;
}

.container {
    margin: 20px auto;
    padding: 20px;
    max-width: 600px;
    background-color: #fff;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1 {
    color: #0056b3; /* Albastru chis */
    margin-bottom: 20px;
}

h2 {
    margin-bottom: 10px;
}

p {
    margin: 10px 0;
}

input[type="text"] {
```

```
padding: 10px;
border: 1px solid #ccc;
border-radius: 5px;
width: calc(100% - 22px);
margin-bottom: 10px;
}
```

```
button {
padding: 10px 20px;
background-color: #0056b3; /* Albastru nchis */
color: white;
border: none;
border-radius: 5px;
cursor: pointer;
}
```

```
button:hover {
background-color: #003d7a; /* Albastru mai chis */
}
```

```
#alert {
margin-top: 10px;
color: red;
font-weight: bold;
}
```

```
.alert-active {
color: red;
}
```

</style>

<script>

```
function distanta() {
  fetch('/distanta')
    .then(response => response.json())
    .then(data => {
      document.getElementById('distance').innerText = data.distance.toFixed(1) + ' cm';
      document.getElementById('code').innerText = data.code || '---'; // Valoare implicita
      if (data.alert) {
        document.getElementById('alert').innerText = "Alerta activa!";
        document.getElementById('alert').classList.add('alert-active');
      } else {
        document.getElementById('alert').innerText = "";
        document.getElementById('alert').classList.remove('alert-active');
      }
    })
    .catch(error => {
      console.error('Eroare la obtinerea distantei:', error);
      // Nu afi?a mesaj de eroare pe pagina
      document.getElementById('distance').innerText = '0 cm'; // Valoare implicita
      document.getElementById('code').innerText = '---'; // Valoare implicita
    });
}
```



```

function verificare() {
    var code = document.getElementById('input_code').value;
    fetch('/verify_code', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ code: code })
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            alert("Cod corect!");
        } else {
            alert("Cod gresit!");
        }
    })
    .catch(error => {
        console.error('Eroare la verificarea codului:', error);
    });
}

setInterval(distan?a, 1000); // Actualizeaza distan?a la fiecare 1 secunda
</script>
</head>
<body>
    <div class="container">
        <h1>SISTEM DE SECURITATE</h1>
        <div class="content">
            <h2>Distan?a detectata:</h2>
            <p><span id="distance">Loading...</span></p>
            <p>Cod de acces: <span id="code"></span></p>
            <input type="text" id="input_code" placeholder="Introdu codul">
            <button onclick="verificare()">Verifica codul</button>
            <p id="alert"></p>
        </div>
    </div>
</body>
</html>

```

Partea de server:

```

import mysql.connector
from flask import Flask, render_template, request, jsonify

app = Flask(__name__, template_folder='templates')

# Func?ie pentru conectarea la baza de date
def get_db_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="stocaredate",

```

```

        database="stocare"
    )

@app.route('/')
def index():
    distance = get_distance_data()[0] if get_distance_data() else 0
    return render_template('index.html', distanta=distance)

@app.route('/distanta')
def distanta():
    data = get_distance_data()
    if data:
        response = {
            'distance': data[0],
            'code': data[1],
            'alert': False
        }
    else:
        response = {
            'distance': 0,
            'code': "",
            'alert': False
        }
    return jsonify(response)

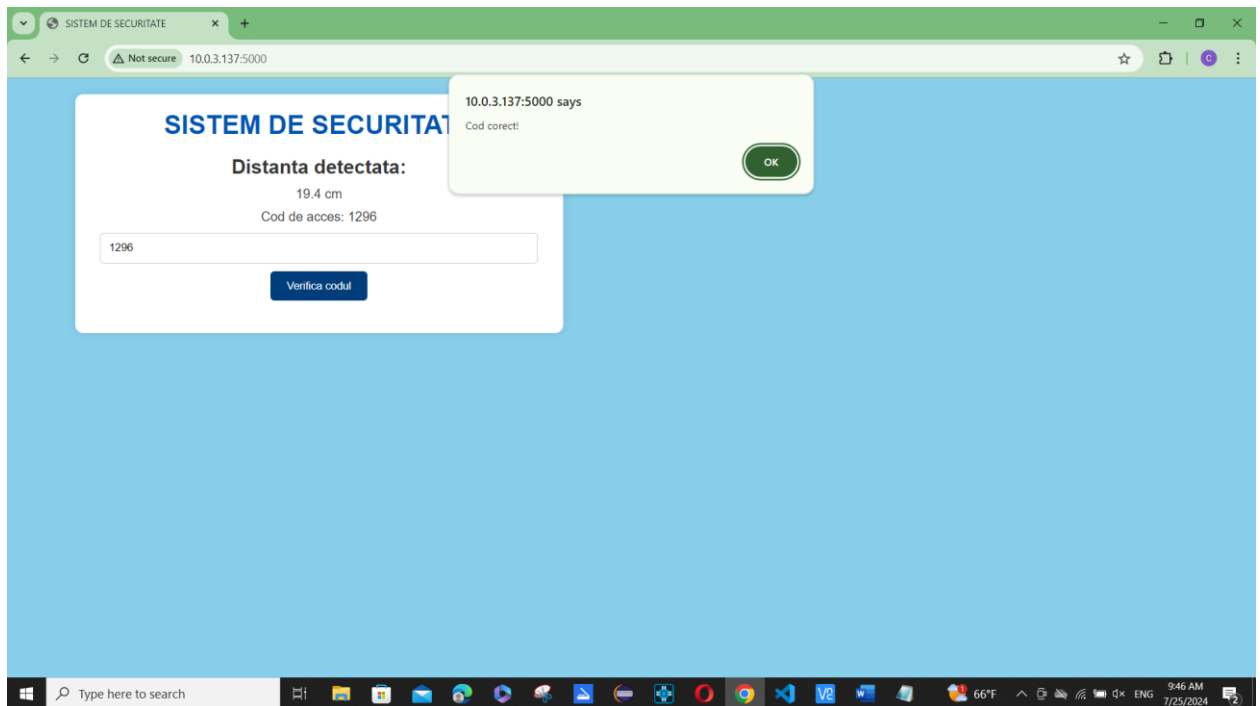
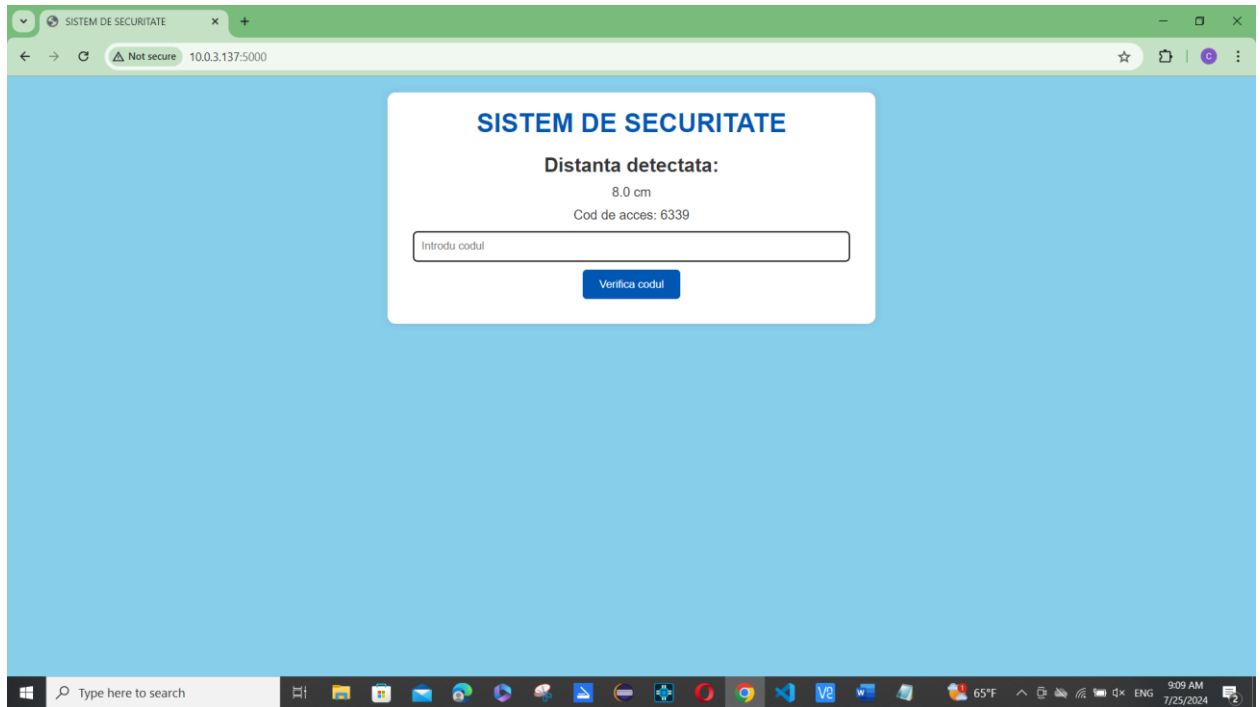
@app.route('/verify_code', methods=['POST'])
def verify_code():
    code = request.json['code']
    connection = get_db_connection()
    cursor = connection.cursor()
    cursor.execute("SELECT 1 FROM Masuratori WHERE coloana2 = %s ORDER BY Nr DESC
LIMIT 1", (code,))
    result = cursor.fetchone()
    cursor.close()
    connection.close()
    if result:
        return jsonify({'success': True})
        lcd.text("ACCES PERMIS",2)
    else:
        return jsonify({'success': False})

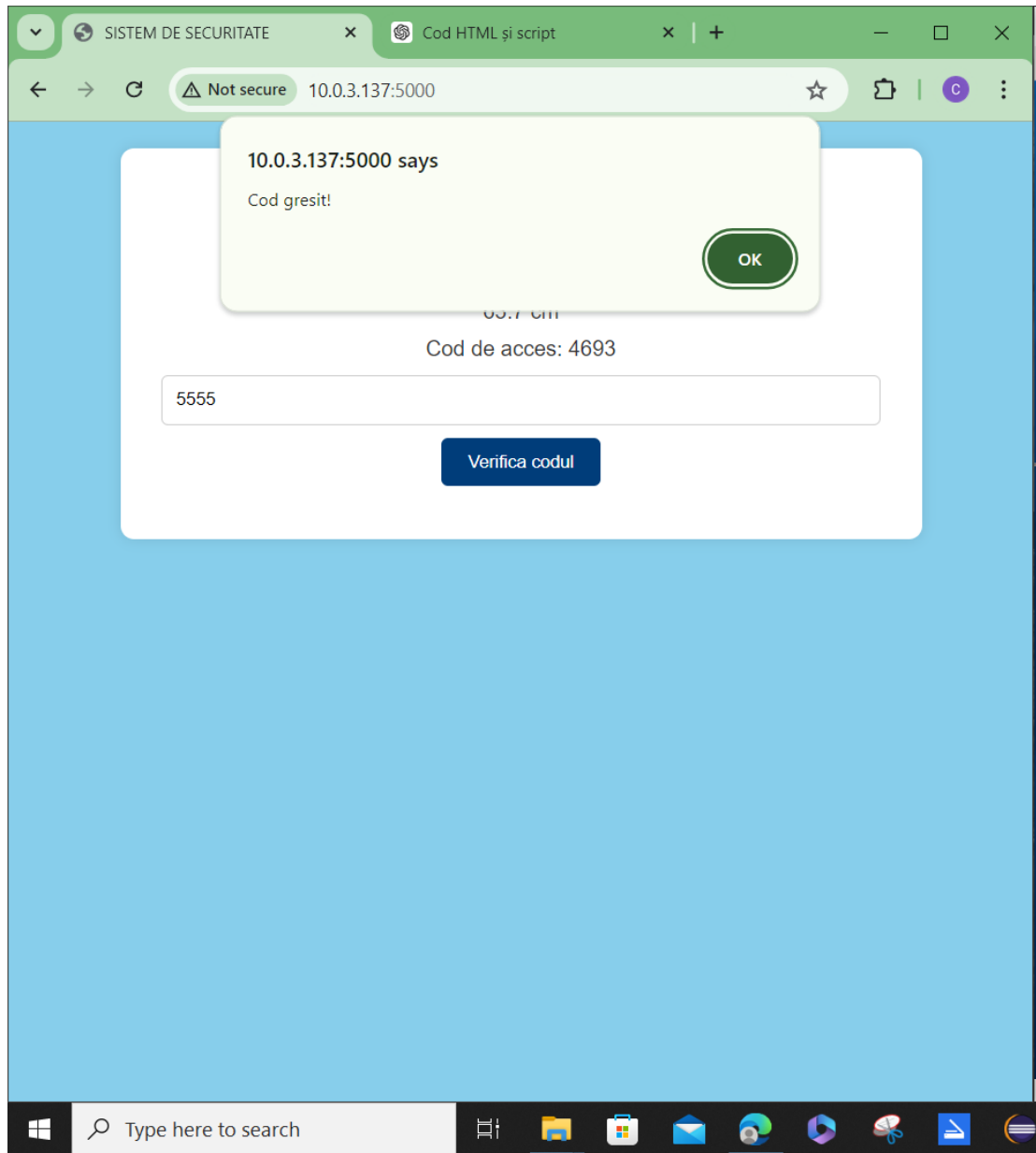
def get_distance_data():
    connection = get_db_connection()
    cursor = connection.cursor()
    cursor.execute("SELECT coloana1, coloana2 FROM Masuratori ORDER BY Nr DESC LIMIT
1")
    data = cursor.fetchone()
    cursor.close()
    connection.close()
    return data

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')

```

INTERFATA PAGINII WEB:





Am adaugat optiunea de a apasa tasta enter pentru verificarea codului:

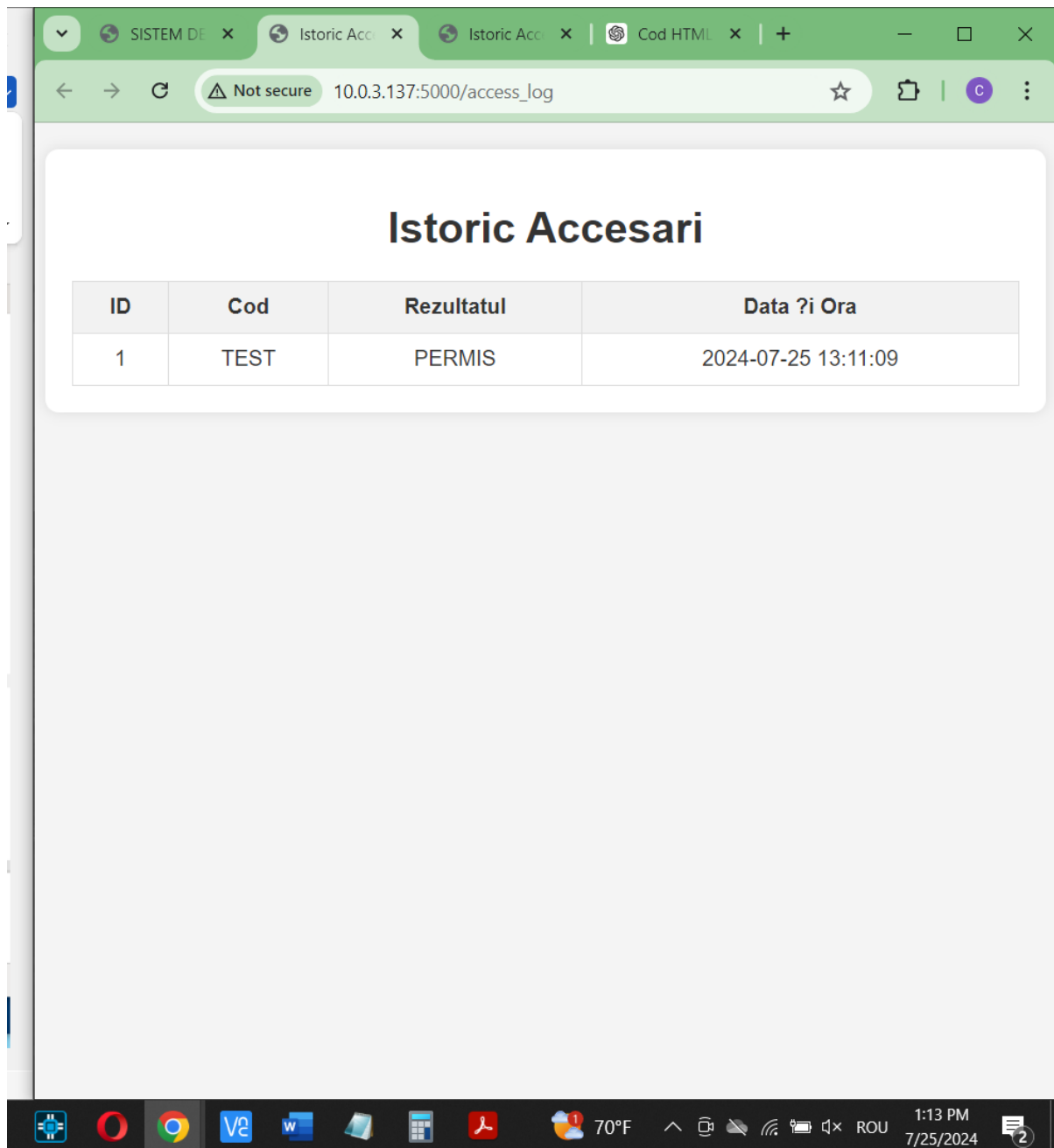
```
        console.error('Eroare la verificarea codului:', error);
    });
}

document.addEventListener('DOMContentLoaded', function(){
    document.getElementById('input_code').addEventListener('keydown', function(event){

        if (event.key=='Enter'){
            event.preventDefault();
            verificare();
        }
    });
});

setInterval(distanța, 1000); // Actualizeaza distanța la fiecare 1 secunda
</script>
</head>
<body>
```

In cele ce urmeaza, am adaugat o alta pagina web pentru a vedea istoricul accesarilor:



```
<!DOCTYPE html>
<html lang="ro">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>SISTEM DE SECURITATE</title>
  <link rel="stylesheet" href="styles.css">
  <style>
  {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
  }
  </style>
</head>
<body>
  <div>
    <h1>Istoric Accesari</h1>
    <table>
      <thead>
        <tr>
          <th>ID</th>
          <th>Cod</th>
          <th>Rezultatul</th>
          <th>Data ?i Ora</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>1</td>
          <td>TEST</td>
          <td>PERMIS</td>
          <td>2024-07-25 13:11:09</td>
        </tr>
      </tbody>
    </table>
  </div>
</body>
</html>
```

```
body {
  font-family: Arial, sans-serif;
  background-color: #87CEEB; /* Albastru deschis */
  color: #333;
  text-align: center;
}

.container {
  margin: 20px auto;
  padding: 20px;
  max-width: 600px;
  background-color: #fff;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1 {
  color: #0056b3; /* Albastru chis */
  margin-bottom: 20px;
}

h2 {
  margin-bottom: 10px;
}

p {
  margin: 10px 0;
}

input[type="text"] {
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  width: calc(100% - 22px);
  margin-bottom: 10px;
}

button {
  padding: 10px 20px;
  background-color: #0056b3; /* Albastru inchis */
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

button:hover {
  background-color: #003d7a; /* Albastru mai inchis */
}

#alert {
  margin-top: 20px;
  color: red;
  font-weight: bold;
}
```

```

}

.alert-active {
  color: red;
}

</style>
<script>
function distanta() {
  fetch('/distanta')
    .then(response => response.json())
    .then(data => {
      document.getElementById('distance').innerText = data.distance.toFixed(1) + ' cm';
      document.getElementById('code').innerText = data.code || '---'; // Valoare implicita
      if (data.alert) {
        document.getElementById('alert').innerText = "Alerta activa!";
        document.getElementById('alert').classList.add('alert-active');
      } else {
        document.getElementById('alert').innerText = "";
        document.getElementById('alert').classList.remove('alert-active');
      }
    })
    .catch(error => {
      console.error('Eroare la obtinerea distantei:', error);
      document.getElementById('distance').innerText = '0 cm'; // Valoare implicita
      document.getElementById('code').innerText = '---'; // Valoare implicita
    });
}

function verificare() {
  var code = document.getElementById('input_code').value;
  fetch('/verify_code', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({ code: code })
  })
    .then(response => response.json())
    .then(data => {
      if (data.success) {
        alert("Cod corect!");
      } else {
        alert("Cod gresit!");
      }
    })
    .catch(error => {
      console.error('Eroare la verificarea codului:', error);
    });
}

function openLog() {
  window.open('/access_log', '_blank');
}

```

```

        setInterval(distan?a, 1000); // Actualizeaza distan?a la fiecare 1 secunda
    </script>
</head>
<body>
    <div class="container">
        <h1>SISTEM DE SECURITATE</h1>
        <div class="content">
            <h2>Distan?a detectata:</h2>
            <p><span id="distance">Loading...</span></p>
            <p>Cod de acces: <span id="code"></span></p>
            <input type="text" id="input_code" placeholder="Introdu codul">
            <button onclick="verificare()">Verifica codul</button>
            <button onclick="openLog()">Vezi Istoric Accesari</button>
            <p id="alert"></p>
        </div>
    </div>
</body>
</html>

```

```

# sensor_script.py
import mysql.connector
import time
from gpiozero import DistanceSensor
import random
from rpi_lcd import LCD

# Configurarea pinilor GPIO
ECHO_PIN = 27
TRIG_PIN = 17

lcd=LCD()

# Crearea obiectului DistanceSensor
sensor = DistanceSensor(echo=ECHO_PIN, trigger=TRIG_PIN)

# Func?ie pentru conectarea la baza de date
def get_db_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="stocaredate",
        database="stocare"
    )

def main():
    try:
        lcd.text("Bine ati venit!",1)
        lcd.text("Introduceti codul:",2)

```



```

time.sleep(3)
lcd.clear()
con = get_db_connection()
cursor = con.cursor()

cod = random.randint(1000, 9999)
last_code_generation_time = time.time()
while True:
    distance = sensor.distance * 100
    print(f"Distanța măsurată: {distance:.2f} cm")
    print(f"Cod generat: {cod}")
    cursor.execute("INSERT INTO Masuratori (coloana1, coloana2) VALUES (%s, %s)",
(distance, cod))
    con.commit()
    print("Datele sunt inserate în baza de date")

    current_time = time.time()
    if current_time - last_code_generation_time > 900:
        cod = random.randint(1000, 9999)
        last_code_generation_time = current_time
        time.sleep(5)
    else:
        time.sleep(1)

except mysql.connector.Error as err:
    print(f"Eroare de conexiune: {err}")

except KeyboardInterrupt:
    print("Program oprit manual")

finally:
    if con.is_connected():
        cursor.close()
        con.close()
        sensor.close()

if __name__ == '__main__':
    main()

```

server.py

```

import mysql.connector
from flask import Flask, render_template, request, jsonify
from rpi_lcd import LCD

app = Flask(__name__, template_folder='templates')
lcd=LCD()

# Funcție pentru conectarea la baza de date
def get_db_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",

```

```
    password="stocaredate",
    database="stocare"
)
```

```
@app.route('/')
def index():
    distance = get_distance_data()[0] if get_distance_data() else 0
    return render_template('index.html', distanta=distance)
```

```
@app.route('/distanta')
def distanta():
    data = get_distance_data()
    if data:
        response = {
            'distance': data[0],
            'code': data[1],
            'alert': False
        }
    else:
        response = {
            'distance': 0,
            'code': "",
            'alert': False
        }
    return jsonify(response)
```

```
@app.route('/verify_code', methods=['POST'])
def verify_code():
    lcd.clear()
    code = request.json['code']
    connection = get_db_connection()
    cursor = connection.cursor()
    cursor.execute("SELECT 1 FROM Masuratori WHERE coloana2 = %s ORDER BY Nr DESC
LIMIT 1", (code,))
    result = cursor.fetchone()
    cursor.close()
    connection.close()
    if result:
        lcd.text("Acces PERMIS",2)
        return jsonify({'success': True})
    else:
        lcd.text("Acces BLOCAT",2)
        return jsonify({'success': False})
    cursor.execute("INSERT INTO AccessLog (code, result) VALUES (%s,%s)",(code, 'Permis' if
success else 'Blocat'))
    connection.commit()
    cursor.close()
    connection.close()
    return jsonify({'success': success})
```

```
@app.route('/access_log')
```

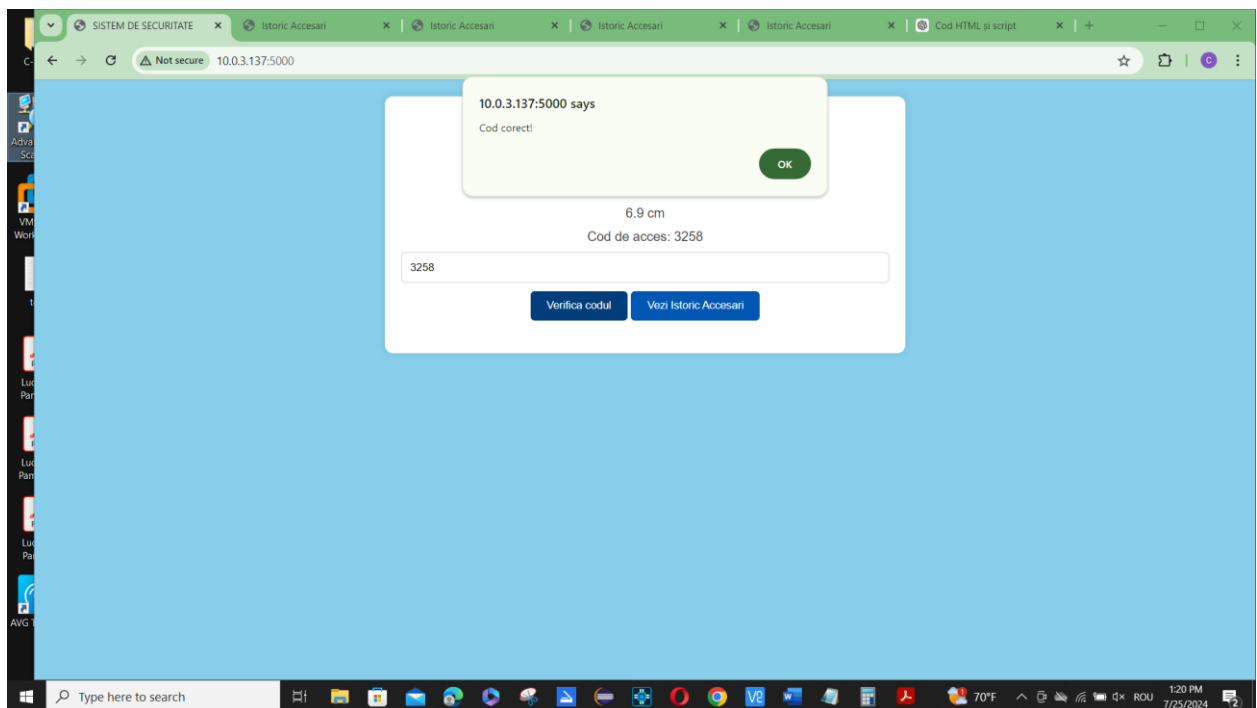
```

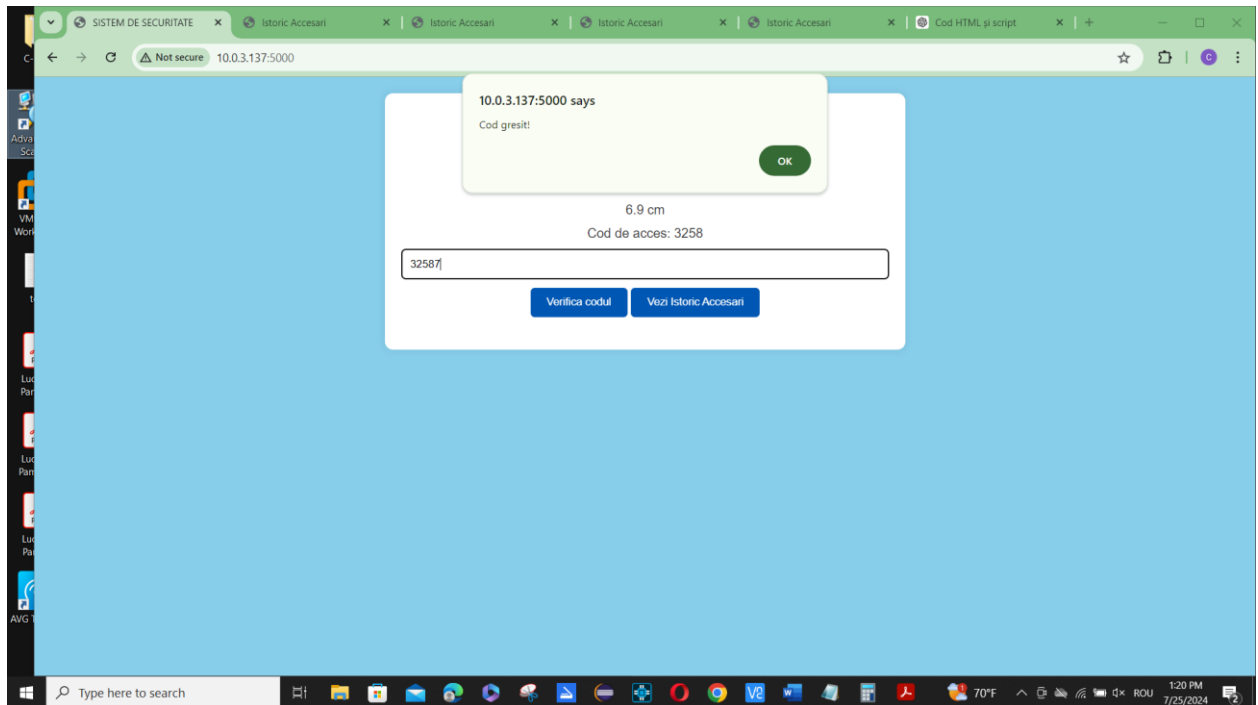
def access_log():
    connection = get_db_connection()
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM AccessLog ORDER BY timestamp DESC")
    logs = cursor.fetchall()
    cursor.close()
    connection.close()
    return render_template('access_log.html', logs=logs)

def get_distance_data():
    connection = get_db_connection()
    cursor = connection.cursor()
    cursor.execute("SELECT coloana1, coloana2 FROM Masuratori ORDER BY Nr DESC LIMIT 1")
    data = cursor.fetchone()
    cursor.close()
    connection.close()
    return data

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')

```





In istoric raman ultimele 10 accesari, ordonate in functie de ora la care a fost verificat codul:

ID	Cod	Rezultatul	Data ?i Ora
8	32587	Blocat	2024-07-25 13:20:51
7	3258	Permis	2024-07-25 13:20:27
6	3258	Permis	2024-07-25 13:20:10
4	3258	Permis	2024-07-25 13:20:09
5	3258	Permis	2024-07-25 13:20:09
3	9999	Blocat	2024-07-25 13:20:02
2	9999	Blocat	2024-07-25 13:20:01
1	TEST	PERMIS	2024-07-25 13:11:09

HTML FINAL:

```
<!DOCTYPE html>
```

```
<html lang="ro">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Istoric Accesari</title>
```

```
  <link rel="stylesheet" href="styles.css">
```

```
<style>
```

```
  /* Stiluri simple pentru afi?area istoricului */
```

```
  body {
```

```

        font-family: Arial, sans-serif;
        background-color: #f4f4f4;
        color: #333;
        text-align: center;
    }
    .container {
        margin: 20px auto;
        padding: 20px;
        max-width: 800px;
        background-color: #fff;
        border-radius: 10px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
    table {
        width: 100%;
        border-collapse: collapse;
    }
    th, td {
        padding: 10px;
        border: 1px solid #ddd;
    }
    th {
        background-color: #f2f2f2;
    }
</style>
</head>
<body>
    <div class="container">
        <h1>Istoric Accesari</h1>
        <table>
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Cod</th>
                    <th>Data si Ora</th>

                </tr>
            </thead>
            <tbody>
                {% for log in logs %}
                <tr>
                    <td>{{ log[0] }}</td>
                    <td>{{ log[1] }}</td>
                    <td>{{ log[2] }}</td>

                </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>
</body>
</html>

```
