**Proiect Arduino  
Proiectarea Microsistemelor Digitale  
  
  
Sistem anti-efractie**

**Prof coord.**  
S.l.dr.ing. GAL Norbert  
S.l.dr.ing. CRISAN-VIDA Mihaela Marcella

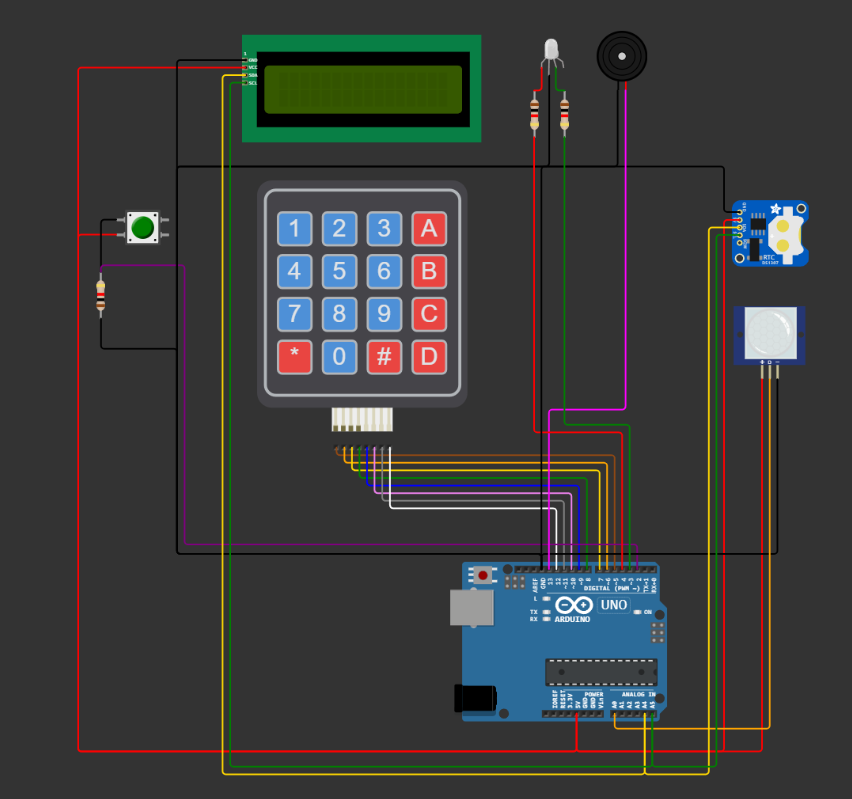
**Student,**  
DUMITRASCU Stefan Tudor

Timisoara, 2023

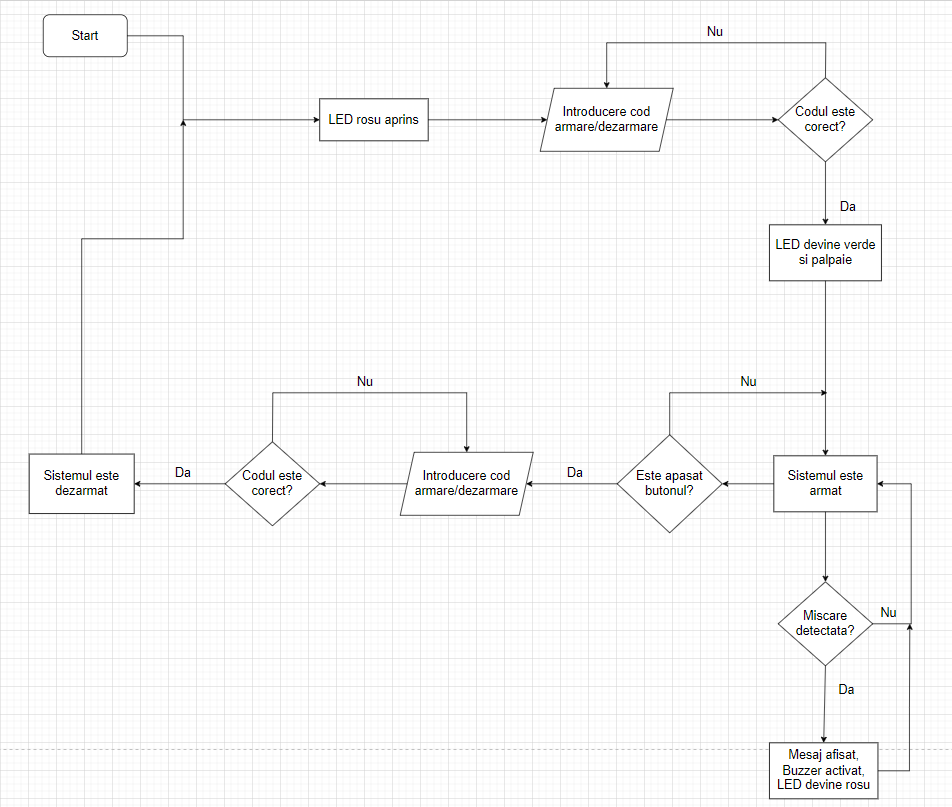
**Cerintele initiale:**  
La apasarea butonului se porneste/opreste sistemul si se aprinde LED-ul rosu/verde.

Daca sistemul de anti efractie este activat si senzorii detecteaza o efractie aceasta efractie este semnalizata pe ecran cu data si ora efractiei. Daca sistemul este armat si nu este nici o efractie LED-ul verde palpaie.

Folosind tastatura se introduce un cod de armare/dezarmare al sistemului. Modulul RTC este folosit pentru a marca timpii la care s-au produs evenimentele.

  
  
**Schema hardware a sistemului**

Sistemul semnaleaza posibile efractii folosindu-se de senzorul de miscare. Starea curenta a sistemului este semnalata de LED-ul RGB, armarea/dezarmarea facandu-se cu ajutorul unui cod, introdus pe tastatura cu 16 butoane. Sistemul afiseaza mesaje pe ecranul LCD in concordanta cu eventuale efractii, iar aceste evenimente sunt marcate cu ajutorul modulului RTC.



**Schema logica a programului de comanda**

**Codul sursa pentru sistemul anti efractie:**  
  
// Includem bibliotecile pentru LCD, RTC, si tastatura

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <RTClib.h>

#include <Keypad.h>

// Definim numarul de randuri si coloane pentru tastatura

const byte ROWS = 4;

const byte COLS = 4;

// Definim formatul tastaturii

char keys[ROWS][COLS] = {

{'1','2','3','A'},

{'4','5','6','B'},

{'7','8','9','C'},

{'\*','0','#','D'}

};

// Conectam pinii tastaturii la Arduino

byte rowPins[ROWS] = { 5, 6, 7, 8 }; // Pinii pentru randuri

byte colPins[COLS] = { 9, 10, 11, 12 }; // Pinii pentru coloane

// Initializam tastatura

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

String enterPass = ""; // Variabila pentru stocarea parolei introduse

// Initializam ecranul LCD si RTC

LiquidCrystal\_I2C lcd(0x27, 16, 2);

RTC\_DS1307 rtc;

// Definim constante pentru debouncing si intarziere

#define DEBOUNCE\_DELAY 200

#define HOLD\_DELAY 1000

#define ButtonPin 2

#define LedGPin 3

#define LedRPin 4

#define PirSensorPin A0

#define BuzzerPin 13

// Variabile pentru gestionarea starii sistemului

bool systemArmed = false; // Starea de armare a sistemului

bool lastButtonState = HIGH; // Ultima stare a butonului pentru debouncing

bool buttonState = false; // Starea actuala a butonului

unsigned long lastButtonChangeTime = 0; // Timpul ultimei schimbari a butonului

bool passEntryStarted = false; // Indica daca introducerea parolei a inceput

bool passRequired = false; // Indica daca este necesara introducerea parolei

int pirSensorValue = 0; // Valoarea citita de la senzorul PIR

// Starile sistemului

enum SystemState { DISARMED, ARMED, INTRUSION };

SystemState currentState = DISARMED; // Starea initiala a sistemului

void setup() {

// Configurarea pinilor pentru buton, LED-uri, senzor PIR si buzzer

pinMode(ButtonPin, INPUT);

pinMode(LedGPin, OUTPUT);

pinMode(LedRPin, OUTPUT);

pinMode(PirSensorPin, INPUT);

pinMode(BuzzerPin, OUTPUT);

// Initializam comunicarea seriala la 9600 bps

Serial.begin(9600);

// Initializam si configuram ecranul LCD

lcd.init(); // Initializeaza LCD-ul

lcd.backlight(); // Activeaza iluminarea din spate a LCD-ului

lcd.clear(); // Curata ecranul LCD

lcd.print("System Ready"); // Afiseaza un mesaj de start

delay(1000); // Asteapta 1 secunda pentru a citi mesajul

// Verificam daca modulul RTC este conectat si funcționeaza corect

if (!rtc.begin()) {

lcd.print("RTC error"); // Afiseaza un mesaj de eroare daca RTC-ul nu este conectat

while (1); // Intra in bucla infinita daca exista o eroare RTC

}

if (!rtc.isrunning()) {

lcd.print("RTC is NOT running!"); // Afiseaza un mesaj daca RTC-ul nu functioneaza

while (1); // Intra in bucla infinita daca RTC-ul nu functioneaza

}

// Actualizam starea sistemului la initializare

updateSystemState();

}

void loop() {

// Citim starea butonului

bool reading = digitalRead(ButtonPin);

// Debouncing pentru buton

if (millis() - lastButtonChangeTime > HOLD\_DELAY) {

if (reading != lastButtonState) {

lastButtonState = reading;

lastButtonChangeTime = millis();

// Verificam daca butonul a fost apasat

if (reading == LOW && !buttonState) {

buttonState = true;

// Solicitam introducerea parolei daca nu este deja necesara

if (!passRequired) {

passEntryStarted = true;

passRequired = true;

// Afisam promptul pentru parola pe LCD

lcd.setCursor(0, 0);

lcd.print("Enter Password:");

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.setCursor(0, 1);

enterPass = "";

}

}

else if (reading == HIGH) {

buttonState = false; // Resetam starea butonului

}

}

}

// Verificam senzorii

checkSensors();

// Procesam introducerea parolei

if (passEntryStarted) {

char key = keypad.getKey();

// Citim fiecare caracter introdus

if (key) {

enterPass += key;

// Afisam caracterul introdus pe LCD

lcd.setCursor(enterPass.length() - 1, 1);

lcd.print(key);

Serial.print(key);

// Verificam lungimea parolei introduse

if (enterPass.length() == 4) {

Serial.println();

// Verificam daca parola introdusa este corecta

if (enterPass == "1234") {

// Schimbam starea sistemului si actualizam afisajul

passEntryStarted = false;

passRequired = false;

systemArmed = !systemArmed;

updateSystemState();

}

else {

// Parola incorecta, afisam mesaj de eroare

lcd.clear();

lcd.print("Access Denied");

Serial.println("Access Denied");

delay(1000);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Enter Password:");

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.setCursor(0, 1);

enterPass = "";

}

}

}

}

}

void updateSystemState() {

lcd.clear();

if (systemArmed) {

lcd.print("System Armed");

currentState = ARMED;

digitalWrite(LedRPin, LOW); // Stinge LED-ul roșu

digitalWrite(LedGPin, HIGH); // Aprinde LED-ul verde când sistemul este armat

Serial.println("System Armed");

blinkGLed(); // Pâlpâirea LED-ului verde

}

else {

lcd.print("System Disarmed");

currentState = DISARMED;

digitalWrite(LedGPin, LOW); // Stinge LED-ul verde

digitalWrite(LedRPin, HIGH); // Aprinde LED-ul roșu când sistemul este dezarmat

Serial.println("System Disarmed");

}

}

// Functia pentru formatarea timpului

String formatTime(int timeValue) {

// Adaugam un zero in fata numerelor mai mici de 10

return timeValue < 10 ? "0" + String(timeValue) : String(timeValue);

}

// Functia pentru formatarea datei si timpului

String formatDateTime(DateTime now) {

// Formatam si returnam data si ora in formatul "DD/MM/YYYY HH:MM:SS"

return String(now.day()) + "/" + String(now.month()) + "/" + String(now.year()) + " " +

formatTime(now.hour()) + ":" + formatTime(now.minute()) + ":" + formatTime(now.second());

}

// Functia pentru inregistrarea evenimentelor

void logEvent(String event) {

DateTime now = rtc.now(); // Obtine ora si data curenta de la RTC

// Afisam evenimentul si ora la care a avut loc pe ecranul LCD

lcd.clear();

lcd.print(event + " at");

lcd.setCursor(0, 1);

lcd.print(formatTime(now.hour()) + ":" + formatTime(now.minute()) + ":" + formatTime(now.second()));

// Afisam aceleasi informatii pe serial monitor

Serial.println(event + ": " + formatDateTime(now));

delay(2000); // Afiseaza mesajul pe ecran timp de 2 secunde

}

// Functia pentru a face LED-ul verde sa palpaie

void blinkGLed() {

// Repetam palpairea de 5 ori

for (int i = 0; i < 5; i++) {

digitalWrite(LedGPin, HIGH); // Aprinde LED-ul verde

delay(500); // Mentine LED-ul aprins timp de 500 ms

digitalWrite(LedGPin, LOW); // Stinge LED-ul verde

delay(500); // Mentine LED-ul stins timp de 500 ms

}

// Dupa finalizarea palpairii, LED-ul verde este lasat aprins

digitalWrite(LedGPin, HIGH);

}

// Functia pentru verificarea senzorilor

void checkSensors() {

// Citim valoarea de la senzorul PIR

pirSensorValue = analogRead(PirSensorPin);

// Verificam daca sistemul este armat si daca senzorul PIR detecteaza miscare

if (systemArmed && pirSensorValue >= 100) {

lcd.clear();

lcd.print("Intrusion detected!");

currentState = INTRUSION; // Schimba starea sistemului la detectarea unei intruziuni

digitalWrite(LedGPin, LOW); // Stinge LED-ul verde

digitalWrite(LedRPin, HIGH); // Aprinde LED-ul rosu

tone(BuzzerPin, 92, 100); // Activeaza buzzerul pentru a emite un sunet

logEvent("Intrusion"); // Inregistreaza evenimentul de intruziune

delay(500); // Mentine afisajul pe ecran pentru 500 ms

}

else {

noTone(BuzzerPin); // Opreste buzzerul

if (currentState != ARMED) {

// Restabilim starea LED-urilor daca sistemul nu este in stare de alerta

digitalWrite(LedGPin, LOW);

digitalWrite(LedRPin, HIGH);

}

}

}

**Dificultati intampinate:**  
  
Am intampinat dificultati in legatura cu codul pentru LED si starile sistemului.  
Nu am reusit sa gasesc o aplicatie pentru schema electrica a sistemului care sa aiba toate componentele necesare.  
  
**Demonstrare Video (Link Google Drive):**[Demonstare Proiect PMD.mkv](https://drive.google.com/file/d/1mqf7_2mlx4E2Yr4rsKwfLVDTa-DA7YoT/view?usp=sharing)