**Proiect Arduino**

**Programare orientate pe obiecte**

**Sistem antiefracție**

**Studenți:**

Camelia BURLACU

Iulia BUTNARU

Cristina CAPĂTA

Raluca CHETREANU

**An II, grupa 1.2**

Timișoara, 2019

Cuprins

1. Cerințele inițiale
2. Schema electrică a sistemului
3. Schema logică
4. Schema UML
5. Funcţiile sistemului
6. Codul sursa a programului si rezultatele testării
7. Concluzii

**Cerințele inițiale**

* **Componente:**

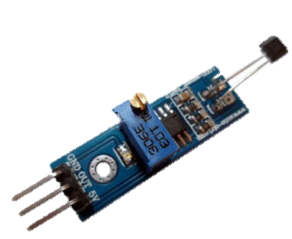
1. Arduino Unor R3

Arduino UNO  este o platformă de procesare open-source, bazată pe software și hardware flexibil și simplu de folosit. Acesta constă într-o platformă de mici dimensiuni (6.8 cm / 5.3 cm – în cea mai des întâlnită variantă), construită în jurul unui procesor de semnal și este capabilă de a prelua date din mediul înconjurător printr-o serie de senzori și de a efectua acțiuni asupra mediului prin intermediul luminilor, motoarelor, servomotoare și a altor tipuri de dispozitive mecanice. Procesorul este capabil să ruleze cod scris într-un limbaj de programare care este foarte similar cu limbajul C++.

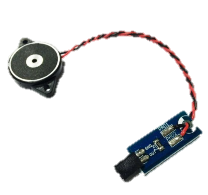
2. LED roșu si verde

LED-urile sunt dispozitive semiconductoare ce emit lumină atunci când sunt parcurse de curent.

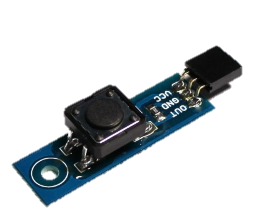
LED-urile nu permit trecerea curentului dacă sunt alimentate invers. Alimentate corect, acestea se deschid (permit curentului să treacă și luminează, intensitatea luminii fiind proporțională cu cea a curentului) după ce tensiunea la borne a depășit o anumită valoare de prag. Pragul de deschidere depinde de culoarea LED-ului, crescând de la roșu ( aproximativ 2.1 – 2.4 volți) spre albastru / alb ( aproximativ 3.2  – 3.8 volți).

3. Senzor magnetic Hall

Modulul cu senzor Hall poate fi utilizat pentru a detecta numărul de rotații pe minut la motoare,pentru a detecta dacă o ușă este sau nu închisă sau în altele proiecte.

4. Senzor de vibrație pentru detectarea mișcării

Senzorul de vibrații brick este o componentă care sesizează vibrațiile mecanice. Spre exemplu, este capabil să detecteze vibrații de genul unei bătăi în ușă.



5. Ecran LCD

6. Buton Brick

* **Cerințe:**

Dacă sistemul de antiefracție este activat și senzorii detectează o efracție, această efracție este semnalizată pe ecran, iar LED-ul roșu se aprinde.

Dacă sistemul este dezarmat și senzorii detectează ceva, pe ecran se afișează un mesaj care conține senzorul activat pentru posibila breșă de securitate și se aprinde ledul verde.

Dacă sistemul este armat și nu este nici o efractie LED-ul verde pâlpâie.

**Funcţiile sistemului**

* **LED RGB brick:**

Este folosit pentru afișarea stării în care se află sistemul anti efracție. Dacă sistemul de antiefractie este activat și senzorii detectează o efracție această efracție LED-ul rgb se colorează în roșu. Dacă sistemul este dezarmat si senzorii detectează ceva RGB luminează in albastru. Dacă sistemul este armat si nu este nici o efractie LED-ul RGB luminează alternative in Roșu și Albastru.

* **Senzor magnetic Hall:**

Acest senzor este 0 când magnetul este în aproape de senzor și 1 când nu este, adică ușa este deschisă.

* **Senzor de vibrație:**

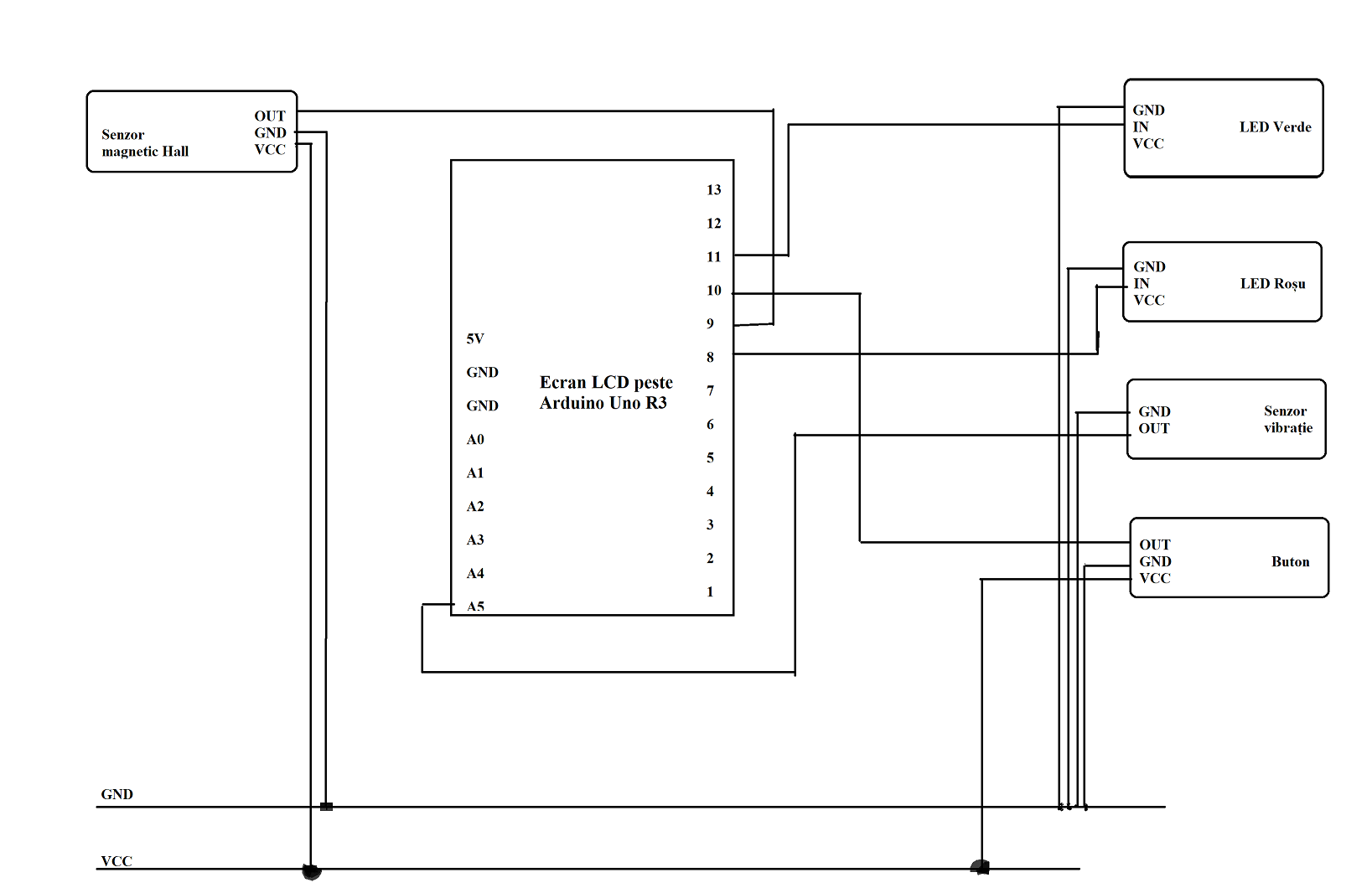
Senzorul este folosit pentru detectarea mișcării.

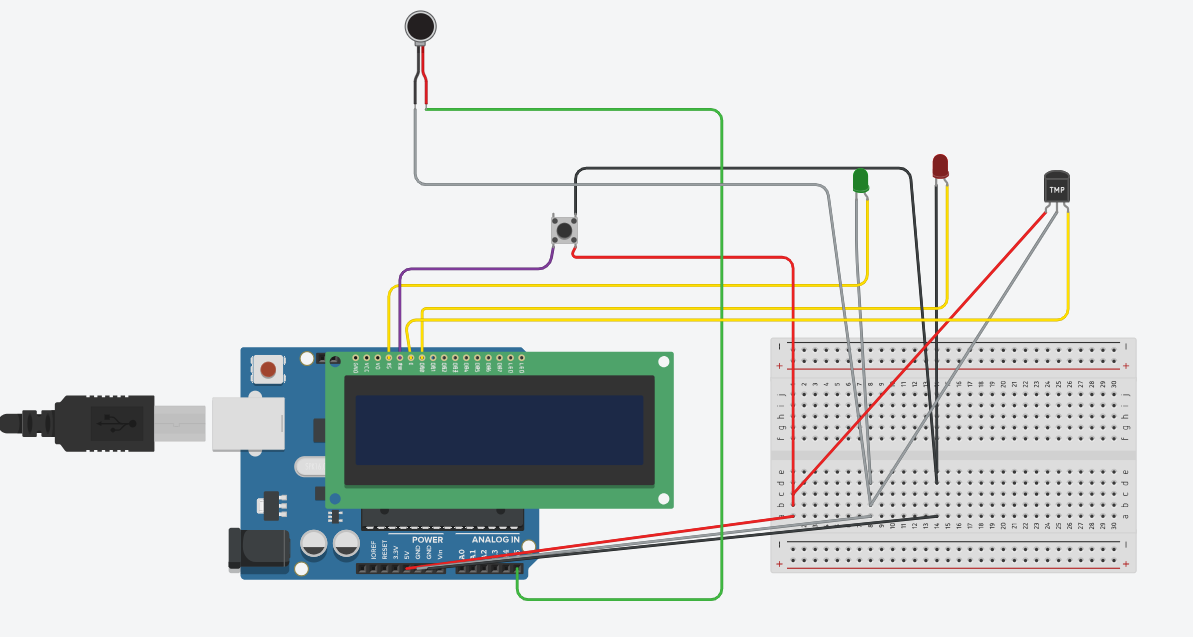
* **Ecran LCD:**

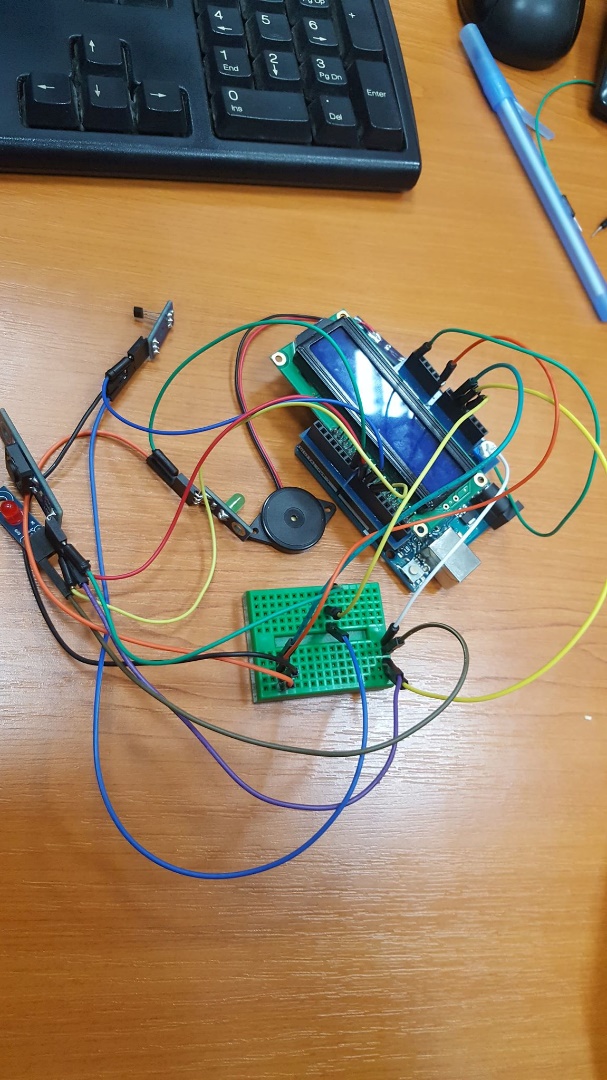
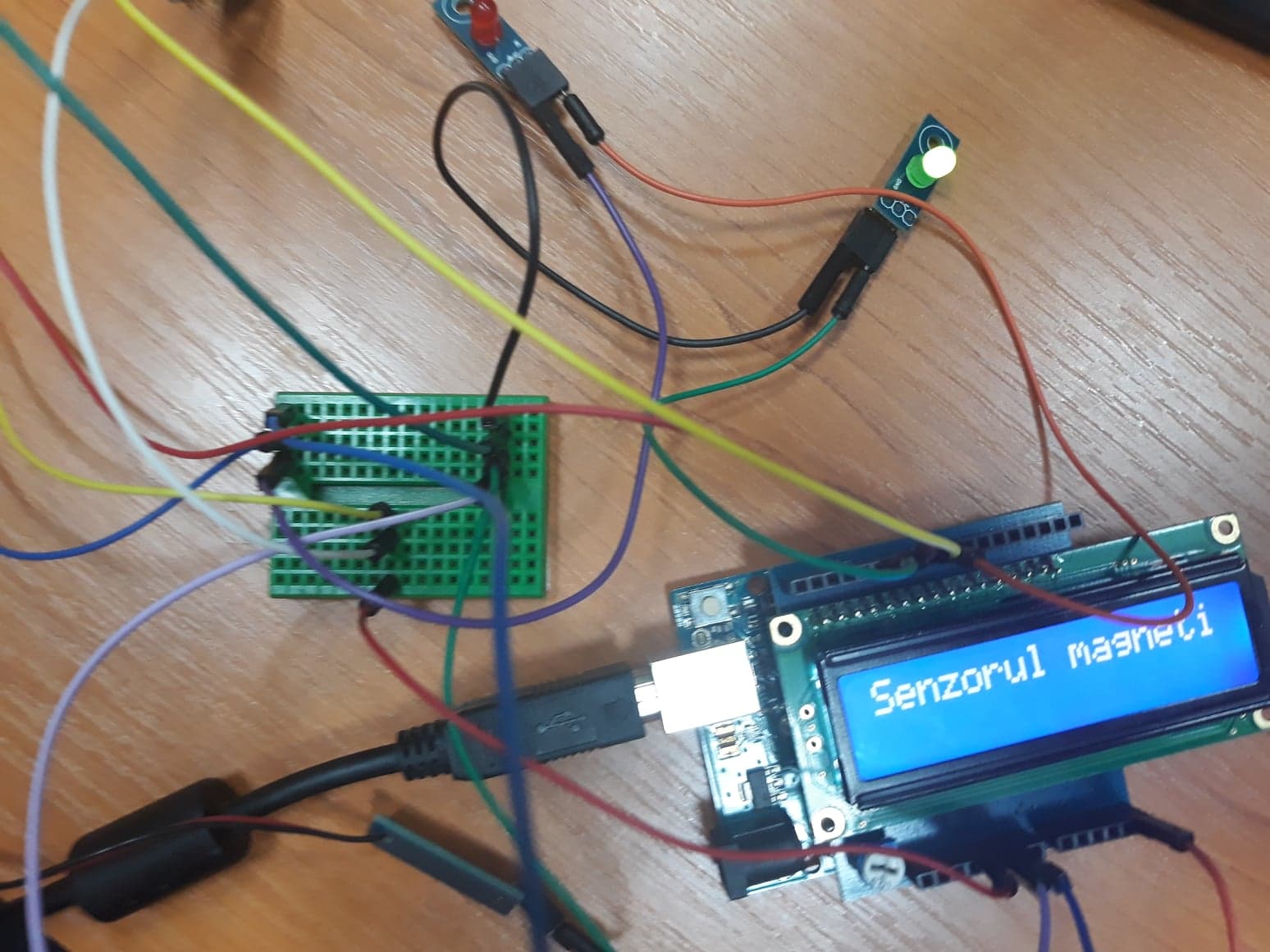
Dacă sistemul este activat şi senzorii detectează mișcare, va apărea mesajul „Efracție!” pe ecran. Dacă sistemul este dezarmat şi senzorii detectează ceva, se va afișa numele senzorului care a detectat efracția.

* **Buton:**

La apăsarea butonului se pornește/oprește sistemul.

**Schema electrică a sistemului**





**Schema logică**

START

Nu detectează

Nu detectează

Detectează

Detectează

Sistem activat

Sistem dezactivat

LCD-Dezarmat

LED-uri oprite

LCD armat

LED roșu aprins

LCD armat

LED-ul verde pâlpâie

LCD -Efracție

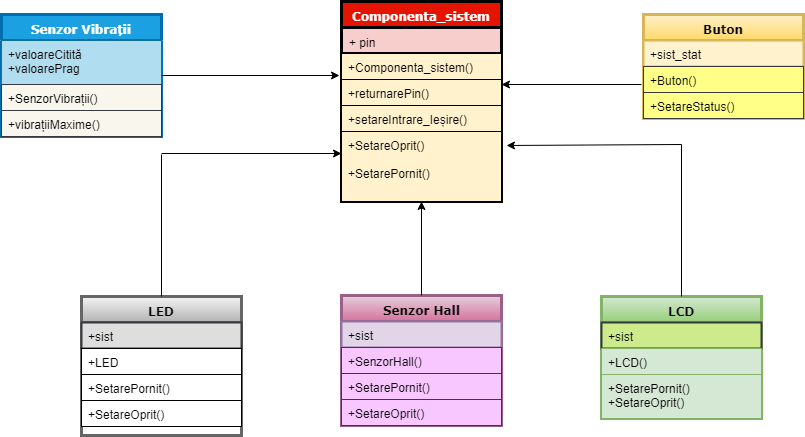
LED verde pornit

Senzor magnetic și de vibrații

Senzor magnetic și de vibrații

Buton

**Schema UML**

****

**Codul sursa a programului si rezultatele testării**

|  |  |  |
| --- | --- | --- |
| **Acțiune** | **Rezultat așteptat** | **Rezultat obținut** |
| Apăsarea butonului | Sistemul se armează și dezarmează la apăsarea butonului. | Sistemul se comportă corect. |
| Se mișcă senzorul de vibrații. | Se detectează efracția și se scrie pe ecranul LCD textul „Efracție!”.  LED-ul roșu se aprinde. | Pe LCD s-a afișat textul „Efracție!” și s-a aprins LED-ul roșu. |
| Se depărtează magnetul de senzorul Hall. | Se detectează efracția și se scrie pe ecranul LCD textul „Efracție!”.  LED-ul roșu se aprinde. | Pe LCD s-a afișat textul „Efracție!” și s-a aprins LED-ul roșu. |
| Se dezarmează sistemul și se mișcă senzorul de mișcare sau se depărtează magnetul de senzorul Hall. | Se afișează numele senzorului care a înregistrat efracția și se aprinde LED-ul verde. | S-a afișat numele senzorului corespunzător fiecărui caz și s-a aprins LED-ulverde. |

Probleme întâmpinate la testările anterioare:

* Delay-ul pentru leduri
* Una dintre stări nu era bine programată (sistemul rămânea blocat pe o stare)
* Senzorul de mișcare este foarte sensibil, așa că a fost nevoie de ajustări la pragul de sensibilitate al acestuia.

Programul:

#include <LiquidCrystal.h> //biblioteca pentru lcd

typedef enum //enumerare care se foloseste pentru setarea sistemului

{

stare\_oprit, stare\_pornit, stare\_efractie, stare\_senzor\_detecteaza

}

stare\_sistem;

class Componenta\_sistem //Clasa de bază

{

protected:

int pin;

public:

Componenta\_sistem(int p) //constructor

{

pin=p;

}

int returnarePin() // functia care returnareaza pin-ului

{

return pin;

}

void setareIntrare\_iesire(int intrare\_iesire) // functie pentru setarea intrarii sau iesirii

{

pinMode(pin, intrare\_iesire);

}

virtual void SetareOprit() { } //functie virtuala pentru oprire a unei componente

virtual void SetarePornit() { } //functie virtuala pentru pornire a unei componente

};

//clasa derivata Led din clasa de baza Componenta\_sistem

class LED: public Componenta\_sistem

{

private:

stare\_sistem sist;

public:

LED(int pin, stare\_sistem s) : Componenta\_sistem(pin) //constructor

{

sist = s;

}

void SetarePornit() //setare pornire led

{

digitalWrite(pin, HIGH);

sist = stare\_pornit;

}

void SetareOprit() //setare oprire led

{

digitalWrite(pin, LOW);

sist = stare\_oprit;

}

};

//butonul este folosit pentru a opri sau a porni sistemul de antiefractie

class Buton : public Componenta\_sistem

{

private:

stare\_sistem sist\_stat;

public:

Buton(int pin, stare\_sistem s) : Componenta\_sistem(pin) //constructor

{

sist\_stat = s;

}

void SetareStatus(stare\_sistem s) //setarea butonului - oprit sau pornit

{

sist\_stat = s;

}

};

//clasa derivata SenzorHall din clasa de baza Componenta\_sistem

class SenzorHall : public Componenta\_sistem

{

public:

SenzorHall(int pin) : Componenta\_sistem(pin) //constructor

{

}

int citire() //functia care citeste valorile senzorului magnetic

{

return digitalRead(pin);

}

};

//clasa derivata Vibratii din clasa de baza Componenta\_sistem

class SenzorVibratii : public Componenta\_sistem // se foloseste pentru senzorul de vibratii

{

private:

int valoareCitita; // valoarea citita de la senzor

public:

SenzorVibratii(int pin):Componenta\_sistem(pin) //constructor

{

}

//functie folosita de clasa SenzorVibratii care ajuta la aflarea valorii maxime a vibratiilor dintr-un interval de timp

int vibratiiMaxime(int pin, int count)

{

int valMaxima = 0;

for(int i = 0; i < count; i++)

{

valMaxima = max(analogRead(pin), valMaxima);

}

return valMaxima;

}

int citesteVibratii() //functie care returneaza valoarea vibratiilor

{

valoareCitita = vibratiiMaxime(pin, 100);

return valoareCitita;

}

};

//declararea variabilelor folosite in program si initializarea lor

stare\_sistem stareSistem = stare\_senzor\_detecteaza;

LiquidCrystal lcd(7,6, 5, 4, 3, 2);

LED LEDverde(11,stare\_oprit);

LED LEDrosu(8,stare\_oprit);

Buton buton(10,stare\_oprit);

SenzorHall senzor\_magnetic(9);

SenzorVibratii s\_vibratii(A5);

//initializarea modului in care sunt setati pinii fiecarei componente (intrare sau de iesire)

void setup()

{

LEDverde.setareIntrare\_iesire(OUTPUT);

LEDrosu.setareIntrare\_iesire(OUTPUT);

buton.setareIntrare\_iesire(INPUT);

s\_vibratii.setareIntrare\_iesire(INPUT);

senzor\_magnetic.setareIntrare\_iesire(INPUT);

lcd.begin(16, 2);

Serial.begin(9600);

}

void loop() //codul propriu-zis ce indeplineste functia ceruta

{

switch (stareSistem) //se verifica in ce stare este sistemul

{

//cazul in care sistemul este oprit dar senzorii detecteaza atunci lcd afiseaza care sensor detecteaza si se aprinde ledul verde

case stare\_senzor\_detecteaza:

LEDverde.SetareOprit();

LEDrosu.SetareOprit();

lcd.clear();

if(s\_vibratii.citesteVibratii()>2)

{

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Senzorul de vibratii activ");

LEDverde.SetarePornit();

delay(2000);

}

else

{ if(senzor\_magnetic.citire()==HIGH)

{

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Senzorul magnetic activ");

LEDverde.SetarePornit();

delay(2000);

}

}

if (digitalRead(buton.returnarePin()) == HIGH)

{

stareSistem = stare\_pornit;

Serial.print("Stare pornit");

}

break;

// in cazul in care este pornit sistemul dar senzorii nu detecteaza nimic, atunci led verde palpaie

case stare\_pornit:

LEDverde.SetareOprit();

LEDrosu.SetareOprit();

Serial.print("Intrat case stare\_pornit");

Serial.print(s\_vibratii.vibratiiMaxime(A5,100));

if(s\_vibratii.citesteVibratii()==0 && senzor\_magnetic.citire()==LOW)

{

lcd.clear();

LEDverde.SetarePornit();

delay(2000);

LEDverde.SetareOprit();

delay(2000);

}

else

{

stareSistem= stare\_efractie;

Serial.print("Stare efractie");

}

//if (digitalRead(buton.returnarePin()) == HIGH)

//{

// stareSistem = stare\_senzor\_detecteaza;

// }

break;

// in cazul in care sistemul e pornit si exista o efractie ,atunci lec afiseaza ecractie si becul rosu se aprinde

case stare\_efractie:

LEDverde.SetareOprit();

LEDrosu.SetareOprit();

Serial.print("A intrat stare efractie!");

if(s\_vibratii.citesteVibratii()>2 || senzor\_magnetic.citire()==HIGH)

{

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Efractie!!!!");

LEDrosu.SetarePornit();

delay(2000);

}

if (digitalRead(buton.returnarePin()) == HIGH)

{

stareSistem = stare\_senzor\_detecteaza;

Serial.print("plecat");

}

break;

}

}

**7.Concluzii**

În urma realizării acestui proiect am învăţat să lucrăm în echipă şi am aprofundat cunoştinţe teoretice cu privire la limbajul C++ și Arduino IDE.

Pe parcursul proiectului au fost întâmpinate și depășite mai multe dificultăţi, atât legate de munca în echipă, cât şi tehnice. Problemele tehnice au fost legate de componente, firele erau în dezordine și trebuia să identificăm fiecare legătură în parte.