Alarm Sensor

Introducere

Proiectul consta in realizarea unui sistem de alarma care foloseste un senzor cu infrarosu pentru a detecta un intrus. Senzorul va fi plasat intr-un aumit loc, iar daca cineva il declanseaza, o alarma sonora si luminoasa se va activa. Daca se doreste dezactivarea sistemului, trebuie introdusa o parola / folosirea unei telecomenzi, iar un LED verde se va activa iar pe ecranul LCD va aparea mesajul “Sys OFF”.

Scopul acestui proiect este a proiecta, la nivel conceptual, un sistem de alarma pentru sporirea sigurantei casei sau a curtii. Fiind un proiect destul de mic, poate fi folosit in incaperi de mai multe dimensiuni.

Ideea de la care am pornit pentru acest proiect a fost de a reusi sa implementez un sistem de securitate fizic. A fost inspirat de situatia de la bunici de la curte, in care un astfel de sistem este important. Mereu am vrut sa vad cat de greu ar fi sa fac eu singur un astfel de sistem, iar cu prilejul acestui proiect, sper ca voi reusi sa implementez cu succes acest proiect. Senzorul infrarosu va fi unul PIR, fiind cel mai des intalnit tip de senzor cu infrarosu care este folosit pentru un sistem de alarma.

Acest proiect fiind un POC, poate fi expandat prin folosirea unor componente mult mai avansate, care pot fi folosite si pe distante mai mari si avand o precizie mult mai bune decat cele din kit-ul pe care il voi folosi eu pentru acest mini proiect. Astfel, poate deservi si ca un punct de inspiratie si pentru alte persoane care doresc sa implementeze mici proiecte / sisteme.

Descriere generală

Schema Bloc

Schema electrica

Descriere intereactiune hardware si a componentelor

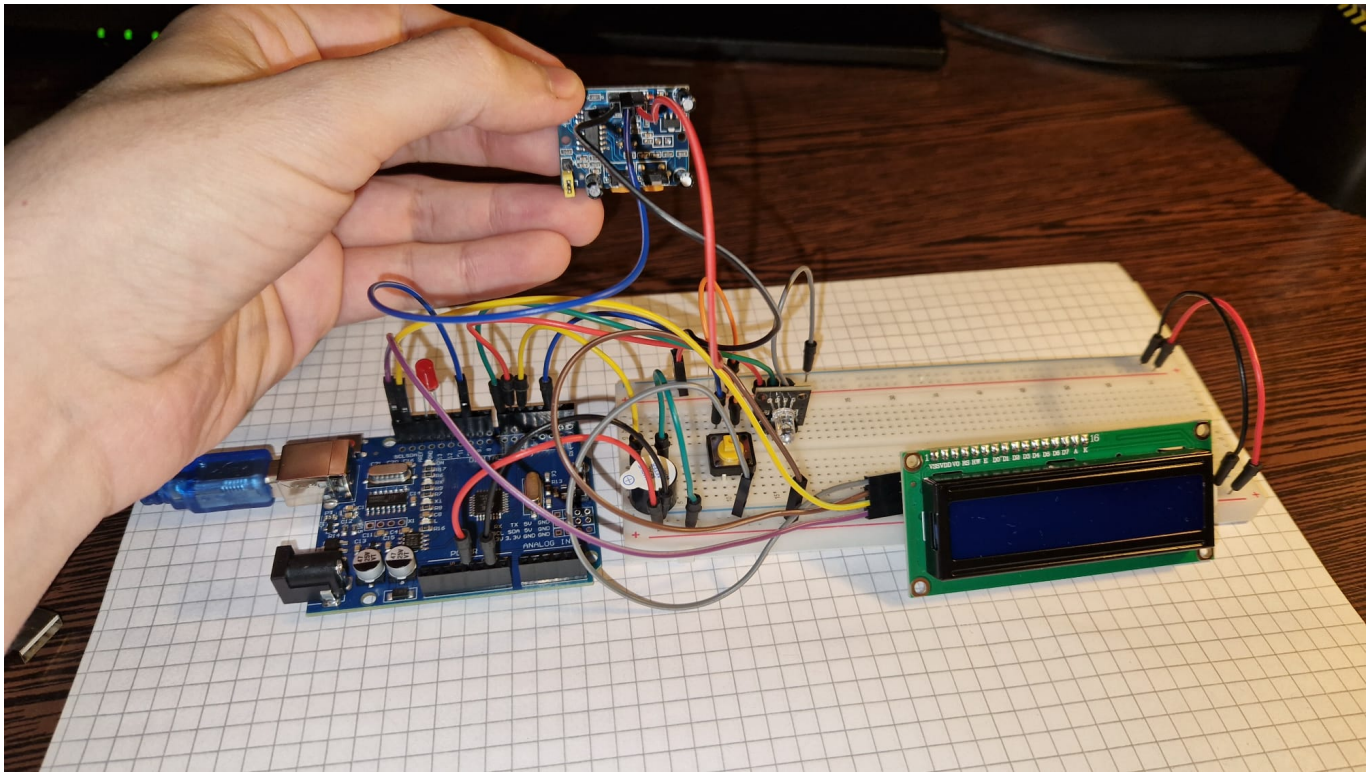
- Dupa pornirea sistemului, senzorul de miscare este inactiv pentru o perioada de timp, pentru a se putea calibra. Acest lucru este indicat de mesajul de pe ecranul LCD.
- Sistemul este armat de la inceput, indicat de LED-ul RGB rosu care este aprins continuu, precum si de mesajul “System On”.
- Prin apasarea unui buton, sistemul se poate dezactiva / activa.
- Daca sistemul este activat, iar senzorul PIR detecteaza miscare, se trimite un semnal catre microprocesor, iar un buzzer si un LED rosu se vor activa, precum si afisarea unui mesaj sugestiv pe ecran. Dupa ce miscarea nu mai este detectata, buzzerul si LED-ul se vor opri.

Hardware Design

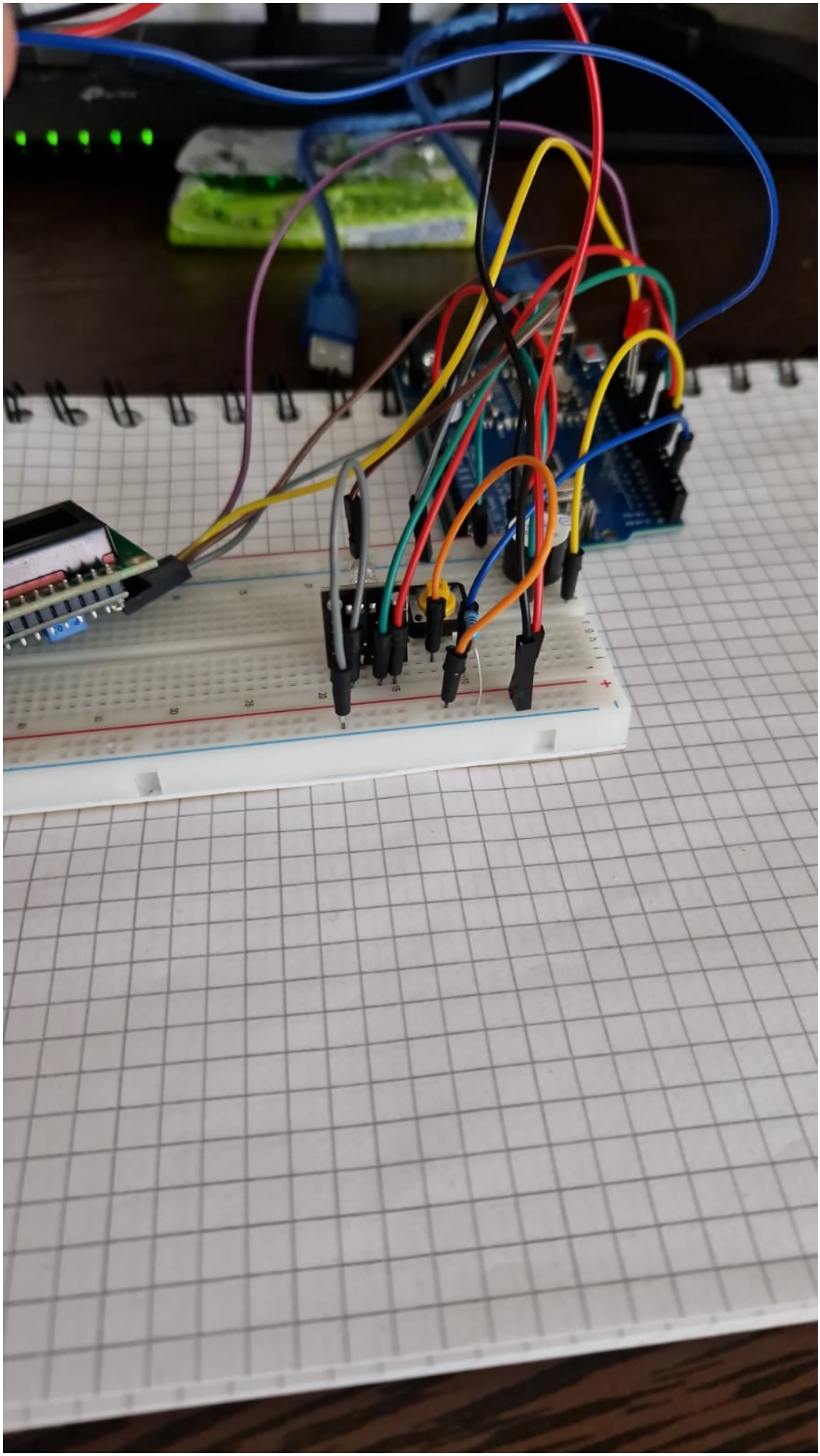
Componentele pe care le voi folosi sunt urmatoarele

- Placuta Arduino Uno R3 (ATMega 328P)
- Breadboard
- LED rosu, care se va aprinde in momentul in care este detectata miscare.
- LED RGB. Culorile rosu si verde sunt folosite pentru a indica daca sistemul este pornit sau oprit.
- Buzzer pasiv care va emite sunet cand se va detecta miscare.
- Modul senzor cu infrarosu (PIR HC-SR501), folosit pentru detectarea miscarii
- 1602 LCD, interfata I2C, folosit pentru a afisa starea sistemului si daca s-a detectat miscare
- Buton

Ansamblu placuta + componente



Fata de schema de pe tinkercad, aici se poate vedea si LED-ul RGB pe care il folosesc. Prin apasarea butonului sau dupa ce trec 15 secunde fara sa fie detectata vreo miscare, sistemul se dezactiveaza, si se aprinde LED-ul verde si se afiseaza pe ecran mesajul corespunzator.



Pini folositi

- Pinul 13 este folosit pentru a aprinde LED-ul rosu in caz ca este detectata miscare
- Pinul 10 este folosit pentru a citit input-ul de la senzor, iar daca se detecteaza miscare, starea acestuia va fi “High”
- Pinul 5 est dedicat buzzer-ului pentru a-l porni / opri
- Pinul 2 este pentru butonul care imi opreste / porneste sistemul
- Pinii 6 si 7 sunt folositi pentru LED-ul RGB, pentru culorile rosu si verde, respectiv.

Software Design

Mediu de Dezvoltare

Arduino IDE, apoi mai tarziu am folosit VSC cu PlatformIO, importand proiectul de Arduino scris anterior.

Librarii 3rd-party

Am folosit pentru display-ul LCD biblioteca arduino LiquidCrystal_I2C.

Detalii despre cod

- Functia setup(): am setat toti pinii ca INPUT sau OUTPUT, am initializat LCD-ul, senzorul de miscare si am configurat intreruperile pe buton, folosint functii Arduino, si pe timer, folosind registrii.

```
void setup()
{
  Serial.begin(9600);
  pinMode(pirPin, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(RGB_RED_PIN, OUTPUT);
  pinMode(RGB_GREEN_PIN, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP); // Set button pin as input with pull-up
resistor

  digitalWrite(pirPin, LOW);
```

```
  lcd_1.init();
  lcd_1.backlight();
  Serial.println("Hello World!");
  lcd_1.print("Starting...");
  pinMode(9, OUTPUT);

  // Attach interrupt to button pin
  attachInterrupt(digitalPinToInterrupt(buttonPin), toggleSystem, FALLING);

  // Timer1 configuration: 1Hz (1 second interval)
  cli();                      // Disable all interrupts
  TCCR1A = 0;                           // Set entire TCCR1A register to 0
  TCCR1B = 0;                           // Same for TCCR1B
  TCNT1 = 0;                            // Initialize counter value to 0
  OCR1A = 15624;                        // Compare match register
(16MHz/1024/1Hz - 1)
  TCCR1B |= (1 << WGM12);               // CTC mode
  TCCR1B |= (1 << CS12) | (1 << CS10); // 1024 prescaler
  TIMSK1 |= (1 << OCIE1A);              // Enable timer compare interrupt
  sei();                          // Enable all interrupts

  // Give the sensor some time to calibrate
  Serial.print("Calibrating sensor ");
  for (int i = 0; i < calibrationTime; i++)
  {
    Serial.print(".");
    delay(1000);
  }
  Serial.println("\nDone");
  Serial.println("SENSOR ACTIVE");
  delay(50);
}
```

- Functia loop(): Aici se intampla toata “magia” din cod. Se verifica la fiecare iteratie daca sistemul este pornit sau oprit si daca este miscare detectata.

```
 void loop()
{
  // Set RGB LED state
  if (isOn)
  {
    digitalWrite(RGB_RED_PIN, HIGH);
    digitalWrite(RGB_GREEN_PIN, LOW);
  }
  else
  {
    digitalWrite(RGB_RED_PIN, LOW);
    digitalWrite(RGB_GREEN_PIN, HIGH);
  }

  // Update LCD if state changed
```

```
  if (isOn != lastState)
  {
    lastState = isOn;
    lcd_1.clear();
    lcd_1.setCursor(0, 0);
    if (isOn)
    {
      lcd_1.print("System ON");
    }
    else
    {
      lcd_1.print("System OFF");
    }
  }

  // Handle PIR sensor
  if (isOn)
  {
    if (digitalRead(pirPin) == HIGH)
    {
      digitalWrite(buzzerPin, HIGH); // Turn on the buzzer
      if (lockLow)
      {
        lockLow = false;
        Serial.println("---");
        Serial.print("Motion detected at ");
        lcd_1.clear();
        lcd_1.print("Detected Movement");
        Serial.print(millis() / 1000);
        Serial.println(" sec");
        delay(50);
      }
      takeLowTime = true;
      lastActivityTime = millis(); // Reset the activity timer
    }

    digitalWrite(ledPin, digitalRead(pirPin));
    digitalWrite(buzzerPin, digitalRead(pirPin)); // Turn on the buzzer

    if (digitalRead(pirPin) == LOW)
    {
      digitalWrite(ledPin, LOW);    // Visualize sensor output state with
LED
      digitalWrite(buzzerPin, LOW); // Turn off the buzzer
      if (takeLowTime)
      {
        lowIn = millis();
        takeLowTime = false;
      }
      if (!lockLow && millis() - lowIn > pause)
```

```
      {
        lockLow = true;
        Serial.print("Motion ended at ");
        Serial.print((millis() - pause) / 1000);
        Serial.println(" sec");
        lcd_1.clear();
        lcd_1.print("Movement Stopped");
        delay(50);
      }
    }
  }
}
```

- Variabile si tratarea intreruperilor:
- Pentru intreruperile de timer si buton, am o rutina speciala in care opresc / pornesc sistemul si actualizez mesajul de pe LCD si culoarea LED-ului RGB corespunzator.

```
LiquidCrystal_I2C lcd_1(0x27, 16, 2);
int calibrationTime = 5; // Calibration time for the PIR sensor

volatile bool isOn = true; // Make isOn volatile since it will be accessed
from an interrupt
bool lastState = false;
long unsigned int lowIn;
long unsigned int pause = 200; // Pause duration before assuming motion has
stopped
boolean lockLow = true;
boolean takeLowTime = false;
int ledPin = 13;   // LED pin for visualization
int pirPin = 10;   // PIR sensor pin
int buzzerPin = 5; // Buzzer pin
int buttonPin = 2; // Button pin for interrupt

int RGB_RED_PIN = 6;
int RGB_GREEN_PIN = 7;

boolean systemActive = true;

// Debounce variables
volatile unsigned long lastDebounceTime = 0;
const unsigned long debounceDelay = 50; // 50 milliseconds debounce time

// Timer variables
volatile unsigned long lastActivityTime = 0; // Last time activity was
detected
const unsigned long shutdownDelay = 15000;   // 10 seconds delay before
shutdown

void toggleSystem()
{
  // Check if enough time has passed since the last press to debounce
```

```
  if ((millis() - lastDebounceTime) > debounceDelay)
  {
    isOn = !isOn;
    lastDebounceTime = millis(); // Update the last debounce time
    lastActivityTime = millis(); // Reset the activity timer
  }
}
ISR(TIMER1_COMPA_vect)
{
  // Timer1 ISR: Increment seconds counter
  if (isOn && millis() - lastActivityTime >= shutdownDelay)
  {
    isOn = false;
  }
}
```

Explicatii cod

Dupa ce am asignat fiecarei componente un pin, am setat corespunzator acel pin ca fiind de input sau output. Apoi am oferit senzorului PIR un timp de 5 secunde pentru calibrare, altfel ar avea un comportament imprevizibil.

Intreruperile pe timer si pe buton sunt folosite doar pentru a opri sau a porni sistemul.

In momentul in care senzorul de PIR detecteaza miscare, trimite un semnal catre microcontroller, iar in cod se verifica daca sistemul este pornit sau oprit. In cazul in care sistemul este oprit, nu se va intampla nimic. In caz contrar, cat timp este miscare detectata se va aprinde un LED rosu si buzzerul va emite un sunet, indicand prezenta unui intrus.

Daca timp de 15 secunde nu este detectata nicio miscare, sistemul se va opri.

Rezultate Obţinute

Am invatat cum sa conectez, folosind un bradboard, componentele, urmarind data-sheet-ul pentru fiecare componenta in parte. Apoi am invatat cat de usor este sa scriu cod Arduino pentru a face un mic proiect, chiar daca pe alocuri am facut niste greseli stupide, fie in cod, fie cand incercam sa conectez componentele.

Am reusit, totusi, sa implementez cu succes un mic sistem de alarma care foloseste un senzor cu infrarosu, chiar daca acum imi dau seama ca nu era chiar atat de greu pe cat ma asteptam initial. Cautand explicatii pe internet sau prin trial and error, in decursul a cateva zile am ajuns de la o multime de piese intr-o cutie de plastic la ceva practic si functional.

Concluzii

A fost un proiect interesant si amuzant, desi mi-as fi dorit sa am mai mult timp la dispozitie sa fac ceva mai complex si mai interesant, dar am crezut la inceput ca va fi foarte greu si de aceea am ales o tema mai usor de implementat. Daca as putea sa aleg din nou tema proiectului, as alege tot un sistem de alarma, dar l-as face mai complex si la scara poate mai mare.

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

- Dupa ce am ales tema, am comandat kit-ul cu toate (sau aproape toate :P) componentele de pe Emag.
- Am inceput sa caut pe net toate datasheet-urile componetelor care mi-au venit si mi-am dat seama ca imi lipseste senzorul de miscare, fix componenta principala din proiect, deci am comandat-o separat.
- Am facut mai multe simulari pe TinkerCad, din frica de a nu arde componente si de a ma asigura ca problemele care apar in cod pot fi reparate inainte de produsul final.
- Am inceput asamblarea fizica a sistemului.
- M-am chinuit sa imi dau seama ca senzorul de miscare este mult prea sensibil, deci a trebuit sa rezolv asta. A fost un fix usor de facut, doar am umblat la potentiometre pana cand am obtinut un senzor destul de sensibil incat sa detecteze miscare, dar care sa nu fie 24/7 pornit.
- Am scris codul initial, am testat si totul mergea ok, apoi am inceput sa folosesc intreruperi si sa schimb niste cod sa fie folositi si registrii, nu doar functii arduino.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

proiect_pm.zip

Export to PDF