**НП ”Обучение за ИТ умения и кариера”**

**Модул 8: Въведение в операционни и вградени системи**

**КУРСОВ ПРОЕКТ**

на тема:

**Охранителна система (СОТ)**

Изготвили:

Михаил Тенев

Група 08

гр. Хасково

2025 г.

Съдържание:

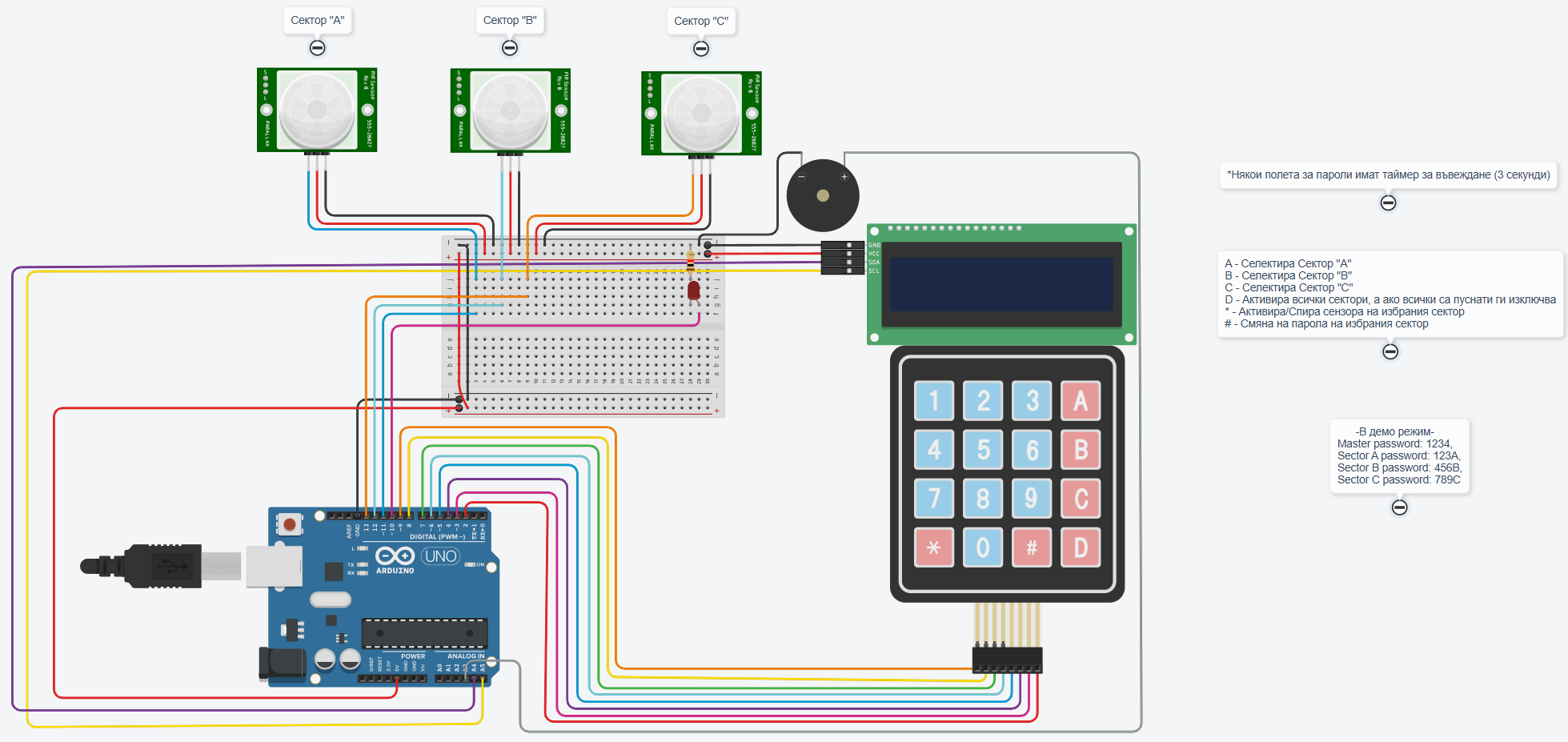
1. Описание на проекта
2. Електронна схема
3. Блокова схема
4. Електрическа схема
5. Списък съставни части
6. Сорс код – описание на функционалността
7. Заключение

[Thinkercad](https://www.tinkercad.com/things/92IRcRXdFbw-sot)

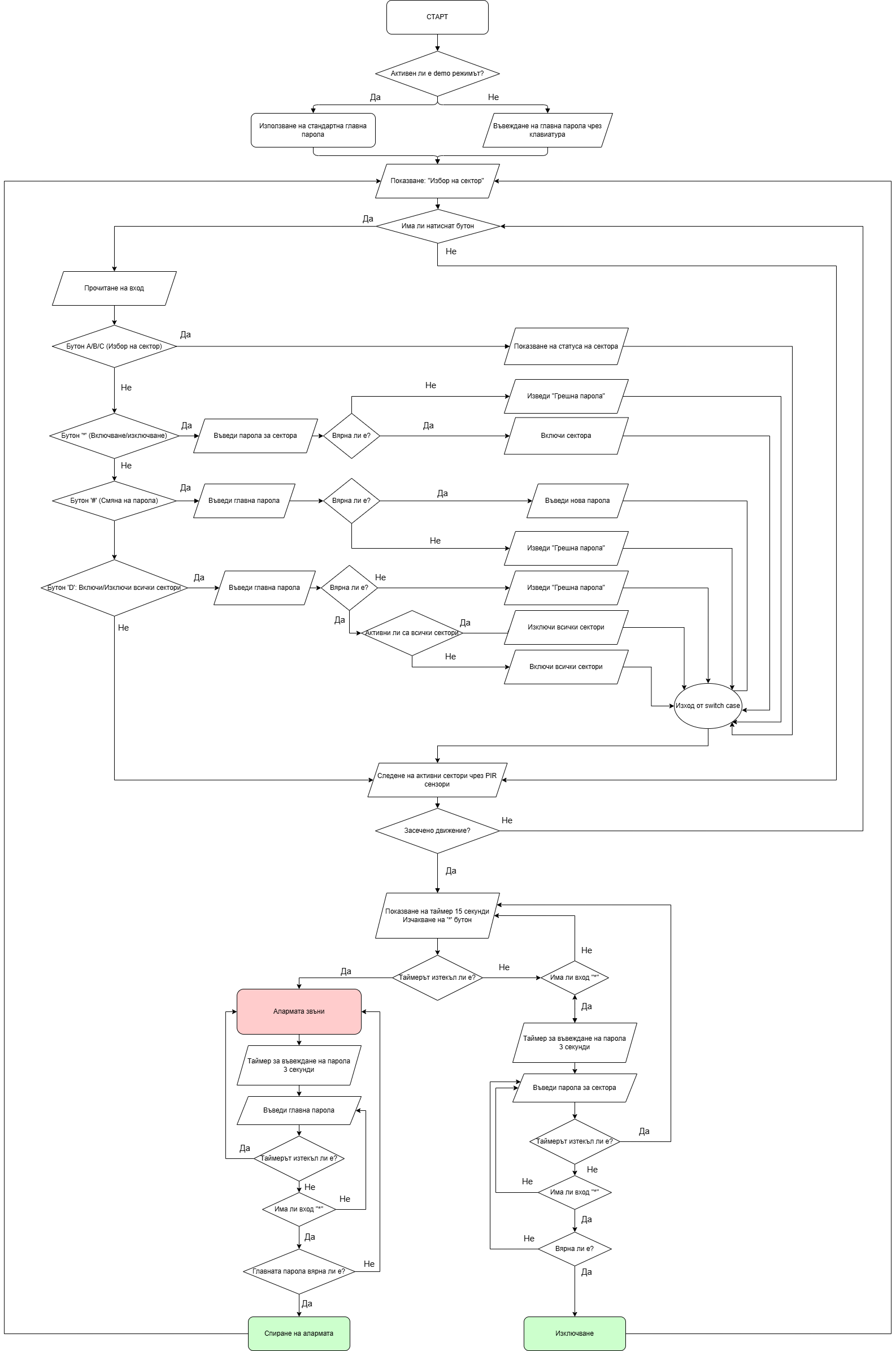
1. **Описание на проекта**

Проектът реализира охранителна система, изградена с помощта на Arduino, която позволява потребителят да управлява и защитава три независими зони (сектори). Всяка зона може да бъде активирана или деактивирана с индивидуална парола. При засичане на движение в активиран сектор, се стартира аларма.

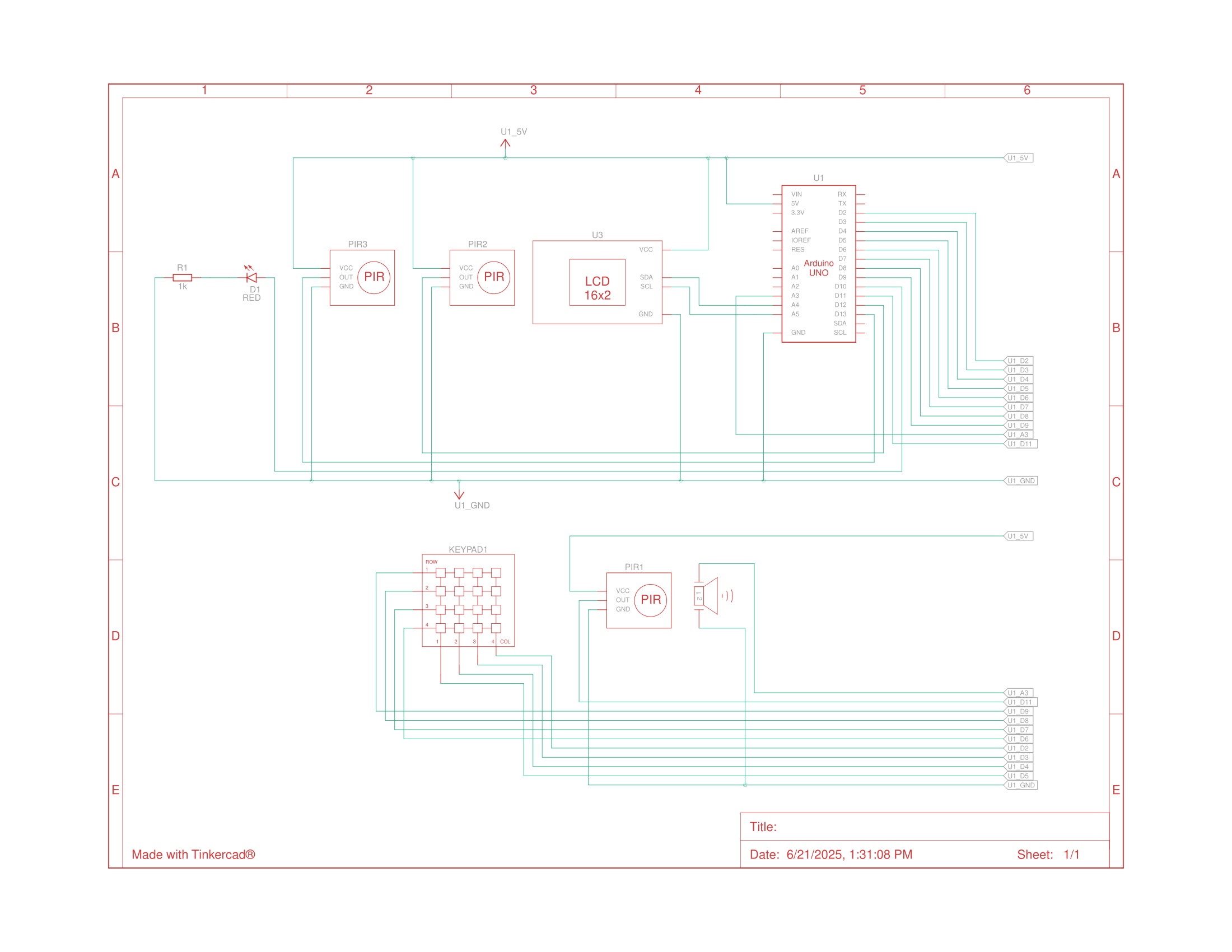
1. **Електронна схема**



1. **Блокова схема**



1. **Електрическа схема**

****

1. **Списък съставни части**

|  |  |
| --- | --- |
| **Компоненти** | **Бройка** |
| Arduino Uno R3 | 1 |
| PIR Sensor | 3 |
| MCP23008-based, 32 (0x20) LCD 16 x 2 (I2C) | 1 |
| Red LED | 1 |
| 1 kΩ Resistor | 1 |
| Keypad 4x4 | 1 |
| Piezo | 1 |

1. **Сорс код и описание на функционалността**

|  |
| --- |
| **#include <Adafruit\_LiquidCrystal.h>**  **#include <Keypad.h>**  **const int LED = 10;**  **const int BUZZ = A3;**  **const byte ROWS = 4;**  **const byte COLS = 4;**  **char hexaKeys[ROWS][COLS] = {**  **{'1', '2', '3', 'A'},**  **{'4', '5', '6', 'B'},**  **{'7', '8', '9', 'C'},**  **{'\*', '0', '#', 'D'}**  **};**  **bool onStart = false;**  **bool demo = true;**  **byte rowPins[ROWS] = {9, 8, 7, 6};**  **byte colPins[COLS] = {5, 4, 3, 2};**  **Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);**  **Adafruit\_LiquidCrystal lcd(0);**  **class Password {**  **private:**  **int length;**  **char\* passwordSequence;**  **public:**  **Password(const char\* input) {**  **this->length = strlen(input);**  **passwordSequence = new char[length + 1];**  **strcpy(passwordSequence, input);**  **}**  **~Password() {**  **delete[] passwordSequence;**  **}**  **bool check(const char\* input) {**  **if (strlen(input) != length) return false;**  **for (int i = 0; i < length; i++) {**  **if (input[i] != passwordSequence[i]) return false;**  **}**  **return true;**  **}**  **};**  **Password\* masterPassword;**  **char\* EnterPass(bool censor, bool timer = false) {**  **const int maxLen = 9;**  **char\* input = new char[maxLen + 1];**  **int index = 0;**  **unsigned long timeoutMillis = 3000;**  **unsigned long startTime = millis();**  **while (index < maxLen) {**  **if (timer && (millis() - startTime) > timeoutMillis) {**  **input[0] = '\0';**  **return input;**  **}**  **char key = customKeypad.getKey();**  **if (key) {**  **startTime = millis();**  **if (key == '\*' && index > 0) break;**  **if (key != '\*') {**  **input[index++] = key;**  **lcd.print(censor ? '\*' : key);**  **}**  **}**  **delay(10);**  **}**  **input[index] = '\0';**  **return input;**  **}**  **class Sector {**  **public:**  **String name;**  **int sensorPin;**  **bool status;**  **private:**  **Password\* password;**  **public:**  **Sector(String name, int sensorPin, const char\* input) {**  **this->name = name;**  **this->sensorPin = sensorPin;**  **status = false;**  **password = new Password(input);**  **}**  **~Sector() {**  **delete password;**  **}**  **void ChangeState() {**  **lcd.clear();**  **lcd.print("Enter sector");**  **lcd.setCursor(0, 1);**  **lcd.print("password:");**  **char\* enteredPass = EnterPass(true);**  **if (password->check(enteredPass)) {**  **delete[] enteredPass;**  **lcd.clear();**  **lcd.print("Access Granted");**  **status = !status;**  **delay(1500);**  **lcd.clear();**  **} else {**  **delete[] enteredPass;**  **lcd.clear();**  **lcd.print("Wrong Password");**  **delay(1500);**  **lcd.clear();**  **}**  **}**  **void Edit() {**  **lcd.clear();**  **lcd.print("Enter Master");**  **lcd.setCursor(0, 1);**  **lcd.print("password:");**  **char\* enteredPass = EnterPass(true);**  **if (masterPassword->check(enteredPass)) {**  **delete[] enteredPass;**  **lcd.clear();**  **lcd.print("Access Granted");**  **delay(1500);**  **lcd.clear();**  **} else {**  **delete[] enteredPass;**  **lcd.clear();**  **lcd.print("Wrong Password");**  **delay(1500);**  **lcd.clear();**  **return;**  **}**  **lcd.print("New password:");**  **lcd.setCursor(0, 1);**  **char\* newPass = EnterPass(false);**  **delete password;**  **password = new Password(newPass);**  **delete[] newPass;**  **}**  **void Alarm() {**  **lcd.clear();**  **lcd.print("MOTION in ");**  **lcd.setCursor(0, 1);**  **lcd.print(name);**  **delay(1500);**  **unsigned long startTime = millis();**  **const unsigned long waitTime = 15000; // 15 seconds**  **lcd.clear();**  **lcd.print("Press \* to");**  **lcd.setCursor(0,1);**  **lcd.print("disarm");**  **// Allow user to disarm sector within 15 seconds**  **while (millis() - startTime < waitTime) {**  **lcd.setCursor(7, 1);**  **lcd.print("(");**  **lcd.print((waitTime - (millis() - startTime)) / 1000);**  **lcd.print("s) ");**  **char key = customKeypad.getKey();**  **if (key == '\*') {**  **lcd.clear();**  **lcd.print("Enter Sector");**  **lcd.setCursor(0, 1);**  **lcd.print("password:");**  **char\* enteredPass = EnterPass(true, true);**  **if (password->check(enteredPass)) {**  **delete[] enteredPass;**  **lcd.clear();**  **lcd.print("Disarmed!");**  **status = false;**  **delay(1500);**  **lcd.clear();**  **lcd.print("-Select Sector-");**  **return;**  **} else if(enteredPass[0] == '\0'){**  **delete[] enteredPass;**  **lcd.clear();**  **lcd.print("Timeout");**  **delay(1500);**  **lcd.clear();**  **lcd.print("Press \* to");**  **lcd.setCursor(0,1);**  **lcd.print("disarm");**  **delay(1500);**  **} else {**  **delete[] enteredPass;**  **lcd.clear();**  **lcd.print("Wrong password");**  **delay(1500);**  **lcd.clear();**  **lcd.print("Press \* to");**  **lcd.setCursor(0,1);**  **lcd.print("disarm");**  **delay(1500);**  **}**  **}**  **}**  **// ALARM TRIGGERED**  **while (true) {**  **Indicator();**  **lcd.clear();**  **lcd.print("Master password");**  **lcd.setCursor(0, 1);**  **lcd.print("to disarm: ");**  **char\* masterAttempt = EnterPass(true, true);**  **if (masterPassword->check(masterAttempt)) {**  **delete[] masterAttempt;**  **lcd.clear();**  **lcd.print("Alarm Off");**  **status = false;**  **delay(1500);**  **lcd.clear();**  **lcd.print("-Select Sector-");**  **return;**  **} else if(masterAttempt[0] == '\0'){**  **delete[] masterAttempt;**  **} else {**  **delete[] masterAttempt;**  **lcd.clear();**  **lcd.print("Wrong Master");**  **lcd.setCursor(0,1);**  **lcd.print("password");**  **delay(1500);**  **}**  **}**  **}**  **};**  **void Indicator(){**  **lcd.clear();**  **lcd.print("!!! ALARM !!!");**  **digitalWrite(LED, HIGH);**  **tone(BUZZ, 1000);**  **delay(500);**  **digitalWrite(LED, LOW);**  **noTone(BUZZ);**  **delay(500);**  **digitalWrite(LED, HIGH);**  **tone(BUZZ, 1000);**  **delay(500);**  **digitalWrite(LED, LOW);**  **noTone(BUZZ);**  **delay(500);**  **}**  **Sector\* sectors[3];**  **void SetupPass() {**  **lcd.clear();**  **lcd.print("Enter master");**  **lcd.setCursor(0, 1);**  **lcd.print("password:");**  **char\* mpass = EnterPass(false);**  **masterPassword = new Password(mpass);**  **delete[] mpass;**  **}**  **void EnableAll() {**  **lcd.clear();**  **lcd.print("Master password");**  **lcd.setCursor(0, 1);**  **lcd.print("to toggle all: ");**  **char\* masterAttempt = EnterPass(true, true);**  **if (masterPassword->check(masterAttempt)) {**  **delete[] masterAttempt;**    **// Determine if all sectors are currently enabled**  **bool allEnabled = true;**  **for (int i = 0; i < 3; i++) {**  **if (!sectors[i]->status) {**  **allEnabled = false;**  **break;**  **}**  **}**  **// Toggle all: if all are enabled, disable; otherwise enable all**  **for (int i = 0; i < 3; i++) {**  **sectors[i]->status = !allEnabled;**  **}**  **lcd.clear();**  **lcd.print(allEnabled ? "All Disabled." : "All Enabled.");**  **delay(1500);**  **} else if (masterAttempt[0] == '\0') {**  **lcd.clear();**  **lcd.print("Timeout");**  **delay(1500);**  **} else {**  **delete[] masterAttempt;**  **lcd.clear();**  **lcd.print("Wrong Master");**  **lcd.setCursor(0, 1);**  **lcd.print("password");**  **delay(1500);**  **}**  **}**  **void Cycle() {**  **static int currentSectorIndex = 0;**  **char key = customKeypad.getKey();**  **if (key) {**  **switch (key) {**  **case 'A':**  **currentSectorIndex = 0;**  **break;**  **case 'B':**  **currentSectorIndex = 1;**  **break;**  **case 'C':**  **currentSectorIndex = 2;**  **break;**  **case '\*':**  **if (sectors[currentSectorIndex] != nullptr)**  **sectors[currentSectorIndex]->ChangeState();**  **delay(1000);**  **break;**  **case '#':**  **if (sectors[currentSectorIndex] != nullptr)**  **sectors[currentSectorIndex]->Edit();**  **delay(1000);**  **break;**  **case 'D':**  **EnableAll();**  **break;**  **}**  **// Show selected sector info immediately on key press**  **lcd.clear();**  **Sector\* sector = sectors[currentSectorIndex];**  **if (sector != nullptr) {**  **lcd.print(currentSectorIndex + 1);**  **lcd.print(". -");**  **lcd.print(sector->name);**  **lcd.print("-");**  **lcd.setCursor(0, 1);**  **lcd.print(sector->status ? "On" : "Off");**  **} else {**  **lcd.print(currentSectorIndex + 1);**  **lcd.print(". -Empty-");**  **}**  **}**  **// Check sensors and trigger alarm if needed**  **for (int i = 0; i < 3; i++) {**  **if (sectors[i] != nullptr && sectors[i]->status) {**  **if (digitalRead(sectors[i]->sensorPin) == HIGH) {**  **sectors[i]->Alarm();**  **break;**  **}**  **}**  **}**  **}**  **void setup() {**  **lcd.begin(16, 2);**  **pinMode(LED, OUTPUT);**  **pinMode(BUZZ, OUTPUT);**  **pinMode(11, INPUT);**  **pinMode(12, INPUT);**  **pinMode(13, INPUT);**  **char\* p1 = new char[5]{'1', '2', '3', 'A', '\0'};**  **char\* p2 = new char[5]{'4', '5', '6', 'B', '\0'};**  **char\* p3 = new char[5]{'7', '8', '9', 'C', '\0'};**  **sectors[0] = new Sector("Sector A", 11, p1);**  **sectors[1] = new Sector("Sector B", 12, p2);**  **sectors[2] = new Sector("Sector C", 13, p3);**  **delete[] p1;**  **delete[] p2;**  **delete[] p3;**  **if (demo == true) {**  **onStart = true;**  **char\* pass = new char[5]{'1', '2', '3', '4', '\0'};**  **masterPassword = new Password(pass);**  **delete[] pass;**  **lcd.print("-Select Sector-");**  **}**  **}**  **void loop() {**  **if (!onStart) {**  **SetupPass();**  **lcd.clear();**  **onStart = true;**  **lcd.print("-Select Sector-");**  **}**  **Cycle();**  **}** |

**Принцип на работа:**

1. **Инициализация**:
   * При стартиране се задава master парола (ако не е в демо режим).
   * Създават се 3 сектора с пароли по подразбиране.
2. **Избор на сектор**:
   * Чрез клавиши A, B, C се избира сектор.
   * LCD показва името и състоянието на текущия сектор.
3. **Управление на сектор**:
   * С \* се активира/деактивира сектор (със съответната парола).
   * С # се променя паролата на сектора (с master паролата).
4. **Аларма**:
   * Ако активиран сектор засече движение:
     + Потребителят има 15 секунди да въведе правилната парола.
     + При неуспех се активира аларма (LED мига, зумер свири).
     + Алармата се спира само с master паролата.
5. **Масово управление**:
   * Чрез клавиш D се включват/изключват всички сектори наведнъж с master паролата.
6. **Заключение**

Разработената охранителна система успешно демонстрира как чрез микроконтролер (Arduino), сензори за движение и прост потребителски интерфейс може да се създаде **функционално, ефективно и достъпно решение за охрана на обекти**. Системата предлага **възможност за управление на няколко независими сектора**, защита с пароли и **автоматично реагиране при засечено движение**.

Въведените механизми за сигурност – като индивидуални пароли, master достъп и таймер за деактивиране – допринасят за по-висока надеждност и контрол от страна на потребителя. Проектът показва, че с използването на **достъпни електронни компоненти** може да се изгради **гъвкава и практична охранителна система**, подходяща както за домашна, така и за малко бизнес приложение.

Този проект също така полага основите за бъдещи разширения – например интеграция с интернет (IoT), добавяне на мобилно известяване, или включване на допълнителни биометрични методи за идентификация.