Sistem alarma pe baza de miscare

Introducere

Proiectul meu este un sistem de alarma. Sistemul foloseste un senzor PIR pentru a detecta miscarea, iar in functie de starea alarmei aprinde LED-uri, afiseaza mesaje pe un LCD si porneste un buzzer.

Ideea proiectului a fost sa fac un sistem simplu de securitate, asemanator cu o alarma de camera. Alarma poate fi armata si dezarmata folosind un buton. Cand sistemul nu este armat, LED-ul verde este aprins. Cand alarma este armata LED-ul rosu este aprins si cand senzorul PIR detecteaza miscare, LED-ul rosu palpaie, buzzer-ul incepe sa sune, iar pe LCD apare un mesaj de alerta.

In plus, proiectul trimite mesaje catre un PC prin Bluetooth, folosind modulul HC-05. Astfel, calculatorul poate primi informatii despre starea sistemului, de exemplu daca alarma a fost armata, dezarmata sau daca a fost detectata miscare.

Scopul proiectului este sa combin mai multe notiuni lucrate la laborator, cum ar fi folosirea pinilor GPIO, PWM pentru buzzer, comunicatia USART pentru Bluetooth si I2C pentru LCD. Am ales aceasta idee deoarece este usor de inteles, dar include mai multe componente hardware si mai multe tipuri de comunicatie.

Proiectul poate fi util ca un exemplu simplu de sistem de securitate pentru o camera, birou sau dulap. Chiar daca este un prototip, principiul este asemanator cu cel folosit in sistemele reale de alarma.

Descriere generală

- ATmega328P XMINI este placa principala a proiectului. Ea citeste datele de la senzori si controleaza toate celelalte module: LED-uri, buzzer, LCD si Bluetooth.

- Senzorul PIR HC-SR501 este folosit pentru detectarea miscarii. Cand senzorul detecteaza miscare, trimite un semnal digital catre microcontroller.

- Butonul este folosit pentru armarea si dezarmarea sistemului. Prin apasarea lui se schimba starea alarmei: armata sau dezarmata.

- LED-ul verde indica faptul ca alarma este dezarmata. Cand sistemul nu este activ, acest LED ramane aprins.

- LED-ul rosu indica faptul ca alarma este armata. Cand sistemul este armat, LED-ul rosu sta aprins continuu. Daca este detectata miscare, LED-ul rosu incepe sa palpaie.

- Buzzerul pasiv este folosit pentru semnalizarea sonora a alarmei. Acesta este controlat prin PWM,

astfel incat microcontrollerul genereaza semnalul necesar pentru sunet.

- LCD-ul 1602 cu interfata I2C afiseaza starea sistemului. Pe ecran pot aparea mesaje precum “Sistem dezarmat”, “Sistem armat” sau “Alerta miscare”.

- Modulul Bluetooth HC-05 este folosit pentru comunicatia cu PC-ul. Acesta este conectat la microcontroller prin USART folosit in mod asincron, asemanator unei comunicatii UART.

- PC-ul/laptopul primeste prin Bluetooth mesaje trimise de sistem. De exemplu, cand alarma este armata, dezarmata sau cand este detectata miscare, aceste informatii pot fi vazute intr-un terminal serial.

- Alimentarea sistemului se face la 5V, prin placa ATmega328P XMINI. Modulele principale sunt alimentate de la aceeasi sursa, iar toate componentele au GND comun.

- Sistemul are doua stari principale: dezarmat si armat. In starea dezarmat, LED-ul verde este aprins si miscarea detectata de PIR nu declanseaza alarma. In starea armat, LED-ul rosu este aprins, iar daca PIR-ul detecteaza miscare, alarma se activeaza.

- Cand alarma este declansata, microcontrollerul face mai multe actiuni in acelasi timp: LED-ul rosu palpaie, buzzerul suna prin PWM, LCD-ul afiseaza mesaj de alerta, iar prin Bluetooth se trimite un mesaj catre PC.

- Comunicarea dintre module se face prin mai multe tipuri de interfete: GPIO pentru LED-uri, buton si senzor PIR, PWM pentru buzzer, I2C pentru LCD si USART/UART pentru modulul Bluetooth.

Hardware Design

Lista componente

Componenta	Cantitate	Rol in proiect
ATmega328P XMINI	1	Microcontrollerul.
Senzor PIR HC-SR501	1	Detecteaza miscarea
LCD 1602 cu adaptor I2C	1	Afiseaza starea sistemului: dezarmat, armat sau alerta.
Modul Bluetooth HC-05	1	Trimite mesaje catre PC/laptop
Buzzer pasiv	1	Alerta sonora
LED verde 5mm	1	Este aprins cand alarma este dezarmata.
LED rosu 5mm	1	Este aprins cand alarma este armata si palpaie cand este detectata miscare.
Buton push-button	1	Folosit pentru armarea si dezarmarea sistemului.
Rezistori 1kΩ	5	Limiteaza curentul prin LED-uri si divizor de tensiune pt modul HC-05.
Breadboard	1	Folosit pentru montajul circuitului fara lipire.
Fire jumper tata-tata	mai multe	Conexiuni pe breadboard.
Fire jumper tata-mama	mai multe	Conexiuni intre module si placa/breadboard.
Cablu USB	1	Alimentare

Pini folositi

Pin ATmega328P	Eticheta	Componenta	Rol
PB0	LED_ROSU	LED rosu + rezistor 1kΩ	Iesire digitala - indica alarma armata sau alerta
PB1	LED_VERDE	LED verde + rezistor 1kΩ	Iesire digitala - indica alarma dezarmata
PB2	BUZZER_PWM	Modul buzzer pasiv	Iesire PWM - genereaza semnalul sonor al alarmei
PD2	PIR_OUT	Senzor PIR HC-SR501	Intrare digitala - detecteaza miscarea
PD3	BUTON	Buton push-button	Intrare digitala - armare/dezarmare alarma, folosind pull-up intern
PD0	BT_RX	Modul Bluetooth HC-05	Intrare USART - primeste date de la modulul Bluetooth
PD1	BT_TX	Modul Bluetooth HC-05	Iesire USART - trimite date catre HC-05 prin divizor de tensiune
PC4	SDA	LCD 1602 I2C	Linie de date I2C/TWI pentru LCD
PC5	SCL	LCD 1602 I2C	Linie de ceas I2C/TWI pentru LCD
VCC / +5V	VCC	Toate modulele	Magistrala pozitiva de alimentare, 5V de la placa
GND	GND	Toate modulele	Masa comuna a circuitului

Schema

Schema prezinta conexiunile dintre placa ATmega328P Xplained Mini si toate modulele folosite in proiect. Sunt incluse alimentarile comune la +5V si GND, conexiunile pentru LED-ul rosu si LED-ul verde prin rezistori de limitare a curentului, senzorul PIR conectat la un pin digital pentru detectarea miscarii, butonul folosit pentru armarea si dezarmarea sistemului cu pull-up intern, buzzerul pasiv controlat prin PWM, LCD-ul 1602 conectat prin interfata I2C pe liniile SDA si SCL, precum si modulul Bluetooth HC-05 conectat prin USART/UART pentru transmiterea mesajelor catre PC. In schema este reprezentat si divizorul de tensiune pentru pinul RXD al modulului HC-05, deoarece acesta lucreaza mai sigur cu un semnal logic de aproximativ 3.3V. Practic, schema arata cum microcontrollerul citeste intrarile de la buton si senzorul PIR, apoi controleaza iesirile sistemului: LED-uri, buzzer, LCD si comunicatia Bluetooth.

In aceasta poza se vede montajul initial pe breadboard, unde sunt conectate butonul, rezistentele si LED-urile folosite pentru indicarea starii sistemului.

Aceasta poza prezinta LCD-ul I2C montat si testat. Pe ecran este afisat mesajul “Salut! Functionez”, pentru a demonstra ca display-ul este conectat corect si comunica cu microcontrollerul. Se vad de asemenea si restul componentelor legate la placa.

In ultima poza se observa mesajele primite in monitorul serial prin modulul Bluetooth HC-05, folosit pentru transmiterea datelor de la sistem catre PC.

Software Design

Descrierea codului aplicatiei

Codul aplicatiei este scris in limbaj C pentru microcontrollerul ATmega328P de pe placa ATmega328P Xplained Mini. Proiectul lucreaza direct cu registrii microcontrollerului AVR. Astfel, configurarea pinilor, comunicatia USART, interfata TWI/I2C si timerul folosit pentru PWM sunt realizate manual, prin registri precum DDRB, PORTB, DDRD, PORTD, UCSR0B, UCSR0C, TWCR, TWBR, TCCR1A si TCCR1B.

Firmware-ul implementeaza un sistem de alarma cu trei stari principale: dezarmat, armat si alarma declansata. In starea dezarmat, LED-ul verde este aprins si senzorul PIR nu declanseaza alarma. In starea armat, LED-ul rosu este aprins continuu, iar sistemul monitorizeaza senzorul PIR. Daca este detectata miscare, sistemul trece in starea de alerta, LED-ul rosu incepe sa palpaie, buzzerul este activat intermitent prin PWM, LCD-ul afiseaza mesajul de alerta, iar prin Bluetooth se trimite o notificare catre PC sau telefon.

Codul este organizat pe functii separate pentru fiecare modul hardware: initializarea pinilor GPIO, comunicatia USART pentru Bluetooth, comunicatia TWI/I2C pentru LCD, controlul buzzerului prin PWM, citirea butonului, citirea senzorului PIR si logica principala a sistemului. Aceasta structura face codul mai usor de testat si de modificat, deoarece fiecare componenta poate fi verificata separat.

Mediu de dezvoltare

Pentru dezvoltarea firmware-ului am folosit Visual Studio Code impreuna cu PlatformIO. PlatformIO a fost folosit pentru compilarea si incarcarea codului pe placa ATmega328P Xplained Mini. Programarea placii se face prin interfata integrata a placii, folosind avrdude si protocolul xplainedmini.

Fisierul principal al aplicatiei este main.c, iar proiectul este configurat prin fisierul platformio.ini. Frecventa microcontrollerului este setata la 16 MHz, pentru ca functiile de delay, comunicatia USART si configurarea timerului pentru PWM sa fie calculate corect.

Librarii folosite

Pentru proiect nu am folosit librarii externe de tip Arduino sau biblioteci 3rd-party. Codul este scris direct in C pentru AVR, folosind doar headerele standard necesare pentru lucrul cu microcontrollerul:

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdint.h>
#include <string.h>
```

Biblioteca avr/io.h este folosita pentru acces direct la registrii microcontrollerului, cum ar fi DDRB,

PORTB, DDRD, UCSR0B, TWCR sau TCCR1A. Biblioteca util/delay.h este folosita pentru intarzieri simple, de exemplu la debounce-ul butonului si la initializarea LCD-ului. Am ales aceasta abordare pentru ca proiectul foloseste direct functionalitatile studiate la laborator, fara functii de nivel inalt.

Structura firmware-ului

Codul este impartit pe mai multe zone, fiecare corespunzand unei functionalitati hardware sau unei parti din logica sistemului:

- configurarea pinilor GPIO pentru LED-uri, buton, PIR si buzzer;
- comunicatia USART pentru modulul Bluetooth HC-05;
- comunicatia TWI/I2C pentru LCD-ul 1602;
- controlul buzzerului prin PWM folosind Timer1;
- logica de stare a sistemului de alarma;
- procesarea comenzilor primite prin Bluetooth;
- numararea miscarilor detectate de senzorul PIR.

Sistemul foloseste o masina simpla de stari, cu trei stari principale: dezarmat, armat si alarma declansata.

```
typedef enum {
    STARE_INIT = 255,
    STARE_DEZARMAT = 0,
    STARE_ARMAT,
    STARE_ALARMA
} AlarmState;

AlarmState state = STARE_INIT;
```

In starea dezarmat, LED-ul verde este aprins si alarma nu reactioneaza la miscarea detectata de PIR. In starea armat, LED-ul rosu este aprins si senzorul PIR este monitorizat. Daca este detectata miscare, sistemul trece in starea de alarma, unde LED-ul rosu palpaie, buzzerul suna intermitent, LCD-ul afiseaza alerta si se trimit mesaje prin Bluetooth.

Functionalitati implementate

Pentru pinii digitali am folosit direct registrii DDRx, PORTx si PINx. De exemplu, LED-urile si buzzerul sunt configurate ca iesiri, iar PIR-ul si butonul sunt configurate ca intrari:

```
DDRB |= (1 << LED_ROSU) | (1 << LED_VERDE) | (1 << BUZZER);

DDRD &= ~(1 << PIR_PIN);
PORTD &= ~(1 << PIR_PIN);

DDRD &= ~(1 << BUTTON);
```

```
PORTD |= (1 << BUTTON);
```

Butonul foloseste pull-up intern, deci cand nu este apasat pinul este HIGH, iar cand este apasat pinul este tras la GND si devine LOW. Pentru a evita citiri gresite din cauza contactului mecanic, am folosit un debounce simplu de 50 ms.

```
if (last_state == 1 && current_state == 0) {
    _delay_ms(50);

    if (!(PIND & (1 << BUTTON))) {
        last_state = 0;
        return 1;
    }
}
```

Comunicatia Bluetooth prin USART

Modulul Bluetooth HC-05 este conectat la USART-ul hardware al microcontrollerului. Comunicatia se face la 9600 baud, cu format 8N1: 8 biti de date, fara paritate si 1 bit de stop.

```
#define BAUD 9600
#define UBRR_VALUE ((F_CPU / 16 / BAUD) - 1)

void USART_init(void)
{
    UBRR0H = (uint8_t)(UBRR_VALUE >> 8);
    UBRR0L = (uint8_t)UBRR_VALUE;

    UCSR0B = (1 << TXEN0) | (1 << RXEN0);
    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
}
```

Prin Bluetooth sistemul trimite mesaje precum ARMAT, DEZARMAT, ALERTA sau numarul total de miscari detectate. De asemenea, sistemul poate primi comenzi simple, precum ARM, DISARM si STATUS.

Comunicatia cu LCD-ul prin I2C/TWI

LCD-ul 1602 foloseste un adaptor I2C cu adresa 0x27. Pentru comunicatie am folosit perifericul TWI al microcontrollerului. In timpul testarii, LCD-ul a functionat stabil cu frecventa I2C mai mica si cu pull-up intern activat pe liniile SDA si SCL.

```
void TWI_init(void)
{
```

```
    PORTC |= (1 << PC4) | (1 << PC5);

    TWSR = 0x00;
    TWBR = 152;
}
```

LCD-ul este folosit pentru afisarea starii curente a sistemului: sistem pornit, sistem dezarmat, sistem armat sau alerta de miscare. Functia folosita pentru afisarea mesajelor pe doua linii este:

```
void LCD_show_message(const char *line1, const char *line2)
{
    LCD_clear();

    LCD_set_cursor(0, 0);
    LCD_print(line1);

    LCD_set_cursor(1, 0);
    LCD_print(line2);
}
```

PWM pentru buzzer

Buzzerul este controlat prin PWM folosind Timer1. Semnalul este generat pe pinul PB2, care corespunde iesirii OC1B. Timer1 este configurat in modul Fast PWM, cu TOP in ICR1.

```
ICR1 = 999;
OCR1B = 100;

TCCR1A = (1 << COM1B1) | (1 << WGM11);
TCCR1B = (1 << WGM13) | (1 << WGM12) | (1 << CS11);
```

Valoarea ICR1 stabileste frecventa semnalului. Pentru F_CPU = 16 MHz, prescaler 8 si ICR1 = 999, frecventa este aproximativ 2 kHz. Valoarea OCR1B controleaza duty cycle-ul. In cazul proiectului, am folosit un duty cycle mai mic pentru ca buzzerul sa fie mai putin puternic si pentru a reduce incalzirea modulului.

Buzzerul este activ doar in starea de alarma si suna intermitent, nu continuu.

Algoritmi si structuri implementate

Logica principala a proiectului este bazata pe o masina de stari. Functia set_state() schimba starea sistemului si actualizeaza automat iesirile: LED-uri, buzzer, LCD si Bluetooth.

```
void set_state(AlarmState new_state)
{
```

```
    if (state == new_state)
        return;

    AlarmState old_state = state;
    state = new_state;

    if (state == STARE_DEZARMAT) {
        PORTB |= (1 << LED_VERDE);
        PORTB &= ~(1 << LED_ROSU);
        buzzer_stop();

        LCD_show_message("Sistem", "dezarmat");
        USART_send_string("DEZARMAT\r\n");
    }
}
```

Pentru senzorul PIR am implementat un counter de miscari. Sistemul nu numara de fiecare data cand pinul este HIGH, ci doar tranzitia LOW → HIGH. Astfel, aceeasi detectie nu este numarata de mai multe ori cat timp senzorul tine iesirea activa.

```
if (last_pir_state == 0 && current_pir_state == 1) {
    motion_count++;
}
```

Cand alarma este dezarmata dupa o alerta, sistemul trimite prin Bluetooth numarul total de miscari detectate.

Validarea functionalitatilor

Functionalitatile proiectului au fost validate incremental, testand mai intai fiecare modul separat si apoi integrarea lor in sistemul final.

Pentru LED-uri, am incarcat initial un program simplu care aprindea alternativ LED-ul rosu si LED-ul verde. Astfel am verificat ca pinii PB0 si PB1 sunt configurati corect ca iesiri si ca rezistorii/LED-urile sunt conectate corect.

Pentru buton, am testat separat citirea pinului PD3 cu pull-up intern. In stare neapasata pinul este HIGH, iar la apasare este conectat la GND si devine LOW. Validarea s-a facut prin schimbarea starii LED-urilor la fiecare apasare.

Pentru senzorul PIR, am folosit un test simplu in care LED-ul verde era aprins cand nu exista miscare, iar LED-ul rosu se aprindea cand senzorul detecta miscare. Astfel am verificat ca iesirea OUT a senzorului este citita corect pe PD2.

Pentru LCD, am testat comunicatia I2C separat. Initial LCD-ul afisa doar patrate, iar dupa ajustarea frecventei TWI si activarea pull-up-urilor interne pe SDA/SCL, afisarea a functionat corect. Adresa modulului LCD a fost confirmata ca fiind 0x27.

Pentru Bluetooth, am incarcat un program de test care trimitea periodic mesaje prin USART catre modulul HC-05. Mesajele au fost citite pe laptop/telefon printr-un terminal Bluetooth serial, confirmand functionarea comunicatiei la 9600 baud.

Pentru buzzer, am testat separat generarea semnalului PWM pe pinul PB2. Deoarece modulul folosit este low-level trigger, am ajustat functia de oprire astfel incat pinul sa fie pus pe HIGH atunci cand buzzerul trebuie oprit.

Dupa validarea individuala, modulele au fost integrate in firmware-ul final. Validarea sistemului complet s-a facut prin urmatorul scenariu: sistemul porneste dezarmat, butonul armeaza alarma, miscarea detectata de PIR declanseaza alerta, LCD-ul afiseaza mesajul de alarma, LED-ul rosu palpaie, buzzerul suna intermitent, iar prin Bluetooth se trimit mesaje despre starea sistemului si numarul de miscari detectate. La dezarmare, sistemul revine in starea initiala si trimite totalul miscarilor detectate.

Functionalitati din laborator folosite

Proiectul foloseste mai multe functionalitati studiate la laborator:

- GPIO: pentru LED-ul rosu, LED-ul verde, buton si senzorul PIR;
- pull-up intern: pentru citirea butonului fara rezistor extern;
- USART: pentru comunicatia cu modulul Bluetooth HC-05;
- TWI/I2C: pentru controlul LCD-ului 1602 prin adaptor I2C;
- Timer/PWM: pentru generarea semnalului sonor al buzzerului;

Aceasta abordare a fost aleasa pentru a demonstra folosirea directa a perifericelor microcontrollerului ATmega328P.

Stadiul actual al implementarii software

In stadiul actual, toate modulele principale au fost testate si integrate in firmware. LED-urile indica starea sistemului, butonul armeaza si dezarmeaza alarma, senzorul PIR detecteaza miscarea, LCD-ul afiseaza mesaje de stare, buzzerul se activeaza in starea de alarma, iar modulul Bluetooth transmite mesaje catre PC sau telefon.

Sistemul poate fi controlat local prin buton si poate transmite informatii prin Bluetooth. In plus, firmware-ul numara miscarile detectate in timpul unei alarme si trimite totalul atunci cand sistemul este dezarmat.

Elementul de noutate al proiectului

Elementul de noutate al proiectului este combinarea unei alarme locale cu raportare prin Bluetooth. Sistemul nu doar porneste un buzzer cand detecteaza miscare, ci afiseaza starea pe LCD, trimite

notificari prin Bluetooth si numara miscarile detectate cat timp alarma este activa. Astfel, utilizatorul poate vedea dupa dezarmare cate detectii au avut loc in timpul alarmei.

Calibrarea senzorului PIR

Senzorul PIR HC-SR501 a fost calibrat folosind potentiometrele de pe modul. Un potentiometru controleaza sensibilitatea senzorului, iar celalalt controleaza timpul cat iesirea ramane HIGH dupa detectarea miscarii. In timpul testarii, timpul de activare a fost setat relativ mic, pentru ca senzorul sa poata detecta mai multe miscari succesive.

A fost observat ca senzorul are un mic delay intre detectii. Acest comportament vine de la modulul PIR, nu de la cod. Pentru a evita numararea falsa a aceleiasi detectii, in firmware sunt numarate doar tranzitiile de la LOW la HIGH.

Optimizari realizate

In cod au fost realizate cateva optimizari simple pentru stabilitate:

- LCD-ul foloseste o frecventa I2C mai mica, deoarece la frecventa mai mare initializarea nu era stabila;
- liniile SDA si SCL au pull-up intern activat;
- buzzerul este activat intermitent, nu continuu, pentru a reduce consumul si incalzirea modulului;
- mesajele pe LCD sunt actualizate doar la schimbarea starii, nu in fiecare iteratie a buclei principale;
- miscarile PIR sunt numarate doar pe front crescator, pentru a evita cresterea artificiala a counterului.

Aceste optimizari au fost facute dupa testarea individuala a modulelor pe breadboard.

DEMO VIDEO

In video prezint cum pot arma sistemul de alarma pe bluetooth, apoi senzorul detecteaza miscare, porneste buzzer-ul si incepe sa palpaie led-ul rosu. Apoi opresc de la buton alarma, armez de la buton, o las sa sune si apoi o dezarmeaz prin bluetooth, unde se poate observa si ca inregistreaza numarul de miscari.

Link: https://youtube.com/shorts/7qR14ZF2nFI?is=Ke8virHbPhahAdF4

Download

GitHub: https://github.com/Fane600/Proiect_PM

Bibliografie/Resurse

- Laborator PM 1 - GPIO
- Laborator PM 2 - intrari digitale / intreruperi
- Laborator PM 3 - Timere si PWM
- Laborator PM 6 - I2C/TWI
- ATmega328P Datasheet
- PlatformIO Atmel AVR Documentation
- Documentatii/module folosite: HC-SR501 PIR, HC-05 Bluetooth, LCD 1602 I2C.