TimeLock Box

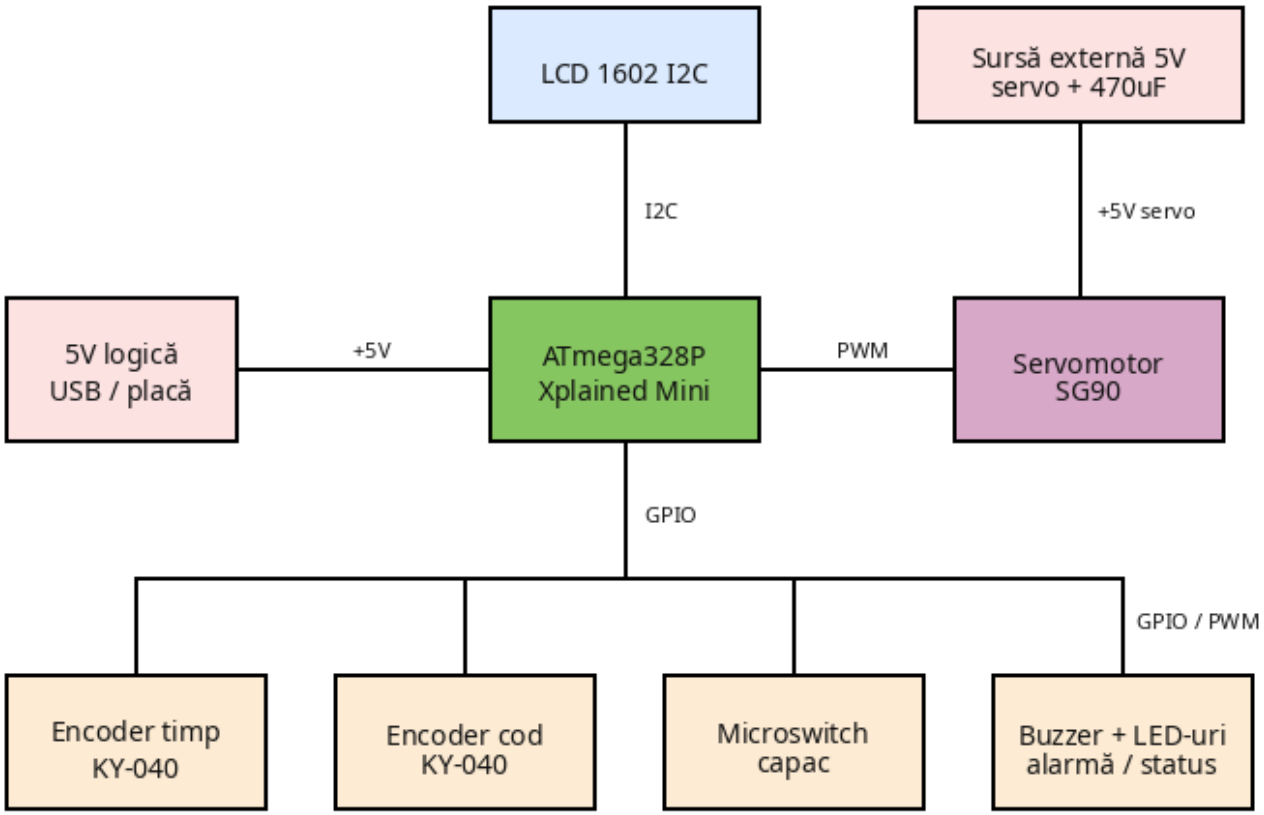
Introducere

Proiectul constă într-o cutie inteligentă cu închidere mecanică, destinată depozitării telefonului sau a altor obiecte în timpul unei sesiuni de lucru. Utilizatorul selectează o durată de blocare, iar dispozitivul blochează capacul folosind un servomotor. Deblocarea se poate face doar după expirarea timpului setat sau prin introducerea unei combinații rotative.

Proiectul este gândit ca un sistem fizic de limitare a accesului la obiecte care pot distrage atenția, în special telefonul. Spre deosebire de o aplicație software de productivitate, TimeLock Box obligă utilizatorul să își respecte intervalul de lucru printr-o blocare mecanică reală. Sistemul afișează starea curentă pe un LCD și semnalizează vizual sau acustic evenimentele importante.

Descriere generală

Componentele principale ale proiectului sunt vizibile în schema bloc de mai jos. Placa **ATmega328P Xplained Mini** este blocul central al sistemului: citește intrările utilizatorului, actualizează afișajul și comandă mecanismul de blocare.



Encoderul de timp este folosit pentru alegerea duratei de blocare, iar encoderul de cod este folosit pentru introducerea combinației de deblocare. LCD-ul comunică prin I2C și afișează starea sistemului. Servomotorul SG90 este comandat prin PWM și acționează zăvorul mecanic. Microswitch-ul detectează ridicarea capacului, iar buzzerul și LED-urile oferă feedback acustic și vizual. Servo-ul este alimentat separat la 5V, cu masă comună cu restul sistemului.

Hardware Design

Schema electrică a proiectului este realizată în KiCad și este atașată mai jos. Pentru lizibilitate, conexiunile sunt grupate prin etichete de semnal, iar alimentarea servo-ului este separată de alimentarea logică.

Listă de componente

Componentă	Cantitate	Rol în proiect
ATmega328P Xplained Mini	1	control principal
Encoder rotativ KY-040 pentru timp	1	setare durată
Encoder rotativ KY-040 pentru cod	1	introducere cod

LCD 1602 cu interfață I2C	1	afișare status
Servomotor SG90	1	acționare zăvor
Buzzer pasiv 5V	1	alarmă acustică
Microswitch cu levier (Panasonic)	1	detecție capac
LED roșu	1	indicator alarmă / blocat
LED verde	1	indicator OK / deschis
Rezistențe 330Ω	2	limitare curent LED
Condensator electrolitic 470uF / 16V	1	stabilizare servo
Breadboard 830 puncte	1	prototipare
Fire jumper tată-tată și mamă-tată	mai multe	conexiuni
Sursă externă 5V pentru servo	1	alimentare servo

Conexiuni și justificarea pinilor

Modul	Conexiuni	Motiv
Encoder cod KY-040	CLK - PD2; DT - PD3; SW - PD4; VCC → +5V; GND → GND	Interfață dedicată pentru setarea și introducerea codului.
Encoder timp KY-040	CLK - PD5; DT - PD6; SW - PD7; VCC → +5V; GND → GND	Interfață dedicată pentru alegerea duratei de blocare.
LCD 1602 I2C	SDA - PC4; SCL - PC5; VCC → +5V; GND → GND	Afișează meniul, codul și timpul rămas.
Servomotor SG90	Semnal - PB1; VCC → +5V servo; GND → GND comun	Acționează zăvorul mecanic.
Buzzer pasiv	+ → PB0; - → GND	Feedback sonor și alarmă.
Microswitch capac	NO - PB2; C - GND	Detectează deschiderea capacului.
LED roșu	anod - PC0; catod → rezistență 330Ω → GND	Indică seif blocat / alarmă.
LED verde	anod - PC1; catod → rezistență 330Ω → GND	Indică seif deschis / OK.

Alimentare

Linie alimentare	Componente alimentate	Observații
+5V logică	ATmega328P Xplained Mini, LCD 1602 I2C, encodere KY-040	Pentru componente cu consum redus.
+5V servo	Servomotor SG90	Alimentare separată pentru vârfurile de curent.
GND comun	Toate modulele	Leagă sursa servo de masa plăcii.
Condensator 470uF / 16V	Între +5V servo și GND	Stabilizează alimentarea servo-ului.

Sumar pini

Pin ATmega328P	Funcție în proiect
PD2	CLK encoder cod
PD3	DT encoder cod
PD4	SW encoder cod (Buton)

PD5	CLK encoder timp
PD6	DT encoder timp
PD7	SW encoder timp (Buton)
PB0	Buzzer pasiv (Alarmă/Confirmare)
PB1	Semnal servomotor SG90 (Zăvor)
PB2	Intrare senzor ușă (Microswitch Panasonic)
PC0	Ieșire LED roșu
PC1	Ieșire LED verde
PC4	Date I2C pentru ecran LCD
PC5	Ceas I2C pentru ecran LCD
PD0, PD1	UART pentru debug, lăsați liberi

Dovezi de funcționare

În această secțiune voi încărca video-ul realizat în laborator:

- montajul general pe breadboard;
- testul pentru buzzer și LED roșu;
- testul pentru LCD și LED-ul verde.

Software Design

Mediu de dezvoltare

Firmware-ul este dezvoltat în **PlatformIO**, folosind framework-ul **Arduino** pentru microcontrollerul **ATmega328P Xplained Mini**. Configurația proiectului folosește platforma atmelavr, placa custom atmega328p_xplained_mini și frecvența de lucru de 16 MHz.

Biblioteci folosite

- Arduino.h - funcționalități de bază Arduino: setup(), loop(), millis(), digitalRead(), pinMode(), tone().
- Wire.h - comunicația I2C/TWI folosită de LCD.
- LiquidCrystal_I2C.h - afișarea meniurilor, codului și timpului rămas pe LCD 1602.
- Servo.h - generarea semnalului PWM pentru servomotorul SG90 care acționează zăvorul.
- avr/io.h - acces direct la registre AVR pentru LED-uri, prin DDRC și PORTC.

Element de noutate

Proiectul implementează un **TimeLock Box** cu două encodere rotative separate: unul pentru alegerea timpului de blocare și unul pentru setarea/introducerea codului. Codul de deblocare este setat direct din hardware la începutul sesiunii, fără a fi recompilat firmware-ul și fără cod de deblocare hardcodat în firmware. În plus, cutia are un sistem de alarmă folosind microswitch-ul, care detectează deschiderea capacului cât timp seiful este blocat.

Funcționalități din laboratoare

- **GPIO** - citire butoane encoder, microswitch capac, comandă LED-uri.
- **Întreruperi externe** - citirea encoderului pentru cod folosind pinii PD2 și PD3.
- **Timere / PWM** - semnalul pentru servomotor și semnalele sonore pentru buzzer.
- **I2C / TWI** - comunicația cu LCD-ul 1602 prin adaptorul I2C.
- **USART** - serialul este folosit pentru debug în timpul testării.

Structura aplicației

Logica este organizată ca o mașină de stări:

```
enum SafeState {
  SET_CODE,
  SET_TIME,
  LOCKED,
  ENTER_CODE,
  ALARM
};
```

Stările principale sunt:

- SET_CODE - utilizatorul alege codul de sesiune folosind encoderul de cod;
- SET_TIME - utilizatorul alege durata de blocare folosind encoderul de timp;
- LOCKED - servo-ul ține zăvorul închis, LED-ul roșu este aprins și cronometrul scade;
- ENTER_CODE - utilizatorul introduce codul pentru deblocare sau pentru oprirea alarmei;
- ALARM - microswitch-ul a detectat capacul ridicat cât timp seiful era blocat.

Pe lângă aceste stări există variabile de control pentru starea fizică a seifului:

```
bool safe_locked = false;
bool alarm_active = false;
long remaining_seconds = 0;
int session_code[3] = {0, 0, 0};
```

Algoritm

În funcția setup() se inițializează LCD-ul, encoderele, microswitch-ul, buzzerul, LED-urile, servomotorul

și comunicația serială. LED-ul verde este aprins la pornire, iar servo-ul este poziționat în starea de zăvor deschis.

În funcția loop() se execută următorii pași:

- se verifică microswitch-ul; dacă seiful este blocat și capacul este ridicat, se activează alarma;
- în starea de alarmă, codul poate fi introdus doar după ce capacul revine închis;
- dacă seiful este blocat, cronometrul este actualizat cu millis(), fără delay() pentru scurgerea timpului;
- timpul continuă să scadă inclusiv în starea ENTER_CODE;
- encoderul de timp modifică durata în trepte de 5 minute;
- encoderul de cod modifică cifra curentă între 0 și 9;
- apăsarea butonului encoderului de cod confirmă cifra curentă;
- la introducerea celor 3 cifre, codul este comparat cu session_code;
- dacă acest cod este corect, servo-ul retrage zăvorul, LED-ul verde se aprinde și sesiunea se resetează;
- dacă timpul expiră natural, seiful se deblochează automat și revine la setarea unui cod nou.

Pentru a reduce problemele de bouncing și citirile greșite ale encoderelor, afișajul LCD nu este redesenat continuu. Se folosește un flag de refresh, astfel încât LCD-ul este actualizat doar când se schimbă o valoare sau o stare:

```
bool screen_dirty = true;

// exemplu: după rotirea encoderului
current_digit++;
screen_dirty = true;
```

Repartizare pini în firmware

```
#define CODE_CLK_PIN 2    // PD2
#define CODE_DT_PIN 3     // PD3
#define CODE_SW_PIN 4     // PD4

#define TIME_CLK_PIN 5    // PD5
#define TIME_DT_PIN 6     // PD6
#define TIME_SW_PIN 7     // PD7

#define BUZZER_PIN 8       // PB0
#define SERVO_PIN 9        // PB1
#define DOOR_SWITCH_PIN 10 // PB2
```

LED-urile sunt controlate direct prin registre:

```
DDRC |= (1 << PC0) | (1 << PC1); // PC0 roșu, PC1 verde

void stareDeschis_LED() {
  PORTC |= (1 << PC1);
```

```
  PORTC &= ~(1 << PC0);
}

void stareInchis_LED() {
  PORTC |= (1 << PC0);
  PORTC &= ~(1 << PC1);
}
```

GPIO și întreruperi

Microswitch-ul și butoanele encoderelor sunt configurate cu rezistențele interne de pull-up. Astfel, apăsarea butonului sau închiderea microswitch-ului produce nivel logic LOW.

```
pinMode(DOOR_SWITCH_PIN, INPUT_PULLUP);
pinMode(CODE_SW_PIN, INPUT_PULLUP);
pinMode(TIME_SW_PIN, INPUT_PULLUP);
```

Encoderul pentru cod este citit prin întreruperi externe pe ambele linii (CLK și DT). Tranzițiile sunt decodate cu un tabel de cuadratură, ceea ce reduce citirile inverse produse de bouncing.

```
attachInterrupt(digitalPinToInterrupt(CODE_CLK_PIN), code_encoder_interrupt,
CHANGE);
attachInterrupt(digitalPinToInterrupt(CODE_DT_PIN), code_encoder_interrupt,
CHANGE);
```

PWM, servo și buzzer

Servomotorul SG90 este controlat prin biblioteca Servo.h. În implementarea curentă, unghiul 0 reprezintă zăvorul retras, iar unghiul 90 reprezintă zăvorul coborât.

```
Servo lockServo;

lockServo.attach(SERVO_PIN);
lockServo.write(0);   // zăvor deschis
lockServo.write(90);  // zăvor închis
```

Buzzerul este folosit pentru confirmări scurte, erori și alarmă:

```
void sunaBuzzer(int durata_ms) {
  tone(BUZZER_PIN, 1000);
  delay(durata_ms);
  noTone(BUZZER_PIN);
}
```

I2C

LCD-ul 1602 este inițializat la adresa 0x27 și este folosit pentru afișarea meniului, codului și timpului rămas.

```
LiquidCrystal_I2C lcd(0x27, 16, 2);

lcd.init();
lcd.backlight();
lcd.clear();
```

Timpul rămas este afișat în format HHh:MMm:SSs, pentru a putea reprezenta inclusiv sesiuni mai lungi de o oră.

USART

Serialul este folosit pentru debug în timpul dezvoltării și testării:

```
Serial.begin(115200);
```

Calibrări

- Encoderele KY-040 sunt citite pe front descendent și au debounce software scurt, bazat pe millis().
- LCD-ul este actualizat doar la schimbări de stare sau valoare, pentru a evita lag-ul care poate produce citiri greșite ale encoderelor.
- Pentru encoderul de timp, fiecare pas modifică durata cu 5 minute.
- Butoanele encoderelor au un mic delay de confirmare pentru a evita apăsările multiple.
- Unghiurile servo-ului (0 și 90) pot fi ajustate în funcție de zăvorul mecanic real.
- Servo-ul este alimentat separat la 5V, iar GND-ul sursei servo este comun cu GND-ul plăcii.
- Dacă bouncing-ul mecanic persistă, se pot adăuga condensatori de 100nF între CLK și GND, respectiv între DT și GND pentru fiecare encoder.

Rezultate Obţinute

Am obținut o cutie care se blochează pe un timp ales din encoder și se deschide doar cu codul setat sau după ce expiră timpul. Pe LCD se afișează timpul rămas, iar dacă ușa este forțată pornește alarma.

Concluzii

Proiectul funcționează cum mi-am propus. Partea cea mai dificilă a fost legarea și calibrarea componentelor, mai ales encoderele și microswitch-ul.

Download

GitHub

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

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

Export to PDF