Sistem Automat de Administrare a Medicamentelor

Introducere

Acest proiect reprezinta un sistem automat de administrare a medicamentelor. La ora programata, sistemul suna o alarma, roteste o roata compartimentata pentru a elibera pastilele zilei si umple un pahar cu apa folosind o pompa submersibila. Un ecran LCD afiseaza in permanenta ziua si ora curenta.

Scopul proiectului este de a automatiza complet procesul de administrare a medicatiei zilnice, eliminand necesitatea unei persoane care sa supravegheze sau sa reaminteasca acest lucru.

Ideea a pornit de la o situatie reala din familia mea. Bunicul meu ia medicamente zilnic si are nevoie ca cineva sa ii aminteasca sa le ia, sa ii pregateasca pastilele si sa stie ce zi este. Acest proiect incearca sa rezolve exact aceasta problema printr-un dispozitiv simplu de folosit si usor de inteles.

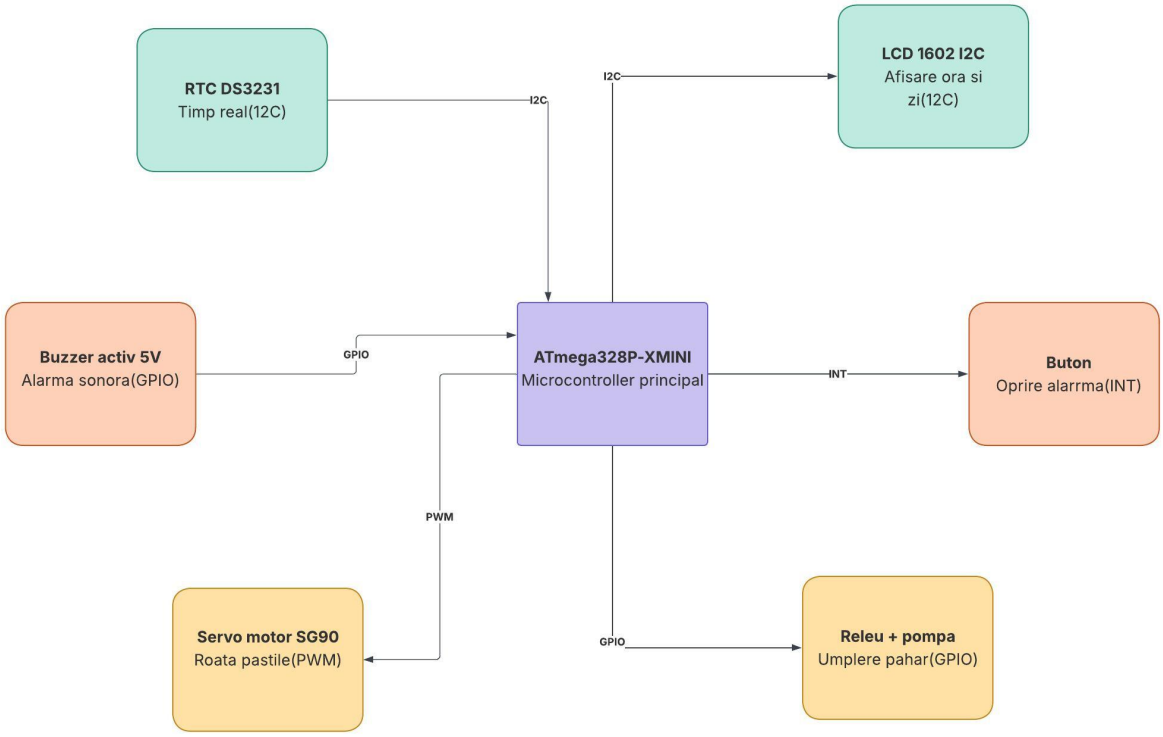
Consider ca un astfel de sistem poate fi util atat pentru persoanele in varsta care iau tratamente zilnice, cat si pentru familiile acestora, care nu mai trebuie sa fie prezente fizic pentru a se asigura ca medicatia a fost administrata corect si la timp.

Descriere generala

Sistemul este compus din urmatoarele module hardware:

- **ATmega328P-XMINI** - microcontrollerul principal care coordoneaza toate modulele
- **RTC DS3231** - retine ora si data exacta, comunicatie prin I2C
- **LCD 1602 I2C** - afiseaza ora si ziua curenta, comunicatie prin I2C
- **Servo motor SG90** - roteste roata de pastile la ora programata, controlat prin PWM
- **Releu + pompa submersibila** - umple paharul cu apa, controlata printr-un pin digital
- **Buzzer activ 5V** - suna alarma la ora programata
- **Buton** - opreste alarma prin intrerupere hardware (INT0)
- **Condensator electrolitic** - filtraj alimentare

La ora programata, RTC-ul semnalizeaza microcontrollerul prin I2C. Buzzerul porneste si LCD-ul afiseaza ora si ziua si mesaje de ghidare. Utilizatorul apasa butonul (intrerupere hardware) pentru a opri alarma. Simultan, servo-ul roteste roata de pastile, iar releul porneste pompa pentru un interval fix de timp pana cand paharul este plin.



Hardware Design

Stadiul actual al implementarii hardware:

Toate componentele au fost conectate pe breadboard si testate individual. Sistemul complet functioneaza: RTC-ul mentine ora corecta, LCD-ul afiseaza data si ora in timp real, buzzerul se activeaza la fiecare minut nou, butonul opreste alarma prin intrerupere hardware, servo-ul roteste roata de pastile, iar pompa umple paharul cu apa timp de 2 secunde.

Lista de componente si rolul lor:

Componenta	Rol in proiect
ATmega328P-XMINI	Microcontroller principal, coordoneaza toate modulele
Modul RTC DS3231	Retine ora si data exacta independent de alimentare
LCD 1602 cu adaptor I2C	Afiseaza ziua saptamanii, data si ora in timp real
Servo motor SG90	Roteste roata compartimentata pentru eliberarea pastilelor
Modul releu 5V cu optocuplor	Izoleaza si controleaza pompa submersibila
Pompa submersibila 3-6V	Umple paharul cu apa dupa eliberarea pastilei
Buzzer activ 5V	Semnalizeaza sonor momentul administrarii medicamentului
Modul buton microswitch	Permite utilizatorului sa confirme luarea medicamentului
Condensator electrolitic 100µF / 10V	Filtreaza zgomotul electric pe linia de alimentare
Breadboard 830 puncte	Suport pentru prototipare si interconectare
Set fire jumper	Realizarea conexiunilor intre componente

Descrierea pinilor utilizati si motivatia alegerii lor:

PC4 (SDA) si PC5 (SCL) — bus I2C: Acestia sunt singurii pini hardware I2C (TWI) ai ATmega328P.

RTC DS3231 si LCD 1602 I2C sunt conectate amandoua pe acelasi bus, diferentiate prin adrese distincte (RTC: 0x68, LCD: 0x27). Folosirea perifericului hardware TWI permite comunicatie stabila la 100kHz fara a ocupa resurse software.

PB1 (OC1A) — semnal PWM servo: PB1 este iesirea comparatorului A al Timer1, singurul pin care poate genera semnalul PWM de 50Hz necesar servo-ului SG90 prin hardware. Folosirea Timer1 (16-bit) permite reglaj precis al latimii pulsului intre 1ms si 2.4ms.

PD2 (INT0) — buton: PD2 este pinul intreruperii externe INT0, cea mai rapida metoda de a detecta apasarea butonului fara polling continuu. Intreruperea este configurata pe front descendent, cu pull-up intern activat.

PD6 — buzzer: Pin digital de iesire standard, ales din portul D disponibil. Buzzerul activ nu necesita semnal PWM, doar nivel logic HIGH/LOW.

PD7 — releu: Pin digital de iesire standard pentru controlul releului. Releul cu optocuplor izoleaza circuitul pompei (3-6V) de circuitul logicii (5V), protejand microcontrollerul.

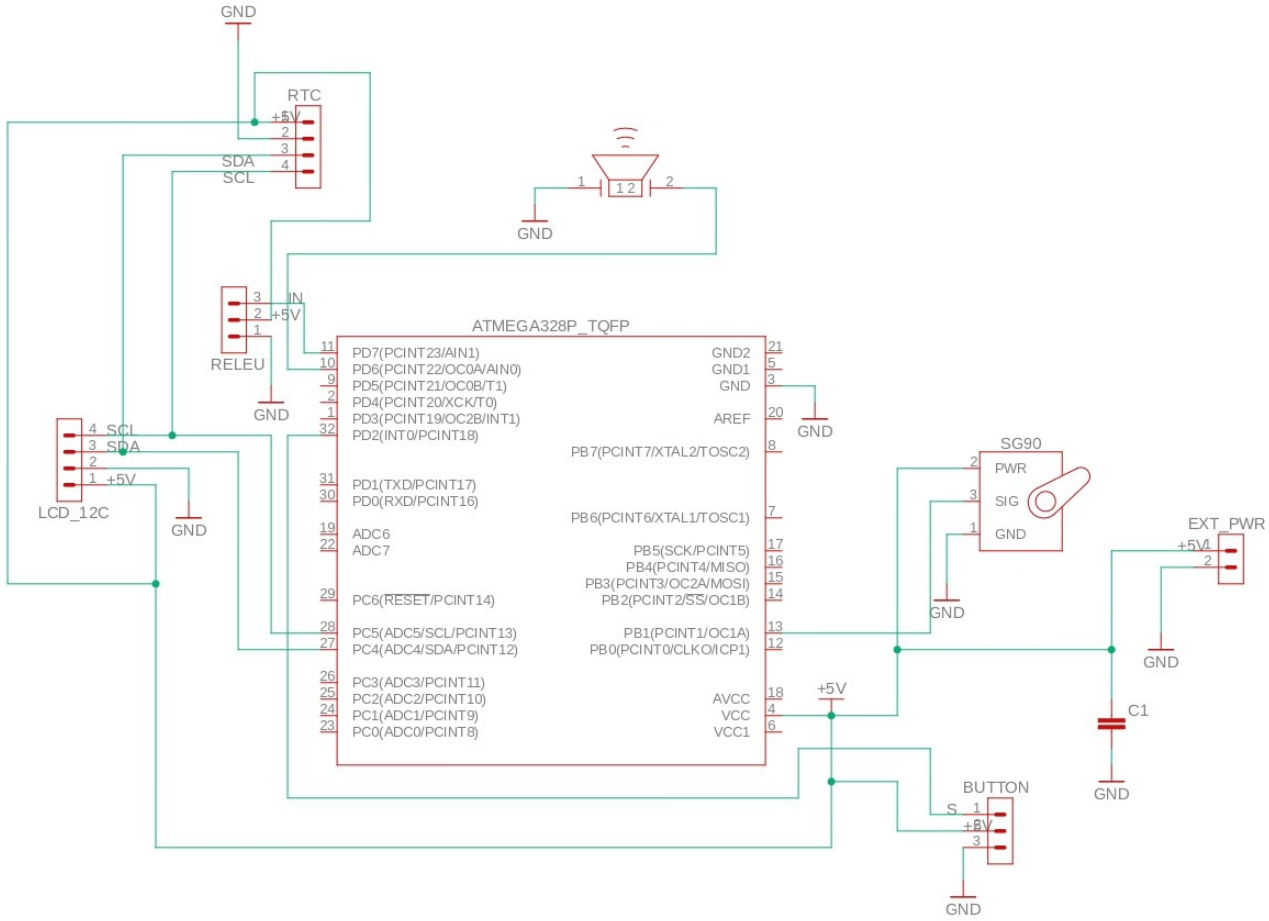
Conexiuni principale:

Componenta	Pin componenta	Pin ATmega
RTC DS3231	VCC	5V
RTC DS3231	GND	GND
RTC DS3231	SDA	PC4 / A4
RTC DS3231	SCL	PC5 / A5
LCD 1602 I2C	VCC	5V
LCD 1602 I2C	GND	GND
LCD 1602 I2C	SDA	PC4 / A4
LCD 1602 I2C	SCL	PC5 / A5
Servo SG90	VCC (rosu)	Sursa externa 5V
Servo SG90	GND (maro)	GND
Servo SG90	Signal (portocaliu)	PB1 / D9
Modul releu	DC+	5V
Modul releu	DC-	GND
Modul releu	IN	PD7 / D7
Buzzer activ	+	PD6 / D6
Buzzer activ	-	GND
Buton	VCC	5V
Buton	GND	GND
Buton	S	PD2 / D2 (INT0)
Pompa	+	Releu contact NO
Pompa	-	GND
Condensator 100µF	+ (anod)	5V
Condensator 100µF	- (catod)	GND

Obs: SDA si SCL sunt partajate intre RTC si LCD pe acelasi bus I2C. Servo-ul este alimentat dintr-o sursa externa de 5V, cu GND comun cu breadboard-ul. Condensatorul de 100µF este plasat pe linia de alimentare pentru filtrarea zgomotului electric.

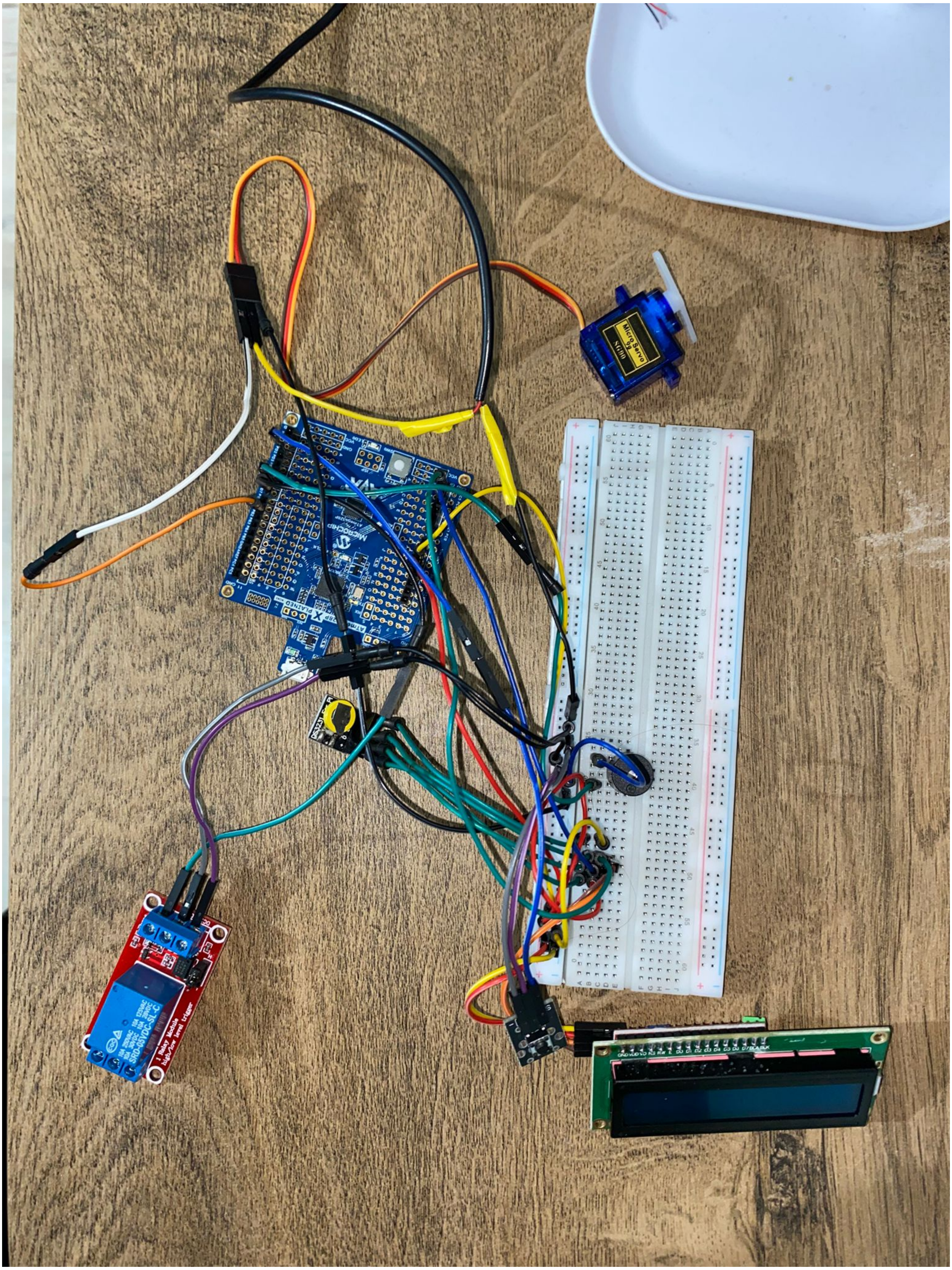
Schema electronica (Fusion 360):

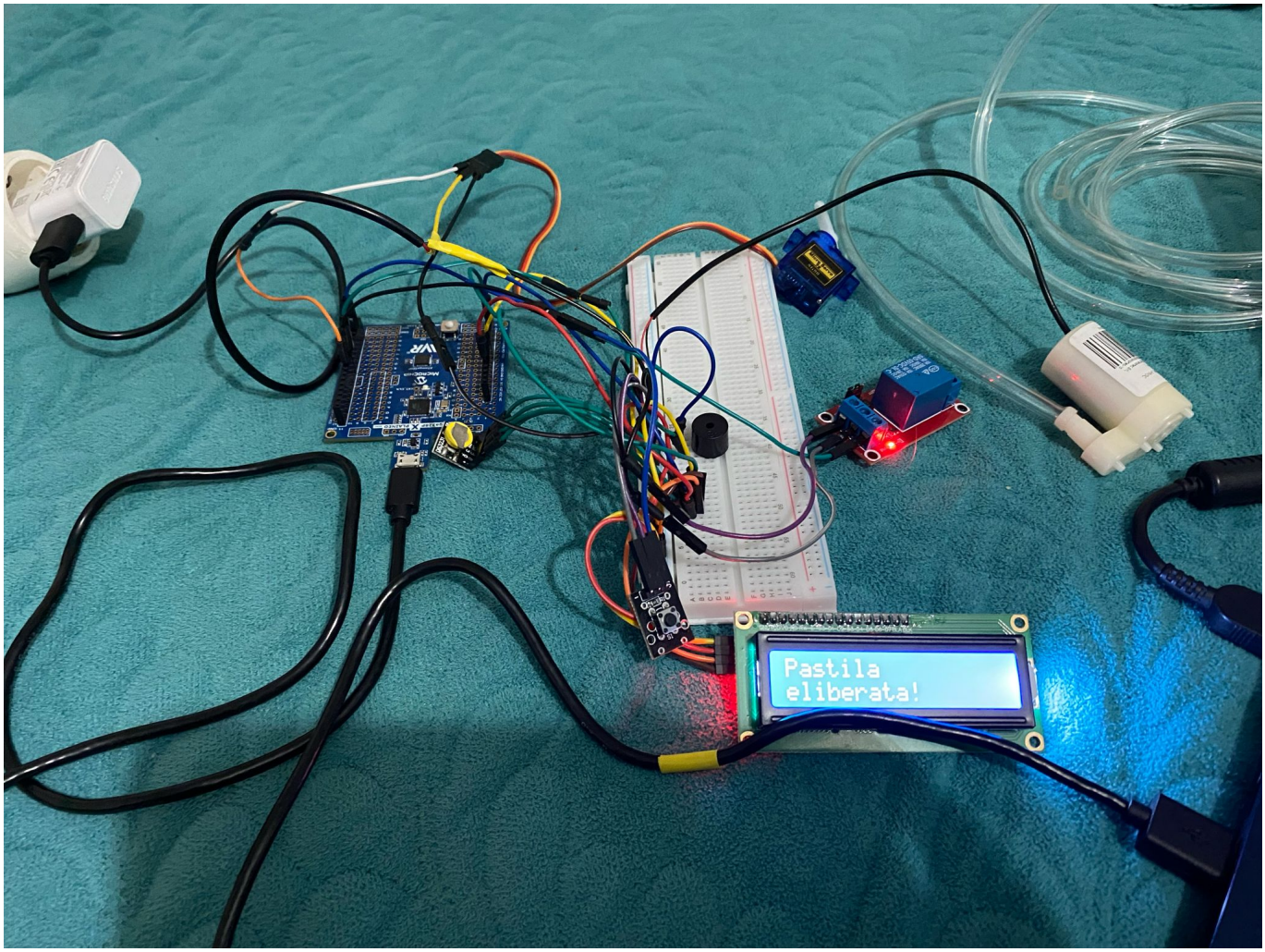
In schema electronica realizata in Fusion 360, nu toate componentele proiectului au fost disponibile in libraria standard. Pentru modulele care nu au fost gasite (RTC DS3231, LCD 1602 I2C, modul releu, buzzer, buton), au fost folosite conectoare generice cu numarul exact de pini corespunzator fiecarui modul, etichetate cu numele componentei respective. Conexiunile electrice sunt identice cu cele din implementarea fizica.



Imagini hardware si dovezi de functionare:

Imaginile de mai jos prezinta componentele conectate pe breadboard si sistemul in timpul testarii. LCD-ul afiseaza ora in timp real citita din RTC si mesaje de succes, confirmand functionarea corecta a comunicatiei I2C si a driver-ului de afisare.


Micro Servo
MICROCHIP
SRD-05VDC-SL-C



Software Design

1. Stadiul actual

Firmware-ul este complet implementat si testat. Toate functionalitatile proiectului sunt operationale: citirea orei din RTC, afisarea pe LCD, declansarea alarmei la fiecare minut nou, oprirea alarmei prin buton, rotirea servo-ului si activarea pompei prin releu.

2. Mediu de dezvoltare

Proiectul a fost dezvoltat in **Visual Studio Code** cu extensia **PlatformIO IDE**. Compilarea se face cu toolchain-ul **avr-gcc**, iar incarcarea pe microcontroller prin **avrdude** cu protocolul **xplainedmini** (ATmega328P-XMINI conectat prin USB).

Configuratia platformei (platformio.ini):

- platform: atmelavr

- board: ATmega328P
- framework: arduino (doar pentru definirea board-ului, codul este C pur AVR)
- upload_protocol: xplainedmini prin USB

3. Librarii si surse - motivatia alegerii

Nu au fost folosite librarii third-party. Intreg firmware-ul este scris in C, folosind doar headerele standard AVR:

- `avr/io.h` - definitii registre si pini, necesara pentru orice acces la hardware
- `avr/interrupt.h` - macro-uri pentru intreruperi (ISR, sei, cli), folosita pentru Timer2 si INT0
- `util/delay.h` - delay hardware precis la nivel de microsecunde, folosit in secventele de initializare LCD unde timpul este critic
- `stdint.h` - tipuri intregi cu dimensiune fixa (uint8_t, uint32_t), asigura portabilitate si claritate

Implementarea manuala a driverelor I2C, LCD si RTC a permis intelegerea completa a protocoalelor si eliminarea oricarui overhead nedorit pe un microcontroller cu resurse limitate (2KB RAM, 32KB Flash).

4. Elementul de noutate

Elementul principal de noutate al proiectului este **integrarea intr-un singur sistem autonom a patru actiuni coordonate**: detectia timpului, alerta sonora si vizuala, eliberarea mecanica a pastilei prin servo si umplerea paharului cu apa prin pompa - toate declansate printr-o singura apasare de buton.

Un al doilea element de noutate este **implementarea unui driver LCD complet in mod 4-bit prin I2C fara nicio librarie externa**, inclusiv secventa de initializare cu nibble-uri conform datasheet-ului HD44780.

5. Justificarea functionalitatilor din laborator

Functionalitate din laborator	Utilizare in proiect
Intreruperi hardware (INT0)	Detectia apasarii butonului fara polling, raspuns imediat
Timer in mod CTC (Timer2)	Generare baza de timp la 1ms pentru delay_ms() non-blocking
Timer in mod Fast PWM (Timer1)	Generare semnal PWM 50Hz pentru controlul servo-ului SG90
Comunicatie I2C (TWI)	Comunicare cu RTC DS3231 si LCD 1602 pe acelasi bus
Registre DDR / PORT	Configurare pini intrare/iesire pentru buzzer, releu, buton
Pull-up intern	Stabilizare semnal buton pe PD2 fara rezistor extern

6. Scheletul proiectului si interactiunea dintre functionalitati

Firmware-ul este structurat intr-un singur fisier `src/main.cpp` cu urmatorul flux:

Initializare (rulata o singura data la pornire): `twi_init`, `timer2_init`, `servo_init`, `buton_init`, `sei`, `lcd_init`, pozitionare servo la 0 grade.

Bucla principala (`while(1)`):

- Se citeste ora din RTC prin I2C la fiecare iteratie
- Daca alarma nu este activa, se afiseaza ora pe LCD
- Se verifica conditia de alarma (sec==0 si minut nou)
- Daca alarma este activa, se porneste buzzerul si se afiseaza mesajul
- Daca butonul a fost apasat (flag setat de ISR), se ruleaza secventa completa: servo → mesaj LCD → pompa → mesaj finalizare
- Delay de 200ms intre iteratii pentru stabilitate

Interactiunea dintre module:

- **Timer2 → delay_ms()** — folosit pentru toate asteptarile din secventa dozator
- **INT0 → butonApasat** — flag citit in bucla principala, nu direct in ISR
- **TWI → RTC** — ora citita este evaluata pentru logica alarmei
- **TWI → LCD** — afisare stare curenta in functie de flagul `alarmaActiva`
- **Timer1 → Servo** — pozitie incrementata ciclic la fiecare confirmare

Validarea functionalitatilor

- LCD-ul a fost testat afisand un mesaj fix inainte de integrarea RTC-ului
- RTC-ul a fost validat verificand ca ora afisata pe LCD coincide cu ora reala
- Servo-ul a fost calibrat manual (descris la sectiunea 8)
- Butonul a fost testat verificand ca ISR-ul seteaza corect flag-ul
- Pompa a fost testata activand releul direct din cod cu un delay fix

7. Demo video

Demo video proiect

Videoclipul prezinta ciclul complet de functionare: afisarea orei pe LCD, declansarea alarmei la 30 secunde(pentru test), afisarea mesajelor cu indicatii, apasarea butonului, rotirea servo-ului, umplerea paharului si afisarea mesajului de finalizare.

8. Calibrarea servo-ului SG90

Servo-ul SG90 are specificatii nominale de 1ms-2ms pentru 0-180 grade, dar in practica valorile difera. Calibrarea s-a realizat empiric:

- S-a pornit cu valorile teoretice: OCR1A = 2000 (1ms) si OCR1A = 4000 (2ms)
- S-a observat ca servo-ul nu ajungea la capetele cursei
- S-au ajustat valorile pana la OCR1A = 1088 pentru 0 grade si OCR1A = 4800 pentru 180 grade
- Aceste valori corespund unui puls intre ~0.544ms si ~2.4ms, specific exemplarului fizic folosit
- Pasul de 50 grade a fost ales astfel incat roata cu compartimente sa fie rotita corect la fiecare doza (4 x 50 grade = pozitii distincte in intervalul 0-150 grade)

9. Optimizari realizate

- **delay_ms() non-blocking bazat pe Timer2** - in loc de `_delay_ms()` care blocheaza complet CPU-ul si dezactiveaza raspunsul la intreruperi pentru durate lungi, s-a implementat o functie proprie bazata pe timestamp. Aceasta permite Timer2 si INT0 sa functioneze corect in timpul asteptarilor.
- **Flag butonApasat procesat in bucla, nu in ISR** - ISR-ul face doar `butonApasat = 1` si returneaza imediat, minimizand timpul petrecut in intrerupere. Logica complexa (servo, pompa, LCD) este executata in bucla principala unde contextul este sigur.
- **ultimulMinutAlarma previne re-declansarea** - fara acest flag, alarma s-ar redeclansa la fiecare iteratie a buclei cat timp sec==0 (aproximativ 1 secunda). Flag-ul asigura o singura declansare per minut.
- **Actualizare LCD doar cand e necesar** - LCD-ul este actualizat cu ora doar daca `alarmaActiva == 0`, evitand scrieri inutile pe I2C in timpul secventei de alarma.

10. Functii implementate

Functie	Descriere
`servo_init()`	Configureaza Timer1 pentru PWM pe PB1
`servo_scrie(int unghi)`	Calculeaza si scrie OCR1A pentru unghiul dorit
`timer2_init()`	Configureaza Timer2 CTC la 1ms
`ISR(TIMER2_COMPA_vect)`	Incrementeaza contorul de milisecunde
`uptime_ms()`	Citire atomica a variabilei milisecunde
`delay_ms(uint32_t ms)`	Delay bazat pe timestamp, non-blocking
`buton_init()`	Configureaza PD2 cu pull-up si INT0
`ISR(INT0_vect)`	Seteaza flag butonApasat la apasare
`twi_init()`	Initializeaza modulul TWI la 100kHz
`twi_start()`	Trimite conditie START pe bus I2C
`twi_stop()`	Trimite conditie STOP pe bus I2C
`twi_write(uint8_t)`	Trimite un octet pe bus I2C
`twi_read_ack()`	Citeste un octet si trimite ACK
`twi_read_nack()`	Citeste ultimul octet si trimite NACK
`lcd_send_nibble(…)`	Trimite 4 biti catre LCD prin I2C
`lcd_send_byte(…)`	Trimite un octet complet catre LCD
`lcd_cmd(uint8_t)`	Trimite comanda catre LCD
`lcd_char(char)`	Afiseaza un caracter pe LCD
`lcd_str(const char*)`	Afiseaza un sir de caractere pe LCD
`lcd_set_cursor(col, row)`	Pozitioneaza cursorul pe LCD
`lcd_init()`	Initializeaza LCD-ul in mod 4-bit
`lcd_clear()`	Sterge continutul ecranului
`bcd2dec(uint8_t)`	Converteste valoare BCD in zecimal
`rtc_citeste()`	Citeste data si ora din DS3231 prin I2C
`afiseaza_doua_cifre(uint8_t)`	Formateaza si afiseaza o valoare cu zero in fata

`afiseaza_lcd(DateTime)`	Afiseaza ziua, data si ora pe LCD
`main()`	Bucla principala, logica alarma si dozator

Concluzii

Proiectul a reusit sa rezolve problema reala de la care a pornit. Sistemul functioneaza complet si autonom, iar procesul de a trece de la o idee simpla, pornita dintr-o situatie din familie, la un dispozitiv fizic functional a fost cea mai valoroasa experienta a acestui proiect.

Download

Codul sursa complet al proiectului este disponibil pe GitHub:

https://github.com/socrateJrr/ProiectPM

Bibliografie/Resurse

Resurse Software:

- Cursuri si laboratoare PM - OCW CS PUB
- Documentatie AVR-libc
- Documentatie PlatformIO

Export to PDF