Modul solar dinamic

Introducere

Proiectul consta in realizarea unui sistem care orienteaza automat un panou solar mic in functie de directia luminii, folosind senzori si motoare pas cu pas. Panoul se poate misca pe doua axe pentru a se pozitiona cat mai bine fata de sursa de lumina, iar tensiunea generata si pozitia acestuia sunt afisate pe un display.

Scopul proiectului este de a arata ca un panou solar orientat corect poate produce mai multa energie decat unul fix. Ideea a pornit de la faptul ca soarele isi schimba pozitia pe parcursul zilei, iar un panou fix nu este mereu orientat optim.

Descriere generală

Descrierea modulelor si interactiunea hardware-software:

- **Microcontroller-ul (ATmega328P):** Reprezinta unitatea centrala a sistemului si este conectat la toate modulele. Acesta ruleaza un program care citeste valorile de la senzori, proceseaza datele si comanda miscarea motoarelor si afisajul.
- **Modulul de Senzori (4 x LDR & Panou Solar):** Fotorezistentele sunt utilizate pentru detectarea directiei luminii, fiind conectate in divizoare de tensiune. Microcontroller-ul citeste valorile prin ADC si determina directia optima de orientare. Panoul solar este folosit pentru masurarea tensiunii generate.
- **Modulul de Control Mecanic (Drivere ULN2003 & Motoare 28BYJ-48):** Motoarele pas cu pas controleaza miscarea panoului pe cele doua axe. Microcontroller-ul trimite secvente de comanda catre drivere pentru a roti motoarele si a ajusta pozitia panoului.
- **Modulul de Afisaj (LCD 1602):** Afisajul este utilizat pentru prezentarea informatiilor, precum tensiunea generata si starea sistemului. Comunicarea se realizeaza prin interfata I2C.
- **Modulul Audio (MicroSD & Amplificator SC8002B):** Sistemul permite redarea unor mesaje audio sau notificari. Microcontroller-ul citeste fisiere audio de pe cardul MicroSD prin interfata SPI si genereaza semnal PWM care este amplificat si redat prin difuzor.

Hardware Design

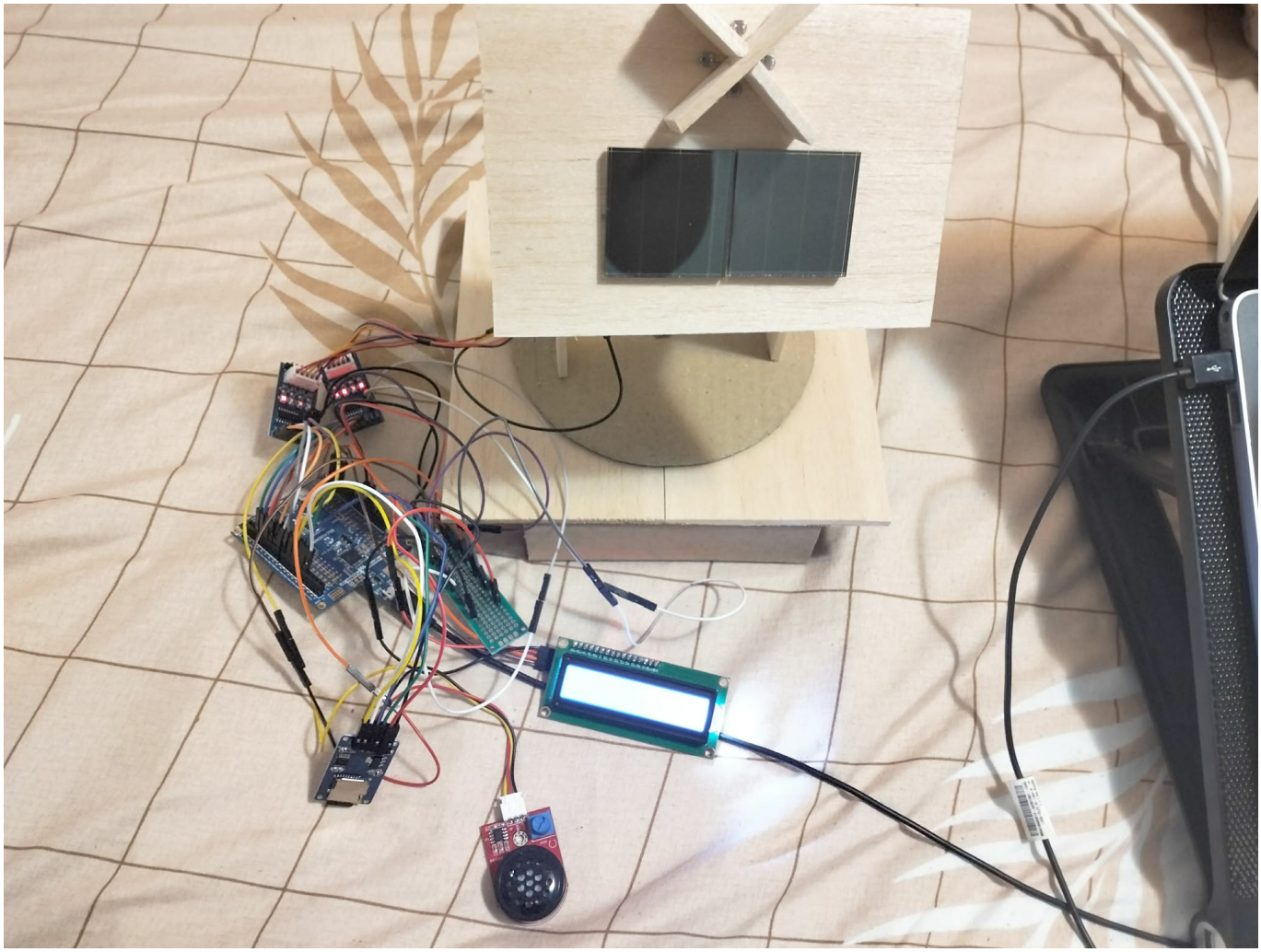
Explicatia schemei electrice: Schema ilustreaza integrarea perifericelor in jurul unitatii centrale (ATmega328P). Alimentarea logica a senzorilor si a modulelor de comunicatie (LCD, MicroSD) se realizeaza la 5V. Achizitia de date analogice (panou si LDR-uri) se face prin convertorul ADC (pinii PC0, PC1, PC2), in timp ce perifericele folosesc magistrale hardware dedicate (I2C pentru display, SPI pentru cardul SD, PWM pentru buzzer).

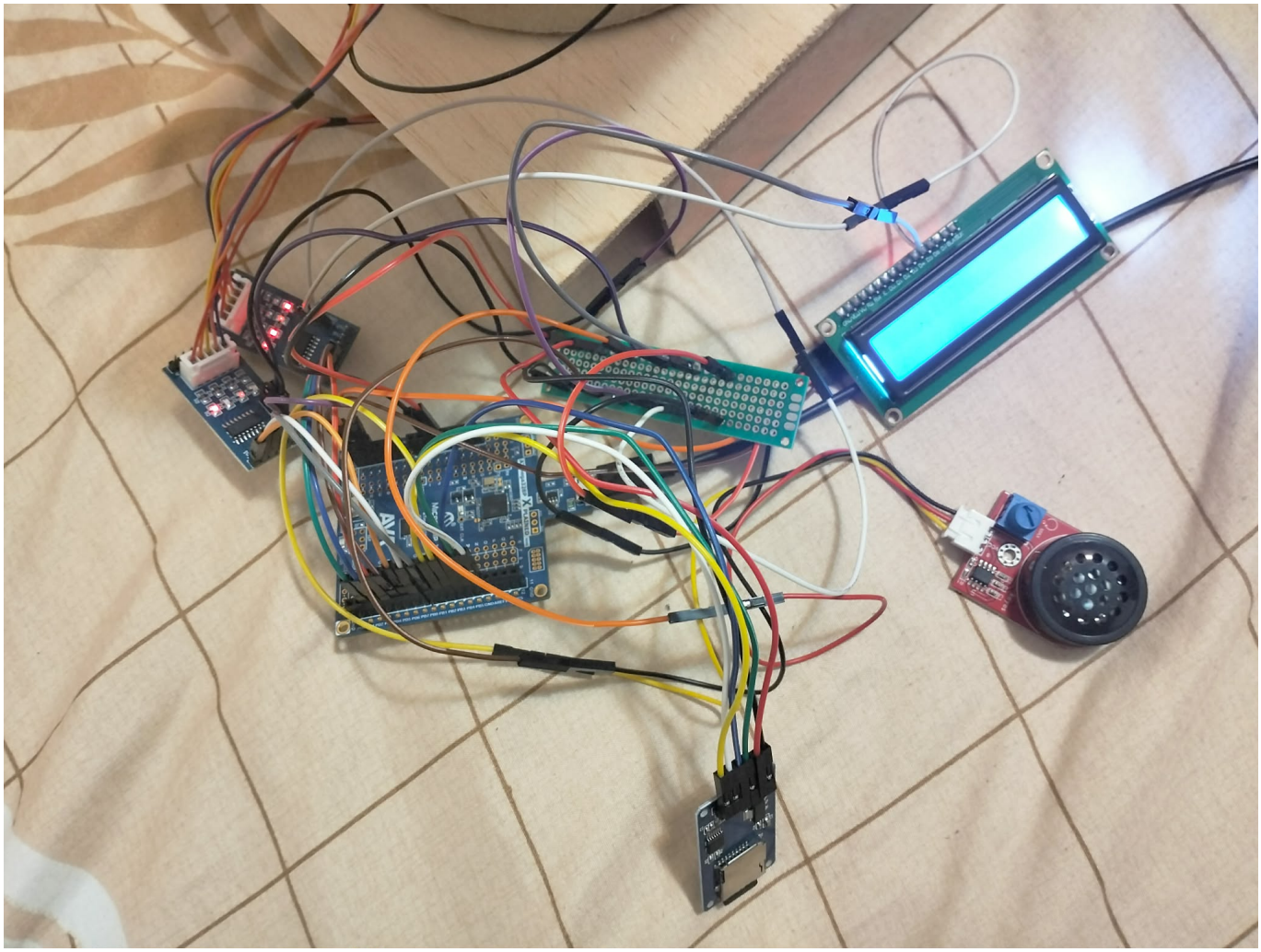
Lista de componente:

Nr. Crt.	Denumire Componenta	Nr. bucati
1	Placa de dezvoltare ATmega328P	1
2	Motor pas cu pas 28BYJ-48	2
3	Driver motor ULN2003	2
4	Senzor de lumina (Fotorezistenta / LDR)	4
5	Panou solar	2
6	Ecran LCD 1602 cu modul I2C	1
7	Modul cititor card MicroSD	1
8	Modul amplificator audio SC8002B	1
9	Difuzor	1

Tabel de conectare a pinilor:

Componenta	Pini ATmega328P	Rol / Functionalitate
Senzori de lumina (LDR)	PC0, PC1	Intrari analogice pentru citirea diferentei de lumina
Panouri solare	PC2	Intrare analogica pentru citirea tensiunii generate
Ecran LCD 1602 (I2C)	PC4 (SDA), PC5 (SCL)	Pinii hardware I2C/TWI pentru comunicare
Modul MicroSD (SPI)	PB2 (CS), PB3 (MOSI), PB4 (MISO), PB5 (SCK)	Pinii hardware SPI pentru salvarea datelor log
Modul Audio & Difuzor	PB1	Iesire PWM pentru alerte sonore
Motor 1 (Stanga-Dreapta)	PD2, PD3, PD4, PD5	Iesiri digitale catre driverul ULN2003 (ax Orizontal)
Motor 2 (Sus-Jos)	PD6, PD7, PB0, PC3	Iesiri digitale catre driverul ULN2003 (ax Vertical)





Stadiul actual al implementarii si testarea componentelor: In imaginile de mai sus este prezentat stadiul curent al montajului fizic. Structura mecanica (pan/tilt) din lemn a fost asamblata, iar modulele electronice au fost cablate preliminar pe breadboard si placute de test.

Dovada functionarii: Asa cum se poate observa in fotografii, sistemul a fost alimentat cu succes, ecranul LCD 1602 primind tensiune (iluminarea de fundal este activa). De asemenea, driverele motoarelor pas cu pas au fost testate separat si primesc corect semnale de comanda.

Software Design

Mediu de dezvoltare

Proiectul a fost dezvoltat folosind mediul **Visual Studio Code**, impreuna cu extensia **PlatformIO**. Aceasta abordare faciliteaza scrierea codului bare-metal in limbajul C, gestionand automat toolchain-ul avr-gcc si procesul de compilare/incarcare a firmware-ului pe microcontrolerul ATmega328P prin intermediul utilitarului avrdude integrat. Pentru depanare (debugging) in timp real, a fost utilizat Serial Monitor-ul oferit nativ de PlatformIO.

Librarii si surse 3rd-party

Pentru a mentine un control strict asupra resurselor limitate de memorie (RAM si Flash), majoritatea modulelor (LCD, ADC, comunicare I2C, actionare motoare, generare semnal PWM) au fost scrise de la zero, lucrand direct cu registrii microcontrolerului.

Singura librarie 3rd-party integrata in proiect este:

- **Petit FAT File System (PetitFS)**: A fost utilizata pentru interfatarea cu cardul SD prin intermediul protocolului SPI. Varianta “Petit” a fost aleasa in mod specific deoarece necesita un consum extrem de mic de memorie RAM de lucru (sub 50 de bytes), fiind ideala pentru arhitectura pe 8 biti a microcontrolerului, permitand parsarea sistemului de fisiere pentru extragerea si redarea fisierelor audio .wav.

Structura fisierelor sursa

Proiectul este organizat modular, fiecare componenta hardware sau software avand propriul set de fisiere header (`.h`) si sursa (`.c`):

- **main.c:** Fisierul principal care orchestreaza intreaga logica a tracker-ului. Implementeaza bucla infinita, algoritmul de mediere pentru senzorii ADC (Oversampling), masina de stari pentru tranzitia zi/noapte si logica de miscare a panoului pe baza diferentelor de iluminare.
- **motor.c / motor.h:** Modulul de actionare mecanica. Gestioneaza secventele de biti pentru driverele ULN2003 aferente motoarelor pas cu pas (axa X si axa Y) si realizeaza multiplexarea manuala a pinilor pe porturile D, B si C. Include functia de oprire (`Motors_Stop`) pentru conservarea energiei.
- **lcd.c / lcd.h:** Driver bare-metal pentru controlul ecranului LCD 1602 prin intermediul modulului I2C (PCF8574). Implementeaza protocolul TWI la nivel de registri pentru a trimite comenzi si date in format de 4 biti.
- **usart.c / usart.h:** Interfata de comunicatie seriala (UART). Initializeaza baud rate-ul si redirecteaza fluxul standard de date (`stdout`), permitand utilizarea functiei `printf()` pentru debugging in Serial Monitor.
- **audio.c / audio.h:** Modulul de procesare si redare audio. Foloseste Timerul 1 pentru a genera semnalul PWM (Fast PWM pe 8-biti) si Timerul 2 pentru a declansa o intrerupere la 8000Hz (rata de esantionare). Tot aici se face parsarea dinamica a header-ului fisierelor .wav.
- **spi.c / spi.h:** Driver pentru protocolul de comunicatie SPI (Serial Peripheral Interface). Configureaza registrii hardware dedicati pentru a stabili o conexiune rapida cu modulul de card SD.
- **sd.c / sd.h:** Modul care trimite comenzile fizice catre cardul SD (ex. CMD0, CMD8, ACMD41) pentru initializare si citirea blocurilor de date.
- **pff.c / pff.h:** Libraria 3rd-party PetitFS. Contine logica sistemului de fisiere FAT16/FAT32, permitand deschiderea, cautarea (lseek) si citirea fisierelor intr-un mediu cu memorie RAM extrem de limitata.
- **integer.h:** Fisier auxiliar de tipologie inclus de libraria PetitFS. Defineste tipurile standard de date (ex. `WORD`, `DWORD`, `BYTE`) pentru a asigura compatibilitatea arhitecturii sistemului de fisiere indiferent de platforma utilizata.

Algoritmi si structuri implementate

Pentru a asigura o functionare stabila si autonoma a tracker-ului solar, s-au implementat urmatoarele mecanisme software:

1. **Automat de Stari (State Machine):** Sistemul foloseste un automat de stari simplu dictat de tensiunea generata direct de panoul solar. Acesta asigura o tranzitie curata si unicat intre modul de “ZI(Activ)” si modul de “NOAPTE”, declansand evenimentele audio specifice.
2. **Oversampling si Filtrare Software (ADC):** Pentru a elimina zgomotul si fluctuatiile fine ale luminii ambientale, citirile de la fotorezistente (LDR) si citirea tensiunii de pe panou trec printr-un algoritm de mediere, esantionand de mai multe ori consecutiv pe acelasi canal.
3. **Zone Moarte (Deadzone):** Pentru controlul motoarelor pas cu pas s-a implementat o toleranta algoritmica de lumina (o marja de eroare fata de centrul ideal). Aceasta previne oscilatiile mecanice continue atunci cand lumina ambientala este uniforma.

Surse si functii implementate

Arhitectura firmware-ului este modularizata pentru a decupla logica senzorilor de cea a actuatorilor.

1. Modulul Principal (main.c) Aici se face preluarea datelor, afisarea pe ecran si rularea masinii de stari pentru tranzitiile audio. Pentru a stabiliza valorile citite de senzorii optici, a fost implementata functia de `Oversampling`, care aduna 8 esantioane consecutive si returneaza media lor, ignorand astfel zgomotul electric:

```
uint16_t ADC_ReadFiltered(uint8_t ch) {
    uint32_t sum = 0;
    for (uint8_t i = 0; i < 8; i++) {
        sum += ADC_Read(ch);
        _delay_us(300);
    }
    return (uint16_t)(sum / 8);
}
```

Logica de tranzitie Zi/Noapte asigura redarea fisierului WAV corespunzator o singura data la trecerea pragului de tensiune, folosind flag-ul `este_zi`:

```
if (voltaj_mv >= PRAG_ZI && este_zi != 1) {
    este_zi = 1;
    printf("--> [EVENIMENT] A rasarit soarele!\n");
    WAV_Play("COCOS.WAV");
}
else if (voltaj_mv <= PRAG_NOAPTE && este_zi != 0) {
    este_zi = 0;
    printf("--> [EVENIMENT] S-a intunecat. Trecem pe modul de noapte.\n");
    WAV_Play("NOAPTE.WAV");
}
```

2. Modulul de Motoare (motor.h / motor.c) Contine secventele de biti pentru driverul ULN2003 (Half-Step, 8 pasi). Deoarece pinii motorului pe axa Y (Verticala) sunt raspanditi pe porturi fizice diferite (PORTD, PORTB, PORTC) pentru a evita conflictele cu interfetele I2C si PWM, functia

`MotorY_Move` implementeaza multiplexarea manuala a semnalelor catre fiecare pin in parte:

```
void MotorY_Move(int8_t directie) {
    if (directie > 0)
        step_index_Y = (step_index_Y + 1) & 7;
    else if (directie < 0)
        step_index_Y = (step_index_Y - 1) & 7;

    uint8_t step = step_sequence[step_index_Y];

    // Curatam pinii specifici de pe porturile D, B si C
    PORTD &= ~((1 << PD6) | (1 << PD7));
    PORTB &= ~(1 << PB0);
    PORTC &= ~(1 << PC3);

    // Mapam fiecare bit din secventa catre pinul hardware corect
    if (step & 0x01) PORTD |= (1 << PD6);
    if (step & 0x02) PORTD |= (1 << PD7);
    if (step & 0x04) PORTB |= (1 << PB0);
    if (step & 0x08) PORTC |= (1 << PC3);
}
```

Rezultat

https://drive.google.com/file/d/1DGZJ26SoPe8rLa0h7vX-gNK8Hjuyt_yA/view?usp=sharing

Concluzii

Acest proiect a reprezentat o provocare complexa si o sinteza excelenta a conceptelor de programare low-level si hardware design. Dezvoltarea unui Solar Tracker hibrid, folosind o abordare strict bare-metal in C, a demonstrat importanta gestionarii eficiente a resurselor limitate ale unui microcontroler (precum memoria RAM de doar 2KB a cipului ATmega328P) in conditiile rularii unor sarcini simultane.

Principalele realizari si lectii invatate includ:

- **Integrarea Hardware-Software:** Am reusit sa sincronizez componente cu cerinte hardware si de timp complet diferite: citirea senzorilor analogici prin oversampling, actionarea fluida a motoarelor pas cu pas, afisarea de date pe I2C si redarea audio in timp real (prin generare de semnal PWM si comunicatie SPI cu un card SD).
- **Optimizarea Resurselor:** Integrarea modulului audio (PetitFS) alaturi de restul perifericelor a provocat initial coliziuni in memorie (Stack Overflow) si caderi de tensiune. Aceasta problema critica a fost depasita prin ajustarea atenta a bufferelor de citire la 256 bytes si optimizarea codului de debugging.
- **Flexibilitate si Control Absolut:** Renuntarea la bibliotecile predefinite abstractizate a oferit un

control total asupra registrilor (Timere, ADC, TWI, SPI). Acest lucru a permis implementarea unor solutii personalizate vitale, precum multiplexarea manuala a pinilor pentru motoare pe porturi diferite si scrierea unui driver propriu, eficient, pentru ecranul LCD.

Download

Bibliografie/Resurse

Acest proiect a fost realizat consultand urmatoarele surse oficiale si documentatii tehnice, structurate pe componente hardware si software:

Resurse Hardware

- **Datasheet ATmega328P (Microchip):** Sursa esentiala pentru intelegerea si configurarea la nivel de bit a registrilor pentru Timere (generare PWM si intreruperi), ADC (conversie analog-digitala), SPI si TWI (I2C).
- **Datasheet PCF8574 (Texas Instruments):** Documentatia pentru modulul “Remote 8-bit I/O expander for I2C-bus”, necesara pentru implementarea algoritmului bare-metal de comunicatie intre microcontroler si ecranul LCD 1602.
- **Datasheet Driver ULN2003:** Specificatiile tehnice ale array-ului de tranzistori Darlington, folosit impreuna cu motorul stepper 28BYJ-48. A fost esential pentru calculul unghiular al pasilor (reductor intern 1:64) si intelegerea secventei corecte de magnetizare a bobinelor in modul Half-Step si Full-Step.

Resurse Software

* **Laboratoarele OCW - Proiectarea Microprocesoarelor:** Suportul teoretic si practic principal pentru fundamentarea cunostintelor despre arhitectura AVR, protocoalele de comunicatie seriala (UART, SPI, I2C) si manipularea la nivel de port.

- **Petit FAT File System Module (Elm-Chan):** Documentatia oficiala si codul sursa pentru libraria PetitFS, utilizata pentru accesarea sistemului de fisiere de pe cardul SD in conditii de memorie RAM extrem de redusa (specific arhitecturilor pe 8 biti).
- **AVR Libc Reference Manual:** Utilizat pentru detaliile de implementare a bibliotecilor standard C pe mediul AVR (in special headerele <avr/io.h> si <avr/interrupt.h>).
- **Standardul RIFF WAVE Audio Format:** Referinta tehnica folosita pentru structurarea functiei custom de parsare a header-ului fisierelor .wav, necesara pentru extragerea automata a frecventei de esantionare (Sample Rate) si a dimensiunii datelor PCM.

Export to PDF