Articulație panou solar "Sunflower"

- Nume: Mărunțiș Andrei
- Grupă: 333CA

Introducere

Tranziția lumii către energie verde este un subiect de mare actualitate în ultima vreme, pentru a susține eforturile de a încetini și opri încălzirea globală. Modul în care doresc să contribui la acest efort este **crearea unei metode de a îmbunătăți eficiența panourilor solare**.

Fabricarea panourilor solare este un proces deosebit de scump, din cauza complexității mari și a nivelului ridicat de puritate al materialelor necesare obținerii unui panou solar de calitate.

Din cauza costurilor mari ale unui panou solar (aproape 10.000 lei pentru a alimenta o gospodărie), un bun complementar care poate maximiza eficiența investiției este binevenit. Articulația propusă în cadrul acestui proiect va orienta automat panoul solar către soare, pentru a optimiza unghiul sub care razele soarelui cad pe acesta și pentru a maximiza producția de energie electrică. Modul de funcționare este inspirat de planta floarea soarelui și comportamentul articulației va fi aproape identic cu aceasta.

Pe lângă beneficiile ecologice ale unui asemenea sistem, articulația poate aduce și beneficii financiare cumpărătorului: creșterea producției de energie electrică a panoului fotovoltaic reduce semnificativ și costul facturilor la energie electrică. Astfel, dacă articulația are un cost de producție suficient de mic, ea poate deveni un bun indispensabil sistemelor fotoelectrice.

Descriere generală

Mai jos prezint schema bloc a sistemului propus pentru articulația solară:

- Nucleul sistemului este un microcontroller Arduino Uno, care se va ocupa de orientarea panoului solar către soare. Voi explica mai jos cum realizează această operație.
- În partea stângă, în chenare roșii, se află cele două elemente care oferă microcontroller-ului date despre locația panoului solar: locația (coordonatele geografice) și poziția în spațiu. Folosindu-se de aceste două informații, precum și de ora curentă (stocată intern) microcontroller-ul poate calcula

exact direcția către care trebuie orientat panoul solar.
- Orientarea panoului solar către direcția calculată se face prin intermediul unui servomoator care controlează unghiul azimutal (stânga-dreapta). Se poate adăuga și un al doilea servomotor pentru a controla altitudinea (sus-jos).
- În cadrul acestui prototip, panoul solar va fi conectat la convertorul analog-digital al microcontller-ului, pentru a putea calcula și afișa puterea generată de panou. Firește, trebuie introdus și un circuit de protecție, în cazul în care puterea generată de panou este suficient de mare pentru a deteriora pinii microcontroller-ului.
- Sistemul va avea atașat și un display LCD, pe care vor fi afișate diverse date despre starea sistemului, precum:
  - Puterea generată de panou
  - Data și ora
  - Locația geografică

Hardware Design

Listă de piese necesare:

- 1x Microcontroller Arduino Uno
- ~~1x Modul Accelerometru 9 axe MPU9250~~
- 1x Modul Accelerometru 6 axe MPU6500
- 1x Modul GPS NEO-6MV2
- 1x Display LCD
- 1x Motor pas cu pas 28BYJ-48

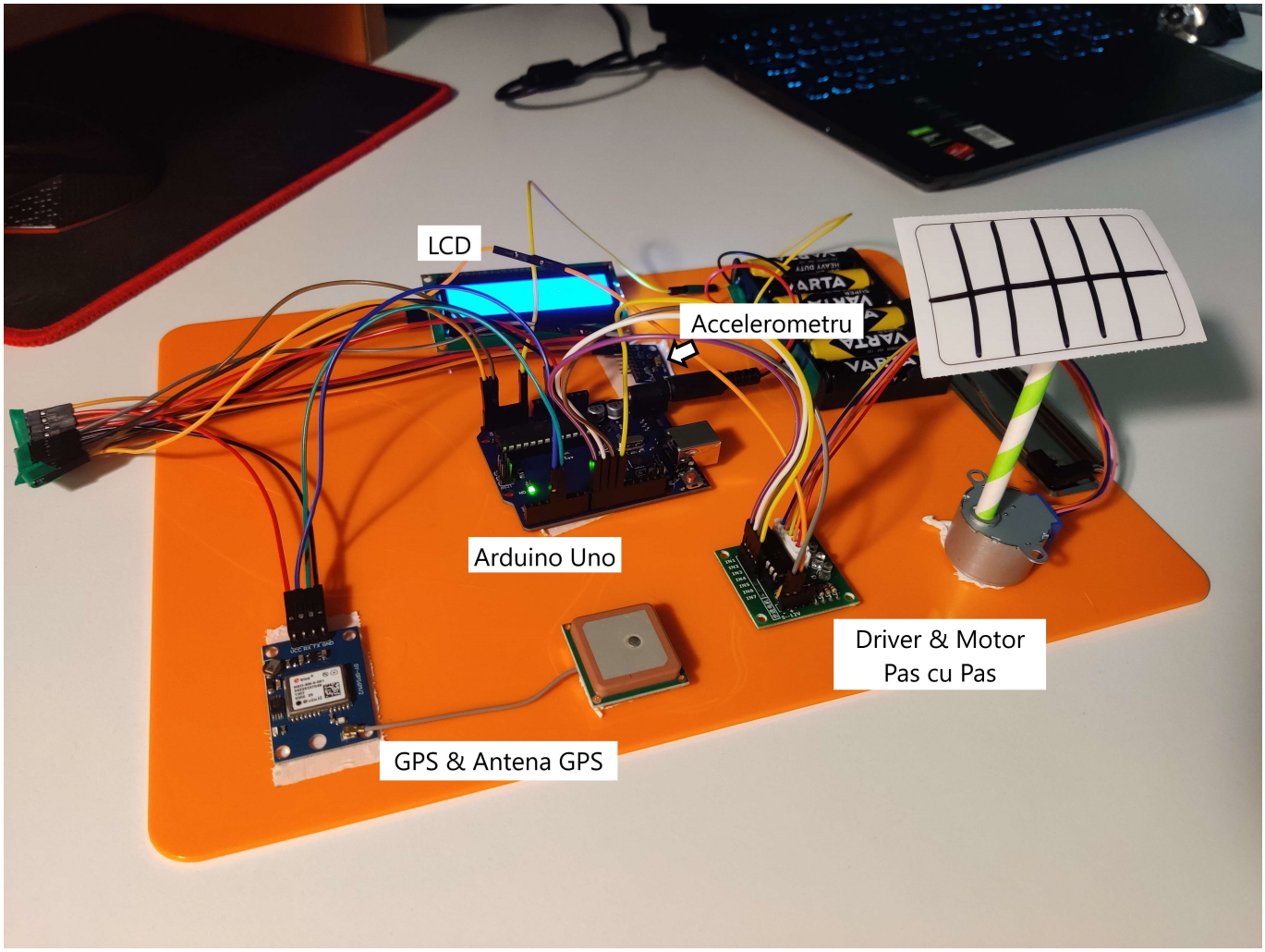
Schema electrică

După cum se poate observa, am conectat perifericele astfel:

- LCD-ul și accelerometrul MPU-6500 comunică prin I2C, astfel că i-am conectat la pinii SCL și SDA corespunzători de pe Arduino (A4, respectiv A5)
- Modulul GPS comunică prin serială (UART), astfel că l-am conectat la pinii TX și RX de pe plăcuță D3, respectiv D4
- Motorul pas cu pas are 4 pini IN1-4 pe care i-am conectat la pinii D8-11 de pe Arduino
- Toate componentele sunt alimentate de la pinii de 5V și GND ai plăcuței Arduino
- “Panoul solar” este conectat la unul dintre pinii analogici ai plăcuței (A0) pentru a putea folosi Convertorul Analog-Digital

Aspect fizic

În poza de mai jos se poate observa cum arată proiectul și fiecare componentă:



Software Design

Pentru dezvoltarea aplicației am folosit modulul **PlatformIO** integrat în aplicația Visual Studio Code.

Librăriile externe folosite sunt următoarele:

- InvensenseImu - pentru interfațarea cu accelerometrul cu 6 axe
- Liquid Crystal I2C - pentru intefațarea cu display-ul LCD
- AltSoftSerial - pentru comunicația cu modulul GPS
- TinyGpsPlus - pentru parsarea datelor de la modulul GPS, în format NMEA
- Unistep2 - pentru controlul **asincron** al motorului pas cu pas
- Time - pentru un ceas facil pe Arduino
- TimerInterrupt - pentru execuția periodică a anumitor instrucțiuni

Modul de funcționare

Înainte de a prezenta codul, este necesar să explic cum funcționează proiectul.

La pornire, acesta trece printr-o fază de inițializare în care modulul GPS caută semnal de la satelit. Conform datasheet, modulul GPS are nevoie de aproximativ 27s pentru a se conecta la sateliți,

presupunând vedere clară a cerului. Căutarea semnalului durează maxim 90s, după care microcontroller-ul trece în faza următoare, știind locația și ora UTC. În cazul în care GPS-ul nu se poate conecta la satelit în cele 90s, este afișat un mesaj de eroare și se va folosi locația default a holului EC din Facultatea de Automatică și Calculatoare.

După ce se termină inițializarea, proiectul începe treaba efectivă de a stabiliza direcția panoului solar montat deasupra servomotorului. Pentru a face asta, se folosește de datele primite de la accelerometru, mai precis viteza unghiulară în jurul axei Oz (rotația stânga-dreapta a panoului). Având în vedere relația matematică dintre viteză și distanță, pot calcula unghiul cu care trebuie rotit panoul prin integrarea pulsației față de timp. La final, microcontroller-ul va comanda motorul pentru a roti panoul. În timp ce microcontroller-ul efectuează aceste operații, pe LCD vor fi afișate date despre tensiunea generată de panoul solar și ora locală.

Module de cod

La începutul fișierului am directivele de preprocesor pentru a include librăriile externe folosite în cadrul proiectului și declararea structurilor de date corespunzătoare diverselor periferice ca variabile globale. Spre exemplu, următoarea bucată de cod îmi inițializează o structura de comunicație cu display-ul având în vedere adresa slave-ului (0x27) și dimensiunea ecranului (2 linii a câte 16 coloane):

```
#include <LiquidCrystal_I2C.h>

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

Orice proiect Arduino are la bază 2 funcții: `setup`, funcție care se rulează o singură dată la pornirea în execuție a codului, și `loop`, funcție care se execută încontinuu cât timp microcontroller-ul este pornit și funcționează.

În cadrul funcției `setup` încep prin a inițializa conexiunile cu perifericele. Spre exemplu, pentru LCD pornesc comunicația prin I2C cu următoarele funcții implementate în biblioteca LiquidCrystal I2C:

```
// Init LCD
lcd.init();
lcd.backlight();
lcd.setCursor(0, 0);
```

În mod asemănător inițializez pe rând fiecare periferic, unele dintre ele afișând eventuale erori pe LCD, cum ar fi accelerometrul:

```
// Start the I2C bus for accelerometer, 0x68 address
Wire.begin();
Wire.setClock(400000);
imu.Config(&Wire, bfs::Mpu6500::I2C_ADDR_PRIM);
/* Initialize and configure IMU */
if (!imu.Begin())
{
    lcd.print("Error initializing communication with IMU");
    while (1)
```

```
    {
    }
}
/* Set the sample rate divider */
if (!imu.ConfigSrd(19))
{
    lcd.print("Error configured SRD");
    while (1)
    {
    }
}
```

În mod special merită menționată comunicația cu Serial Monitorul din VS Code. La începutul fișierului este definită o constantă care spune compilatorului dacă să includă zonele de cod legate de Serial Monitor în sursa finală, iar orice bucată de cod referitoare la Serial Monitor este înconjurată de directive de preprocesor, astfel:

```
#define DEBUG_SERIAL false

#if DEBUG_SERIAL
    Serial.begin(9600);
#endif
```

După ce se termină inițializarea tuturor perifericelor, se trece în etapa în care se caută semnal GPS. Aceasta este ceva mai complexă și are, de altfel, dedicată propria funcție.

Înainte de toate, se inițializează comunicația prin UART cu modulul GPS:

```
#include <SoftwareSerial.h>

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);

void initGPS()
{
    ss.begin(GPSBaud);

    // ...
}
```

Urmează o parte de cod care afișează pe LCD un mesaj cât timp se caută semnal GPS:

```
volatile int count = 0;

void initGPS()
{
    // ...

    lcd.setCursor(0, 0);
    lcd.print("GPS signal");
```

```
    while (1)
    {
        lcd.setCursor(10, 0);
        for (int i = 0; i <= count % 3; i++)
        {
            lcd.print(".");
        }
        for (int i = count % 3; i < 3; i++)
        {
            lcd.print(" ");
        }
        // ...
    }
}
```

Astfel, se afișează mesajul “Searching GPS...”, iar numărul de puncte se schimbă la fiecare secundă. Pentru a contoriza această secundă, inițializez un Timer care generează o întrerupere la fiecare 1s în care incrementez variabila `count` de mai sus:

```
#define TIMER_INTERRUPT_DEBUG 2
#define _TIMERINTERRUPT_LOGLEVEL_ 0
#define USE_TIMER_1 true
#include <TimerInterrupt.h>

void GPSTimerHandler()
{
    count++;
}

void initGPS()
{
    ITimer1.init();
    count = 0;

    if (ITimer1.attachInterruptInterval(TIMER_INTERVAL_MS, GPSTimerHandler,
TIMER_DURATION_MS))
    {
#if DEBUG_SERIAL
        Serial.print(F("Starting  ITimer1 OK, millis() = "));
        Serial.println(millis());
#endif
    }
    else
    {
#if DEBUG_SERIAL
        Serial.println(F("Can't set ITimer1. Select another freq. or timer"));
#endif
    }
}
```

În interiorul buclei se citesc efectiv datele primite de la GPS, caracter cu caracter, în format NMEA

care este parsat cu ajutorul librăriei TinyGps++:

```
void initGPS()
{
    while (1)
    {
        // ...

        if (ss.available() > 0)
        {
            char c = ss.read();
#if DEBUG_SERIAL
            Serial.print(c);
#endif
            gps.encode(c);
            if (gps.location.isUpdated())
            {
                lcd.clear();
                lcd.setCursor(0, 0);
                lcd.print("Lat=");
                lcd.print(gps.location.lat(), 6);
                lcd.setCursor(0, 1);
                lcd.print("Long=");
                lcd.print(gps.location.lng(), 6);
                foundLoc = true;
                break;
            }
        }
    }
}
```

În cazul în care se găsește semnal, se afișează pe display locația primită de la GPS, în caz contrat un mesaj de eroare și locația default anume hol EC.

După ce se termină procesul de căutare a locației se inițializează ceasul intern al microcontrollerului cu valoarea primită de la GPS, astfel:

```
void calibrateTime()
{
    if (gps.time.isValid() && gps.date.isValid())
    {
        int year = gps.date.year();
        int month = gps.date.month();
        int day = gps.date.day();
        int hour = gps.time.hour();
        int minute = gps.time.minute();
        int second = gps.time.second();
        // Set the Time to the time retrieved from the GPS
        // Assume current location is UTC+3
        setTime(hour + 3, minute, second, day, month, year);
    }
```

```
    else
    {
        setTime(19, 0, 0, 27, 5, 2024);
    }
}
```

Pentru a avea ora locală se presupune că timezone-ul în care se află locația găsită este UTC+3 (ora de vară a României). Folosind datele de locație se poate determina timezone-ul, însă asta presupune o catalogare a acestora, întrucât timezone-urile sunt separate doar aproximativ după longitudine.

După terminarea acestui proces se trimite semnalul de standby către GPS, pentru a reduce consumul de energie:

```
#define PMTK_STANDBY "$PMTK161,0*28"

ss.println(PMTK_STANDBY);
```

În următoarea etapă de funcționare a proiectului, microcontroller-ul citește periodic date de la accelerometru pentru a ajusta poziția panoului solar. Inițializez un nou timer pentru asta asemănător cu ce am prezentat mai sus, iar întreruperea va seta o variabilă numită `flag` pe `true` pentru a semnaliza funcției `loop` că trebuie să citească date noi de la accelerometru.

Codul care se ocupă de citirea acestor date de la accelerometru și calculul direcției după care se va orienta panoul solar este următorul:

```
// Defines the number of steps per rotation for stepper motor
const int stepsPerRevolution = 4096;
// Define some steppers and the pins they will use
// pins for IN1, IN2, IN3, IN4, steps per rev, step delay(in micros)
Unistep2 stepper(8, 9, 10, 11, stepsPerRevolution, 1000);

double ang_speed = 0;
double target_angle = 0;
int target_step = 0;
int current_step = 0;

void loop()
{
    // We need to call run() frequently, so we place it in the loop()
    stepper.run();

    // Read accelerometer data
    if (flag)
    {
        if (imu.Read())
        {
            ang_speed = imu.gyro_z_radps();
            ang_speed -= ACCEL_NOISE;
            if (ang_speed > ACCEL_TOL || ang_speed < -ACCEL_TOL)
            {
                Serial.println("Ajung aici");
```

```
                target_angle = target_angle - (ang_speed *
ACCEL_READ_DELAY_MS / 1000) * STEPPER_MULT;
                while (target_angle > 2 * PI)
                {
                    target_angle -= 2 * PI;
                }
                while (target_angle < 0)
                {
                    target_angle += 2 * PI;
                }
            }
            target_step = (int)(stepsPerRevolution * (target_angle + (hour()
> 12 ? PI : 0)) / (2 * PI));
            if (abs(target_step - current_step) > STEPPER_TOL)
            {
                stepper.moveTo(target_step);
            }

            // ....
        }
    }
}
```

Înainte de toate, motorul pas cu pas este pseudo-asincron: microprocesorul din Arduino Uno (ATMega328P) nu este capabil de multithreading, astfel că simulez acest efect apelând la fiecare execuție a loop-ului metoda `stepper.run()` care va roti panoul solar către o poziție setată în prealabil cu un apel al metodei `stepper.moveTo`.

Formulele folosite sunt foarte simple:

- Având în vedere că eșantionez valoarea vitezei unghiulare cu perioadă fixă (0.1s), pot să înmulțesc această valoare cu timpul pentru a obține “aria de sub grafic” anume diferența de unghi față de ultima măsurătoare. Valorile efective le calculez cu o anumită toleranță
- Mă asigur că unghiul obținut (care va fi exprimat în radiani) este în intervalul [0, 2π)
- Calculez poziția la care trebuie să deplasez panoul solar folosind regula de 3 simplă (un cerc este 2π și corespunde 4096 de pași pentru motorul pas cu pas, deci pot să calculez câți pași corespund unghiului calculat)
- Dacă diferența dintre poziția corectă și poziția actuală a motorului este peste un anumit prag, trimit comanda **asincronă** către stepper să se rotească.

În ceea ce privește orientarea automată a panoului către soare, am implementat un model extrem de simplu în care soarele răsare la Est și apune la Vest (emisfera nordică). Se poate implementa un model mai complex cu formulele găsite aici.

Cât timp microcontroller-ul se ocupă de ajustarea unghiului panoului solar, pe LCD se afisează date despre tensiunea generată de panou, respectiv ora, folosind următoarele funcții:

```
void showVoltage(int row)
{
    // Read battery voltage from ADC
    int voltage = analogRead(A0);
```

```
    float voltage_f = voltage * (5.0 / 1023.0);
    // multiply by 10 due to divider
    voltage_f *= 10;
    lcd.setCursor(0, row);
    lcd.print("Voltage: ");
    lcd.print(voltage_f);
    lcd.print("V");
    lcd.print(" ");
}

void showTime(int row)
{
    lcd.setCursor(0, row);
    lcd.print("Time: ");
    if (hour() < 10)
    {
        lcd.print("0");
    }
    lcd.print(hour());
    lcd.print(":");
    if (minute() < 10)
    {
        lcd.print("0");
    }
    lcd.print(minute());
    lcd.print(":");
    if (second() < 10)
    {
        lcd.print("0");
    }
    lcd.print(second());
    lcd.print(" ");
}

void loop()
{
    // Show battery voltage
    showVoltage(0);
    // Show current time
    showTime(1);
}
```

Rezultate Obţinute

În practică, modulul GPS are nevoie de destul de mult timp pentru a se conecta la sateliți. Prima dată când se conectează poate să dureze foarte mult (chiar și 15 minute), mai ales pentru copiile chinezești. După aceea, modulul GPS se va conecta în câteva minute la satelit (1-5 minute), având în vedere blocurile înalte din orașul București. Într-o zonă mai liberă este de așteptat să se conecteze

chiar mai repede de atât.

Cu toate acestea, merită menționat faptul că GPS-ul are nevoie de minim 3 (ideal peste 4) sateliți pentru a stabili locația curentă, însă el are și funcționalitatea de a procura ora universală dacă se poate conecta la minim 1 satelit. Acest lucru se întâmplă adesea chiar și înăuntrul casei, oferind modulului GPS și funcționalitate de RTC fiabil.

În ceea ce privește rotația panoului solar după “soare”, acesta își păstrează direcția corectă cu o eroare mică, în general mai puțin de 30 grade.

Pentru un videoclip de prezentare a proiectului, click mai jos:

Concluzii

Articulația obținută pentru panou este una foarte potrivită pentru dispozitive alimentate solar și care se deplasează, spre exemplu vapoare, automobile. Având în vedere funcționalitatea de stabilizator față de rotații, este un dispozitiv unic pe piață.

Download

Arhiva de cod

Jurnal

- 4 mai 2024: Definitivarea temei proiectului și inițializarea paginii de wiki
- 20 mai 2024: Realizat schema electrică
- 26 mai 2024: Completat pagina de wiki

Bibliografie/Resurse

1. Wikipedia: Coordonate astronomice orizontale
2. Motor pas cu pas pe Arduino
3. Tutorial LCD I2C
4. Ghid modul GPS NEO-6MV2
5. Poziția soarelui
6. Ghid control motor pas cu pas folosing Arduino

Export to PDF