

Documentația proiectului

Designul unui sistem de urmărire solară folosind platforma Arduino

Ciumeica Elena și Gheorghiu Nadejda

1 Importanța proiectului

Sursa de energie fotovoltaică este una dintre cele mai utilizate surse de energie regenerabile din lume. Sistemele fotovoltaice sunt nepoluante, fără zgomot, ușor de instalat și întreținut, și durabile. Cu toate acestea, principalele dezavantaje sunt costul ridicat și eficiența scăzută. În acest context, sistemele de urmărire solară, a căror scop e de a menține panoul fotovoltaic perpendicular radiației solare, sunt cele mai bune dispozitive pentru maximizarea energiei colectate de panou [1].

Lucrarea dată prezintă proiectarea și implementarea a unui sistem simplu de urmărire solară pe două axe cu scopul de a urmări mișcarea soarelui utilizând puține componente și de a avea costuri reduse. Printre obiectivele propuse avem:

1. Mecanismul de urmărire trebuie să fie fiabil și capabil să monitorizeze soarele cu un anumit grad de precizie, chiar și pe timp înnoțat.
2. Readucerea mecanismului în poziția inițială la sfârșitul zilei/apusul soarelui. Aceasta poate fi considerată drept o funcție de protecție la condițiile mediului.

Proiectarea sistemului de urmărire solară e realizată după un design propriu. Mecanismul cu două axe este dezvoltat pentru a înclina panoul fotovoltaic cu ajutorul a două motoare pas cu pas după un sistem automat de urmărire virtuală, care folosește un algoritm matematic pentru a determina poziția soarelui și rotația ulterioară a mecanismului.

Datele de funcționare a sistemului de urmărire și unghiurile de rotație sunt determinate folosind platforma electronică programabilă Arduino UNO [2], un RTC pentru măsurarea trecerii timpului și modulul suplimentar MPU-6050 ce poate funcționa ca accelerometru și giroscop. Programul pentru calcularea unghiurilor de rotație și afișare a valorilor a fost creat în software-lui gratuit Arduino (IDE) și instrumentul de programare vizuală Node-Red, iar dispozitivul în sine și modulele atașate au fost alimentate prin cablu USB.

Dispozitivul poate fi programat și alimentat simultan printr-un singur cablu. Conectarea dispozitivului la o sursă de alimentare externă îi oferă acestuia portabilitate. Iar utilizarea unui sistem de urmărire solară portabil și eficient poate avea o implementare industrială sau rezidențială foarte variată.

Cu toate acestea, în cazul nostru, în scopul de a afișa datele este necesar de a obține răspuns în timp real. Din acest motiv, programul creat afișează valorile într-un interval de timp specificat, cu ajutorul portului serial.

Pentru transmiterea datelor în timp real dispozitivul trebuie să fie conectat continuu la un calculator utilizând un cablu de date. O soluție mai bună ar fi utilizarea unui cablu de rețea și o extensie USB/LAN de o lungime mai mare sau modificarea ulterioară a aplicației pentru a transmite datele prin Wi-Fi.

2 Designul proiectului

Arduino UNO este cea mai bună, robustă, utilizată și documentată placă pentru începători. Arduino este o platformă electronică disponibilă gratuit, bazată pe hardware și software ușor de utilizat. Limbajul de programare Arduino permite să scriem software multistrat pentru a controla dispozitivele conectate la o gamă largă de plăci cu microprocesor și a crea obiecte și medii interactive.

Există mai multe platforme Arduino și o serie de tipuri de senzori. Pentru proiectul nostru am utilizat modelul de microcontroler ATmega16U2 Arduino UNO2, MPU-6050 cu funcția accelerometru și giroscop și un modul RTC DS3231.

Senzorul InvenSense MPU-6050 [3] conține un accelerometru și un giroscop într-un singur cip. Este foarte precis, deoarece conține un convertor digital analogic pe 16 biți pentru fiecare canal. Cipul preia valori din axele de coordonate x, y și z simultan. Senzorul folosește o magistrală I2C pentru a se conecta cu Arduino.

DS3231 [4] este un ceas în timp real extrem de precis, care ține minte ore, minute și secunde, precum și

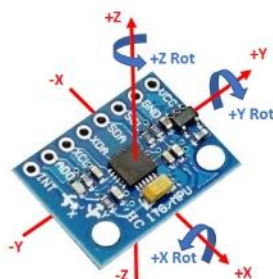


Figure 1: Direcția de preluare de date a modulul pe 3 axe MPU-6050

informații despre zi, lună și an. Modulul RTC funcționează pe o baterie separată și poate ține evidența timpului chiar dacă reprogramăm microcontrolerul sau deconectăm alimentarea principală. Modulul de asemenea folosește protocolul de comunicare I2C.

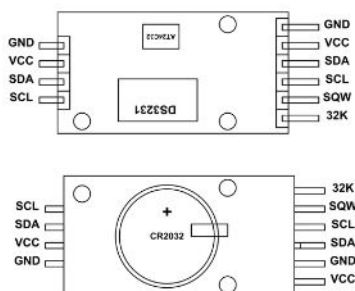


Figure 2: Configurația modulului RTC DS3231

I2C este o magistrală de comunicare foarte populară și utilizată pe scară largă de multe dispozitive electronice, deoarece poate fi implementată cu ușurință în multe modele care necesită comunicare între un dispozitiv master și mai multe dispozitive slave sau master [5]. Fiecare dispozitiv are un ID prestabilit sau o adresă unică, astfel încât dispozitivul master să poată alege cu ce dispozitive comunică. Astfel, adresa dispozitivului MPU-6050 am setat-o la 0x69.

Pentru a programa comunicarea între Arduino și modulele I2C am folosit bibliotecile ajutătoare Wire.h [6] (pentru MPU) și DS3231.h [7] (pentru RTC).

Implementarea ușoară vine cu faptul că sunt necesare doar două fire. Cele două fire sunt numite Serial Clock (SCL) și Serial Data (SDA) [8]. Firul SCL este semnalul de ceas care sincronizează transferul de date între dispozitivele de pe magistrala I2C și este generat de dispozitivul master. Cealaltă linie (SDA) transportă datele.

Comunicarea cu interfața utilizatorului în Node-Red se face prin intermediul portului serial COM inițializat. Aceasta va fi prezentată mai detaliat în capitolul următor.

Pentru a face mișcarea de rotație avem mai multe opțiuni. Fiindcă e vorba de o sarcină mare și rotație mai exactă fără a folosi un magnetometru, soluția cea mai potrivită e motorul pas cu pas.

Motorul pas cu pas este un motor controlat de o serie de bobine electromagnetice. Arborele central are o serie de magneti montați pe el, iar bobinele care înconjoară arborele au alternativ curent sau nu, creând câmpuri magnetice care resping sau atrag magnetii de pe arbore, determinând rotirea motorului. Există două tipuri de bază ale motoarelor pas cu pas, pas cu pas unipolar și pas cu pas bipolar.

Motoarele pas cu pas responsabile pentru mișcarea sistemului pe două axe în proiectul nostru sunt: NEMA 23 [9] responsabil de setarea direcției orizontale și 28BYJ-48 [10] responsabil de setarea direcției verticale a panoului PV.

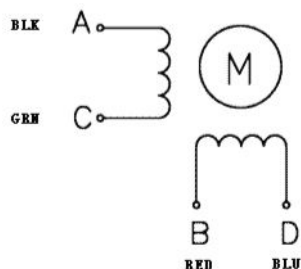


Figure 3: Bobina motorului NEMA 23

Motorul NEMA 23 este un motor pas cu pas bipolar cu 4 fire și 2 bobine. Dacă energizăm ambele bobine în opus în același timp, atunci putem deplasa motorul în trepte. Pentru a controla motorul folosim un driver L293D [11].

L293D unul dintre cele mai ușoare și mai ieftine moduri de a controla motoarele, controlând atât viteza (PWM), cât și direcția de rotire (H-Bridge). L293D este un driver pentru motoare cu două canale H-Bridge capabil să conducă două motoare DC sau un motor pas cu pas.

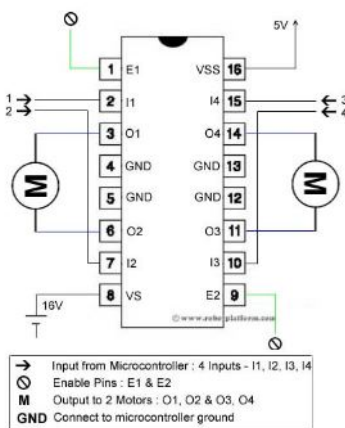


Figure 4: Schema conexiunilor L293D

Driver-ul folosește biblioteca Stepper.h [12] pentru a rezolva problema puterii motorului și va opera motorul de către microcontroler. NEMA 23 folosește 200 de impulsuri pentru a controla pasul de 1.8 grade.

Motorul 28-BYJ48 este un motor unipolar. Motorul pas cu pas unipolar are, de obicei, cinci sau șase fire și patru bobine (sau, mai bine zis, două bobine împărțite prin conexiuni centrale pe fiecare bobină).

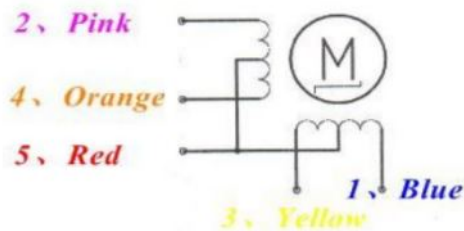


Figure 5: Bobina motorului 28BYJ-48

Conexiunile centrale ale bobinelor sunt legate între ele și utilizate ca conexiune de alimentare. Sunt numiți pași unipolari, deoarece puterea intră întotdeauna pe acest singur pol.

Pentru a manipula cu 28-BYJ48 folosim un driver cu cipul ULN2003A [13]. Acesta este unul dintre cele mai frecvente circuite integrate ale driverelor de motor. Placa are un conector pentru firele motorului, conexiuni pentru patru intrări de control, conexiuni de alimentare cu energie electrică, precum și patru LED-uri care arată activitate pe cele patru linii de intrare de control care indică starea de pas, oferind un aspect vizual a acestuia.

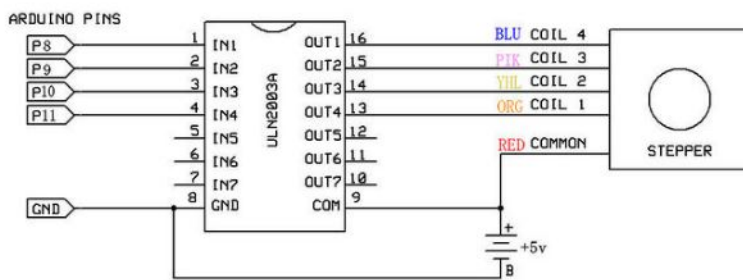


Figure 6: Conectarea motorului 28-BYJ48 cu driver-ul ULN2003A

Conectăm driverul la pini de ieșiri digitale de pe Arduino ce vor fi energizați pe rând pentru a mișca motorul într-o direcție sau alta [14][15]. Experimental, am determinat un pas de aproximativ 1.63 de grade per pas, care la rândul său este regulat folosind datele primite de la MPU-6050.

Aceste două motoare sunt configurate conform diagramei de mai jos.

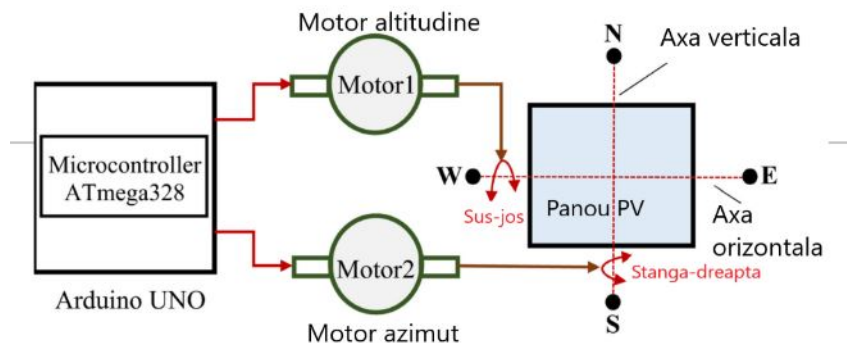


Figure 7: Diagrama schematică a mișcării sistemului dezvoltat pe două axe

Panoul fotovoltaic este conectat la un acumulator care are la bază 3 baterii reîncărcabile a câte 1.2V fiecare. Energia acumulată este arătată de un voltmetru, dar și transferată către interfața utilizatorului ca procentaj de încărcare.

Schema tuturor conexiunilor poate fi văzută în anexe.

3 Calculul poziției soarelui

Odată cu apariția microcontrolerelor Arduino și a hardware-ului asociat pentru controlul motoarelor, inclusiv a motoarelor pas cu pas, a existat un interes considerabil în utilizarea acestui dispozitiv ieftin cu sursă deschisă pentru a alimenta tracker-ele solare. Există două abordări de bază ale acestei probleme: utilizarea senzorilor de lumină căutând cea mai puternică sursă și calcularea poziției solare folosind ecuații astronomice care profită de faptul că, în principiu, poziția soarelui pe cer poate fi prezisă.

Implementarea unui astfel de sistem necesită o longitudine și o latitudine exacte și o poziționare corectă a sistemului - orizontală și aliniată cu nordul geografic (nu nordul magnetic).

Pentru o implementare reușită a poziției solare avem nevoie de a depista punctele principale pe care le vom folosi pentru orientarea sistemului nostru în spațiu: azimutul, zenitul și altitudinea [16].

Azimutul [17] este unghiul în plan orizontal format de planul meridianului unui loc cu planul vertical care trece prin locul respectiv și printr-un punct de referință. Această referință poate fi nordul geografic sau nordul magnetic.

În general, azimutul reprezintă distanța unghiulară dintre direcția Nord (care este definit diferit în funcție de subiectul în cauză) și direcția în care se încadrează perpendiculara unui punct la orizont.

Altitudinea [18] este înălțimea unui punct de pe suprafața Pământului în raport cu un alt punct de referință de pe suprafața terestră, denumită altitudine relativă.

Zenit [19] se numește punctul de intersecție al verticalei locului cu sfera cerească, situat deasupra observatorului (punctul cel mai înalt de pe bolta cerească). Punctul diametral opus zenitului este nadirul.

Aceste trei puncte de reper pot fi observate în figura de mai jos. În codul nostru ele vor determina valoarea unghiurilor salvate în variabilelor globale Azimuth și ElevationAngle.

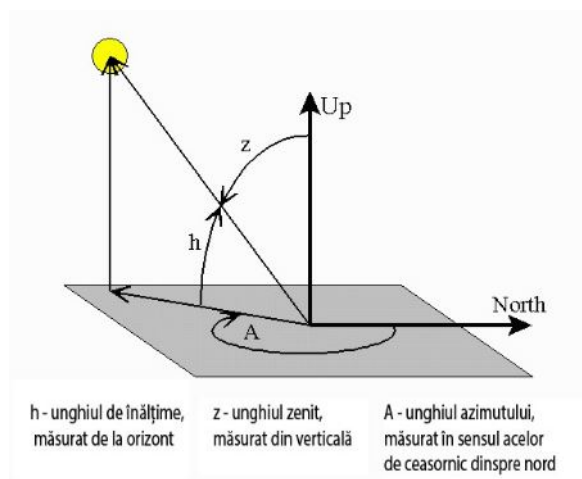


Figure 8: Calculul coordonatelor azimutale

Conform unghiurilor calculate vom mișca panoul fotovoltaic pe cele două axe pentru a găsi poziția optimă în dependență de variația zilnică și sezonieră a poziției soarelui. Scopul acestor mișcări e de a orienta panoul

perpendicular radiației solare. Această logică este definită în funcția loop() o dată la fiecare 15 minute. Mai multe informații despre calculele poziției solare pot fi găsite în anexă.

```
//Run sunrise and sunset calculations
sunTimeCalc();
//Run sun position calculations
sunPos();
//Check if the sun is up
if(CurrentTime >= sRise && CurrentTime <= sSet) {
    //Set rest period as false
    rest = false;
    //Check that 15 minutes have passed since last positioning
    if(CurrentTime - OldTime >= 15 * 60) {
        //Check if elevation angle has changed,
        if (OldElevationValue != ElevationAngle) {
            //Change elevation angle
            changeElevationAngle();
        }
        //Check if azimuth angle has changed
        if (OldAzimuthValue != Azimuth) {
            //Change azimuth angle
            changeAzimuthAngle();
        }
        //Update time of last positioning
        OldTime = CurrentTime;
    }
}
//Check if sun is down and PV panel is still active
else if (CurrentTime > sSet && rest == false) {
    //Change azimuth angle due east
    Azimuth = 90;
    changeAzimuthAngle();
    ElevationAngle = 75;
    changeElevationAngle();
    //Set rest period as true
    rest = true;
}
```

4 Instrumentul de dezvoltare Node-RED

Interfața proiectului am realizat-o cu ajutorul instrumentului de dezvoltare vizuală Node-Red [20]. Node-RED este un instrument pentru conectarea între dispozitive hardware, API-uri și servicii online în moduri noi și interesante. Node-RED face mai ușoară conectarea fluxurilor folosind nodurile din paletă. Fluxurile pot fi apoi implementate în runtime cu un singur clic.

De ce am ales Node-RED:

- **Prototipare rapidă.** Node-Red ne permite să creăm interfața rapid și flexibil, putem prototipa, testa, construi și reconstrui în câteva minute.
- **Cunoștințe limitate de programare.** Doar cu niște cunoștințe minime de programare sau o logică bună putem crea o mulțime de lucruri în mod foarte simplu.
- **Flexibilitate.** Putem urmări și schimba logica foarte rapid, fără a fi nevoie de multe modificări. Node-RED este construit pe baza uneia dintre cele mai fiabile și omniprezente stive de tehnologie - JavaScript. Acest lucru îl face extrem de flexibil și ușor de lucrat cu browser-ul web, partea de server, proiectele IoT și multe altele. Există peste 3000 de noduri gata pregătite pentru a începe să construim orice dorim. Aceste noduri ne permit să injectăm propriile date sau date din alte surse.

Figura de mai jos explică succint termenii de bază în mediul de lucru Node-RED.

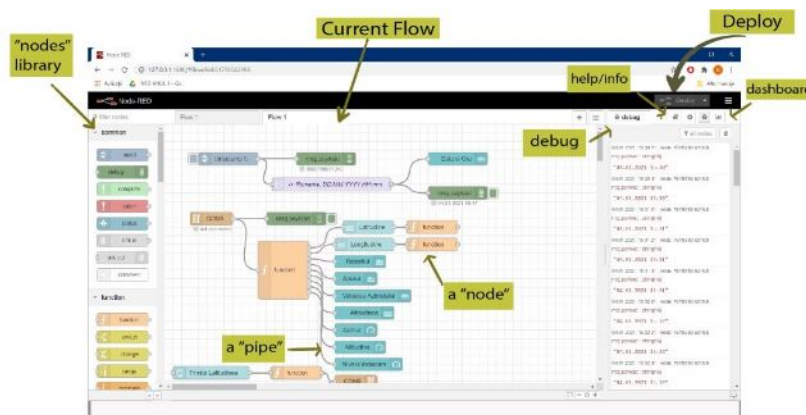


Figure 9: Termeni de bază în fluxul Node-RED

Primii pași pe care îi facem este conectarea plăcii Arduino la laptop, alegerea portului corect și rularea comenzii node-red în terminal. Copiem link-ul serverului local în browser (<http://localhost:1880>) și vedem o fereastră ca în figura de mai sus care arată lansarea Node-RED pe desktop.

Placa Arduino calculează valorile azimutului, latitudinii, longitudinii, altitudinea, apusul, răsăritul și nivelul de încărcare al acumulatorului și le trimite în serie utilizând portul serial COM5. Aceste informații seriale sunt apoi procesate de Node-RED.

Pentru a începe comunicarea cu Arduino avem nevoie de nodul de intrare serial. Acesta trebuie conectat la portul și rata de transmisie corecte.

Interfața utilizatorului ne permite să preluăm date din portul serial și să le afișăm pe ecran, precum și să modificăm datele cu care lucrează placa Arduino. Codul Arduino transmis tipărește pe serial un string de date concatenate.

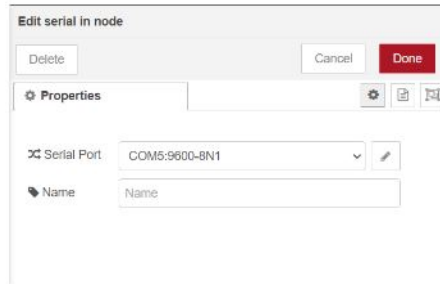


Figure 10: Editarea portului serial în Node-RED

```
//Concatenate string for serial transmission
String generateNodeRedString()
{
    char sunrise[6], sunset[6];
    snprintf(sunrise, sizeof(sunrise), "%02d:%02d", hour(sRise), minute(sRise));
    snprintf(sunset, sizeof(sunset), "%02d:%02d", hour(sSet), minute(sSet));
    String s="lat/";
    s+=String(Latitude);
    s+=";long/";
    s+=String(Longitude);
    s+=";rasarit/";
    s+=String(sunrise);
    s+=";apus/";
    s+=String(sunset);
    s+=";azimut/";
    s+=String(Azimuth);
    s+=";altitudinea/";
    s+=String(90 - ZenithAngle);
    s+=";azimutCurent/";
    s+=String(CurentAzimuth);
    s+=";altitudineCurenta/";
    s+=String(90 - CurentElevation);
    s+=";acumulator/";
    s+=String(Charge);
    return s;
}
```

Funcția principală care transformă datele primite în Node-RED primește un obiect JavaScript și returnează mai multe obiecte msg.payload drept răspuns care apoi sunt preluate de nodurile interfeței. Caracterul ‘ ; ’ împarte informația primită în grupuri de date, iar caracterul ‘ / ’ împarte grupurile în descrierea și valoarea propriu-zisă a acestora. Latitudinea și longitudinea sunt salvate ca variabile globale pentru a putea fi suprascrise ulterior.

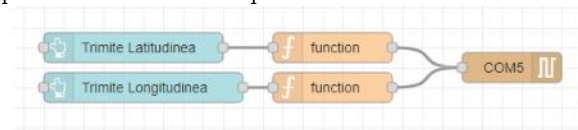

```

var output=msg.payload.split(";");
for(var i=0; i<9; i++){
  var newoutput= output[i].split("/");
  if(newoutput[0]=== "lat"){
    var lat = newoutput[1];
    global.set("lat", lat);
  } else if(newoutput[0]=== "long"){
    var long=newoutput[1];
    global.set("long", long);
  } else if(newoutput[0]=== "rasarit"){
    var rasarit=newoutput[1];
  } else if(newoutput[0]=== "apus"){
    var apus=newoutput[1];
  } else if(newoutput[0]=== "azimut"){
    var azimut=newoutput[1];
  } else if(newoutput[0]=== "altitudinea"){
    var altitudinea=newoutput[1];
  } else if(newoutput[0]=== "azimutCurent"){
    var azimutCurent=newoutput[1];
  } else if(newoutput[0]=== "altitudineCurenta"){
    var altitudineCurenta=newoutput[1];
  } else if(newoutput[0]=== "acumulator"){
    var acumulator=newoutput[1];
  }
}
return [{payload:lat},{payload:long},{payload:rasarit},{payload:apus},
{payload:azimut},{payload:altitudinea},{payload:azimutCurent},
{payload:altitudineCurenta},{payload:acumulator}];

```

Figure 11: Transformarea datelor primite în elemente UI

Node-RED permite rescrierea valorile latitudinii și longitudinii și transmite lor spre portul COM5 la apăsarea butonului respectiv.



În figura de mai jos avem funcțiile ce transmit datele din Node-RED spre Arduino:

<pre> 1 • if(msg.payload===true){ 2 return{ payload : "A"+global.get("lat")}; 3 • } 4 </pre>	<pre> 1 • if(msg.payload === true){ 2 return{ payload : "B"+global.get("long")}; 3 • } 4 </pre>
--	---

Funcția Arduino urmărește apariția pe serial a comunicării de la Node-RED în forma unui string. Se citește prima literă a string-ului care, la moment, poate fi 'A' sau 'B', și suprascrie latitudinea sau, respectiv, longitudinea curentă.

```

// Read and overwrite Latitude and Longitude with incoming data
void readLatitudeLongitude(){
  String incomingString="";

  if(Serial.available()){
    char d = Serial.read();
    while (Serial.available() {
      delay(3);
      if (Serial.available() > 0) {
        char c = Serial.read();
        incomingString += c;
      }
    }
    if(d == 'A'){
      Latitude = incomingString.toFloat();
    } else if(d == 'B'){
      Longitude = incomingString.toFloat();
    }
  }

  incomingString = "";
}

```

Fluxul complet și un exemplu de comunicare bilaterală între Node-RED și Arduino pot fi observați în figurile de mai jos.

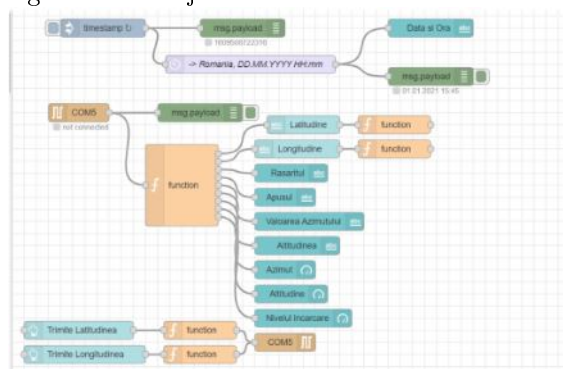


Figure 12: Fluxul proiectului

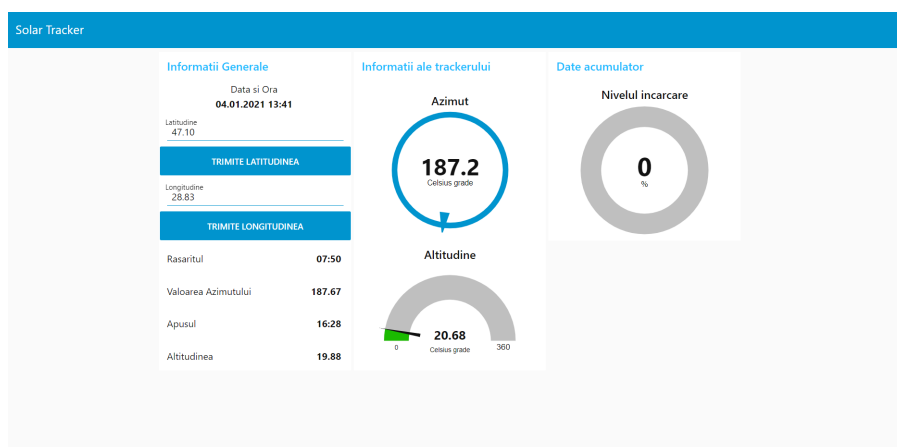


Figure 13: Exemplu efectiv de interfatare a plăcii Arduino cu Node-RED
Pentru mai multă informație privind instalarea instrumentului și bibliotecile folosiți anexa.

5 Unități funcționale ale codului

Mediul de dezvoltare Arduino IDE este o aplicație multi-platformă scrisă cu ajutorul limbajelor de programare Java, C și C++. Am folosit acest mediu pentru a scrie și încărca programe pe placa Arduino.

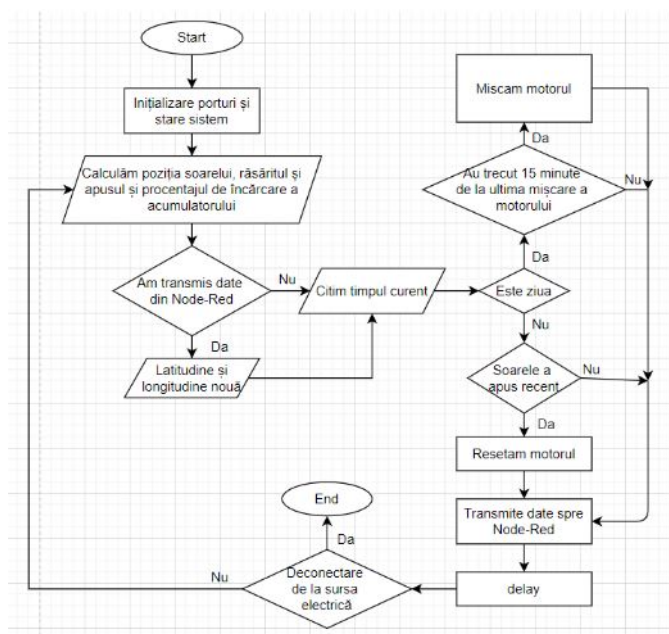
În prima parte a programului nostru includem bibliotecile necesare, ale căror comenzi le folosim, setăm pinii cu care lucrăm și definim obiectele și variabilele globale.

A doua parte a programului conține comenzi executate o singură dată la pornire. Comenzile includ inițierea comunicării cu portul serial COM, definirea pinurilor de ieșire, inițializarea comunicării cu adresele I2C, setarea altitudinii și timpului curent și a vitezei motorului.

A treia parte conține o buclă de calcul care citește valorile trimise de la interfața utilizatorului, calculează valorile poziției soarelui și diferența cu unghiurile anterioare salvate, și modifică înclinarea panoului la fiecare 15 minute. Valorile altitudinii curente sunt obținute cu ajutorul accelerometrului, iar valoarea azimutului reprezintă incrementarea sau decrementarea poziției curente prin intermediul pasului motorului.

Ultima parte a programului poate fi considerată transmiterea în timp real a datelor spre interfața utilizatorului.

Schema logică completă poate fi urmărită mai jos:



Codul disponibil pe Github la adresa: https://github.com/Nyubt/UniProjects/tree/master/Project_EA_final?fbclid=IwAR2J7eGfVZQOfv-7-cIJ3rIUTLEZER9P8AMGUzcAaCggGtOe963PVXok_hM

6 Informații adăugătoare despre proiect

Problemele majore cu care ne-am confruntat:

1. Crearea unui design optim – a fost prima problemă pe care am întâlnit-o, întrucât design-ul și piesele necesare disponibile sunt interdependente. Limita stocurilor disponibile ne-a obligat să schimbăm design-ul de câteva ori.
2. Defectarea pieselor mecanice – Scopul de a avea costuri cât mai reduse, dar și disponibilitatea acestora, ne-a constrâns în alegerea pieselor. Aceasta a rezultat în faptul că unele piese nu au fost fiabile și s-au defectat ușor.

3. Lipsa pieselor necesare – După defectarea acestora am fost nevoiți să căutăm piese de schimb și, ca rezultat, am fost nevoiți să schimbăm design-ul proiectului de câteva ori.
4. Probleme de conexiune – Printre problemele de conexiune putem număra portul serial și motoarele reciclate. Inițial am folosit o plăcuță Arduino Mega 2560 care arunca eroare la conectarea portului. La început, deconectarea și reconectarea plăcuței sau reîncărcarea calculatorului soluționa această problemă, însă, în final, calculatorul a încetat să recunoască plăcuța și să deschidă porturile 3 și 4. Iar motoarele pe care încercam să le folosim ca piese de schimb erau reciclate din aparate vechi la care nu mai poți găsi datasheet și procesul de conectare a acestora era trial and error.

Cum putem îmbunătăți proiectul:

1. Wi-Fi/cablu lung (nu trebuie sa fie mereu conectat la calculator) ceva similar am mentionat in introducere
2. Magnetometru (HMC5883 sau MPU9250) (pentru a calcula azimutul obținând mai puține erori)
3. Momentan – întrerupător
4. Îmbunătățirea ui: transmiterea datelor locale și fusului orar
5. Modul GPS – intoarce timp, lat, long (nu mai avem nevoie de RTC)

Prețul proiectului

<i>Placa Arduino UNO</i>	200 mdl
<i>Breadboard 400 puncte</i>	45 mdl
<i>Fire mama-tata 20 cm</i>	30 mdl
<i>Fire tata-tata 20 cm</i>	30 mdl
<i>Modulul Driver Stepper ULN2003 cu motor Stepper 28BYJ-48</i>	100 mdl
<i>Modul giroscop + accelerometru 3 pe axe</i>	45 mdl
<i>Modul RTC</i>	50 mdl
<i>Motor NEMA 23</i>	reciclat
<i>Acumulator cu baterii</i>	reciclat
<i>Panou fotovoltaic</i>	reciclat
<i>Voltmetru/ampermetru</i>	reciclat
<i>Baza pentru construcție</i>	reciclată
<i>Preț total</i>	500 mdl (~116 ron)

7 Concluzii

Datele primite au fost testate cu predicțiile prin satelit oferite de Heavens-Above GmbH22, cât și un sistem de coordonate polare alipit de baza proiectului. Tabelul din anexă compară poziția solară, unghiurile azimutului și altitudinii calculate și unghiurile de facto a sistemului proiectat pentru data de 08.01.2021. Calcululele și mișcarea sistemului sunt făcute la fiecare 15 minute. Considerând datele urmărite putem prezenta următoarele concluzii:

1. Testele inițiale au arătat că sistemul deviază cu o mică eroare de la poziție optimă în decursul zilei datorită pasului motorului inferior care are o variație de până la 0.5%. În scopul următoarelor măsurări, poziția inițială a azimutului poate fi setată manual;

2. Datele măsurate folosind cipul MPU-6050 au o deviație de până la un grad în dependență de pasul motorului superior și fluctuațiile senzorului, care poate avea o deviere de aproximativ un grad. Pentru o analiză mai precisă a devierilor senzorului MPU-6050 este nevoie de un studiu separat;
3. Chiar având o precizie redusă a calculelor matematice, diferența cele două calcule este mai mică de 1.5 grade.

8 References

- [1] Pentru mai multe informații despre sistemele de urmărire solară priviți anexa
- [2] <https://ww1.microchip.com/downloads/en/DeviceDoc/doc7799.pdf>
- [3] <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>
- [4] <https://datasheetspdf.com/pdf-file/1081920/MaximIntegrated/DS3231/1>
- [5] <https://howtomechatronics.com/tutorials/arduino/how-i2c-communication-works-and-how-to-use-it-with-arduino/>
- [6] <https://www.arduino.cc/en/reference/wire>
- [7] DS3231 - Rinky-Dink Electronics
- [8] Aceștia pot fi gasiți pe placa folosind schema <https://riptutorial.com/arduino/example/15314/arduino-uno-r3>
- [9] <https://components101.com/motors/nema-23-stepper-motor-datasheet-specs>
- [10] <https://www.makerguides.com/wp-content/uploads/2019/04/28byj48-Stepper-Motor-Datasheet.pdf>
- [11] https://www.arduino.cc/documents/datasheets/H-bridge_motor_driver.PDF
- [12] <https://www.arduino.cc/en/reference/stepper>
- [13] <https://www.electronicoscaldas.com/datasheet/ULN2003APCB.pdf>
- [14] Vezi în anexe funcția `stepUpperMotor()`
- [15] http://www.idc-online.com/technical_references/pdfs/electrical_engineering/Step_Sequence_of_Stepper_Motor.pdf
- [16] <http://www.instesre.org/Solar/insolation.htm>
- [17] <https://ro.wikipedia.org/wiki/Azimut>
- [18] <https://ro.wikipedia.org/wiki/Altitudine>
- [19] <https://ro.wikipedia.org/wiki/Zenit>
- [20] <https://nodered.org/>
- [21] https://en.wikipedia.org/wiki/Arduino_IDE
- [22] https://en.wikipedia.org/wiki/Solar_tracker
- [23] <https://sinovoltaics.com/learning-center/csp/active-trackers-and-passive-trackers/>
- [24] <https://www.solarpowerworldonline.com/2020/01/what-is-a-solar-tracker-and-how-does-it-work/>
- [25] <https://www.esrl.noaa.gov/gmd/grad/solcalc/solareqns.PDF>

A

Anexa 1. Sisteme de urmărire solară

Energie regenerabilă este o nevoie care crește în fiecare zi, în special nevoia de energie "verde". Cercetările au dovedit că un sistem de urmărire solară poate maximiza generarea energiei până la 50%. Puterea maximală de generare poate fi extrasă când unghiul de înclinare a panoului este sincronizat cu modificările zilnice și sezoniere a poziției soarelui [22].

Sistemele de urmărire solară pot fi clasificate în: sisteme de urmărire pasive (mecanism mecanic care folosește un gaz comprimat pentru a mișca panoul) și active (care folosesc componente electrice pentru a direcționa panoul) [23]. Ambele pot fi folosite într-o structură de urmărire pe o singură axă sau pe o axă dublă.

Un sistem de urmărire solară pe o singură axă urmărește mișcarea soarelui într-o singură direcție (orizontal, de la sud spre nord care e calea sezonieră a soarelui; sau vertical, de la est spre vest, care e calea zilnică a soarelui).²⁴ Pentru a câștiga putere suplimentară se folosesc sistemele de urmărire duale, care au atât axă verticală, cât și orizontală.

Sistemele pasive sunt relativ simple și funcționează fără comenzi electronice și motoare. Cu toate acestea, precizia este limitată și nu poate funcționa la temperaturi scăzute. Avantajul sistemelor active, deci, este că au o precizie mai bună de urmărire.

Sistemele de urmărire active pot fi clasificate în două categorii după principiul lor de funcționare: sisteme de urmărire astronomice și bazate pe senzori. Urmăritorii solari astronomici funcționează pe baza abordărilor care calculează poziția soarelui din ecuații geometrice și astronomice predefinite. Cu toate acestea, această abordare de urmărire necesită intervenție manuală pentru a modifica latitudinea, data locală și fusul orar.

Pe de altă parte, dispozitivele de urmărire solară bazate pe senzori (tipic 4) utilizează pe scară largă senzori de lumină, cum ar fi fotorezistoare, fotodiode, piometre pentru a urma instantaneu mișcarea soarelui. Această abordare necesită hardware puternic pentru a gestiona senzorii de lumină. De asemenea, este predispus la erori dacă există surse de lumină neprevăzute, cât și în zilele înorate.

Pe lângă acestea, au fost introduse metode noi pentru urmărirea solară folosind inteligența artificială, cum ar fi: logica fuzzy, rețelele neuronale și neuro-fuzzy.

Dotarea unui panou fotovoltaic cu un sistem de urmărire solară va crește cu siguranță costul sistemului. În acest sens, este necesar un compromis între eficiență și cost pentru ca sistemele de urmărire solară să devină competitive.

B

Anexa 2. Calcule generale ale poziției solare

În primul rând, se calculează anul fracționat (γ), în radiani. (Pentru anii bisecți, se utilizează 366 în loc de 365 la numitor)

Din γ , putem estima ecuația timpului (în minute) și unghiul de declinare solară (în radiani)

$$eqtime = 229.18 * (0.000075 + 0.001868 \cos(\gamma) - 0.032077 \sin(\gamma) - 0.014615 \cos(2\gamma) - 0.040849 \sin(2\gamma)) \quad (1)$$

Apoi, timpul solar real este calculat în următoarele două ecuații. Mai întâi, decalajul de timp se găsește, în minute, și apoi timpul solar, în minute.

$$timeoffset = eqtime + 4 * longitude - 60 * timezone \quad (2)$$

unde $eqtime$ este în minute, longitudinea este în grade (pozitivă la est de meridianul principal), fusul orar este în ore de la UTC (U.S. Mountain Standard Time = - 7 ore)

$$tst = hr * 60 + mn + sc/60 + timeoffset \quad (3)$$

unde hr este ora (0 - 23), mn sunt minutele (0 - 59), sc sunt secunde (0 - 59) . Unghiul orar solar, în grade, este:

$$ha = (tst/4) - 180 \quad (4)$$

Unghiul zenit solar poate fi apoi găsit din unghiul orar (ha), latitudinea (lat) și declinarea solară (decl) folosind următoarea ecuație:

$$\cos(z) = \sin(lat)\sin(decl) + \cos(lat)\cos(decl)\cos(ha) \quad (5)$$

Iar azimutul solar (grade în sensul acelor de ceasornic din nord) se găsește din:

$$\cos(180 - teta) = (\sin(lat)\sin(decl) + \cos(lat)\cos(decl))/\cos(lat)\sin(decl) \quad (6)$$

Pentru cazul special al răsăritului sau apusului, zenitul este setat la $90,833^\circ$ (corecția aproximativă pentru refracția atmosferică la răsărit și apus și dimensiunea discului solar), iar unghiul orar devine:

unde numărul pozitiv corespunde răsăritului, iar cel negativ - apusului.

Apoi, ora UTC a răsăritului (sau apusului) în minute este:

$$sunrise = 720 - 4 * (longitude + ha) * eqtime \quad (7)$$

unde longitudinea și unghiul orar sunt în grade și ecuația timpului este în minute.

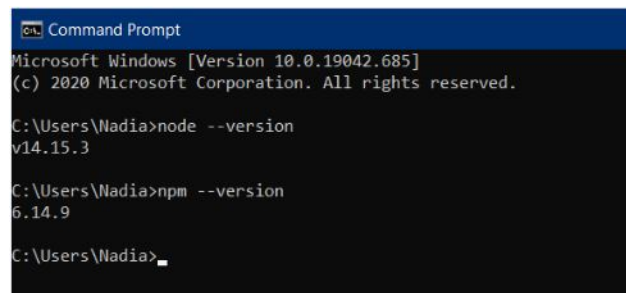
Amiaza solară pentru o anumită locație se găsește de la longitudine (în grade, pozitivă la est de Meridianul principal) și ecuația timpului (în minute):

$$snoon = 720 - 4 * longitude * eqtime \quad (8)$$

C

Anexa 3. Pașii de instalare a tool-ului pentru programare vizuală Node-RED

- Descărcăm ultima versiune de Node.js de pe pagina oficială <https://nodejs.org/en/>
- Odată instalat, deschidem linia de comandă și rulăm următoarele comenzi pentru a ne asigura că Node.js și npm sunt instalate corect.



```

C:\Users\Nadia>node --version
v14.15.3

C:\Users\Nadia>npm --version
6.14.9

C:\Users\Nadia>

```

Figure 14: Linia de comandă

- Deschidem linia de comandă și instalăm Node-RED folosind următoarele comenzi: `node -version` & `npm -version`
- Rulăm Node-RED folosind comanda: **node-red**

```
Select node-red
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Nadia>node-red
6 Jan 15:36:58 - [info]

Welcome to Node-RED
=====
6 Jan 15:36:58 - [info] Node-RED version: v1.2.6
6 Jan 15:36:58 - [info] Node.js version: v14.15.3
6 Jan 15:36:58 - [info] Windows_NT 10.0.19042 x64 LE
6 Jan 15:36:59 - [info] Loading palette nodes
6 Jan 15:37:00 - [info] Dashboard version 2.26.1 started at /ui
6 Jan 15:37:00 - [info] Settings file : C:\Users\Nadia\.node-red\settings.js
6 Jan 15:37:00 - [info] Context store : 'default' [module=memory]
6 Jan 15:37:00 - [info] User directory : \Users\Nadia\.node-red
6 Jan 15:37:00 - [warn] Projects disabled : editorTheme.projects.enabled=false
6 Jan 15:37:00 - [info] Flows file : \Users\Nadia\.node-red\flows_WIN-A07UGEM8PU0.json
6 Jan 15:37:00 - [info] Server now running at http://127.0.0.1:1880/
6 Jan 15:37:00 - [warn]

-----
Your flow credentials file is encrypted using a system-generated key.
```

Figure 15: Rularea Node-RED

- Copiem linkul serverului local in browser (Chrome): <http://localhost:1880/>
- Node-RED nu are noduri preinstalate pentru Serial Port, Dashboard și Moment. Pentru a le instala sunt 2 metode:
 1. Utilizăm următoarele comenzi pentru a instala nodurile respective:
 - npm install node-red -node-serialport
 - npm install node-red -dashboard
 - npm install node-red -contrib-moment
 2. Instalăm direct din Node-RED în modul următor (din menu alegem Manage palette)

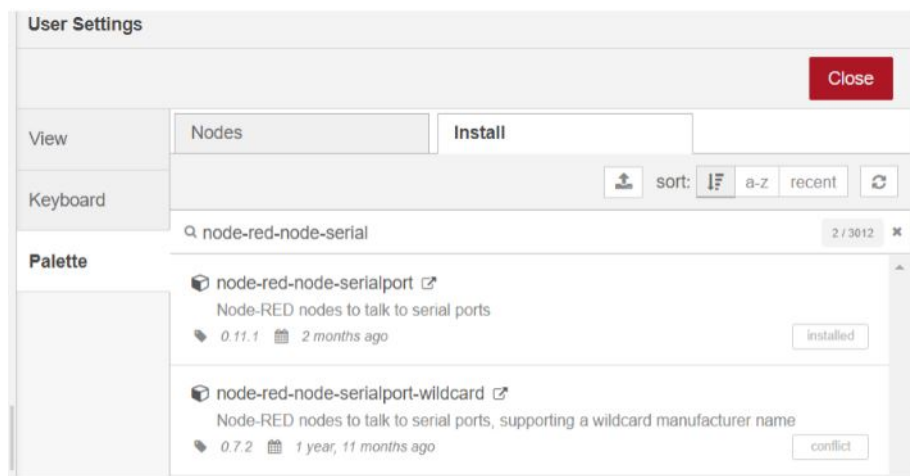







Figure 16: Meniul de biblioteci Node-RED disponibile

C.1 Bibliotecile Node-RED folosite în proiect

1. node-red

 debug	Mesajele au de obicei o proprietate de încărcare - aceasta este proprietatea implicită cu care vor funcționa majoritatea nodurilor. Node-RED adaugă, de asemenea, o proprietate numită <code>_msgid</code> - acesta este un identificator pentru mesaj care poate fi folosit pentru a-i urmări progresul printr-un flux.
 inject	Nodul inject poate iniția un flux cu o valoare specifică. Sarcina utilă implicită este o marcă de timp a timpului curent în milisećuri de la 1 ianuarie 1970.
 function	Funcția returnează multiple message objects.
 ui_button	Adaugă un buton pentru interfața cu utilizatorul. Dacă facem clic pe buton, se generează un mesaj cu <code>msg.payload</code> setat în câmpul Payload.

2. node-red-contrib-moment

 moment	Convertește șirul de dată/oră în obiect dată/oră.
---	---

3. node-red-dashboard



 ui_text	Afișează un câmp care nu poate fi editat pe interfața utilizatorului.
 ui_gauge	Adaugă un widget de tip ecartament pe interfața cu utilizatorul. Am folosit-o pentru a afișa valoarea azimutului, altitudinea și nivelul de încărcare a utilizatorului. Ecartamentul are mai multe moduri. Ecartament regulat, gogoasă, busolă și val.



Figure 17: Exemple de module de encartament

4. node-red-node-serialport

 serial in	Citește datele de la un port serial local.
---	--

D Anexa 4. Calculul, răspunsul sistemului de urmărire solară și schema de conexiune fritzing a proiectului

Date verificate cu ajutorul: <https://www.heavens-above.com/sun.aspx>

Latitudine: 47.0979

Longitudine: 28.8246

Data: 08.01.2021

Răsărit: 07:50

Apus: 16:33

Ora	Poziție azimut	Poziție altitudine	Azimut calculat	Altitudine calculată	Azimutul sistemului	Altitudinea sistemului
08:00	124.5	0.6	124.55	0.68	124.2	1.26
08:15	127.3	2.7	127.34	2.74	126.0	3.19
08:30	130.2	4.7	130.19	4.74	129.6	4.34
08:45	133.1	6.6	133.1	6.64	131.4	6.46
09:00	136.1	8.4	136.09	8.46	135.0	7.87
09:15	139.1	10.1	139.14	10.18	138.6	10.31
09:30	142.2	11.7	142.27	11.8	142.2	11.51
09:45	145.4	13.2	145.48	13.31	144.0	13.26
10:00	148.7	14.6	148.76	14.69	147.6	14.47
10:15	152.1	15.9	142.11	15.95	151.2	15.73
10:30	155.5	17.0	155.54	17.08	154.8	18.5
10:45	159.0	18.0	159.03	18.07	158.4	17.61
11:00	162.5	18.9	162.58	18.91	162.0	18.2
11:15	166.1	19.6	166.19	19.6	165.6	19.63
11:30	169.8	20.1	169.83	20.13	169.2	19.7
11:45	173.5	20.5	173.52	20.5	172.8	20.6
12:00	177.2	20.7	182.78	20.71	181.8	20.6
12:15	180.9	20.7	180.93	20.75	181.8	20.6
12:30	184.6	20.6	184.64	20.63	183.6	20.6
12:45	188.3	20.3	188.34	20.34	187.2	20.6
13:00	191.9	19.9	192.0	19.89	1900.8	19.19
13:15	195.6	19.3	195.63	19.28	194.4	19.19
13:30	199.1	18.5	199.21	18.52	198.0	18.69
13:45	202.7	17.6	202.74	17.61	201.6	18.08
14:00	206.1	16.5	206.2	16.55	205.2	15.69
14:15	209.5	15.3	209.59	15.36	208.8	15.69
14:30	212.8	14.0	212.91	14.04	212.4	13.65
14:45	216.1	12.6	216.15	12.59	216.0	12.91
15:00	219.3	11.0	219.32	11.03	217.8	11.53
15:15	222.3	9.4	22.42	9.36	221.4	9.67
15:30	225.4	7.6	225.44	7.59	225.0	7.86
15:45	228.3	5.7	228.39	5.73	226.8	5.38
16:00	231.2	3.8	231.28	3.78	230.4	2.85
16:15	234.0	1.8	234.1	1.75	234.0	2.05
16:30	236.8	-0.3	236.86	-0.35	235.8	1.35

