Sistem Cartografiere Sol - Profeanu Ioana 333CA

Introducere

Proiectul constă într-un sistem inteligent de monitorizare și înregistrare a diferșilor parametri necesari dezvoltării plantelor, date care, utilizând o bază de date predefinită (stocată pe un card micro SD) cu plante și condițiile lor de mediu, pot fi interogate de utilizator, acesta putând să vadă o listă a plantelor viabile, să vadă toate plantele din baza de date, să afiseze sau reseteze media metricilor.

Sistemul este extrem de util atât pentru cei din domeniul agriculturii, pentru a identifica plantele ce pot crește într-un anumit tip de sol și mediu, precum și pentru cei care își doresc să cultive diverse plante în grădină și vor să afle ce opțiuni au, sau să vadă daca o anumită plantă poate supraviețui condițiilor.

Ideea a pornit de la nevoia de a cultiva plante într-un sol și mediu care să fie propice dezvoltării lor, sistemul ținând cont nu doar de climă, ci și de factorii impredictibili de mediu (secetă, inundații etc).

Descriere generală

Inițial, sistemul se află în starea de stand by, unde afișează date pe ecran despre condițiile de mediu și coordonatele GPS. Acestea sunt recalculate la interval de o oră și adăugate unei medii generale. Tot atunci, mediile generale ale factorilor de meniu sunt inregistrați într-un fișier de log pe cardul SD.

La apăsarea butonului joystick-ului, se afișează 5 opțiuni de meniuri, fiecare apărând pe ecran la interval de 6 secunde. Selectarea meniului dorit se face apăsând din nou pe buton, în timp ce meniul dorit este afișat pe ecran.

Meniurile sunt:

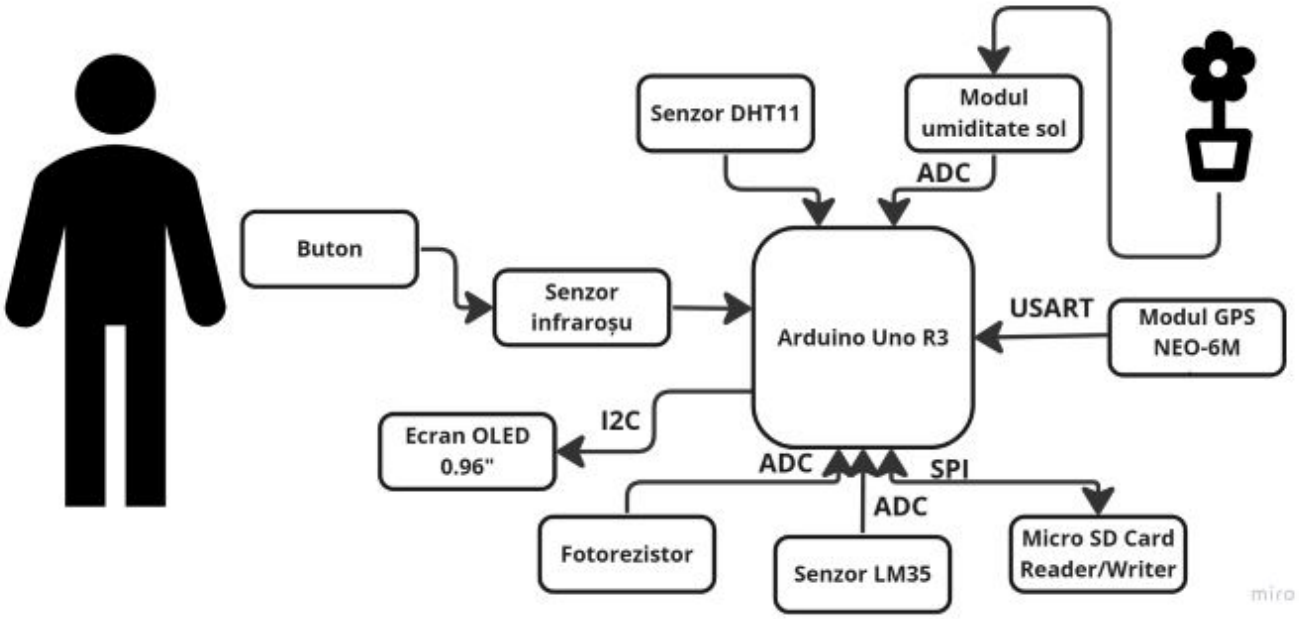
1. **Show valid plants list** : afișează plantele ce pot fi plantate în mediul dat. Plantele sunt afișate la interval de 5 secunde, alături de datele despre mediul în care acestea trăiesc.
2. **Show all plants** : afișează toate plantele stocate pe cardul SD.
3. **Reset metrics** : resetează mediile metricilor.
4. **Show average** : afișează mediile metricilor.
5. **Return to stats** : reîntoarcere la meniul de stand-by (și recalculare a factorilor de mediu).

După o perioadă de inactivitate de 3 minute, sistemul trece inapoi la starea de stand by.

Atunci când se afișează plantele ce respectă condițiile de mediu, se vor căuta întâi după clima dedusă din coordonatele GPS, iar apoi după ceilalți parametrii de mediu (temperatura, umiditate aer și sol, luminozitate). Baza de date cu plante este stocată pe un card micro SD, pentru ca utilizatorul să poată

adăuga baze de date custom.

Schema bloc



Hardware Design

Lista componentelor folosite:

- Arduino Uno R3
- Senzor de temperatura și umiditate DHT11
- Senzor de temperatura LM35
- Fotorezistor
- Modul umiditate sol
- Modul GPS NEO-6M cu antenă
- Ecran OLED 0.96”
- Adaptor card Micro SD
- Modul Joystick
- Rezistență 220Ω
- Breadboard 800 puncte
- Baterie 9V

Deși senzorul DHT11 este și senzor de temperatură, din cauza faptului că poate măsura doar valori între 0 - 50°C, am decis ca temperatura să o măsor cu ajutorul senzorului LM35, acesta având valori între −55 - 150°C, lucru de dorit având în vedere că sistemul este distinat utilizării indiferent de climă.

Software Design

În cadrul proiectului, am folosit cunoștințele următoarele biblioteci:

- Wire.h, SSD1306AsciiWire.h - pentru conexiunea ecranului OLED
- SD.h - pentru utilizarea cardului SD
- dht.h - pentru citirea datelor de pe senzorul DHT11
- SoftwareSerial.h - pentru citirea datelor primite de la modulul GPS

Flow program

În set-up, se inițializează timer-ul, întreruperea butonului, ecranul OLED și se apelează funcția de citire și afișare a datelor de mediu pe ecran. Tot atunci se și loghează pe cardul SD metricile citie.

În loop, se verifică flag-ul pentru întreruperea pe butonul joystick-ului. În acest caz, se pornește afișarea meniurilor o data la 6 secunde. Dacă se depășesc 3 minute de la ultimul input al user-ului, recitesc datele de mediu și se revine la afișarea lor.

Dacă în fereastra de 3 minute user-ul apasa din nou pe buton, se va face acțiunea afișată pe display în acel moment (se verifică displayFrame-ul curent la momentul apăsării).

În lipsa vreunui input de la utilizator, se verifică dacă au trecut 60 de minute de la ultima citire a datelor, iar daca da, se face recitirea.

Programul face diverse verificări și afișări de mesaje pe ecran în cazurile speciale/de eroare, precum căutarea unei plante după resetarea metricilor, imposibilitatea de a citi cardul SD (de exemplu, dacă acesta nu este introdus) și dacă nu există nicio plantă care să poată fi plantată în mediul curent.

Configurare timer

Deoarece în program am nevoie să contorizez diferite acțiuni (recalcularea metricilor o data la o oră, afișarea unui nou meniu la 6 secunde distanță, revenirea la meniul de stand-by in cazul neinteracțiunii user-ului timp de 3 minute), am ales să configurez un timer utilizând Timer1, ce numără aproximativ 2 secunde, folosind un prescaler de 1024 și CTC, utilizând apoi mai multe variabile globale care contorizează de câte ori au trecut cele 2 secunde.

```
// timer contorizare durata de afisaj a statisticilor în stand-by
volatile unsigned long int timerDisplayInfo = 0;
// timer contorizare timp de la ultimul input al user-ului
volatile int timerUser = 0;
// contorizare timpul unui frame
volatile int frameTime = 0;
```

```
ISR(TIMER1_COMPA_vect) {
  timerDisplayInfo++;
  if (timerUser != -1) {
    timerUser++;
    frameTime++;
  }
}

void configure_timer1() {
  // initialize registers
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1 = 0;

  // set for 2 seconds
  OCR1A = 31312;
  TCCR1B |= (1 << WGM12);
  // set prescaler to 1024
  TCCR1B |= (1 << CS12) | (1 << CS10);
        // enable timer compare interrupts
        TIMSK1 |= (1 << OCIE1A);
}
```

Apoi, în loop, variabilele globale de contorizare a timpului sunt verificate și în funcție de acestea sunt afișate datele pe ecranul OLED / se execută diverse acțiuni.

Configurare întrerupere buton joystick

Pentru a detecta apăsarea butonului joystick-ului în cadrul meniului, am utilizat întreruperea externă pe Pinul 2. Utilizez, de asemenea, un sistem de debouncing, pentru a detecta o singură apăsare. Întervalul de debouncing a fost setat pe 300ms.

```
volatile bool buttonState = LOW;
volatile bool lastButtonState = LOW;
volatile unsigned long lastDebounceTime = 0;
volatile bool buttonPressed = false;

ISR(INT0_vect) {
  // read the current button state
  bool currentState = digitalRead(2);

  // Check if the button state has changed
  if (currentState != lastButtonState) {
    // Reset the debounce timer
    lastDebounceTime = millis();
  }
```

```
  // Update the button state
  lastButtonState = currentState;

  // Check if the debounce delay has passed
  if (millis() - lastDebounceTime > 300) {
    // Update the buttonPressed flag if the button is pressed
    if (lastButtonState == LOW) {
      buttonPressed = true;
    }
  }
}

void configure_external_intrerrupt()
{
  // configure external interrupt on falling edge for INT0 (digital pin 2)
  EICRA |= (1 << ISC01);
  EICRA &= ~(1 << ISC00);

  // enable external interrupt for INT0
  EIMSK |= (1 << INT0);
}
```

Utilizarea cardului SD

Pentru lucrul cu cardul SD, am utilizat biblioteca SD.h, ce pune la dispoziție funcții de bază pentru scrierea și citirea datelor de pe cardul SD.

Scrierea metricilor medii

La fiecare recalculare a metricilor, mediile datelor citite sunt stocate și pe cardul SD într-un fișier de log numit metrics.txt, inserându-se media temperaturii, a umitității aerului și solului, a luminozității și numărul de citiri.

```
// function which writes the average metrics to sd card
void writeMetricsToSD()
{
  if (SD.begin(7)) {
    File myFile;

    myFile = SD.open("metrics.txt", FILE_WRITE);
    if (myFile) {
      myFile.print("----- Medium metrics (");
      myFile.print(timesRead);
      myFile.println(" times read data) -----");
```

```
      myFile.print("Temperature: ");
      myFile.print(sumTemps / timesRead);
      myFile.println(" C");
      myFile.print("Air Moisture: ");
      myFile.print(sumAirMoist / timesRead);
      myFile.println("%");
      myFile.print("Soil Moisture: ");
      myFile.println(getValueOfMetrics(sumSoilMoist / timesRead));
      myFile.print("Luminousity: ");
      myFile.println(getValueOfMetrics(sumLight / timesRead));
      if (climate != INVALID_CLIMATE) {
        myFile.print("Climate: ");
        myFile.println(getClimateFromMetric(climate));
      }
      myFile.close();
    } else {
      Serial.println("Couldn't create file.");
    }
  } else{
    Serial.println("Initialization failed.");
  }
}
```

Citirea plantelor

Pe cardul SD, datele sunt stocate în fișierul plants.txt, fiecare linie reprezentând datele unei plante. Modul de stocare este:

Nume, Clima, MinTemp, MaxTemp, MinUmidAer, MaxUmidAer, MinUmidSol, MaxUmidSol, MinLumina, MaxLumina

Exemplu:

Lavender, 3, 10, 30, 30, 60, 1, 3, 1, 3

Temperatura este măsurată în ºC, iar umiditatea aerului în procente.

Datele cu valori între 1-4 au următoarele semnificații:

Valoare	Climă	Umiditate Sol	Luminozitate
1	Tropical	LOW	LOW
2	Subtropical	MEDIUM	MEDIUM
3	Temperate	HIGH	HIGH
4	Polar		

Pentru modul de afișare a plantelor ce pot trăi în mediul curent, se compară valorile medii pentru fiecare dintre categoriile de metrici, primul criteriu de comparare fiind clima, afișându-se doar

plantele care se încadrează în parametrii dați.

Pentru modul de afișare a tututor plantelor, se afișează pe ecran toate plantele, fără a face vreo filtrare.

```
// functions which displays the valid plants
// on the display
void displayPlantsList(bool showAll)
{
  display.clear();
  display.setCursor(0, 0);
  // try to read from SD card
  if (SD.begin(7)) {
    File myFile;
    if (SD.exists("plants.txt")) {
      myFile = SD.open("plants.txt");
      // if the file exists, read from it
      if (myFile) {
        int noPlantsOk = 0;
        // parse the data of the plant by extracting it;
        // each data is delimited by a comma
        while (myFile.available()) {
          bool okPlant = false;
          String line = myFile.readStringUntil('\n');
          String name;
          int values[9];
          int index = 0;
          // read line
          while (line.length() > 0) {
            int commaIndex = line.indexOf(',');
            if (commaIndex != -1) {
              String element = line.substring(0, commaIndex);
              line = line.substring(commaIndex + 2);

              if (index == 0) {
                name = element;
              } else {
                values[index - 1] = element.toInt();
                // check if the plant is between parameters
                if (showAll == false) {
                  if (index == 1) {
                    if (climate != INVALID_CLIMATE) {
                      if (values[0] != climate) {
                        break;
                      }
                    }
                  }

                  if (index == 3) {
                    int mediumTemp = sumTemps / timesRead;
```

```
                    if ((mediumTemp >= values[1] && mediumTemp <= values[2])
                        == false) {
                      break;
                    }
                  }

                  if (index == 5) {
                    int mediumAirMoist = sumAirMoist / timesRead;
                    if ((mediumAirMoist >= values[3] && mediumAirMoist
                        <= values[4]) == false) {
                      break;
                    }
                  }

                  if (index == 7) {
                    int mediumSoilMoist = sumSoilMoist / timesRead;
                    if ((mediumSoilMoist >= values[5] && mediumSoilMoist
                        <= values[6]) == false) {
                      break;
                    }
                  }
                }
              }

              index++;
            } else {
              // read last value from line
              String element = line;
              values[index - 1] = element.toInt();
              int mediumLight = sumLight / timesRead;
              if (showAll == false && (mediumLight >= values[7]
                  && mediumLight <= values[8]) == false) {
                break;
              }

              okPlant = true;
              noPlantsOk++;
              break;
            }
          }
          // if a plant was found, display it
          if (okPlant) {
            displayPlantDetails(name, values);
          }
        }
        // if no plants were found
        if (noPlantsOk == 0) {
          display.println("No plants matched!");
          display.displayRows();
          delay(8000);
```

```
        }
        myFile.close();

      } else {
        // if the file is not found
        display.println("Add plants.txt file first!");
        display.displayRows();
        delay(8000);
      }
    }
  } else {
    // if unable to read file
    display.println("Failed to display plants!");
    display.displayRows();
    delay(8000);
  }
}
```

Parsare date GPS

Pentru a utiliza modulul GPS NEO-6M, am folosit biblioteca SoftwareSerial.h. Deși în mod normal modulul utilizează USART, nu am folosit pinii RX și TX Arduino-ului, deoarece acest lucru m-ar fi împiedicat din a face debugging utilizând afișarea serială. Am decis, de asemenea, să îmi implementez propria funcție de parsare a input-ului primit de la GPS, din considerente de memorie (introducerea unei biblioteci adiționale ar fi costat memorie), dar și pentru că doream să extrag doar latitudinea din input-ul primit (doar de aceasta este nevoie pentru a determina clima).

GPS-ul returnează propoziții NMEA, cele care sunt de interes fiind cele $GPRMC$ și $GPGGA$. Acestea sunt generate de GPS doar după ce s-a conectat cu succes la cel puțin 4 sateliți. Pentru propoziții de tipul $GPRMC$, latitudinea se află după a 3-a virgulă, iar pentru cele de tip $GPGGA$, după a 2-a.

```
// function which, from an input NMEA sentence,
// extracts a string with the latitude
String extractCoordinates(const String& sentence)
{
  int commas;
  if (sentence.indexOf("GPRMC") != -1) {
    commas = 3;
  } else if (sentence.indexOf("GPGGA") != -1){
    commas = 2;
  } else {
    return "";
  }

  int commaCount = 0;
  int startIndex = -1;
  int endIndex = -1;
```

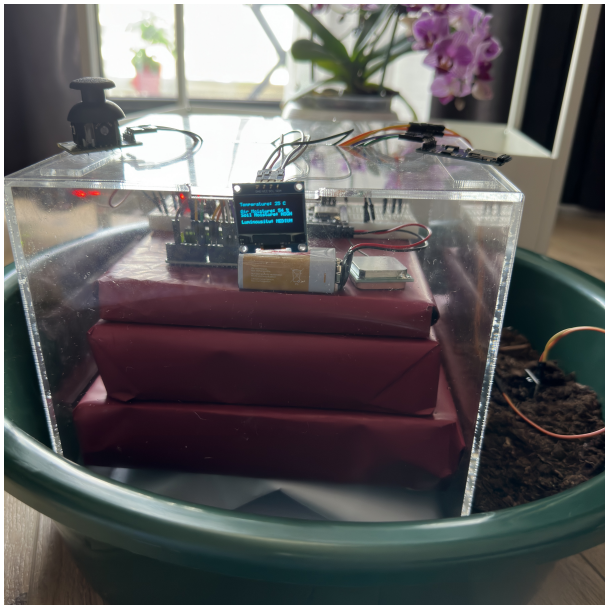
```
  // count commas and extract the value of the latitude
  for (size_t i = 0; i < sentence.length(); i++) {
    if (sentence[i] == ',') {
      commaCount++;
      if (commaCount == commas) {
        startIndex = i + 1;
      } else if (commaCount == commas + 1) {
        endIndex = i;
        break;
      }
    }
  }

  if (startIndex != -1 && endIndex != -1) {
    return sentence.substring(startIndex, endIndex);
  } else {
    return "";
  }
}

// function which reads NMEA sentences from
// software serial for 20 seconds and returns the
// latitude in case the sentence is valid
String getGPGGAsentence()
{
  String sentence = "";
  bool startedSentence = false;
  int currentTime = timerDisplayInfo;
  SoftwareSerial gpsSerial(RXPin, TXPin);
  gpsSerial.begin(9600);

  while (timerDisplayInfo - currentTime < MINUTE / 4) {
    if (gpsSerial.available()) {
      char c = gpsSerial.read();
      if (c == '$') {
        // check if we have successfuly read an entire GPGGA sentence
        String extractedCoordinate = extractCoordinates(sentence);
        if (!extractedCoordinate.equals("")) {
          return extractedCoordinate;
        }

        sentence = "";
        sentence += c;
        startedSentence = true;
      } else {
        if (startedSentence == true) {
          sentence += c;
        }
      }
    }
```

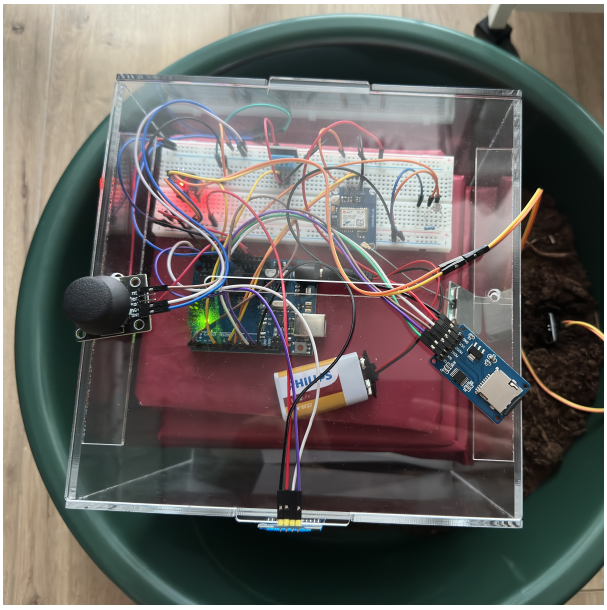
```
  }
  sentence = "";
  return sentence;
}
```

Se încearcă citirea datelor de pe serială timp de 20 de secunde sau până la citirea unei propoziții valide. Verificarea validității se face după începutul propoziției și după numărul de virgule necesare extragerii latitudinii.

Rezultate obținute

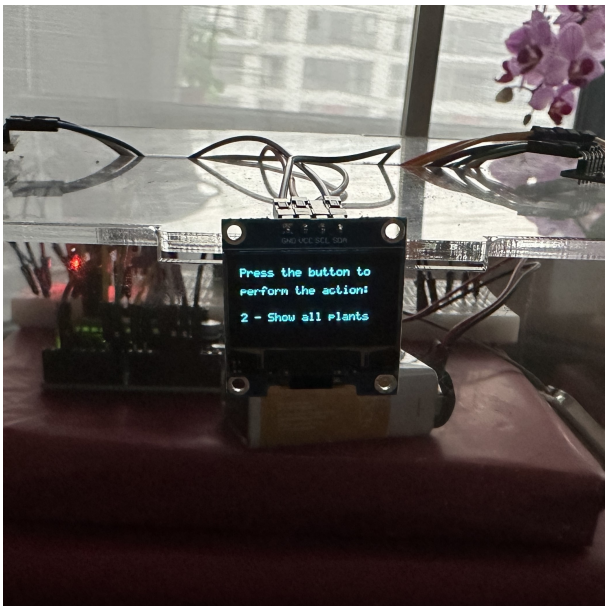
Overview-ul sistemului



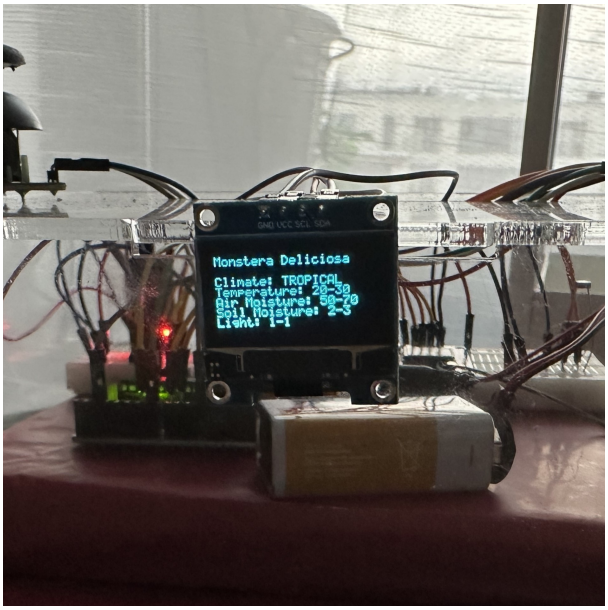


Meniul de stand-by

Exemplul de display al unuia din cele 5 meniuri



Exemplu de afișare a datelor despre plante



Demo

Limitări

Din păcate, există o limitare a uneia dintre funcționalitățile proiectului, și anume faptul că modulul GPS nu se conectează la suficienți sateliți atunci când stă în interior, trebuind să stea afară o perioadă de timp pentru a face conexiunea. În cod, am implementat ca, în cazul în care nu se detectează o latitudine validă, clima să rămână aceeași ca cea din citirile anterioare (însă o resetare a metricilor va șterge această valoare).

Cu toate acestea, această limitare nu e neapărat un lucru rău, mai ales dacă se dorește plantarea în interior, acolo unde nu contează clima.

Concluzii

Mi s-a părut extrem de interesant acest proiect, este pentru prima oară când am lucrat atât pe partea de software cât și pe partea de hardware simultan. Am învățat cât de limitată este programarea unei plăcuțe precum Arduino Uno R3 din punctul de vedere al memoriei, deoarece am întâmpinat dificultăți din cauza utilizării ecranului OLED și a unei biblioteci ce ocupa prea multă memorie, trebuind să o schimb cu una ce ocupa mai puțină memorie, dar oferea facilități mai limitate. Similar, inițial doream să folosesc o telecomandă cu senzor infraroșu pentru a controla meniul, dar din nou memoria era insuficientă, așa că am decis să folosesc un buton (singurul buton pe care îl aveam la dispoziție fiind cel al joystick-ului).

De asemenea, am înțeles mult mai bine materia din laborator, am învățat să am grijă la alegerea pieselor pentru că pinii sunt limitați și nu toate componentele pot fi folosite simultan.

Download cod

Codul poate fi vizualizat aici sau descărcat aici.

Jurnal

- 19 aprilie - alegere temă proiect
- 22 aprilie - achiziționare piese
- 2 mai - realizare documentație
- 5-8 mai - realizare parte hardware
- 10-20 mai - realizare parte software
- 28-29 mai - finalizare documentație

Bibliografie/Resurse

- Piese: https://cleste.ro/
- https://www.instructables.com/OLED-I2C-DISPLAY-WITH-ARDUINO-Tutorial/
- https://www.brainy-bits.com/post/how-to-use-the-dht11-temperature-and-humidity-sensor-with-an-arduino
- http://educ8s.tv/arduino-sd-card-tutorial/
- https://www.youtube.com/watch?v=qKku-mmwNIA&list=LL&index=8&ab_channel=BINARYUPDATES
- https://docs.arduino.cc/learn/built-in-libraries/software-serial
- https://content.meteoblue.com/en/research-education/educational-resources/meteoscool/general-climate-zones
- ChatGPT pentru găsirea plantelor care să se încadreze în anumitele tipuri de climă

Export to PDF