Universitatea
Transilvania
din Brașov
FACULTATEA DE INGINERIE ELECTRICĂ
ȘI ȘTIINȚA CALCULATOARELOR

Documentație Proiect SS2

Mandan Andrei

ETTI / II / 4LF633

Cuprins

Componente Hardware:

Arduino Mega

Proiectul are la bază un modul de dezvoltare de tip Arduino Mega. Este bazată pe pe microcontrolerul Atmega2560 și este utilizat pentru proiecte de electronică și programare. Este o alegere populară pentru aplicații complexe datorită numărului mare de pini și a capacității de memorie extinse.

Placa conține 54 de pini digitali, 16 analogici, are o memorie flash de 256 KB dintre care 8 KB rezervați pentru bootloader, 8KB SRAM și 4KB EEPROM; are 4 porturi UART, un conector I2C și SPI pentru conexiuni cu alte module sau senzori. Placa funcționează la o tensiune de 5V și are o ieșire de 5V și alta de 3.3V. Are o dimensiune de 5.3x10cm.

Modulul este potrivit pentru proiecte mai mari care necesită conexiuni hardware simulan. Suporta periferice complexe cum ar fi ecrane, motoare sau senzori multipli, fiind ideal pentru proiecte de robotica, automatizări sau dispozitie IoT.

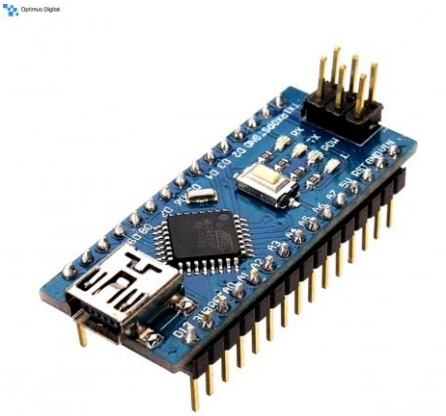
În proiectul meu Arduino Mega-ul este placa principală, este cea care primește și afișează pe ecran toate datele de la senzorii conectați la ea cât și informațiile transmise de către Nano.

Arduino Nano

Placa Arduino Nano este o platformă de dezvoltare compactă, bazată pe microcontrolerul ATmega328 (sau ATmega168 în variante mai vechi), concepută pentru proiecte care necesită dimensiuni reduse și eficiență. Este una dintre cele mai populare alegeri pentru aplicații portabile și sisteme integrate.

Modulul conține 14 pini digitali (dintre care 6 pot fi folosiți ca PWM) și 8 analogici. Ca memorie are 32 KB flash (dintre care 2KB rezervați pentru bootloader), 2KB SRAM și 1 KB EEPROM. Placa lucrează la 5V, are un conector Mini-USB pentru alimentare și programare și are o dimensiune de 18 mm x 45 mm.

Nano-ul are în proiectul meu rolul de transmițător, deoarece el transmite prin intermediul unui transceiver wireless nRF24L01 către Mega-ul principal datele recepționate.



Senzor de temperatură și umiditate DHT22

Senzorul DHT22 este un senzor de umiditate și temperatură mic care folosește un senzor capacitiv de umiditate și un termistor pentru a măsura aerul din jur. Acesta oferă semnal digital pe pinul de date.

Senzorul are 3 pini: primul, marcat cu "+" se conectează la 5V sau 3.3V; al doilea, marcat cu "OUT" se conectează la un pin digital; al treilea, marcat cu "-" se conectează la un GND al plăcii de dezvoltare.

Modul cu ceas în timp real DS3231

Acest modul este un ceas în timp real extrem de precis, alimentat la 3.3V-5V. Scopul lui în proiect este de a ține minte ora în cazul în care proiectul nu este alimentat.

Modul comunicare wireless nRF24L01

Modulul are ca scop realizarea comunicației dintre părțile proiectului. El este un transceiver wireless bazat pe tehnologia radio frecvență. Funcționează pe banda de frecvență 2.4 GHz, are o viteză de transmisie de până la 2 mbps și, în condiții ideale, poate transmite până la 100 de metrii in aer liber.

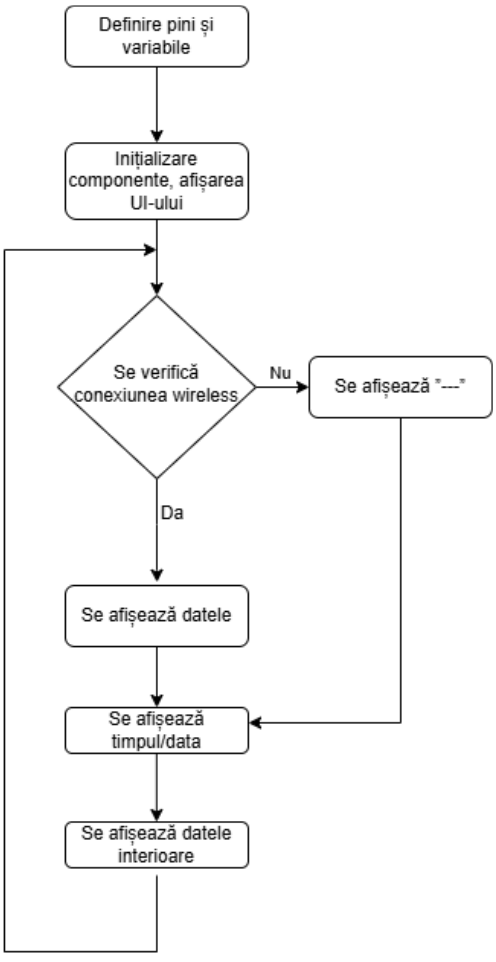
Display TFT LCD

Acest display este folosit pentru a afișa datele furnizate de senzori, cât și alte date relevante. Motivul pentru care am ales acest display a fost posibilitatea de a realiza o interfață grafică. El are o rezoluție de 480 x 320 pixeli, este bazat pe controllerul ILI9486 și este alimentat la 5V.

Alte componente

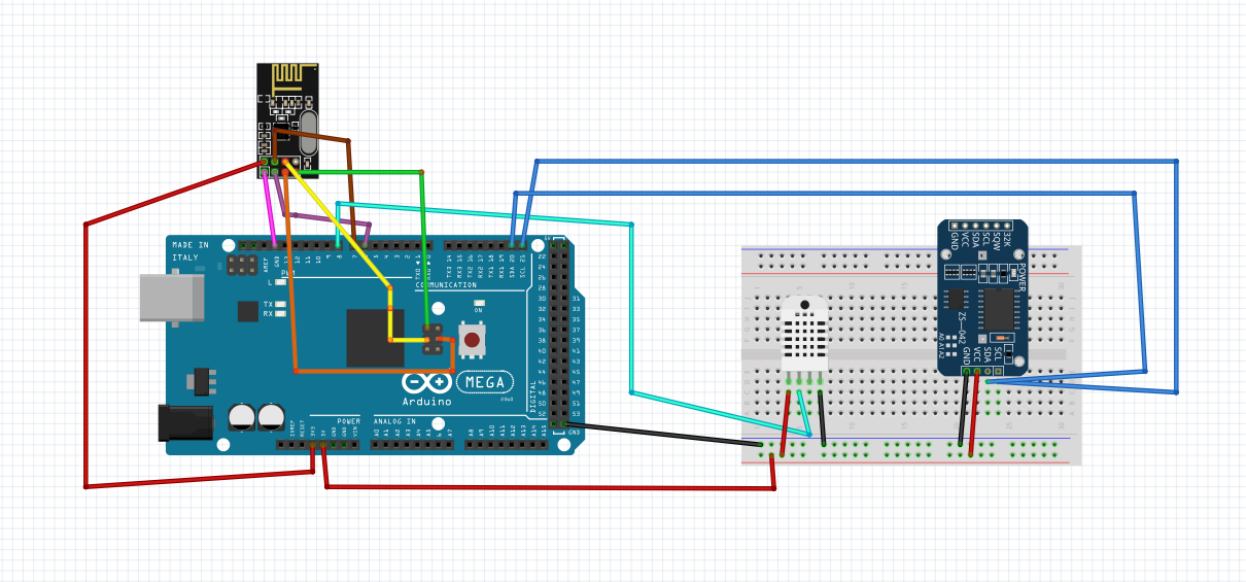
De asemenea aceste componente au trebuit interconectate. Pentru a face asta am folosit 2 breadboard-uri de 400 pini, cabluri mamă-tată, mamă-mamă și headeri de pini.

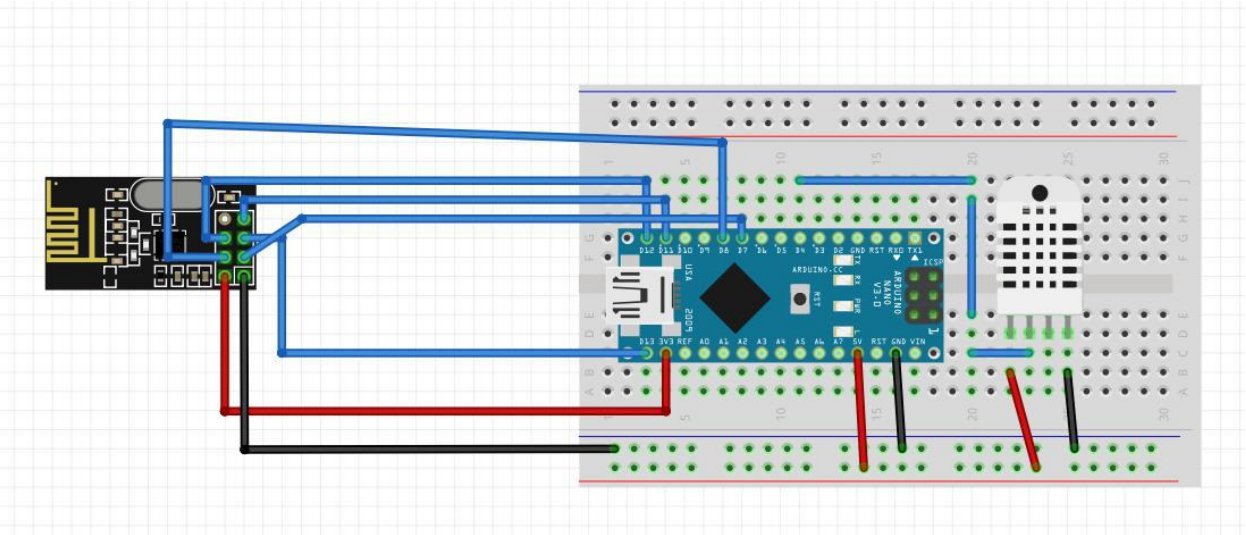
Schema logică



Diagrame de conectare

În programul Fritzing am realizat schema ambelor module.





Biblioteci folosite

În cadrul acestui proiect am folosit câteva biblioteci open-source pentru a putea folosi cu ușurință componentele. Acestea sunt:

- TFT_HX8357-master: pentru display
- RF24-master: pentru tranceiverul wireless
- DHT-sensor-library-master: pentru senzorul de temperatură
- Sodaq_DS3231-master: pentru modulul Real-Time Clock

Exemple din codul pentru Arduino

Funcția pentru afișare a UI-ului:

```
void printUI()
{

    tft.drawRoundRect(5,5,470,71,5,TFT_WHITE);tft.drawRoundRect(6,6,470,71,5,TFT_WHITE);
    tft.drawRoundRect(5,90,220,225,5,TFT_WHITE); tft.drawRoundRect(6,91,220,225,5,TFT_WHITE);
    tft.drawRoundRect(250,90,220,225,5,TFT_WHITE); tft.drawRoundRect(251,91,220,225,5,TFT_WHITE);

    tft.fillRect(26,90,180,40,TFT_GREEN); tft.fillRect(270,90,180,40,TFT_CYAN);

    tft.setCursor(62,100);  tft.setTextColor(TFT_BLACK); tft.setTextSize(3); tft.print("TRANSM");
    tft.setCursor(290,100); tft.setTextColor(TFT_BLACK); tft.setTextSize(3); tft.print("INTERIOR");
    tft.setCursor(162,165); tft.setTextColor(TFT_GREEN); tft.setTextSize(6); tft.print("%");
    tft.setCursor(412,165); tft.setTextColor(TFT_CYAN);  tft.setTextSize(6); tft.print("%");
    tft.setCursor(162,230); tft.setTextColor(TFT_GREEN); tft.setTextSize(6); tft.print("C");
    tft.setCursor(145,230); tft.setTextColor(TFT_GREEN); tft.setTextSize(2); tft.print("o");
    tft.setCursor(412,230); tft.setTextColor(TFT_CYAN);  tft.setTextSize(6); tft.print("C");
    tft.setCursor(395,230); tft.setTextColor(TFT_CYAN);  tft.setTextSize(2); tft.print("o");

}
```

Funcția pentru afișare a temperaturii interioare:

```
void printIndoorTemperature()
{
    if(indoorTemperature != previousIndoorTemperature){

      String temperature = String(indoorTemperature,1);
      tft.fillRect(270,232,120,40,TFT_BLACK);
      tft.setCursor(270,234); tft.setTextColor(TFT_CYAN); tft.setTextSize(5); tft.print(temperature);
      previousIndoorTemperature = indoorTemperature;

    }

}
```

Funcția pentru afișare a umidității exterioare:

```
void printRemoteHumidity(){

    String humidity;

    if(remoteHumidity != previousRemoteHumidity){
      if(remoteHumidity == 0.0 && remoteTemperature == 0.0){
        humidity = "---";
      } else {
          humidity = String(remoteHumidity,1);
        }

    tft.fillRect(20,167,120,40,TFT_BLACK);
    tft.setCursor(20,167); tft.setTextColor(TFT_GREEN); tft.setTextSize(5); tft.print(humidity);
    previousRemoteHumidity = remoteHumidity;

    }
}
```

Funcțiile pentru comunicare și achiziție wireless:

```
void startWirelessCommunication()
{
  myRadio.begin();
  myRadio.setChannel(115);
  myRadio.setPALevel(RF24_PA_MAX);
  myRadio.setDataRate( RF24_250KBPS ) ;
  myRadio.openReadingPipe(1, addresses[0]);
  myRadio.startListening();
  delay(100);
}
```

```
void checkForWirelessData(){

  if ( myRadio.available())
    {
      while (myRadio.available())
       {

          myRadio.read( &data, sizeof(data) );
          previousRemoteTemperature = remoteTemperature;
          previousRemoteHumidity = remoteHumidity;
          remoteTemperature = data.temperature;
          remoteHumidity = data.humidity;

       }

          String mesaj = "";
          mesaj += "Package:\n";
          mesaj += "Temp ext: " + String(data.temperature) + "\n";
          mesaj += "Umidit ext: " + String(data.humidity) + "\n";
          mesaj += "Temp int: " + String(dht.readTemperature()) + "\n";
          mesaj += "Umidit int: " + String(dht.readHumidity());

          Serial.println(mesaj);

    }

}
```

Program Python cu GUI

Scurt rezumat:

Programul este o aplicație grafică scrisă în Python folosind biblioteca Tkinter, care permite afișarea în timp real a datelor provenite de la o stație meteo conectată prin port serial, în cazul de față cu un Arduino Mega. Interfața include patru etichete pentru afișarea temperaturii și umidității interioare și exterioare, precum și un buton pentru conectarea la portul serial. La apăsarea butonului, utilizatorul este invitat să selecteze un port COM, iar programul stabilește o conexiune serială. Odată conectat, aplicația citește periodic (la fiecare 500ms) datele transmise de dispozitiv, le decodifică și actualizează automat valorile afișate pe ecran. În plus, o bară de stare indică dacă conexiunea este activă sau nu. Programul este util pentru monitorizarea condițiilor meteo în timp real.

Snippets din cod:

```
# titlu
titlu = Label(window, text="Stație Meteo", font=('Arial', 16, 'bold'),
              bg="#e0f7fa", fg="#006064")
titlu.pack(pady=10)
```

```
# Labeluri pentru valori
temp_ext_label = Label(data_frame, text="Temp ext: -- °C", font=("Arial", 11),
                       anchor="w", bg="#b2ebf2")
temp_ext_label.pack(fill="x", padx=5, pady=2)

umid_ext_label = Label(data_frame, text="Umid ext: -- %", font=("Arial", 11),
                       anchor="w", bg="#b2ebf2")
umid_ext_label.pack(fill="x", padx=5, pady=2)

temp_int_label = Label(data_frame, text="Temp int: -- °C", font=("Arial", 11),
                       anchor="w", bg="#b2ebf2")
temp_int_label.pack(fill="x", padx=5, pady=2)

umid_int_label = Label(data_frame, text="Umid int: -- %", font=("Arial", 11),
                       anchor="w", bg="#b2ebf2")
umid_int_label.pack(fill="x", padx=5, pady=2)
```

```python
# actualizare date
def update():
    if serialInst.is_open:
        while serialInst.in_waiting:
            linie = serialInst.readline().decode('utf-8', errors='ignore').strip()

            if linie.startswith("Temp ext:"):
                valoare = linie.split(":")[1].strip()
                temp_ext_label.config(text=f"Temp ext: {valoare} °C")

            elif linie.startswith("Umidit ext:"):
                valoare = linie.split(":")[1].strip()
                umid_ext_label.config(text=f"Umid ext: {valoare} %")

            elif linie.startswith("Temp int:"):
                valoare = linie.split(":")[1].strip()
                temp_int_label.config(text=f"Temp int: {valoare} °C")

            elif linie.startswith("Umidit int:"):
                valoare = linie.split(":")[1].strip()
                umid_int_label.config(text=f"Umid int: {valoare} %")
```

```python
# functie conectare la port
def conecteaza_serial():
    global serialInst, portVar

    # inchide portul daca e deja deschis
    if serialInst.is_open:
        serialInst.close()

    # cautare porturi disponibile
    ports = serial.tools.list_ports.comports()
    portList = []

    for onePort in ports:
        portList.append(str(onePort))
        print(str(onePort))

    val = input("Select Port: COM")
    portVar = None
    for x in range(len(portList)):
        if portList[x].startswith("COM" + str(val)):
            portVar = "COM" + str(val)
            print(f"Conectat la: {portVar}")

    if portVar:
        try:
            serialInst.port = portVar
            serialInst.baudrate = 9600
            serialInst.timeout = 1
            serialInst.open()
            status_bar.config(text=f"Conectat la {portVar}")
        except Exception as e:
            status_bar.config(text=f"Eroare la conectare {portVar}")
            print("Eroare:", e)
    else:
        status_bar.config(text="Port invalid")
```