Proiect stație meteo

1. Scopul proiectului

Proiectul urmărește realizarea unei stații meteo autonome, capabilă să monitorizeze și să transmită în timp real parametri esențiali de mediu: temperatura, umiditatea, presiunea atmosferică, viteza și direcția vântului, nivelul de precipitații și tensiunea de încărcare solară a bateriei.

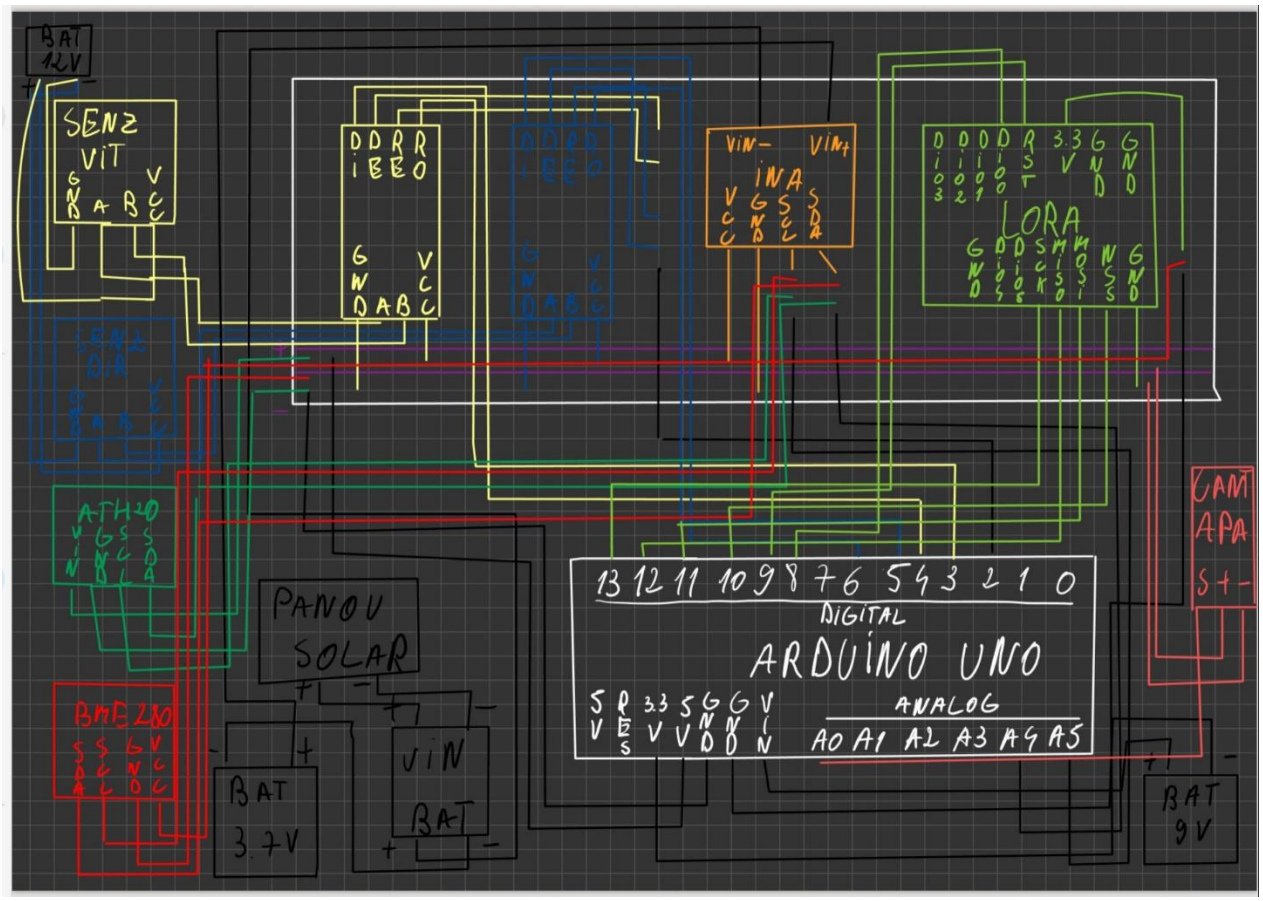
2. Componente hardware utilizate

- **Placă de dezvoltare**: Arduino Uno & ESP32
- **Senzori de mediu**:
    - **BME280** – presiune atmosferică
    - AHT20 – temperatura si umiditate
    - **Anemometru + giruetă** – pentru viteza și direcția vântului
    - **Senzor nivel apa**– pentru măsurarea cantității de precipitații
- **Sursă de alimentare**: Panou solar
- **Modul de comunicație**:
    - **LoRa** – pentru transmisie pe distanțe mari
- **Modul CN3791** – pentru încărcarea bateriei
- Adaptor pentru anemometru si giruetă

3. Funcționalități principale

- Măsurarea continuă a parametrilor meteo: temperatură, umiditate, presiune, viteza și direcția vântului, precipitații
- Determinarea cantității totale de precipitații într-un interval de timp
- Monitorizarea tensiunii de încărcare a bateriei de la panoul solar
- Transmiterea datelor către platforma ThingSpeak
- Transmisie date la distanta prin LoRa

4.Schema Arduino



5.Cod Arduino

```
#include <SoftwareSerial.h>

#include <ModbusMaster.h>

#include <Adafruit_AHTX0.h>

#include <Adafruit_BMP280.h>

#include <Adafruit_INA219.h>

#include <SPI.h>

#include <LoRa.h>

#include <LowPower.h>


#define RE_DE_PIN 2

#define RX_PIN1 3

#define TX_PIN1 4

#define RX_PIN2 5

#define TX_PIN2 6

#define WATER_LEVEL_PIN A0

#define SS_PIN 10

#define RESET_PIN 9

#define DIO0_PIN 8


Adafruit_AHTX0 aht;

Adafruit_BMP280 bmp;

Adafruit_INA219 ina219;


SoftwareSerial RS485Serial1(RX_PIN1, TX_PIN1);

SoftwareSerial RS485Serial2(RX_PIN2, TX_PIN2);
```

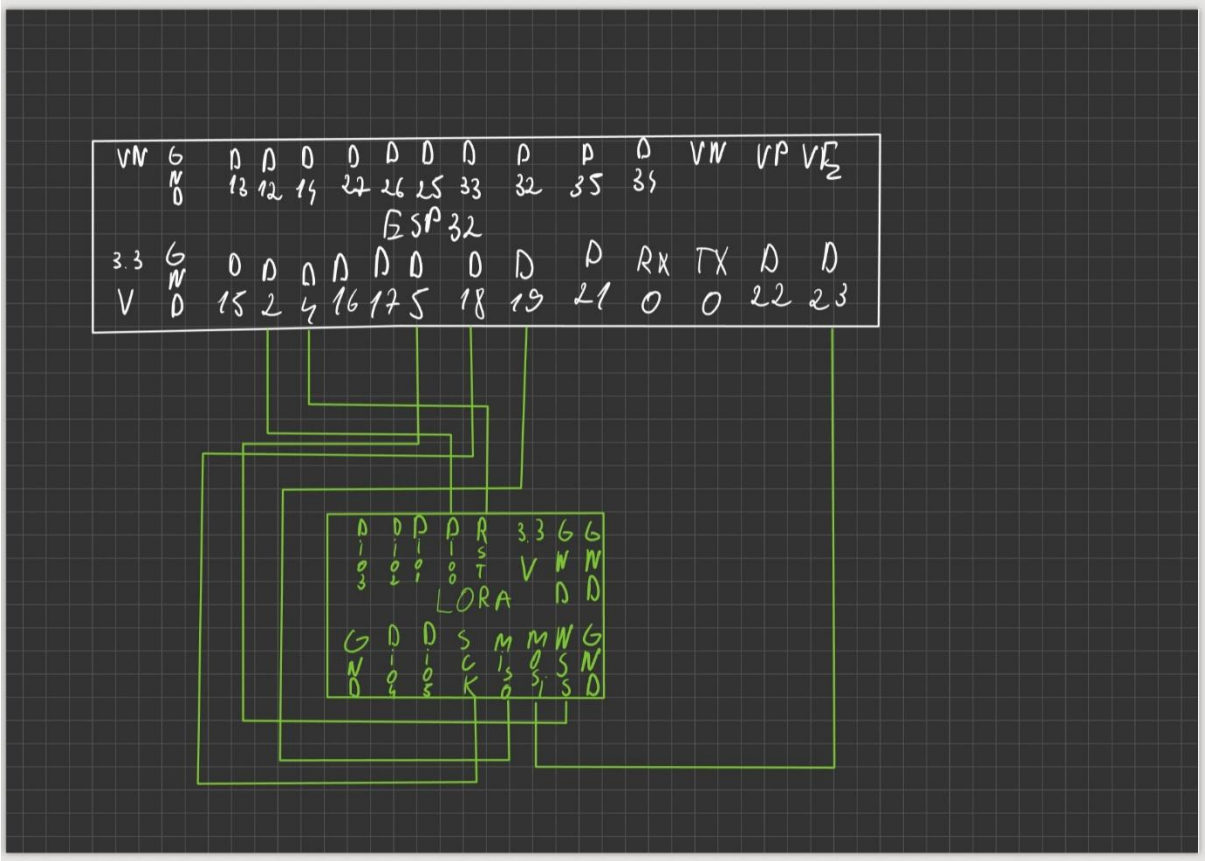
```
ModbusMaster node1;
ModbusMaster node2;

void preTransmission() {
  digitalWrite(RE_DE_PIN, HIGH);
}

void postTransmission() {
  digitalWrite(RE_DE_PIN, LOW);
}

int getDirectionCode(float degrees) {
  degrees *= 10;
  if (degrees >= 337.5 || degrees < 22.5) return 1;
  if (degrees < 67.5) return 2;
  if (degrees < 112.5) return 3;
  if (degrees < 157.5) return 4;
  if (degrees < 202.5) return 5;
  if (degrees < 247.5) return 6;
  if (degrees < 292.5) return 7;
  return 8;
}

void setup() {
  Serial.begin(9600);
  pinMode(RE_DE_PIN, OUTPUT);
  digitalWrite(RE_DE_PIN, LOW);
  pinMode(WATER_LEVEL_PIN, INPUT);
```

```
  RS485Serial1.begin(4800);
  RS485Serial2.begin(4800);

  node1.begin(1, RS485Serial1);
  node2.begin(1, RS485Serial2);
  node1.preTransmission(preTransmission);
  node1.postTransmission(postTransmission);
  node2.preTransmission(preTransmission);
  node2.postTransmission(postTransmission);

  LoRa.setPins(SS_PIN, RESET_PIN, DIO0_PIN);
  if (!LoRa.begin(433E6)) {
   Serial.println(F("Eroare la inițializarea LoRa!"));
   while (1);
  }
  Serial.println(F("LoRa initializat!"));

  if (aht.begin()) Serial.println(F("AHT20 OK"));
  else Serial.println(F("Eroare AHT20"));

  if (bmp.begin(0x76)) Serial.println(F("BMP280 OK"));
  else Serial.println(F("Eroare BMP280"));
  if (ina219.begin()) Serial.println(F("INA219 OK"));
  else Serial.println(F("Eroare INA219"));
}
```

```
void sendLoRaJSON(String jsonMessage) {
  LoRa.beginPacket();
  LoRa.print(jsonMessage);
  LoRa.endPacket();
  Serial.print(F("JSON trimis: "));
  Serial.println(jsonMessage);
}
void loop() {
  uint16_t data[2];
  uint16_t directionData[2];
  String json = "{";
  // Viteza vântului
  RS485Serial1.listen();
  float windSpeed = 0.0;
  if (node1.readHoldingRegisters(0x00, 2) == node1.ku8MBSuccess) {
    data[0] = node1.getResponseBuffer(0);
    data[1] = node1.getResponseBuffer(1);
    windSpeed = ((data[0] << 16) | data[1]);
  }
  json += "\"viteza\":" + String(windSpeed, 2) + ",";
  delay(1000);
  RS485Serial2.listen();
  int directionCode = 0;
  if (node2.readHoldingRegisters(0x00, 2) == node2.ku8MBSuccess) {
    directionData[0] = node2.getResponseBuffer(0);
    directionData[1] = node2.getResponseBuffer(1);
    float windDir = ((directionData[0] << 16) | directionData[1]) / 10.0;
    directionCode = getDirectionCode(windDir);}
```

```
  json += "\"directie\":" + String(directionCode) + ",";
  delay(1000);
  sensors_event_t humidity, temp;
  aht.getEvent(&humidity, &temp);
  json += "\"temp\":" + String(temp.temperature, 2) + ",";
  json += "\"umid\":" + String(humidity.relative_humidity, 2) + ",";
  delay(1000);
  float tensiune = ina219.getBusVoltage_V();
  json += "\"ten\":" + String(tensiune, 2) + ",";
  float presiune = bmp.readPressure() / 100.0F;
  json += "\"presiune\":" + String(presiune, 2) + ",";
  int nivelApa = analogRead(WATER_LEVEL_PIN);
  json += "\"n\":" + String(nivelApa);
  json += "}";
  delay(1000);
  sendLoRaJSON(json);
  Serial.println(F("--------------------------"));
  delay(1000);
  LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
  LowPower.powerDown(SLEEP_2S, ADC_OFF, BOD_OFF);
}
```

6. Schema ESP32



7.Cod ESP32

```
#include <WiFi.h>

#include <SPI.h>

#include <LoRa.h>

#include <HTTPClient.h>


const char* ssid = "Galaxy S23 Ultra";

const char* password = "xlod8705";

const char* apiKey = "R91DUFDE9OGJ809G";

const char* server = "http://api.thingspeak.com/update";
```

```
#define SS_PIN 5
#define RESET_PIN 4
#define DIO0_PIN 2

void setup() {
 Serial.begin(115200);
 delay(1000);

 WiFi.mode(WIFI_STA);
 WiFi.begin(ssid, password);
 Serial.println("\nConnecting to WiFi");

 while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");
  delay(100);
 }

 Serial.println("\nConnected to the WiFi network");
 Serial.print("Local ESP32 IP: ");
 Serial.println(WiFi.localIP());

 LoRa.setPins(SS_PIN, RESET_PIN, DIO0_PIN);
 if (!LoRa.begin(433E6)) {
  Serial.println("Eroare la inițializarea LoRa!");
  while (1);
 }
 Serial.println("LoRa initializat! Aștept mesaje...");
}
```

```
void loop() {
 int packetSize = LoRa.parsePacket();
 if (packetSize) {
  String message = "";
  while (LoRa.available()) {
   message += (char)LoRa.read();
  }

  Serial.print("Mesaj primit: ");
  Serial.println(message);

  message.replace("{", "");
  message.replace("}", "");

  float viteza = message.substring(message.indexOf("viteza\":") + 8,
message.indexOf(",\"directie")).toFloat();
  int directie = message.substring(message.indexOf("\"directie\":") + 11,
message.indexOf(",\"temp")).toInt();
  float temp = message.substring(message.indexOf("\"temp\":") + 7,
message.indexOf(",\"umid")).toFloat();
  float umid = message.substring(message.indexOf("\"umid\":") + 7,
message.indexOf(",\"ten")).toFloat();
  float ten = message.substring(message.indexOf("\"ten\":") + 6,
message.indexOf(",\"presiune")).toFloat();
  float presiune = message.substring(message.indexOf("\"presiune\":") + 11,
message.indexOf(",\"n")).toFloat();
  int nivelApa = message.substring(message.indexOf("\"n\":") + 4).toInt();

  Serial.println("Trimitem datele către ThingSpeak...");
```

```
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String url = String(server) + "?api_key=" + apiKey +
                 "&field1=" + String(viteza) +
                 "&field2=" + String(directie) +
                 "&field3=" + String(temp) +
                 "&field4=" + String(umid) +
                 "&field5=" + String(ten) +
                 "&field6=" + String(presiune) +
                 "&field7=" + String(nivelApa);

    http.begin(url);
    int httpCode = http.GET();
    if (httpCode > 0) {
      Serial.println("Date trimise cu succes!");
    } else {
      Serial.println("Eroare la trimitere ThingSpeak.");
    }
    http.end();
  }
 }
}
```