Realizarea unei rețele de senzori, folosind standardul de comunicație Bluetooth - IEEE 802.15.1

Sisteme Incorporate

Autori : Friškan Dario, Drăgoi Christian Alexandru, Ene Ion Sebastian

An Studiu III

Descrierea Proiectului

The project consists of developing a network of autonomous terminal sensor nodes that collect environmental data using temperature, humidity, and water level sensors. The goal is to gain insight into the environment in which we operate or to maintain an optimal environment for various applications. Each terminal node is equipped with two different types of sensors, a Bluetooth module, and an Arduino Uno.

The essential functionality is to allow the user to have a clear vision of their environment, enabling them to control the environment in which they operate and improve ambient conditions for healthy living. This includes monitoring environmental conditions, adjusting necessary parameters for a healthy environment, and optimizing the resources used. The user will thus be able to create a beneficial space for their health and productivity, as well as for the comfort of their home or office.

Circuit Documentation

Summary

The circuit in question is designed to collect various environmental data using a set of sensors and communicate this information via Bluetooth. It consists of two Arduino UNO microcontrollers, a pair of Bluetooth HC-06 modules for wireless communication, a temperature sensor (LM35), two DHT11 humidity and temperature sensors, and a water level sensor. The system is powered by two 9V batteries. The sensors collect temperature, humidity, and water level data, which are then sent to a remote device through the Bluetooth modules.

Component List

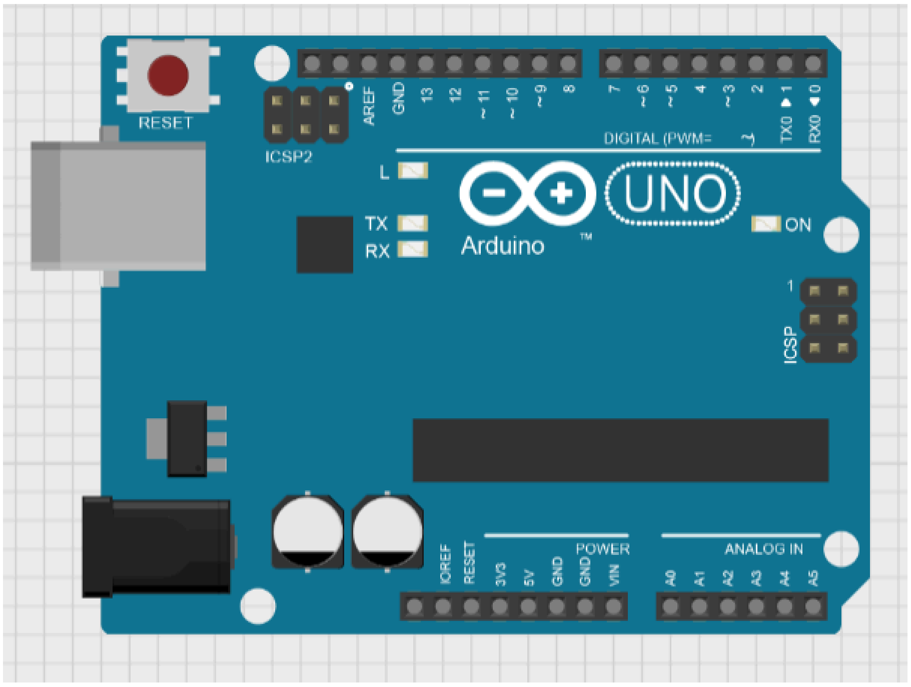
Microcontrollers

- **Arduino UNO(2)**: A microcontroller board based on the ATmega328P. It has digital input/output pins, analog inputs, a USB connection for programming, and power management features.
  The Arduino Uno features an ATmega328P microcontroller from ATMEL (now part of Microchip Technology). The ATmega328P is an 8-bit microcontroller with the following key specifications:

1. Clock Speed: 16 MHz
2. Flash Memory: 32 KB, with 0.5 KB used by the bootloader

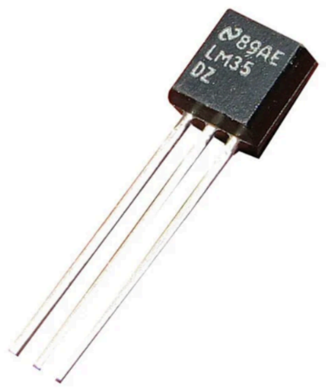
3. SRAM: 2 KB
4. EEPROM: 1 KB
5. Operating Voltage: 5V
6. Input Voltage (recommended): 7-12V
7. Digital I/O Pins: 14 (of which 6 can provide PWM output)
8. Analog Input Pins: 6
9. DC Current per I/O Pin: 20 mA
10. DC Current for 3.3V Pin: 50 mA

The ATmega328P supports serial communication via UART, I2C (TWI), and SPI. It's well-suited for a wide range of applications, from simple LED control to more complex data acquisition and processing tasks. The Arduino Uno board also features easy-to-use programming via the Arduino IDE, which simplifies coding and uploading programs to the microcontroller.



Sensors

Temperature Sensor (LM35): An analog temperature sensor that provides a voltage output proportional to the ambient temperature.



Specifications:

- Operating Voltage: 4 to 30V DC
- Temperature Range:
- LM35: -55°C to 150°C
- LM35C: -40°C to 110°C
- Accuracy:
- At 25°C: ±0.5°C
- Over a full temperature range: ±1°C
- Output Voltage: 10mV per °C
- Output Impedance: 0.1Ω for 1mA load
- Current Consumption: 60μA (typical)
- Response Time: 1.5 seconds (typical, with a 10μF bypass capacitor)
- Dimensions: TO-92 package: 5.04mm x 4.19mm x 3.68mm
- Weight: Approx. 0.4g (for TO-92 package)

Features:

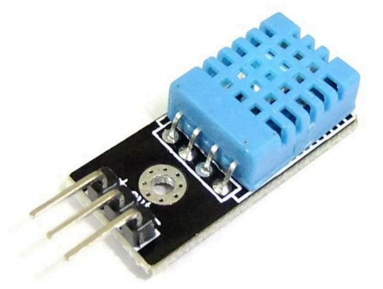
- Low Cost: The LM35 is an affordable sensor.
- Linear Output: The output voltage is directly proportional to the Celsius temperature.
- High Precision: Provides a precise measurement with a low offset voltage.
- Wide Operating Range: Suitable for a wide range of applications across various temperatures.
- Low Self-Heating: The sensor consumes low power, which minimizes self-heating.
- Calibrated Directly in Celsius: Outputs a linear voltage that corresponds directly to the temperature in degrees Celsius.
- Low Output Impedance: Makes it easier to interface with ADCs and microcontrollers.

Pin Configuration:

The LM35 sensor typically comes in a 3-pin TO-92 package:

- VCC: Power supply (4V to 30V)
- Output: Analog voltage output
- GND: Ground

DHT11 Humidity and Temperature Sensor(2): A digital sensor that measures ambient humidity and temperature.



Technical Specifications:

- Operating Voltage: 3.0 to 5.5V DC
- Temperature Range: 0°C to 50°C (32°F to 122°F)
- Temperature Accuracy: ±2.0°C
- Humidity Range: 20% to 90% RH (Relative Humidity)
- Humidity Accuracy: ±5.0% RH
- Resolution:
- Temperature: 1°C
- Humidity: 1% RH
- Sampling Period: 1 second (1 Hz)
- Dimensions: 15.5mm x 12mm x 5.5mm
- Weight: Approx. 2.4g
- Interface: Single-wire digital signal
- Power Consumption:
- Measuring: 0.3mA (average)
- Standby: 60µA
- Response Time:
- Humidity: ≤5s
- Temperature: ≤10s

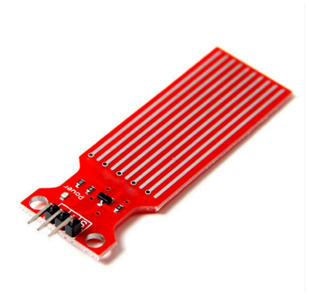
Features:

- Low Cost: The DHT11 is an inexpensive sensor, making it accessible for various projects.
- Easy to Use: It has a simple interface and requires only one data pin for communication.
- Calibrated Output: Provides fully calibrated digital output, eliminating the need for extra components.
- Wide Operating Range: Suitable for a wide range of applications in normal temperature and humidity environments.
- Stable and Long-Term Operation: Reliable for long-term use with proper calibration.
- Pin Configuration

  ## DHT11 pins:

- VCC: Power supply (3.0V to 5.5V)
- Data: Serial data output
- NC: No connection (not used)
- GND: Ground

- **Water Level Sensor**: A sensor that detects the level of water in a container or environment.



Specifications:

- Operating Voltage: 3.3V to 5V DC
- Output Type: Analog
- Current Consumption: Less than 20mA
- Sensing Range: Typically 0 to 40mm (depending on the sensor model)
- Output Voltage Range: 0V to Vcc (typically 0V to 5V)
- Dimensions: Varies by model, commonly around 62mm x 20mm x 8mm
- Weight: Approx. 3g to 5g
- Response Time: Less than 500ms
- Operating Temperature: -10°C to 60°C
- Humidity Range: 20% to 95% RH

Features:

- Easy to Use: Simple interface for easy integration with microcontrollers.

- Low Power Consumption: Suitable for battery-operated applications.
- High Sensitivity: Can detect small changes in water level.
- Compact Size: Fits in tight spaces.
- Analog Output: Provides continuous data on water level.

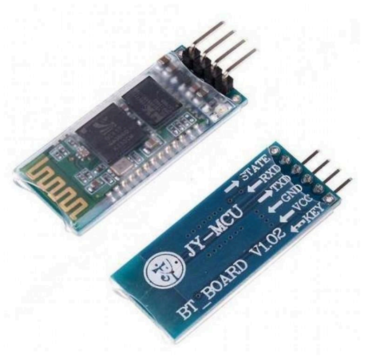
Pin Configuration:

A typical water level sensor has three pins:

- VCC: Power supply (3.3V to 5V)
- GND: Ground
- OUT: Analog output voltage

Communication Modules

- **Bluetooth HC-06(2)**: A wireless Bluetooth module that allows for serial communication over Bluetooth.



Specifications:

- Bluetooth Protocol: Bluetooth V2.0+EDR
- Frequency: 2.4GHz ISM band
- Modulation: GFSK (Gaussian Frequency Shift Keying)
- Transmit Power: ≤4 dBm (Class 2)
- Sensitivity: ≤-80 dBm at 0.1% BER
- Speed: Asynchronous: 2.1 Mbps/160 kbps, Synchronous: 1 Mbps/1 Mbps
- Power Supply: +3.3V DC (Regulated), up to 3.6V
- Operating Voltage: 3.3V
- Current Consumption: Paired mode: ~8 mA, Unpaired mode: ~30 mA
- Range: ~10 meters (33 feet)
- Dimensions: 28mm x 15mm x 2.35mm (approx.)

Interface:

- Serial Communication: UART
- Default Baud Rate: 9600 bps
- Data Bits: 8
- Stop Bits: 1
- Parity: None
- Flow Control: None

Pin Configuration:

1. VCC: Power supply input
2. GND: Ground
3. TXD: Transmit Data
4. RXD: Receive Data
5. STATE: Connection status (High when connected, Low otherwise)

Features:

- Ease of Use: Simple 4-pin interface (VCC, GND, TXD, RXD)
- Communication: Transparent serial wireless communication
- Compatibility: Compatible with most Bluetooth-enabled devices (e.g., smartphones, tablets, PCs)
- Operating Modes: Supports both Command and Data modes
- Indicator LED: Indicates connection status (blinking when not paired, steady when paired)

Power Sources

- **9V Battery(2)**: Provides power to the circuit components.

Wiring Details

Arduino UNO

- **5V**: Connected to the VCC pins of the Water Level Sensor, Bluetooth HC-06, and DHT11 Humidity and Temperature Sensor.
- **GND**: Connected to the GND pins of the Water Level Sensor, Bluetooth HC-06, and

DHT11 Humidity and Temperature Sensor.
- **Digital Pins (D2, D3, D10)**: Used for communication with the Bluetooth HC-06(D2, D3), for both nodes and DHT11 sensor(D10) for both nodes.
- **Analog Pins (A1)**: Used to read the analog voltage from the LM35 temperature sensor for the first node and the Water Level Sensor for the second node.

Temperature Sensor (LM35)

- **+Vs**: Connected to the 5V output from the Arduino UNO.
- **Vout**: Connected to the A1 analog input on the Arduino UNO.
- **GND**: Connected to the ground (GND) on the Arduino UNO.

DHT11 Humidity and Temperature Sensor

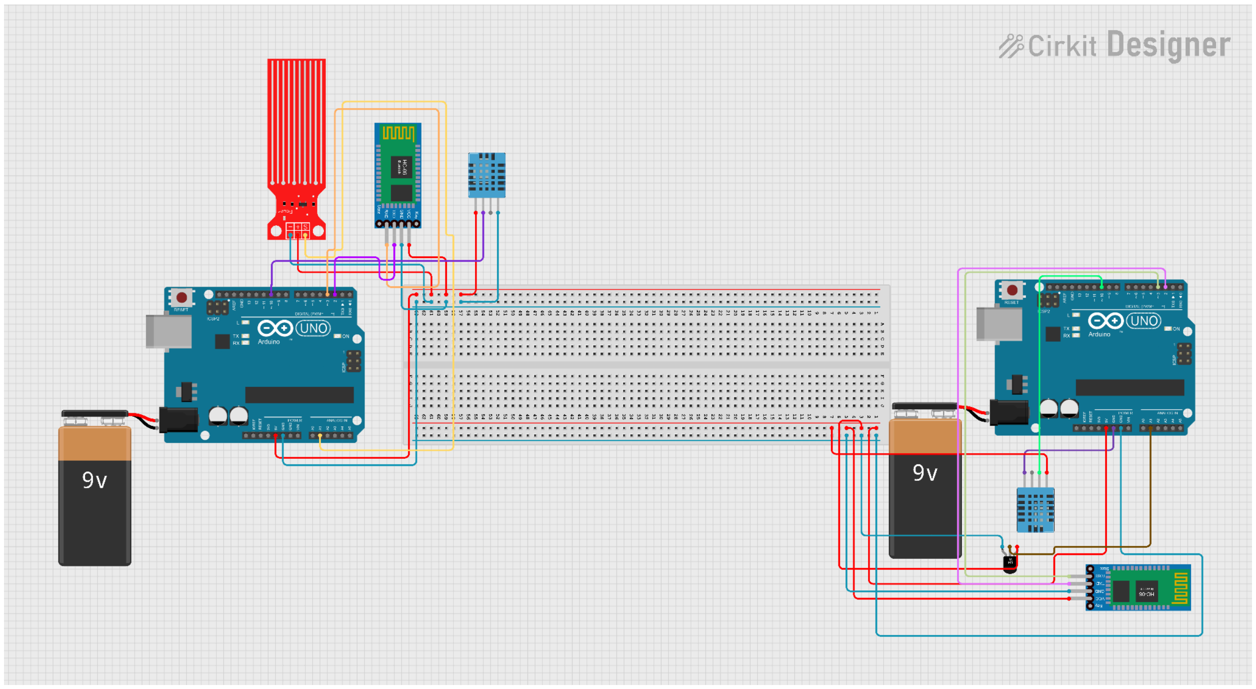
- **VDD**: Connected to the 5V output from the Arduino UNO.
- **DATA**: Connected to the digital pins D10 on the Arduino UNO (on both nodes).
- **GND**: Connected to the ground (GND) on the Arduino UNO.

Bluetooth HC-06

- **VCC**: Connected to the 5V output from the Arduino UNO.
- **GND**: Connected to the ground (GND) on the Arduino UNO.
- **TXD**: Connected to the digital pins D2 on the Arduino UNO (on both nodes).
- **RXD**: Connected to the digital pins D3 on the Arduino UNO (on both nodes).

Water Level Sensor

- **SIG**: Connected to the A1 analog input on the Arduino UNO.
- **VCC**: Connected to the 5V output from the Arduino UNO.
- **GND**: Connected to the ground (GND) on the Arduino UNO.



Documented Code

Arduino UNO (First Microcontroller)

File Name - arduinoNod1

```
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
#include <DHT.h>

#define DHTPIN 10     // Pinul digital conectat la senzorul DHT11
#define DHTTYPE DHT11 // Definirea tipului de senzor DHT11
#define LM35_PIN A1   // Pinul analogic la care este conectat senzorul LM35

DHT dht(DHTPIN, DHTTYPE);

const int rxPin = 2;
const int txPin = 3;

SoftwareSerial myBluetooth(rxPin, txPin);
```

```
void setup() {
  Serial.begin(9600);
  myBluetooth.begin(9600);
  Serial.println("Bluetooth este pregătit pentru comunicare");

  dht.begin(); // Inițializează senzorul DHT11
}

void loop() {
  // Citirea umidității de la DHT11
  float humidity = dht.readHumidity();

  // Verifică dacă citirea a reușit și dacă nu este NaN
  if (isnan(humidity)) {
    Serial.println("Eroare la citirea de la DHT11");
  } else {
    Serial.print("Umiditate: ");
    Serial.print(humidity);
    Serial.println(" %");
  }

  // Citirea temperaturii de la LM35
  int lm35Val = analogRead(LM35_PIN);
  float voltage = lm35Val * (5.0 / 1023.0);
  float temperatureLM35 = voltage * 100;  // Conversie tensiune în
temperatură

  Serial.print("Temperatură LM35: ");
  Serial.print(temperatureLM35);
  Serial.println(" *C");

  if (myBluetooth.available()) {
    char received = myBluetooth.read();
    Serial.print("Recepționat: ");
    Serial.println(received);
  }

  static unsigned long lastSendTime = 0;
  if (millis() - lastSendTime > 2000) {
    StaticJsonDocument<200> jsonDoc;
    if (!isnan(humidity)) {
      jsonDoc["DHT11"] = humidity;
    } else {
      jsonDoc["DHT11"] = "Error";
    }
    jsonDoc["LM35"] = temperatureLM35;

    String jsonString;
    serializeJson(jsonDoc, jsonString);

    myBluetooth.println(jsonString);

    Serial.print("Trimis JSON: ");
    Serial.println(jsonString);

    lastSendTime = millis();
  }
  delay(2000);
}
```

Code Description:

The code reads values from an LM35 temperature sensor and a DHT-11 humidity sensor, then sends this data via Bluetooth. The Bluetooth module (using SoftwareSerial) allows these data to be transmitted to an external device every 2 seconds in JSON format. The code can also receive data sent via Bluetooth and display it on the serial monitor. In the setup() section, it initializes serial communication and Bluetooth. In the main loop() section, the code reads the sensor values and periodically sends the sensor data via Bluetooth, also displaying it on the serial monitor.

Arduino UNO (Second Microcontroller)

File Name - arduinoNod2

```
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
#include <DHT.h>

#define DHTPIN 10     // Pinul digital conectat la senzorul DHT11
#define DHTTYPE DHT11 // Definirea tipului de senzor DHT11
#define WATER_SENSOR_PIN A1 // Pinul analogic la care este conectat
senzorul de nivel al apei

DHT dht(DHTPIN, DHTTYPE);
```

```
const int rxPin = 2;
const int txPin = 3;

SoftwareSerial myBluetooth(rxPin, txPin);

void setup() {
  Serial.begin(9600);
  myBluetooth.begin(9600);
  Serial.println("Bluetooth este pregătit pentru comunicare");

  dht.begin(); // Inițializează senzorul DHT11
}

void loop() {
  // Citirea umidității de la DHT11
  float humidity = dht.readHumidity();

  // Verifică dacă citirea a reușit și dacă nu este NaN
  if (isnan(humidity)) {
    Serial.println("Eroare la citirea de la DHT11");
  } else {
    Serial.print("Umiditate: ");
    Serial.print(humidity);
    Serial.println(" %");
  }

  // Citirea valorii de la senzorul de nivel al apei
  int waterLevelValue = analogRead(WATER_SENSOR_PIN);

  Serial.print("Nivelul apei: ");
  Serial.println(waterLevelValue);

  if (myBluetooth.available()) {
    char received = myBluetooth.read();
    Serial.print("Recepționat: ");
    Serial.println(received);
  }

  static unsigned long lastSendTime = 0;
  if (millis() - lastSendTime > 2000) {
    StaticJsonDocument<200> jsonDoc;
    if (!isnan(humidity)) {
      jsonDoc["DHT11-2"] = humidity;
    } else {
      jsonDoc["DHT11-2"] = "Error";
    }
    jsonDoc["WLVL"] = waterLevelValue;

    String jsonString;
    serializeJson(jsonDoc, jsonString);

    myBluetooth.println(jsonString);

    Serial.print("Trimis JSON: ");
    Serial.println(jsonString);

    lastSendTime = millis();
  }
  delay(2000);
}
```

Code Description:

The code reads values from a Water Level Sensor and a DHT-11 humidity sensor, then sends this data via Bluetooth. The Bluetooth module (using SoftwareSerial) allows these data to be transmitted to an external device every 2 seconds in JSON format. The code can also receive data sent via Bluetooth and display it on the serial monitor. In the setup() section, it initializes serial communication and Bluetooth. In the main loop() section, the code reads the sensor values and periodically sends the sensor data via Bluetooth, also displaying it on the serial monitor.

Extensions:

- `ArduinoJson - by Benoid`
- `DHT Sensor Library - by Adafruit`

Bibliography

- ***Arduino. (n.d.). Arduino Uno Rev3.*** https://www.arduino.cc/en/Main/ArduinoBoardUno
- ***Microchip Technology Inc. (2018). ATmega328/P Datasheet.*** http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega328P-Data-Sheet-DS40002061A.pdf
- ***Adafruit Learning System. (n.d.). Using a LM35 Temperature Sensor.*** https://learn.adafruit.com/tmp36-temperature-sensor

- ***Instructables. (n.d.). How to Interface a Water Level Sensor with Arduino.*** https://www.instructables.com/How-to-Interface-a-Water-Level-Sensor-with-Arduino/
- ***Arduino Project Hub. (n.d.). How to use the HC-06 Bluetooth Module.*** https://create.arduino.cc/projecthub/Aritro/how-to-use-the-hc-06-bluetooth-module-958ae
- ***Random Nerd Tutorials. (n.d.). Guide for DHT11/DHT22 Humidity and Temperature Sensor with Arduino.*** https://randomnerdtutorials.com/complete-guide-for-dht11dht22-humidity-and-temperature-sensor-with-arduino/