COMP413 Internet of Things (IoT)

Project Final Report

IoT Based ECG & Temperature Monitoring System

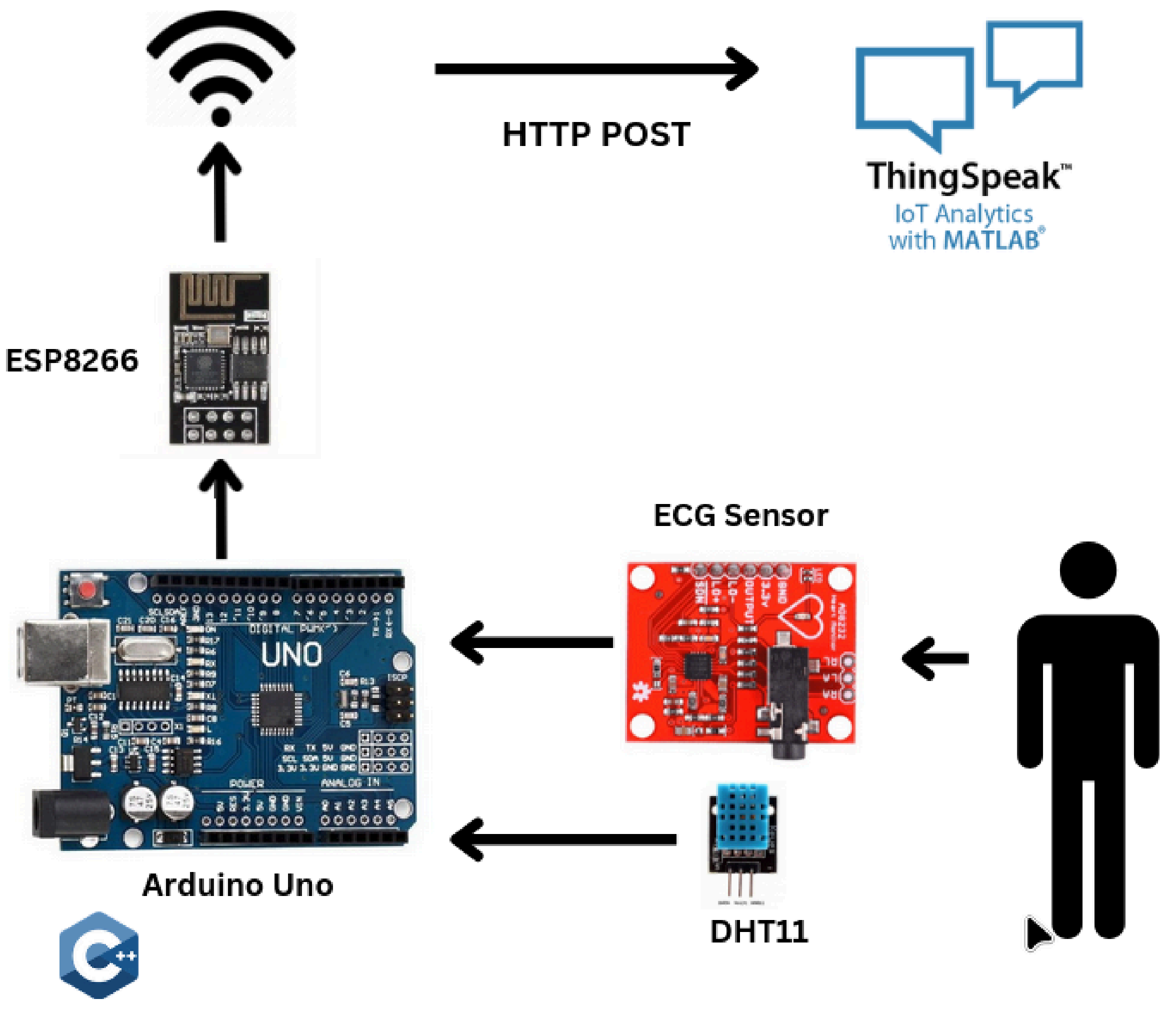
Team Members:
Elif Fidancı - 2011011042
Muhsin Ertekin - 110510210

Introduction

The increasing demand for remote healthcare solutions has led to the integration of Internet of Things (IoT) technologies in the medical field. IoT-enabled health monitoring systems provide real-time insights into patients' health, reducing the need for frequent hospital visits while enabling proactive care. In this project, we developed an IoT-based ECG and temperature monitoring system aimed at tracking a patient's vitals remotely.

This system utilizes ECG and DHT11 sensors to measure heart activity and body temperature, respectively. The sensors are interfaced with an Arduino Uno microcontroller, which processes the data by calculating the average of readings over 5-second intervals. The processed data is transmitted to a cloud platform, ThingSpeak, via the ESP8266 WiFi module, allowing healthcare professionals or caregivers to monitor the patient's condition in real-time. This project serves as a cost-effective and scalable prototype for health monitoring in remote or underserved areas.

System Model



The system architecture for the IoT-based ECG and temperature monitoring system is designed to ensure accurate data acquisition, processing, and transmission to a cloud platform. The architecture consists of the following components:

1. **Sensors:**
    - **ECG Sensor:** Measures the electrical activity of the heart and generates raw ECG signals.
    - **DHT11 Sensor:** Captures temperature data from the patient's environment.
2. **Microcontroller:**
    - **Arduino Uno:** Acts as the central processing unit for the system. It reads raw data from the sensors, calculates the average of readings over 5 seconds, and prepares the data for transmission.
3. **WiFi Module:**

    - **ESP8266:** Facilitates wireless connectivity to the internet. It is connected to the Arduino Uno via jumper wires and enables HTTP POST requests to the ThingSpeak platform.
4. **Cloud Platform:**
    - **ThingSpeak:** A cloud-based IoT analytics platform used to store, analyze, and visualize sensor data. Data transmitted by the ESP8266 is logged and displayed on the ThingSpeak dashboard for monitoring.
5. **Communication Flow:**
    - Sensor data is captured by the ECG and DHT11 sensors.
    - The Arduino Uno processes the data by calculating the average of the readings over a 5-second interval.
    - The ESP8266 module transmits the averaged data to ThingSpeak using HTTP POST requests.
    - ThingSpeak logs the incoming data and provides visualizations for the ECG and temperature readings.
6. **Circuit Connections:**
    - The ECG and DHT11 sensors are interfaced with the Arduino Uno through their respective pins.
    - The ESP8266 is connected to the Arduino Uno using jumper wires, ensuring a stable communication link for data transmission.

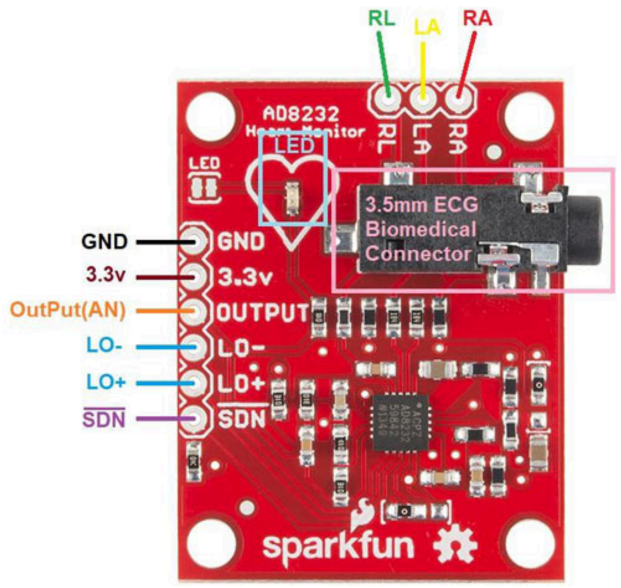
This architecture ensures a streamlined and efficient system for remote monitoring of ECG and temperature data, making it an essential tool for modern healthcare solutions.

---

6. Hardware Design

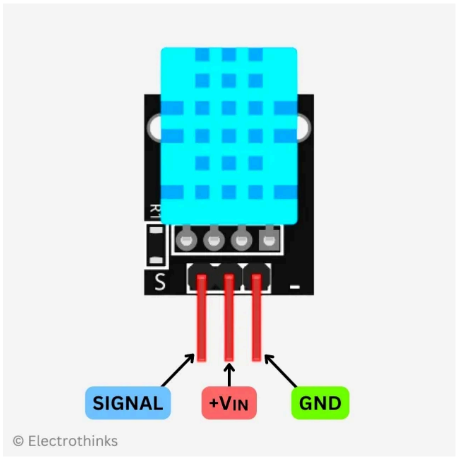
Sensors

ECG Sensor



- **Specifications:**
    - Input Voltage: 3.3V–5V.
    - Output: Analog signal representing ECG waveforms.
    - Leads: Standard electrode cables for signal acquisition.
- **Working Principle:**
  The ECG sensor captures the electrical activity of the heart by detecting voltage changes on the skin's surface. These signals are amplified and filtered to produce a clean analog output that the Arduino Uno reads.

DHT11 Sensor

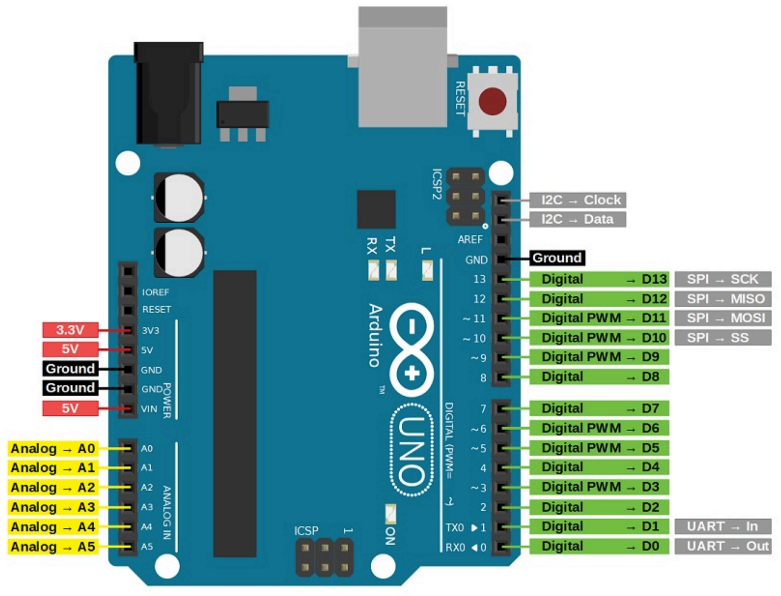


- **Specifications:**
    - Voltage Range: 3.5V–5.5V.
    - Temperature Range: 0°C–50°C with ±2°C accuracy.

    - Output: Digital signal.
- **Temperature Sensing Mechanism:**
  The DHT11 uses a thermistor to measure temperature changes. It converts the resistance changes caused by temperature fluctuations into a digital signal, which is transmitted to the Arduino Uno.

Microcontroller

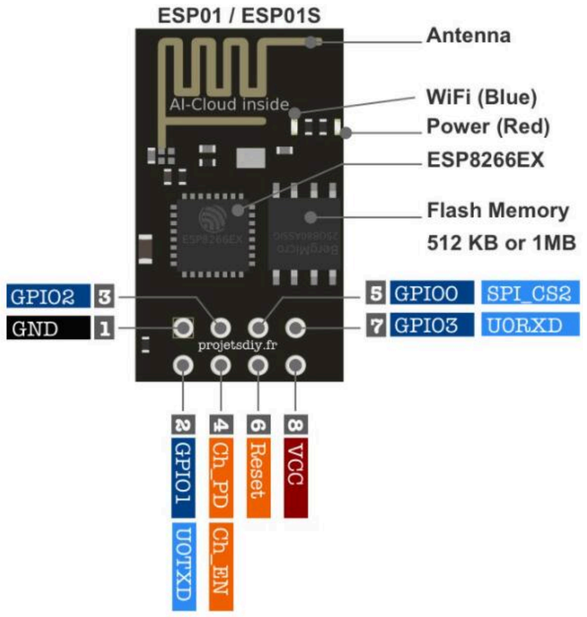
Arduino Uno



- **Specifications:**
    - Microcontroller: ATmega328P.
    - Operating Voltage: 5V.
    - Digital I/O Pins: 14 (6 PWM outputs).
    - Analog Input Pins: 6.
    - Flash Memory: 32 KB.
    - Communication Ports: UART, I2C, SPI.
- **Role in the Project:**
  The Arduino Uno serves as the central hub, interfacing with both sensors and the WiFi module. It reads analog and digital signals from the ECG and DHT11 sensors, respectively, processes the data, and sends it to the ESP8266 module for cloud transmission.

WiFi Module

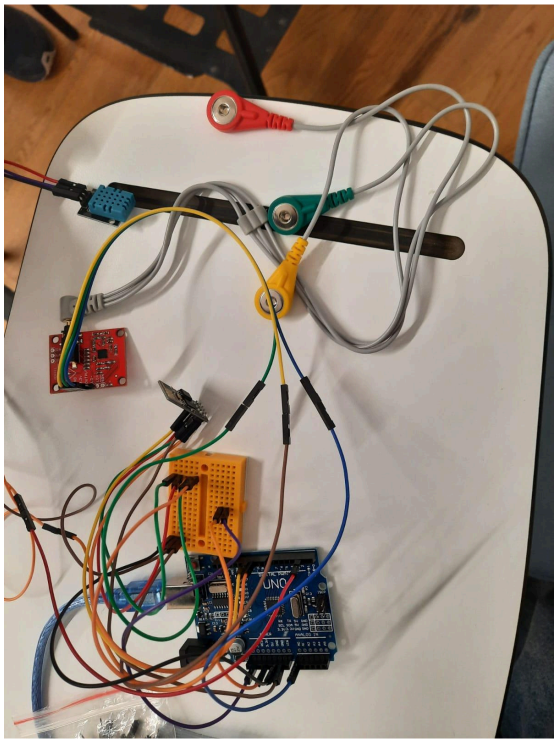
ESP8266



- **Specifications:**
  - Operating Voltage: 3.3V.
  - WiFi Standard: IEEE 802.11 b/g/n.
  - GPIO Pins: 8.
  - Communication Protocols: UART, SPI, I2C.
- **Configuration and Connection with Arduino Uno:**
  The ESP8266 connects to the Arduino Uno through UART pins (TX and RX) with jumper wires. It is configured to use HTTP POST requests to send processed data to the ThingSpeak platform. Proper voltage regulation and a logic level shifter are used to ensure compatibility with the Arduino's 5V logic.

Connections

Jumper Wires



- **Pin Configurations and Setup:**
  - **ECG Sensor:** Analog output pin connected to an analog input pin on the Arduino Uno. Power and ground pins connected to 5V and GND of the Arduino.
  - **DHT11 Sensor:** Digital output pin connected to a digital input pin on the Arduino Uno. Power and ground pins connected to 5V and GND.
  - **ESP8266 Module:** TX and RX pins of the ESP8266 connected to digital pins on the Arduino Uno (using voltage dividers if necessary). Power connected to a 3.3V source.
    These connections enable seamless communication between components, ensuring reliable data acquisition and transmission.

---

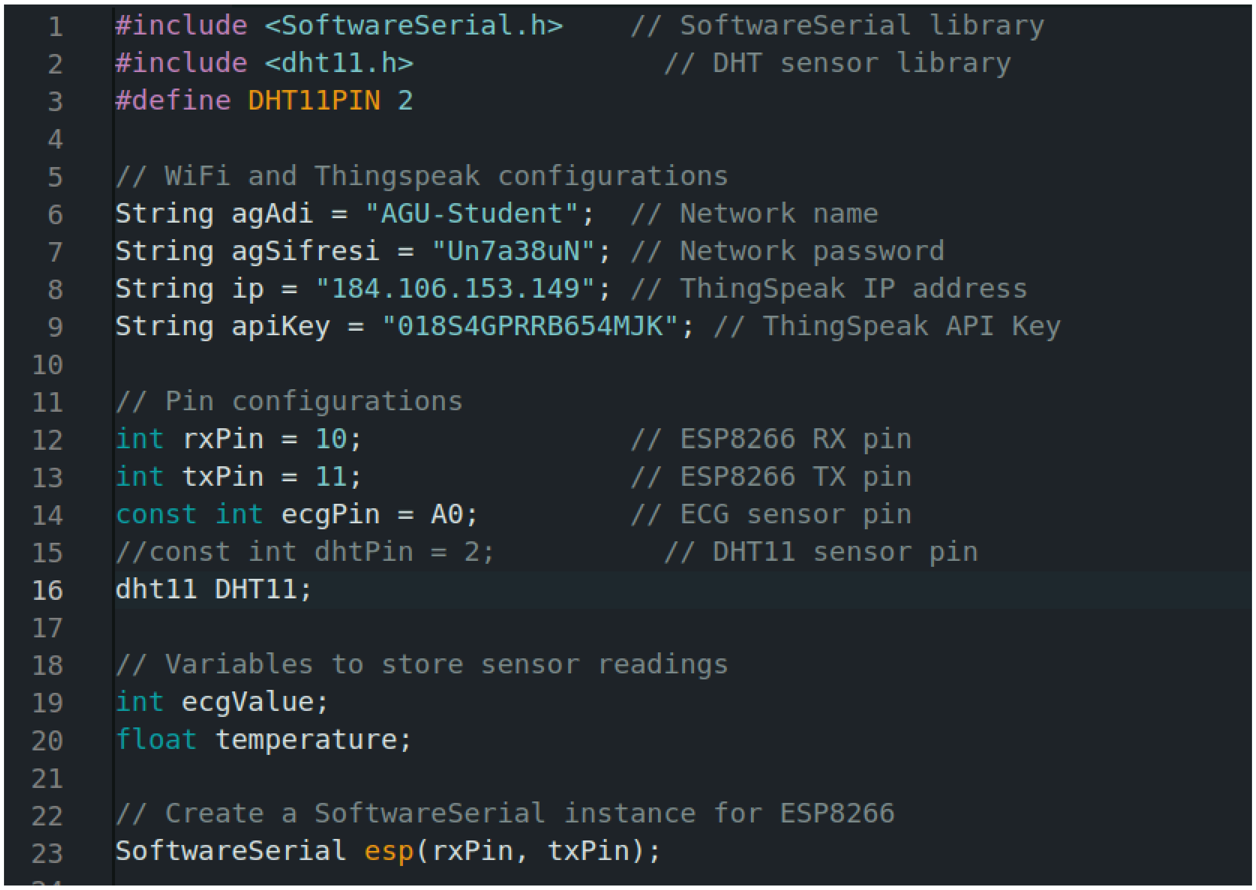
8. Methodology

- **Data Acquisition:**
  - Reading data from ECG and DHT11 sensors.
- **Data Processing:**
  - Averaging 5 seconds of sensor readings.
- **Data Transmission:**

    - HTTP POST requests every 5 seconds to ThingSpeak.
- **Visualization:**
    - Monitoring ECG and temperature data on ThingSpeak dashboard.

---

9. Software Design



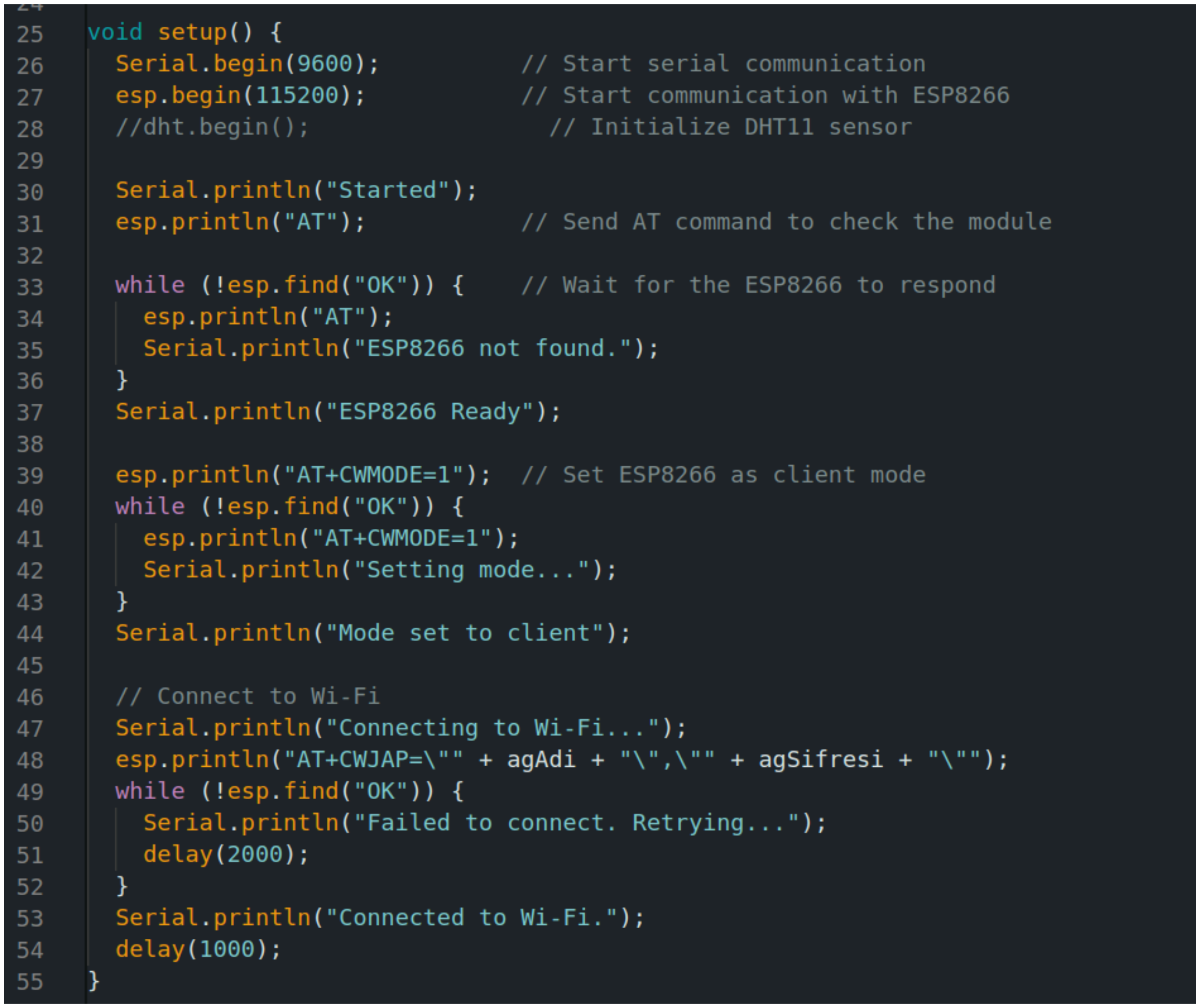
```
#include <SoftwareSerial.h>    // SoftwareSerial library
#include <dht11.h>                // DHT sensor library
#define DHT11PIN 2

// WiFi and Thingspeak configurations
String agAdi = "AGU-Student";  // Network name
String agSifresi = "Un7a38uN"; // Network password
String ip = "184.106.153.149"; // ThingSpeak IP address
String apiKey = "018S4GPRRB654MJK"; // ThingSpeak API Key

// Pin configurations
int rxPin = 10;                // ESP8266 RX pin
int txPin = 11;                // ESP8266 TX pin
const int ecgPin = A0;         // ECG sensor pin
//const int dhtPin = 2;          // DHT11 sensor pin
dht11 DHT11;

// Variables to store sensor readings
int ecgValue;
float temperature;

// Create a SoftwareSerial instance for ESP8266
SoftwareSerial esp(rxPin, txPin);
```



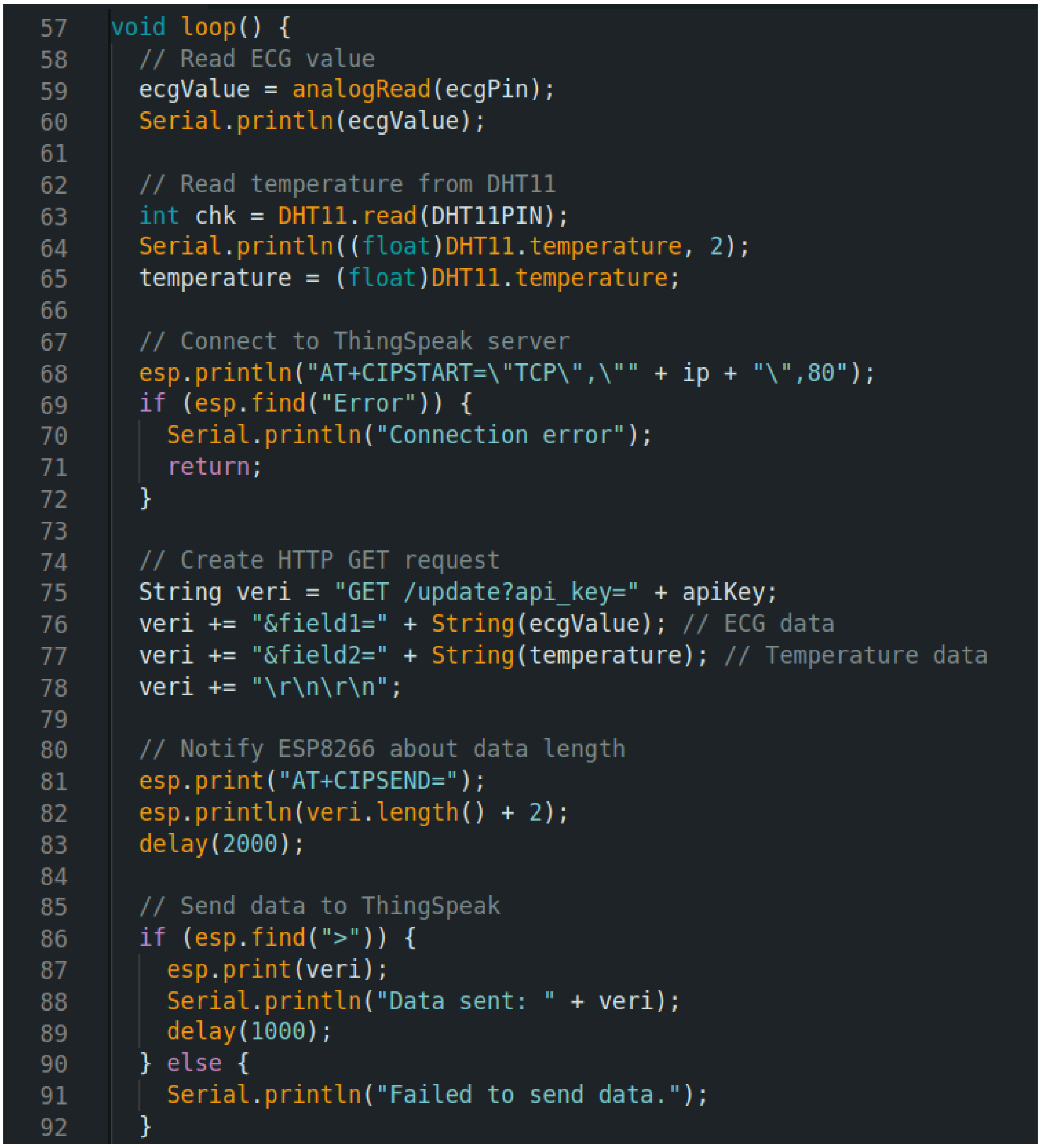
```
void setup() {
  Serial.begin(9600);          // Start serial communication
  esp.begin(115200);           // Start communication with ESP8266
  //dht.begin();                 // Initialize DHT11 sensor

  Serial.println("Started");
  esp.println("AT");           // Send AT command to check the module

  while (!esp.find("OK")) {    // Wait for the ESP8266 to respond
    esp.println("AT");
    Serial.println("ESP8266 not found.");
  }
  Serial.println("ESP8266 Ready");

  esp.println("AT+CWMODE=1");  // Set ESP8266 as client mode
  while (!esp.find("OK")) {
    esp.println("AT+CWMODE=1");
    Serial.println("Setting mode...");
  }
  Serial.println("Mode set to client");

  // Connect to Wi-Fi
  Serial.println("Connecting to Wi-Fi...");
  esp.println("AT+CWJAP=\"" + agAdi + "\",\"" + agSifresi + "\"");
  while (!esp.find("OK")) {
    Serial.println("Failed to connect. Retrying...");
    delay(2000);
  }
  Serial.println("Connected to Wi-Fi.");
  delay(1000);
}
```



```
void loop() {
  // Read ECG value
  ecgValue = analogRead(ecgPin);
  Serial.println(ecgValue);

  // Read temperature from DHT11
  int chk = DHT11.read(DHT11PIN);
  Serial.println((float)DHT11.temperature, 2);
  temperature = (float)DHT11.temperature;

  // Connect to ThingSpeak server
  esp.println("AT+CIPSTART=\"TCP\",\"" + ip + "\",80");
  if (esp.find("Error")) {
    Serial.println("Connection error");
    return;
  }

  // Create HTTP GET request
  String veri = "GET /update?api_key=" + apiKey;
  veri += "&field1=" + String(ecgValue); // ECG data
  veri += "&field2=" + String(temperature); // Temperature data
  veri += "\r\n\r\n";

  // Notify ESP8266 about data length
  esp.print("AT+CIPSEND=");
  esp.println(veri.length() + 2);
  delay(2000);

  // Send data to ThingSpeak
  if (esp.find(">")) {
    esp.print(veri);
    Serial.println("Data sent: " + veri);
    delay(1000);
  } else {
    Serial.println("Failed to send data.");
  }
```

Code Snippets and Explanation

Library Imports and Pin Definitions:

```
#include <SoftwareSerial.h>
#include <dht11.h>
#define DHT11PIN 2
```

- The **SoftwareSerial library** is used for serial communication with the ESP8266.
- The **DHT11 library** is included for temperature sensing.

WiFi and ThingSpeak Configuration:

```
String agAdi = "AGU-Student";
String agSifresi = "Un7a38uN";
String ip = "184.106.153.149";
String apiKey = "018S4GPRRB654MJK";
```

- ECG values are read as analog inputs.
- Temperature values are fetched using the DHT11 library.

HTTP GET Request Creation:

```
String veri = "GET /update?api_key=" + apiKey;
veri += "&field1=" + String(ecgValue);
veri += "&field2=" + String(temperature);
```

- Creates an HTTP GET request with the ECG and temperature data.

Data Transmission to ThingSpeak:

```
esp.println("AT+CIPSTART=\"TCP\",\"" + ip + "\",80");
esp.print("AT+CIPSEND=");
esp.println(veri.length() + 2);
esp.print(veri);
```

- Establishes a TCP connection to the ThingSpeak server and sends data using the HTTP GET request.

Monitoring and Debugging:

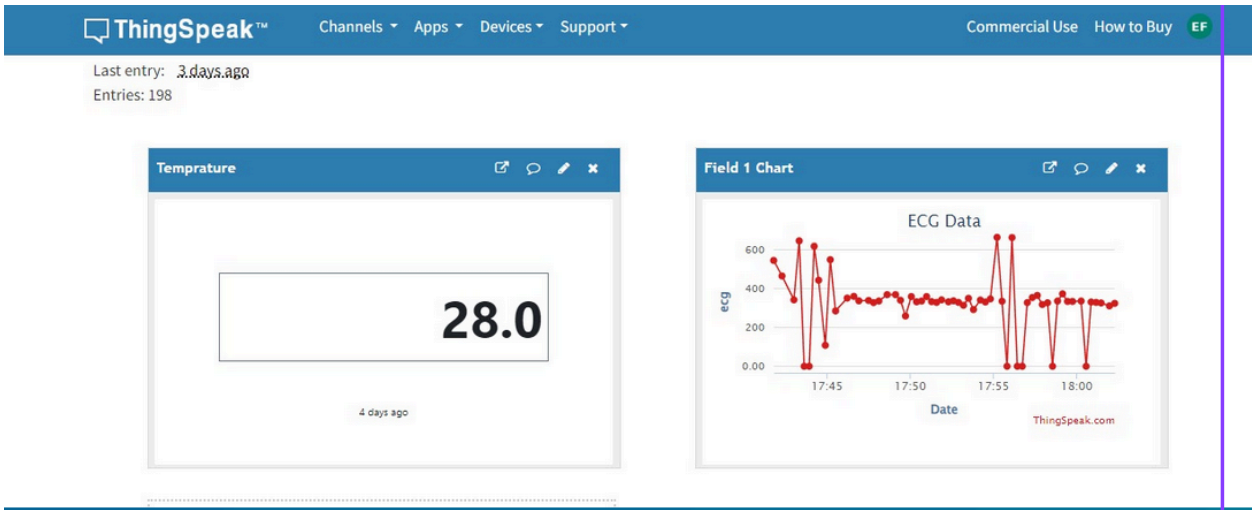
```
Serial.println("Data sent: " + veri);
Serial.println("Closing connection.");
```

Outputs log messages to the serial monitor to confirm successful operation and identify any issues.

Results:

- Effective integration of sensors and microcontroller for data acquisition.
- Reliable data transmission to ThingSpeak for real-time monitoring.
- Efficient data transmission is crucial for real-time IoT health monitoring.
- Interfacing various components can be challenging but is essential for creating cohesive systems.
- Cloud platforms like ThingSpeak offer accessible solutions for remote health monitoring.

- **Cloud Platform:**
    - ThingSpeak:
        - Simple set up a ThingSpeak Channel,
        - Connect via its API key and monitor.



Last entry: 3 days ago
Entries: 198

28.0
4 days ago

12. Conclusion

This project successfully developed an IoT-based ECG and temperature monitoring system using an Arduino Uno, ECG and DHT11 sensors, and an ESP8266 WiFi module. The system captures and transmits heart activity and temperature data to ThingSpeak for remote monitoring.

This project highlights the potential of IoT in healthcare and provides a scalable foundation for future applications.