# Mini-project #3: Data Upload Using Bluetooth


## TEAM MEMBERS

## RAJ VARSHITH – 16325723

## SHIVA REDDY – 16352875

## KARTHIKEYA – 16354793


---

**Introduction**

This project uses an **ESP32** microcontroller to read temperature and humidity data from a **DHT11** sensor and broadcast it over **Bluetooth Low Energy (BLE)**. The data is transmitted via BLE characteristics that support notification. Additionally, a simulated battery level characteristic is implemented, which decreases over time to demonstrate power drain. The BLE device is named **"Team 7"**, and the project adheres to BLE standards for transmitting environmental sensor data.

**Components Required:**

- ESP32 Development Board
- DHT11 Temperature and Humidity Sensor
- Jumper wires
- Breadboard (optional)
- BLE Debugging App (e.g., nRF Connect or LightBlue)

**Libraries Used:**

1. **BLEDevice.h**: Manages BLE functionality on the ESP32.
2. **DHT.h**: For interfacing with the DHT11 sensor.
3. **Wire.h**: Enables communication over I2C (if needed for future expansions).

**Project Setup**

**Hardware Connections:**

1. **Connect the DHT22 sensor:**
    - **VCC** to **3.3V** on the ESP32
    - **GND** to **GND** on the ESP32
    - **Data Pin** to **GPIO 14** on the ESP32

**Software Setup:**

1. **Install Arduino IDE**: Ensure the latest version of the Arduino IDE is installed.
2. **Install Required Libraries**:
    - Install the DHT sensor library using the Arduino Library Manager.
    - Install the BLE library for ESP32.
3. **Install BLE Debugging App**:
    - Install **nRF Connect** or **LightBlue** on your mobile phone to verify BLE services and characteristics.

---

**Code Overview**

**1. Initialization**

```
BLEDevice::init("Team 7");
```

- The BLEDevice::init function initializes the BLE device and sets the device name to **"Team 7"**.

**2. BLE Services and Characteristics**

- The project implements the **Environmental Sensing Service** with two characteristics: **Temperature** and **Humidity**. Additionally, a **Battery Service** is created to simulate battery level.

**BLE UUIDs:**

- **Environmental Sensing Service UUID**: 0x181A
- **Temperature Characteristic UUID**: 0x2A6E
- **Humidity Characteristic UUID**: 0x2A6F

- **Battery Service UUID**: 0x180F
- **Battery Level Characteristic UUID**: 0x2A19

```
BLEService *envService = pServer->createService(ENVIRONMENTAL_SENSING_SERVICE_UUID);

BLECharacteristic *temperatureCharacteristic = envService->createCharacteristic(TEMPERATURE_CHARACTERISTIC_UUID, BLECharacteristic::PROPERTY_NOTIFY);

BLECharacteristic *humidityCharacteristic = envService->createCharacteristic(HUMIDITY_CHARACTERISTIC_UUID, BLECharacteristic::PROPERTY_NOTIFY);
```

- The **Environmental Sensing Service** contains the temperature and humidity characteristics, both set to support notification properties.

**3. DHT11 Sensor Integration**

```
float temperature = dht.readTemperature();

float humidity = dht.readHumidity();
```

- The **DHT11** sensor reads temperature and humidity, and the values are then converted to 16-bit integers for transmission over BLE.

**4. Encoding Data for BLE**

- Temperature and humidity values are scaled by 100 to account for two decimal places.

```
int16_t temperatureBLE = (int16_t)(temperature * 100);

uint16_t humidityBLE = (uint16_t)(humidity * 100);
```

- The sensor values are then notified to the client:

```
temperatureCharacteristic->setValue((uint8_t*)&temperatureBLE, sizeof(int16_t));

humidityCharacteristic->setValue((uint8_t*)&humidityBLE, sizeof(uint16_t));

temperatureCharacteristic->notify();

humidityCharacteristic->notify();
```

- BLE clients receive real-time updates for temperature and humidity through notifications.

**5. Battery Simulation**

```
if (batteryLevel > 0) {
   batteryLevel--;
   batteryCharacteristic->setValue(&batteryLevel, 1);
   batteryCharacteristic->notify();
}
```

- A simple simulation of battery drain reduces the battery level by 1% every minute, notifying connected BLE clients of the updated battery status.

**6. Connection Handling**

```
class MyServerCallbacks : public BLEServerCallbacks {
   void onConnect(BLEServer* pServer) { deviceConnected = true; }
   void onDisconnect(BLEServer* pServer) { deviceConnected = false; pServer->startAdvertising(); }
};
```

- The **MyServerCallbacks** class handles BLE connections and disconnections. When the device disconnects, it automatically starts advertising again to allow new connections.

**7. Advertising**

```
pServer->getAdvertising()->start();
Serial.println("Waiting for a client connection to notify...");
```

- The ESP32 begins advertising itself as a BLE device after setting up the services and characteristics.

---

**Testing**

**Steps:**

1. **Upload Code to ESP32**: Flash the ESP32 with the provided code.
2. **Open BLE Debugging App**: Use an app like **nRF Connect** to find **"Team 7"** and connect.

3. **Verify Notifications**: Ensure that the temperature, humidity, and battery level notifications are being received on the mobile app.
4. **Check Serial Monitor**: Use the Arduino IDE Serial Monitor to observe real-time readings of temperature, humidity, and battery level.

---

## Code Screenshots:

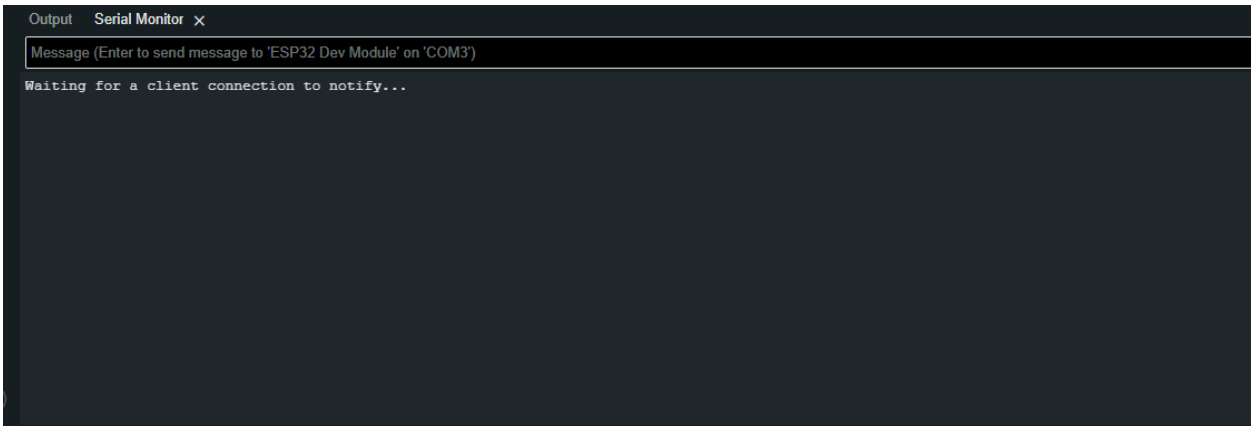
BLE.ino

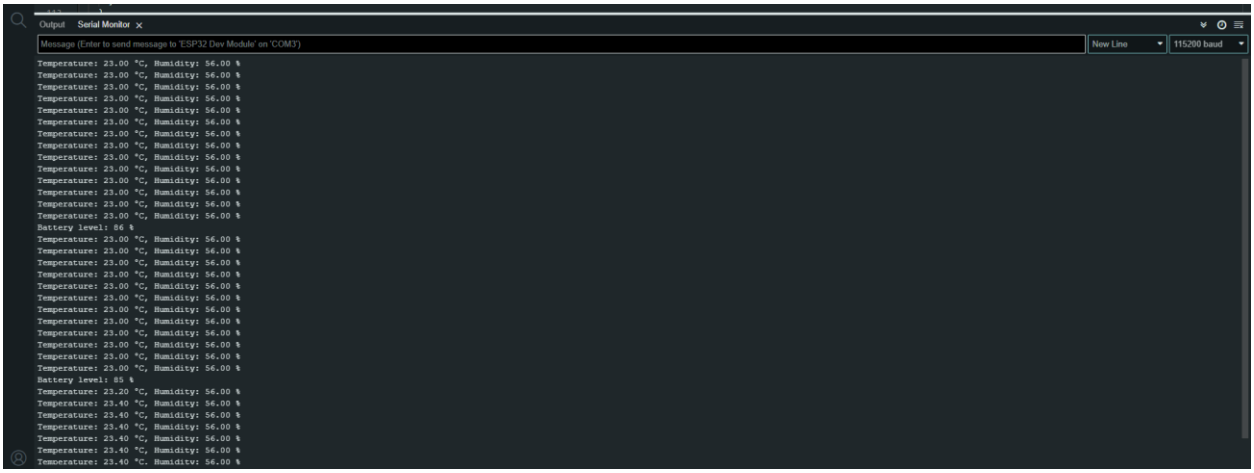
```ino
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
#include <DHT.h>

#define DHTPIN 14 // Pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11 (AM2302) sensor

DHT dht(DHTPIN, DHTTYPE);

// BLE UUIDs
#define ENVIRONMENTAL_SENSING_SERVICE_UUID "181A"
#define TEMPERATURE_CHARACTERISTIC_UUID "2A6E"
#define HUMIDITY_CHARACTERISTIC_UUID "2A6F"
#define BATTERY_SERVICE_UUID "180F"
#define BATTERY_LEVEL_CHARACTERISTIC_UUID "2A19"

BLECharacteristic *temperatureCharacteristic;
BLECharacteristic *humidityCharacteristic;
BLECharacteristic *batteryCharacteristic;

bool deviceConnected = false;
uint8_t batteryLevel = 100; // Simulate 100% battery level
unsigned long lastBatteryUpdate = 0;
unsigned long lastSensorUpdate = 0;

// Server callbacks to handle connection and disconnection
class MyServerCallbacks : public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
      deviceConnected = true;
    }

    void onDisconnect(BLEServer* pServer) {
      deviceConnected = false;
      pServer->startAdvertising(); // Restart advertising
    }
};

void setup() {
  Serial.begin(115200);
  dht.begin();

  // Create the BLE Device
  BLEDevice::init("Team 7");
```

```
  // Create the BLE Server
  BLEServer *pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());

  // Create the Environmental Sensing Service
  BLEService *envService = pServer->createService(ENVIRONMENTAL_SENSING_SERVICE_UUID);

  // Create Temperature Characteristic
  temperatureCharacteristic = envService->createCharacteristic(
                                TEMPERATURE_CHARACTERISTIC_UUID,
                                BLECharacteristic::PROPERTY_NOTIFY
                              );

  // Create Humidity Characteristic
  humidityCharacteristic = envService->createCharacteristic(
                             HUMIDITY_CHARACTERISTIC_UUID,
                             BLECharacteristic::PROPERTY_NOTIFY
                           );

  // Start the Environmental Sensing Service
  envService->start();

  // Create the Battery Service
  BLEService *batteryService = pServer->createService(BATTERY_SERVICE_UUID);

  // Create Battery Level Characteristic
  batteryCharacteristic = batteryService->createCharacteristic(
                            BATTERY_LEVEL_CHARACTERISTIC_UUID,
                            BLECharacteristic::PROPERTY_NOTIFY
                          );

  // Start the Battery Service
  batteryService->start();

  // Start advertising
  pServer->getAdvertising()->start();
  Serial.println("Waiting for a client connection to notify...");
}

void loop() {
  // Simulate temperature and humidity readings every 5 seconds
  if (millis() - lastSensorUpdate >= 5000) {
    lastSensorUpdate = millis();
```

```
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();

    // Update and notify temperature and humidity
    if (!isnan(temperature) && !isnan(humidity)) {
      // Convert temperature to sint16 in 0.01°C
      int16_t temperatureValue = (int16_t)(temperature * 100);
      // Convert humidity to uint16 in 0.01%
      uint16_t humidityValue = (uint16_t)(humidity * 100);

      // Set and notify the temperature and humidity values
      temperatureCharacteristic->setValue((uint8_t*)&temperatureValue, sizeof(temperatureValue));
      humidityCharacteristic->setValue((uint8_t*)&humidityValue, sizeof(humidityValue));

      if (deviceConnected) {
        temperatureCharacteristic->notify();
        humidityCharacteristic->notify();
        Serial.print("Temperature: ");
        Serial.print(temperature);
        Serial.print(" °C, Humidity: ");
        Serial.print(humidity);
        Serial.println(" %");
      }
    }
  }

  // Simulate battery drain every minute
  if (millis() - lastBatteryUpdate >= 60000) {
    lastBatteryUpdate = millis();

    if (batteryLevel > 0) {
      batteryLevel--;
      batteryCharacteristic->setValue(&batteryLevel, 1);
      if (deviceConnected) {
        batteryCharacteristic->notify();
        Serial.print("Battery level: ");
        Serial.print(batteryLevel);
        Serial.println(" %");
      }
    }
  }

  delay(100);
}
```

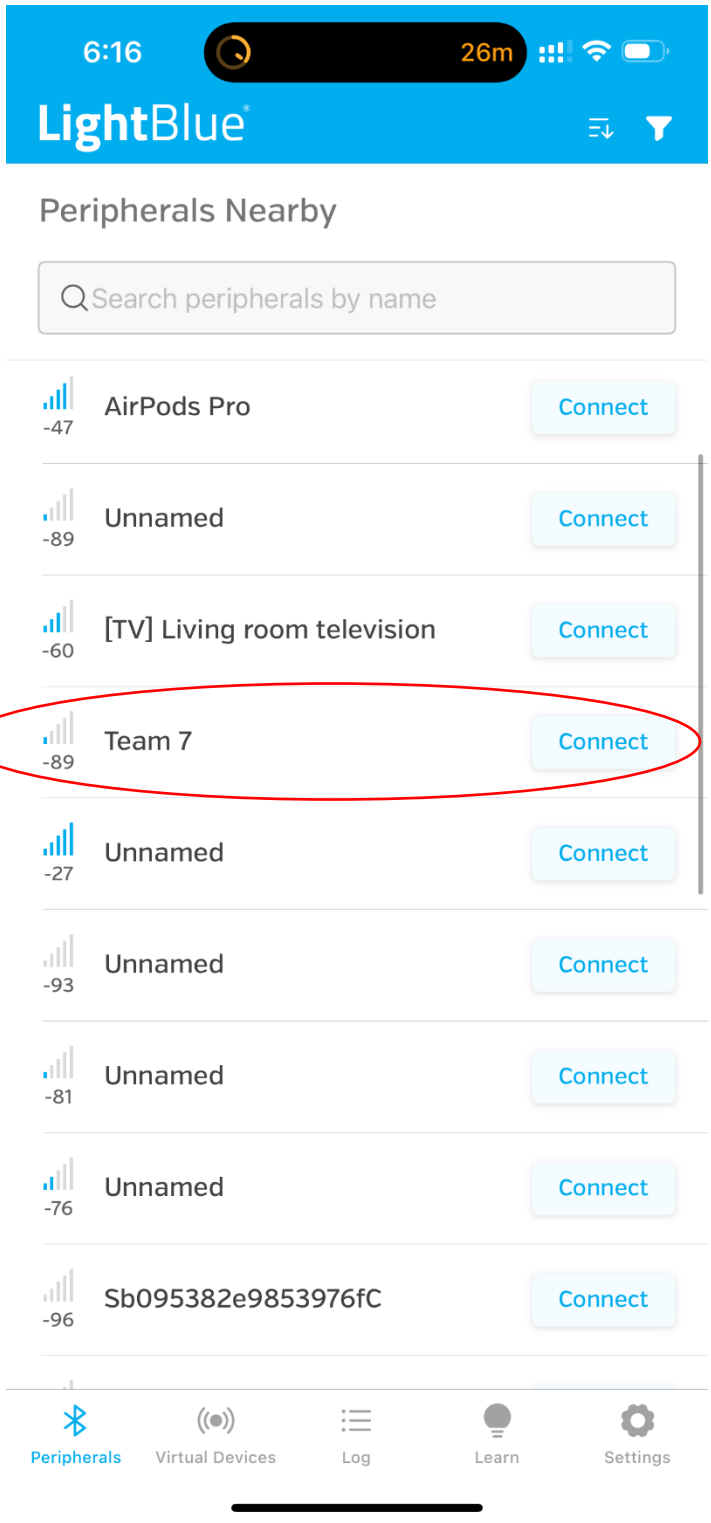
## Outputs:


Output
Serial Monitor
Message (Enter to send message to 'ESP32 Dev Module' on 'COM3')
Waiting for a client connection to notify...

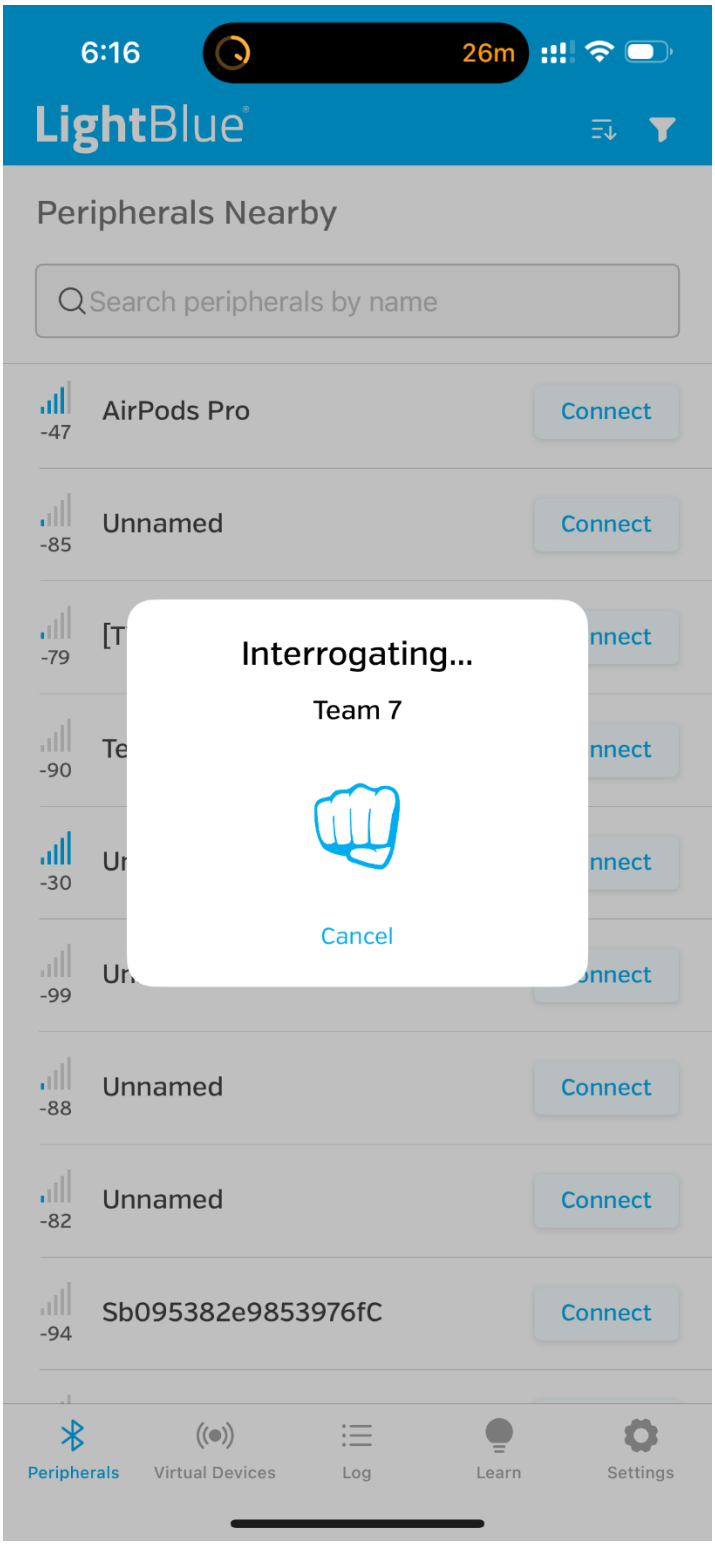

Output
Serial Monitor
Message (Enter to send message to 'ESP32 Dev Module' on 'COM3')
New Line
115200 baud
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Battery level: 86 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Temperature: 23.00 °C, Humidity: 56.00 %
Battery level: 85 %
Temperature: 23.20 °C, Humidity: 56.00 %
Temperature: 23.40 °C, Humidity: 56.00 %
Temperature: 23.40 °C, Humidity: 56.00 %
Temperature: 23.40 °C, Humidity: 56.00 %
Temperature: 23.40 °C, Humidity: 56.00 %
Temperature: 23.40 °C, Humidity: 56.00 %
Temperature: 23.40 °C, Humidity: 56.00 %

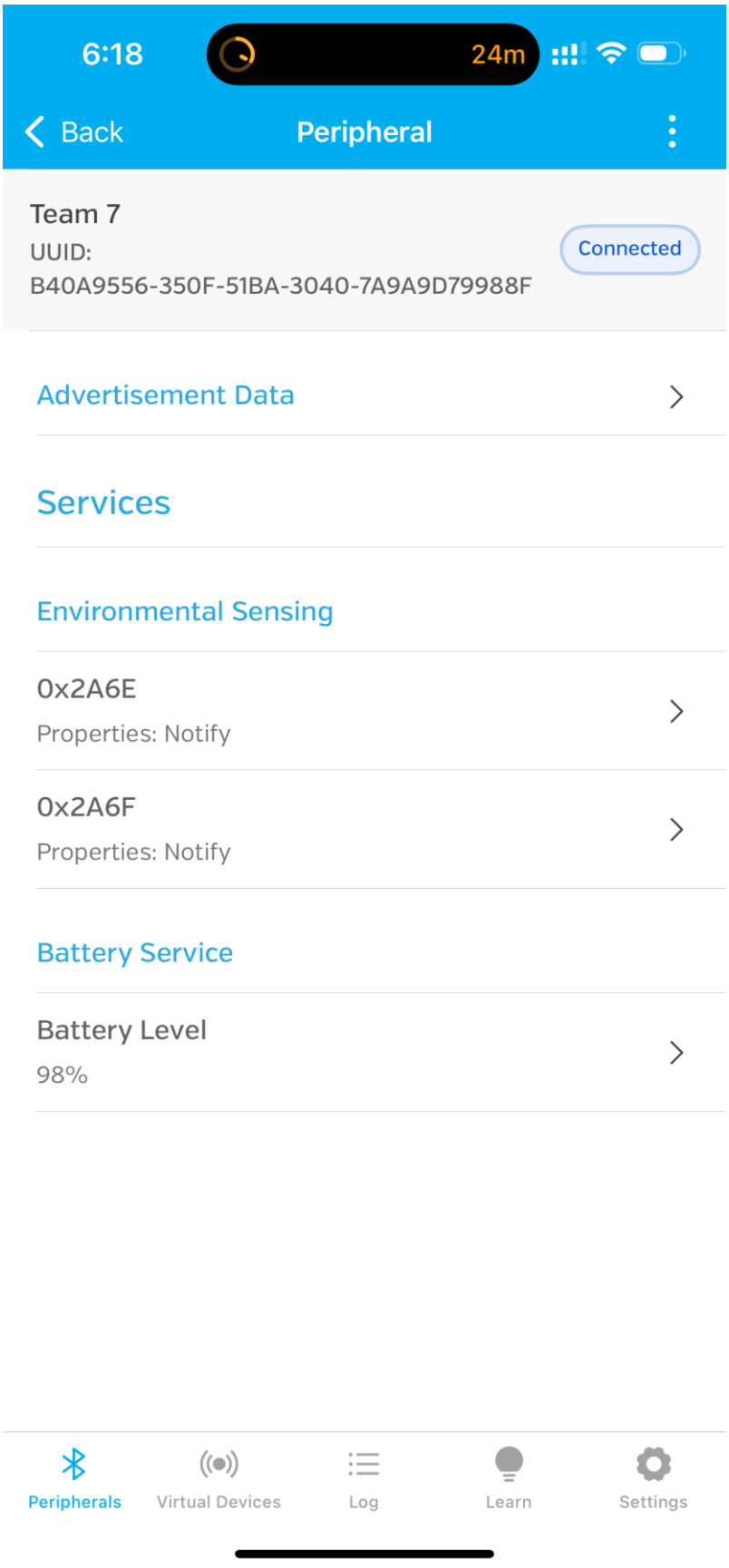
**Output Screenshots from App:**

Team 7:

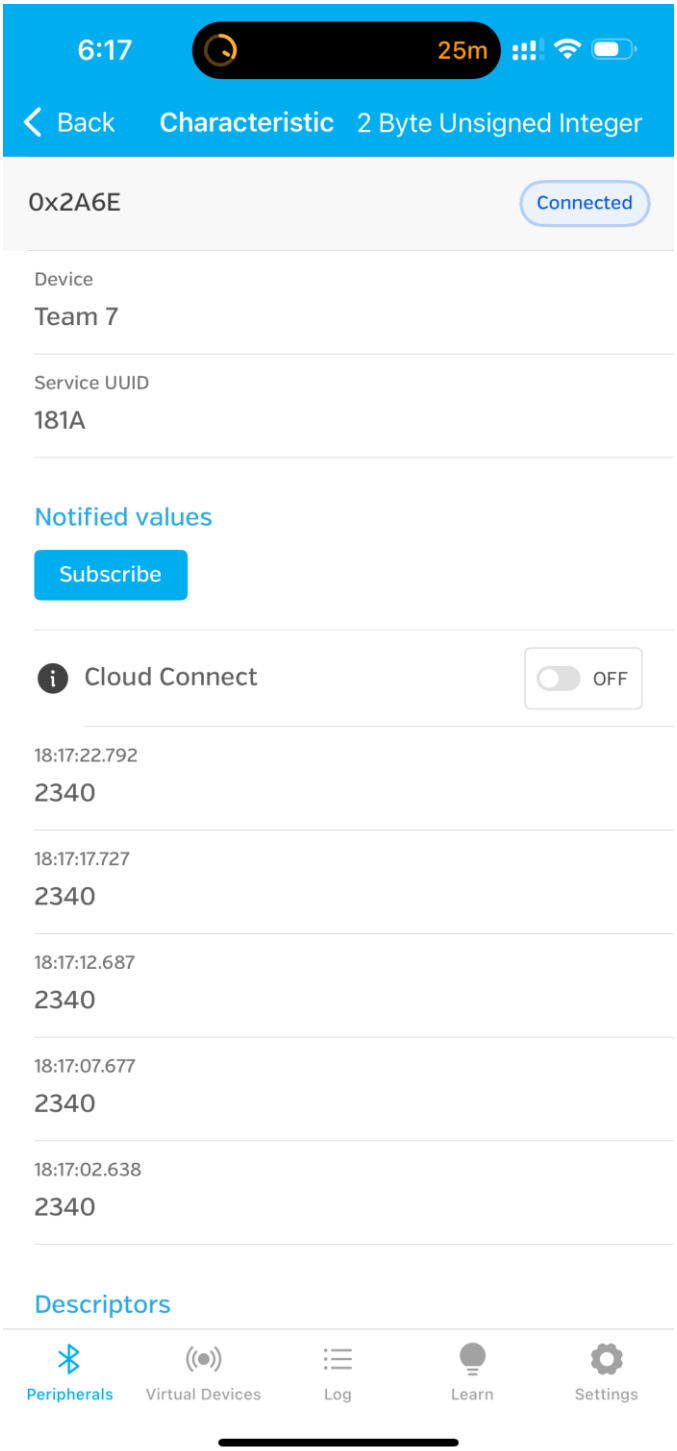


**Connecting to the Device**

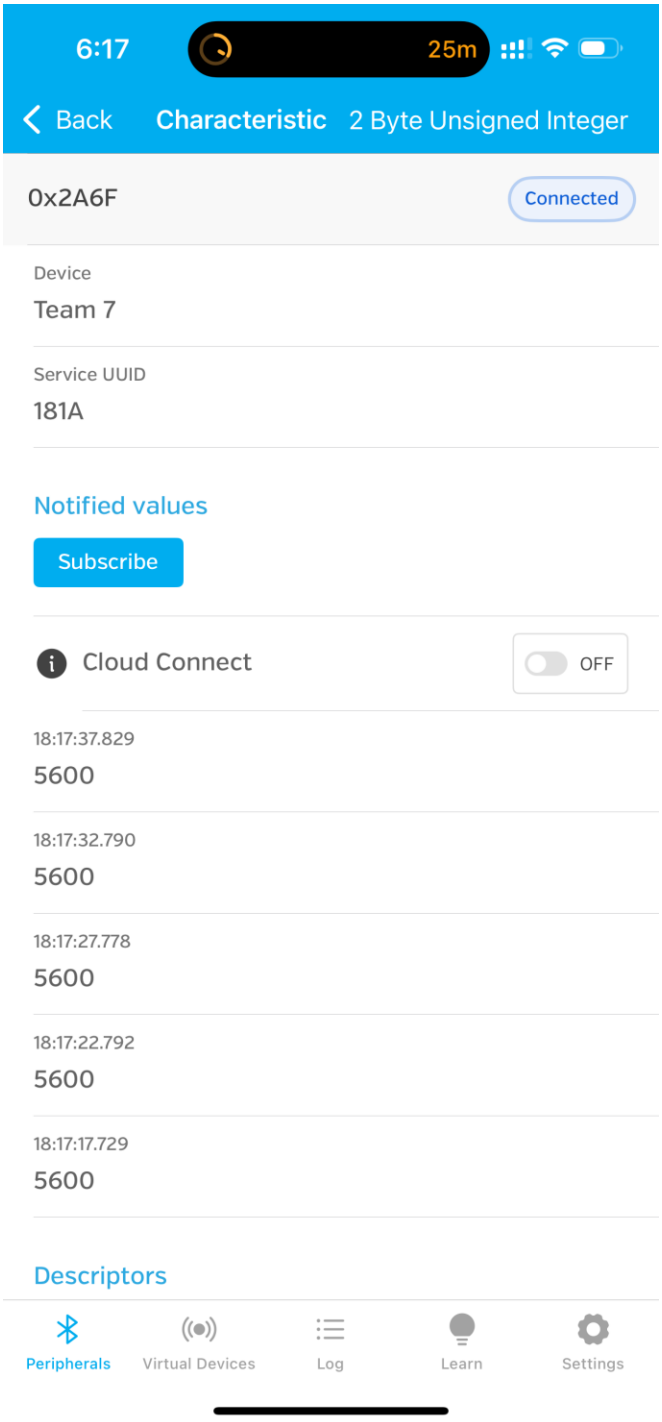



6:18
24m
Back
Peripheral
Team 7
UUID:
B40A9556-350F-51BA-3040-7A9A9D79988F
Connected
Advertisement Data
Services
Environmental Sensing
0x2A6E
Properties: Notify
0x2A6F
Properties: Notify
Battery Service
Battery Level
98%
Peripherals
Virtual Devices
Log
Learn
Settings

**Reading Temperature:**



**Reading Humidity:**



**Reading the Battery Percentage:**