Project: ESP32 to Mobile App using BLE

Introduction

The purpose of this project is to develop a system that collects data from various sensors (EC value, pH value, water pressure, temperature, and weight) and transmits it to a mobile app using Bluetooth Low Energy (BLE) technology. The system utilizes an ESP32 microcontroller unit to act as a BLE server, creating a service with five characteristics for each sensor data transmission. The mobile app functions as a BLE client, receiving and displaying the sensor data in real-time.

Components Used

* ESP32 microcontroller board
* EC sensor (Gravity: Analog Electrical Conductivity Sensor /Meter V2 (K=1))
* pH sensor (Gravity: Analog pH Sensor/Meter Kit V2)
* Water pressure sensor (Gravity: Analog Water Pressure Sensor)
* Temperature sensor (Gravity: Waterproof DS18B20 Temperature Sensor Kit)
* Weight sensor (use RS232 protocol)
* Mobile device with BLE capability (Android/iOS)

System Architecture

The system architecture consists of the following components:

**ESP32 microcontroller unit**: Responsible for collecting sensor data, creating a BLE server, and transmitting data to the mobile app.

**Sensor Module**: Each sensor is connected to the ESP32 board to measure the respective parameter.

**Mobile App**: Serves as a BLE client, connects to the ESP32 server, and receives and displays the sensor data in a user-friendly interface.

ESP32 as BLE Server

The ESP32 microcontroller is programmed to create a BLE server and provide the following functionality:

1. Advertising the BLE service to nearby devices.
2. Establishing a connection with the mobile app when it detects a compatible client.
3. Organizing sensor data transmission through characteristics within the BLE service.

BLE Service and Characteristics

The BLE service created by the ESP32 server includes the following characteristics for data transmission:

1. **EC Value Characteristic**: Provides the real-time EC (electrical conductivity) value measured by the sensor.
2. **pH Value Characteristic**: Transmits the current pH value obtained from the pH sensor.
3. **Water Pressure Characteristic**: Sends the water pressure reading from the pressure sensor.
4. **Temperature Characteristic**: Delivers the temperature reading recorded by the temperature sensor.
5. **Weight Characteristic**: Transmits the weight measurement obtained from the weight sensor.

Code Explanation

At first code is divided into multiple header file such as –

* **ph.h**: It contains three function –

1. configPH() 🡪 Configure the DFRobot Analog pH Sensor.
2. getPH() 🡪 Read the ADC value and convert it into pH value.
3. sendPH() 🡪 Send the pH value on its particular characteristic.

* **ec.h**:

1. configEC() 🡪 Configure the DFRobot Analog EC Sensor.
2. getEC() 🡪 Read the ADC value and convert it into EC value.
3. sendEC() 🡪 Send the EC value on its particular characteristic.

* **pressure.h**

1. getPressure() 🡪 Read the ADC value and subtracts it’s offset and get water pressure value.
2. sendPressure() 🡪 Send the pressure value on its particular characteristic.

* temperature.h
  + Required Library: OneWire(<https://github.com/PaulStoffregen/OneWire>)

1. readTemperature() 🡪 Read temperature using OneWire library in Celsius unit.
2. sendTemperature() 🡪 Send the temperature on its particular characteristic.

* weight.h

1. configRS232() 🡪 Configure the RS232 module as a Serial communication.
2. getWeight() 🡪 Read the serial value as weight.
3. sendWeight() 🡪 Send the weight value on its particular characteristic.

* **ble\_server.h**
* Required Library:

ESP32\_BLE\_Arduino (<https://github.com/nkolban/ESP32_BLE_Arduino/tree/master>)

1. First define service and all characteristic UUID (Universally Unique Identifier)
2. Declare all characteristics.
3. Define two callback function for checking device connectivity and receiving text from client.
4. configBLE() 🡪 Creating a service and five characteristic under the service and also start and advertise the server.
5. send\_data() 🡪 Sending data on the particular characteristic.

These are the required library for measuring pH and EC value from DFRobot and ESP32 compatible. But here contains a little compilation error, so need a little change on these .cpp file. So added these library (.h and .cpp) file on the main code.

* DFRobot\_ESP\_EC.h
* DFRobot\_ESP\_EC.cpp
* DFRobot\_ESP\_PH.h
* DFRobot\_ESP\_PH.cpp

This is the main file where setup() and loop() function is located.

* Final\_code.ino

In setup() function call all configure functions and in loop() function call all get functions for reading sensors data.

Sensor Calibration

There are pH, EC and pressure sensors whose need to calibrate for accurate result. For pH and EC sensor use libraries where required commands for taking these sensors in calibration mood. There are required to put these commands using Serial Monitor. For example –

If use Arduino IDE, first upload the code on ESP32, connect it with the Mobile APP using BLE and open the Serial Monitor and put this command on the Serial Monitor and press enter. After that the sensor goes in calibration mood which is shown on Serial Monitor.

For pH Sensor –

* enterph 🡪 enter the calibration mode.
* calph 🡪 calibrate with the standard buffer solution, two buffer solutions (4.0 and 7.0) will be automatically recognized.
* exitph 🡪 save the calibrated parameters and exit from calibration mode.

For EC Sensor –

* enterec 🡪 enter the calibration mode.
* calec 🡪 calibrate with the standard buffer solution, two buffer solutions (1413us/cm and 12.88ms/cm) will be automatically recognized.
* exitec 🡪 save the calibrated parameters and exit from calibration mode.

For pressure sensor –

connect the 3-pin wire to the ESP-32 (VCC, GND and Signal) without connecting the sensor to the water pipe and run the program for once. Mark down the LOWEST voltage value through the serial monitor and revise the "OffSet" value to complete the calibration.

Mobile App as Client

This App(blechem) is programmed to act as a BLE client. It has the following functionality:

1. If the device’s Bluetooth Adapter state is off it shows the user a screen to turn the adapter on.

2. Then the Find Device screen provides a scanning facility. BLE devices are enlisted in this screen.

3. After building a connection with the ESP32, the App navigates to the sensor Data screen. Here all the data is shown in a separate dashboard.

4. The Sensor Data screen also has two buttons, generating a CSV file and emailing the file to User’s email id.

Below is the illustration,

A picture containing text, screenshot, computer, operating system

Description automatically generated

As the Bluetooth Adapter is off after installing the app. So, this screen is initiated.

Click on **TURN ON**, it will show a pop-up message to allow Bluetooth.

A picture containing text, screenshot, font, software

Description automatically generated

By **Allowing** the Bluetooth will turn on and Navigate to Find device Screen.

A picture containing text, screenshot, software

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

By Clicking the **CONNECT** Button The app will navigate to the Sensor data screen and display the Dashboard and other functionalities.

**Scanning Button**

This button will Scan the available devices. And list like the right-sided pictures

A picture containing text, screenshot, font, design

Description automatically generated

This button will **send mail** to Users mail id.

This Button Will generate a **CSV** file and store it to download folder.

A screenshot of a device

Description automatically generated with medium confidence

First of all, this screen will show Waiting, as the connection between the App and ESP32 server establishes the dashboards will appear with data.

Conclusion

The sensor data collection and transmission system using BLE and ESP32 provides a convenient and efficient method for monitoring and analyzing various parameters. With the ESP32 as the BLE server and the mobile app as the client, users can effortlessly access real-time sensor data, enabling them to make informed decisions based on the collected information. This system offers flexibility and scalability, allowing for the integration of additional sensors or functionalities as per specific requirements.