ECE Laboratory

**DREXEL UNIVERSITY**

**To: Dr. Peters**

**From: Ehi Simon**

**Re: ECE 304 Lab 3 - Bluetooth and Bluetooth Low Energy**

**PURPOSE:**

In this week’s in-lab experiment, you are to remotely control your in-lab circuit by placing the ESP32 in Soft Access Point and Station modes simultaneously. On one HTML page, develop a form for being able to submit LED inputs (blue LED ON/OFF and red LED intensity), and display the following information on the HTML page once the data is submitted:

* BME 280 sensor
* Air temperature
* Air humidity
* Air pressure
* Elevation
* BNO085 sensor
  + Roll
  + Pitch
  + Yaw

**Discussion:**

**A picture containing electronics, electronic engineering, computer component, computer hardware

Description automatically generated**

*Fig. 1. Circuit Connection for Project 2*

The circuit for the lab was built like the one above. It consists of 2 330 resistors, a red LED, a blue LED, a BME280 Environmental Sensor, an Adafruit BNO085 IMU, and an ESP32S microcontroller. Just like last week, an integer (between 0 and 8191) will be sent to the ESP32 for decoding. The breakdown for the number is based on the bit makeup:

* Bit 0(LSB): Take and display temperature measurement (1), or not (0).
* Bit 1: Take and display humidity measurement (1), or not (0)
* Bit 2: Take and display pressure measurement (1), or not (0)
* Bit 3: Take and display altitude measurement (1), or not (0)
* Bit 4: Turn blue LED ON (1) or OFF(0)
* Bits 5-12(MSB): Set the intensity of the red LED.

**BluetoothMain.cpp**

In my main.cpp file, I initialized the multiple libraries that were needed for the sensors to work and provide readings. The figure below shows the libraries I initialized. There is an additional library from the first project and that is the BluetoothSerial header file. This file is what is going to enable me to connect the ESP32 to Bluetooth and control it from a Bluetooth serial terminal.

A picture containing text, screenshot, font

Description automatically generated

*Fig. 2. Figure Showing Initialization of Libraries*

The main function consisted mostly of the setup and loop functions. Before the setup functions, I did some essential things such as defining the pins for the LEDs, defining the PWM parameters, and defining a reference pressure for sea level. Also, I created both a BME object and a BluetoothSerial object. This required the BluetoothSerial header file. After this, I created a structure named euler\_t that defined the yaw, pitch, and roll and set them to be floating values. Lastly, I reset the BNO085 chip. This is shown in the figure below.

A picture containing text, screenshot, font

Description automatically generated

*Fig. 3. Figure Showing Definition of Variables and Creation of Objects*

A screen shot of a computer program

Description automatically generated with low confidence

*Fig. 4. Figure Showing Setup Function*

The figure above is concerned with the setup of the ESP32. The “serial.begin” line starts serial communications with a 115200-bps baud rate. The ESP\_BT.begin(“ESP32\_Peters”) line starts the Bluetooth communication and names the device ESP32\_Peters. The next two lines set the pins for the LEDs for OUTPUT. The two lines below those are specially for the ESP32. They are used to set up the LED behavior for the red LED. The next few lines start the BME280, and check if the BME280 is found. They also start the BNO085 chip, and check if it is found. After this in the setup function, we also print out a prompt for the user to enter an integer between 0 and 8191.

The figure below shows the first main section of the loop function. It starts with the line “int command = Serial.parseInt()” reading a sequence of characters from the serial monitor, converting them into an integer value, and assigning it to the variable command. Then, it prompts the user to enter a number between 0 and 8191. These commands are read using the bitRead function and assigned to variables based on their bits. Based on the command entered, the function will also check if the blue LED is on or off and print the output to the serial terminal, as well as the Bluetooth serial terminal. It will also print the value of the red LED to both terminals. After this, there are if functions that will read the temperature, pressure, humidity, and altitude from the BME sensor and print the value to both terminals.

A screen shot of a computer program

Description automatically generated with medium confidence

*Fig. 5. Section of the Loop Function*

A table mapping the command bits to their descriptions can be found below:

|  |  |
| --- | --- |
| **Bit** | **Description** |
| 0 (LSB) | Temperature: (0/1 for no measurement/measurement) |
| 1 | Humidity: (0/1 for no measurement/measurement) |
| 2 | Pressure: (0/1 for no measurement/measurement) |
| 3 | Altitude: (0/1 for no measurement/measurement) |
| 4 | Blue LED: (0/1 for OFF/ON) |
| 5-12 (MSB) | Blue LED intensity (Leftmost 8 bits for 0-255) |

*Table 1. Table Showing Command Bits and Their Descriptions*

The last main section of the loop function deals with obtaining the yaw, pitch and roll from the BNO08X chip and printing them to the serial monitor. This can be found in the figure below.

A picture containing text, screenshot, software

Description automatically generated

*Fig. 6. Section of the Loop Function*

The final result was output in both serial terminal and Bluetooth serial terminal that can be found below.

A screenshot of a computer screen

Description automatically generated with medium confidence

*Fig. 7. Output on Both Serial Terminals*

**BlEMain.cpp**

In my main.cpp file, I initialized the multiple libraries that were needed for the sensors to work and provide readings. The figure below shows the libraries I initialized. There are additional libraries other than those form the previous program and those are the BLEDevice, BLEServer, and BLEUtils header files. This file is what is going to enable me to connect the ESP32 to Bluetooth and control it from a Bluetooth serial terminal. The LEDs are defined, the sea level reference pressure is defined and the BNO08X chip is reset. Also, BLE characteristics are created for temperature, altitude, humidity, and pressure to be read from the BME, which has its object created. UUIDs are then created for these characteristics. This can all be seen in the figure below.

A screen shot of a computer program

Description automatically generated with low confidence

*Fig. 8. Figure Showing Initialization of Libraries and Definition of Variables*

The BLEMain.cpp file has multiple custom classes that were created for it to work. These can be found below:

A screen shot of a computer program

Description automatically generated with low confidence

*Fig. 9. Figure Showing Custom Classes and Functions to Interact with BLE Devices*

The setup and loop functions are very similar to that of the previous program except that they now include these characteristics and classes that were created to interact with BLE devices. They can be found in the figures below.

A screen shot of a computer program

Description automatically generated with medium confidence

*Fig. 9. Figure Showing Setup Function*

A screen shot of a computer program

Description automatically generated with low confidence

*Fig. 10. Figure Showing Loop Function*

The final result came as notifications sent to the BLE device as well as values printed on the serial monitor. The output can be found below.

A screenshot of a computer screen

Description automatically generated with low confidence

*Fig. 11. Output in Serial Monitor Showing LED Values and Yaw, Pitch, and Roll*

A picture containing text, screenshot, computer

Description automatically generated

*Fig. 12. Figure Showing Output in BLE Terminal for Temperature*

A screenshot of a computer

Description automatically generated

*Fig. 13. Figure Showing Output in BLE Terminal for Humidity*

A screenshot of a computer

Description automatically generated

*Fig. 14. Figure Showing Output in BLE Terminal for Altitude*

A screenshot of a computer

Description automatically generated

*Fig. 15. Figure Showing Output in BLE Terminal for Pressure*

A screenshot of a computer

Description automatically generated

*Fig. 16. Figure Showing Output in BLE Terminal for LED Values*

**Conclusion**

In this experiment, I learned about the integration of an ESP32 microcontroller with various sensors and Bluetooth connectivity. By initializing libraries and creating objects, I was able to read data from sensors, control LEDs, and communicate via Bluetooth.