Gilian Bensoussan gilianb@campus.technion.ac.il

Bluetooth Low Energy (BLE) Project Summary

1. Introduction to Bluetooth Low Energy (BLE)

Bluetooth Low Energy (BLE) is a wireless communication technology designed for low-power devices. It allows devices to exchange small amounts of data efficiently while consuming minimal power. BLE is widely used in IoT applications, fitness trackers, smart home devices, and more.

2. Difference Between BLE and Classic Bluetooth

Feature	BLE	Classic Bluetooth
Power Consumption	Very Low	High
Data Rate	Lower (125 kbps - 2 Mbps)	Higher (1 Mbps - 3 Mbps)
Range	Longer (up to 100m)	Shorter (typically 10m)
Connection Time	Fast	Slower
Use Case	IoT, sensors, fitness devices	Audio streaming, file transfer

3. Implementing BLE in Flutter/Dart for Android

Step 1: Add Dependencies

Add the flutter_blue_plus package to pubspec.yaml:

```
dependencies:
  flutter:
    sdk: flutter
  flutter_blue_plus: ^1.34.5  # Ensure you use the correct version
```

Step 2: Request Permissions

Modify AndroidManifest.xml to include the necessary permissions:

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Step 3: Scan devices

(see Example)

Step 4: Connect and start Communication

(see Example)

4. How BLE Works with ESP32

Setting Up BLE on ESP32

- Install the ESP32 BLE Arduino library (check if it's already installed before proceeding).
- Write a BLE server sketch in the Arduino IDE. (Examples are available in the BLE library.)

5. Common Mistakes and Problems

Memory Issues:

The ESP32 may run into memory constraints. To mitigate this, change the board to ESP32 Dev Module in the Arduino IDE.

Dependencies:

Ensure all dependencies are up to date and that all required permissions are included in AndroidManifest.xml.

Update:

Note that all that examples and code are working fine with this version of flutter_blue_plus: ^1.34.5. In a future version there might be some changes, but the global idea remains the same.

Delay:

Be sure to add some delays in your Arduino code because BLE is fast but it still takes time to send and receive data (a few milliseconds)

Library:

Be sure to have the correct library installed on Arduino and do not have it installed twice it may cause problems

6. Example

1st example: basic read and write

Connecting the App to ESP32 and Data Exchange

- The Flutter app scans and finds the ESP32 BLE device.
- The app connects to the ESP32.
- The app writes data to the ESP32 to say that it's ready to received data
- The ESP32 write Data to the app

- The app reads data from the ESP32 BLE service.

This setup allows seamless communication between an Android app and an ESP32 using BLE, enabling IoT applications such as smart home automation, health monitoring, and industrial control.

2nd example: multiple characteristics use (temperature sensor)

Using multiple CHARACTERISTIC for separate stream at the same time

One is used to write to the APP numbers starting from 1 and increases by one everytime

The second is used to read from the APP

And the third one is to NOTIFY in real time (example of a temperature sensor)

3rd example: (too big packets)

BLE is Limited to send 20 bytes per packet so we need to cut the packet in multiple small packets

Here we have a very long packet (about 300 bytes) and we cut it into smaller packets (17 bytes per packet) and then the app acknowledge that it received it and the app reconstruct the full packet and display it on the screen.

8. Video Demonstration (60 seconds)

A short Video for each example on the folder Video/Final_video