Esp32BleAndroid Design Tips

Esp32BleAndroid is an adapter interface tool for transfer from ESP32 to Unity via Android mobile phone using BLE (Bluetooth low energy). By connecting the sensors device (gyro, accelerometer, GPS, etc.) with ESP32, Unity can input various data from the sensors device. This design tip explains how to transfer data to Unity by referencing ESP32 and Unity sample code in the purchased asset.

1. How to set up data for transfer to Unity in ESP32
2. How to get data from ESP32 in Unity

1. How to set up data for transfer to Unity in ESP32

ESP32 code ”esp32ble.ino” generates 3D Object tilt data “x, y, z” using the “sin” function and convert it to strings using the “sprintf” function. ESP32 code needs to change according to the input procedure of the sensors when ESP32 connects some sensors device.　This code adds, to confirm to transfer it correctly, the data “12” and “34” to buffer ”formatted_msg” and increases buffer size “i.” The “pCharacteristic->setValue” function sends the data in buffer ”formatted_msg” to Unity.

```
void loop() {
    …
    x = sin(degx[value] / (180 / PI));
    y = sin(degy[value] / (180 / PI));
    dtostrf(x, 4, 2, sx);
    dtostrf(y, 4, 2, sy);

    sprintf(formatted_msg, "%s,%s,%s,", sx, sy, z);
    Serial.println(formatted_msg);
    int i = 0;
    for (i = 0; formatted_msg[i] != '¥0'; ++i);
    formatted_msg[i] = 12;
    formatted_msg[i+1] = 34;
```

```
    i+= 2;

    …
    pCharacteristic->setValue((uint8_t*)formatted_msg, i);
    pCharacteristic->notify();
    …
```

2. How to get data from ESP32 in Unity

Unity script ” Esp32BleAndroidSampleCode.cs” receives BLE data from ESP32 as string data “str” of return value of “obj.Call<string>("UpdateRead")” function. “str.Split(',').Select(byte.Parse).ToArray()” function changes the string data “str” into array byte data “data”. Array byte data “data” encodes string code using “System.Text.Encoding.UTF8.GetString” function and, using “text.Split” function and “float.Parse” function, the string code is converted to float position data.

```
        …
// Update is called once per frame
void Update()
{
    float[] acceldata = new float[3];
    byte[] data;

    string str = obj.Call<string>("UpdateRead");
    UnityEngine.Debug.LogWarning(TAG + " Update: " + str);

    if (str == null)
    {
        return;
    }

    updatetext.text = str;
```

```
    try
    {
        data = str.Split(',').Select(byte.Parse).ToArray();
        string text = System.Text.Encoding.UTF8.GetString(data);

        string[] arr = text.Split(',');
        acceldata[0] = float.Parse(arr[0]);
        acceldata[1] = float.Parse(arr[1]);
        acceldata[2] = float.Parse(arr[2]);
        UnityEngine.Debug.LogWarning(TAG + " x: " + acceldata[0] + " y: " +
acceldata[1] + " z: " + acceldata[2]);

        accelx = acceldata[0] * 100 + (-90);
        accely = acceldata[1] * 100;
        accelz = acceldata[2] * 100;

        transform.rotation = Quaternion.AngleAxis(accelx, Vector3.up) *
Quaternion.AngleAxis(accely, Vector3.right);
    }
    catch (Exception e)
    {
        exceptiontext.text = e.ToString();
        UnityEngine.Debug.LogWarning(TAG + e.ToString());
    }
    finally
    {}
}
    ...
```