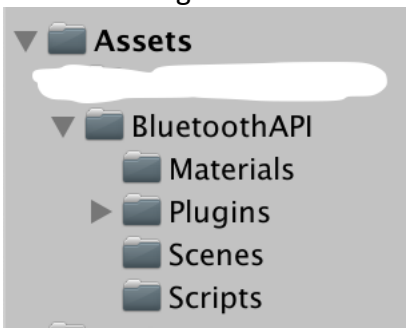
Arduino Unity Plugin

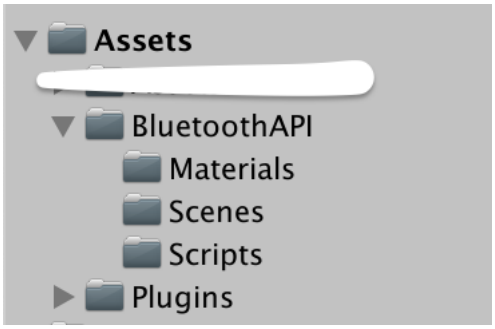
Requirements

1. Switch Scripting runtime version to 4.x as 3.5 is already deprecated (found in Build Settings)

2. Move the Plugin Folder to the main Assets Folder



Before



After

Supported Devices

1. Android
2. Windows PC

BluetoothHelper Class

Static Vars and Methods

1. GetInstance(string *deviceName*)
    - deviceName: string identifying the Bluetooth module you are going to connect to
    - Returns BluetoothHelper Instance
    - Throws:
        i. BlueToothNotEnabledException: Bluetooth not turned on
        ii. BlueToothNotSupportedException: Bluetooth not supported
        iii. BlueToothNotReadyException: the Bluetooth device is not paired
        iv. BlueToothPermissionNotGrantedException: this is caused by not moving the plugin folder to the main assets folder
2. Bool SERIAL_COMM:
    - Default: False => connect to destination device using Bluetooth
    - True => connect to destination device using USB Cable. In this case, *deviceName* variable refers to the COM port name (example: *COM5*)
    - Serial communication is ONLY available on windows PC, and setting it to True for android devices has no effect.

3. Bool BLUETOOTH_SIMULATION
    - Default: False => Connect to actual Bluetooth device
    - True => Emulate Bluetooth connected by providing a GUI interface to simulate receiving messages
    - This variable ONLY has effect on Windows PC so you can simulate connecting to Bluetooth device if your laptop doesn't have Bluetooth, so you can always develop
    - On not supported platforms, like iOS, MacOS… this is the default mode

Properties and Methods

1. isDeviceFound():
    - return true if the device is already paired
    - return false if the device is not paired
2. SendData(string *data*):
    - Send string data to the Bluetooth devices
3. SendData (byte[] data):
    - Send byte array data to the Bluetooth devices
4. Connect()
    - Connect to Bluetooth device
    - Invokes 2 events:
        i. OnConnected: when successfully connected to the device
        ii. OnConnectionFailed: when failed to connect to the Bluetooth device
5. setLengthBasedStream()
    - sets reading and writing mode of the stream based on its length.
      Example: Sending {0x02, 0x04, 0x65, 0xE5} from unity will result in sending: {0x55, 0x55, 0x00, 0x04, 0x02, 0x04, 0x65, 0xE5} knowing that 0x00 and 0x04 are the array length encoded on 2 bytes and 0x55, 0x55 are the preamble, to detect the start of the message. You don't have to worry about the encoding procedure or adding the preamble, as it is done automatically by the plugin.
      From the Arduino, to get the message follow this code:

```
void readBT()
{
  if(Serial.available() >= 2)
  {
    data_length = 0;
    //reading the preambles
    byte pre1 = Serial.read();
    byte pre2 = Serial.read();
    if(pre1 != 85 || pre2 != 85) return;
    while(Serial.available() < 2) continue;
    byte x1 = Serial.read();
    byte x2 = Serial.read();

    data_length = x1 << 8 | x2;

    data = new byte[data_length];
    i=0;
    while(i<data_length)
    {

      if(Serial.available()==0){
        continue;
      }
      timeout=0;
      data[i++] = Serial.read();
    }

    // process the data …

    delete[] data;
  }
}
```

Now sending messages from the Arduino,
{0x02, 0x04, 0x65, 0xE5} will be sent as:
{0x55, 0x55, 0x00, 0x04, 0x02, 0x04, 0x65, 0xE5}
use this function to send from the arduino:

```
void sendBT(const byte *data, int length)
{
       byte len[4];
       //YOU HAVE TO PUT THE PREAMPLE WHEN SENDING FROM THE ARDUINO
       len[0] = 85; //preamble
       len[1] = 85; //preamble
       len[2] = (length >> 8) & 0x000000FF;
       len[3] = (length & 0x000000FF);
       Serial.write(len, 4);
       Serial.flush();
       Serial.write(data, l);
       Serial.flush();
}
```

6. setTerminatorBasedStream(string str)
    - set the writing and reading mode based on a terminator string to delimit the messages. Example, using \n (new line) to delimit incoming messages. "Hello\nHow are you" will be considered as 2 incoming messages in this case
7. StartListening()
    - Start listening for incoming messages
    - Invokes 1 event:
      OnDataReceived: called when a message is received
    - Throws:
        i. BluetoothListeningMethodIsNotSetException: when neither of *setTerminatorBasedStream* or *setLengthBasedStream* has been called
8. StopListening()
    - Stops listening for incoming messages and disconnects from the Bluetooth device.
    - This method must be called in the OnDestroy() method in a MonoBehaviour class:

```
void OnDestroy()
{
        bluetoothHelperInstance.StopListening();
}
```

9. isConnected()
    - returns True if we are connected to the bluetooth device
10. Bool Available
    - returns True if we have incoming messages waiting to be read
11. Read()
    - Return a string representation of the incoming messages when available
    - In case of want binary data representation from the string, call

```
char[] data = bluetoothHelper.Read().ToCharArray ();
```

Events to Listen to

1. OnConnected
2. OnConnectionFailed
3. OnDataReceived

These events are already explained above,
To Listen to them, use this syntax:

```
//this could be written is the Start() function for example
bluetoothHelperInstance.OnConnected += OnConnectedFunction;


void OnConnectedFunction()
{
	//Yes, we are now connected, maybe we should start listening for incoming messages ☺
	bluetoothHelperInstance.StartListening();
}
```

Or this lambda expression syntax

```
bluetoothHelperInstance.OnConnected += () => {
	bluetoothHelperInstance.StartListening();
};
```

Thank you for using this plugin
You can always contact me via email abouzaidan.tony@gmail.com is you have any question.
This plugin will always be Here!