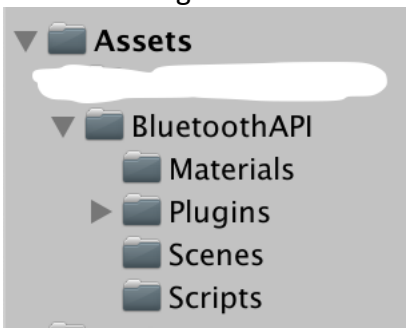
Arduino Unity Plugin

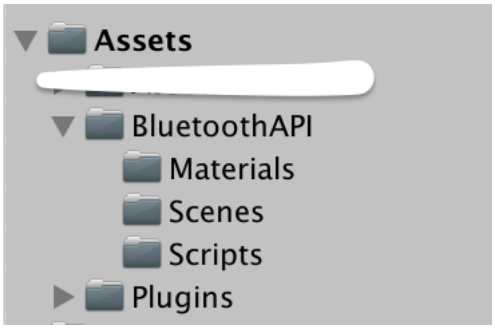
Requirements

1. Switch Scripting runtime version to 4.x as 3.5 is already deprecated (found in Build Settings). No need to change on unity 2019.X and later.

2. Move the Plugin Folder to the main Assets Folder

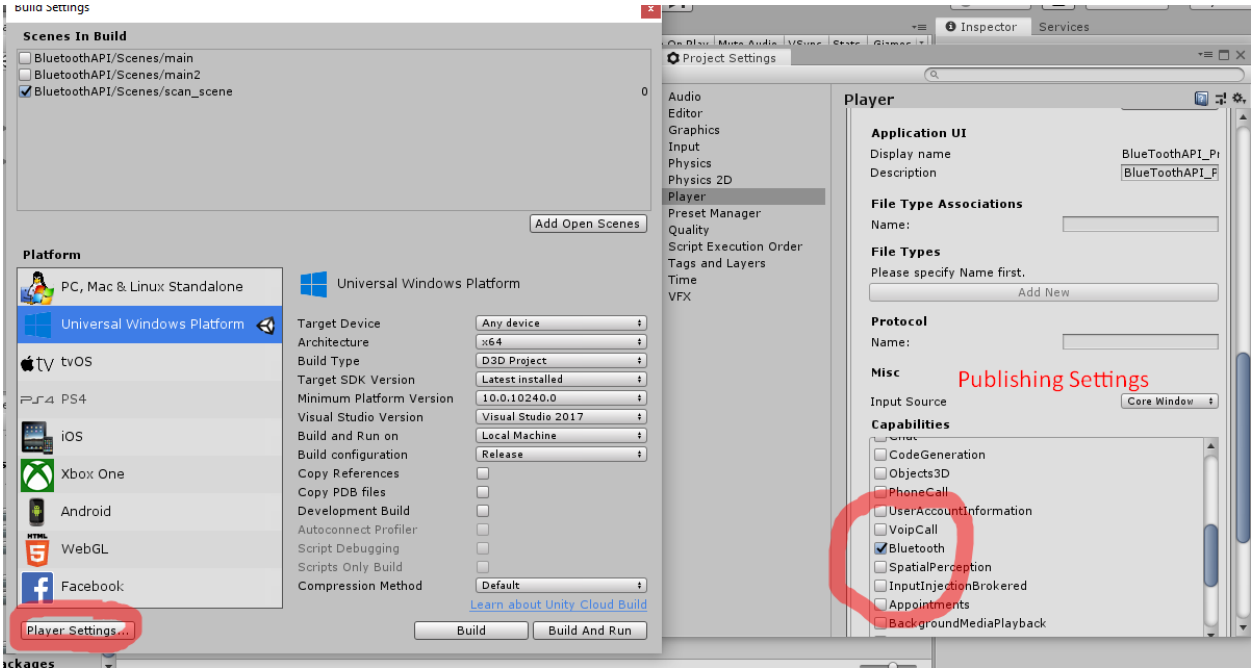


Before

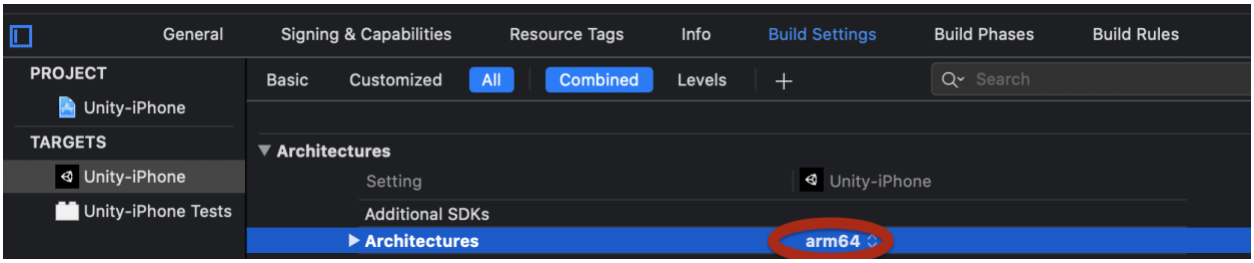


After

For UWP



3. For iOS



And add “NSBluetoothAlwaysUsageDescription” in plist file.

Supported Devices

1. Android
2. iOS
3. MacOS
4. Windows PC
5. UWP

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. GetInstance()
    - Returns the BluetoothHelper Instance
    - deviceName will be set later when attempting to connect
    - This is helpful when there’s a need to scan for nearby Bluetooth and connect to one of them based on the scan result
    - Throws same exceptions as (1)
3. 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.
4. 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
- SET THIS VARIABLE TO *TRUE* IF YOUR COMPUTER DOESN'T SUPPORT BLUETOOTH

5. Bool BLE
    - Default: False => use Bluetooth Classic technology
    - True => use Bluetooth Low Energy technology
    - *Changing this value has no effect on windows version, since BLE is not currently supported on Windows PC by this plugin. Only Android, MacOS and UWP. Regarding IOS, only BLE is supported.*
      BLE is supported on UWP, and Bluetooth Classic on both Windows Desktop App and UWP
6. Bool ASYNC_EVENTS
    - Default : False => events raise by the plugin are synchronized with the main UI thread.
    - True : the events are raised asynchronously independently from the UI thread. If set to true, you can't update the UI when any plugin event is raised.

Properties and Methods

1. isDevicePaired():
    - Bluetooth Classic
        i. return true if the device is already paired
        ii. return false if the device is not paired
    - BLE:
        i. Returns true if, after scanning nearby devices, the Bluetooth device is found
        ii. Returns false if, after scanning for nearby devices, the Bluetooth device is not found
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. setDeviceName(string *deviceName*)
6. setDeviceAddress(string *deviceAddress*)
    - These 2 functions set the properties of the device you wish to connect to.

One will override the other. So, you either connect by name or by mac address. setDeviceAddress function will trigger setDeviceName for IOS or MacOS BLE since connecting to a device by its mac address on IOS since it's not supported by Apple.
These 2 functions are useful if you are trying to connect to multiple devices, so you set the name of a device,connect, transmit data, call disconnect, set the name of another device connect and so on…

7. ScanNearbyDevices()
    - Scan for nearby Bluetooth devices
    - Return true if scan has started
    - Calling this function is a must when using BLE technology before connecting to a BLE device.
    - Invokes 1 event:
        i. OnScanEnded: returns a list of devices found
    - Not available for Windows PC (Desktop App)
8. 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();
}
```

9. 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. This mode is not recommended for binary data transmission unless its delimited by the null character (\0). (for the moment)

10. setFixedLengthBasedStream(int length)
    - set the reading mode based on the number of received bytes. Each "length" bytes will be considered 1 message. To get the data, use ReadBytes()
11. 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* or *setFixedLengthBasedStream* has been called
    - No need for calling this method when subscribing to custom characteristics in BLE.
12. Disconnect()
    - 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.Disconnect();
}
```

13. isConnected()
    - returns True if we are connected to the bluetooth device

14. Bool Available
    - returns True if we have incoming messages waiting to be read
15. Read()
    - Return a string representation of the incoming messages when available
    - In case of want binary data representation from the data, use ReadBytes()
16. ReadBytes()
    - Recommended when reading binary data.
17. EnableBluetooth(bool enable)
    - Used to enable/disable Bluetooth using the plugin. ONLY available on Android.
18. IsBluetoothEnabled()
    - Checks if the Bluetooth is enabled or not. Available on Android, MacOS and iOS.
    - For MacOS and iOS, it's not recommended to do the check in the Start function as it will always return false. A good practice will be in the Update function.
19. Subscribe(BluetoothHelperService service)
    - To subscribe to a characteristic of a service for "Notify on Change", create a BluetoothHelperService instance (constructor takes service UUID) and call addCharacteristic function (taking BluetoothHelperCharacteristic as a parameter).
    - Android UUID format is :
        i. "ABCD" (4 chars)
        ii. "123e4567-e89b-12d3-a456-426655440000"
    - MacOS and iOS format:
        i. "ABCD" (4 chars)
20. WriteCharacteristic(BluetoothHelperCharacteristic characteristic, byte[] value)
21. WriteCharacteristic(BluetoothHelperCharacteristic characteristic, string value)
    - There two functions are used to write a value to a custom characteristic on which the service name must be set by calling characteristic.setService(string)
    - Throws:
        i. ServiceNotSetException : service name of the characteristic is null.
        ii. ServiceNotFound
        iii. CharacteristicNotFound
22. ReadCharacteristic(BluetoothHelperCharacteristic characteristic)
    - Read the value of a characteristic asynchronously, result received in event : OnCharacteristicChanged
    - Throws:
        i. ServiceNotSetException : service name of the characteristic is null.
        ii. ServiceNotFound
        iii. CharacteristicNotFound

Events to Listen to

1. OnConnected
2. OnConnectionFailed
3. OnDataReceived

4. OnScanEnded
5. OnServiceNotFound
6. OnCharacteristicNotFound
7. OnDescriptorNotFound
8. OnCharacteristicChanged

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();
};

bluetoothHelper.OnScanEnded += (nearbyDevices) =>
{
	//nearbyDevices is a LinkedList containing nearby devices
};
bluetoothHelper.OnCharacteristicChanged += (value, characteristic) =>
{
	//value is a byte array
	//characteristic is a BluetoothHelperCharacteristic instance
};
```

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!