***Custom Bluetooth Plugin***

After hours of trying to find and use a Bluetooth plugin for unity, I developed a Bluetooth:Unity plugin from scratch.

*Installation*

This plugin is called “VertiGoneBluetoothPlugin.” To install this plugin into your Unity Application, move the “VertiGoneBluetoothPlugin.jar” and “AndroidManifest.xml” files to the “/Assets/Plugins/Android/” folder in your Unity project. If these folders do not exist, create them. Then move the “BleWrapper.cs” script to the “/Assets/Plugins/” folder of your Unity project.

*Function*

The function of the plugin is to connect to the nRF8001 bluetooth device (with the name “VGONE”) and read accelerometer/gyroscope values.” The Bluetooth device must have the name “VGONE” to be registered by the plugin. Once connected, the sensor values within the plugin will be updated from values sent from the Bluetooth device. The Bluetooth device can send strings of the form:

“100,150,300” for gyroscope data

“A111,222,333” for accelerometer data

using the Arduino code shown below.

The plugin right now has no capacity to rescan for lost devices. If the connection is lost, you must restart the application to reconnect. Upon restart, the plugin should immediately connect to the device. The plugin will continuously read *until the application is closed.* To close the connect, you must close the application.

*Examples*

I provided an example Unity Project that utilizes the plugin to receive data from the Bluetooth and update the user on the current Bluetooth connection status. Please take a look at it.

*Uploading Unity Project to Android*

The application requires Android 19 and higher, so please be aware of that when uploading the project to your Android device.

*Troubleshooting*

If the device is not connecting, check your connections. Use a third party app (like the UART app) to verify that a connection can be made. Use arduino’s serial monitoring to verify connections. You can also use eclipses logcat with the phone pluged in and the tags “Plugin” or “BleStatus” to debug.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

VertiGone Arduino Code for sending to Unity

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// This version uses the internal data queing so you can treat it like Serial (kinda)!

#include <SPI.h>

#include "Adafruit\_BLE\_UART.h"

// Connect CLK/MISO/MOSI to hardware SPI

// e.g. On UNO & compatible: CLK = 13, MISO = 12, MOSI = 11

#define ADAFRUITBLE\_REQ 10

#define ADAFRUITBLE\_RDY 2 // This should be an interrupt pin, on Uno thats #2 or #3

#define ADAFRUITBLE\_RST 9

Adafruit\_BLE\_UART BTLEserial = Adafruit\_BLE\_UART(ADAFRUITBLE\_REQ, ADAFRUITBLE\_RDY, ADAFRUITBLE\_RST);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

Configure the Arduino and start advertising with the radio

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setup(void)

{

Serial.begin(9600);

while(!Serial); // Leonardo/Micro should wait for serial init

Serial.println(F("Adafruit Bluefruit Low Energy nRF8001 Print echo demo"));

BTLEserial.setDeviceName("VGONE"); //THIS MUST BE "VGONE" TO BE PICKED UP! \* 7 characters max! \*/

BTLEserial.begin();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

Constantly checks for new events on the nRF8001

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

aci\_evt\_opcode\_t laststatus = ACI\_EVT\_DISCONNECTED;

long getDecimal(float val)

{

int intPart = int(val);

long decPart = 1000\*(val-intPart); //I am multiplying by 1000 assuming that the foat values will have a maximum of 3 decimal places

//Change to match the number of decimal places you need

if(decPart>0)return(decPart); //return the decimal part of float number if it is available

else if(decPart<0) return((-1)\*decPart); //if negative, multiply by -1

else if(decPart=0) return(00); //return 0 if decimal part of float number is not available

}

String floatToString(float f) {

String stringVal = "";

stringVal += String(int(f))+ "." + String(getDecimal(f));

return stringVal;

}

void sendStringToBle(String s) {

uint8\_t sendbuffer[20];

s.getBytes(sendbuffer, 20);

char sendbuffersize = min(20, s.length());

Serial.print(F("\n\* Sending -> \"")); Serial.print((char \*)sendbuffer); Serial.println("\"");

// write the data

BTLEserial.write(sendbuffer, sendbuffersize);

}

void loop()

{

// Tell the nRF8001 to do whatever it should be working on.

BTLEserial.pollACI();

//Change this code to your gyroscope/accelerometer data, 20 bytes at a time

//Example: Send accelerometer data by PREFIXING string with "A"

sendStringToBle("A100,1503,3004");

//Example: Send gyroscope data by sending string with NO PREFIX

sendStringToBle("A160,15603,30604");

}