University of Ottawa | Université d'Ottawa

GNG2101 | Fall 2022

uOttawa

GNG2101

Introduction to Product Development and Management for Engineers and Computer Scientist

Course Professor: Dr. Emmanuel Bouendeu

Deliverable F

Prototype 2

Presented to: Melika Ataebi

Prepared by Group B12

Team Members:

Family Name, Name:	Student ID:
Keryakes, Laura	300265134
Balachandiran, Vivethen	300245080
Ouloum, Yassine	300263213
Ma, Jiayi	300263220
Vadakkeveettilan Hilariyos, Chelse Rose	300214163

Date: November 6, 2022

Introduction

In this deliverable, the prototype created in the last deliverable will be evaluated and improved based on the client & peer feedback given. Not only will it include a revised version of the previous prototype, but it will also include a revised testing plan which will be demonstrated through the most critical product assumption up to this point. Our main goal for this prototype is to ensure some sort of physical functionality using the same Arduino components that will be used in the final product. Thus, this prototype will resemble the most to our final product with some limited functionality. More specifically, we will be utilizing the Arduino Nano BLE (which is the Bluetooth module with sensor integrated) and the 6-Axis gyroscope accelerometer sensors to output coding through the Arduino software. We will also start the 3D modeling of an Arduino case to ensure durability of our product when it will be adhered on the racquet. In other words, we will be creating a high fidelity physical and comprehensive prototype. For testing, we will be testing the accuracy and credibility of the coding by analyzing the output values depending on the various movements we will be making with the sensors. The testing will be of medium fidelity since we will not be installing the sensors on a racquet but rather we will be moving the sensors freely, only mimicking various movements that can happen during a tennis performance.

Client Feedback

Our client liked our prototype 1 and 2, but did have some suggestions to improve our design. A suggestion was to make the breadboard of the current prototype we have smaller by connecting the battery to the controller. This will allow for a smaller and lighter device that can be easily mounted onto the racket. Another suggestion was to implement a switch that turns on and off the tracking or set a threshold in the arduino code which only detects acceptable values. This is so that if the player is swinging their racket around without the purpose of practicing tennis techniques, there will not be unnecessary information stored on the device. This can furthermore make our code for the Arduino and tracking more efficient. Furthermore, a case was also suggested for the arduino so that its separate pieces can be kept together and can be mounted onto the racket allowing for a durable product.Finally, our team needs to find a method to send information from a platform to the website associated with the product. This is so that users can see their information sent from the devices on a website that has more user-friendly data.

Critical Product Assumption

Our prototype is using IMU sensors, and calculations involving acceleration and angular acceleration in which we assumed that the measurement frequence was high enough to consider those measurements constant during that time interval. We also assumed that the motion was relatively short to not take into account possible drift and other errors coming from the gyroscope and accelerometer.

Having only two measurement devices, placing them on different positions on the racket should be accurate enough for our use (given the size of the racket compared to the range of motion).

Prototype 2

Our second prototype consists of 1 gyroscope+accelerometer sensor and 1 gyroscope+accelerometer built in the arduino nano BLE 33 that measure acceleration and angular acceleration in the 3 axes.

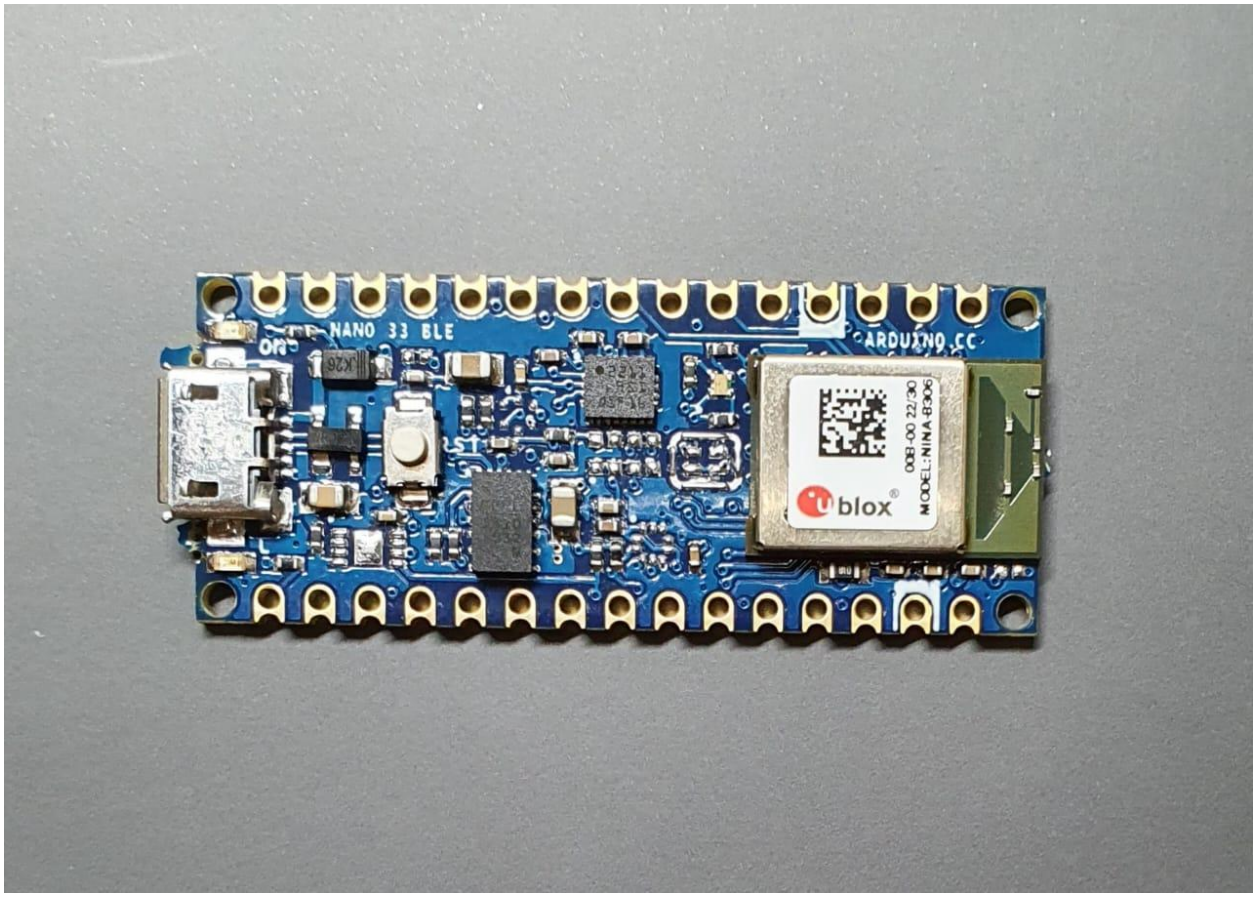


Figure 1*: Arduino Nano BLE 33*

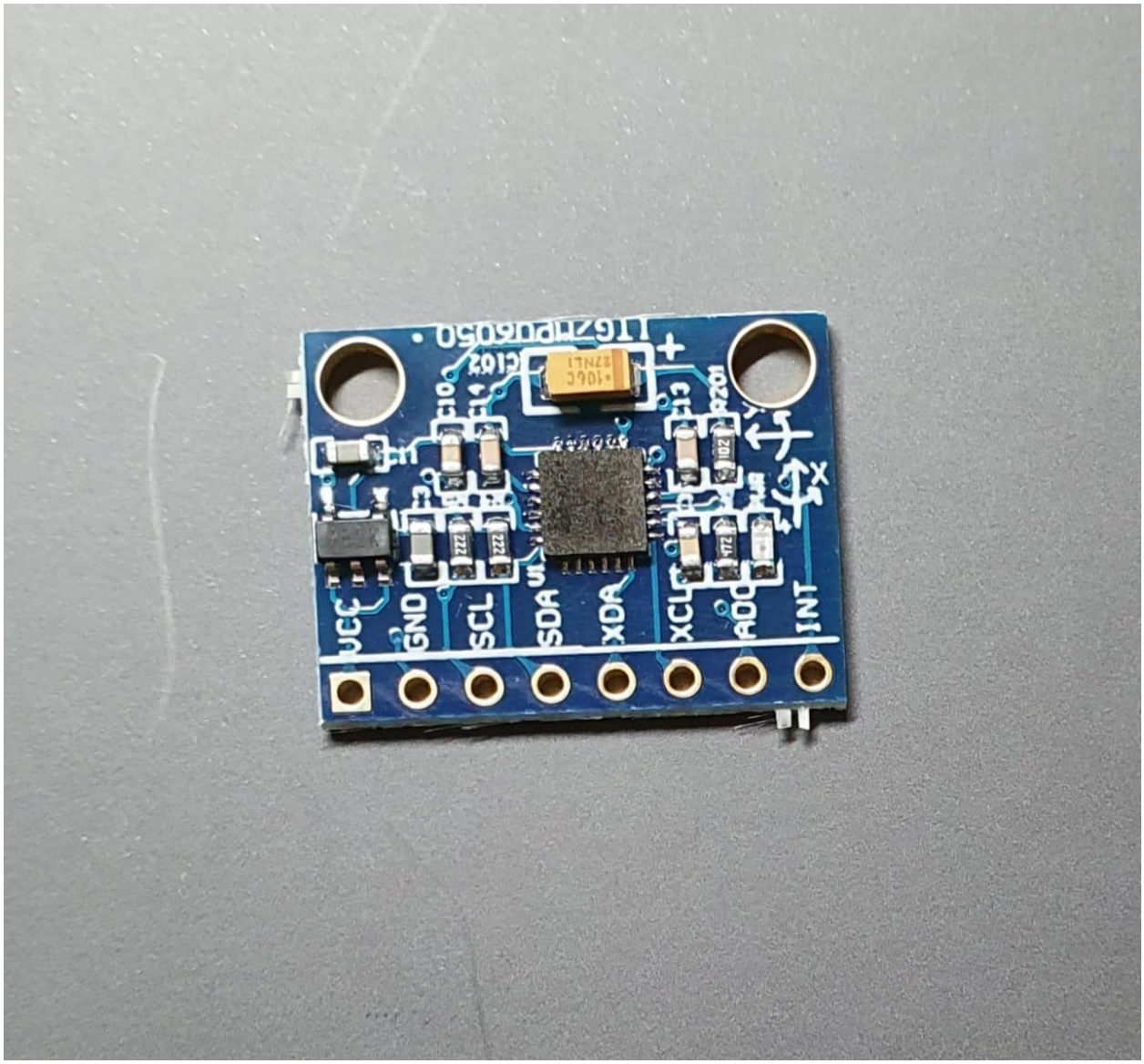


***Figure 2**: IMU-6050*

The sensor is wired to the arduino which is related to a battery that powers up the device.

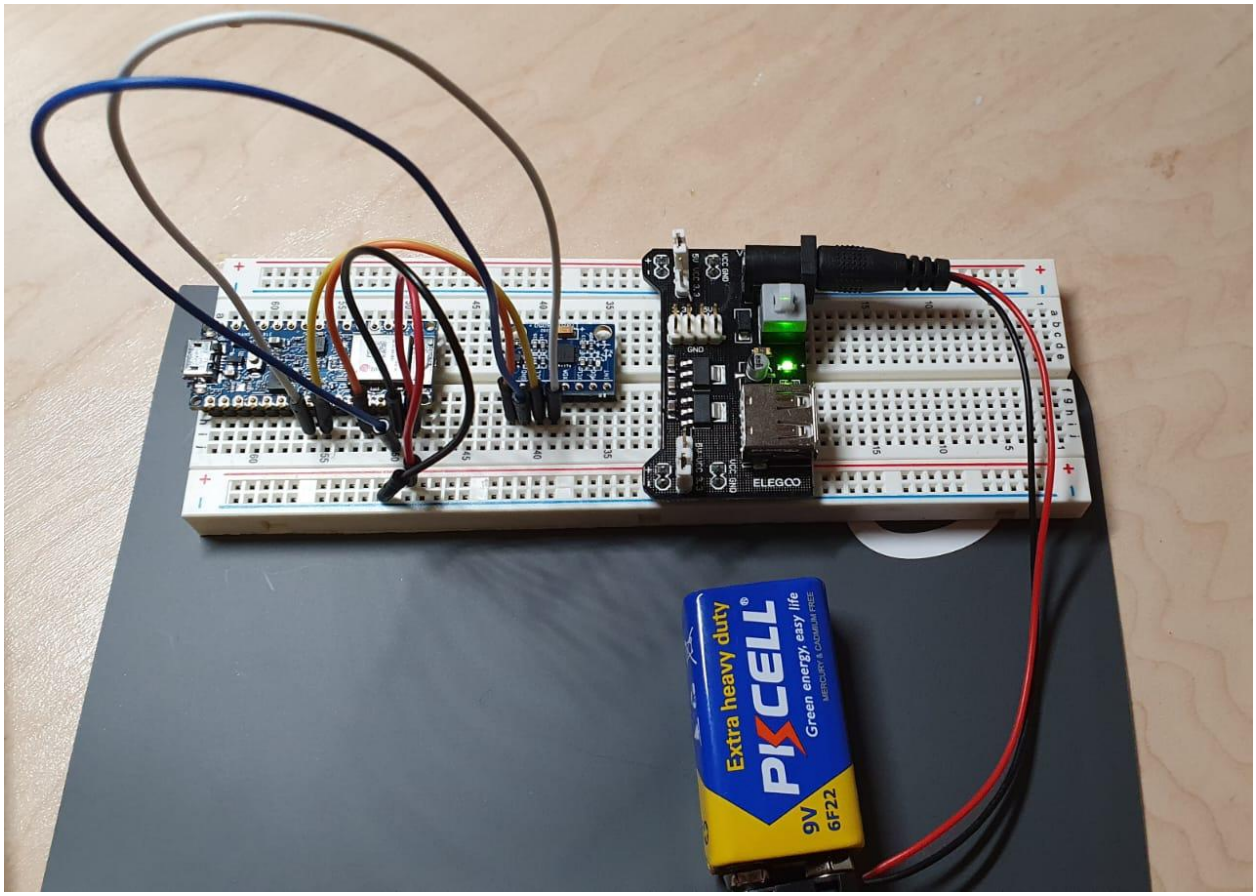


***Figure 3**: Assembly of the components*

The device functions based on instructions embed in the arduino, the program is as follows:

```
#include <Arduino_LSM9DS1.h>
#include <ArduinoBLE.h>
#include <Arduino_LSM9DS1.h>


//------------------------------------------------------------------------
-----------------------------------------------
// BLE UUIDs
//------------------------------------------------------------------------
-----------------------------------------------

#define BLE_UUID_TEST_SERVICE
"9A48ECBA-2E92-082F-C079-9E75AAE428B1"
#define BLE_UUID_ACCELERATION                "2713"
```

```
#define BLE_UUID_COUNTER
"1A3AC130-31EE-758A-BC50-54A61958EF81"
#define BLE_UUID_RESET_COUNTER
"FE4E19FF-B132-0099-5E94-3FFB2CF07940"

//-------------------------------------------------------------------------
-----------------------------------------
// BLE
//-------------------------------------------------------------------------
-----------------------------------------

BLEService testService( BLE_UUID_TEST_SERVICE );
BLEFloatCharacteristic accelerationCharacteristicX( BLE_UUID_ACCELERATION,
BLERead | BLENotify );
BLEFloatCharacteristic accelerationCharacteristicY( BLE_UUID_ACCELERATION,
BLERead | BLENotify );
BLEFloatCharacteristic accelerationCharacteristicZ( BLE_UUID_ACCELERATION,
BLERead | BLENotify );BLEFloatCharacteristic accelerationCharacteristicX2(
BLE_UUID_ACCELERATION, BLERead | BLENotify );
BLEFloatCharacteristic accelerationCharacteristicY2(
BLE_UUID_ACCELERATION, BLERead | BLENotify );
BLEFloatCharacteristic accelerationCharacteristicZ2(
BLE_UUID_ACCELERATION, BLERead | BLENotify );
BLEUnsignedLongCharacteristic counterCharacteristic( BLE_UUID_COUNTER,
BLERead | BLENotify );
BLEBoolCharacteristic resetCounterCharacteristic( BLE_UUID_RESET_COUNTER,
BLEWriteWithoutResponse );




void setup()
{
  Serial.begin( 9600 );
  //  while ( !Serial );


  if ( !IMU.begin() )
  {
    Serial.println( "Failed to initialize IMU!" );
```

```
    while ( 1 );
  }
  Serial.print( "Accelerometer sample rate = " );
  Serial.print( IMU.accelerationSampleRate() );
  Serial.println( " Hz" );

  if( setupBleMode() )
  {
    Serial.println( “working” );
  }
  if (!IMU.begin()){
    Serial.println("Failed to initialize MPU6050");
    while(1){
      delay(10);
    }

} // setup


void loop()
{
  static unsigned long counter = 0;
  static long previousMillis = 0;

  // listen for BLE peripherals to connect:
  BLEDevice central = BLE.central();

  if ( central )
  {
    Serial.print( "Connected to central: " );
    Serial.println( central.address() );

    while ( central.connected() )
    {
      if( resetCounterCharacteristic.written() )
      {
        counter = 0;
      }

      long interval = 2000;
```

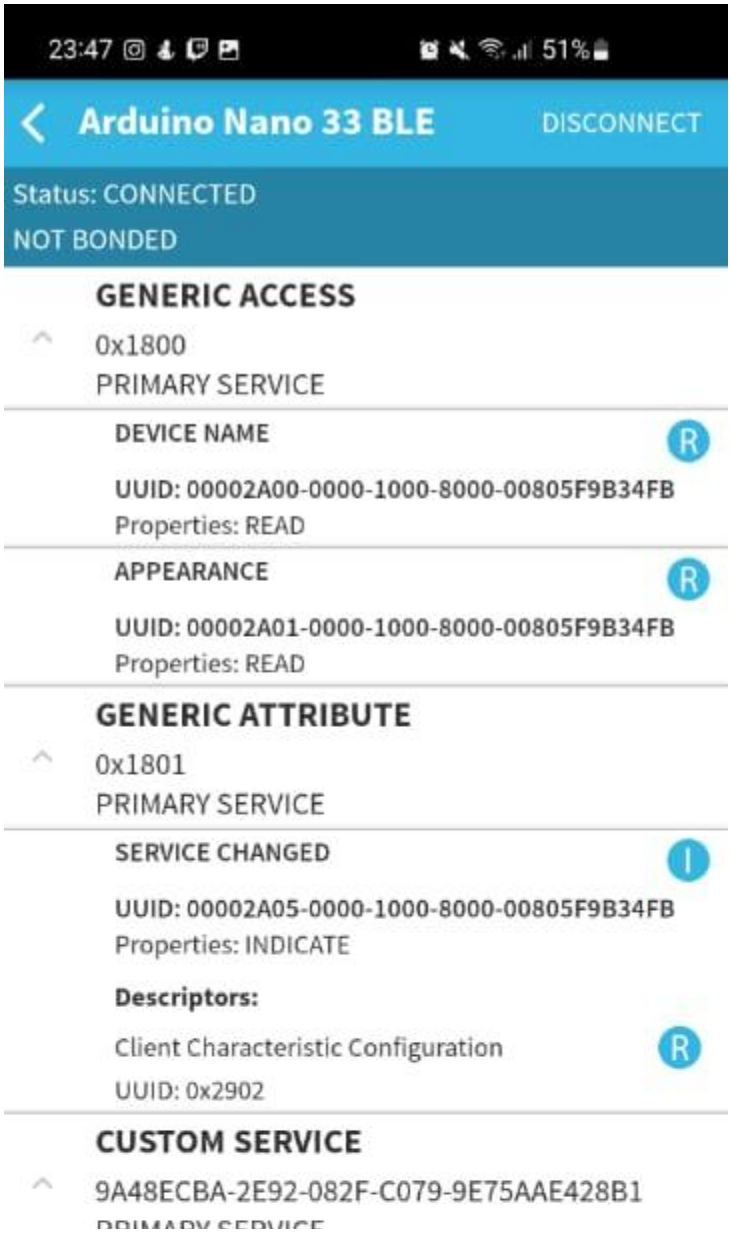
```
      unsigned long currentMillis = millis();
      if( currentMillis - previousMillis > interval )
      {
        previousMillis = currentMillis;

        Serial.print( "Central RSSI: " );
        Serial.println( central.rssi() );

        if( central.rssi() != 0 )
        {
          float accelerationX, accelerationY, accelerationZ,
RotationX,RotationY,RotationZ;
          if ( IMU.accelerationAvailable() )
          {
            IMU.readAcceleration( accelerationX, accelerationY,
accelerationZ );
            accelerationCharacteristicX.writeValue( accelerationX );
            accelerationCharacteristicY.writeValue( accelerationY );
            accelerationCharacteristicZ.writeValue( accelerationZ );
          if (IMU.gyroscopeAvailable()) {
            IMU.readGyroscope(RotationX, RotationY, RotationZ);
            gyroscopeCharacteristicX.writeValue( RotationX);
            gyroscopeCharacteristicY.writeValue( RotationY);
            gyroscopeCharacteristicZ.writeValue( RotationZ);

          }

          counter++;
          counterCharacteristic.writeValue( counter );
        }

      } // interval
    } // while connected

    Serial.print( F( "Disconnected from central: " ) );
    Serial.println( central.address() );
  } // if central
} // loop
```

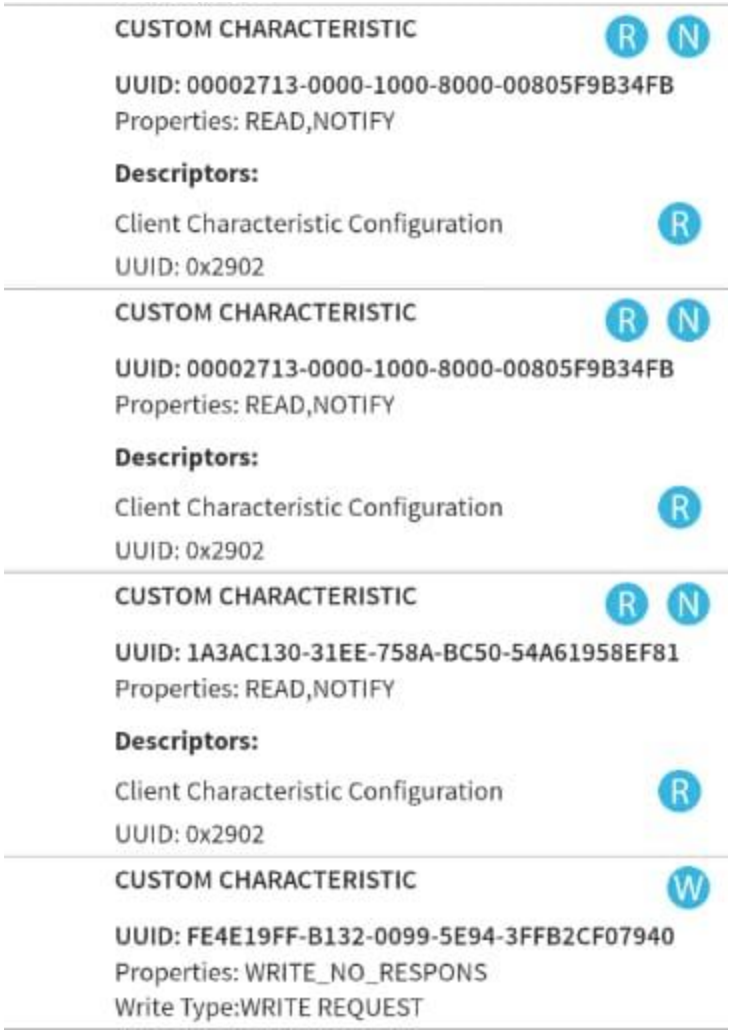
```
bool setupBleMode()
{
  if ( !BLE.begin() )
  {
    return false;
  }

  // set advertised local name and service UUID:
  BLE.setDeviceName( "Arduino Nano 33 BLE" );
  BLE.setLocalName( "Arduino Nano 33 BLE" );
  BLE.setAdvertisedService( testService );

  // BLE add characteristics
  testService.addCharacteristic( accelerationCharacteristicX );
  testService.addCharacteristic( accelerationCharacteristicY );
  testService.addCharacteristic( accelerationCharacteristicZ );
  testService.addCharacteristic( gyroscopeCharacteristicX );
  testService.addCharacteristic( gyroscopeCharacteristicY );
  testService.addCharacteristic( gyroscopeCharacteristicZ );

  testService.addCharacteristic( counterCharacteristic );
  testService.addCharacteristic( resetCounterCharacteristic );

  // add service
  BLE.addService( testService );

  // set the initial value for the characeristic:
  accelerationCharacteristic.writeValue( 0.0);
  accelerationCharacteristicY.writeValue( 0.0);
  accelerationCharacteristicZ.writeValue( 0.0);
  gyroscopeCharacteristicX.writeValue(0.0 );
  gyroscopeCharacteristicY.writeValue( 0.0);
  gyroscopeCharacteristicZ.writeValue( 0.0);
  counterCharacteristic.writeValue( 0 );

  // start advertising
  BLE.advertise();

  return true;
```

```
}
```

Testing

Using the phone application "**BLE Scanner**" we can see that our data is properly transferred to any device that supports BLE:





Now all that is left is to import that data and transfer it to the program that is going to sort and treat it.

Conclusion

In this deliverable, we refined the functionality of our prototype. For the final prototype, we are aiming to test this current prototype during an actual performance to see how long the product can accurately collect data. The 3D-printed case will be installed and the website will be created to see how effective the output of the data is towards the users. Thus, the final, comprehensive prototype, will in other words be the final product, but with minor modifications.

Project Plan Update

Snapshot:

https://www.wrike.com/frontend/ganttchart/index.html?snapshotId=fi7FnjYbZXCAY0tqHmogRBQ3OtkSZ0Nm%7CIE2DSNZVHA2DELSTGIYA