```
#include<ArduinoBLE.h>
#include<LowPower.h>
#include<Arduino_LSM6DS3.h>

//Add BLE service (essentially a list of characteristics)
BLEServicegaugeService("1101");

//Add characteristics to the BLE service
//A characteristic can hold one value (integers and doubles are used
here)
BLEIntCharacteristic sensor1char("2101", BLERead | BLENotify); //SG1
BLEIntCharacteristic sensor2char("2102", BLERead | BLENotify); //SG2
BLEIntCharacteristic sensor3char("2103", BLERead | BLENotify); //SG3
BLEIntCharacteristic sensor4char("2104", BLERead | BLENotify); //SG4
BLEIntCharacteristic sensor5char("2105", BLERead | BLENotify); //SG5
BLEIntCharacteristic sensor6char("2106", BLERead | BLENotify); //SG6
BLEIntCharacteristic sensor7char("2107", BLERead | BLENotify); //SG7
BLEIntCharacteristic sensor8char("2108", BLERead | BLENotify); //SG8
BLEIntCharacteristic timeMillis("2109", BLERead | BLENotify);
//timestamp
BLEIntCharacteristic accelX("2110", BLERead | BLENotify);
//accelerometer x axis
BLEIntCharacteristic accelY("2111", BLERead | BLENotify);
//accelerometer y axis
BLEIntCharacteristic accelZ("2112", BLERead | BLENotify);
//accelerometer z axis

void setup()
{
    // put your setup code here, to run once:
    pinMode(2, OUTPUT);    // MUX Ad0
    pinMode(3, OUTPUT);    // MUX Enable
    pinMode(4, OUTPUT);    // pot3 CS
    pinMode(5, OUTPUT);    // pot3 UD
    pinMode(6, OUTPUT);    // pot2 CS
    pinMode(7, OUTPUT);    // pot2 UD
    pinMode(8, OUTPUT);    // pot1 CS
    pinMode(9, OUTPUT);    // pot1 UD
    pinMode(20, OUTPUT);   // MUX Ad1
```

```
  pinMode(21, OUTPUT);   // MUX Ad2

  pinMode(A2, INPUT);    // Ain+
  pinMode(A3, INPUT);    // Ain- (tied to AGND)

  IMU.begin();

 Serial.begin(9600);
  //while (!Serial);

 pinMode(LED_BUILTIN, OUTPUT);
  if (!BLE.begin())
   {
      Serial.println("starting BLE failed!");
       while (1);
   }
BLE.setLocalName("StrainGaugeMonitor");

BLE.setAdvertisedService(gaugeService);
gaugeService.addCharacteristic(sensor1char);
gaugeService.addCharacteristic(sensor2char);
gaugeService.addCharacteristic(sensor3char);
gaugeService.addCharacteristic(sensor4char);
gaugeService.addCharacteristic(sensor5char);
gaugeService.addCharacteristic(sensor6char);
gaugeService.addCharacteristic(sensor7char);
gaugeService.addCharacteristic(sensor8char);
gaugeService.addCharacteristic(timeMillis);
gaugeService.addCharacteristic(accelX);
gaugeService.addCharacteristic(accelY);
gaugeService.addCharacteristic(accelZ);



 BLE.addService(gaugeService);

  BLE.advertise();
  Serial.println("Bluetooth device active, waiting for connections...
");
```

```
    //turn off modules to save power
}

//multiplexer control pins
int MUXen = 20;
int Ad0 = 21;
int Ad1 = 2;
int Ad2 = 3;

//50k bridge potentiometer (pot1)
int CS1 = 6;
int UD1 = 7;

//5k bridge potentiometer (pot2)
int CS2 = 8;
int UD2 = 9;

//50k amplifier potentiometer (pot3)
int CS3 = 4;
int UD3 = 5;

//analog inputs
int AinP = A2;
int AinN = A3;

//initialize the eight sensor reading variables
double volt1 = 0.00;
double volt2 = 0.00;
double volt3 = 0.00;
double volt4 = 0.00;
double volt5 = 0.00;
double volt6 = 0.00;
double volt7 = 0.00;
double volt8 = 0.00;

//turns and stays false after initialization of potentiometers is
complete and power remains on
bool startUp = true;
```

```
//will turn true when initialization is complete, needs to be true for
strain sensing
boolstrainSensing;

//determines time between reading sensors
//group frequency = 1 / (delay time [seconds] * number of sensors)
int delayTime = 3;  //milliseconds

//holds timestamp in milliseconds, refreshed just before reading the
eight sensors
unsigned long deviceTime = 0;

//signal transmission variables
int v1 = 0;
int v2 = 0;
int v3 = 0;
int v4 = 0;
int v5 = 0;
int v6 = 0;
int v7 = 0;
int v8 = 0;

float a_x = 0.0;
float a_y = 0.0;
float a_z = 0.0;

int acc_x = 0;
int acc_y = 0;
int acc_z = 0;



//sensor measurement variables
int Adr0;
int Adr1;
int Adr2;

intcurrentReading1;
intcurrentReading2;
```

```
intcurrentReading3;
intcurrentReading4;
intcurrentReading5;

intcurrentReadingAvg;

////////////////////////////////////////////////////////////////////////
//////////////////////////
////////////////////////////////  END SETUP VARIABLES
////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
//////////////////////////

void loop() {

   //for this board, writing HIGH to LED turns it off, writing LOW
turns it on
  digitalWrite(LED_BUILTIN, HIGH);

   //enable multiplexer, fix to channel 8 (60kOhm max)
  digitalWrite(MUXen, HIGH);

   //switch to arbitrary sensor
  digitalWrite(Ad0, LOW);
  digitalWrite(Ad1, LOW);
  digitalWrite(Ad2, LOW);

   if (startUp == true)
    {
       //five blinks to indicate startup
       //by the time the fifth blink occurs, the patient should be
sitting still with the surgical knee bent
       LEDblink(5, 250);

      //minimize potentiometers with unstrained sensors (bent knee),
brings signal up to Vcc
       minOut(1);
       minOut(2);
      Serial.println("potentiometers minimized");
```

```
      //find most resistive sensor in unstrained state
      findHighSensorIndex();

       tuneBridge();
     Serial.println("bridge tuned, beginning transduction");

      //two long blinks to indicate that the device is tuned
      //this would mean the patient is now free to move around
      LEDblink(2, 1000);

     //shut off potentiometer serial lines to conserve power
      digitalWrite(UD1, LOW);
      digitalWrite(UD2, LOW);
      digitalWrite(CS1, LOW);
      digitalWrite(CS2, LOW);
      digitalWrite(CS3, LOW);
      digitalWrite(UD3, LOW);

       Serial.
println("#########################################################");
       startUp = false;
   }

   //BLE code
   BLEDevice central = BLE.central();

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

      while (central.connected())
       {
            //1023 = 03FF
           //a hex number 0x03AB will be shown as 0xAB03 (Least
Significant Byte first)

            deviceTime = millis();
```

```
            v1 = probeSensor(1);
            v2 = probeSensor(2);
            v3 = probeSensor(3);
            v4 = probeSensor(4);
            v5 = probeSensor(5);
            v6 = probeSensor(6);
            v7 = probeSensor(7);
            v8 = probeSensor(8);

           if (IMU.accelerationAvailable())
            {
               IMU.readAcceleration(a_x, a_y, a_z);
               acc_x = int(1E8*a_x);
               acc_y = int(1E8*a_y);
               acc_z = int(1E8*a_z);
            }

          Serial.print(String(deviceTime) + " -- " + String(v1) + "
-- " + String(v2) + " -- " + String(v3) + " -- " + String(v4)
                              + " -- " + String(v5) + " -- " +
String(v6) + " -- " + String(v7) + " -- " + String(v8) + " -- ");

          //For the 16 bit IMU, override Arduino's 2 decimal place
limit for printing floating point values
          Serial.print(a_x, 8);  Serial.print(" -- ");
          Serial.print(a_y, 8);  Serial.print(" -- ");
          Serial.print(a_z, 8);  Serial.println(" -- ");


          sensor1char.writeValue(v1);
          sensor2char.writeValue(v2);
          sensor3char.writeValue(v3);
          sensor4char.writeValue(v4);
          sensor5char.writeValue(v5);
          sensor6char.writeValue(v6);
          sensor7char.writeValue(v7);
          sensor8char.writeValue(v8);
          timeMillis.writeValue(deviceTime);
```

```
            accelX.writeValue(acc_x);
            accelY.writeValue(acc_y);
            accelZ.writeValue(acc_z);

        }

    }
   digitalWrite(LED_BUILTIN, LOW);
   Serial.print("Disconnected from central: ");
  Serial.println(central.address());
   delay(2000);
}

//note - "increment" = moving wiper towards terminal A (decreasing
resistance between A and W)
//    - "decrement" = moving wiper towards terminal B (increasing
resistance between A and W)

void LEDblink(int numTimes, int duration)
{
    for (int i = 1; i <= numTimes; i++)
    {
      digitalWrite(LED_BUILTIN, HIGH); delay(duration);
      digitalWrite(LED_BUILTIN, LOW);  delay(duration);
    }
}

void minOut(int potNum)
{
    int CSpin;
    int UDpin;
   int currentReading;

    if (potNum == 1)
    {
      CSpin = 8;
      UDpin = 9;
    }
```

```
   if (potNum == 2)
   {
     CSpin = 6;
     UDpin = 7;
   }

   if (potNum == 3)
   {
     CSpin = 4;
     UDpin = 5;
   }

   //initialize UD to high, then drive CS low
   //to increment, drop UD low, then drive it to high again
   //this rising edge is what triggers the wiper to move up towards
terminal A, decreasing A-W resistance
   digitalWrite(CSpin, HIGH);
   digitalWrite(UDpin, HIGH);
    delay(25);
   digitalWrite(CSpin, LOW);
    delay(25);
  Serial.println("minimizing");
    for (int i = 0; i < 64; i++)
    {
     digitalWrite(UDpin, LOW);
      delay(25);
     digitalWrite(UDpin, HIGH);
      delay(25);

     currentReading = analogRead(AinP);
     Serial.print("potentiometer ");
     Serial.print(potNum);
      Serial.print(" - ");
     Serial.print(i);
     Serial.print(" - current reading - ");
    Serial.println(currentReading);
   }

  digitalWrite(CSpin, HIGH);
```

```
  digitalWrite(UDpin, HIGH);
}

void maxOut(int potNum)
{
 int CSpin;
 int UDpin;
 int currentReading;

 if (potNum == 1)
 {
    CSpin = 8;
    UDpin = 9;
 }

 if (potNum == 2)
 {
    CSpin = 6;
    UDpin = 7;
 }

 if (potNum == 3)
 {
    CSpin = 4;
    UDpin = 5;
 }

 //initialize UD to low, then drive CS low
 //to decrement, drive UD high, then drive it low again
 //this rising edge is what triggers the wiper to move down towards
terminal B, increasing A-W resistance
 digitalWrite(CSpin,HIGH);
 digitalWrite(UDpin,LOW);
 delay(25);
 digitalWrite(CSpin,LOW);
 delay(25);
Serial.println("maximizing");

 for (int i = 0; i <= 64; i++)
```

```
 {
   digitalWrite(UDpin, HIGH);
    delay(25);
   digitalWrite(UDpin, LOW);
    delay(25);

   currentReading = analogRead(AinP);
   Serial.print("potentiometer ");
   Serial.print(potNum);
    Serial.print(" - ");
   Serial.print(i);
    Serial.print(" - current reading - ");
  Serial.println(currentReading);
 }

digitalWrite(CSpin,HIGH);
digitalWrite(UDpin,HIGH);
}

voidfindHighSensorIndex()
{
   //make sure that you minimize the pot before launching this one
   int currentReading = analogRead(AinP);

   //increase pot1 A-W resistance until it renders a bridge voltage
of  600 (1.6 V)
   digitalWrite(CS1, HIGH);
   digitalWrite(UD1, LOW);
    delay(25);
   digitalWrite(CS1, LOW);
    delay(25);
   while (currentReading > 600) //1024 discrete intervals / 64 wiper
steps = 16/1024 code points per resistance increase
    {
     digitalWrite(UD1, HIGH);
      delay(25);
     digitalWrite(UD1, LOW);
      delay(25);
     currentReading = analogRead(AinP);
```

```
  Serial.print("tuning bridge pot2 - Current reading - ");
  Serial.println(currentReading);
 }

//initialize eight sensor variables
int sensorValues[8];

 //probe sensor 1
digitalWrite(Ad0, LOW);
digitalWrite(Ad1, LOW);
digitalWrite(Ad2, LOW);
 delay(10);
sensorValues[0] = analogRead(AinP);

 //probe sensor 2
digitalWrite(Ad0, HIGH);
digitalWrite(Ad1, LOW);
digitalWrite(Ad2, LOW);
 delay(10);
sensorValues[1] = analogRead(AinP);

 //probe sensor 3
digitalWrite(Ad0, LOW);
digitalWrite(Ad1, HIGH);
digitalWrite(Ad2, LOW);
 delay(10);
sensorValues[2] = analogRead(AinP);

 //probe sensor 4
digitalWrite(Ad0, HIGH);
digitalWrite(Ad1, HIGH);
digitalWrite(Ad2, LOW);
 delay(10);
sensorValues[3] = analogRead(AinP);

 //probe sensor 5
digitalWrite(Ad0, LOW);
digitalWrite(Ad1, LOW);
digitalWrite(Ad2, HIGH);
```

```
   delay(10);
   sensorValues[4] = analogRead(AinP);

   //probe sensor 6
   digitalWrite(Ad0, HIGH);
   digitalWrite(Ad1, LOW);
   digitalWrite(Ad2, HIGH);
   delay(10);
   sensorValues[5] = analogRead(AinP);

   //probe sensor 7
   digitalWrite(Ad0, LOW);
   digitalWrite(Ad1, HIGH);
   digitalWrite(Ad2, HIGH);
   delay(10);
   sensorValues[6] = analogRead(AinP);

   //probe sensor 8
   digitalWrite(Ad0, HIGH);
   digitalWrite(Ad1, HIGH);
   digitalWrite(Ad2, HIGH);
   delay(10);
   sensorValues[7] = analogRead(AinP);

   Serial.print("The starting values are: ");
   Serial.println(String(sensorValues[0]) + " -- " +
String(sensorValues[1]) + " -- " + String(sensorValues[2]) + " -- " +
String(sensorValues[3]) + " -- " +
                  String(sensorValues[4]) + " -- " +
String(sensorValues[5]) + " -- " + String(sensorValues[6]) + " -- " +
String(sensorValues[7]));

   //find sensor with highest starting resistance value (highest
reading)
   int highSensorIndex = 0;

   for (int i = 0; i < 8; i++)
   {
     if (sensorValues[i] > sensorValues[highSensorIndex])
```

```
    {
       highSensorIndex = i;
    }
}

int Adr0;
int Adr1;
int Adr2;

if (highSensorIndex == 0)
{
    Adr0 = 0;
    Adr1 = 0;
    Adr2 = 0;
}

if (highSensorIndex == 1)
{
    Adr0 = 1;
    Adr1 = 0;
    Adr2 = 0;
}

if (highSensorIndex == 2)
{
    Adr0 = 0;
    Adr1 = 1;
    Adr2 = 0;
}

if (highSensorIndex == 3)
{
    Adr0 = 1;
    Adr1 = 1;
    Adr2 = 0;
}

if (highSensorIndex == 4)
{
```

```
        Adr0 = 0;
        Adr1 = 0;
        Adr2 = 1;
    }

   if (highSensorIndex == 5)
    {
        Adr0 = 1;
        Adr1 = 0;
        Adr2 = 1;
    }

   if (highSensorIndex == 6)
    {
        Adr0 = 0;
        Adr1 = 1;
        Adr2 = 1;
    }

   if (highSensorIndex == 7)
    {
        Adr0 = 1;
        Adr1 = 1;
        Adr2 = 1;
    }

   digitalWrite(Ad0, Adr0);
   digitalWrite(Ad1, Adr1);
   digitalWrite(Ad2, Adr2);

   Serial.println("The high sensor is number " +
String(highSensorIndex+1));
   delay(3000);
}



voidtuneBridge()
{
```

```
  int currentReading = analogRead(AinP);

  //increase pot1 A-W resistance until it renders a bridge voltage
under 16/1024 (15.6 mV)
  digitalWrite(UD1, LOW);
  delay(25);
  digitalWrite(CS1, LOW);
  delay(25);
  while (currentReading > 16) //1024 discrete intervals / 64 wiper
steps = 16/1024 code points per resistance increase
   {
     digitalWrite(UD1, HIGH);
      delay(25);
     digitalWrite(UD1, LOW);
      delay(25);
     currentReading = analogRead(AinP);
     Serial.print("tuning pot1 - Current reading - ");
    Serial.println(currentReading);
   }
  digitalWrite(CS1, HIGH);

  //decrease pot1 three times
  digitalWrite(UD1, HIGH);
  delay(25);
  digitalWrite(CS1, LOW);
  delay(25);
  for (int i = 1; i <= 3; i++) //1024 discrete intervals / 64 wiper
steps = 16/1024 code points per resistance increase
   {
     digitalWrite(UD1, LOW);
      delay(25);
     digitalWrite(UD1, HIGH);
      delay(25);
     currentReading = analogRead(AinP);
     Serial.print("tuning pot1 - Current reading - ");
    Serial.println(currentReading);
   }

  //lock off potentiometer 1
```

```
  digitalWrite(CS1, HIGH);
  delay(25);
  digitalWrite(UD1, HIGH);

  Serial.println("bridge tuning - potentiometer 1 set.");
  delay(1000);

  //increase pot2 A-W resistance until it renders a bridge voltage
under 24/1024
  digitalWrite(UD2, LOW);
  delay(25);
  digitalWrite(CS2, LOW);
  delay(25);
  while (currentReading > 24) //1024 discrete intervals / 64 wiper
steps = 16 of 1024 code points per resistance change
   {
     digitalWrite(UD2, HIGH);
     delay(25);
     digitalWrite(UD2, LOW);
     delay(25);
     currentReading = analogRead(AinP);
     Serial.print("tuning pot2 - Current reading - ");
     Serial.println(currentReading);
   }

  //increase one more time to bias slightly below zero
   digitalWrite(UD2, HIGH);
   delay(25);
   digitalWrite(UD2, LOW);
   delay(500);
   currentReading = analogRead(AinP);
   Serial.print("pot2 tuned - Current reading - ");
  Serial.println(currentReading);

   delay(1000);
}



int probeSensor(int sensNum)
```

```
{

    if (sensNum == 1)
    {
        Adr0 = 0;
        Adr1 = 0;
        Adr2 = 0;
    }

    if (sensNum == 2)
    {
        Adr0 = 1;
        Adr1 = 0;
        Adr2 = 0;
    }

    if (sensNum == 3)
    {
        Adr0 = 0;
        Adr1 = 1;
        Adr2 = 0;
    }

    if (sensNum == 4)
    {
        Adr0 = 1;
        Adr1 = 1;
        Adr2 = 0;
    }

    if (sensNum == 5)
    {
        Adr0 = 0;
        Adr1 = 0;
        Adr2 = 1;
    }

    if (sensNum == 6)
    {
```

```
        Adr0 = 1;
        Adr1 = 0;
        Adr2 = 1;
    }

    if (sensNum == 7)
    {
        Adr0 = 0;
        Adr1 = 1;
        Adr2 = 1;
    }

    if (sensNum == 8)
    {
        Adr0 = 1;
        Adr1 = 1;
        Adr2 = 1;
    }

   digitalWrite(Ad0, Adr0);
   digitalWrite(Ad1, Adr1);
   digitalWrite(Ad2, Adr2);

   delayMicroseconds(2000);

   currentReading1 = analogRead(AinP);
   currentReading2 = analogRead(AinP);
   currentReading3 = analogRead(AinP);
   currentReading4 = analogRead(AinP);
   currentReading5 = analogRead(AinP);

   currentReadingAvg = (currentReading1 + currentReading2 +
currentReading3 + currentReading4 + currentReading5) / 5;

   return currentReadingAvg;
}

//not utilized currently
void tuneGain()
```

```
{
  //int currentReading = analogRead(AinP);

   int numWipes = 55;

   //initialize UD to high, then drive CS low
   //to increment, drop UD low, then drive it to high again
   //this rising edge is what triggers the wiper to move up towards
terminal A, decreasing A-W resistance
  Serial.println("decreasing gain resistor...");

  digitalWrite(UD3, HIGH);
   delay(25);
  digitalWrite(CS3, LOW);
   delay(25);

   for (int i = 1; i <= numWipes; i++)
   {
     digitalWrite(UD3, LOW);
      delay(50);
     digitalWrite(UD3, HIGH);
      delay(25);

     //currentReading = analogRead(AinP);
     Serial.print("potentiometer 3 - ");
     Serial.println(i);
     //Serial.print(i);
     //Serial.print(" - current reading - ");
    //Serial.println(currentReading);
   }

  digitalWrite(CS3, LOW);
  digitalWrite(UD3, LOW);
}



//voidtuneGain()
//{
```

```
//  //int currentReading = analogRead(AinP);
//
//   int numWipes = 55;
//
//   //initialize UD to high, then drive CS low
//   //to increment, drop UD low, then drive it to high again
//   //this rising edge is what triggers the wiper to move up towards
terminal A, decreasing A-W resistance
//  Serial.println("decreasing gain resistor...");
//
//  digitalWrite(UD1, HIGH);
//   delay(25);
//  digitalWrite(CS1, LOW);
//   delay(25);
//
//   for (int i = 1; i <= numWipes; i++)
//   {
//     digitalWrite(UD1, LOW);
//      delay(50);
//     digitalWrite(UD1, HIGH);
//      delay(25);
//
//     //currentReading = analogRead(AinP);
//     Serial.print("amp potentiometer 1 - ");
//     Serial.println(i);
//     //Serial.print(i);
//     //Serial.print(" - current reading - ");
//    //Serial.println(currentReading);
//   }
//
//  digitalWrite(CS1, LOW);
//  digitalWrite(UD1, LOW);
//}
```