3P04 Design Project: Remote Posture and Gait Monitoring after Spinal Stenosis Surgery

Hamdan Basharat, Fatima Ahmad, Konrad Grala, and Sharda Kotru
Integrated Biomedical Engineering & Health Sciences
McMaster University, Canada
Email: basham1@mcmaster.ca

***Abstract*—After spinal stenosis surgery, patients are often left with postural and gait imbalances that lead them to seek rehabilitation. Our team has developed a prototype measurement system that assists in this by providing hip angle data to the user and their care team. By considering the end user's needs and any constraints in their use of the device, the system was designed to take IMU sensor readings from a belt clip-on unit and send them to a software user-interface on a laptop or phone. The gait cycle data can be used by health care professionals to track recovery process and suggest new interventions.**

I. Introduction & Background Theory

Lower back pain is estimated to affect 80% of adults worldwide at some point in their life. It can be acute, sub-acute, or chronic and is a major cause of disorders and disabilities. A possible factor to lower back pain is Lumbar Spinal Stenosis (LSS), which occurs when the lumbar spine nerves are compressed due to the narrowing of the spinal canal. These nerves innervate the lower back and legs, and when compressed cause pain, numbness, and dysfunction in these innervated muscles. If conservative therapy doesn't work, lumbar stenosis surgery is the next option, where the spine may be fused to stabilize it, and/or the roof of the vertebrae is removed through a laminectomy to make space for nerves. After the operation, rehabilitation is required for the patient to regain mobility of the affected regions and to strengthen their spine. The patient's walking gait is heavily affected and can be characterized by a Lumbar Spinal Stenosis (LSS) gait and/or a Trendelenburg gait. Monitoring their walking mechanics is imperative to their rehabilitation as it indicates which movements need to improve and which muscles are still weak.

Our aim was to help this process by designing and prototyping a device/measurement system that monitors and supports patient rehabilitation after their spinal stenosis surgery. With an accurate measurement and graphical representation of their gait cycle, patients along with the help of their physiotherapists, will be able to better understand where in their gait and posture they are lacking, and what needs to be done for improvement. This device could serve as an imperative tool to progress patient health and quality of life.

II. Methods

We had looked into other novel and available solutions to this challenge when designing our own and found plenty of ways to implement a monitoring system. These varied from strain gauges adhered to the patient's back in order to measure spinal flexion and extension, to smart garments containing myosensors in them to track contraction in hip abductor and adductor muscles. By far, the most widely used method, was tracking the patients spine or hip position using inertial measurement unit (IMU) sensors and their built-in gyroscopes. We had proposed and decided on a similar solution but we were limited to two IMU sensors at most. Many of these designs used 3 to 6 sensors along the hip and spine to track minute changes in gait and posture. We would not be able to compete in accuracy, so in order to establish our project as different from the rest of the market we looked at what a lot of these products were missing. We aimed to design a system that not only gathered postural data and displayed it, but could also analyse gait and postural swaying with potential training exercises to help the patient recover from post surgery symptoms. Our design is targeted towards the patient and their care team, and aims to easily provide them with the information they need to improve the rehabilitation process.

A. Design Criteria

Accuracy - As our device will be a class I medical tool, it is imperative that it can provide accurate information about the patients hip position during gait. The user will rely on this data to make further judgements about the patient's recovery and necessary strengthening exercises.
Remote - The device/measurement system must be able to be used in a remote setting such as the patients home, a clinic, or in the field. This allows for the user to easily collect data whenever necessary, whether they are the patient in their home or a physiotherapist in their clinic observing the patient's progress.
Flexibility & Adaptability - Ideally, the device should be able to be used by a variety of patients. This means that with or without being adjusted, it should accommodate patients of different height and weight when tracking walking hip gait.
Weight - The entire wearable unit should be very lightweight when worn by the user. This ensures that the addition of the device to the person will not affect their gait and skew results.
Installation & Handling - The installation and set-up for the system will be very straightforward and easy to follow. Users will only have to worry about turning the device on, then pairing the device with their laptop or cell phone. Data processing and visualization will happen automatically and can

also be sent to their rehab professional.

Maintenance - Daily use should require very minimal user maintenance/intervention to uphold optimal functioning. The user may be required to recharge or replace the batteries on occasion which should not interfere or reduce the ability to use the device.

Durability - Must withstand regular day-to-day wear and fatigue including drops and falls. Sealed packaging should provide some water resistance and protection of the internals from rain and splashes.

Cost - The parts used in the device, including sensors and the micro-controller, will be reasonably priced in order to reduce manufacturing and product cost.

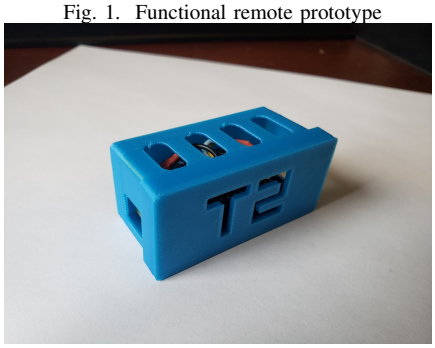
B. Prototype Development

To start building a prototype we first came up with several potential designs for a remote wearable measurement system. We took inspiration from available products and looked at what we could do to implement our own solution. Initially, our design relied on placing IMU sensors along the lumbar and sacral positions of the spine in order to the track the spinal curves and the patients posture. We decided to move away from this idea for two reasons: limited availability of sensors, and project focus. Since the sensors we were using were provided by the Design Studio and there was a limited supply available, we only had access to two IMUs. This would hinder our ability to accurately measure spinal curvature. Also, while poor posture is a key issue found in post-spinal stenosis surgery patients, it is a derivative of abnormal hip movements during gait. From here we pivoted to a design that placed a sensor on the side of the patients hip in order to map their hip angle in the coronal. sagittal, and transverse planes while walking.

By setting up the ESP32 micro-controller with the IMU sensor, we were able to conduct some primary tests. This allowed us to determine how the sensor would be oriented on the patient during use, and which movements would change the angle in the corresponding plane. To conduct an actual walking test, the communication between the micro-controller and computer was changed to Bluetooth and the hardware was placed in a fanny pack adjusted onto a team member's hip. By having him do several walking trials at different lengths with the same pace, we achieved a good understanding of the data being retrieved. Further iterations of this included shortening the wires used till the unit was compact and could be taped onto the members belt to avoid it moving too much. Eventually, several 3D printed PLA housings were developed to contain the hardware till a final lightweight, durable iteration was achieved.

At this point, the base data collection was refined so that during and after a walking trial the angle of hip in the coronal, sagittal, and transverse planes could be seen at a high sample collection rate. Now the effort was shifted to work towards developing a user-interface that would display the results of the data and perform further processing. Many iterations of this interface were developed using the graphical

Fig. 1. Functional remote prototype



Java software, Processing 3. We looked at what we found would be necessary information for the user and their care team and included it in a sleek, visually appealing interface. At this stage in our prototype development process, we had the opportunity to have our design reviewed by a panel of of experts after a presentation and demonstration. Using the very relevant feedback we had received, considerations were taken to improve the validity of our design. Modifications to the calibration of the sensor were made in order to reduce the noise that was influencing the hip angle. As well, the addition of a second IMU sensor was proposed which we decided to consider once a valid prototype with one sensor was established.

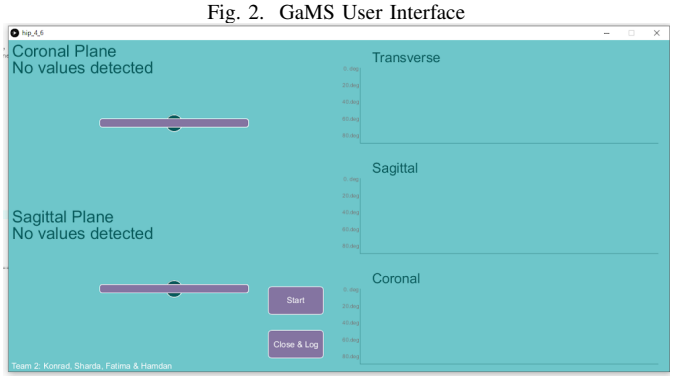
The prototype was now functioning as intended with both the physical data collection, and the graphical user interface. The next step was to further validate our results and attempt to distinguish individual gait cycles.

C. Analysis

Data Collection - To acquire data from the BNO055 unit, serial communication from the device to our computer was set up. In the Arduino IDE the Adafruit library gave us the ability to find orientation, angular velocity, and linear acceleration events on the device. The ESP was made to sample these events at the specified baud rate and combine them at each serial event into a comma separated string. This string was outputted from the ESP over Bluetooth serial communication to a COM port on our computer. Using the Processing 3 Java-based software we were able to search the computers UART communication ports for a Bluetooth port receiving data at the same baud rate and establish a connection. While the connection is active, the string of values is decoded and saved to temporary variables with which we update visuals on the user interface.

User Interface - The user interface consists of two representative visuals of the patient's hip angle from a coronal and sagittal view which updates in real time giving the user immediate feedback on their posture. Alongside the two animations, there are three graphs of the hip angle in real time of the three planes. Two buttons at the bottom of the display allow the user start and stop logging the angle data, which is then saved to the computer as a text log file. This was all created using Processing 3 software with the Java programming language.

Fig. 2. GaMS User Interface



Calibration - The gyroscopic measurements from the IMU were calibrated using the Adafruit calibration library. Minimal noise was received when collecting data, so further calibration methods such as physical noise reduction with op-amps was determined unnecessary.

Processing Algorithm - In order to distinguish the patients gait so it could be examined by a medical professional, we had to find a way to normalize the data. Several trials were run to collect data to test an algorithm we developed. It was also determined that gyroscope data from the y (sagittal) and z (coronal) planes would be most relevant to identifying gait abnormalities. To identify the start and end of a gait cycle, first a moving average to determine the mid line of the data. With half the signal above the mean and half the signal below, we identified the period of the signal which is one high plus one low. From here we know the start and end of each gait cycle and can plot it in terms of percentage gait cycle with which physicians compare patient performance to a baseline.

III. Results

Our final design, under the name ”GaMS: A Remote Gait Monitoring System”, is a user-friendly remote system that uses gyroscopic measurements from an IMU sensor to track and display the angle of the user’s hip. The 3D printed PLA housing contains the device hardware, and with the metal clip on it’s back it can be attached to the patient’s belt. The rechargeable battery can easily slip into the patient’s pocket during use and has the device’s power button on it. The ESP32 micro-controller can be connected by Bluetooth connection to the user’s laptop or phone on which the associated software displays a sleek user interface. With the user interface, the user can visually see the angle of the patient’s hip in the coronal, sagittal, and transverse in real time as either a bar with fulcrum or a moving graph. On-screen buttons allow for the data to be logged and saved onto the device. A built-in algorithm distinguishes the start and end of each gait cycle for interpretation. This data can be sent to the patient’s care team to help determine what exercises must be done by the patient to further their rehabilitation and to track their progress.

To begin use, the user must first install the associated software with the user interface onto their device of choice. For use on a computer, the program Processing 3 and the GaMS processing file is required. For use on a phone, the GaMS android phone application is required. Next, by pressing the button on the battery the user can power on the GaMS unit. On their chosen device, by navigating to available Bluetooth connections they will now find a connection named ”GaMS” that they can connect to. Once the Bluetooth connection is made, they will press ”Run” on the Processing software, or start the Android application. The user interface will appear showing the real-time bar and graphs. Now the user can clip the housing onto their belt adjacent to their hip and place the rechargeable battery pack in their pocket. As they walk, they will be able to see the real time models update with their hip angle. By pressing the ”Start” button on the interface, the software will start logging their angle data as they proceed to walk. When finished, they can press the ”Close & Log” button which will save the logged data and close the program. The log file can now be sent to the care team so they can view the gait cycle results.

IV. Discussion

Patients who have undergone lumbar spinal stenosis surgery are often inflicted with poor posture and gait conditions such as Lumbar Spinal Stenosis (LSS) gait and/ or Trendelenburg gait. Post-surgery rehabilitation requires them to perform muscle strengthening exercises to help get them back to normal. With our design, we were able to create a prototype measurement system that assists in this process by providing accurate hip angle data that shows the patients gait cycle. With this data, the patients care team can effectively monitoring their recovery process and suggest new exercises and treatment. The design of the system is simple, and can be seamlessly integrated into the patient home and clinical setting. The ability to monitor the patient’s gait is desirable for good rehabilitation. The cost of the device is low due to the low price of components being used. Only the physical hardware must be purchased and the software application can be downloaded onto the user’s own laptop or phone.

In the current design, we are limited in feedback and production. Feedback to the user can only be presented on the user interface, however in the future additional hardware can be added such as vibration modules or a speaker. We are limited in our production capabilities for the devices, and in the future can look towards financing to streamline mass production. Other considerations we have for the future are to work with licensed therapists to further develop the software to allow for therapeutic feedback. Ideally, the device would have long-term data storage and two-way communication between the patient and therapist. As well, the housing could be made more attractive and durable, and the battery could be replaced with a much smaller LiPo version.

V. Conclusions

As a team, we were able to develop a prototype remote gait monitoring system to be used to assist in the rehabilitation of patients who have undergone lumbar spinal stenosis surgery.

The prototype design has a physical hardware unit that communicates with a computer or phone software displaying hip angle data and recording it for use by the patient's care team.

REFERENCES

[1] Patients, American Chiropractic Association Home. [Online]. Available: https://www.acatoday.org/Patients/What-is-Chiropractic/Back-Pain-Facts-and-Statistics. [Accessed: 06-Apr-2020].

[2] K. Townsend, Adafruit BNO055 Absolute Orientation Sensor, Adafruit Learning System. [Online]. Available: https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/overview. [Accessed: 06-Apr-2020].

[3] K. O'Sullivan, L. O'Sullivan, A. Campbell, P. O'Sullivan, and W. Dankaerts, Towards monitoring lumbo-pelvic posture in real-life situations: Concurrent validity of a novel posture monitor and a traditional laboratory-based motion analysis system, Manual Therapy. 2012;17(1):7783. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/22015373. [Accessed January 28, 2020]

[4] Q. Wang, M. Toeters, W. Chen, A. Timmermans, and P. Markopoulos, Zishi, Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA 16, 2016. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/2851581.2890262?download=true. [Accessed January 28, 2020]

[5] J. Xu, T. Bao, U. H Lee et al. Configurable, wearable sensing and vibrotactile feedback system for real-time postural balance and gait training: proof-of- concept. J Neuroeng Rehabil. 2017;14(102):110. [Online]. Available: https://jneuroengrehab.biomedcentral.com/articles/10.1186/s12984-017-0313-3. [Accessed January 27, 2020].

APPENDIX A.
TEAM MEMBER ROLES

While everyone had their own responsibilities for this project, their contribution were not limited to what is stated below. Every member had a hand in helping each other with tasks other than their own.

Hamdan Basharat - Lead Data Acquisition Programmer (worked in Arduino and with circuitry to collect and send data from IMU)
Fatima Ahmad - Lead CAD Designer (designed all iterations of the housing), Administrator (organizing tasks, scheduling meeting & rooms, written work review)
Konrad Grala - Lead User-Interface Software Programmer (worked in Processing 3 to develop an appealing and functional UI)
Sharda Kotru - Lead Data Analyst (calibrated data and developed an algorithm to determine gait cycles)

For the tasks that I was involved in, I did the majority of the data acquisition of measurements from the IMU. I sourced and wrote code in Arduino that allowed a Bluetooth connection from the ESP32 to the computer allowing serial communication at a COM port. From there I refined what data was sent and what was not and worked with Sharda on device calibration. On Processing, I again set up serial communication to synchronize data collection with the program and helped Konrad with the interface. Specifically, I made the functional buttons that allow for the logging of data.

APPENDIX B.
DRAWINGS

Fig. 3. Design Idea Sketch 1

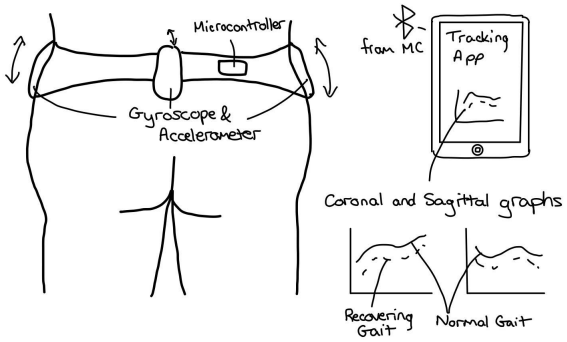


Fig. 4. Design Idea Sketch 2

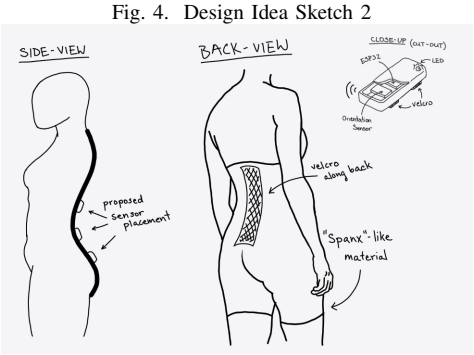


Fig. 5. Initial CAD Housing View 1

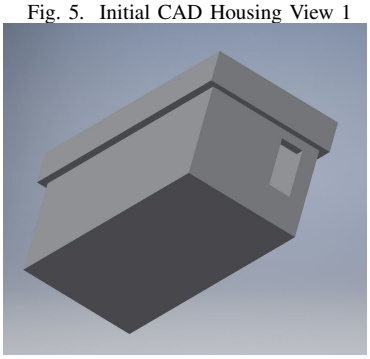
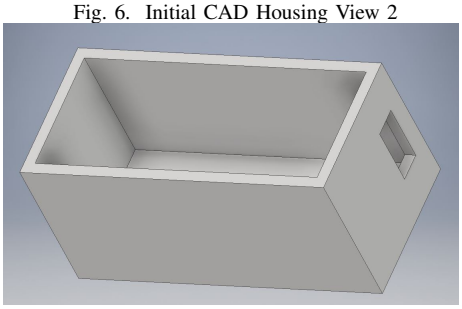


Fig. 6. Initial CAD Housing View 2



APPENDIX C.
ANALYSIS CODE

Arduino Code

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>

#include "BluetoothSerial.h"

/* This driver uses the Adafruit unified
sensor library (Adafruit_Sensor),
   which provides a common 'type' for sensor
   data and some helper functions.

   To use this driver you will also need to
   download the Adafruit_Sensor
   library and include it in your libraries
   folder.

   You should also assign a unique ID to
   this sensor for use with
   the Adafruit Sensor API so that you can
   identify this particular
   sensor in any data logs, etc.
To assign
   a unique ID, simply
   provide an appropriate value in the
   constructor below (12345
   is used by default in this example).

   Connections
   ===========
   Connect SCL to analog 5
   Connect SDA to analog 4
   Connect VDD to 3.3-5V DC
   Connect GROUND to common ground

   History
   =======
   2015/MAR/03  - First release (KTOWN)
*/

/* Set the delay between fresh samples */
uint16_t BNO055_SAMPLERATE_DELAY_MS = 100;

// Check I2C device address and correct
line below (by default address is 0x29
or 0x28)
//                                   id, address
Adafruit_BNO055 bno = Adafruit_BNO055
(55, 0x28);

BluetoothSerial SerialBT;

void setup(void)
{
  Serial.begin(115200);
  SerialBT.begin("Group2ESP");

  //SerialBT.println("Orientation Sensor
  Test"); Serial.println("");
  /* Initialise the sensor */
  if (!bno.begin())
  {
    /* There was a problem detecting the
     BNO055 ... check your connections */
    Serial.print("Ooops, no BNO055 detected
     ... Check your wiring or I2C ADDR!");
    while (1);
  }

  delay(1000);
}

void loop(void)
{
  //could add VECTOR_ACCELEROMETER,
  VECTOR_MAGNETOMETER,VECTOR_GRAVITY...
  sensors_event_t orientationData ,
  angVelocityData , linearAccelData;
  bno.getEvent(&orientationData ,
  Adafruit_BNO055::VECTOR_EULER);
  bno.getEvent(&angVelocityData ,
  Adafruit_BNO055::VECTOR_GYROSCOPE);
  bno.getEvent(&linearAccelData ,
  Adafruit_BNO055::VECTOR_LINEARACCEL);

  String dataString = "";

  String od = printEvent(&orientationData);
  String av = printEvent(&angVelocityData);
  String la = printEvent(&linearAccelData);

  Serial.println(od+av+la);

  dataString = "o" + od + "a" + av + "l"
   + la ;          //special data string
   to be sent through bluetooth
  SerialBT.println(dataString);

  //send string through bluetooth


  delay(BNO055_SAMPLERATE_DELAY_MS);
}

String printEvent(sensors_event_t* event) {
  Serial.println();
  Serial.print(event->type);
```

```
  double x = -1000000, y = -1000000 ,
  z = -1000000; //dumb values, easy to
  spot problem
  if (event->type ==
  SENSOR_TYPE_ACCELEROMETER) {
    x = event->acceleration.x;
    y = event->acceleration.y;
    z = event->acceleration.z;
  }
  else if (event->type ==
  SENSOR_TYPE_ORIENTATION) {
    x = event->orientation.x;
    y = event->orientation.y;
    z = event->orientation.z;
  }
  else if (event->type ==
  SENSOR_TYPE_MAGNETIC_FIELD) {
    x = event->magnetic.x;
    y = event->magnetic.y;
    z = event->magnetic.z;
  }
  else if ((event->type ==
  SENSOR_TYPE_GYROSCOPE) ||
  (event->type ==
  SENSOR_TYPE_ROTATION_VECTOR)) {
    x = event->gyro.x;
    y = event->gyro.y;
    z = event->gyro.z;
  }

  Serial.print(": x= ");
  Serial.print(x);
  Serial.print(" | y= ");
  Serial.print(y);
  Serial.print(" | z= ");
  Serial.println(z);

  String xyz = "x" + String(x) +
  "y" + String(y) + "z" + String(z);
  return xyz;
}


//get each piece of sensor data
//combine into a single string
//send string to console or through
bluetooth

//orientation (xyz), angle velocity
(xyz), linear acceleration(xyz)
// sendString = "ox####y####z####ax
####y####z####lx####y####z####"
```

Processing Code

```
/* ************************************
 * Spinal Stenosis Post-Op Monitoring
 * Authors: Konrad Grala, Hamdan Basharat,
Sharda Kotru, and Fatima Ahmad
 * Version 4.5
 ************************************/
import processing.serial.*;
//for bluetooth communication
import org.gicentre.utils.stat.*;
//for chart classes.

Serial mySerial;
PImage pic;
Table table;
String filename;

XYChart graph1;
XYChart graph2;
XYChart graph3;

float buttX, buttY, butt2X, butt2Y;
int buttSize = 100;
color rectColor;
color rectHighlight;
boolean rect1Over = false;
boolean rect2Over = false;
boolean startFlag = false;

float[] xhipVals = new float[30];
float[] xVals = new float[30];
float[] temphipx = new float[30];
float[] tempx = new float[30];
float[] yhipVals = new float[30];
float[] yVals = new float[30];
float[] temphipy = new float[30];
float[] tempy = new float[30];
float[] zhipVals = new float[30];
float[] zVals = new float[30];
float[] temphipz = new float[30];
float[] tempz = new float[30];
float ox, oy, oz, accx, accy, accz;
String[] temp;

float xPoint;
float yPoint;
float zPoint;
String foundPort;
boolean portFound = false;
String currentString = "";

void setup() {
  frameRate(240);

  //setup background form (the window)
  and dimension
  //search through all ports to find
  the right connection
```

```
size(1200,600);
background(8,151,157); //set background
colour
rectColor = color(132,116,161);
rectHighlight = color(204,171,216);

//pic = loadImage("pelvis.png");
table = new Table();
table.addColumn("X");
table.addColumn("Y");
table.addColumn("Z");
/*
table.addColumn("acc X");
table.addColumn("acc Y");
table.addColumn("acc Z");
*/

//=========== FIND AND SETUP BLUETOOTH
while (portFound == false){
  String[] portList = mySerial.list();
  for(String strPort : portList){
    println(strPort);
    try{
      mySerial = new Serial(this,
      strPort, 115200);
      //waits until you have the whole
      line because of BufferUntil. "10"
      is referring to ASCII linefeed
      mySerial.bufferUntil(10);
      delay(1000);
    } catch (RuntimeException e){
        println("error");
      }
    if (currentString.length() > 0){
      portFound = true;
      foundPort = strPort;
      print("PORT FOUNDDDD!");
      break;
    }
  }
  delay(100);
}
println();
print("found ");
println(foundPort);

buttX = width/1.9-buttSize-10;
buttY = height/1-buttSize/2;
butt2X = buttX;
butt2Y = height/1.15-buttSize/2;

//=========== SETUP GRAPHS
// create graph objects
graph1 = new XYChart(this);
graph2 = new XYChart(this);
graph3 = new XYChart(this);
  setupGraph(graph1);
  setupGraph(graph2);
  setupGraph(graph3);
}

void draw(){
  update(mouseX, mouseY);
  background(110,198,202); //set background
   colour (this will also erase all previous
   shapes that were drawn

  if(rect1Over){fill(rectHighlight);}
  else{fill(rectColor);}
  stroke(255);
  rect(buttX, buttY, buttSize, buttSize/2,7);

  if(rect2Over){fill(rectHighlight);}
  else{fill(rectColor);}
  stroke(255);
  rect(butt2X, butt2Y, buttSize, buttSize/2,7);

  fill(255,255,255);
  textSize(buttSize*0.15);
  text("Close & Log", width/2.03-buttSize-10,
  height/0.99-buttSize/2);

  fill(255,255,255);
  textSize(buttSize*0.15);
  text("Start", width/1.95-buttSize-10,
  height/1.14-buttSize/2);

  fill(255,255,255);
  textSize(buttSize*0.15);
  text("Team 2: Konrad, Sharda, Fatima &
  Hamdan", 5, height-5);

  //==##==##==  draw hip rectangles
  hipRect(0, 0, width/2, height/2, zPoint ,
  "Coronal Plane");
  hipRect(0, height/2, width/2, height/2,
  yPoint, "Sagittal Plane");

  drawGraph(graph1,width/2,0,width/2,
  height/3,xVals,xhipVals, "Transverse");
  drawGraph(graph2,width/2,height/3,
  width/2,height/3,yVals,yhipVals, "Sagittal");
  drawGraph(graph3,width/2,height*2/3,
  width/2,height/3,zVals,zhipVals, "Coronal");
}

//=========== READ AND UPDATE DATA
void update(int x, int y){
    if(overRect(buttX, buttY, buttSize,
    buttSize/2)){
      rect1Over = true;
```

```
    }
    if(overRect(butt2X, butt2Y, buttSize,
    buttSize/2)){
      rect2Over = true;
    }
}

void mousePressed(){
  if(rect2Over){
    startFlag = true;
  }
  if(rect1Over){
    filename = "gaittest_log.csv";
    saveTable(table, filename);
    exit();
  }
}

boolean overRect(float x, float y, float
width, float height){
  if(mouseX >= x && mouseX <= x+width &&
   mouseY >= y && mouseY <= y+height){
    return true;
  }
  else{
    return false;
  }
}

//function that is automatically called
every time there is data available
void serialEvent(Serial foundPort){
  //read new data string
  //and update 'currentPoint'
  currentString = foundPort.readString();
  println(currentString);

  temp = split(currentString,',');
  ox = float(temp[0]);
  oy = float(temp[1]);
  oz = float(temp[2]);
  accx = float(temp[3]);
  accy = float(temp[4]);
  accz = float(temp[5]);

  //println(ox,accx);
  //println();

  //update rectangle's current data point
  xPoint = ox;
  yPoint = oy;
  zPoint = oz;

  if(startFlag == true){
    TableRow newRow = table.addRow();
    newRow.setFloat("X", xPoint);
    newRow.setFloat("Y", yPoint);
    newRow.setFloat("Z", zPoint);
  }
  /*
  newRow.setFloat("acc X", accx);
  newRow.setFloat("acc y", accy);
  newRow.setFloat("acc Z", accz);
  */

  //update graph's x and y value arrays
  for (int i =0; i < temphipx.length - 1;
  i++){
     temphipx[i] = xhipVals[i+1];
     tempx[i] = xVals[i+1];
  }
  //append newest recorded data point
  temphipx[temphipx.length - 1] = xPoint;
  tempx[tempx.length - 1] =
  tempx[xVals.length - 2] + 1.0; //increase
  by 1
  xhipVals = temphipx;
  xVals = tempx;

  for (int i =0; i < temphipy.length - 1;
  i++){
     temphipy[i] = yhipVals[i+1];
     tempy[i] = yVals[i+1];
  }
  //append newest recorded data point
  temphipy[temphipy.length - 1] = yPoint;
  tempy[tempy.length - 1] =
  tempy[yVals.length - 2] + 1.0; //increase
  by 1
  yhipVals = temphipy;
  yVals = tempy;

  for (int i =0; i < temphipz.length - 1;
   i++){
     temphipz[i] = zhipVals[i+1];
     tempz[i] = zVals[i+1];
  }
  //append newest recorded data point
  temphipz[temphipz.length - 1] = zPoint;
  tempz[tempz.length - 1] =
  tempz[zVals.length - 2] + 1.0; //increase
  by 1
  zhipVals = temphipz;
  zVals = tempz;
}

//=========== DRAW RECTANGLES
// draws the rectangle and circle that
represents the hip's current angle relative
to the floor
//top-left cornerpoint (cx,cy)
//width of window
```

```
//height of window
//angle of rotation
void hipRect(float cx,float cy, float w,
float h, float ang, String label){
  fill(5,91,92);
  //draw the current data value in the top
  right corner
  textSize(h*0.1);
  text(label, cx+w*0.01, cy+h*0.1);
  if (portFound == false) {
    text("No values detected", cx+w*0.01,
    cy+h*0.2);
  }
  else {
    text(ang + "    ", cx+w*0.1, cy+h*0.2);}
  //center mode means can draw rectangle by
  referring to a center x and y coord instead
  of by coordinates of a corner
  rectMode(CENTER);
  float rl = h*0.9;   //rectangle length
  float rw = h*0.05;  //rectangle width
  //draw black circle at centerpoint
  fill(204,171,216);
  rect(cx+0.5*w, cy+0.5*h, rl, rw,5); //draw
  rectangle at default 0 angle position

  pushMatrix();
  translate(cx+0.5*w, cy+0.5*h); //shift
  everything by these coordinates (allows
  us to write 0,0 and have the shape in the
  'middle' of the window)
  //draw another circle cause why not
  fill(5,91,92);
  circle(0,0,rw*2);
  //draw rect at rotated position based on
  angle
  rotate(radians(ang));
  fill(132,116,161);
  //image(pic,-pic.height/4,-pic.width/4,
  pic.height/2,pic.width/2);
  rect(0, 0, rl,rw,5);
  popMatrix();
}

//=========== SETUP GRAPH
void setupGraph(XYChart graph){
  //setup for the graph
  textFont(createFont("Arial",10),10);
  // Axis formatting and labels.
  graph.showXAxis(true);
  graph.showYAxis(true);
  graph.setMinY(-90);

  graph.setYFormat("###,### deg.");
  // degrees
  graph.setXFormat("0000");
   // counter value

  // Symbol colours
  graph.setAxisColour(color(5,91,92));
  graph.setPointColour(color(5,91,92));
  graph.setLineColour(color(8,151,157));
  graph.setPointSize(5);
  graph.setLineWidth(2);
}

//=========== DRAW GRAPH
//graph object to be modified
//top-left cornerpoint (cx,cy)
//width of window
//height of window
//x-values
//y-values
void drawGraph(XYChart graph, float cx,
float cy, float w, float h, float[] xlist,
float[] ylist, String label){
  //refresh/update graph data values
  graph.setMinX(xlist[1]);
  graph.setMaxX(xlist[xlist.length - 1]);
  graph.setData(xlist,ylist);

  //Draw the graph
  fill(5,91,92);
  textSize(h*0.05);

  try {
    graph.draw(cx, cy+h*0.2, w-w*0.01, h-h*0.2);
  } catch (ArrayIndexOutOfBoundsException e){
      println("invalid");
    }

  // Draw title
  fill(5,91,92);
  textSize(h*0.125);
  text(label, cx+w*0.1, cy+h*0.2);
}
```