Operations and Assembly Manual

Smart Helmet

Omar Alomar

Fares Alotaibi

Mana Alyami

Race Oshiro

Titus Yazzie

ME 486C- Spring 2019

Project Sponsor: Northern Arizona University
Instructor: Dr. David Trevas
Client: Dr. Hesam Moghaddam

Table of Contents

Introduction

This manual explains how to assemble and operate, maintain, and troubleshooting the smart helmet. How to put together the smart helmet will be discussed in the assembly section. The operation section will discuss how to run the system of the smart helmet. In maintaining the smart helmet, this section explains how to make sure the system is working properly. In the troubleshooting section, the section will address the different problems.

Assembly and Operation

This section will explain the assembly and operation of the smart helmet system. The assembly of the system includes necessary parts for the system and how the parts are brought together. The operation of the system explains the proper way to use the system for its intended purpose.

Assembly

The first step in the assembly process is to acquire the necessary parts for the device. Table 1 includes a list of the parts needed for the smart helmet system. Once all the parts are acquired, the construction of the smart helmet can begin.

Table 1: Parts List

D3O Material
Laser Sensor VL53L0X
RTCSD-01
HC-05 (Bluetooth Arduino)
Accelerometer 3- Axis sensor 200g
Triple- Axis Gyroscope
Arduino Pro Micro
Right Angle Male Pin Header Connector
Male to Female Wire Jumper
Helmet

Installing D3O Material

The original padding of the helmet needs to be removed to allow space for the viscoelastic D3O Material to be placed in the helmet. The D3O Material is cut into strips to contour to the shape of the helmet. Some of the original padding was saved and reused in conjunction with the D3O material to add comfort. The last step for installing the padding is to cut the necessary holes to hold the sensors of the smart system in the helmet.

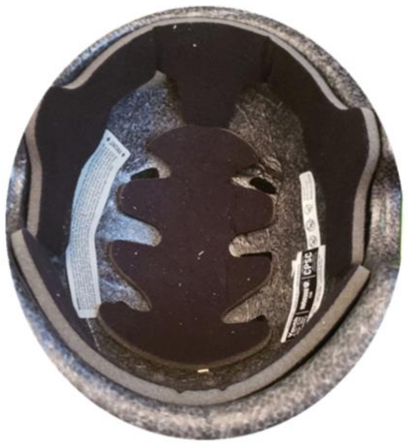


Figure 1: Helmet with Original Material

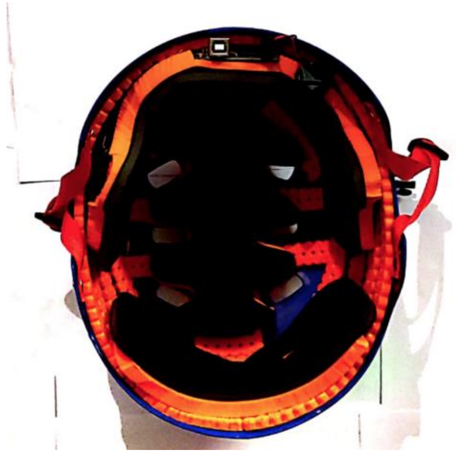


Figure 2: Helmet with D3O Material and Smart System

Constructing Smart System

The construction of the smart system will explain the process for combining the Arduino parts from the parts list above. The original pin header connectors need to be removed from all the sensors and boards to be replaced with right angle pin header connectors. To remove the old connectors a soldering iron is used to melt the solder, and a solder sucker is used to remove old solder. The right-angle pin connectors are then soldered to each board to help decrease the size of the system. Next the wires for each board are connected to those right-angle pins as shown in figure 3-7. Figure 8 shows the completed system excluding the laser sensor. The laser sensor can be connected following the pins of the gyroscope if needed for implementation for future work but was not included in the current system. The system is also powered by a nine-volt battery that was connected to the Uno board and an on/off switch was soldered to the battery connection.

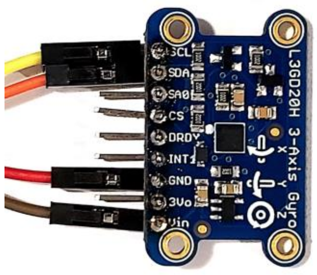


Figure 3: Triple- Axis Gyroscope

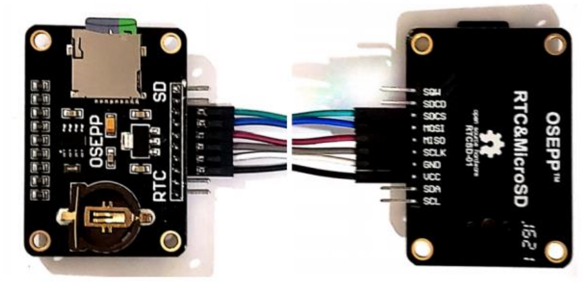


Figure 4: RTCSD-01

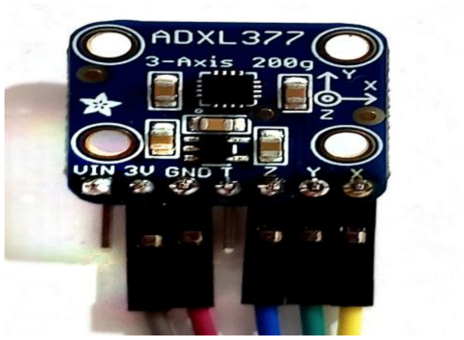


Figure 5: Triple-Axis Accelerometer (+-200g)

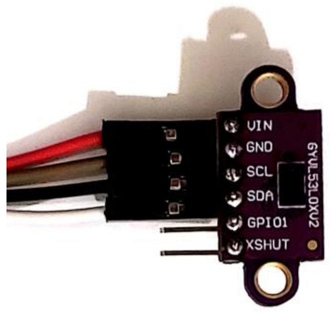


Figure 6: Laser Sensor VL53L0X

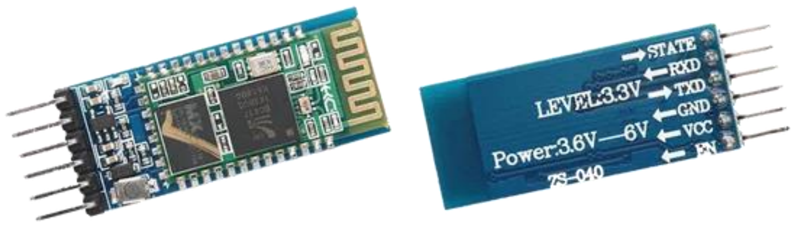


Figure 7: Bluetooth Transmitter

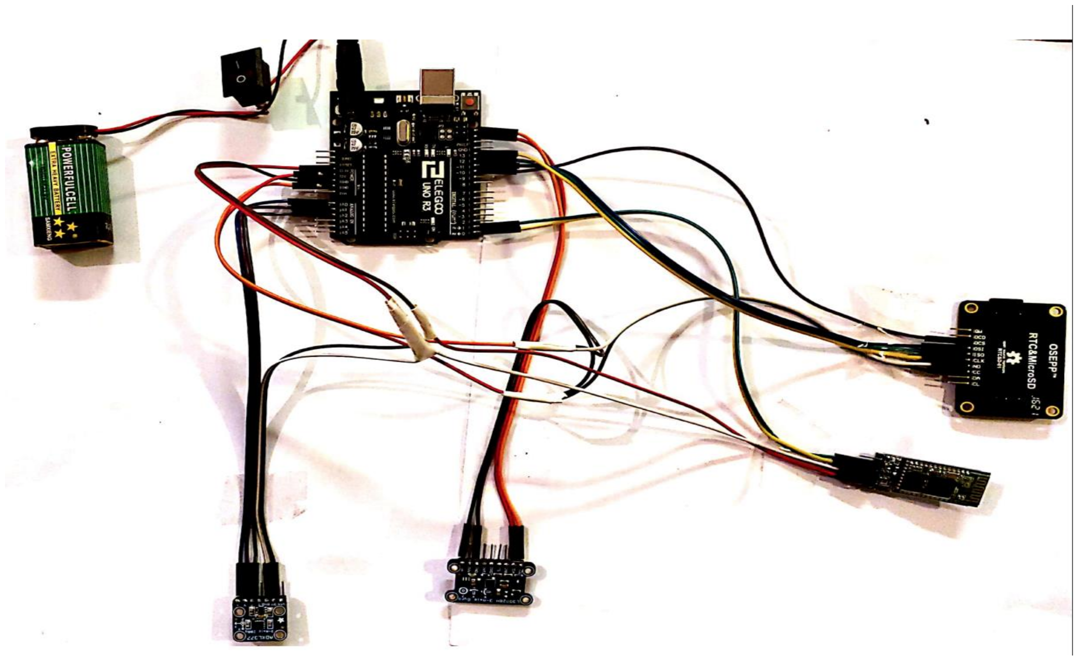
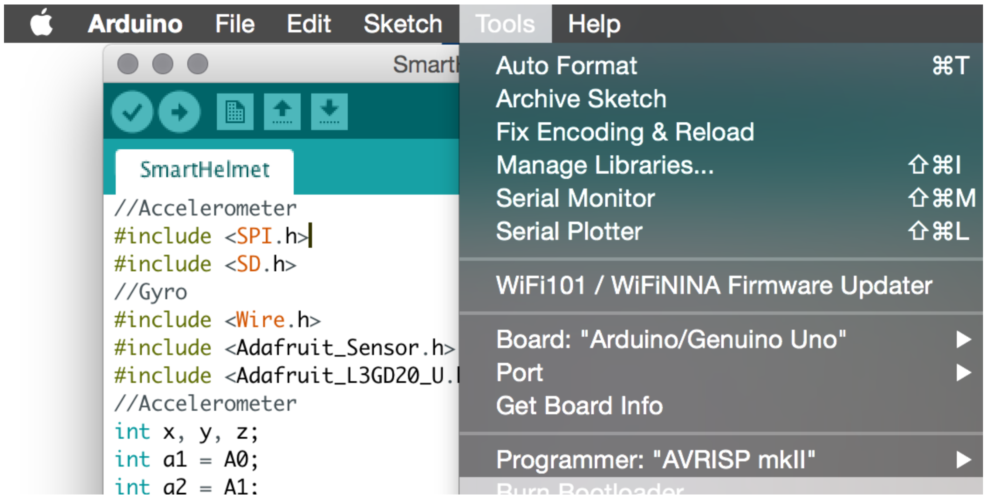


Figure 8: Complete System

Operation

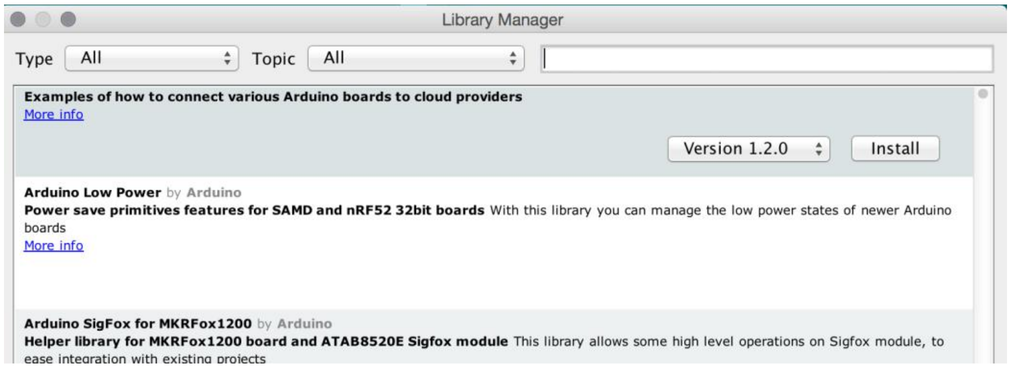
Arduino

Once the helmet and system are assembled, the smart helmet user is ready to operate the system. The first step is to download the Arduino application from the Arduino website. Then the user will have to install the necessary libraries to run the code. When Arduino is open, the user will need to go to "manage libraries" as shown in figure 9. Figure 10 below shows the library manager screen where the user will search for the necessary libraries in the top right corner of the library manager screen.



```
//Accelerometer
#include <SPI.h>
#include <SD.h>
//Gyro
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_L3GD20_U.
//Accelerometer
int x, y, z;
int a1 = A0;
int a2 = A1;
```

Figure 9: Managing Libraries



Examples of how to connect various Arduino boards to cloud providers
More info

Arduino Low Power by Arduino
Power save primitives features for SAMD and nRF52 32bit boards With this library you can manage the low power states of newer Arduino boards
More info

Arduino SigFox for MKRFox1200 by Arduino
Helper library for MKRFox1200 board and ATAB8520E Sigfox module This library allows some high level operations on Sigfox module, to ease integration with existing projects

Figure 10: Library Manager

Table 2: Necessary Libraries

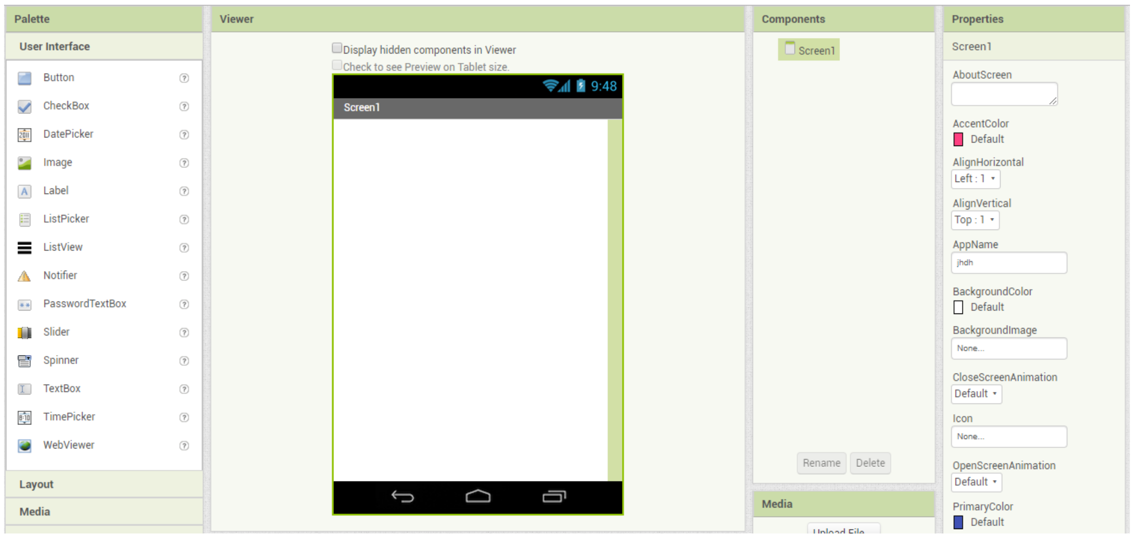
SD
Adafruit HX8340B
Adafruit L3GD20 U
Adafruit_Sensor_master
HC-05

Table 2 shows the necessary libraries needed to be installed. Once the necessary libraries are installed, the Arduino system needs to be connected to the computer. When connected to the computer, the Arduino board being used needs to be selected with the “Board:” tab in figure 9, and the “Port” tab in figure 9 needs to be selected to the “usb” port. Finally, the user will upload the code from Appendix A and click the “upload” button in Arduino. Once the code is uploaded to the system the user just needs to turn the system on with the switch to start recording linear and angular acceleration data to the microSD card.

Android Application and Bluetooth

To transmit the values from the Arduino to a Bluetooth connected device, the user must install the Smart Helmet application to an android device, “MIT AI2 Companion”. The app needs to be installed to an Android device from the Google Play Store.

Once the app is installed on a device, to create the app, go to the website, http://ai2.appinventor.mit.edu and create an account and start a new project. The screen will look like this in figure 11, below.



AboutScreen

AccentColor
Default

AlignHorizontal
Left : 1

AlignVertical
Top : 1

AppName
jhdh

BackgroundColor
Default

BackgroundImage
None...

CloseScreenAnimation
Default

Icon
None...

OpenScreenAnimation
Default

PrimaryColor
Default

Figure 11: Screenshot of New Project.

To create a similar layout as the Smart Helmet's app, these steps should be followed:

1. Click on the Layout button on the right hand side of the screen and then click and drag the HorizontalArrangement button onto the Viewer screen.
    a. In the Properties menu, make the background color yellow and then go to height to make the height of 150 pixels. Then go to width and click on the Fill parent.
2. Go to the User Interface menu and click and drag ListPicker into the HorizontalArrangement.
    a. In Properties, make the height and width to 150 pixels each and insert picture by clicking on Image, this could be a simple bluetooth symbol (the Smart Helmet team created their own logo and used this as a logo).
3. Go back to the User Interface menu and drag a label button into the same HorizontalArrangement component.
    a. In Properties, make the background red, the font size 20, height at 30 pixels and width as fill parent, and put the text as "CONNECT FIRST".
4. In User Interface menu, drag another label beneath the first HorizontalArrangement.
    a. In Properties, make the height as 25 pixels and width as fill parent, then change the text to "Reading:", and the text alignment as center.
5. In User Interface menu, drag a second HorizontalArrangement below the label.
    a. Put three labels into the HorizontalArrangement, from right to left. Have each label have their own text as "Value", "0.000", and "G-Forces" from right to left.
    b. Have the font of 20.0 for each label and height and width as automatic.
    c. For the "0.000" label, change the textcolor to red.
6. In Palette, go the the menu and click on Connectivity.
    a. Drag the BluetoothClient onto the Viewer screen. There BluetoothClient will show up on the bottom.
7. In Palette, click on the Sensors menu and drag a clock onto the Viewer screen.

Following these steps should have the Viewer and Components screen look like figure 12, down below.

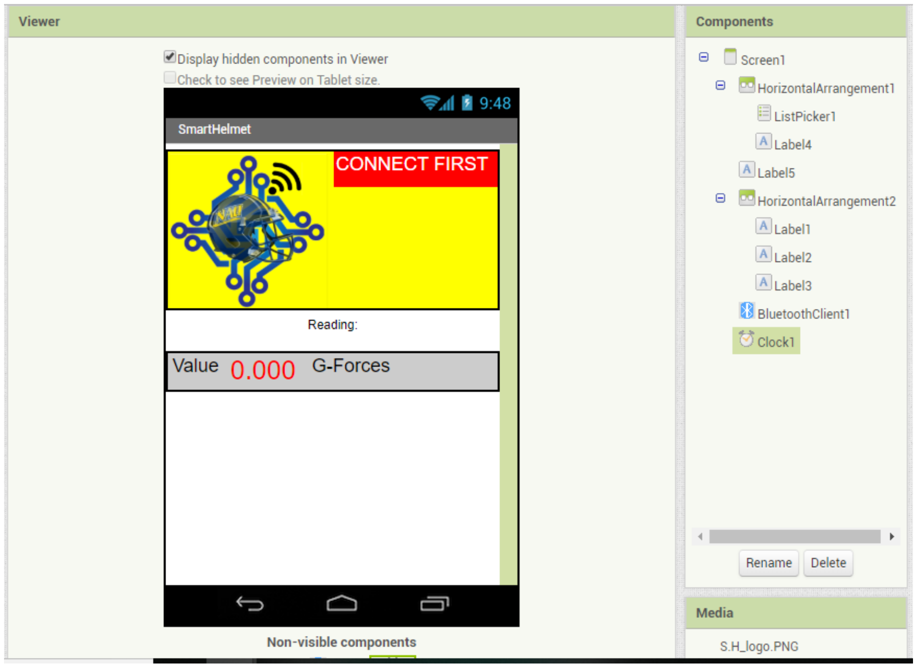


Figure 12: Complete App Layout

Next, go to the "blocks" that is near the top left screen, between the Properties and user sign in. This will take the user onto a new screen as seen in figure 13. This section will show the steps to have the app connect to the system and send data to the device.

.

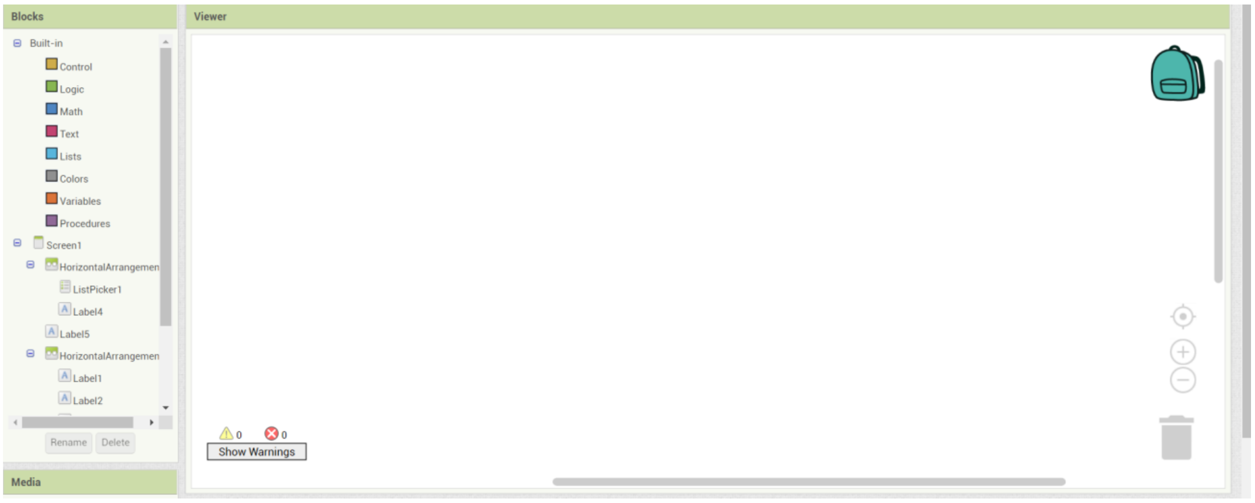
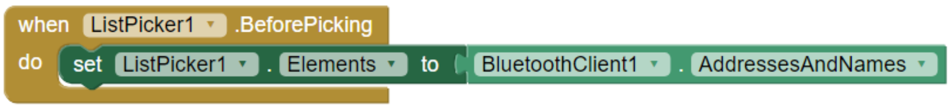


Figure 13: New Blocks Page.

1. In the Blocks menu, scroll down to find and click on the ListPicker. A sidebar will appear.

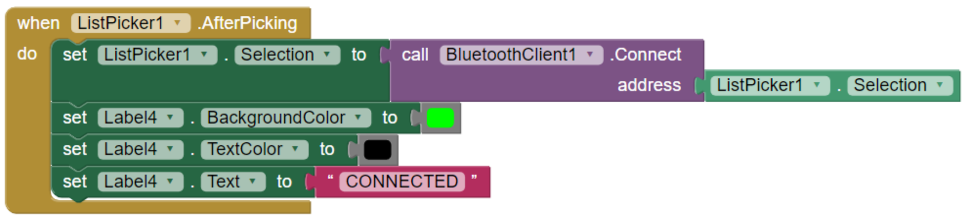
a. Once the sidebar appears, look and click on the block that says "when ListPicker.BeforePicking". Drag the block into the Viewer screen.
b. Click the ListPicker again then click and drag the block "set ListPicker.Elements" and connect the block to the first block right next to "do".
c. In the Block menu, go to "BluetoothClient" and a sidebar will appear. Locate and drag the block "BluetoothClient .AddressesAndNames" and connect to the previous block.
d. The first block should look like this:



```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames
```

Figure 14: First Block View

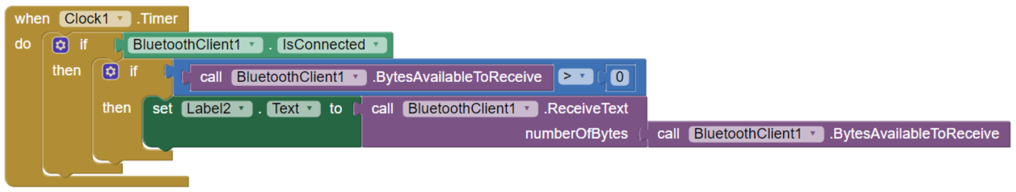
2. Start a new block by clicking on ListPicker then select and drag the block "when ListPicker.AfterPicking" onto the Viewer screen.
   a. Again, click on the ListPicker and search for the block "set ListPicker.Selection to" and connect the block next to "do" in the main block.
   b. Look for the BluetoothClient and click and find the block "BluetoothClient.Connect address" and connect next to the previous step block.
   c. Go back to ListPicker and search for "ListPicker.Selection" and connect to the block.
   d. In the Blocks menu, locate the Label that is beneath the ListPicker. Click and find the block "Label.BackgroundColor" then connect under the "ListPicker.Selection to".
   e. In the Blocks, under the Built-In menu, find the Colors function and click and find the color green block and connect to "Label.BackgroundColor".
   f. In the same Label as chosen previously, look for the block "set Label.TextColor to" and connect under step e.
   g. Set the color text to black by following the similar step as e.
   h. Again, go back to the same Label and look for the block "Label.Text to" and connect under the last block.
   i. Look for the Text function in Built-In and locate the block that has " " and insert the text 'CONNECTED' into the quotations.
   j. The entire block should look like this:



```
when ListPicker1 .AfterPicking
do  set ListPicker1 . Selection to call BluetoothClient1 .Connect
                                                   address ListPicker1 . Selection
    set Label4 . BackgroundColor to 🟩
    set Label4 . TextColor to ⬛
    set Label4 . Text to " CONNECTED "
```

Figure 15: Second Block Completed

3. Start a new block and in Blocks menu, find and click on the Clock function and drag the block “when Clock.Timer do” onto the Viewer screen.
    a. In Built-In section, go to the Control function and drag the “if then” block and connect the block to the new main block.
    b. Go to BluetoothClient function and find the block “BluetoothClient.IsConnected” and attach block next to the step a.
    c. Insert another “If then” block and connect that block to the first “if then” block.
    d. In Built-In menu, go the the Math function and drag the second block from the top next to the “if” in the “if then” block. (There should be an open space, “=” drop down menu, and another open space)
    e. Go to the BluetoothClient function and drag the block “call BluetoothClient.BytesAvailableToReceive” into the first open space in the Math block.
    f. Go to the “=” sign and there is a drop down menu, click the menu and change the sign to “>”.
    g. In the Math function, select the “0” block and leave the value to zero and insert to the second open space in the Math block.
    h. In the second “if then” block, find the label that corresponds to the “0.000” in the Designer view.
    i. When locating the correct label, click on the corresponding Label and find the block “Label.Text to” and connect to the “then” statement.
    j. Go to the BluetoothClient function, find the block “call BluetoothClient.ReceiveText numberOfBytes” and connect next to the previous block.
    k. Lastly, go back to the BluetoothClient function and look for the block “call BluetoothClient.BytesAvailableToReceive” and connect next the the block in the previous step.
    l. The last main block should look this:



```
when Clock1 .Timer
do  if  BluetoothClient1 . IsConnected
    then  if  call BluetoothClient1 .BytesAvailableToReceive > 0
          then  set Label2 . Text to  call BluetoothClient1 .ReceiveText
                                                  numberOfBytes  call BluetoothClient1 .BytesAvailableToReceive
```

Figure 16: Last Main Block Completed

The app should be able to work. To connect an Android device to the app, simply follow these next steps:

1. In the top drop-down menus, look for the “Connect” menu and select “AI Companion”.
2. Open the app in the chosen Android device and either connect the app by a code given or by the barcode. Either option is the same.
3. The device will sync with website app.

4. Turn on the smart helmet system and connect to the bluetooth “HC 05”.
5. The app will let the user know if the bluetooth is connected.
6. Begin testing the smart helmet and values from the linear accelerometer will show on the Android device.

Troubleshooting

Troubleshooting the smart helmet explains how to make sure the system continues to work properly. There are two people who will get the benefits for this smart helmet. One is the user and the second one is the one who observes the player. The user is whoever will use the helmet and run the system of the helmet. However, the observer is the one who is responsible for the safety of the user. Thus, this smart helmet cannot be used without these important members. There are many processes of troubleshooting the smart helmet, here are some steps to check if things are not working as intended.

Wiring the system is one of the troubleshooting processes for the smart helmet. If a sensor isn’t working as intended, the first thing to check is the wiring. All wires should be connected at both ends, sensors should follow the wiring shown in figure 3-7. Some steps to consider assuring the system works as expected are followed:

1. Verify all libraries are installed for the appropriate sensors.
2. Make sure to have all the required wires and parts ready before setting them up.
3. Insure that all the wires are connected and are connected between the correct pins.
4. For the SD, the user should verify the chip select number in the Arduino code matches the digital pin connected to the chip select pin on the SD.
5. The user should run the codes between changes to see if troubleshooting worked.
6. Read and understand the output codes.
7. Make sure the switch is turned on and that the battery is charged.
8. Clean system to prevent dirt and grime from disrupting the connections.

SD Card troubleshooting

The SD card is one of the most important transmitting data sensors in smart helmet. The major issue about any SD card in the world is to make sure the card is working properly. Formatting and coding are two major parts of making sure the SD card is working correctly. When formatting the SD card, then the user should make sure to format of the card is FAT16 or FAT32. Next, after formatting the SD card, the user needs to run the codes. In order to check if the card is ready to use or not, then the user needs to open the Cardinfo code. This Cardinfo codes helps in justifying whether the card is ready to use or not. The Cardinfor code is in the examples section of Arduino and is in the SD section. In order to test the card, the user should initialize the card first. As soon as a massage of Initialization appears in the serial monitor, then the card is ready to use. Next step is to Read/Write the codes. In order to do that the user should download the Read/Write codes from the Arduino library. Once this code installed, then the user can check the reading and writing anything to the SD card.

Bluetooth Troubleshooting

The transmission of data from the helmet to the receiver is an important part of the project. When the app is not corresponding correctly, check the connection from the app to the website. Either by restarting the app by going to the “Connect” drop down menu and press “Reset Connection”. The user will have to restart the app on the Android device by exiting the app and reconnect the app and website again.

Appendix A: Smart Helmet Code

```
//Accelerometer
#include <SPI.h>
#include <SD.h>
//Gyro
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_L3GD20_U.h>
//Accelerometer
int x, y, z;
int a1 = A0;
int a2 = A1;
int a3 = A2;
//Gyro
Adafruit_L3GD20_Unified gyro = Adafruit_L3GD20_Unified(20);
#define SENSORS_DPS_TO_RADS

void setup() {
  Serial.begin(115200);
  //SD
  while (!Serial)
  {
    ;
  }
  Serial.println("Initializing SD card...");
  if (!SD.begin(10))
  {
    Serial.println("Card failed, or not present");
    while (1);
  }
  Serial.println("Initialization done.");
  //Gyro
  gyro.enableAutoRange(true);
   Serial.println("Gyroscope Test"); Serial.println("");
  if(!gyro.begin())
  {
    Serial.println("Ooops, no L3GD20 detected ... Check your wiring!");
    while(1);
    }
}

void loop() {
   //Accelerometer
  double gx, gy, gz, mag;
  String dataString = "";

  x = analogRead(a1);
  y = analogRead(a2);
  z = analogRead(a3);
  gx = (double)(x-330)/1.65;
  gy = (double)(y-330)/1.65;
  gz = (double)(z-330)/1.65;
  mag = sqrt(gx*gx+gy*gy+gz*gz);

  dataString += String(gx);
  dataString += ",";
  dataString += String(gy);
  dataString += ",";
  dataString += String(gz);
  dataString += ",";
  dataString += String(mag);
  dataString += ",";

  //Gyro
  sensors_event_t event;
  gyro.getEvent(&event);
  dataString += String(event.gyro.x);
  dataString += ",";
  dataString += String(event.gyro.y);
  dataString += ",";
  dataString += String(event.gyro.z);
  dataString += ",";
  Serial.println(dataString);

   //SD
  File myFile = SD.open("Rec.csv", FILE_WRITE);
  if (myFile)
  {
  myFile.println(dataString);
  myFile.close();
  }
}
```