IBEHS 3P04 Final Report

Yousam Asham

ashamy1

400174082

April 25th, 2021

Abstract

The design challenge chosen was “A Balancing Act After ACL Surgery”. This was mainly due to YESS’s (our team’s name) recognition for an increased demand for ACL reconstruction rehab devices [1]. The challenge was to design and prototype a device which would allow users to independently monitor post-ACL reconstruction surgery balance at home. The target audience would be young adolescent individuals who intend to return to sport (RTS). The SA(c)LT was developed to solve this challenge. SA(c)LT utilizes the effectiveness of the hop test and provides quantitative and qualitative results using Limb Symmetry Indices (LSI) to the user through an easy-to-use user interface.

Introduction and Background Theory

The clinical problem that was mainly addressed throughout this project was the patient’s balance post-ACL reconstruction surgery. Since the ACL is an important stabilizer of the knee joint, many sports player struggle to stand on the injured foot. However, studies conclude that with proper training, this balance can be restored to injured patients with the appropriate training and rehabilitation program in place [2].

Due to this injury, injured athletes find it hard to recognize when it is safe to RTS. This decision of when to RTS must not be rushed and should be based on evidence since the risk of reinjury is significant (37.1%) [3]. This percentage is high and dangerous especially considering that these types of repeated injuries each take their own toll on the injured leg. This means that repeated reinjury could end an athlete’s career.

Therefore, a wise decision must be made to indicate when it is appropriate for the athlete to return to sport. This involves measuring rehabilitation quantitatively, which is exactly what the Limb Symmetry Index (LSI) aims to provide. LSIs are a method of quantitatively representing the performance of the injured leg in comparison to the non-injured leg and is a useful tool that is used in rehabilitation clinics to assess rehab progress. In fact, LSI scores improvement have been linked to absolute increases in the injured leg’s ability as well as balance [4].

There are many types of tests to include in ACL rehabilitation programs. One of the late-stage rehabilitation tests that is performed is called the Single Leg Hop Test (SLHT), or hop test in short. The SLHT has been recognized as one of the most helpful tests in the rehab process’ late stages [4]. This was communicated to YESS through different means such as Expert Design Review with Dr. Turak and Mr. Hollingham. The SLHT is known to be a reliable and valid performance-based outcome measure for patients undergoing rehab [5].

With all of this in mind, YESS decided to develop and prototype the SA(c)LT (Socks for ACL Tears). This device would be easy-to-use, non-disruptive, user friendly, portable, complete, and to be used by only one person without the need of help from anyone. Given the circumstances of COVID-19, YESS had to develop this protype using limited, and budget friendly equipment. This constraint affected the prototype, but YESS placed a detailed plan in place regarding the detailed parts which will be used (including a bill of materials). Another constraint that our team faced was regarding the fact that the user would be highly active while using our device. This would mean that our device would be constrained to comfortable materials. This would also mean that our device would have a natural feel to it as to not affect the results of the SLHT while it is being worn.

Methods

The hardware of the SA(c)LT did not change much throughout the iterations. It consists of an ESP32 microcontroller, a BNO055 accelerometer and gyroscope, and force sensitive resistors (FSR). In the first iteration, our device would be mounted onto a single sock. This would provide us with a device that is comfortable and would feel natural to the user. The sock would be worn on the injured leg's foot. Refer to Appendix E for design iteration sketches.

For the first iteration, the accelerometer was to be placed along the distal end of the Achilles' tendon, and the two FSRs would go on either side along the medial and lateral aspects of the foot. The plan was for the FSRs to detect forces on either sides of the foot and for the accelerometer to determine the linear acceleration values along all three axes.

For the second iteration, three major changes were made. The first was the introduction to another sock, now making them a pair of socks. The second change was reducing the number of FSRs used to only one. The final change would be the use of the FSR as a binary indicator which communicates to the program the status of the hop test. The first sock would house the FSR at the sole, while the other sock would hold the BNO055 at the distal part of the Achilles' tendon. The placement of the BNO055 there would be strategic for measuring the foot's orientation while performing the hop test since it would allow the user to compare the operative versus the non-operative foot orientations. The sock with the FSR would be for the uninjured foot, while the other sock with the BNO055 would be worn on the injured foot. Moreover, a computer program flowchart was developed (See Appendix C).

The fine details were finalized in the third and final iteration. Some changes include the placement of the ESP32 at the lower aspect of the user's back, and the placement of the wires connecting the BNO055 and the FSR. To fulfill the easy-to-use, user friendly and complete design objectives, it was decided that timing of the SLHT would be fully automated. It was also decided that the BNO055 would be attached to the socks using clips. Their method of attachment would be like a hair pin (see Appendix B). This would ensure that the BNO055 is in the correct orientation to avoid inaccuracies in the reading of the angles. The FSR would be glued onto the sole of the sock as the pressure of the foot onto it would keep it in place.

The SA(c)LT uses an ESP32 microcontroller. This handles all the computing, storing, and logic of the wearable device. To record an attempt, the user needs to perform two hop tests, one while hopping with the injured leg, the other with the user hopping with the uninjured leg. The SA(c)LT reads the leg's orientation at a frequency of 2 Hz (twice per second or once every half second) and records the orientation of all three axes (X, Y, and Z) in three integer arrays. These would be used to calculate the average angles of the foot while the user is performing the hop test. Since the attempt requires the user to do another hop test with the uninjured foot, the average angles of the uninjured foot will be the point of comparison for the user to assess their own rehabilitation progress along the process. This fulfills the device's design objective of being independent since the user could independently track their rehabilitation progress.

Most of the SA(c)LT's interactions would be through Bluetooth. This enables the user to use their phone or computer to interact with the device. This fulfills the design objective of being user friendly since the user does not need to learn or get accustomed to dealing with a different interfacing

device, assuming the user has experience with using a phone or computer (which is a reasonable assumption, given that the target end user is a young adult).

Testing was carried out to calibrate and validate the signals that the sensors pick up. This is because YESS had to ensure that the sensors were reliable, given that the budget was more towards the low end due to previously mentioned constraints. For calibration, specific methods from the BNO055 API were used to ensure that the sensor can successfully detect gravitational acceleration at around +/- 9.81 m/s^2. The getVector() method was used to calibrate the BNO055 and to ensure that the acceleration being read is accurate with minimal error. This was be repeated for all three axes. Similarly, the BNO055 was calibrated by ensuring that the orientation outputs are accurate. This was done using the getOrientation() method present in the API. To calibrate, the BNO055 was placed on a flat, even surface against a wall and a ruler was used to ensure that the BNO055 is in a straight orientation. The orientation would then be obtained through the API and the results obtained were compared to the expected results. This is to be done for all three axes as well.

The FSR also required calibration since it is an important component of the device which helped with timing. For calibration of the FSR, simple objects of known weight (~20 N) were placed on the FSR to see if the weight threshold needed calibration to account for the error in the FSR. However, the weight threshold was not changed.

To validate the accuracy of our sensors, preliminary tests were put in place first to check if the sensors could detect motion, orientation, and force. The sensors were placed onto the correct locations on the prototype socks and hop test movements were simulated to validate the outputs from the BNO055 and the FSR. Similarly, the device's calculation of the LSI was validated to make sure that it was being calculated in a proper manner. This was done by allowing an uninjured user to test the SA(c)LT. An LSI greater than 90% would be considered a recovered leg [6].

Results

Firstly, the physical prototype was developed. Crew socks were used, and the sensors were represented by hard, rigid cardboard pieces. Cardboard was used to allow YESS to mimic the weight and feeling of the sensors on the foot during the performance of the hop test. The location selected for the BNO055 was not interfering with hop test motion and therefore this was not much of a concern.

YESS was also developing the computing program prototype simultaneously (see Appendix C for Analysis Code). One of the many unique features of the SA(c)LT is its auto-timing feature. There were many ways of implementing this into the device's program. One of which includes the importing of an external library which allows for the usage of an NTP client. The ESP32 would connect to this client's server through Wi-Fi. Through its API, this client can obtain live date and time data. This was the original plan for the computing prototype.

It was also previously decided that Bluetooth technology would be used to facilitate an easy user experience. The ESP32 comes with a Bluetooth antenna and so this would not be a hard feature to implement in our design. It would be connected to a Bluetooth serial terminal (either on the user's phone or computer) and the user would interact with the device from there.

To begin recording a hop test attempt, the user would need to simply enter 's' into the Bluetooth serial terminal input line to let the SA(c)LT know that the injured leg hop test is ready to start.

The SA(c)LT then lets the user know that it is ready for recording and that it will begin recording as soon as they lift their other foot off the ground and start hopping. When the acceleration is detected to go past a certain threshold, the SA(c)LT begins recording foot orientations in all three axes and start the timer. As soon as the user puts their other foot down, the timer is halted. The end of the timer would mean one of two things. It could mean that the user concluded a successful hop test, or it could mean that they lost their balance and had to put their other foot down. For that reason, the SA(c)LT would ask the user whether this hop test was a success or a pass. If the hop test was a successful one, then the user would enter a 'p', otherwise they may enter anything else to indicate a failed hop test. Only successful hop tests are records. If the hop test was a failed hop test, they are given the option to retry the failed hop test. Once the successful hop test is scored, this process repeats again to ask a user to record another hop test, this time for the uninjured leg. After both hop tests are successfully completed and recorded, the attempt is recorded. In the process of recording the attempt (attempt is made up of an injured and uninjured leg hop test), the LSI is calculated.

Following the storing of the attempt and the calculation of its LSI, the results are displayed to the user through the Bluetooth serial terminal. There are three forms of results displayed: LSI score, LSI average, and the average injured and uninjured foot orientations/angles. The LSI score is the current attempt's LSI displayed to the user so they can see how this attempt matches with the previous attempt (comparison to previous attempt is omitted if this is the user's first attempt). The LSI average is a cumulative average of all the user's previous attempt's LSI scores. This gives the user a general idea about how their rehab process is progressing/worsening. Finally, the average angles output gives the user an indication about their balance and how it their injured leg compares to their uninjured leg. The LSI-oriented score gives the user an idea about their progress towards returning to sport while the average angles give the user an opportunity to consciously think about their balance and how it is affected by the injury. Usually, these balancing issues go unnoticed as the body tends to rely on other joints to provide balance, other than the knee joint (e.g., hips, ankles, etc.). The average angles output could bring these issues to the forefront and allow the user to think about ways of improving their balancing form.

Discussion

This design specifically answers to the design challenge provided since it accomplishes the two tasks defined in the design challenge: aids in restoring post-ACL reconstruction surgery balance and provides injured athletes a benchmark to determine when it is safe to RTS. These two tasks are simply fulfilled by understanding the output results. The LSI-oriented results give the user an idea about how the late-stage rehabilitation is progressing. A progress in LSI is indicative of a safer RTS. The average orientation of the injured and uninjured legs may bring attention to the balancing issues that could be evident due to the reconstruction surgery.

In term of economic viability, the SA(c)LT is economically viable. The cost of producing a unit would be around $80 (CAD). Other indirect competitors on the market right now are significantly more expensive. For comparison, a CPM machine would cost around $700 (CAD). A chart view and indented view WBS was created and evaluated, as well as an AON representation of the CPM and a Bill of Materials (see Appendix D). These representations and structure additionally support the financial viability of the SA(c)LT.

Current market research shows that no direct competitors are currently on the market. The current developed prototype is very user friendly, accurate, and is almost ready for manufacturing. For these reasons, YESS believes that the SA(c)LT is feasible. Due to current limitations, there are three main future refinements which would better the SA(c)LT. Limitations involving time, resources, and in-person presence limited YESS' design space. However, given more time and resources YESS would like to focus on three major refinements: implementation of a backend database where all attempts (passed and failed) would be stored, integration of a more user-friendly visual output, conducting trials to determine optimal lower leg angles, and the use of a microcontroller with better connectivity and specifications than the ESP32.

A backend database would be beneficial to the user since it allows them to assess their long-term rehab progress. Since a database would provide us with more memory capacity than the memory currently available on the ESP32 (which is where the data is being stored currently), this would give the SA(c)LT the ability to also store the failed and pass attempts. This would unlock potential of providing even more useful output data such as hop test success percentage.

The implementation and integration of a user-friendly visual output such as LED lights or an OLED display would improve the overall user experience. Moreover, the SA(c)LT would be used without depending on laptop or a phone with Bluetooth capabilities. YESS strives for improvement involving our design objectives, this future refinement would allow our device to be easy-to-use.

YESS found that there was a lack of academic resources outlining healthy lower leg angles during performing hop tests. For that reason, one of the future refinements would be conducting trials involving healthy individuals performing hop tests. This would allow the SA(c)LT to compare the average angles obtained to healthy angles determined through the trials, and not only compared to uninjured leg average angles.

YESS would also like to incorporate a better microcontroller to carry out the computational logic of the SA(c)LT. A better microcontroller would have better memory capabilities. Although the device would still be feasible with the ESP32, a better microcontroller would allow for the use of Wi-Fi and Bluetooth simultaneously. Currently, the ESP32 allowed only one of two functionalities. This is because of the Wi-Fi and the Bluetooth using the same antenna. For that reason, the team could not follow up with the previously mentioned plan of using an NTP client to get in contact with a server to request start and stop timestamps for accurate timing. Instead, YESS had to resort to calculate the time taken to complete the hop test by using the frequency that the orientation measurements are taken at, and the number of angle measurements taken during a hop test.

Conclusions

Through the SA(c)LT, YESS can provide a solution to athletes who have had an ACL reconstruction surgery and are planning to RTS. The SA(c)LT provides results that are indicative of LSI which is a great indication of ACL reconstruction rehab. The SA(c)LT also provides information about lower leg orientations to bring balancing issues to the attention of the user. Reliable results coupled with low cost (relative to indirect competitor products), comfort, and ease-of-use provides the SA(c)LT with a great initiative to break through the market.

Reflections

I found the design process in IBEHS 3P04 to be great. Every design studio was oriented towards a different milestone and this allowed my team and I to focus on one thing at a time. This avoided any confusion regarding our goals along the process and no one on the team was left behind, the team was together on the same page. As a team, we took the collaborative approach to designing our solution. This incorporated brainstorming as many ideas as we could in the beginning then we would filter out which ideas would be feasible, realistic, attainable, and work with our current design. Throughout the process, we tried to maximize our utilization of the feedback given from Mr. Hollingham and Dr. Turak. Their feedback helped guide us towards our final design.

I contributed to the team throughout the planning stages by adding ideas and helping filter the ideas that were feasible and helpful. I also contributed by working on the wiring and coding the computing prototype. Being the only person on my team who is studying software engineering, I was very excited to jump into the coding and testing of the computing prototype. I planned out and executed the computing prototype while simultaneously referring to my team whenever any sort of limitations would arise (such as memory and performance limitations of the ESP32). See Appendix A for team member roles, responsibilities, and Gantt Chart.

As a software engineering student, I had to consider the program's general layout and coding structure. Planning out the program structure was something I have previously done numerous times in previous software courses. Getting to plan out the general structure of the program still had many aspects that were like what I was taught in software engineering. The creation of classes in Arduino was something I explored by myself as I was not taught how to code in Arduino. However, because I was proficient in other coding languages, I took an approach to compare between Arduino code and another language's code (Java) to learn Arduino (which is C++). This helped me learn Arduino efficiently and apply lots of what I was taught in other languages into my code. I also had to be aware of our program's efficiency and performance given that the ESP32 has reduced memory capacity. Through my understanding of the C programming language, I optimized the performance of our program.

For future refinements related to my discipline, my team would like to integrate the usage of a web database. The microcontroller would connect to Wi-Fi and push the user's results to this database. This would improve the program's performance since it uses significantly less memory on the microcontroller. However, this would come at a cost of upgrading to a microcontroller which could simultaneously use Bluetooth and Wi-Fi technology, now that both functionalities will be needed.

References

[1] J. T. Cavanaugh and M. Powers, "ACL Rehabilitation Progression: Where Are We Now?," *Current reviews in musculoskeletal medicine*, Sep-2017. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5577427/. [Accessed: 16-Apr-2021].

[2] "Recovering Your Balance after ACL Surgery," *eOrthopod.com*. [Online]. Available: https://eorthopod.com/news/recovering-your-balance-after-acl-surgery/. [Accessed: 16-Apr-2021].

[3] Kamath GV;Murphy T;Creighton RA;Viradia N;Taft TN;Spang JT; "Anterior Cruciate Ligament Injury, Return to Play, and Reinjury in the Elite Collegiate Athlete: Analysis of an NCAA Division I Cohort," *The American journal of sports medicine*. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/24981340/. [Accessed: 16-Apr-2021].

[4] E. Rohman, J. T. Steubs, and M. Tompkins, "Changes in Involved and Uninvolved Limb Function During Rehabilitation After Anterior Cruciate Ligament Reconstruction: Implications for Limb Symmetry Index Measures - Eric Rohman, J. Tyler Steubs, Marc Tompkins, 2015," *SAGE Journals*. [Online]. Available: https://journals.sagepub.com/doi/abs/10.1177/0363546515576127. [Accessed: 16-Apr-2021].

[5] Reid A;Birmingham TB;Stratford PW;Alcock GK;Giffin JR; "Hop testing provides a reliable and valid outcome measure during rehabilitation after anterior cruciate ligament reconstruction," *Physical therapy*. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/17311886/. [Accessed: 16-Apr-2021].

[6] S. Ryan, "Single Limb Hop Tests," *AbilityLab*. [Online]. Available: https://www.sralab.org/rehabilitation-measures/single-limb-hop-tests. [Accessed: 19-Apr-2021].

Appendix A: Team Member Roles, Responsibilities, and Gantt Chart

Individual Responsibilities

Emnpreet	Stiv	Sukhvir	Yousam
• Creative Lead • Editor • Submitter	• Design Lead • Editor	• Meeting Notetaker • Editor • Moderator/Facilitator	• Technical Lead • Editor

Meeting Notetaker → Keeps notes during meetings.

Submitter → Submits all team assignments, preferably by midnight the night before a DS.

Leader for Different Branches:

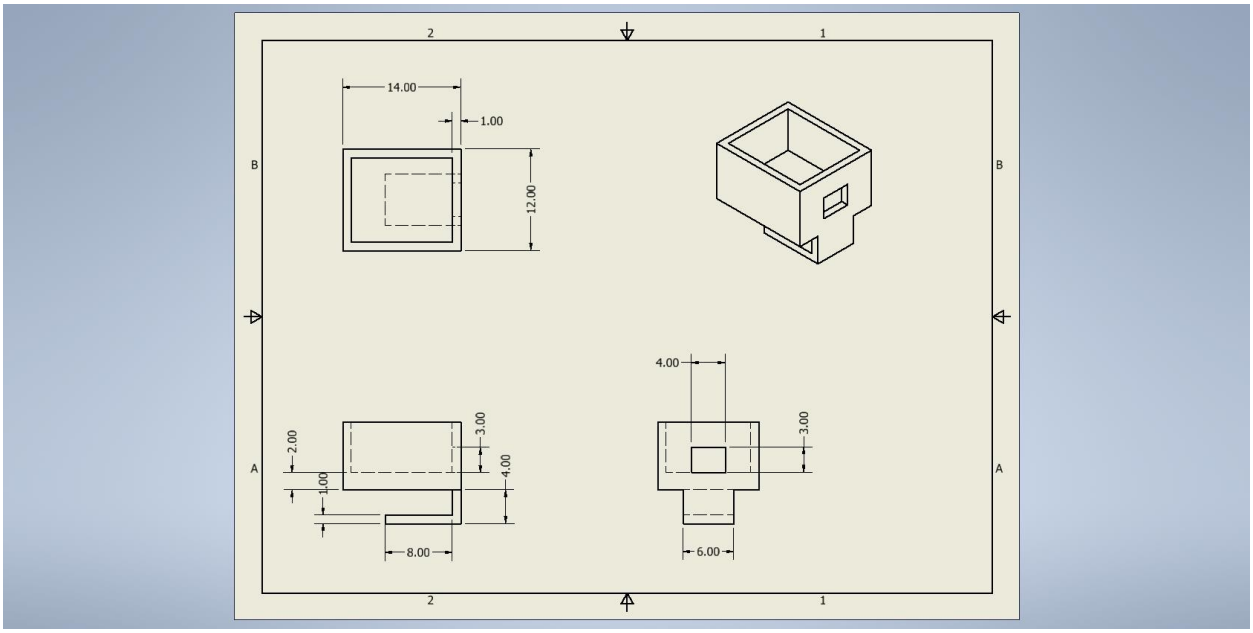
Creative Lead → Finalizes design elements, oversees the creation of art elements, ensures the final product matches the original design concept.
Design Lead → Responsible for maintaining the quality and delivery of the product.
Technical Lead → Oversees software development of the project.

Moderator/Facilitator → Facilitates and reminds others of team meetings/deadlines.

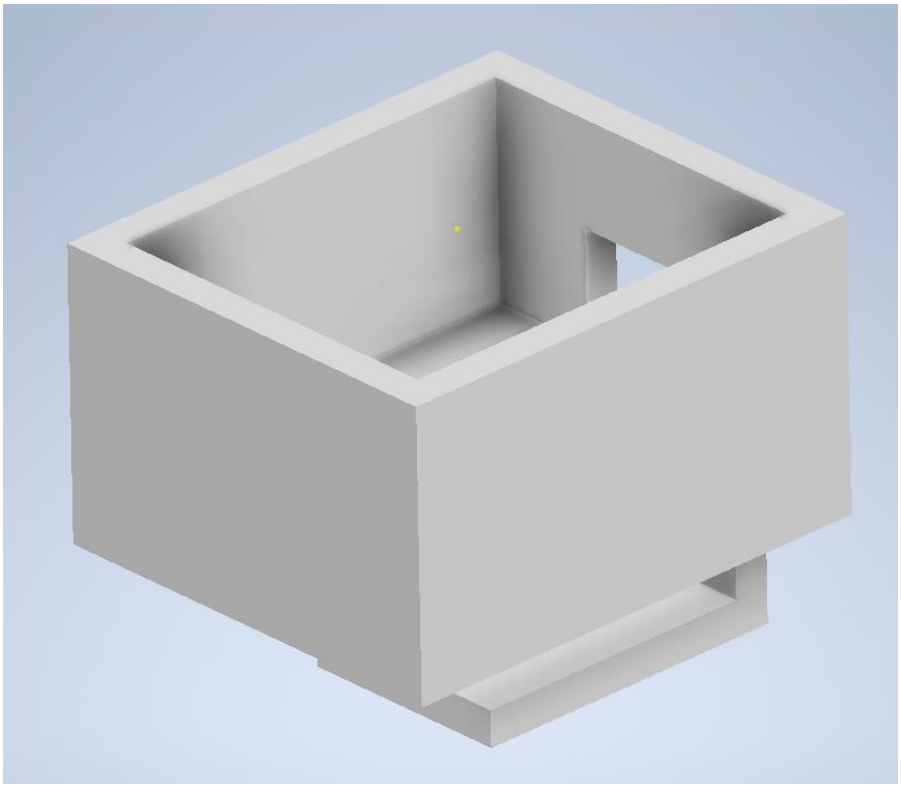
Editor → Works to edit and finalize documents for milestones and final deliverables.

Click here to view the final project Gantt Chart. (Access given only to McMaster Emails).

Appendix B: CAD Drawings



Engineering technical drawing for BNO055 clip attachment.



CAD Model for BNO055 clip attachment.

Appendix C: Analysis Code and Programming Flowchart

```
#include "BluetoothSerial.h" //Includes the header file that will be used to connect to the bluetooth/wireless terminal.
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>
//Blutooth object to connect to phone
BluetoothSerial SerialBT;

//FSR variables
int fsrPin = 37;     // the FSR and 10K pulldown are connected to pin 37.
int fsrReading;     // the analog reading from the FSR resistor divider.

//Accelerometer variables
Adafruit_BNO055 bno = Adafruit_BNO055(55);

//Local HopTest variables
int attemptNumber = 0;
int Xorientation[120];
int Yorientation[120];
int Zorientation[120];
int arrayCounter = 0;
float timeTakenLocal;
float prevLSI;
float avgLSI = 0;
float hopTestAvgXAngle;
float hopTestAvgYAngle;
float hopTestAvgZAngle;
float injHopTestAvgXAngle;
float injHopTestAvgYAngle;
float injHopTestAvgZAngle;

class HopTest{
  //private variables/functions.
  private:
    //state variables that each Attempt object will have.
    int hopTestNo;
    int timeTaken;
    int* orientationX;
    int* orientationY;
    int* orientationZ;

  //public variables/functions that can be called from outside the class (the API).
```

```
  public:
    //generic constructor, takes in no arguments and does nothing but create a new Attempt object.
    HopTest(){};

    //a modified constructor that takes in each of the state variables as arguments and assigns them accordingly.
    HopTest(int hopTestNo, int timeTaken, int orientationX[], int orientaionY[], int orientationZ[]){
      //this->date = date;
      this->hopTestNo = hopTestNo;
      this->timeTaken = timeTaken;
      this->orientationX = orientationX;
      this->orientationY = orientationY;
      this->orientationZ = orientationZ;
      };

    //a getter method to retrieve the hop test number.
    int getHopTesttNo(){
      return this->hopTestNo;
      };

    //a getter method to retrieve the time taken to complete this hop test.
    int getTimeTaken(){
      return this->timeTaken;
      }

    //a getter method to retrieve the array of X orientations
    int *getOrientationX(){
      return this->orientationX;
      }

    //a getter method to retrieve the array of Y orientations
    int *getOrientationY(){
      return this->orientationY;
      }

    //a getter method to retrieve the array of Z orientations
    int *getOrientationZ(){
      return orientationZ;
      }
  };

class Attempt{
  //private variables/functions.
  private:
```

```
    HopTest injured;
    HopTest normal;
    float LSI;
  //public variables/functions that can be called from outside the class (the API).
  public:
    //default contructor
    Attempt(){};

    //official contructor, initializes state variables
    Attempt(HopTest injured, HopTest normal){
      this->injured = injured;
      this->normal = normal;
      this->LSI = (float(normal.getTimeTaken())/float(injured.getTimeTaken()))*100.0;
      };

    //getter method to retrieve the injured Hop Test from the Attempt object
    HopTest getInjuredTest(){
      return this->injured;
      };

    //getter method to retrieve the normal Hop Test from the Attempt object
    HopTest getNormalTest(){
      return this->normal;
      };

    //getter method to retrieve the LSI of the attempt
    float getLSI(){
      return this->LSI;
      };
  };

//HopTest variables used in the loop() section
HopTest injuredHopTest, uninjuredHopTest;

void setup() {
  Serial.begin(115200); //initializes serial monitor

  //Setting up bluetooth
  if(!SerialBT.begin("SA(c)LT")){
    Serial.println("An error occurred initializing Bluetooth");
  }else{
    Serial.println("Bluetooth initialized");
  }
```

```
  //checking if the BNO055 is connected successfully
  if(!bno.begin())
  {
    Serial.print("Ooops, no BNO055 detected ... Check your wiring or I2C ADDR!"); //Cannot connect to BNO055
    while(1); //Endless loop forcing the user to restart their code.
  }
}



void loop() {


  //reset appropriate values to 0 depending if this is a normal or injured leg Hop Test
  if (attemptNumber%2 == 0){
    hopTestAvgXAngle = 0.0;
    hopTestAvgYAngle = 0.0;
    hopTestAvgZAngle = 0.0;
    }
  else{
    injHopTestAvgXAngle = 0.0;
    injHopTestAvgYAngle = 0.0;
    injHopTestAvgZAngle = 0.0;
  }


  //declares and initializes additional variables required
  timeTakenLocal=0;
  Attempt;
  arrayCounter = 0;
  sensors_event_t event; //Set up a new sensor event.
  bno.getEvent(&event);


  imu::Vector<3> euler = bno.getVector(Adafruit_BNO055::VECTOR_EULER); //Set the variable 'euler' to be a Vector object which
holds the euler angles.
  imu::Vector<3> linAccel = bno.getVector(Adafruit_BNO055::VECTOR_LINEARACCEL); //Sets the variable 'linAccel' to be a Vector
object which hold the linear acceleration values.


  fsrReading = analogRead(fsrPin); //takes analog reading of the FSR pin


  SerialBT.println("Enter 's' to start a new Hop Test.");
  while(SerialBT.available() == 0){};
  char startingTrigger = SerialBT.read();
  if (startingTrigger == 's'){
    Serial.println(startingTrigger);
    SerialBT.println("Lift the leg you will not be hopping with to begin.");
    while(fsrReading > 1000){
```

```
      fsrReading = analogRead(fsrPin);
      }

    SerialBT.print("Ready to begin HopTest...I will start the counter as soon as the hopping starts!\n");

    while(abs(linAccel.z()) < 15){
      linAccel = bno.getVector(Adafruit_BNO055::VECTOR_LINEARACCEL); //Sets the variable 'linAccel' to be a Vector object which
hold the linear acceleration values.
      delay(100);
      }

    fsrReading = analogRead(fsrPin);
    SerialBT.println("Hop Test recording has started.");
    //determining when the users puts their other foot down
    while (fsrReading < 1000 && arrayCounter <= 119){
      euler = bno.getVector(Adafruit_BNO055::VECTOR_EULER);
      Xorientation[arrayCounter] = euler.x();
      Yorientation[arrayCounter] = euler.y();
      Zorientation[arrayCounter] = euler.z();
      arrayCounter++;
      if (attemptNumber%2 == 0){
        hopTestAvgXAngle += float(euler.x());
        hopTestAvgYAngle += float(euler.y());
        hopTestAvgZAngle += float(euler.z());
      }
      else{
        injHopTestAvgXAngle += float(euler.x());
        injHopTestAvgYAngle += float(euler.y());
        injHopTestAvgZAngle += float(euler.z());
        }
      delay(500);
      fsrReading = analogRead(fsrPin);
      }

    //calculates the average angles to be displayed later
    if (attemptNumber%2 == 0){
      hopTestAvgXAngle /= float(arrayCounter+1);
      hopTestAvgYAngle /= float(arrayCounter+1);
      hopTestAvgZAngle /= float(arrayCounter+1);
      }
    else{
      injHopTestAvgXAngle /= float(arrayCounter+1);
      injHopTestAvgYAngle /= float(arrayCounter+1);
```

```
      injHopTestAvgZAngle /= float(arrayCounter+1);

      }


    timeTakenLocal = float(arrayCounter+1.00)*0.50; //calculates the taken depending on the number of acceleration readings taken

    SerialBT.println("Hop Test ended. Indicate if this was a passed Hop Test by entering 'p'. Enter anything else to indicate a
failed Hop Test...");

    while(SerialBT.available() == 0){};

    char success = SerialBT.read();


    //Records the hop test if this a passed hop test

    if(success == 'p'){

      attemptNumber++;


      //determines if it was an injured or normal leg hop test

      if (attemptNumber%2 == 1){

        uninjuredHopTest = HopTest(attemptNumber, timeTakenLocal, Xorientation, Yorientation, Zorientation);

        SerialBT.println("Uninjured leg Hop Test recorded.");

        }

      else{

        injuredHopTest = HopTest(attemptNumber, timeTakenLocal, Xorientation, Yorientation, Zorientation);

        SerialBT.println("Injured leg Hop Test recorded.");

        attempt = Attempt(injuredHopTest, uninjuredHopTest);

        if (attemptNumber >=4)

          avgLSI = ((avgLSI*((attemptNumber/2)-1))+attempt.getLSI())/((attemptNumber/2));

        else

          avgLSI = attempt.getLSI();


        //if there was a previous attempt, compare the previous LSI to the current LSI

        if (attemptNumber > 2){

          if (attempt.getLSI() > prevLSI){

            SerialBT.print("Your LSI has improved from ");

            SerialBT.print(prevLSI);

            SerialBT.print(" to ");

            SerialBT.print(attempt.getLSI());

            SerialBT.println(". Great work!");

            }

          else if (attempt.getLSI() < prevLSI){

            SerialBT.print("Your LSI has gone down from ");

            SerialBT.print(prevLSI);

            SerialBT.print(" to ");

            SerialBT.print(attempt.getLSI());

            SerialBT.println(". You can do this!");

            }
```

```
        else SerialBT.println("Your LSI did not change. Strive for improvement!");
        }
      prevLSI = attempt.getLSI(); //save previous LSI

      //Display the results for the user
      SerialBT.println("_____________________________________________________");
      SerialBT.println("_________________Attempt Summary_________________");
      SerialBT.print("Uninjured leg time: ");
      SerialBT.println(float(uninjuredHopTest.getTimeTaken()));
      SerialBT.print("Injured leg time: ");
      SerialBT.println(float(injuredHopTest.getTimeTaken()));
      SerialBT.print(" This attempt's LSI is:   ");
      SerialBT.println(attempt.getLSI());
      SerialBT.print(" Your average LSI is:     ");
      SerialBT.println(avgLSI);
      SerialBT.print(" Normal leg Avg. X Angle: ");
      SerialBT.println(hopTestAvgXAngle);
      SerialBT.print(" Normal leg Avg. Y Angle: ");
      SerialBT.println(hopTestAvgYAngle);
      SerialBT.print(" Normal leg Avg. Z Angle: ");
      SerialBT.println(hopTestAvgZAngle);
      SerialBT.print("Injured leg Avg. X Angle: ");
      SerialBT.println(injHopTestAvgXAngle);
      SerialBT.print("Injured leg Avg. Y Angle: ");
      SerialBT.println(injHopTestAvgYAngle);
      SerialBT.print("Injured leg Avg. Z Angle: ");
      SerialBT.println(injHopTestAvgZAngle);
      SerialBT.println("_____________________________________________________");
    }
  }
  //allow the user to record another hop test is this one was failed.
  else{
    SerialBT.println("Failed Hop Test not recorded.");
    }
  }
}
```

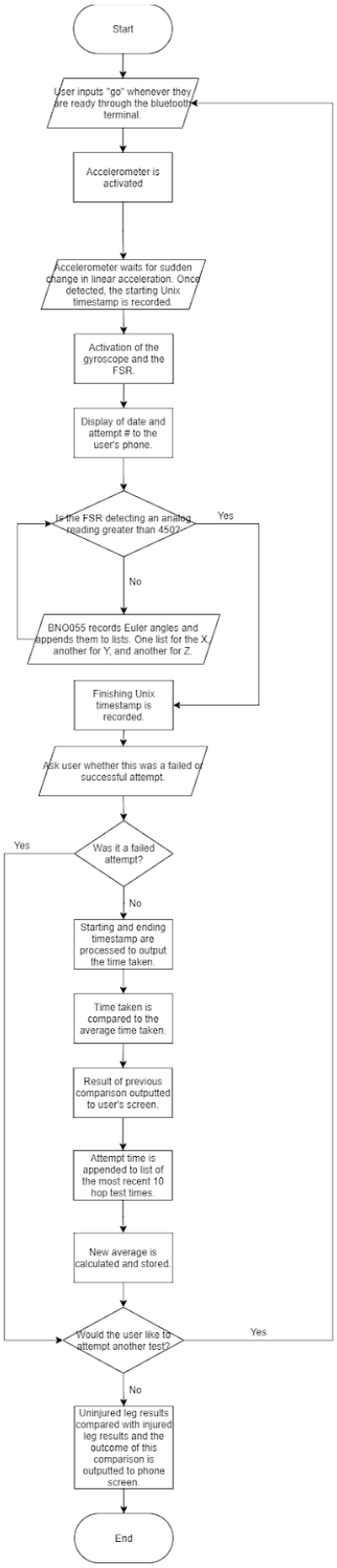


Figure 1Code flowchart outlining how the program will be executed.

Appendix D: Chart and Indented Views of WBS, and Bill of Materials

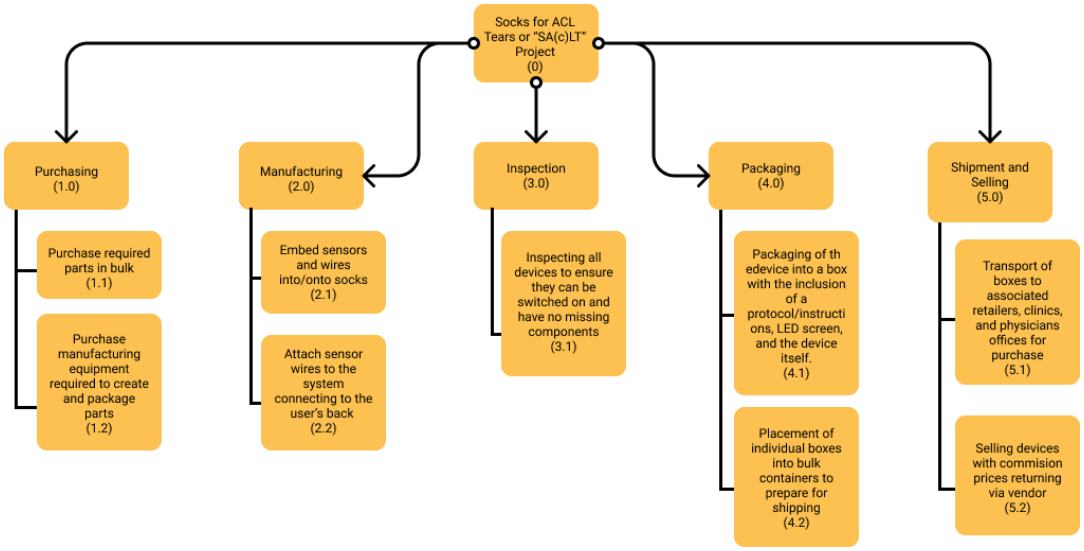


Chart View of WBS

Indented View of the WBS

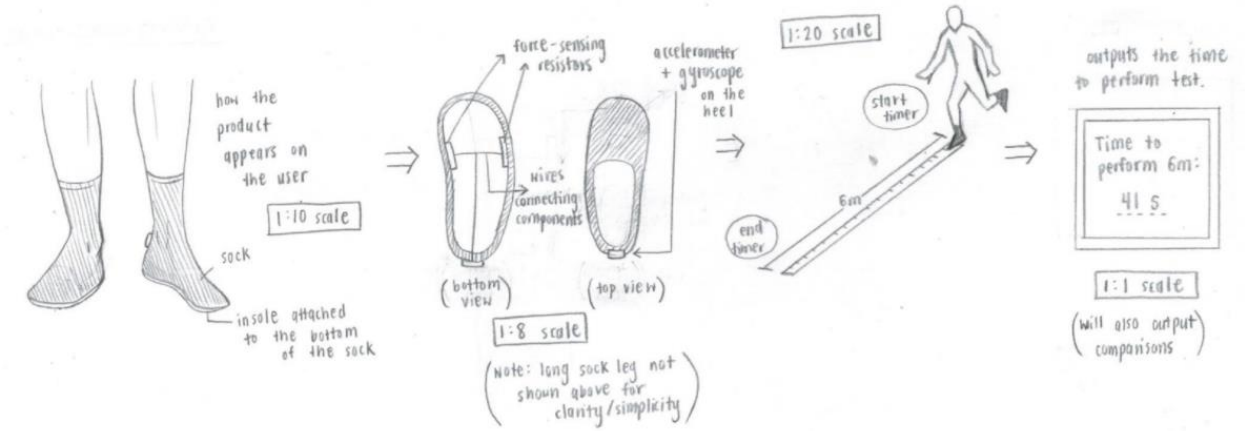
Work Level		Work Element	Expected Duration (days)
0.0		Socks for ACL Tears or "SA(c)LT" Project	
1.0		Purchasing	
	1.1	Purchase required parts in bulk	30
	1.2	Purchase manufacturing equipment required to create and package parts	35
2.0		Manufacturing	
	2.1	Embed sensors and wires into/onto socks	35
	2.2	Attach sensor wires to the system connecting to the user's back	35
3.0		Inspection	
	3.1	Inspecting all devices to ensure they can be switched on and have no missing components	14
4.0		Packaging	
	4.1	Packaging of the device into a box with the inclusion of a protocol/instructions, and the device itself.	21

	4.2	Placement of individual boxes into bulk containers to prepare for shipping	14
5.0		Shipment and Selling	
	5.1	Find vendors/clinics who will sell our product	49
	5.2	Transport of boxes to associated retailers, clinics, and physician offices for purchase	42
	5.3	Selling devices with commission prices returning via vendor	100+

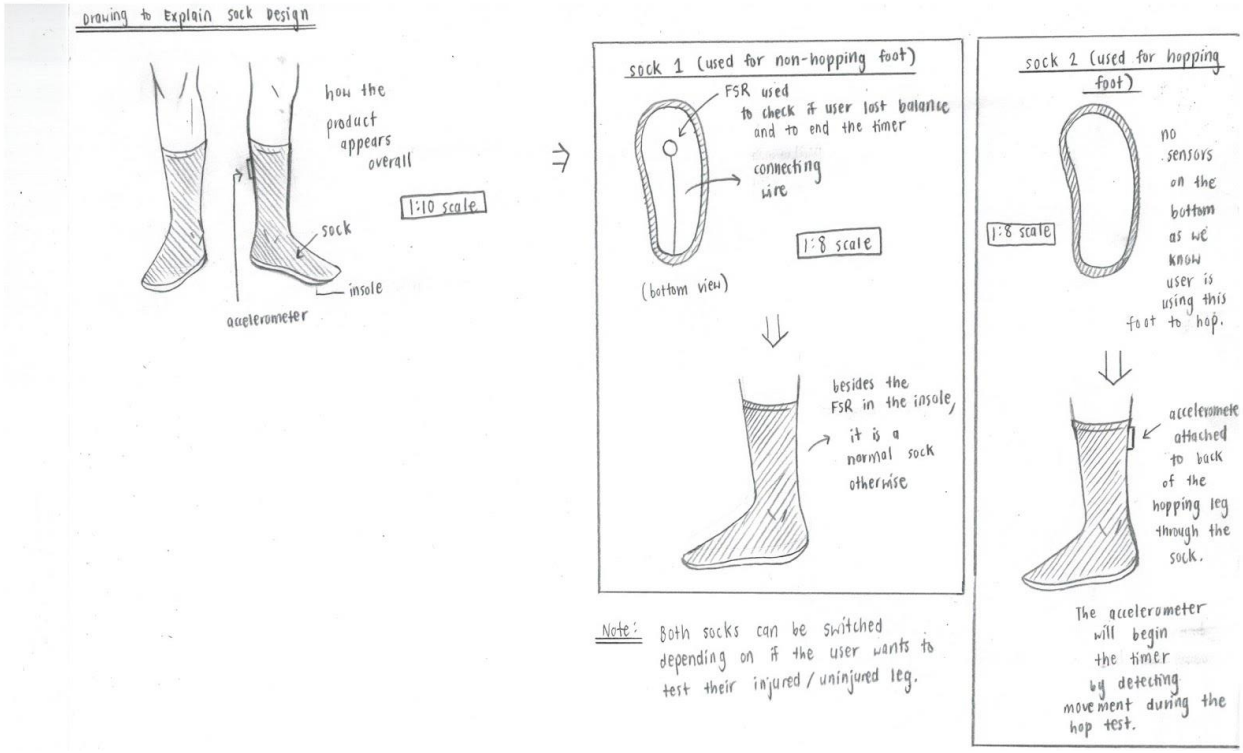
Bill of Materials

Material	Cost
Accelerometer [3]	$27 / sensor
FSR [4]	$9 / sensor
ESP32 [5]	$20
Crew Socks with Rubber Soles [6]	$16 / 4 pairs → $4 / pair
Longer Wires [7]	$90 / 75 m → $2 / 1.5 m
Manufacturing	TBD, however, this is included in the $80 as an approximation
Total:	~ **$80** making SA(c)LT financially viable

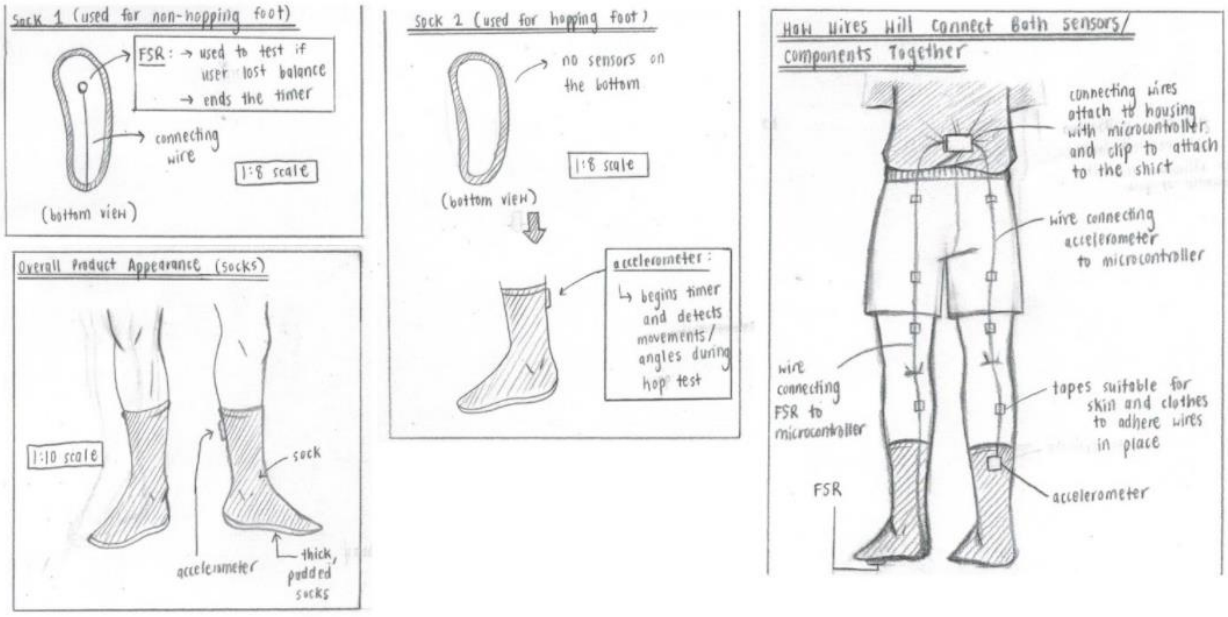
Appendix E: Design Sketches Over the Iterations



First iteration sketches.



Second iteration sketches



Third and Final Iteration of the SA(c)LT.