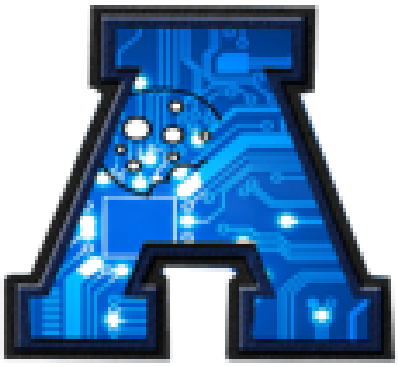
CPE 4850 - Senior Project Design, Spring 2021

Final Project Report

On-Track Smart Shoes

Assorted Bytes



Date: May 8, 2021

Electrical and Computer Engineering

Kennesaw State University

Faculty: Dr. Jeffrey L Yiin

TABLE OF CONTENTS

Project Summary

Introduction:

The idea for the On-Track smart shoes stemmed from inspiration where we wanted the customer to have access to multiple aspects of health data with requiring a device that may not be always on their person throughout the day. We also want to be able to incorporate multiple facets into the design that would allow the user to not get basic health information but include something that the user may only be able to do when at home. Throughout this report, the group wants to make it clear what our objective was in designing the product and working through that process with how we did it. The following sections will allow the reader to see our approach and design throughout the process.

Problem Statement:

There is a constant increase with the use of technology within our society every day. Even with this truth, there are still areas of society where certain devices are not technologically advanced, but there should not be anything holding them back from being upgraded and having more technology implemented within them. A user's health data is very important their everyday schedule and accomplishing certain tasks throughout their day. How can we integrate multiple facets of health information to allow the user to maintain their activity levels throughout the day while not requiring them to use outside devices to get accurate responses?

Objective:

The objective for our group was to design a device that would allow the user to not require outside devices to keep track of their health data in everything they do throughout the day. The system would incorporate multiple different facets to collect health data and not be refrained to targeting one area of measurements or focus within the device. The system would also include a battery system that would be able to keep the device charged and functioning when the user wants to use it without having to wait. Within the system, there will be a single computer that connects all the sensors and battery pack together to make it all work flawlessly together without any interruptions or changes necessary each time the user wants to use the shoes during the day.

Proposed Solution:

The device we decided to create focuses on an already invented product, shoes, but plans to greatly advance the technology located within the shoes. Our On-Track Smart Shoes would contain numerous sensors to collect data including pressure sensors to measure body weight, an ultrasonic sensor to check a person's posture, vibration motors to be used as massagers, and an accelerometer to count steps, track distance and speed over time. There would be a wireless charging system introduced within the shoe as well, this system would use a Lithium-Polymer battery, a charger, and a Qi charging module to allow the user to put against a wireless charger before use each day. This system would connect to the Raspberry Pi Zero W that we decided to use because of its small size and functional capability to accomplish the tasks required within our design. The data that is being collected will be sent to a mobile application on all devices to where the user can communicate and have their collected data displayed in front of them on the app for them to see throughout the day.

Design and Prototype Outcome:

Our design process allowed us to look at multiple different areas of which type of sensors we wanted to incorporate within the shoes, it was easy to decide to use an accelerometer to count steps, distance traveled, and speed over time because due to our research we learned that most people would keep track of this data most of the time, if not already. Then we wanted to incorporate some sensors that users normally must be in one location, such as home or the gym, to collect and see certain measurements. Therefore, we chose to add the vibration motors as massagers and the weight sensors to calculate a person's body weight, so they can do it at any point throughout the day without needing to be in a specific location. Lastly, we wanted to try something different so that is why we implemented the ultrasonic sensor, people never check to make sure they are standing correctly, which can show multiple different things if their posture is off, so we wanted to be able to check the user's posture and show them if they are standing correctly or not.

Our prototype turned out well except for numerous extended wires all over the place and the spacing being a lot more restricted than originally thought of beforehand. The only thing we were able to fit into the tongue of the shoe was the battery/wireless charging system while the rest of the equipment had to hang out of the shoe a little bit. The vibration motors, ultrasonic sensor, weight sensors, and accelerometer all were able to fit where we intended them to be in our design process beforehand which was beneficial in allowing us to attempt to limit any extra sizing that was necessary. The wireless charging system worked how we thought it would which was a very big positive, since that is key in making sure the system will last over time. Except for a few size issues and the excess amount of wiring that was needed, our prototype turned out how we expected it to and with some extra work I am sure we could cut down on the size of the overall system altogether.

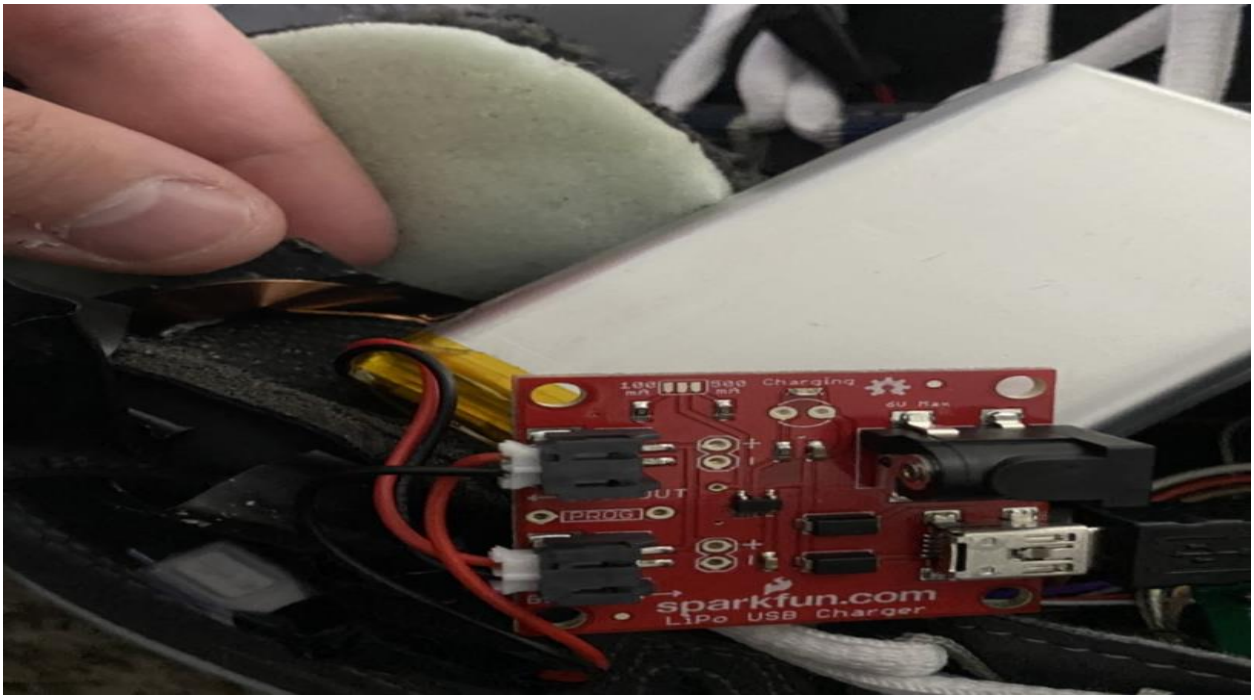


Figure 1: Contents of tongue of shoe including battery, charger, and Qi charging module

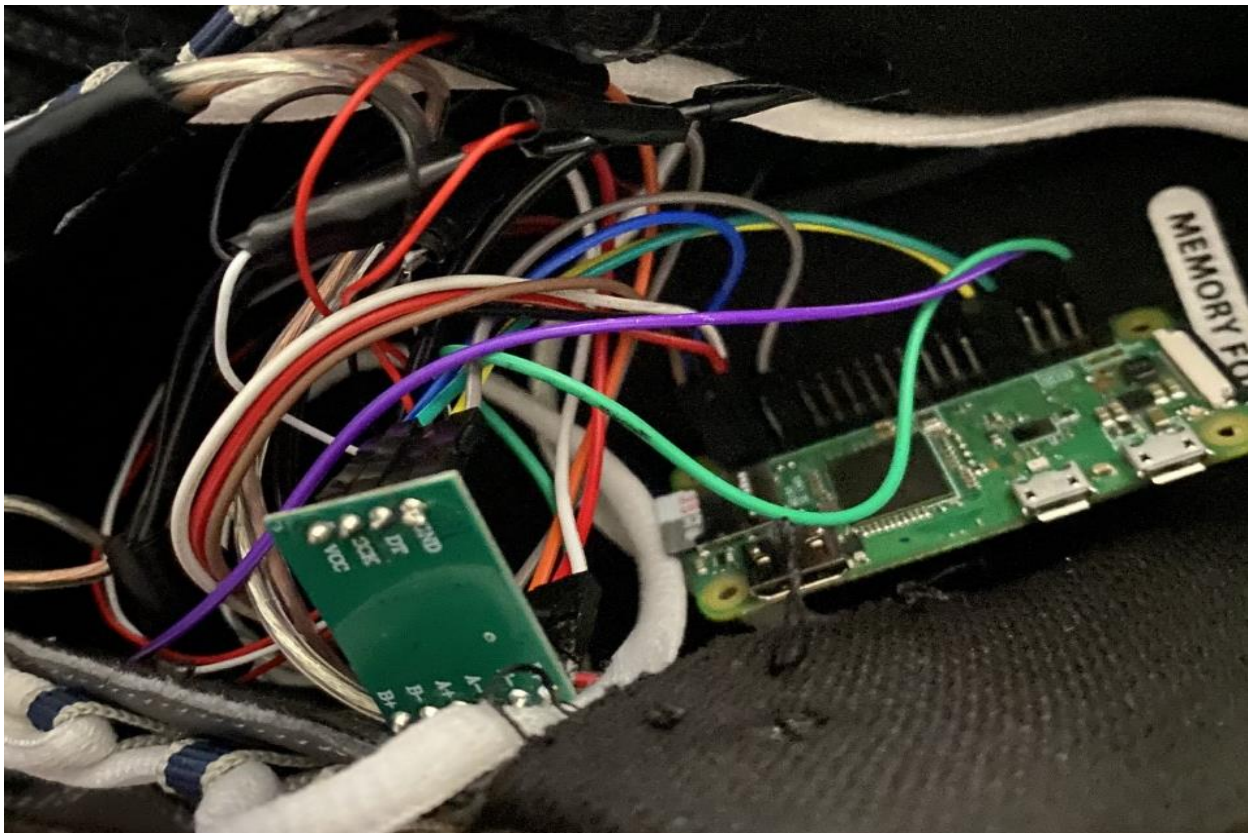


Figure 2: Contents of shoe that sat on the outside with wires, Raspberry Pi Zero W and weight sensor module

Product Description

Our product is a pair of smart shoes that contain multiple different sensors to collect numerous sets of data. Within the shoe there is a step counter that can not only count your steps, but also track how far you have traveled (distance over time) and your speed throughout the duration of your day. Another feature of the shoes is the massager, that has three separate settings (low, medium, and high), that can be turned on and changed at any moment the user feels necessary. The weight sensors are another feature within the shoes that can accurately show your weight at any time during the day with the press of a button when you are standing balanced within the shoes. The last sensor feature is used to help determine if you have good posture or not, which uses the distance between both feet when you are standing to give you a response. Besides the senor features within the shoes, there is one more big feature incorporated that turned out well, this feature is the wireless charging. There is a 4500 mAh battery that keeps the Raspberry Pi Zero W running for us to use the sensors, to keep the battery charged there needs to be a charger input into the system somehow. There is a Qi (wireless) charging module within the tongue of the shoe that allows the user to charge their shoes when they are not using them either during the day or overnight. Ultimately our system works together with all features being active when the Raspberry Pi is powered up and should cause minimal issues of the user in collecting the data they want to see.

Figure 3: Prototype with contents inside

Figure 4: Prototype with exposed contents in both shoes

Hardware System Design

Smart Shoe Diagram:

Figure 5. Smart shoe with diagrammed sensor locations

Hardware Block Diagram:

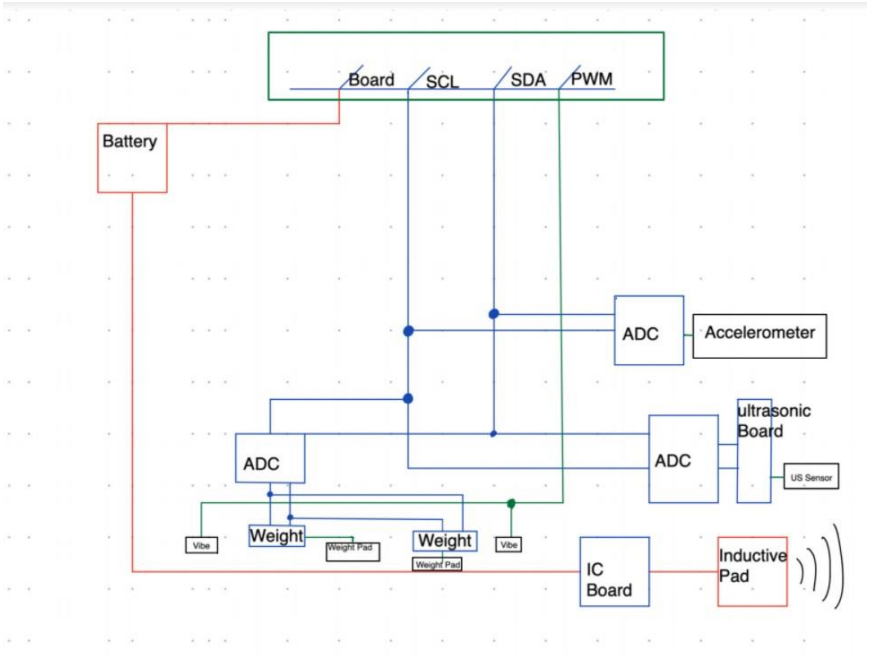


Figure 6. Block Diagram for smart shoe hardware.

Component Functional Description:

- **Vibration**: Starting with a bottom to top approach of the shoe we begin in the sole of the shoe. In each shoe there are two vibration motors (Tokoto with range of 8,000-16,000 RPM). These Vibration motors were connected to the microcontroller using the PWM GPIO pins.
- **Ultrasonic**: Exclusive to the right shoe to measure posture is the ultrasonic sensor (SR04T). Which connects to the raspberry Pi through an analog conversion board.
- **Pressure**: Belonging in the shoe underneath the insole cushion belongs the weight/pressure sensors (hx711 and load cells) of which there are two in each shoe. These sensors wire together and through the Hx711 board connect to the embedded devices GPIO.
- **Accelerometer**: Exclusive to the left shoe belongs the accelerometer (ADXL345) in the interior of the ankle. This is connected directly to the GPIO of the embedded device.
- **Battery System**: For the battery we used a wireless charging receiver (Qi wireless receiver module by Adafruit), this was connected to a charging board where our 4,500Mah Lithium Polymer better was connected. This board also allowed us to connect the battery to the embedded device for operation.
- **Embedded System**: the system we chose was the Raspberry Pi Zero W, we chose this for the Linux environment and wealth of GPIO availability. As well as wireless interfacing being a part of the system and not needing any additional modules.

PCB Schematic:

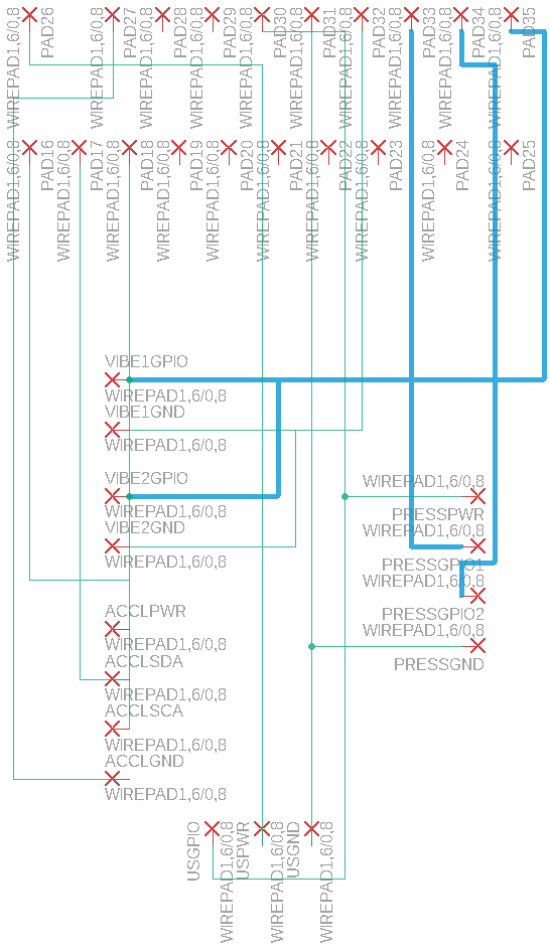


Figure 7. Schematic for Smart Shoe System.

We designed a PCB for our project, but ultimately decided against using one since it would be an extra cost that was unnecessary and once, we discovered our space capacities were a lot smaller than we originally anticipated, we determined using a PCB was not feasible, especially since the main goal for us would have been just connecting the sensors which we were capable of doing just fine without a PCB needed.

Bill of Materials:

Component	Cost	Quantity	Total Cost
Raspberry Pi Zero w/microcontroller	$20	2	$40
HX711 Weight Load Sensor	$4	4	$16
ADXL 345 Accelerometer	$4	1	$4
Battery pack 4500 mAh Li-Po	$14	2	$28
SparkFun USB LiPoly Charger	$17	2	$34
Pair of Shoes	$40	1	$40
Qi Charging Receiver	$15	2	$30
3V 5000 RPM Vibration Motors	$3	4	$12
JSN-SR04T Integrated Ultrasonic Module	$10	1	$10
16gb MicroSD card (non-volatile Storage)	$7	2	$14

Encasing Design

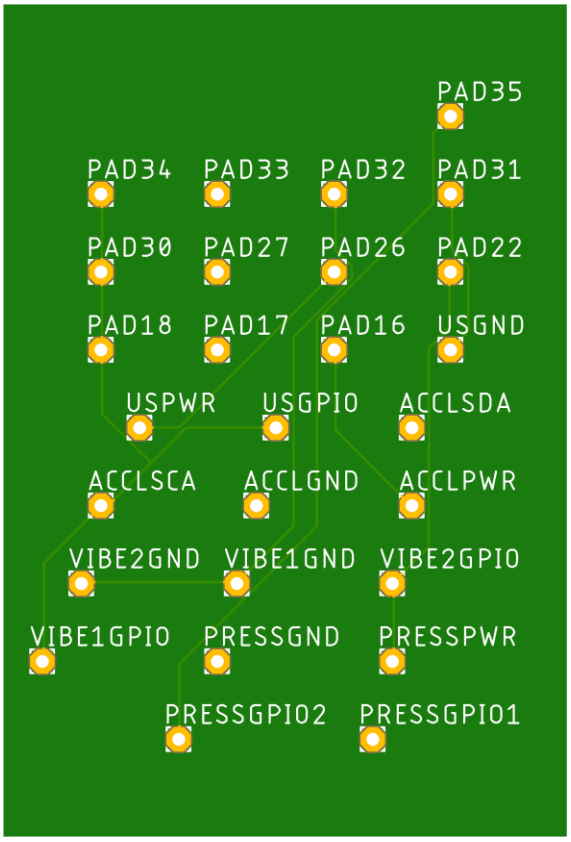


Figure 8. PCB design for connection routing in smart shoe.

Our primary encasing for the project was our shoe, this is what we modified to encase our sensors and embedded system. We drilled holed in the sole for the vibration motors and for the ultrasonic sensor. We modified the cloth insides by making cuts and mending them to place our sensors. We also made cuts and mends to the tongue of the shoe in order to place the battery and embedded device in the shoe.

We planned secondarily to use a flexible PCB to route the wires through the shoe (design shown above, however the model will not produce flexible examples), however shipping and production were not feasible for a flexible PCB product in our frame of time.

Software System Design

For our system's software we connected directly to the Raspberry Pi Zero W using the Raspberry Pi OS where we set up our drivers for the sensors through terminal. We wrote python files for each sensor listed below and in our GitHub repository. Using these python files, we set up bash scripts to allow them to be activated allowing the user to obtain the required data at anytime when they press a button within our mobile application we developed. All the files would start on boot up of the Raspberry Pi Zero W, but only the accelerometer file would always be collecting data while the others would respond on the press of a button. Speaking of mobile application, we designed a hybrid application built within React Native on our PC that would allow all users with either android, iOS, or web to be able to see and start the application.

SW Flow Charts:

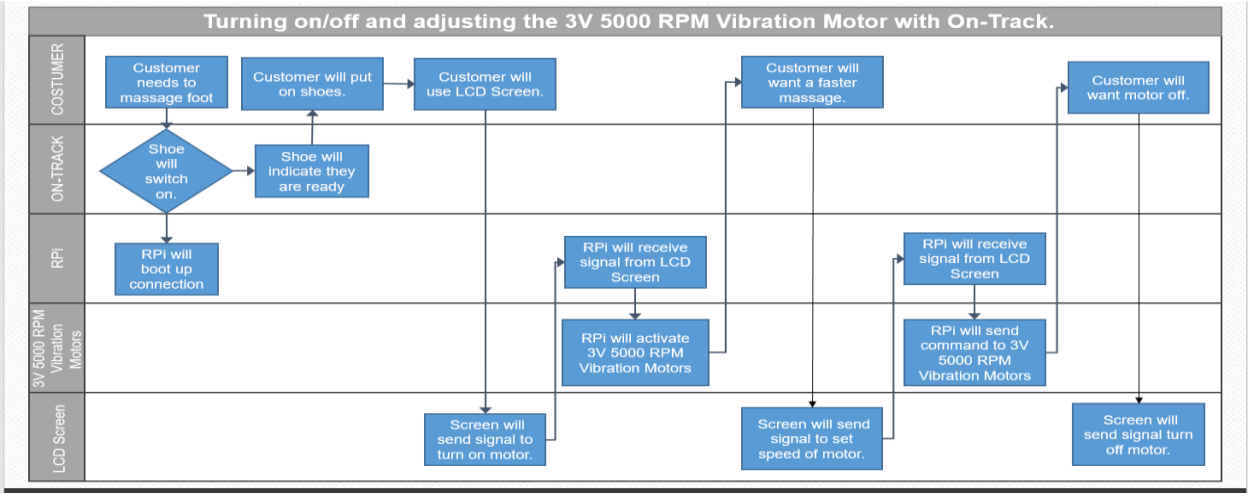


Figure 9: Turning on/off the Vibration Motor

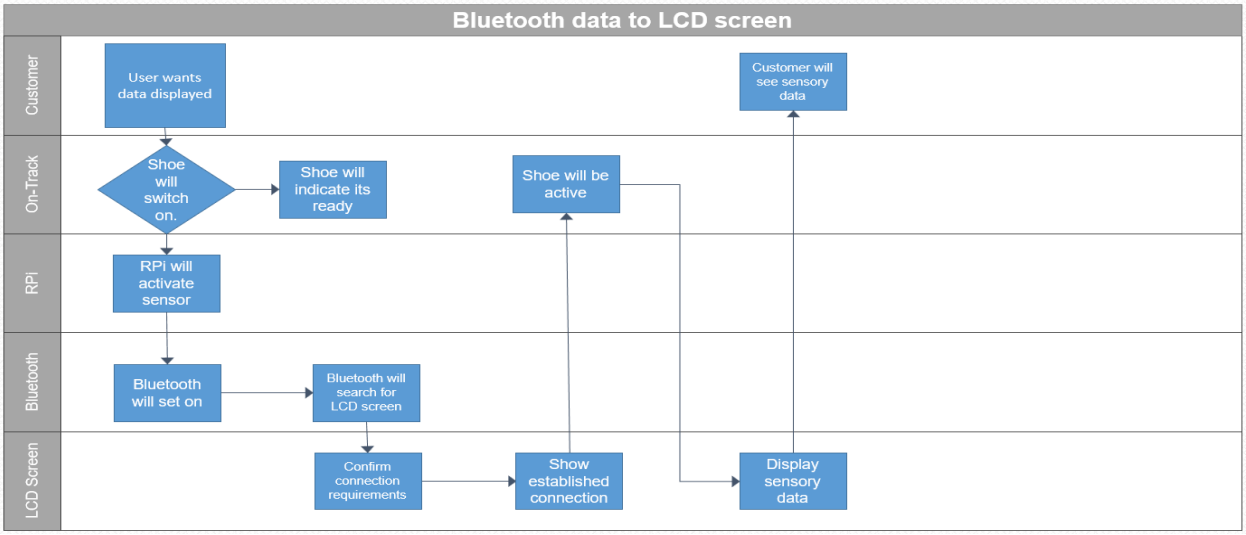


Figure 10: Bluetooth data to the mobile device

Use Cases:

1. User wants to track the number of steps they take during a workout and how far they have traveled through the mobile app the user can access the steps screen to see their data that is collected over time.
2. User wants to weigh themselves after a good workout session through the mobile app the user can access the weight screen and press the button to capture their body weight.
3. User wants to check their posture to see if they are standing correctly through the mobile app the user can check to see if their standing at the correct width.

Key Algorithms:

- All source code will be in the appendix section.

```
import RPi.GPIO as GPIO                    #Import GPIO library
import time                                #Import time library
GPIO.setmode(GPIO.BCM)                     #Set GPIO pin numbering

GPIO_TRIGGER = 23
GPIO_ECHO = 24                             #Associate pin 14 to Echo

print ("Posture Test in progress")

GPIO.setup(GPIO_TRIGGER,GPIO.OUT)          #Set pin as GPIO out
GPIO.setup(GPIO_ECHO,GPIO.IN)              #Set pin as GPIO in

while True:

  GPIO.output(GPIO_TRIGGER, False)         #Set TRIG as LOW
  print ("Checking Stance")                #Checking the User stance
  time.sleep(10)                           #Checking the user stance for 30 Seconds

  GPIO.output(GPIO_TRIGGER, True)          #Set TRIG as HIGH
  time.sleep(0.00001)                      #Delay of 0.00001 seconds
  GPIO.output(GPIO_TRIGGER, False)         #Set TRIG as LOW

  while GPIO.input(GPIO_ECHO)==0:          #Check if Echo is LOW
    pulse_start = time.time()              #Time of the last  LOW pulse

  while GPIO.input(GPIO_ECHO)==1:          #Check whether Echo is HIGH
    pulse_end = time.time()                #Time of the last HIGH pulse

  pulse_duration = pulse_end - pulse_start #pulse duration to a variable

  distance = pulse_duration * 17150        #Calculate distance
  distance = round(distance, 2)            #Round to two decimal points

  if distance > 0 and distance < 20:       #Testing the Stance of the user within shoulder length which is on average 8-10cm apart
    print (":",distance - 0.5,"cm")        #Distance with calibration
  else:
    print ("User not standing correctly")  #User was not standing at 8-10cm apart
```

Figure 11: Ultrasonic code to test posture. The

```python
import RPi.GPIO as GPIO
import time
import sys
from weight import HX711

def cleanAndExit():
    print("Cleaning...")
    GPIO.cleanup()
    print("Bye!")
    sys.exit()

hx = HX711(9, 11)

# I've found out that, for some reason, the order of the bytes is not always the same between versions of python, numpy and the hx711 itself.
# Still need to figure out why does it change.
# There is some code below to debug and log the order of the bits and the bytes.
# The first parameter is the order in which the bytes are used to build the "long" value.
# The second paramter is the order of the bits inside each byte.
# According to the HX711 Datasheet, the second parameter is MSB so you shouldn't need to modify it.
hx.set_reading_format("LSB", "MSB")

# HOW TO CALCULATE THE REFFERENCE UNIT
# To set the reference unit to 1. Put 1kg on your sensor or anything you have and know exactly how much it weights.
# In this case, 92 is 1 gram because, with 1 as a reference unit I got numbers near 0 without any weight
# and I got numbers around 184000 when I added 2kg. So, according to the rule of thirds:
# If 2000 grams is 184000 then 1000 grams is 184000 / 2000 = 92.
#hx.set_reference_unit(113)
hx.set_reference_unit(1)

hx.reset()
hx.tare()

while True:
    try:
        # These three lines are usefull to debug wether to use MSB or LSB in the reading formats
        # for the first parameter of "hx.set_reading_format("LSB", "MSB")".
        # Comment the two lines "val = hx.get_weight(5)" and "print val" and uncomment the three lines to see what it prints.
        #np_arr8_string = hx.get_np_arr8_string()
        #binary_string = hx.get_binary_string()
        #print binary_string + " " + np_arr8_string
```

```python
hx.set_reference_unit(1)

hx.reset()
hx.tare()

while True:
    try:
        # These three lines are usefull to debug wether to use MSB or LSB in the reading formats
        # for the first parameter of "hx.set_reading_format("LSB", "MSB")".
        # Comment the two lines "val = hx.get_weight(5)" and "print val" and uncomment the three lines to see what it prints.
        #np_arr8_string = hx.get_np_arr8_string()
        #binary_string = hx.get_binary_string()
        #print binary_string + " " + np_arr8_string

        # Prints the weight.
        val = hx.get_weight(5)
        print(val)

        hx.power_down()
        hx.power_up()
        time.sleep(0.5)
    except (KeyboardInterrupt, SystemExit):
        cleanAndExit()
```

Figure 12: Weight Sensor code

```python
import time
import board
import busio
import array
import math
import adafruit_adxl34x

i2c = busio.I2C(board.SCL, board.SDA)
accelerometer = adafruit_adxl34x.ADXL345(i2c)

accel = []
upperthreshold = 5.0
lowerthreshold = -5.0
steps = 0
count = 0
height = 72
stride = 0
distance = 0
speed = 0
start = time.time()

#height = height * 0.0254

while True:
    StrideT = time.time() - start
    StrideTime = (int(StrideT))
    #print(StrideTime)
    x, y, z = accelerometer.acceleration
    mag = math.sqrt((x**2)+(y**2)+(z**2))
    accel.append(mag)
    mean = sum(accel) / len(accel)
    magzero = mag - mean
    print(magzero)
    #print(accelerometer.acceleration)
    time.sleep(0.25)

    if ( magzero > upperthreshold ):
        steps = steps + 1
        print(steps)
        count = count + 1
        #print(count)
        #distance = distance + (stride * steps)
        #print(distance)
    if ( magzero < lowerthreshold ):
        steps = steps + 1
        print(steps)
        count = count + 1
        #print(count)
        #distance = distance + (stride * steps)
        #print(distance)
    #else:
        #print( "No step detected" )

        if(StrideTime % 2 == 0):
            if (count == 1):
                stride = height / 5
                #print(stride)
                distance = distance + (steps * stride)
                #print(distance)
                speed = count * (stride / 2)
                print(speed)
            if (count == 2):
```

```
                stride = height / 4
                #print(stride)
                distance = distance + (steps * stride)
                #print(distance)
                speed = count * (stride / 2)
                print(speed)
            if (count == 3):
                stride = height / 3
                #print(stride)
                distance = distance + (steps * stride)
                #print(distance)
                speed = count * (stride / 2)
                print(speed)
            if (count == 4):
                stride = height / 2
                #print(stride)
                distance = distance + (steps * stride)
                #print(distance)
                speed = count * (stride / 2)
                print(speed)
            if (count == 5):
                stride = height / 1.2
                #print(stride)
                distance = distance + (steps * stride)
                #print(distance)
                speed = count * (stride / 2)
                print(speed)
            if (6 <= count < 8):
                stride = height
                #print(stride)
                distance = distance + (steps * stride)
```

```
                #print(distance)
                speed = count * (stride / 2)
                print(speed)
            if (count >= 8):
                stride = height * 1.2
                #print(stride)
                distance = distance + (steps * stride)
                #print(distance)
                speed = count * (stride / 2)
                print(speed)
        else:
            count = 0
```

Figure 13: ADXL 345 Accelerometer

```
import RPi.GPIO as GPIO
from time import sleep
led_pin = 21            # Initializing the GPIO pin 21 for vibration sensor
GPIO.setmode(GPIO.BCM)          # We are using the BCM pin numbering
GPIO.setup(led_pin, GPIO.OUT)   # setting pin 21 as output pin
pwm = GPIO.PWM(led_pin, 100)    # PWM object is created at 100Hz
pwm.start(0)                    # Started PWM at 0% duty cycle
try:
    test_text = input ("Enter a number between 0 and 100: ")
    test_text = int(test_text)
    while 1:                    # Loop will run forever
      if ((test_text>=1)and(test_text<=30)):
       pwm.ChangeDutyCycle(60) # Change duty cycle
       sleep(.10)
      elif ((test_text>=31)and(test_text<=60)):
       pwm.ChangeDutyCycle(90)
       sleep(.10)
      elif((test_text>=61)and (test_text<=100)):
       pwm.ChangeDutyCycle(100) # Change duty cycle
       sleep(.10)
      else:
       print "enter an accepetable value"
       pwm.ChangeDutyCycle(0)

# If keyboard Interrupt (CTRL-C) is pressed
except KeyboardInterrupt:
    pass
pwm.stop()
GPIO.cleanup()  # Make all the output pins LOW
```

Figure 14: Vibration sensor using PWM

GPIO Pinout

Pinout for Ultrasonic sensor

GPIO 23, GPIO 24, 5v Power and GND

The test function for the ultrasonic sensor will assumes correct posture is met when your feet are between 9 to 20 cm apart.

Pinout for AXL334 Accelerometer

GPIO 2 (SDA), GPIO 3 (SDL), 5V power and GND

The test function to count steps assumes a step is taken if certain conditions are met. Else the entire function will count zero steps.

Pinouts for Weight sensor

GPIO 9, GPIO 11, 5V power and GND

The test function returns the weight of the object applying pressure to the metal plate. If no pressure is applied, then code assumes no force is being applied to the metal plates.

Pinout for vibration

GPIO 21(PWM) and GND

The GPIO pin is controlled using a pulse with modulator. That allows for different frequencies to output out from test function.

User Application Design

We used React Native Expo to create an app that will help users access OnTrack's features more effectively and efficiently. First, the user will connect to the shoes and the app via Bluetooth. After pairing, the user will be taken to the welcome screen where they have the option of creating an account or if they already have an account, they can login to the app. The user will then be taken to the home screen where they can keep track of and manage various aspects of daily life such as daily steps, speed/distance, record body weight, check posture, and access massage features. Our app is not fully finished, but we have the prototype of how we want it to look as shown in the flowchart below.

On-Track App Design Concept

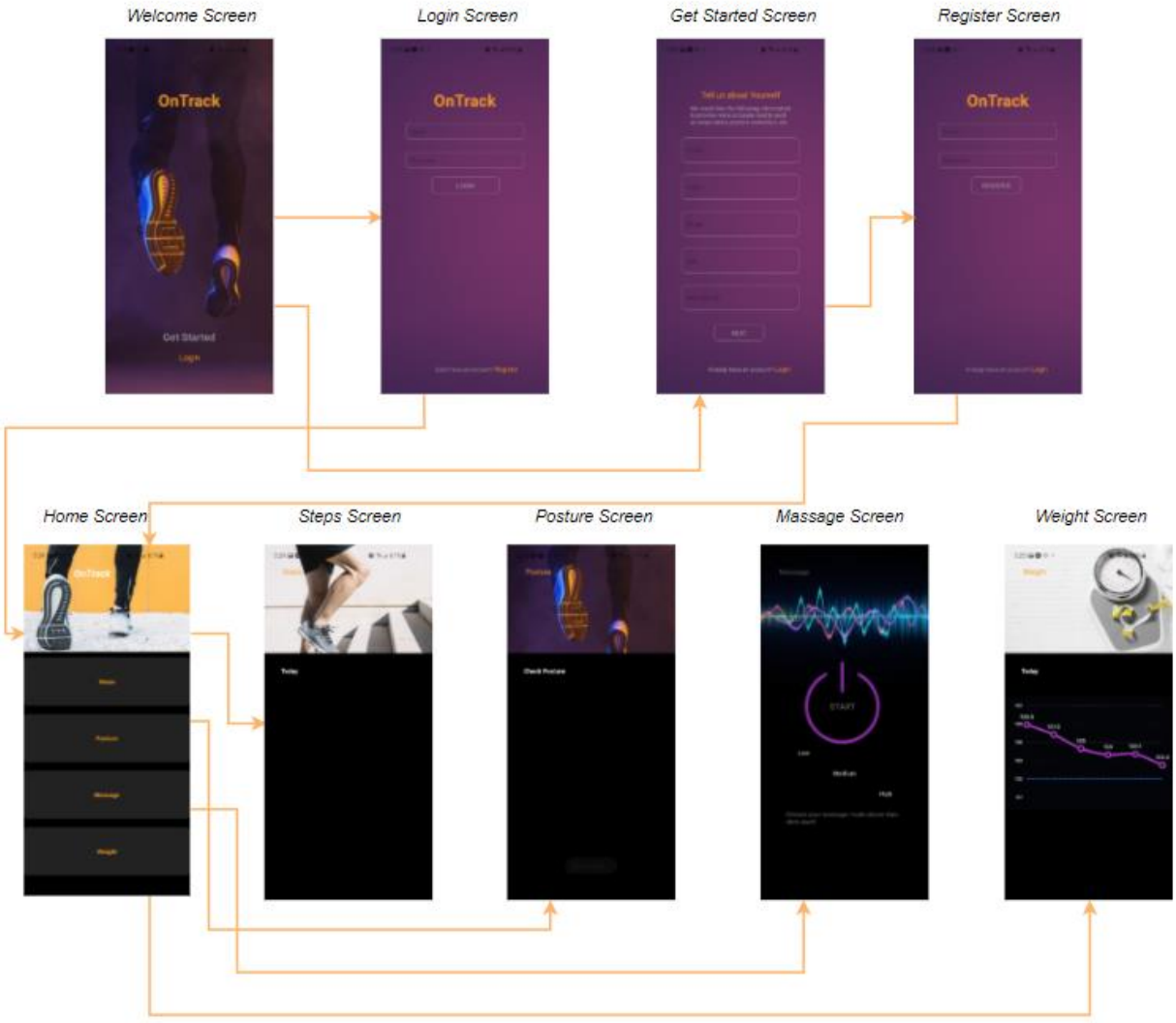


Figure 15: Mobile application flowchart

Product Requirement Analysis

Top 10 PRD Items:

1. The Embedded System shall use the Raspberry Pi Zero w/ microcontroller. - Compliant
2. The Embedded System shall interact with the load sensors to detect the weight of the user and use this information to display other measurements. - Compliant
3. The Embedded System shall interact with the vibration motors to massage the user's foot. -Compliant
4. The Embedded System shall interact with the accelerometer to count steps and detect speed/distance and display them on the mobile device. -Compliant
5. The Embedded System shall display the data from the load sensors, accelerometers and control the motors on the mobile device. -Compliant
6. The system used to massage the inside of the shoe using multiple 3V 5000 RPM vibration motors allowing for the user to target certain areas depending on needs
7. The system used to detect the weight of the user from the HX711 weight load sensors when wearing the shoes and converting that data into other measurements. A JSN-SR04T Integrated Ultrasonic Module is implemented to determine posture by distance between shoes for the user. -Compliant
8. An inductive charger shall be available to keep the Embedded System from dying. - Compliant
9. The device communications system, allowing users to be able to see their data and progress they have made. -Compliant
10. The system used to determine number of steps and speed/distance over time from ADXL 345 Accelerometers within each shoe. -Compliant

Lessons Learned

Throughout the past two semesters, there was a lot to be learned about ourselves and the project itself. Last semester was focused on making sure we were on the same page throughout the semester, and we all knew exactly what the On-Track shoes entailed and contained as far as sensors and equipment is concerned. Teamwork was very important during this time, since it was a new group and there was a lot of ideas being thrown around and suggestions being made throughout the process that were conflicting with one another, so we needed to make sure we were on the same page throughout all of it and reached our end goal no matter what. While last semester was a lot more research, it prepared us for this semester and the work that needed to be done and how we were going to work together as a team and what would be an effective way to get work done amongst each other during this crazy time.

This semester was where we learned the most about ourselves as individuals and as a team as well. We were able to learn our strengths and weaknesses, as well as our boundaries throughout the process. For myself, as project manager, I was able to learn what the best way to approach my group throughout the semester and while we work on the project and the different areas that needed to be completed at each milestone. One key thing was making sure that if no progress was being made, that we pushed on from that task to accomplish other tasks that needed to be done.

As it pertains to product development, we learned very quickly that time and effort is very important to having a working prototype by the end of the semester. Throughout this process we hit numerous bumps, whether it was because some sensors did not fit properly or we did not have the room we imagined, it allowed us to see that some adjustments were necessary and that we should have started to design and build a physical model earlier rather than later. With this in mind, we also saw how necessary this can be to making sure what we are designing will fit and work together as a system with all sensors and devices connected within the smart shoe. This was arguably the biggest take away that we learned this semester, along with making sure we start application development early enough to have a properly working app through all sources even if it takes different implementations for each.

Future Work

This project was only just the beginning for smart shoes. The shoe that was used for this project was a low-grade shoe that did not take much to take apart and was not put together very well in the first place. Ultimately, in the future the shoe would need to meet certain requirements to maintain comfort while also collecting accurate data for the user to see. With more resources at our disposal, we could have worked around the unnecessary size and wiring of all the items within the shoe. We could build a holder for the battery within the outer sole of the shoe to allow for allow more room within the tongue of the shoe while also maintaining the safety of not ruining the battery. Using more advanced sensors within the shoes could help with increased accuracy while maintaining the battery usage of the Qi charging that is in place already. Another area that can continue to be worked on is application integration, with the little allotted time this past year designated to designing an app, there was not much that could be done to make sure everything worked properly. Going forward this could be resolved by further research or designing two separate apps within their respective frameworks of iOS and Android at least. On the student scale the design was able to complete all tasks necessary, but with some more industrial grade manpower and devices, this smart shoe could be the future with potential upgrades that could not be completed as our project.

Summary

Our team designed and built smart shoes that allow the user to keep track of multiple different areas of data useful to the user and their health. We were able to get a functional shoe without the mobile application paired completely but still presentable with multiple screens where the data will be displayed. Our user can track their steps, distance traveled, and speed over time whether they are running or walking. They are also able to check their weight, posture and massage their feet at any time throughout the day. This was very important because it allowed us to put multiple different devices that we found important for the user and their daily health adventures into one single device and showing that it was possible and works. Through much research and integration, we did just that and was able to implement all things we set out to including the wireless charging set up allowing the user to make sure they always stay prepared every day when they need the shoes.

Appendix

Source Code:

https://github.com/kckey/AssortedBytes-SmartShoeSystem-