CPEN 291
Project 2 Report

Title: **IntelliBike**

Lab section: L2B Group #: G18 Group's Lab-Bench #'s: 6C and 7C

Maya Painter	12439162	1/6
Thomas Broatch	23380165	1/6
Brennan Cathcart	24873168	1/6
Michael Muszynski	22756167	1/6
Joel Ritter	29844792	1/6
Zach Rivard	19544162	1/6

Contribution summary:

Maya: Maya wrote all the code and created the circuits for the second Arduino, which interfaced the high power LEDs, force sensitive resistor, photocell, and switches. She also organized the circuits for the security functionality, soldered components, and helped with testing the hall effect sensor.

Brennan: Wrote the code for the keypad, set up bluetooth on the Pi which included a script that ran on startup to open a rfcomm port. He also created a thread on the Pi that communicated with the Android phone, consistently edited and debugged the top-level code, designed the GUI for the web page, designed the transistor circuit for the LEDs.

Thomas: Thomas wrote the hall effect sensor code including distance/speed tracking. He was also responsible for the slave Arduino, communication between devices, and helped write the overall top_level code.

Zach: Zach wrote the mobile application and handled communications between the web server and mobile device. He also set up and maintained the server, as well as created the appropriate database structure.

Joel: Wrote the code to interface the accelerometer with the RPi and interface the altimeter / temperature sensor with the Arduino and tested both, wrote the code that allowed the RPi to communicate with the slave Arduino through the serial port and helped in writing the slave Arduino code. He also soldered components and did the fritzing for the RPi / Slave hardware.

Michael: Michael handled displaying and analyzing ride data which Zach stored in mySql onto the main webpage. To do this, he wrote createKMLwithID.php as well as the google maps api section in index.php. He wrote the framework for the top_level.py code, which is the code the Raspberry Pi runs. He also drafted most of the ideas related to the functionality of the bike and created the software and hardware systems diagrams for the bike. He implemented everything concerning the OLED screen connected to the slave Arduino and interfaced it with the Raspberry Pi through the serial port. He continuously helped debug software and integrate various modules together in top_level.py.

B. Introduction and motivations

The team initially was planning on designing a home security system but after further consultation, we realized that the market was saturated. We then settled on our backup plan to design a “smart bike” which would also incorporated features of the home security system. Many university students find biking to be the most efficient and affordable way to get to and from classes, but bike theft and accidents are unfortunately common. Our system addresses both safety and security concerns, as well as provides the user with helpful data and wayfinding for every ride.

The inspiration for tracking a cycling route came from Michael’s recent long-distance bike trek on an EuroVelo route in Brittany, France. In his research before the trip, he discovered a large community of passionate, long-distance cyclists committed to exploring the world by bike. On various forums, these cyclists would often post KML files (which can be plotted in google earth or google maps) in order to share their routes, show safe routes in a certain country, compare distances, or just to share a token of an accomplishment they felt proud of. These KML files were rather bland, however, and did not show much other then a track and possibly altitude change. The tracking functionality of the Intelli-Bike serves to breathe life into this widely-practiced act of amateur route tracking. It colors paths based on velocity, it lets a user

see his acceleration readings, velocity readings, distances, and altitude readings at any specific point in his track. It analyzes his data and supplies information about average speeds, top-speeds, elevation gains, and tracks his overall statistics for all of his bike journeys. In a final model of the Intelli Bike, it would be able to attach to any bike after a brief installation process and riders would have access to these personalized and easily shareable KML files for all their biking journeys.

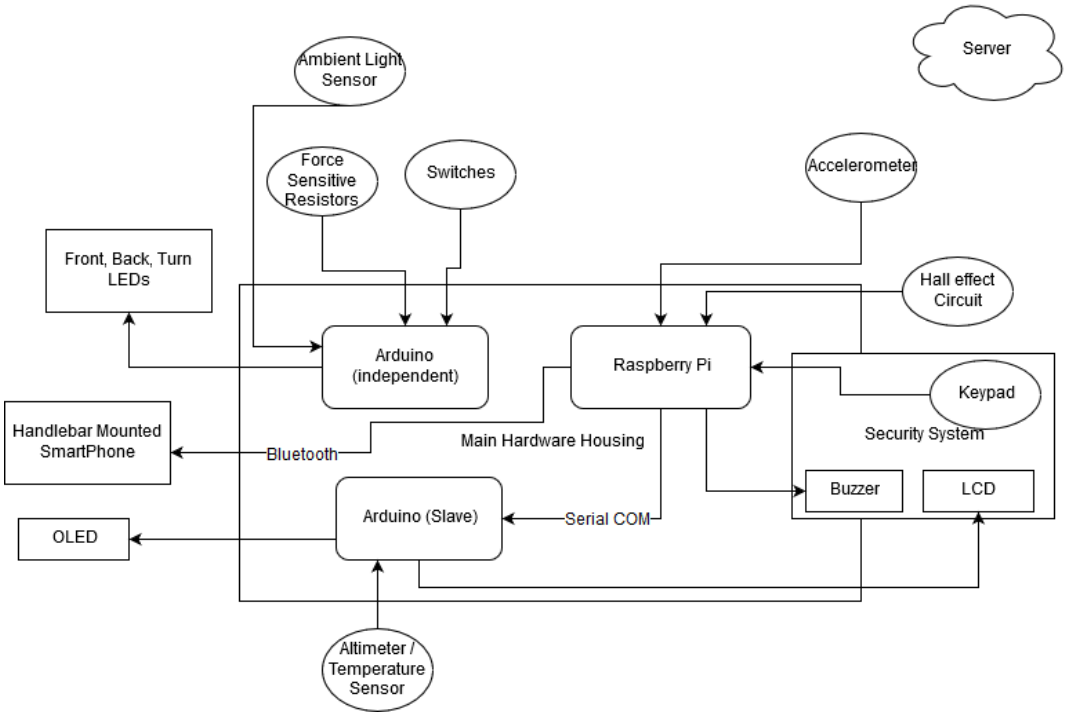
Cycling in the city can often be dangerous, with one of the primary issues being able to communicate with other road users. To increase the rider's safety, we attached switches to control two orange high-powered LEDs for the turn signals, a red high-powered LED brake light that would activate when force resistors detected the brakes were pushed, and a front white high-powered LED that would activate when a ambient-light sensor detected an unsafe level of darkness. Another danger of cycling is crashing, particularly in an isolated area. While these features were not incorporated at the time of the demo, the team planned for IntelliBike to address this issue by sending an automated emergency phone call and text-message to a chosen recipient when a crash is detected (accelerometer registers rapid deceleration).

Furthermore, with the most bike thefts per capita in Canada*[1], bikes are not safe from theft even if they are securely locked. Our system implements a second layer of security with an alarm, that the user can activate whenever they finish a ride. A loud alarm is triggered if movement is detected that only can be deactivated when the user enters the passcode. A planned functionality that was not implemented was to have the system notify the user via phone if the alarm is activated and for the user to be able to disable it if they wish to. In addition, the user would be able to arm and disarm the alarm remotely via app.

[1] https://biv.com/article/2016/07/vancouver-leads-way-bike-thefts-capita

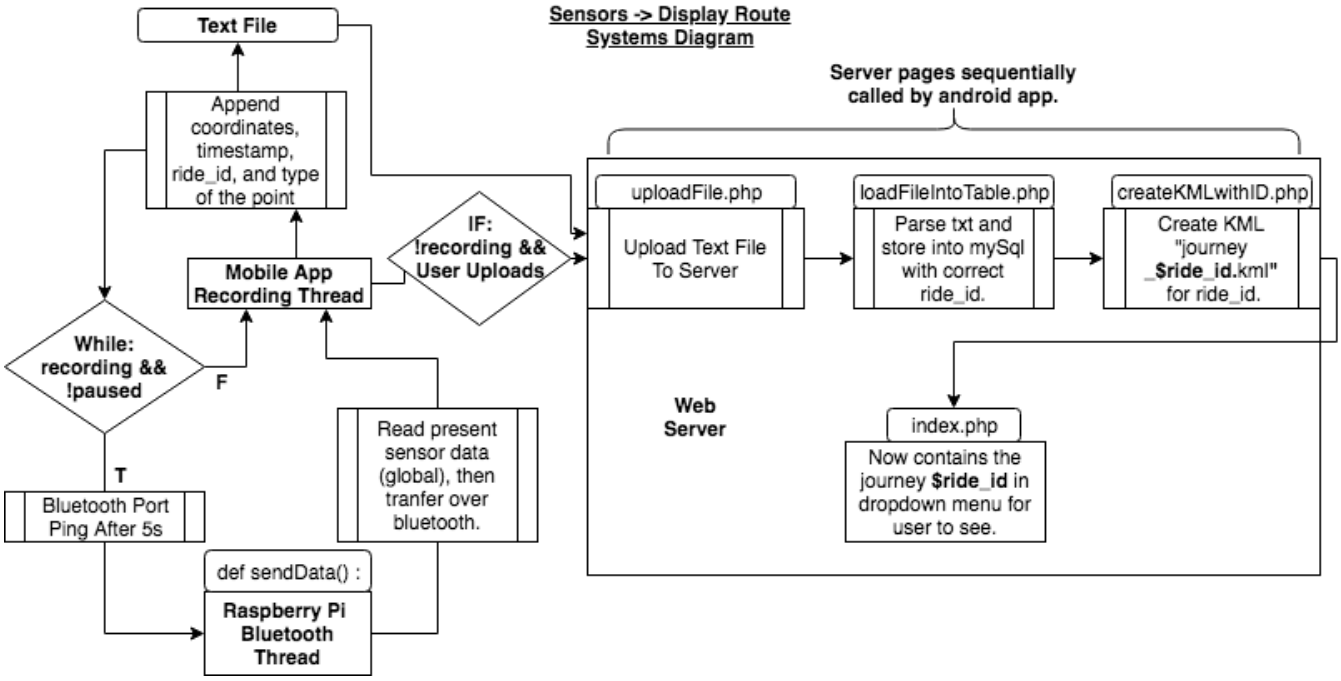
C. System diagrams

Hardware

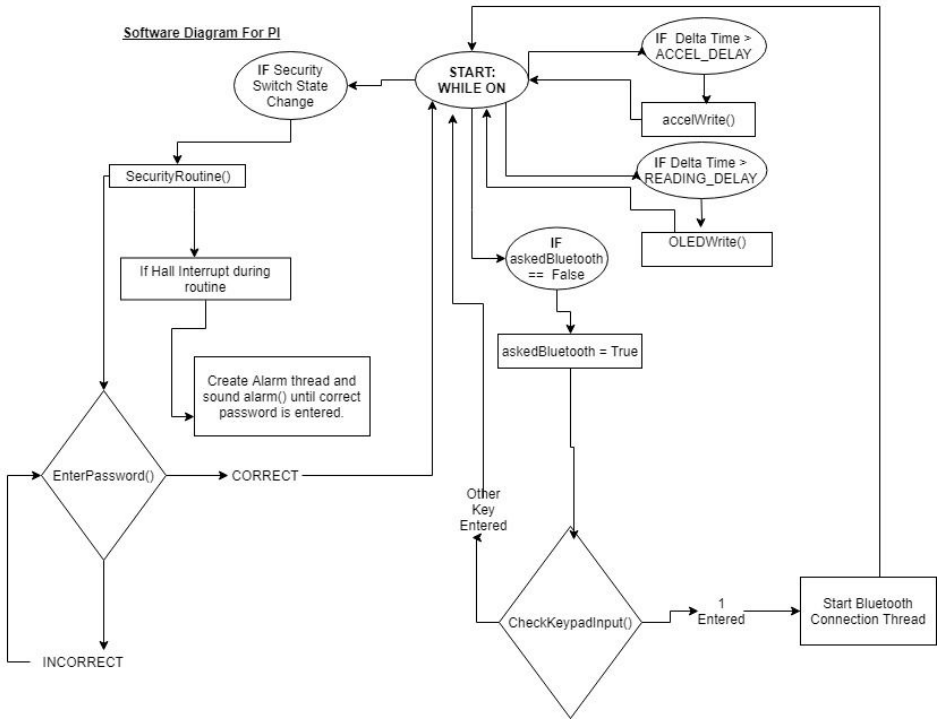


Software

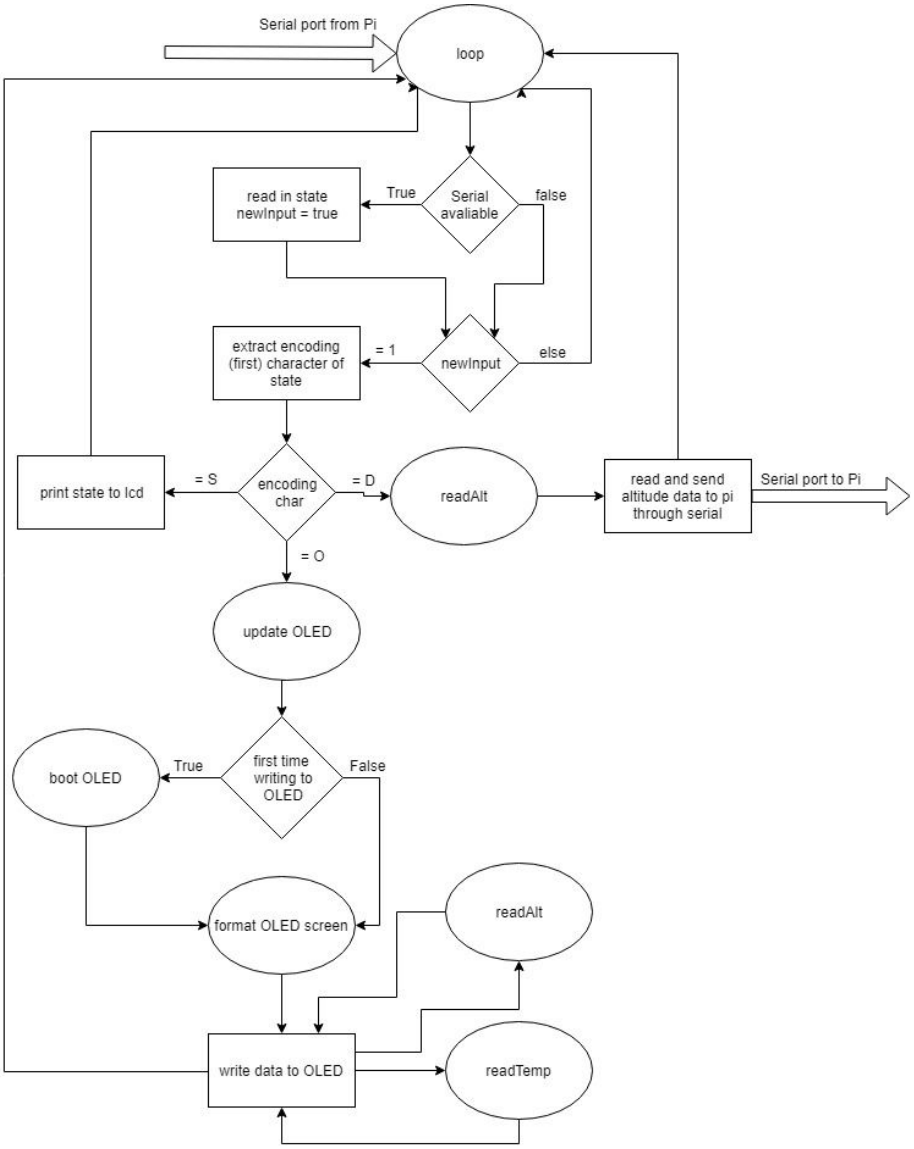
Server/app:



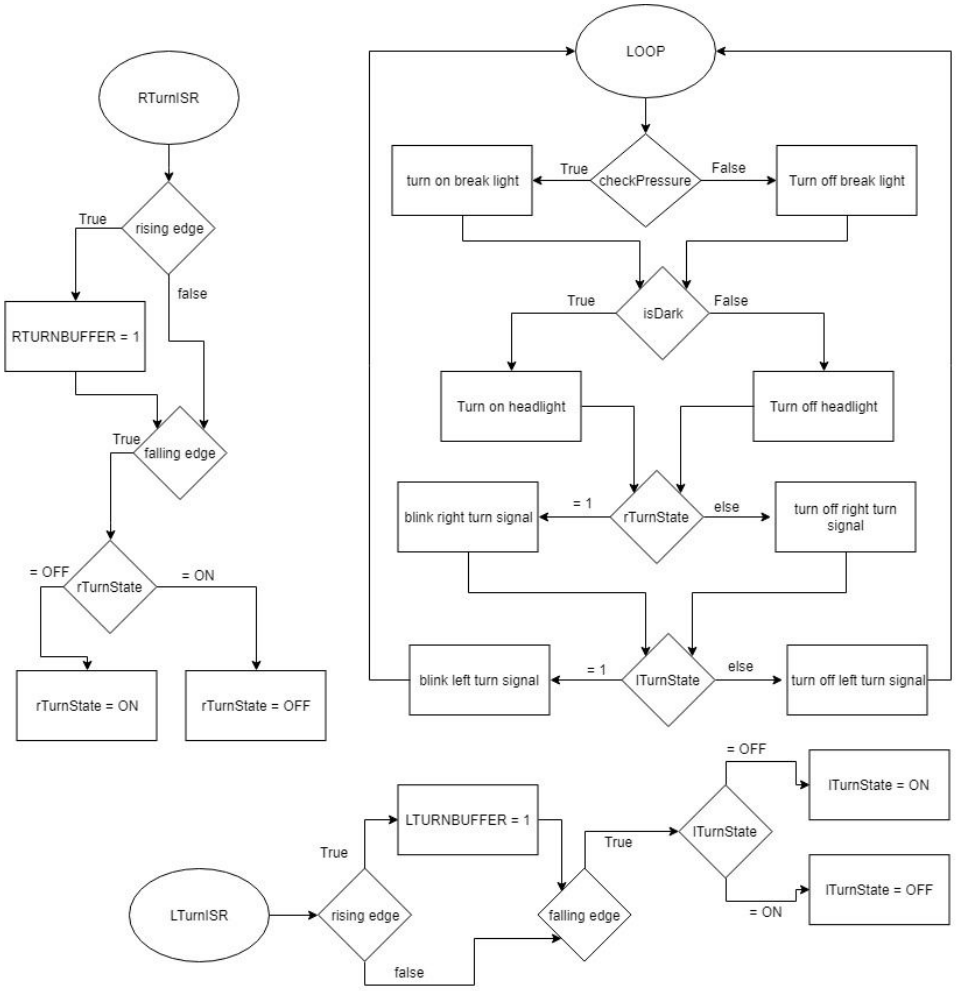
Raspberry Pi:



Slave Arduino:


Serial port from Pi
loop
read in state
newInput = true
True
Serial
avaliable
false
extract encoding
(first) character of
state
= 1
newInput
else
print state to lcd
= S
encoding
char
= D
readAlt
read and send
altitude data to pi
through serial
Serial port to Pi
= O
update OLED
boot OLED
True
first time
writing to
OLED
False
format OLED screen
readAlt
write data to OLED
readTemp

Safety System Arduino:



D. Project Description

Product Overview

The IntelliBike incorporates three main functionalities: a security system to prevent the bike from being stolen, a safety system to keep the rider safe on the road, and a data logging system to keep track of all the rides and rider statistics. By combining all of these systems into a single project, the IntelliBike aims to provide users with an enjoyable and safe alternative to driving and public transit.

Hardware

General Design

The Pi was our main device for our system which controlled the Slave Arduino and the alarm buzzer and took input from the accelerometer, Hall Effect Sensor and the Keypad. The Pi also interfaced over bluetooth with the mobile phone.

Although our intention was to primarily utilize the Raspberry Pi, we required a large number of pins to support both the LCD and OLED, so an Arduino was necessary. The RPi and the slave Arduino were connected by USB, and could therefore communicate through the serial port. The slave Arduino was responsible for controlling the LCD display and the OLED display and took inputs from the altimeter/temperature sensor and the Pi.

In addition, the OLED library required the majority of memory for a single Arduino. To accommodate for this, we designated a separate Arduino for the safety features which include high-powered LEDs and the hardware to support them (force sensitive resistors, switches, and photocell) on a seperate Arduino.

General Placement

Originally, we planned on placing the Pi and Arduino circuits on the frame of the bike as they would be closer to the hall effect sensors and force sensitive resistors. However, we realized that the hardware would be difficult to secure and hard to access. Thus the team decided the majority of the hardware would be placed in a waterproof box on the pannier rack and that wires would branch outwards to the components such as the OLED or LEDs that required external placement. The longer wiring distance was not a significant issue as we could cleanly tape busses of wires along the frame.

Since the bike we used did not have a functioning front brake, we only used a force sensitive resistor on the right brake lever. The force resistor was placed on the outside of the lever closer to the end of the bar as we found this was a natural place to make contact when braking after each team member tested it.

The two orange 1 watt LEDs (turn signals) were placed on each back corner of the pannier rack facing backwards. The turn signals are controlled by switches mounted on the bottom of the handlebars. We also considered putting turn signal LEDs on the front but we concluded that this was mostly an unnecessary cost as a rider can more easily communicate with traffic in their field of vision through hand signals and eye-contact. The red 1W LED (brake light) was placed directly between the two turn signals, and is controlled by a pressure pad mounted on the brake lever. The white 1W LED was placed at the front of the pike and acts as a headlight, using an onboard photocell to recognize ambient light and activate in dim conditions.

Hall effect Sensors

We used a hot glue gun to secure a strong magnet to one of the spokes of the wheel, and lined up the hall effect sensor with it. While we could have used more than one hall effect sensor, the difference in accuracy would be negligible due to the high speed of wheel rotations. Implementing more sensors around the wheel would require them to be exactly evenly spaced out, which would be challenging given the irregularity of the spokes. It was also necessary to surround the perf board for the hall effect sensor with electrical tape, due to the conductivity of the aluminum bike.

Security System

The hardware for the security system involves the same circuit as in the lab 2 keypad circuit, as well as the previously made LCD circuit. In addition to the keypad and LCD, there is an alarm mounted on the rear of the bike that will sound if the Pi detects wheel rotation without disarming the system by entering the appropriate passcode. To make the alarm loud enough to act as a viable security measure, the alarm is attached to an op-amp in order to amplify the volume output by the alarm. The op-amp used is said to give 10x amplification, and the resulting alarm is appropriately loud as a result.

Software

Security System

Since the alarm would trigger whenever the hall effect sensor would read if the alarm was armed, the hall effect function needed to be an interrupt. To disable the alarm, the user would enter the required passcode, which would allow them to start riding.

The keypad inputs were verified by turning each column on, then checking each row to locate which key was pressed. If the correct key was pressed the corresponding element in the correctDigitEntered array was set to true and false otherwise which would be checked for correctness after four digits were entered. If the passcode was correctly entered, the security system would be disarmed and the system would be in “riding mode”. To arm the switch, the status of the push button pin was constantly checked. If the status changed, then the security would be armed.

Measuring Speed

We measured the circumference of the wheel and the time taken between hall effect readings to calculate the speed of the bicycle. We chose to only display the speed every couple of seconds rather than every rotation, as the wheel usually spins several times per second making it tough for the user to read a constantly changing value.

Another consideration was how to detect wheel stopping. Since the speed is calculated by the time taken of the most recent rotation, if the wheel stops, the hall effect sensor will not read. To solve this, if during ride mode a certain amount of time passes without a magnet reading (after testing we found around 4.5 seconds was optimal), the wheel speed will record as 0. This concept is inline with most commercial speed detection sensors, which generally require a certain threshold speed to record.

Arduino-Pi Communication

Both the Arduino and the Pi can send and receive data. The Pi would send data to the Arduino to display security system data on the LCD such as current password being entered and whether password was correct. The Pi would also send riding data to display on the OLED, such as speed, and distance traveled. This was done with the serial.write function, which would send a string to the Arduino. The first character of the string was used to determine the function of the message, for instance the first character ‘S’ indicates the lcd should be cleared, then print the message. If the first character was ‘D’, the Arduino sends data to the Pi from its sensors using the Serial.print function, which gives a String for the Pi to read. The Pi code formats the string using the String.decode() function, then sends the data to the phone via bluetooth. To ensure the messages don’t become jumbled, we added the ‘\0’ character at the end of each

message so that the Arduino could differentiate between messages in the serial port's input buffer.

Mobile App

The mobile handles all of the internet requests for the system, and manages all of the data that is gathered during the ride. It is responsible for taking the user's desired start and end locations for a ride and plotting an appropriate course. This is achieved with the help of the Google Maps and Directions API. Upon obtaining a reply, it draws the desired route on the map and begins a logging cycle. The app periodically signals the Pi to transmit its most recent speed, distance, acceleration and altitude, and pairs it with its own location and timestamp to complete a log entry. Upon completion of the ride, the user can choose whether or not to upload the data to a remote server. If the user chooses to do so, the app uploads the log file and deletes it from memory in order maintain a low memory footprint on the device. The app may also send/receive special requests to the Pi in order to change configuration settings or to call for emergency help.

Web Server

The server was built using Apache and is written in a combination of HTML, CSS, and PHP. The server hosted a mySQL database which was used to maintain records of all files uploaded to the web service. The primary goal of the web service was to provide users with a method to access their past bike rides. The main web page, index.php outlines a window that contains a Google Maps map, a drop down menu of all past rides for the user, and a statistics table that would display a user's all-time stats. In order to render the Google Maps overlay, a KML file had to be produced for each specific ride. The process for going from immediate sensor readings to a route is illustrated in the software systems diagram illustrating our usage of the server.

E. Test, Evaluations and Challenges

One of the biggest challenges of this project was learning new languages. Between learning how do what we wanted in python and learning html/kml from scratch, a lot of time and effort was spent researching and figuring out how these languages work.

Hall effect Sensors

We started by testing the hall effect sensors in their own separate file, printing out whether the sensors were detecting a magnet. Next, we mounted them to the bike, making sure to orient the magnets properly before, as we found that the sensor was not strong enough to sense a magnet facing away from it. We then integrated the hall effect code with the security system and attached the sensors with the rest of the circuitry, and used the same testing procedure as before they were attached. During testing we found the software was reliable, however the ground wire would occasionally become unplugged.

Security System

The security system had several components which all needed to be tested individually and integrated together.

The first component was the keypad, which is used to enter the passcode. One challenge we had was ensuring that holding down the key would not enter a digit more than once. This was solved by running a while loop after the key was pressed, which would wait for the key to be released before inputting the data. There were three main test cases that needed to be covered: checking that entering the correct password would disable the security system and enable riding mode, entering the incorrect password would prompt the user to enter the password again, and if the alarm sounded, entering the correct password would disable the alarm.

The second component was the push button, which would arm the security system if in riding mode. Since it works by checking if the input pin has changed state on each while loop, one issue we had was using too high of a resistor value in the button circuit. This was tough to debug since the pin would always read “off”, which led us to believe there was a wiring issue. Another mistake we made was updating the previous state of the button before entering the security routine, which meant that if the user accidentally pressed the button during the security routine, the code would detect a change in button state and immediately rearm the system. This was solved by updating the most recent state of the button immediately after disarming the security system.

The third component of the system was the alarm, which would turn on the buzzer if the hall effect read while the security system armed. Since there was little software for this component,

the primary issues were with hardware. We found that the buzzer was not loud enough to be effective when connected to only the RPi, so we attempted to amplify the output with an op-amp. Using knowledge gained in our electronic circuits class we set up the op-amp to amplify the input signal approximately 10 times, which needed to be wired carefully so that it did not accidentally trigger the alarm.

High Power LEDs and Associated Switches/FSRs/Photocell

The LEDs would often unexpectedly turn off at times because the connections between the wires and LED terminals were not strong/secure. Eventually the team added a substantial amount of solder to the connections which fixed the problem. In addition, in order to fit everything in the casing, we mounted the LEDs circuit vertical on the side of the case. This caused many connection issues as every time we fixed one wire connection another wire seemed to unplug. To solve this issue, we carefully added some glue to the connections which mostly fixed the problem. If the LEDs did not light up as expected when triggered we found that printing to the serial port was the fastest way to identify the source of the problem. By printing digitalRead() values of the switches, analogRead() values of the force sensitive resistor (FSR), and setting all the LEDs to high, one could see if the problem was due to the triggering device reading incorrect values or the LED not being connected properly. We also intended to have the FSR taped securely to the brake lever, but this caused false positive readings from the sensitive device. We needed to keep the pressure pad untaped to prevent this, and as a result was left exposed and not held exactly flush against the bike.

Altitude/Temperature Sensor and Accelerometer

Testing the Altitude / Temperature Sensor mainly consisted of connecting it to the Arduino and using the Arduino library code to print the values to the serial monitor. Once we were content with the values it was returning, we modified the code to work with our slave file. We had challenges getting an accurate reading from the altimeter until we realized that we had to give it the current local sea level pressure to use as a base line. Testing the accelerometer was fairly similar, except there was no Python library for it, we modified an initialization code we found online and added the functionality that we needed. When printed accelerometer values to the screen we initially believed that we had connected it wrong, since we were getting acceleration in the X, Y, and Z direction at all times. After some research we discovered that gravity needs to be accounted for in the chip that we were using. By taking the square root of the sum of the

squares if X, Y, and Z, we found the acceleration to be 9.8 exactly when the bike was not moving. We dealt with this by subtracting 9.8 from all calculated acceleration values. We then tested printing it to the Pi's monitor, and then tested importing it into our Top Level code to interface with the rest of the project.

Arduino-Pi Communication

One challenge with the Arduino-Pi communication was due to the serial.write and serial.read between the Arduino and Pi we could not use the serial monitor for debugging. Instead, we needed to use the lcd screen as a serial monitor. However, since we occasionally had wiring errors in the lcd, if nothing was displaying we would run a lcd test file first, which would simply call some lcd.print() statements. Since the message sent from the Pi to the Arduino consisted of one encoding character at the beginning, one test we ran was to print out the encoded character and also run a print statement inside each of the encoding if statements to ensure the Arduino decoded the character from the Pi correctly.

The Arduino also sent altimeter data to the Pi, which was much easier to test as we could run a serial monitor. We simply printed out the altitude received from ser.read() on the Pi to ensure the data was being read correctly.

Mobile App

The mobile app was tested in stages, with each of the stages building on the work of the previous stages. Having worked with Bluetooth in Project 1, it was simple to transfer to communicating with the Pi. The testing of new functionality began with implementing the Google Maps and Directions API, and testing that it was able to provide a suitable route given the the start and end locations. The next stage of testing revolved around making HTTP requests to the web server to execute simple PHP scripts. Once proper communications were established to the server, sending files over HTTP requests was the final hurdle for the app, which it is successfully able to do.

Web Server

Testing the web server comprised mainly of writing various mixes of HTML, PHP and CSS to create a page that was not only visually appealing but was able able to perform all of the

required tasks (file uploads and mySQL operations). The web page GUI was tested mainly through trial and error in order to get everything aligned and looking nicely.

F. References and bibliography

Sources:

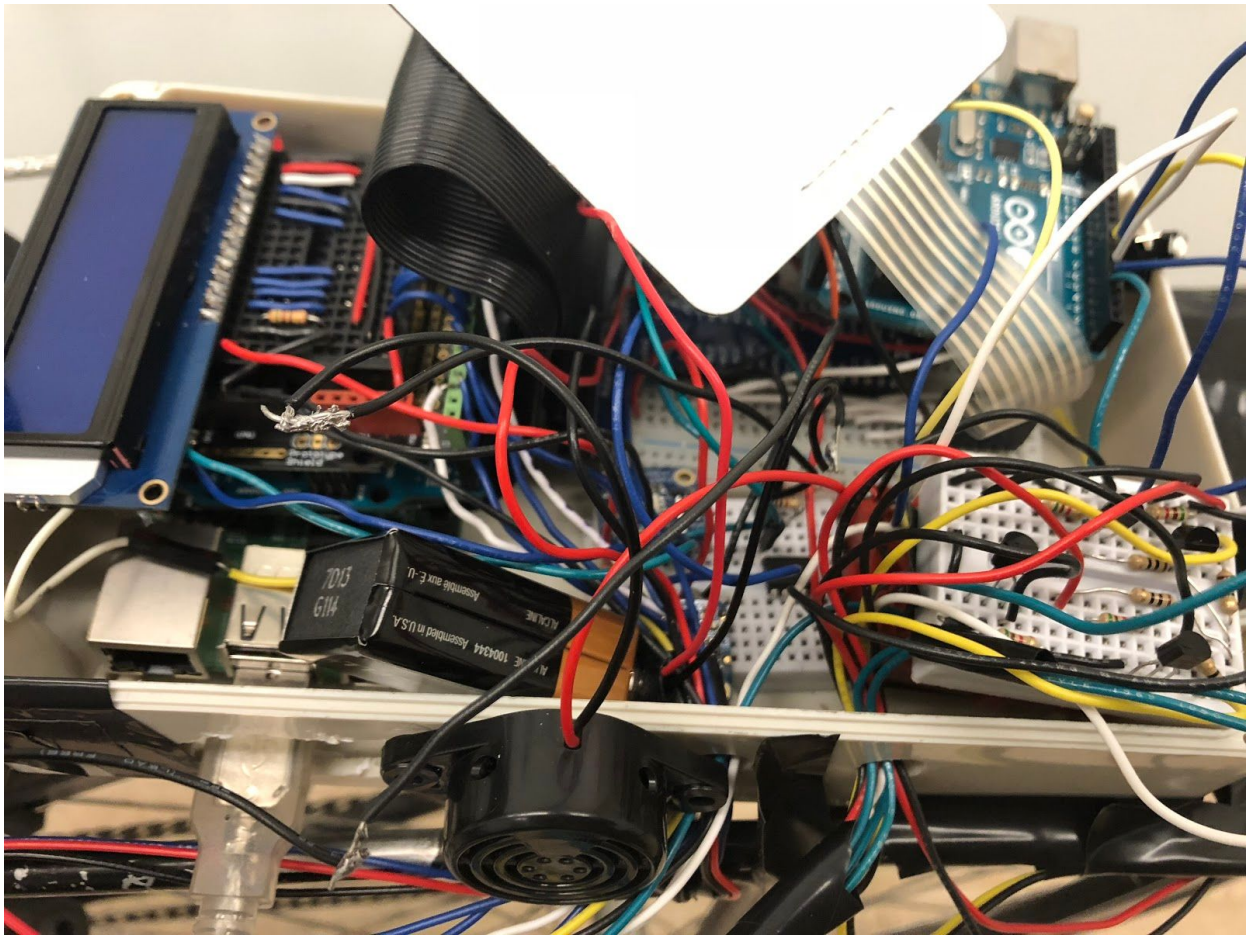
- Draw.io: Tool for system block diagrams
- https://github.com/massixone/mma8451: Base for our accelerometer code
- https://www.allaboutcircuits.com/technical-articles/using-interrupts-on-Arduino/
- https://oscarliang.com/connect-Raspberry-pi-and-Arduino-usb-cable/ : Tutorial to connect Rpi to Arduino via serial port
- https://developers.google.com/maps/documentation/javascript/kml: For displaying kml files on the server
- https://developers.google.com/maps/ : Google Maps API Android
- https://developers.google.com/kml/documentation/kmlreference?csw=1#colorstyle : How to color map in kml
- http://www.kmlvalidator.com/home.htm : Helped to validate KML files

Files:

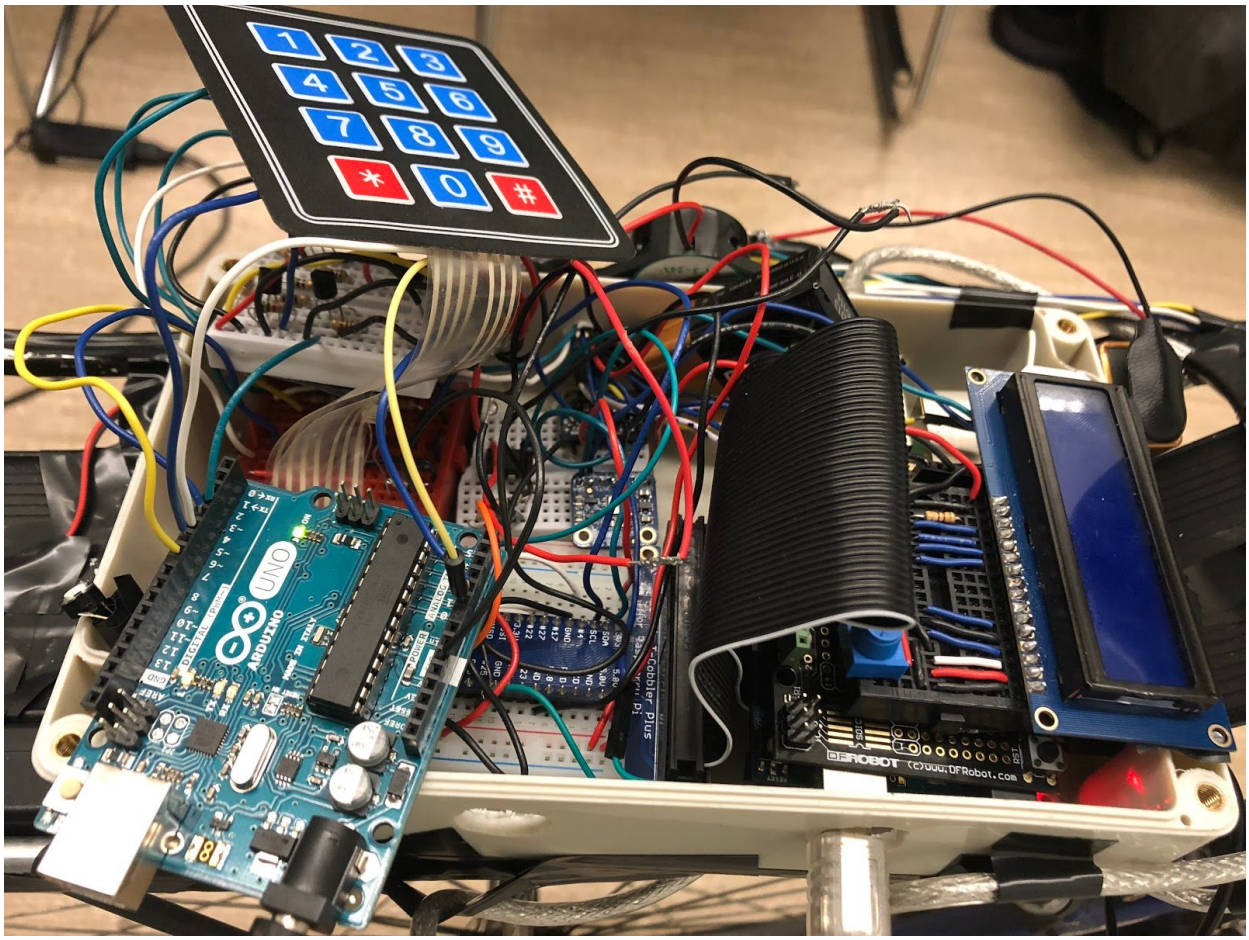
- Top Level.py: Main code running on the Pi
- MapsActivity.java: Main mobile app code
- ScriptExecutor.java: HTTP client for mobile app
- index.php: Launcher script for the webpage
- loginPhone.php: Contains login information for the database
- uploadedFilesInfo.php: Contains information as to where to store the uploaded data
- uploadFile.php: Accepts HTTP file transfers and copies the file to the uploads/ folder
- loadFileIntoTable.php: Parses the uploaded files and places all of the values into the database tables
- createKMLwithID.php: Creates a KML file based off of the passed ride_id, and places it in the kml/ folder
- Arduino_Slave.ino: code running on the slave Arduino
- MMA8451_code.py: code for the Pi to interface with the accelerometer
- Arduino_Safety_System.ino: code running on the other Arduino that controlled the LEDs
- __init__.py: code to allow the Top Level code to import the MMA8451_code file
- Project2_fritzing.fzz: fritzing file for the entire project

Appendix A1 – Project pictures (hardware)

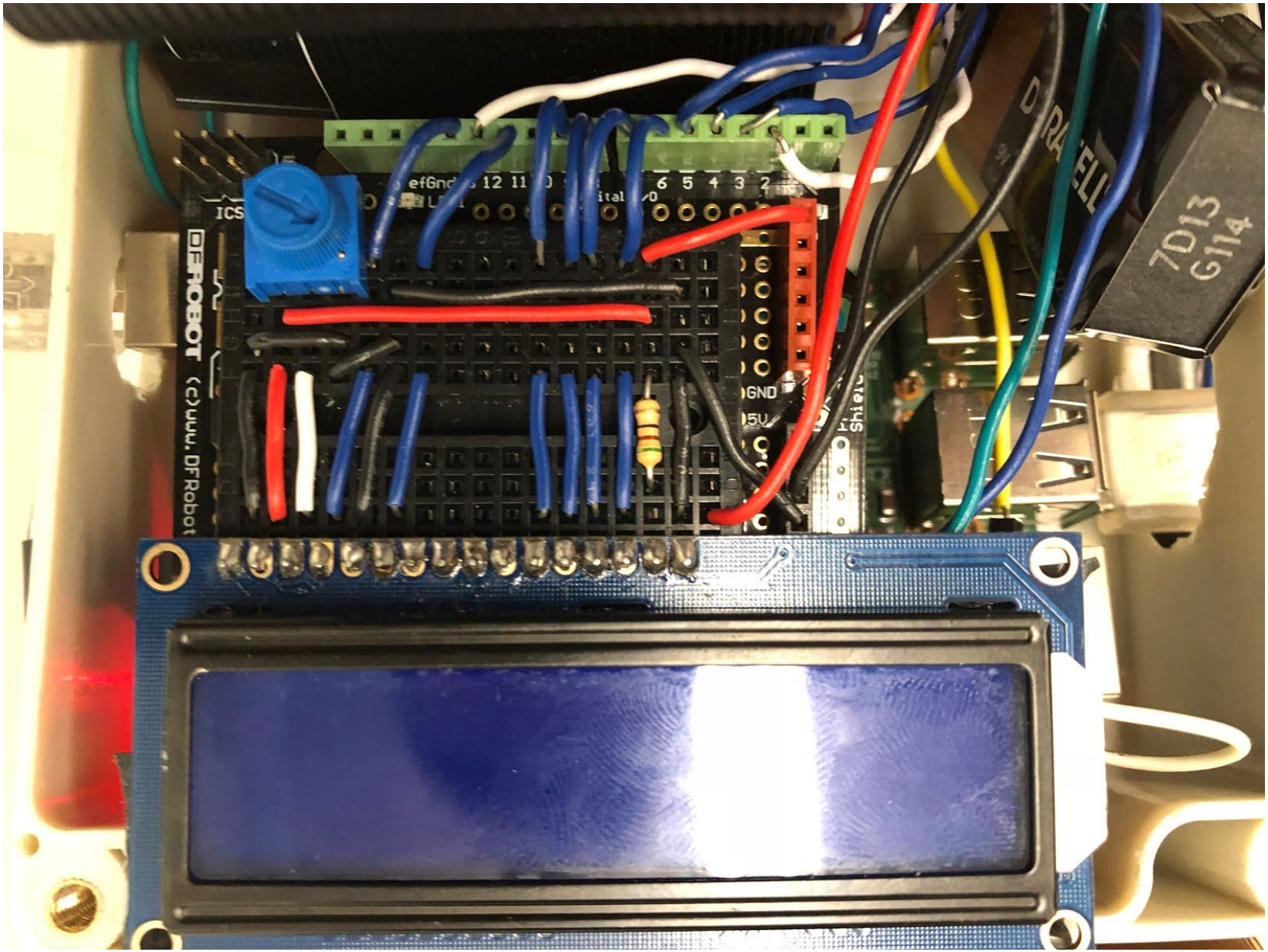
Hardware casing Left:



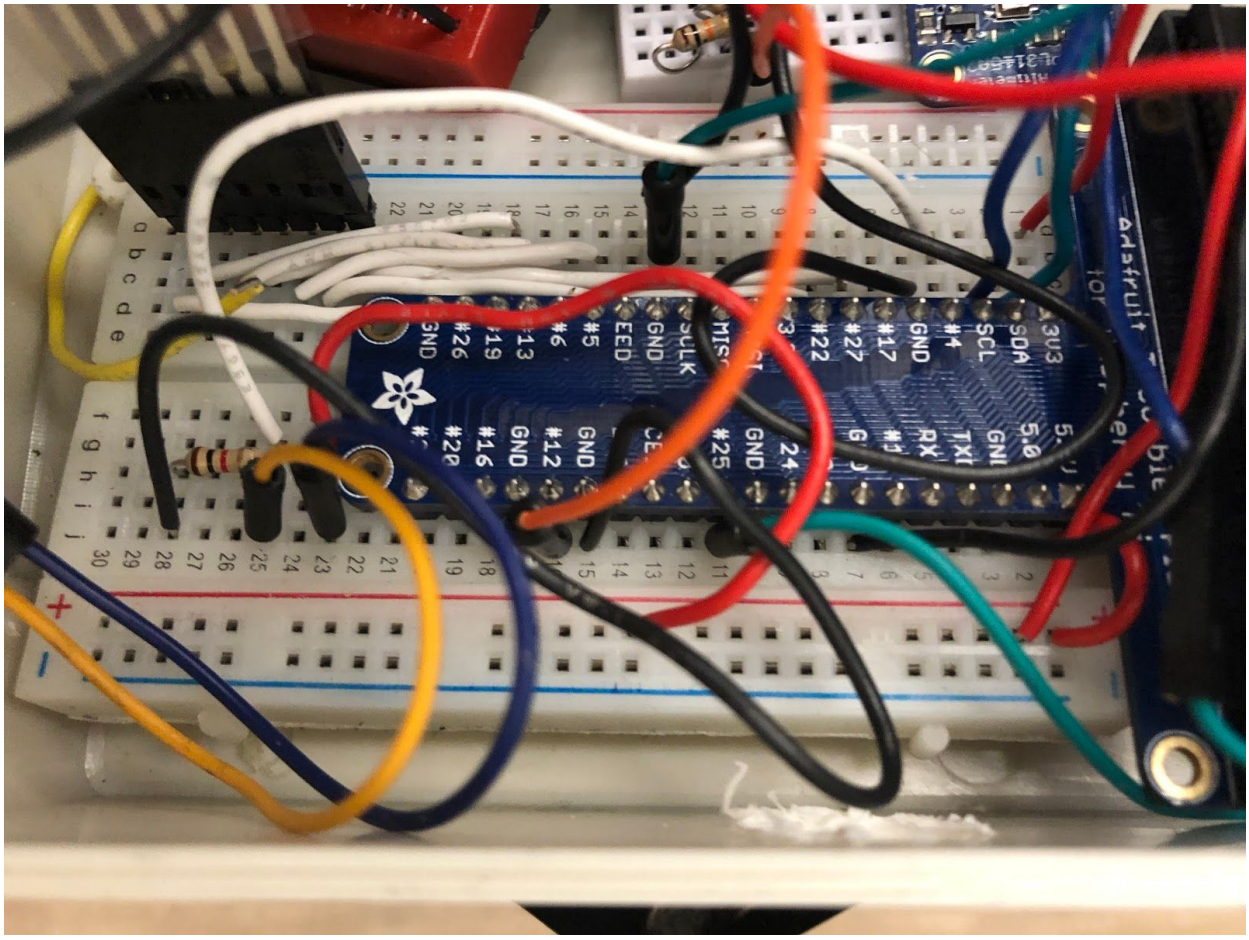
Hardware Casing Right:



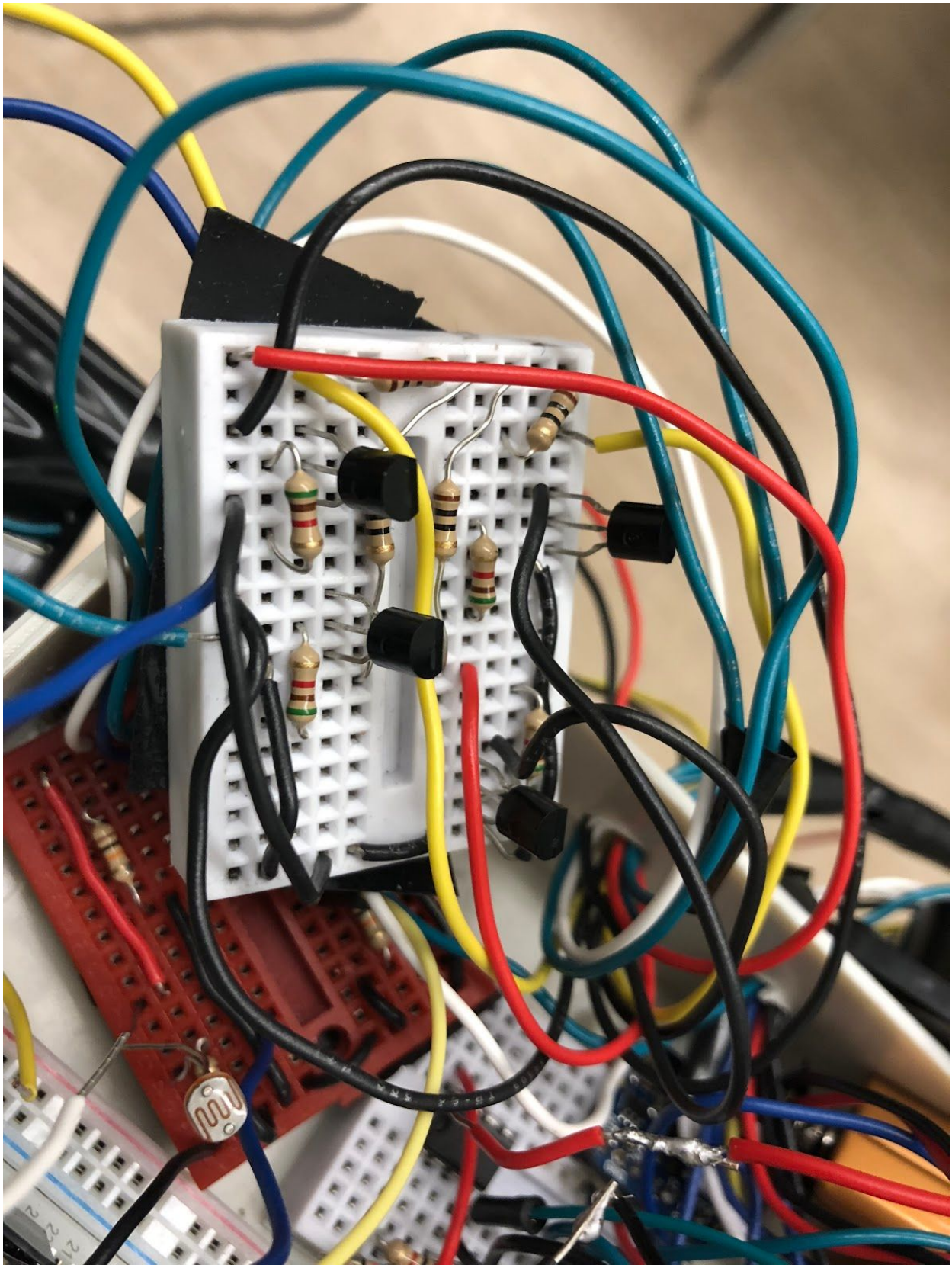
LCD circuit


DFROBOT (c)www.DFRobot
GND
5V
7D13
G114

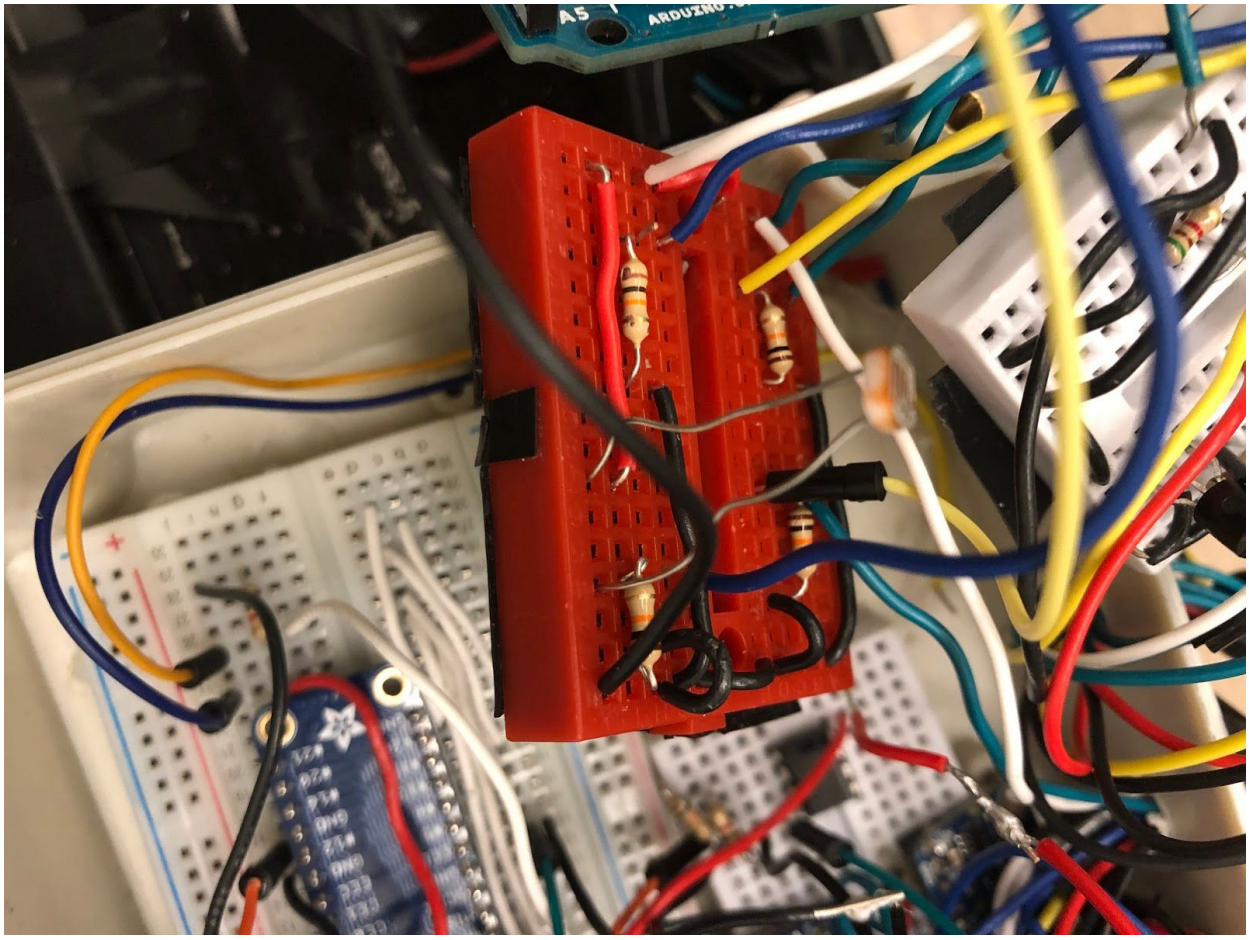
Raspberry Pi cobbler connections:



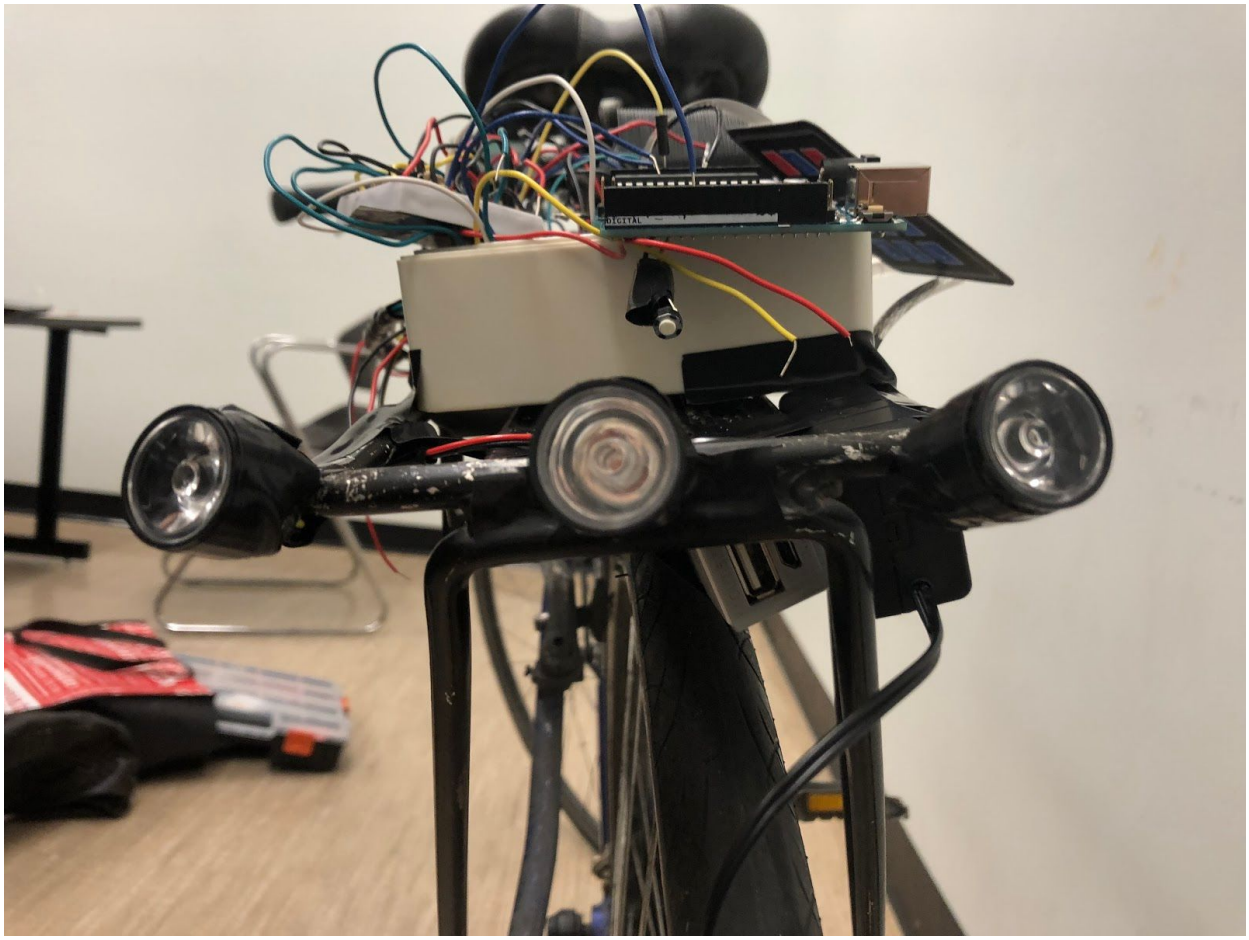
Transistor current amplification circuit for high power LEDs



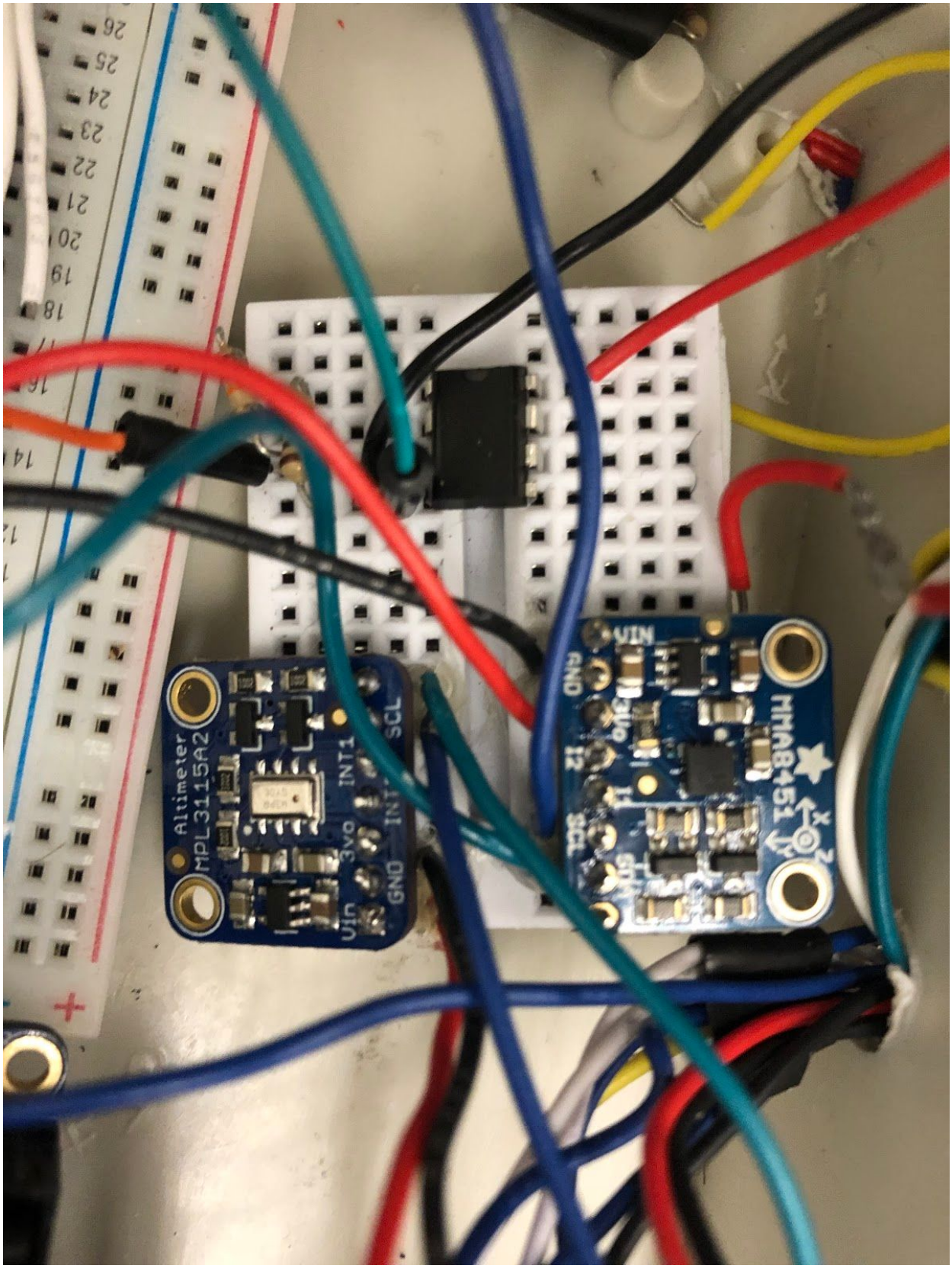
High power LED control bread



Brake lights and turn signals:



Altimeter and accelerometer breadboard:



Main bus to handlebars:

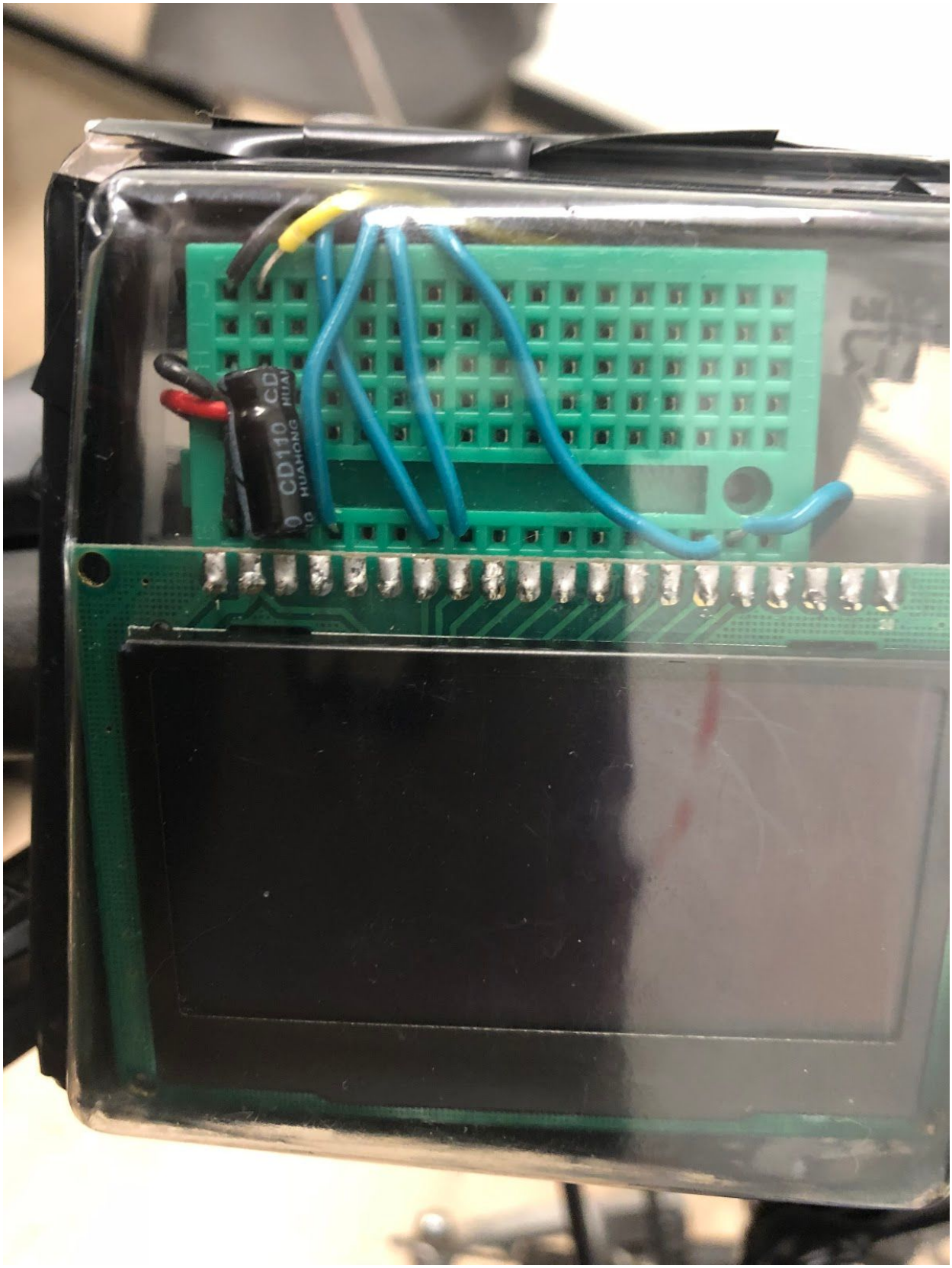
Hall effect sensor and magnet configuration:

Front automatic headlight:

Handlebar turn signal toggle (left side)::

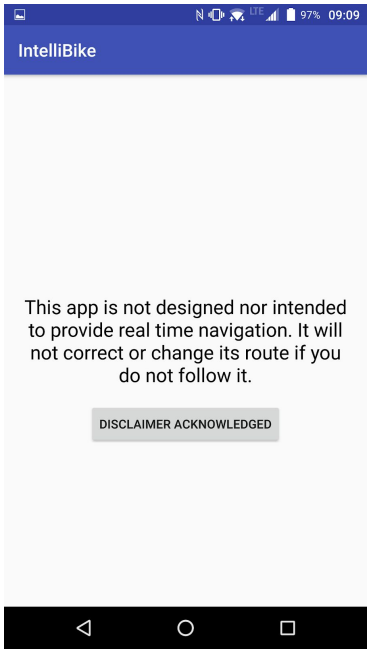
Force sensitive resistor for brake light control:

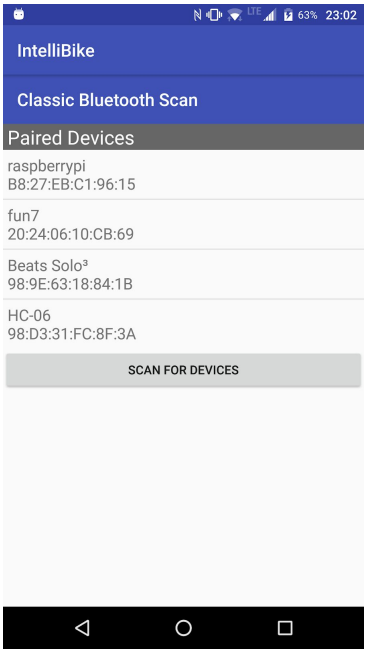
Handlebar mounted OLED display:

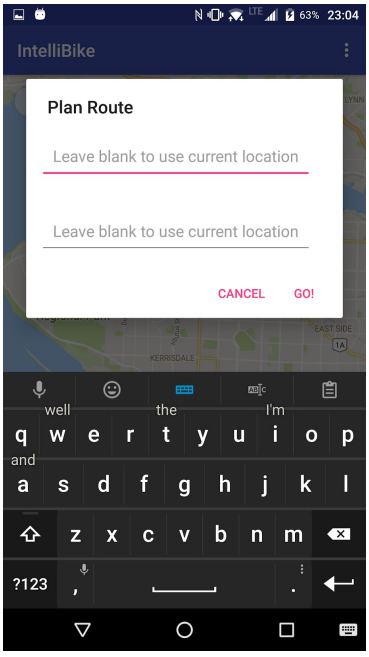


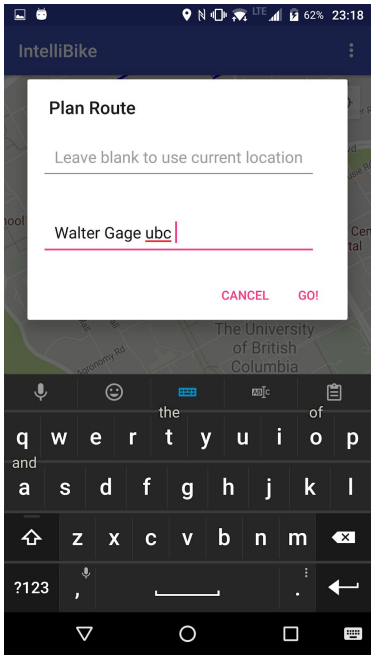
Appendix A2 – Project pictures (software)

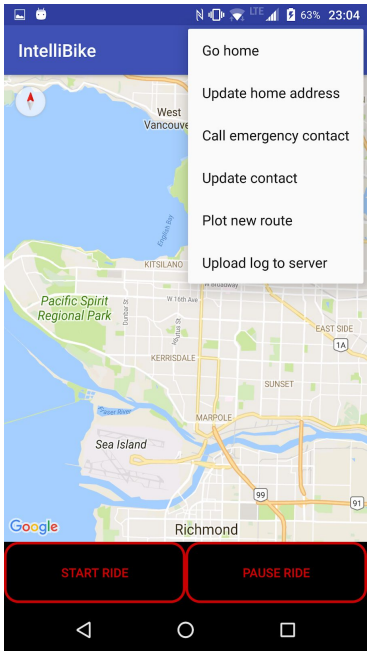
Include clear images/snapshots of the user-interface for software (Web, app…).

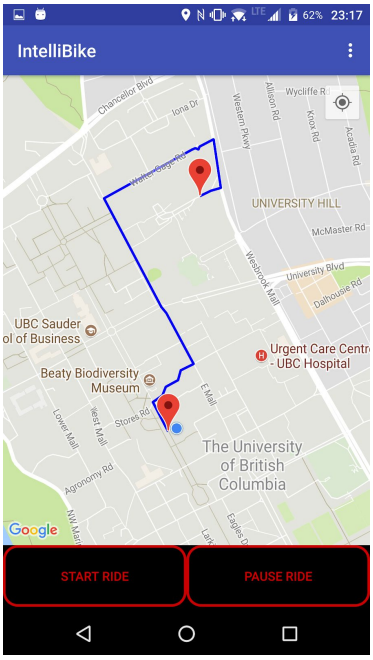

IntelliBike
This app is not designed nor intended to provide real time navigation. It will not correct or change its route if you do not follow it.
DISCLAIMER ACKNOWLEDGED

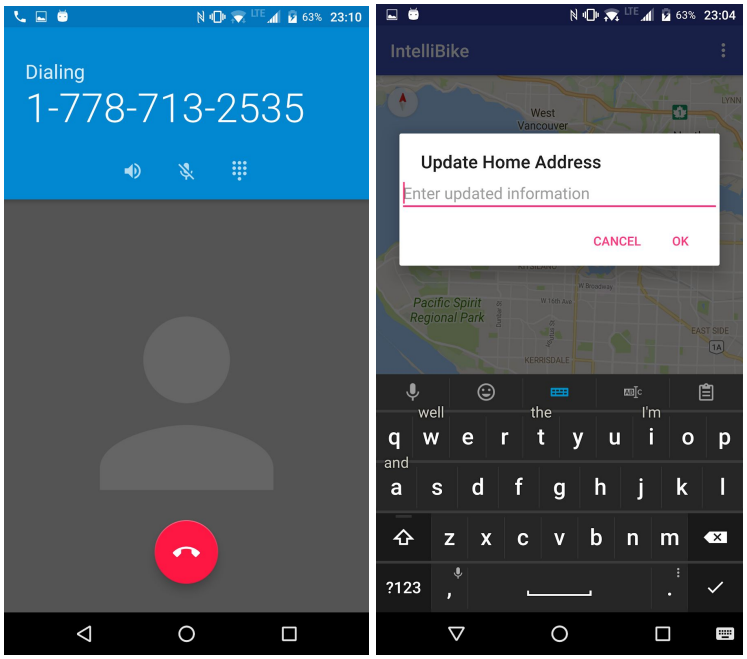

IntelliBike
Classic Bluetooth Scan
Paired Devices
raspberrypi
B8:27:EB:C1:96:15
fun7
20:24:06:10:CB:69
Beats Solo³
98:9E:63:18:84:1B
HC-06
98:D3:31:FC:8F:3A
SCAN FOR DEVICES


IntelliBike
Plan Route
Leave blank to use current location
Leave blank to use current location
CANCEL
GO!

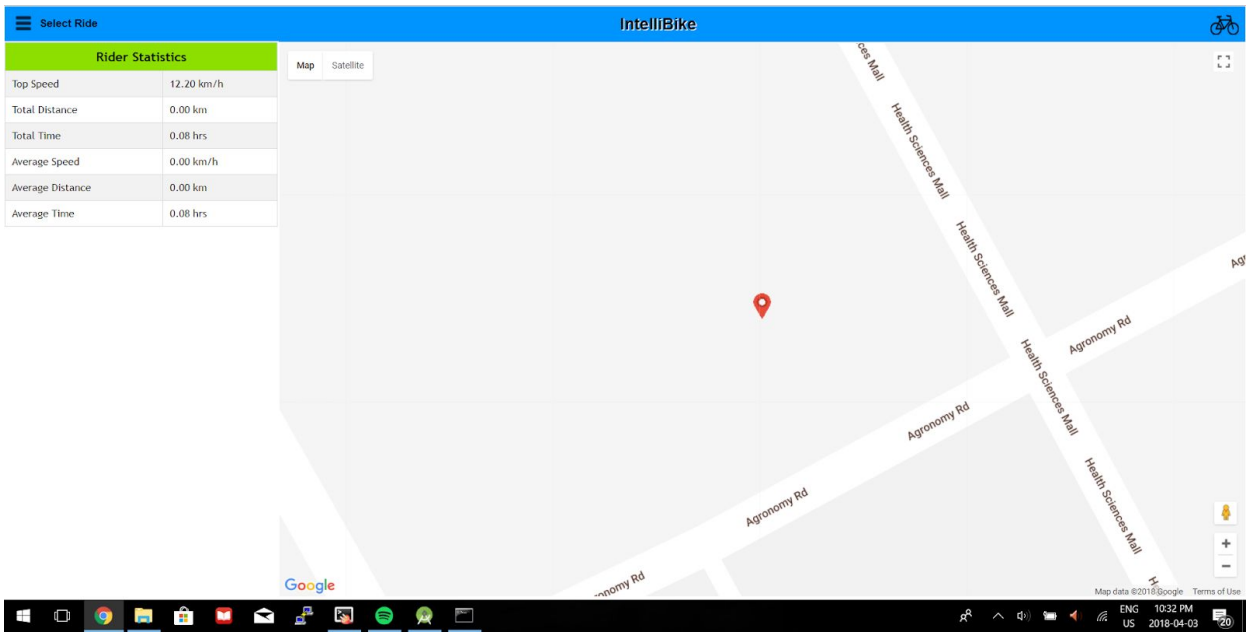

IntelliBike
Plan Route
Leave blank to use current location
Walter Gage ubc
CANCEL
GO!


IntelliBike
Go home
Update home address
Call emergency contact
Update contact
Plot new route
Upload log to server
START RIDE
PAUSE RIDE

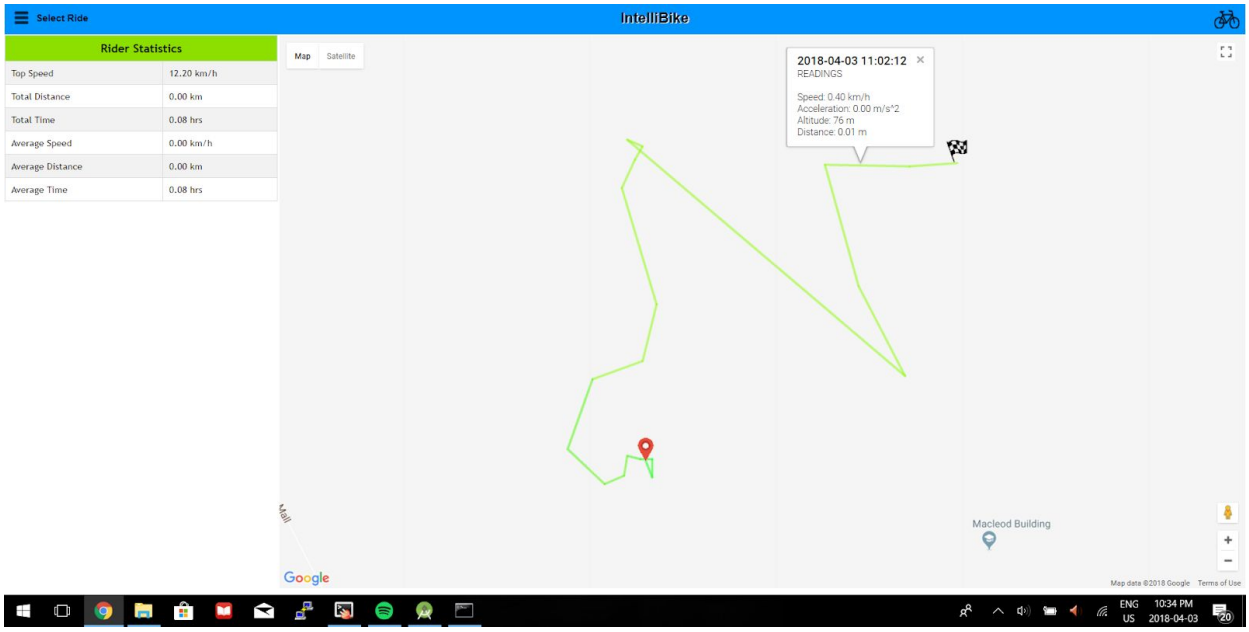

IntelliBike
START RIDE
PAUSE RIDE



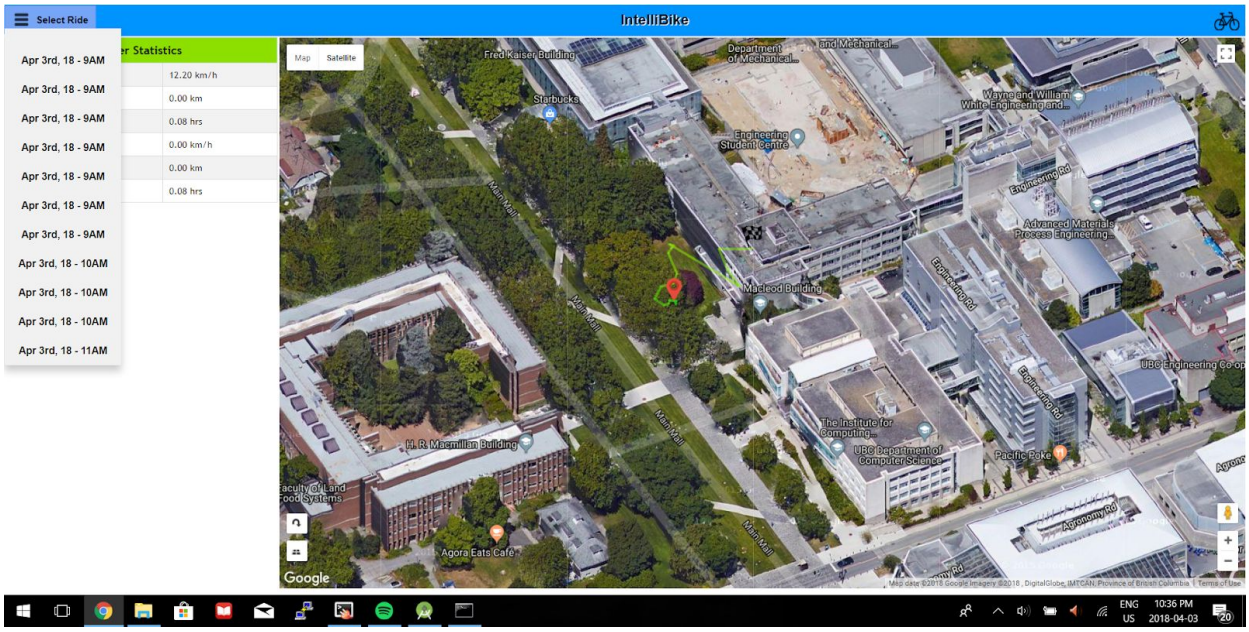
Mobile App Progresssion



Default Web Page



After Selecting a Ride (This ride wa inside of MCLD hence why it appears like there is no map)



Same ride as previous, but with satellite view and the dropdown menu clicked

Appendix B – Component list

Part #	Description	Quantity used
US5881	Hall Effect Sensor	1
	Arduino Uno	2
TC1602A-09T	Liquid Crystal Display	1
	Raspberry Pi 3	1
MMA8451	Triple-Axis Accelerometer	1 128x64 OLED
1528-1136-ND	Keypad Switch	1
296-14997-5-ND	IC OPAMP JFET 4MHZ 8DIP	1
	Photocell	1
	NPN Transistor	4
Purchased Components		
SSD1305	Monochrome 2.42"	1
	BUZZER 3-24V DC CONTINUOUS	1
MPL3315A2	I2C Barometric Pressure/Altitude/Temperature Sensor	1

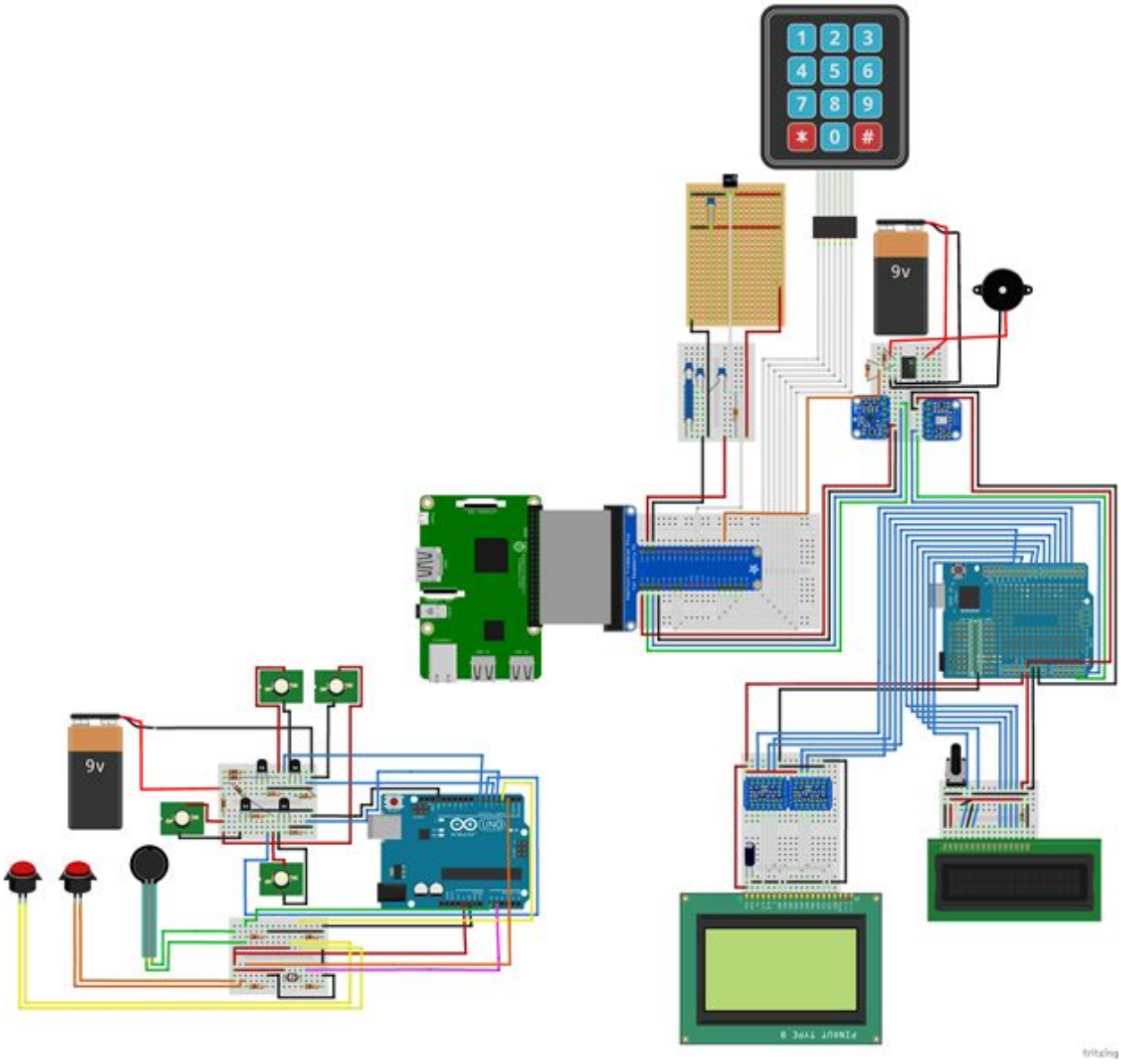
If you have acquired or purchased any component for the project, please include the complete info:

Item: Monochrome 2.42" 128x64 OLED
Vendor: Adafruit.com
Purchase Link: https://www.adafruit.com/product/2719
Link to datasheet: https://cdn-shop.adafruit.com/product-files/2719/2719+DATA.pdf
Price: $39.95 USD

Item: BUZZER 3-24V DC CONTINUOUS
Vendor: Lee's Electronics
Purchase Link:
https://leeselectronic.com/en/product/4133.html?search_query=buzzer&results=58
Link to datasheet: N/A
Price: $3.65 CAD

Item: MPL3315A2 Barometric Pressure/Altitude/Temperature Sensor
Vendor: Adafruit.com
Purchase Link:
https://www.adafruit.com/product/1893https://leeselectronic.com/en/product/4133.html?search_query=buzzer&results=58
Link to datasheet: https://cdn-shop.adafruit.com/datasheets/1893_datasheet.pdf
Price: $9.95 USD

Appendix C - Fritzing



Appendix D – Raspberry Pi 3 Code

```
# ====================== SERIAL PORTS / SETUP ========================
import serial
from time import sleep
#setup serial port
ser = serial.Serial('/dev/ttyACM0', 9600)
sleep(2)

# ====================== GLOBAL VARIABLES AND CONSTANTS
========================

NAME_OF_BIKE = "IntelliBike"

#========== GPIO PINS ================
#Keypad Pins
ROW0 = 26
ROW1 = 19
ROW2 = 13
ROW3 = 6
COL0 = 5
COL1 = 22
COL2 = 27
ALARM_ARMING_SWITCH = 4
HALL_P = 25 #Hall effect
BUZZER_P = 12 #Buzzer Pin
#========== KEYPAD ================
DIGIT_1 = 1 #Password digits
DIGIT_2 = 2
DIGIT_3 = 3
DIGIT_4 = 4

passcode = [DIGIT_1, DIGIT_2, DIGIT_3, DIGIT_4] #Array of passcode digits
correctDigitEntered = [-1, -1, -1, -1] # array that will be populated with
true/false as passcode is entered
NUM_DIGITS = 4 #digits in passcode
insideKeyInterrupt = 0

#========== HALL EFFECT ================
NUM_MAGNETS = 1 # number of magnets on wheel
WHEEL_DIAMETER = 70 #Diameter of bike wheel
wheelTime = 0 #Updated by interrupt
lastPrint = 0

#========== ALARM ================
#Updated by switch on casing polled in sensorRead().
ALARM_ARMED = 0
ALARM = 0
```

```
ALARM_BUFFER = 0 #was not used in our demo
ALARM_BREAK = 0

#========== OLED ================
last_distance = -1
stopcount = 0 #increments on consecutive writes to OLED without reading hall
effect
last_OLED_write = 0 #time of last write to oled
READING_DELAY = 0.5
#number of consecutive OLED writes without a hall reading before declaring bike
as stopped
STOP_THRESHOLD = 4

#========== ACCELEROMETER ================
ACCEL_DELAY = 0.5
last_accel_read = 0

#========== BLUETOOTH ================
#Bluetooth interrupt sends GPS data every N seconds, sets LOG_DATA to 1.
#this causes sensor and gps data to be logged to the usb drive.
RequestedBluetooth = False

#========== SENSORS ================
#A global array for the current sensor data.
#(Other threads and interrupts write to these variables, the main threadReads
it.
SENSORS = [ 0.0,     0.0,      0.0]
      #SPEED  #|ACCEL|.  #DISTANCE (km's)
current_altitude = ""

#Data transfer from Pi to Android phone
transfer_data = True

# ============================ INTERRUPTS ============================

#occurs on the falling edge of the the hall effect pin, updates the speed
#and distance, or triggers alarm if the security system is armed
def hallInterrupt(self):

  global wheelTime
  global ALARM_ARMED
  global numMagnets
  global WHEEL_DIAMETER
  global ALARM_BUFFER
  global lastPrint

  print("HALL INTERRUPT")
```

```
  #Trigger the alarm if the alarm is armed.
  #ALARM_BUFFER safeguards against multiple intterupts triggering the alarm
  if ALARM_ARMED == 1 and ALARM_BUFFER == 0:
      ALARM_BUFFER = 1
      print("Alarm triggered")
      alarm()
  else:
      prevWheelTime = wheelTime
      wheelTime = time.time()
      speed = 1 / NUM_MAGNETS * 3.14159 * (WHEEL_DIAMETER / 100) / (wheelTime -
prevWheelTime)*3.6
      #Update Data
      SENSORS[2] += (WHEEL_DIAMETER / 100000) * 3.14159 #updates distance
      SENSORS[0] = speed #updates speed
      print(int(speed),"km/h")

#Helper function for hallInterrupt only. Not an interrupt, but is called by
hallInterrupt.
def alarm():
  global ALARM_ARMED
  global ALARM_BUFFER
  global ALARM_BREAK

  #This is the main routine, execute for as long as
  #The alarm is still armed. The main thread must disarm it.
  while(ALARM_ARMED == 1 and ALARM_BREAK == 0):
      ser.write(bytes("SAlarm!\0", 'ASCII'))
      buzzer.on()
      sleep(1)
      buzzer.off()
      sleep(1)

  ALARM_BREAK = 0

  print("Alarm Disarmed")
  ser.write(bytes("SAlarm Disarmed\0", 'ASCII')) #prints to lcd
  sleep(.1)
  ALARM_BUFFER = 0

#================ REST OF SETUP ====================
#LIBRARIES
import MMA8451_code
from MMA8451_code import Accel
import RPi.GPIO as GPIO
import time
import datetime
from gpiozero import Buzzer
from time import sleep
```

```
from threading import Thread
import serial
#GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
buzzer = Buzzer(BUZZER_P)
#PIN SETUP
GPIO.setup(ROW0, GPIO.IN, GPIO.PUD_UP)
GPIO.setup(ROW1, GPIO.IN, GPIO.PUD_UP)
GPIO.setup(ROW2, GPIO.IN, GPIO.PUD_UP)
GPIO.setup(ROW3, GPIO.IN, GPIO.PUD_UP)
GPIO.setup(ALARM_ARMING_SWITCH, GPIO.IN)
GPIO.setup(COL0, GPIO.OUT)
GPIO.setup(COL1, GPIO.OUT)
GPIO.setup(COL2, GPIO.OUT)
GPIO.setup(HALL_P, GPIO.IN);

# tell the GPIO library to look out for an
# event on pin HALL_P and deal with it by calling
# the hallInterrupt function
GPIO.add_event_detect(HALL_P,GPIO.FALLING, callback=hallInterrupt)

#================== NON INTERRUPT FUNCTIONS==========================

#writes sensor data to OLED
def OLEDWrite():
  global stopcount
  global last_distance
  global STOP_THRESHOLD

#every time the program writes to OLED, stores most recent distance. If it has
#not changed between readings, then the stopcount is incremented until it
reaches
#a threshold, where it will declare the wheel as stopped
  if(last_distance == SENSORS[2]): #hard coding case in the event that wheel is
stopped
      stopcount += 1
  else:
    stopcount = 0
  speed = SENSORS[0]

  if(stopcount >= STOP_THRESHOLD):
      speed = 0
      print("insufficient speed")

  last_distance = SENSORS[2] #updates last_distance
  #sends data to OLED
```

```
  writeStr = "O1," + str(speed) +"," + str(SENSORS[1]) +"," + str(SENSORS[2]) +
"\0"
  print("writing to OLED")
  print(writeStr)
  ser.write(bytes(writeStr, 'ASCII'))
  print("Wrote to OLED")

def accelWrite(): #updates acceleration from accelerometer
  axes = MMA8451.getAxisValue()
  SENSORS[1] = MMA8451.getTotalAcceleration(axes['x'], axes['y'], axes['z'])
  print("acceleration = ", SENSORS[1])

#===============KEYPAD INTERFACING FUNCTIONS==============

#Activates a column for reading by setting the corresponding pin to Low (False)
def setColumn(COL1_status, COL2_status, col3_status):
      GPIO.output(COL0, COL1_status)
      GPIO.output(COL1, COL2_status)
      GPIO.output(COL2, col3_status)

# continuously every keypad button
# return true if the correct digit was entered
# return if an incorrect digit was entered
def getKeypadInput(row, column, waitingForRelease):
      while True: #loops until button is pressed
            # check for button press in column 0
            setColumn(False, True, True)
            if not GPIO.input(ROW0):
                  if (waitingForRelease == False):
                  #only write to Arduino on intial push, not while button is
being held down
                  print("Entered  1")
                  ser.write(bytes("1\0", 'ASCII')) #serial communication
                  return row == 0 and column == 0 #returns false if wrong
button was pressed
            if not GPIO.input(ROW1):
                  if (waitingForRelease == False):
                  print("Entered  4")
                        ser.write(bytes("4\0", 'ASCII'))
                  return row == 1 and column == 0
            if not GPIO.input(ROW2):
                  if (waitingForRelease == False):
                  print("Entered  7")
                  ser.write(bytes("7\0", 'ASCII'))
                  return row == 2 and column == 0
            if not GPIO.input(ROW3):
                  if (waitingForRelease == False):
                  print("Entered  *")
```

```
        ser.write(bytes("*\0", 'ASCII'))
        return row == 3 and column == 0

# check for button press in column 1
setColumn(True, False, True)
if not GPIO.input(ROW0):
        if (waitingForRelease == False):
            print("Entered  2")
        ser.write(bytes("2\0", 'ASCII'))
        return row == 0 and column == 1
if not GPIO.input(ROW1):
        if (waitingForRelease == False):
        print("Entered  5")
        ser.write(bytes("5\0", 'ASCII'))
        return row == 1 and column == 1
if not GPIO.input(ROW2):
        if (waitingForRelease == False):
        print("Entered  8")
        ser.write(bytes("8\0", 'ASCII'))
        return row == 2 and column == 1
if not GPIO.input(ROW3):
        return row == 3 and column == 1

# check for button press in column 2
setColumn(True, True, False)
if not GPIO.input(ROW0):
        if (waitingForRelease == False):
        print("Entered  3")
        ser.write(bytes("3\0", 'ASCII'))
        return row == 0 and column == 2
if not GPIO.input(ROW1):
        if (waitingForRelease == False):
        print("Entered  6")
        ser.write(bytes("6\0", 'ASCII'))
        return row == 1 and column == 2
if not GPIO.input(ROW2):
        if (waitingForRelease == False):
        print("Entered  9")
        ser.write(bytes("9\0", 'ASCII'))
        return row == 2 and column == 2
if not GPIO.input(ROW3):
if (waitingForRelease == False):
print("Entered  #")
ser.write(bytes("#\0", 'ASCII'))
return row == 3 and column == 2
setColumn(True, True, True)

if waitingForRelease:
```

```
            return -1
      return False

#checks for keypad input: digit is the correct number to be entered, index is
#the index of the password
def checkKeypadInput(digit, index):
      # calculate the location of the digit
      row = (int) ((digit-1)/3)
      column = (digit+2)%3

      # determine whether or not correct digit was entered
      correctDigitEntered[index] = getKeypadInput(row, column, False)
      # wait until the button is released
      while not getKeypadInput(row, column, True) == -1:
      print("waiting for release")

      print("Row: ", row)
      print("Column: ", column)
      print("Digit entered: ", correctDigitEntered[index])
      return correctDigitEntered[index]

#=================== END OF KEYPAD FUNCTIONS ==============

# Retrieves the inputs from the keypad and disarms security of correct password
entered
def enterPassword():
  global ALARM_ARMED
  ser.flushOutput()
  sleep(.1)
  # start keypad routing if the alarm is armed
  if ALARM_ARMED == 1:
      setColumn(True, True, True)
      print("Enter Passcode...")

      # retrieves the inputs from the keypad
      for x in range (0, NUM_DIGITS):
      checkKeypadInput(passcode[x], x)

      # sums the entries (entries are either true or false)
      sum = 0
      for x in range (0,NUM_DIGITS):
      sum += correctDigitEntered[x]

      # Executes if correct password
      if sum == NUM_DIGITS:
      #Disable alarm
      ALARM_ARMED = 0
```

```
        # empty the passcode array
        for x in range(0, NUM_DIGITS):
        correctDigitEntered[x] = -1
        print("PASS ENTERED")
        ser.write(bytes("SPassword Correct!\0", 'ASCII'))
        sleep(2)
        # Executes if incorrect password
        else:
        print("PASS FAILED")
        ALARM_ARMED = 1
        ser.write(bytes("SIncorrect Password\0", 'ASCII')); #pass failed


# securityRoutine -
#   Called by Top-Level loop. Executes as long as the alarm is armed and has
not been disarmed.
#   IF The bike moves whilst still armed, call alarm()
#   ELSE Display keys pressed on LCD and unlock if right combination.
def securityRoutine() :
  global ALARM_ARMED

  #Alarm routine runs on interrupt basis. While ALARM_ARMED = 1, if hall-effect
is detected,
  #a seperate thread for soundAlarm() executes.

  ALARM_ARMED = 1
  print("Security armed!")

  #Bike is now idle and locked.
  while ALARM_ARMED == 1 :
      #Sleep until # press, then call keypad as long as the alarm is armed.
      ser.write(bytes("SEnter code ", 'ASCII'))
      enterPassword()

  return

#This function updates the global current_altitude by asking the
#Arduino over serial for the current altitude.
def updateAltitude() :
  global current_altitude

  ser.write(bytes("D\0", 'ASCII'))
  inData = ser.readline()
  inData = inData.decode().strip('\n') #removes formatting sent form Arduino
  current_altitude = inData

#This function runs on a seperate thread in the case that we are operating with
#A bluetooth connection. If so, we send sensor data so the phone can log it.
```

```
#This happens everytime the phone pings the raspberry pi, causing inWaiting
#to be > 0. Then, we read the global SENSORS array and pass it it.
#Other threads write to this array.
def send_data(port):

  global current_altitude
  global SENSORS

  while transfer_data:
      # send data when the phone app signals it wants it
      if port.inWaiting() > 0:
      #needs to be formatted specifically so bluetooth can parse
      writeStr = "|" + current_altitude +"|" + str(SENSORS[0]) +"|" +
str(SENSORS[2]) +"|" + str(SENSORS[1]) + "|"
      print("read Line")
      port.write(bytes(writeStr, 'ASCII'))
      port.flushInput()

#Asks user on LED to connect to bluetooth and responds accordingly
def askToConnect():
  #Now determine whether the user wants to connect with bluetooth.
  global RequestedBluetooth
  print("checking keypad input")
  checkKeypadInput(1, 0)
  print("correctDigitEntered[0] =  ", correctDigitEntered[0])
  if correctDigitEntered[0] == 1 :
   # Setup communication to Phone
      bluetooth_port = serial.Serial("/dev/rfcomm0", baudrate=9600,
timeout=60.0)
      # Create thread to handle Android-pi data exchange
      thread = Thread(target = send_data, kwargs={'port':bluetooth_port})
      thread.start()
      print("Created thread")
      ser.write(bytes("SBluetooth connected!\0", 'ASCII'))
      RequestedBluetooth = True
      sleep(2)
      ser.write(bytes("S\0", 'ASCII'))

  else:
      print("didn't create thread")
  # erase the entry
  correctDigitEntered[0] = -1
  RequestedBluetooth = True

#======================== END OF FUNCTIONS ================================

prevState = GPIO.input(ALARM_ARMING_SWITCH) #initializes previous state of
button
```

```
#accelerometer initiliaztion
MMA8451 = Accel()
MMA8451.init()
#========================== MAIN EXECUTION LOOP ======================
while True :
  #If security ARM switch has been turned ON from OFF state.
  #print(GPIO.input(ALARM_ARMING_SWITCH))
  if not (GPIO.input(ALARM_ARMING_SWITCH) == prevState):
      print("Security")
      securityRoutine()
      prevState = GPIO.input(ALARM_ARMING_SWITCH)
      ser.write(bytes("SRiding Mode\0", 'ASCII'))

  if((time.time() - last_accel_read) > ACCEL_DELAY):
      last_accel_read = time.time()
      accelWrite()
  if((time.time() - last_OLED_write) > READING_DELAY):
      last_OLED_write = time.time()
      OLEDWrite()
      updateAltitude()

  if (RequestedBluetooth == False) :
      print("Asking to connect.")
      ser.write(bytes("S1 = bt 2 = no bt\0", 'ASCII'))#writes to lcd
      askToConnect()
      ser.write(bytes("SRiding Mode\0", 'ASCII')) #writes to lcd

#============= BLUETOOTH OVERRIDE FUNCTIONS (UNTESTED) ====================

def setArmedOverride():
  ALARM_ARMED = 1;

def setDisarmedOverride():
  ALARM_ARMED = 0;

#change password
def changePass(digit0, digit1, digit2, digit3):
      passcode[0] = digit0
      passcode[1] = digit1
      passcode[2] = digit2
      passcode[3] = digit3
#turns alarm off
def alarmOff():
      ALARM_BREAK = 1
```

Appendix E – Web app code

/**************************************createKMLwithID.php**************************************/

```php
<?php
include 'loginPhone.php';

//Parameters
$RideID = $_GET['id'];

$TableName = 'logged_data';
$OutputFile = "journey_$RideID.kml";
$Folder = '/var/www/html/kml/';

//Fill in these paths with a HTML request
//$RideID = $_GET['rideid']
$OutputPath = $Folder . $OutputFile; //Combines the directory with the file
name

//Globals
error_reporting(-1);
ini_set('display_errors', 'On');

$START_ICON =
'https://cdn0.iconfinder.com/data/icons/small-n-flat/24/678111-map-marker-512.p
ng';
$START_PATH_ICON =
'https://emojipedia-us.s3.amazonaws.com/thumbs/160/facebook/65/bicycle_1f6b2.pn
g';
$FINISH_PATH_ICON =
'https://emojipedia-us.s3.amazonaws.com/thumbs/160/facebook/65/bicycle_1f6b2.pn
g';
$FINISH_ICON =
'https://cdn2.iconfinder.com/data/icons/ios-7-icons/50/finish_flag-128.png';
$MARKER_ICON =
'https://cdn4.iconfinder.com/data/icons/iconsimple-places/512/pin_1-512.png';

//Ride statistic variables.
$totalReadings = 0;
$speedSummation = 0;
$avgspeed = 0;
$topspeed = 0;
$topaccel = 2;
$initialAlt = 0;

//Keep track of how many paths we have so we can label them numerically/
$PATHS = 1;

// Creates the Document.
$dom = new DOMDocument('1.0', 'UTF-8');
```

```php
$dom->preserveWhiteSpace = true;
$dom->formatOutput = true;

//Tier 0
//Creates the root KML element and appends it to the root document.
$node = $dom->createElementNS('http://www.opengis.net/kml/2.2', 'kml');
$rootNode = $dom->appendChild($node);
$mainDoc = $dom->createElement('Document');
$parNode = $rootNode->appendChild($mainDoc);

//Tier 1
$name = $dom->createElement('name', "KML for journey: $RideID");
$waypointsFolder = $dom->createElement('Folder');
$tracksFolder = $dom->createElement('Folder');

$parNode->appendChild($name);
$parNode->appendChild($waypointsFolder);
$parNode->appendChild($tracksFolder);

//Tier 2, Setup the folders
$waypointsFolder->setAttribute('id', 'Waypoints');
$tracksFolder->setAttribute('id', 'Tracks');

//Connect with Server and populate the rest of the data!
//Create the connection
$conn = new mysqli($servername, $username, $password, $db);
//Check if it is alive
if($conn->connect_error){
      die("Connection failed: " . $conn->connect_error);
}

//Select from the table specified and the correct ride number.
$sql = "SELECT * FROM " . $TableName . " WHERE ride_id = ". $RideID;
echo $sql;
$result = $conn->query($sql);

//Create the first track folder.
$currentTrackFolder = $dom->createElement('Folder');
$currentTrackFolder->setAttribute('id', 'TrackOne');

//prevRow is initially the start row.
$prevRow = $result->fetch_assoc();
$startPlacemark = $dom->createElement('Placemark');

//Add auxillary things for start node, (style, name)..
$name = $dom->createElement('name');
$name->nodeValue = '<!CDATA['.'<b><div> Journey '.$GLOBALS['TableName'].'
Start</div></b>';
```

```
$startPlacemark->appendChild($name);

//Add rest of info by reading row.
$startDesc = $dom->createElement('description');
$startDesc->nodeValue =
'Coordinates :'.$prevRow["COOR_lng"].', '.$prevRow["COOR_lat"];
$startPlacemark->appendChild($startDesc);
appendWaypointAuxillaryElements($startPlacemark, $dom, 'START');
$startCoorString =
$prevRow["COOR_lng"].','.$prevRow["COOR_lat"].','.$prevRow["COOR_alt"];
$startCoor = $dom->createElement('coordinates', $startCoorString);
$startPoint = $dom->createElement('Point');
$startPoint->appendChild($startCoor);

$GLOBAL['initialAlt'] = $prevRow["COOR_alt"];

//This is what needs to be done for each placemark. Start is done now.
$startPlacemark->appendChild($startPoint);

//Append it to the waypoints.
$waypointsFolder->appendChild($startPlacemark);

	//Convert rest of the data.
	while($row = $result->fetch_assoc()){

	//Update journey statistics.
	  updateStatistics($row, $prevRow);

	//In the case this is a new path start, create a new track folder with a
waypoint indicating the start.
	if ($row["type"] == "START_P") {
		$PATHS++;
		$currentTrackFolder = $dom->createElement('Folder');
		$currentTrackFolder->setAttribute('id', "Track".$PATHS);

		$placemarkNode = $dom->createElement('Placemark');
	$name = $dom->createElement('name', "Start of Path " .
$GLOBALS['PATHS']);
	$placemarkNode->appendChild($name);
		appendWaypointAuxillaryElements($placemarkNode, $dom, 'START_P');
		//Add waypoint point data
	$CoorString =
$prevRow["COOR_lng"].','.$row["COOR_lat"].','.$row["COOR_alt"];
	$Coor= $dom->createElement('coordinates', $CoorString);
	$Point= $dom->createElement('Point');
	$Point->appendChild($Coor);
	$placemarkNode->appendChild($Point);
		$waypointsFolder->appendChild($placemarkNode);
```

```
	}

	//A Path is always added from the point in previous row to the point in current row unless
	//The current row is the start of a new path.
	else {
		addPath($prevRow, $row, $currentTrackFolder, $dom);

		//Determine what kind of way point to add, if any.
		//Also determines if we should append the track folder.
		  switch( $row["type"]) {
		    case 'MARKER' :
		       $placemarkNode = $dom->createElement('Placemark');
		$name = $dom->createElement('name', 'User Marker');
		$placemarkNode->appendChild($name);
		   appendWaypointAuxillaryElements($placemarkNode, $dom,'MARKER');

		    //Add waypoint point data
		$CoorString = $prevRow["COOR_lng"].','.$row["COOR_lat"].','.$row["COOR_alt"];
		$Coor= $dom->createElement('coordinates', $CoorString);
		$Point= $dom->createElement('Point');
		$Point->appendChild($Coor);
		$placemarkNode->appendChild($Point);
		       $waypointsFolder->appendChild($placemarkNode);
		       break;
		   case 'FINISH_P':
		       $placemarkNode = $dom->createElement('Placemark');
		$name = $dom->createElement('name', "End of Path ".$GLOBALS['PATHS']);
		$placemarkNode->appendChild($name);
		       appendWaypointAuxillaryElements($placemarkNode, $dom,'FINISH_P');
		    //Add waypoint point data
		$CoorString = $prevRow["COOR_lng"].','.$row["COOR_lat"].','.$row["COOR_alt"];
		$Coor= $dom->createElement('coordinates', $CoorString);
		$Point= $dom->createElement('Point');
		$Point->appendChild($Coor);
		$placemarkNode->appendChild($Point);
		       $waypointsFolder->appendChild($placemarkNode);
		       $tracksFolder->appendChild($currentTrackFolder);
		       break;
		    case 'FINISH' :
		       $placemarkNode = $dom->createElement('Placemark');
		   $name = $dom->createElement('name');
		    $name->nodeValue = '<!CDATA['.'<b><div> Journey '.$GLOBALS['TableName'].' End</div></b>';
```

```
            $placemarkNode->appendChild($name);

            global $initialAlt;
            $netAlt = $row["COOR_alt"] - $initialAlt;

                    $finishDesc = $dom->createElement('description');
            $finishDesc->nodeValue = '<!CDATA['.
                  'Finish time:            '.date('Y-m-d
h:i:s',$row['worldTime']).'<br>'
                  .'Coordinates:           '.$row['COOR_lng'].',
'.$row['COOR_lat'].'<br>'
                  .'Altitude:              '.$row['COOR_alt'].'<br><br>'
                  .'<u><b>STATISTICS</b></u><br>'
                  .'Distance Travelled: '.$row['distance'].' m<br><br>'
                  .'Top Speed:          '.$GLOBALS['topspeed'].' m/s   <br>'
                  .'Average Speed:   '.$GLOBALS['avgspeed'].' m/s <br><br>'
                  .'Top Acceleration:   '.$GLOBALS['topaccel'].' m/s^2<br><br>'
                  .'Net Altitude Gain:  '.$netAlt.' m ';

                   $placemarkNode->appendChild($finishDesc);
                   appendWaypointAuxillaryElements($placemarkNode,
$dom,'FINISH');
                //Add waypoint point data
            $CoorString =
$row["COOR_lng"].','.$row["COOR_lat"].','.$row["COOR_alt"];
            $Coor= $dom->createElement('coordinates', $CoorString);
            $Point= $dom->createElement('Point');
            $Point->appendChild($Coor);
            $placemarkNode->appendChild($Point);
                  $waypointsFolder->appendChild($placemarkNode);
                  $tracksFolder->appendChild($currentTrackFolder);
                  break;
               case 'PATH' : break;
               case 'START': echo "Found Another START node!"; break;
               default :

                    echo "Bad row type read from SQL. Got: ".$row["type"]."
END";
               }
      }

      $prevRow = $row;
      }


//Wrap things up.
$conn->close();
$dom->save($OutputPath);
```

```php
$kmlOutput = $dom->saveXML();

echo $kmlOutput;

//Appends a Placemark child to the trackfolder specified, the placemark will
contain
//A correctly colored path from sqlRowPrev to sqlRowCurr with data that is the
readings of
//sqlRowPrev
function addPath($sqlRowPrev, $sqlRowCurr, &$currentTrackFolder, &$dom) {
	$placemarkNode = $dom->createElement('Placemark');
	$name = $dom->createElement('name');
	$name->nodeValue = '<!CDATA['.'<u><span
style=\'color:#bfe600\'>'.date('Y-m-d
h:i:s',$sqlRowPrev['worldTime']).'</span></u>';

	$desc = $dom->createElement('description');
	$desc->nodeValue = '<![CDATA['
	.'<b><div> READINGS </div></b><br>'
	.'Speed:     '.$sqlRowPrev['speed'].' km/h<br>'
	.'Acceleration: '.$sqlRowPrev['acceleration'].' m/s^2<br>'
	.'Altitude:  '.$sqlRowPrev['COOR_alt'].' m<br>'
	.'Distance:  '.$sqlRowPrev['distance'].' m'
	;

	$coordinatesString =
$sqlRowPrev["COOR_lng"].','.$sqlRowPrev["COOR_lat"].'
'.$sqlRowCurr["COOR_lng"].','.$sqlRowCurr["COOR_lat"].' ' ;
	$coordinates = $dom->createElement('coordinates', $coordinatesString);

	$LineString = $dom->createElement('LineString');

    //The line color is based on the speed of the first point.
    $colorhex = getLineColor($sqlRowPrev["speed"]);

    $Style = $dom->createElement("Style");
    $LineStyle = $dom->createElement("LineStyle");
    $width = $dom->createElement("width", '3');
    $color = $dom->createElement("color", $colorhex);

    $LineString->appendChild($coordinates);

    $LineStyle->appendChild($color);
    $LineStyle->appendChild($width);
    $Style->appendChild($LineStyle);
    $placemarkNode->appendChild($name);
    $placemarkNode->appendChild($desc);
    $placemarkNode->appendChild($Style);
```

```
    $placemarkNode->appendChild($LineString);

    //Finnaly..
    $currentTrackFolder->appendChild($placemarkNode);
}

//Returns a <color> value eg. ff00e6bf, depending on the speed passed.
//Maxes at
function getLineColor($speed) {
    $speedUsed = $speed;
      if ($speed > 70)
      $speedUsed = 70;

      //Max value RED at 70km/h w/ r = 255 g = 0 b =0
      //at 35km/h YELLOW w/ r= 255 g = 255 b =0
      //At 0km/h GREEN w/ r= 0 g = 255 b = 0
      //Returns hex in form ABGR
      // a = 0x80 = 128 << 6 = 2147483648

      //70 -> 255
      //0 -> 65280
      //so m = -928.9285714
      $hexVal = (int) ((-928.9285714)*($speedUsed) + 65280 + 2147483648);
      return (dechex($hexVal));
}

//Updates global journey statistics variables
function updateStatistics($row, $prevRow) {
      $speed = $row["speed"];
      $accel = $row["acceleration"];
      global $topaccel;
      global $topspeed;
      global $totalReadings;
      global $speedSummation;

      if ($speed != 0) {
      $totalReadings++;
      $speedSummation += $speed;
      }

      if ($speed > $topspeed) {
      $topspeed = $speed;
      }

      if ($accel > $topaccel) {
      $topaccel = $accel;
      }
}
```

```
//Helper function for creating a Waypoint,
//The Placemarker node of the waypoint is passed along with a type specifer,
//which is either
//  START, FINISH, START_P, FINISH_P, or MARKER
//This will append <name> and <style> for the node, thus the only fields left
to
//fill will be <desc> and <Point>
function appendWaypointAuxillaryElements(&$PlacemarkNode, &$dom, $type) {
      //All types have these fields.
      $Style = $dom->createElement('Style');
      $IconStyle = $dom->createElement('IconStyle');
      $scale = $dom->createElement('scale', 1);
      $IconStyle->appendChild($scale);
      $Icon = $dom->createElement('Icon');

      switch ($type) {
      case 'START':
      $href = $dom->createElement('href', $GLOBALS['START_ICON']);

$PlacemarkNode->appendChild($Style)->appendChild($IconStyle)->appendChild($Icon
)->appendChild($href);
      break;
      case 'FINISH':
      $href = $dom->createElement('href', $GLOBALS['FINISH_ICON']);

$PlacemarkNode->appendChild($Style)->appendChild($IconStyle)->appendChild($Icon
)->appendChild($href);
        break;
      case 'START_P':
      $href = $dom->createElement('href', $GLOBALS['START_PATH_ICON']);

$PlacemarkNode->appendChild($Style)->appendChild($IconStyle)->appendChild($Icon
)->appendChild($href);
      break;
      case 'FINISH_P':
        $href = $dom->createElement('href', $GLOBALS['FINISH_PATH_ICON']);

$PlacemarkNode->appendChild($Style)->appendChild($IconStyle)->appendChild($Icon
)->appendChild($href);
      break;
      case 'MARKER':
      $href = $dom->createElement('href', $GLOBALS['MARKER_ICON']);

$PlacemarkNode->appendChild($Style)->appendChild($IconStyle)->appendChild($Icon
)->appendChild($href);
      break;
```

```
        default :
        echo "AHHHH BUG IN createKML.php AHHHH! Invalid Placemark type!";
        }
}
```

/**********End createKMLwithID.php**********/

/**********index.php**********/

```
<!-- Maps API Key: AIzaSyAQLGxd2t-CdzhMMVCgLRqJcpNEj1s114Q -->
<!-- https://developers.google.com/maps/documentation/javascript/kml#sidebar
-->
<!DOCTYPE html>
<html>
  <head>
        <meta name="viewport" content="initial-scale=1.0, user-scalable=no">
        <meta charset="utf-8">
        <title>Route Data</title>
        <style>
        html, body {
        height: 100%;
        padding: 0;
        margin: 0;
        }

        #wrapper{
        height: 93%;
        width: 100%;
        }

        #gui{
        height: 100%;
        width: 22%;
        float: left;
        overflow: hidden;
          clear: left;
        bottom: -20px;
        position: relative;
        top: -7px;
        border: 1px solid white;
        }


        #map {
        height: 100%;
        overflow: hidden;
        border: 1px solid white;
        }
```

```
/* The container <div> - needed to position the dropdown content */
.dropdown {
position: relative;
display: inline-block;
}

/* Dropdown Contetn (Hidden by Default) */
.dropdown-content {
display: none;
    position: absolute;
background-color: #f1f1f1;
min-width: 160px;
box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
z-index: 1;
font-size: 70%;
}

/* Links inside the dropdown */
.dropdown-content a {
color: black;
padding: 12px 16px;
text-decoration: none;
display: block;
}

/* Change color of dropdown links on hover */
.dropdown-content a:hover {
background-color: #ddd;
}

/* Show the dropdown menu (use JS to add this class to the
.dropdown-content
container when the user clicks on the dropdown button0 */
.show {
display:block;
}

/* Brennan's edits */
.header {
background-color: #0094FF;
text-align: center;
color: black;
font-size: 150%;
font-family: Helvetica;
padding 0;
border: 1px solid white;
text-shadow: 1px 1px #fff;
```

```
        }

        h5 {
        margin-top: 10px;
          margin-bottom: 10px;
        }
        /* Menu button in to left corner used to select rides */
        #button {
        float: left;
        background-color: #085DAD;
        position: relative;
        top: -9px;
        text-decoration: none;
        border: none;
        background:
url(https://upload.wikimedia.org/wikipedia/commons/thumb/8/8d/VisualEditor_-_Icon_-_Menu.svg/32px-VisualEditor_-_Icon_-_Menu.svg.png) 10px 10px no-repeat;
        padding: 10px 10px 10px 43px;
        font-family: Helvetica;
        font-weight: bold;
        background-position: 7px 4px;
        height: 40px;
        }
        /* Dropdown button on hover & focus */
        #button:hover, #button:focus {
        background-color: #77A6F7;
        }

        /* Formatting for table that holds rider statistics */
        #ride-stats {
        font-family: "Trebuchet MS", Arial, Helvetica;
        border-collapse: collapse;
        width: 100%;
        }

        #ride-stats td, #ride-stats th {
        border: 1px solid #ddd;
        padding: 8px;
        font-size: 80%;
        }

        #ride-stats tr:nth-child(even) {
        background-color: #f2f2f2;
        }

        #ride-stats tr:hover {
        background-color: #ddd;
        }
```

```
        #ride-stats th {
        font-size: 100%;
        background-color: #8DDF00;
        }
        /* bike image in top right corner */
        #bike-image {
        height: "30px";
        width: "30px";
        float: "right";
        position: "relative";
        padding: "10px";
        }

        </style>

 </head>
  <body>
        <!-- Title Banner -->
        <div class="header">
        <h5>IntelliBike

        <!-- Top left dropdown menue button -->
              <button onclick="showDrop()"  id="button">Select Ride</button>

              <img src="bike-icon2.png" alt="BIKE" height="25px" width="35px"
style="float: right; margin-right: 8px" align="left">
                    <!-- Dropdown menu -->
                    <div id="myDropdown" class="dropdown-content">

                        <!-- Template for what I want from mySQL
                    <a href="#">Link 1</a>
                    <a href="#">Link 2</a>
                    <a href="#">Link 3</a>
                    -->

                    <?php
                          include 'loginPhone.php';

                          $conn = new mysqli($servername, $username, $password,
$db);
                          if($conn->connect_error){
                          die("Connection failed: " . $conn->connect_error);
                          }

                          $sql = "SELECT * FROM ride_history";
                          $result = $conn->query($sql);
```

```
                        while($row = $result->fetch_assoc()){
                            $timestamp = $row['date'];
                        $date = date("M jS, y - gA", $timestamp);

                        echo '<a
href="http://intellibikeubc.com/?id='.$row["ride_id"].'">' .$date. '</a>';
                        }

                    ?>

            </div>

            <script>

                /* When the user click on the button, toggle between hding
                   and showing the dropdown content */
                function showDrop(){

document.getElementById("myDropdown").classList.toggle("show");
                }

                /* Close the dropdown if the user clicks outside of it */
                window.onclick = function(event) {
                    if(!event.target.matches('#button')){
                        var dropdowns =
document.getElementsByClassName("dropdown-content");
                        var i;

                        for(i = 0; i < dropdowns.length; i++){
                        var openDropdown = dropdowns[i];
                        if(openDropdown.classList.contains('show')){
                              openDropdown.classList.remove('show');
                        }
                        }
                    }
                }
            </script>
      </h5>
      </div>

      <div id="wrapper">
      <div style="background-color:white" id="gui">
            <!-- Rider statistics table -->
            <table id="ride-stats">
            <tr>
                  <th colspan="3">Rider Statistics</th>
            </tr>
```

```php
<?php
                        include 'loginPhone.php';

                        $conn = new mysqli($servername, $username, $password,
$db);
                        //Check if it is alive
                    if($conn->connect_error){
                        die("Connection failed: " . $conn->connect_error);
                        }

                echo" <tr>";
                    echo" <td>Top Speed</td>";


                       $sql = "SELECT * FROM logged_data ORDER BY speed DESC
LIMIT 1";
                        $result = $conn->query($sql);
                        $row = $result->fetch_assoc();


                        echo "<td>".$row['speed']." km/h</td>";

                echo"</tr>";

                echo" <tr>";
                    echo" <td>Total Distance</td>";


                        $sql = "SELECT ride_id, MAX(distance) FROM logged_data
GROUP BY ride_id";
                          $result_d = $conn->query($sql);

                        $totDist = 0;

                        foreach ($result_d as $val){
                        //echo $val['distance'];
                        $totDist += $val['MAX(distance)'];
                        }

                        $kms = $totDist / 1000;


                        echo "<td>".number_format((float)$kms, 2, '.', '')."
km</td>";

                echo"</tr>";

                    echo" <tr>";
```

```
                    echo" <td>Total Time</td>";


                        $sql = "SELECT ride_id, MAX(worldTime) FROM logged_data
GROUP BY ride_id";
                        $result = $conn->query($sql);

                    $time = 0;

                        foreach ($result as $val){
                        //echo $val['distance'];
                        $time += $val['MAX(worldTime)'];
                        }

                        $sql = "SELECT ride_id, MIN(worldTime) FROM logged_data
GROUP BY ride_id";
                        $result = $conn->query($sql);

                        foreach ($result as $val){
                        //echo $val['distance'];
                        $time -= $val['MIN(worldTime)'];
                        }

                        $hrs = $time / 3600;

                        echo "<td>". number_format((float)$hrs, 2, '.', '')."
hrs</td>";

                echo"</tr>";

                echo" <tr>";
                   echo" <td>Average Speed</td>";

                        $avgSpeed = $kms / $hrs;

                    echo "<td>".number_format((float)$avgSpeed, 2, '.', '')."
km/h</td>";
                echo"</tr>";


                echo" <tr>";
                   echo" <td>Average Distance</td>";


                        $avgKms = $kms / count($result_d);
```

```
                        echo "<td>".number_format((float)$avgKms, 2, '.',
'')." km</td>";

                echo"</tr>";


                echo" <tr>";
                   echo" <td>Average Time</td>";


                        $avgHrs = $hrs / count($result_d);


                        echo "<td>".number_format((float)$avgHrs, 2, '.',
'')." hrs</td>";

                echo"</tr>";
?>

            </table>
        </div>

        <div id="map"></div>
        <div id="capture"></div>

        <script>
        var map;

        function loadMap() {
              map = new google.maps.Map(document.getElementById('map'), {
                center: new google.maps.LatLng(-19.257753, 146.823688),
                zoom: 16,
                   mapTypeId: 'terrain'
              });

              var kmlLayer = new google.maps.KmlLayer(
                        <?php
                        $Folder = 'http://intellibikeubc.com/kml/';
                        $id = $_GET["id"];
                             if($id != ""){
                               $path = $Folder."journey_$id.kml";
                        }else{
                               $path = $Folder.'DefaultKML.kml';
                        }
                           echo "'".$path."'";
                        ?>, {
                suppressInfoWindows: false,
                preserveViewport: false,
```

```
                map: map
            });
        }
        </script>

        <script  defer

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAQLGxd2t-CdzhMMVCgLRqJcp
NEj1s114Q&callback=loadMap">
        </script>
        </div>


    </body>
</html>
```

/***End index..php***/

/*************************************loadFileIntoTable.php*************************************/

```
<?php
error_reporting(-1);
ini_set('display_errors', 'On');

include 'loginPhone.php';
include 'uploadedFilesInfo.php';

$into_table = $_GET['table'];
$file_name = $_GET['type'].'.txt';
$file_path = $rootFolder.$file_name;


$conn = new mysqli($servername, $username, $password, $db);
if ($conn->connect_error) {
        die("Could not connect: " . $conn->connect_error);
}

$sql = "LOAD DATA LOCAL INFILE '".$file_path."' INTO TABLE $into_table FIELDS
TERMINATED BY '|'";

$result = $conn->query($sql);

$conn->close(); //All the data is now in the table
```

```
?>
```

/**********End loadFileIntoTable.php**********/

/**********uploadFile.php**********/

```
<?php
include 'uploadedFilesInfo.php';


$file_name = $_GET['type'].'.txt';
$file_path = $rootFolder.$file_name;


if($_SERVER['REQUEST_METHOD']=='POST'){

	try{
	//Move the file into the server
	copy($_FILES['image']['tmp_name'], $file_path);

	}catch(Exception $e){
	die($e->getMessage());
	}
}
?>
```

/**********End uploadFile.php**********/

Appendix F – Mobile app code

For the sake the length of this report, the layout XML files as well as the Manifests have been omitted from this section, but can be found on git here.

/**********Disclaimer.java**********/

```
//This is a file that will show the user a disclaimer stating that the app can not do real-time
//	navigation in compliance with the Google Directions API

package ca.ubc.zachrivard.self.test;

import android.Manifest;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
```

```
import android.content.Intent;
import android.content.IntentSender;
import android.content.pm.PackageManager;
import android.location.LocationManager;
import android.os.Bundle;
import android.provider.Settings;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.Button;

import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.common.api.PendingResult;
import com.google.android.gms.common.api.ResultCallback;
import com.google.android.gms.common.api.Status;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.location.LocationSettingsRequest;
import com.google.android.gms.location.LocationSettingsResult;
import com.google.android.gms.location.LocationSettingsStatusCodes;


public class Disclaimer extends AppCompatActivity {
   private static final String TAG = Disclaimer.class.getSimpleName();

   private static final int PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION = 1;
   private static final int PERMISSIONS_REQUEST_CALL_ACCESS = 2;
   private static final int PErMISSIONS_REQUEST_READ_EXTERNAL = 3;
   private static final int PErMISSIONS_REQUEST_WRITE_EXTERNAL = 4;
   private static final int PERMISSIONS_REQUEST_INTERNET = 5;
   private boolean mLocationServicesEnabled;
   Context context;




   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_disclaimer);


       context = this;

       // Prompt the user for permissions.
       // Follow the chain in the callback as we
```

```
        // can only request one at a time
        getLocationPermission();

        //See if we have location access
        checkLocationServices();

        //Prompt the user for location access
        if (!mLocationServicesEnabled) {
            getLocationServices();
        }

        Button accept = (Button) findViewById(R.id.accept_button);
        accept.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                checkLocationServices();

                if (!mLocationServicesEnabled) {
                    getLocationServices();
                } else {
                    Intent intent = new Intent(getApplicationContext(),
ScanBtActivity.class);
                    startActivity(intent);
                }
            }
        });
    }


    /**
     * Prompts the user for permission to use the device location.
     */
    private void getLocationPermission() {
        /*
         * Request location permission, so that we can get the location of the
         * device. The result of the permission request is handled by a callback,
         * onRequestPermissionsResult.
         */
        if (ContextCompat.checkSelfPermission(this.getApplicationContext(),
                android.Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
            Log.d(TAG, "Fine location permissions granted");
        } else {
            displayLocationSettingsRequest(this);
        }
    }

    /**
     * Prompts the user for permission to write to external storage
     */
```

```
    private void getExternalWritePermissions() {
        /*
         * Request location permission, so that we can get the location of the
         * device. The result of the permission request is handled by a callback,
         * onRequestPermissionsResult.
         */
        if (ContextCompat.checkSelfPermission(this.getApplicationContext(),
                Manifest.permission.WRITE_EXTERNAL_STORAGE)
                == PackageManager.PERMISSION_GRANTED) {
            Log.d(TAG, "External write permissions granted");
        } else {
            ActivityCompat.requestPermissions(this,
                    new
String[]{android.Manifest.permission.WRITE_EXTERNAL_STORAGE},
                    PErMISSIONS_REQUEST_WRITE_EXTERNAL);
        }
    }

    /**
     * Prompts the user for permission to read from external storage
     */
    private void getExternalReadPermissions() {
        /*
         * Request location permission, so that we can get the location of the
         * device. The result of the permission request is handled by a callback,
         * onRequestPermissionsResult.
         */
        if (ContextCompat.checkSelfPermission(this.getApplicationContext(),
                Manifest.permission.READ_EXTERNAL_STORAGE)
                == PackageManager.PERMISSION_GRANTED) {
            Log.d(TAG, "External read permissions granted");
        } else {
            ActivityCompat.requestPermissions(this,
                    new
String[]{android.Manifest.permission.READ_EXTERNAL_STORAGE},
                    PErMISSIONS_REQUEST_READ_EXTERNAL);
        }
    }

    /**
     * Prompts the user for permission to use the device's phone
     */
    private void getCallPermission(){


        if (ContextCompat.checkSelfPermission(this.getApplicationContext(),
                android.Manifest.permission.CALL_PHONE)
                == PackageManager.PERMISSION_GRANTED) {
            Log.d(TAG, "Call permissions granted");
        } else {
```

```
            ActivityCompat.requestPermissions(this,
                    new String[]{android.Manifest.permission.CALL_PHONE},
                    PERMISSIONS_REQUEST_CALL_ACCESS);
        }
    }


    /**
     * Prompts the user for permission to use internet
     */
    private void getInternetPermissions(){

        if (ContextCompat.checkSelfPermission(this.getApplicationContext(),
                Manifest.permission.INTERNET)
                == PackageManager.PERMISSION_GRANTED) {
            Log.d(TAG, "Call permissions granted");
        } else {
            ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.INTERNET},
                    PERMISSIONS_REQUEST_CALL_ACCESS);
        }
    }


    /**
     * Handles the result of the request for location permissions.
     */
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String
permissions[], @NonNull int[] grantResults) {

        switch (requestCode) {
            case PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION: {
                // If request is cancelled, the result arrays are empty.
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED)
{

                    Log.d(TAG, "Fine location permissions granted");
                    getCallPermission(); //Next in the chain


                }else{

                    AlertDialog.Builder builder = new
AlertDialog.Builder(context);
                    builder.setTitle("Location permission not enabled");
                    builder.setMessage("Location services are required in order
for the app to run");
```

```
                    builder.setPositiveButton("Try agian", new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            getLocationPermission();
                        }
                    });
                    builder.setNegativeButton("Quit app", new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            finish();
                            System.exit(0); //FORCE QUIT THE APP
                        }
                    });

                }
                break;
            }
            case PERMISSIONS_REQUEST_CALL_ACCESS: {
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED)
{
                    Log.d(TAG, "Call permissions granted");
                    getExternalReadPermissions(); //Next in the chain
                }else{

                    AlertDialog.Builder builder = new
AlertDialog.Builder(context);
                    builder.setTitle("Call services not enabled");
                    builder.setMessage("Calling services are required in order for
the app to run");
                    builder.setPositiveButton("Try agian", new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            getCallPermission();
                        }
                    });
                    builder.setNegativeButton("Quit app", new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            finish();
                            System.exit(0); //FORCE QUIT THE APP
                        }
                    });

                }
                break;
```

```
            }
            case PErMISSIONS_REQUEST_READ_EXTERNAL: {
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED)
{
                    Log.d(TAG, "External read permissions granted");
                    getExternalWritePermissions(); //Next in the chain
                }else{

                    AlertDialog.Builder builder = new
AlertDialog.Builder(context);
                    builder.setTitle("External Read Permission Denied");
                    builder.setMessage("External reading services are required in
order for the app to run");
                    builder.setPositiveButton("Try agian", new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            getExternalReadPermissions();
                        }
                    });
                    builder.setNegativeButton("Quit app", new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            finish();
                            System.exit(0); //FORCE QUIT THE APP
                        }
                    });

                }
                break;
            }
            case PErMISSIONS_REQUEST_WRITE_EXTERNAL: {
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED)
{
                    Log.d(TAG, "External write permissions granted");
                    getInternetPermissions();
                }else{

                    AlertDialog.Builder builder = new
AlertDialog.Builder(context);
                    builder.setTitle("Internal Read Permission Denied");
                    builder.setMessage("Internal reading services are required in
order for the app to run");
                    builder.setPositiveButton("Try agian", new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
```

```
                        getExternalWritePermissions();
                    }
                });
                builder.setNegativeButton("Quit app", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        finish();
                        System.exit(0); //FORCE QUIT THE APP
                    }
                });

            }
            break;
        }

        case PERMISSIONS_REQUEST_INTERNET: {
            if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED)
{
                Log.d(TAG, "Internet permissions granted");
            }else{

                AlertDialog.Builder builder = new
AlertDialog.Builder(context);
                builder.setTitle("Internet Permission Denied");
                builder.setMessage("Internet services are required in order
for the app to run");
                builder.setPositiveButton("Try agian", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        getInternetPermissions();
                    }
                });
                builder.setNegativeButton("Quit app", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        finish();
                        System.exit(0); //FORCE QUIT THE APP
                    }
                });

            }
            break;
        }
    }
}
```

```
    /**
     * Updates the status of the mLocationServicesEnabled
     * flag based on network and GPS location services
     */
    public void checkLocationServices(){
        LocationManager manager =
(LocationManager)context.getSystemService(Context.LOCATION_SERVICE);
        boolean gpsEnabled = false;
        boolean networkEnabled = false;

        try {
            gpsEnabled = manager.isProviderEnabled(LocationManager.GPS_PROVIDER);
        } catch(Exception ex) {
            Log.d(TAG, "GPS location not enabled");
        }

        try {
            networkEnabled =
manager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
        } catch(Exception ex) {
            Log.d(TAG, "Network location not enabled");
        }

        if(gpsEnabled || networkEnabled){
            mLocationServicesEnabled = true;
        }
    }

    /**
     * Opens a dialog that prompts the user to enable
     * location services on their device
     */
    public void getLocationServices(){

        // notify user
        final AlertDialog.Builder dialog = new AlertDialog.Builder(context);
        dialog.setTitle("Enable location services");
        dialog.setMessage("Location services are needed for the app to run. Hit
the back button once you enable location services");
        dialog.setPositiveButton("Enable", new DialogInterface.OnClickListener()
{
            @Override
            public void onClick(DialogInterface paramDialogInterface, int
paramInt) {
                Intent myIntent = new Intent(
Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                context.startActivity(myIntent);
            }
        });
```

```
        dialog.setNegativeButton("Close app", new
DialogInterface.OnClickListener() {

            @Override
            public void onClick(DialogInterface paramDialogInterface, int
paramInt) {
                finish();
                System.exit(0); //FORCE QUIT THE APP
            }
        });
        dialog.show();
    }


    /**
     * Asks the user to allow location
     * services to be enabled for the app
     * @param context
     */
    private void displayLocationSettingsRequest(Context context) {
        GoogleApiClient googleApiClient = new GoogleApiClient.Builder(context)
                .addApi(LocationServices.API).build();
        googleApiClient.connect();

        LocationRequest locationRequest = LocationRequest.create();
        locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        locationRequest.setInterval(10000);
        locationRequest.setFastestInterval(10000 / 2);

        LocationSettingsRequest.Builder builder = new
LocationSettingsRequest.Builder().addLocationRequest(locationRequest);
        builder.setAlwaysShow(true);

        PendingResult<LocationSettingsResult> result =
LocationServices.SettingsApi.checkLocationSettings(googleApiClient,
builder.build());
        result.setResultCallback(new ResultCallback<LocationSettingsResult>() {
            @Override
            public void onResult(LocationSettingsResult result) {
                final Status status = result.getStatus();
                switch (status.getStatusCode()) {
                    case LocationSettingsStatusCodes.SUCCESS:
                        Log.i(TAG, "All location settings are satisfied.");
                        break;
                    case LocationSettingsStatusCodes.RESOLUTION_REQUIRED:
                        Log.i(TAG, "Location settings are not satisfied. Show the
user a dialog to upgrade location settings ");

                        try {
```

```
                            // Show the dialog by calling
startResolutionForResult(), and check the result
                            // in onActivityResult().
                            status.startResolutionForResult(Disclaimer.this,
PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION);
                        } catch (IntentSender.SendIntentException e) {
                            Log.i(TAG, "PendingIntent unable to execute
request.");
                        }
                        break;
                    case LocationSettingsStatusCodes.SETTINGS_CHANGE_UNAVAILABLE:
                        Log.i(TAG, "Location settings are inadequate, and cannot
be fixed here. Dialog not created.");
                        break;
                }
            }
        });
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
        switch (requestCode) {
            // Check for the integer request code originally supplied to
startResolutionForResult().
            case PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION:
                switch (resultCode) {
                    case Activity.RESULT_OK:
                        Log.i(TAG, "User agreed to make required location settings
changes.");
                        break;
                    case Activity.RESULT_CANCELED:
                        Log.i(TAG, "User chose not to make required location
settings changes.");
                        displayLocationSettingsRequest(this);
                        break;
                }
                break;
        }
    }
}
```

/**********End Disclaimer.java**********/

/**********ScanBtActivity.java**********/

```
package ca.ubc.zachrivard.self.test;

import android.annotation.TargetApi;
```

```
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Handler;
import android.provider.Settings;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.Toolbar;

import java.util.Set;

@TargetApi(25)
public class ScanBtActivity extends AppCompatActivity {

    // Return Intent extra
    public static String EXTRA_DEVICE_ADDRESS = "device_address";
    public static String EXTRA_DEVICE_NAME = "device_name";
    private static final int REQUEST_ENABLE_BT = 2;
    private static final int BT_SCAN_TIME = 15000; //15 seconds

    // Member fields
    private BluetoothAdapter mBtAdapter;
    private ArrayAdapter<String> mPairedDevicesArrayAdapter,
mScannedDevicesArrayAdapter;
    Set<BluetoothDevice> pairedDevices;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_scan_bt);

        Toolbar toolbar = (Toolbar) findViewById(R.id.btscantoolbar);
        toolbar.setTitle("Classic Bluetooth Scan");
        setActionBar(toolbar);

        // Get the local Bluetooth adapter
        mBtAdapter = BluetoothAdapter.getDefaultAdapter();
```

```
        // If the adapter is null, then Bluetooth is not supported
        if (mBtAdapter == null) {
            Toast.makeText(this, "Bluetooth is not available",
                    Toast.LENGTH_LONG).show();
            finish();
            return;
        }
        // If BT is not on, request that it be enabled.
        if (!mBtAdapter.isEnabled()) {
            Intent enableIntent = new Intent(
                    BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
        }

        // Register for broadcasts when a device is discovered.
        IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
        registerReceiver(mReceiver, filter);


        // Initialize the button to perform device discovery
        Button scanButton = (Button) findViewById(R.id.button_scan);
        scanButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mScannedDevicesArrayAdapter.clear();
                mScannedDevicesArrayAdapter.notifyDataSetChanged();
                Toast.makeText(getApplicationContext(), String.format("Device
Scan enabled for %d seconds", BT_SCAN_TIME/1000),Toast.LENGTH_LONG).show();
                mBtAdapter.startDiscovery();

                new Handler().postDelayed(new Runnable() {
                    @Override
                    public void run() {
                        if(mBtAdapter.isDiscovering()) {
                            mBtAdapter.cancelDiscovery();
                            Toast.makeText(getApplicationContext(), "Scan
Finished", Toast.LENGTH_SHORT).show();
                        }
                    }
                },BT_SCAN_TIME);
            }
        });

        // Initialize array adapters. One for already paired devices and
        // one for newly discovered devices
        mPairedDevicesArrayAdapter = new ArrayAdapter<>(this,
R.layout.device_name);
        mScannedDevicesArrayAdapter = new ArrayAdapter<>(this,
R.layout.device_name);
```

```
        // Find and set up the ListView for paired devices
        ListView pairedListView = (ListView) findViewById(R.id.paired_devices);
        pairedListView.setAdapter(mPairedDevicesArrayAdapter);
        pairedListView.setOnItemClickListener(mDeviceClickListener);
        updatePairedDevices();

        // Find and set up the ListView for scanned devices devices
        ListView scannedListView = (ListView) findViewById(R.id.new_devices);
        scannedListView.setAdapter(mScannedDevicesArrayAdapter);
        scannedListView.setOnItemClickListener(mDeviceClickListener);
    }

    protected void updatePairedDevices(){
        // Get a set of currently paired devices
        pairedDevices = mBtAdapter.getBondedDevices();

        // If there are paired devices, add each one to the ArrayAdapter
        if (pairedDevices.size() > 0) {
            mPairedDevicesArrayAdapter.clear();
            findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);
            for (BluetoothDevice device : pairedDevices) {
                mPairedDevicesArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
            }
        } else {
            String noDevices = "NO PAIRED DEVICES";
            mPairedDevicesArrayAdapter.add(noDevices);
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        // Make sure we're not doing discovery anymore
        if (mBtAdapter != null) {
            mBtAdapter.cancelDiscovery();
            unregisterReceiver(mReceiver);
        }
    }

    // The on-click listener for all devices in the ListViews
    private final AdapterView.OnItemClickListener mDeviceClickListener = new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3) {
            if(mBtAdapter.isDiscovering()) {
                // Cancel discovery because it's costly and we're about to connect
                mBtAdapter.cancelDiscovery();
```

```
                Toast.makeText(getApplicationContext(), "Scan
Terminated",Toast.LENGTH_SHORT).show();
            }

            if(av.getId() == R.id.new_devices){
                Toast.makeText(getApplicationContext(),"Please Accept Pairing
Request", Toast.LENGTH_SHORT).show();
            }

            // Get the device MAC address, and name
            String info = ((TextView) v).getText().toString();
            //address is last 17 chars of the view
            String address = info.substring(info.length() - 17);
            //name is up to the \n
            String name = info.substring(0, info.indexOf('\n') - 1);

            // Create the result Intent and include the MAC address + name
            Intent intent = new Intent(getApplicationContext(),
MapsActivity.class);
            intent.putExtra(EXTRA_DEVICE_ADDRESS, address);
            intent.putExtra(EXTRA_DEVICE_NAME, name);
            startActivity(intent);
        }
    };

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        switch (requestCode) {
            case REQUEST_ENABLE_BT:
                // When the request to enable Bluetooth returns
                if (resultCode != Activity.RESULT_OK) {
                    // User did not enable Bluetooth or an error occurred
                    Toast.makeText(this, "Bluetooth must be enabled",
                            Toast.LENGTH_SHORT).show();
                    finish();
                }
                else {
                    //Make sure the user can see all the paired devices
                    updatePairedDevices();
                }
        }
    }

    // Create a BroadcastReceiver for ACTION_FOUND.
    private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                // Discovery has found a device. Get the BluetoothDevice
                // object and its info from the Intent.
```

```
                BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                String deviceName = device.getName();
                if(deviceName == null){
                    deviceName = "Device Name Not Available";
                }
                String deviceHardwareAddress = device.getAddress(); // MAC
address
                findViewById(R.id.title_new_devices).setVisibility(View.VISIBLE);
                mScannedDevicesArrayAdapter.add(deviceName + "\n" +
deviceHardwareAddress);
                mScannedDevicesArrayAdapter.notifyDataSetChanged();
            }
        }
    };
}
```

/**********End ScanBtActivity.java**********/

/**********MapsActivity.java**********/

```
package ca.ubc.zachrivard.self.test;

import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.location.Location;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Environment;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.text.InputType;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.FrameLayout;
import android.widget.LinearLayout;
import android.widget.TextView;
```

```
import android.widget.Toast;

import com.bluecreation.melody.SppService;
import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.CameraPosition;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;

import org.json.JSONArray;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

import static ca.ubc.zachrivard.self.test.JSONParser.decodePoly;
import static ca.ubc.zachrivard.self.test.ScanBtActivity.EXTRA_DEVICE_ADDRESS;
import static ca.ubc.zachrivard.self.test.ScanBtActivity.EXTRA_DEVICE_NAME;

/**
* An activity that displays a map showing the place at the device's current
location.
*/
public class MapsActivity extends AppCompatActivity implements
OnMapReadyCallback {
   private boolean SEND_REQS = true; //Disable this if you don't want to do GPS

   private static final String TAG = MapsActivity.class.getSimpleName();
   private static final String KEY_CAMERA_POSITION = "camera_position";
   private static final String KEY_LOCATION = "location";
   private static final long LOG_DELAY = 5000; //5 seconds
   private static String PHONE_CONTACT = "17787132535"; //Thanks Brennan
   private static String HOME_ADDRESS = "2356 Main Mall"; //Macleod Building

   private static String TYPE_START = "START";
```

```
    private static String TYPE_FINISH = "FINISH";
    private static String TYPE_PATH = "PATH";
    private static String TYPE_START_P = "START_P";
    private static String TYPE_FINISH_P = "FINISH_P";
    private static String TYPE_PAUSE = "PAUSE";

    String currentType = TYPE_START;
    static int rideNumber;
    long rideStartTime;



    //URLs to execute
    String UPLOAD_URL = "http://intellibikeubc.com/uploadFile.php?type="; //need
to append a type ("history", or "log");
    //Data logging
    String TO_TABLE_URL = "http://intellibikeubc.com/loadFileIntoTable.php?type=";
//need to append a table and a type
    String makeKMLURL = "http://intellibikeubc.com/createKMLwithID.php?id=";
//need to append the id

    private final Context context = this;
    private  GoogleMap mMap;
    private CameraPosition mCameraPosition;


    private volatile boolean isLogging = true;



    // The entry point to the Fused Location Provider.
    private FusedLocationProviderClient mFusedLocationProviderClient;

    // A default location (Sydney, Australia) and default zoom to use when location
permission is
    // not granted.
    private final LatLng mDefaultLocation = new LatLng(-33.8523341, 151.2106085);
    private static final int DEFAULT_ZOOM = 15;


    // The geographical location where the device is currently located.
    private Location mLastKnownLocation;
    private LatLng currentLatLng;

    //String representation of the route start and end points
    private String startLocation;
    private String endLocation;

    //Line that will show the route on the map
    private Polyline polyline;
```

```
//Bluetooth stuff
SppService mSppService;

BluetoothAdapter mBluetoothAdapter;
String deviceAddress, deviceName;

ProgressDialog btDialog;


//Data logging
//Path to the log file
public String path = Environment
        .getExternalStorageDirectory()
        .getAbsolutePath() + "/intellibike";
File directory;
File logFile;
File rideNumberFile;
File rideHistoryFile;



/**
 * PRECONDITION
 *      Location services are enabled and turned on
 *      Location permissions are granted
 *
 * Both of these conditions are forced to be satisfied
 * by the Disclaimer class as it will not let users
 * through without it.
 */
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    directory = new File(path);
    logFile = new File(path + "/datalog.txt");
    rideNumberFile = new File(path + "/ridenum.txt");
    rideHistoryFile = new File(path + "/ridehistory.txt");

    // Get local Bluetooth adapter
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

    // If the adapter is null, then Bluetooth is not supported
    if (mBluetoothAdapter == null) {
        Toast.makeText(this, "Bluetooth is not available",
                Toast.LENGTH_LONG).show();
        finish();
        return;
    }
```

```
        Bundle extras = getIntent().getExtras();

        if (extras!= null) {

            btDialog = new ProgressDialog(context);
            btDialog.setTitle("Connecting to Device");
            btDialog.setMessage("Please wait...");
            btDialog.setIndeterminate(true);
            btDialog.show();


            deviceAddress = extras.getString(EXTRA_DEVICE_ADDRESS);
            deviceName = extras.getString(EXTRA_DEVICE_NAME);

            mSppService = SppService.getInstance();
            mSppService.registerListener(sppListener);

            BluetoothDevice device =
mBluetoothAdapter.getRemoteDevice(deviceAddress);
            try {
                mSppService.connect(device);
            } catch (Exception e) {
                Log.e(TAG, e.getMessage());
            }
        }


        // Retrieve location and camera position from saved instance state.
        if (savedInstanceState != null) {
            mLastKnownLocation = savedInstanceState.getParcelable(KEY_LOCATION);
            mCameraPosition =
savedInstanceState.getParcelable(KEY_CAMERA_POSITION);
        }

        // Retrieve the content view that renders the map.
        setContentView(R.layout.activity_maps);

        //will only init map once BT is connected

        final Button endRideButton = (Button) findViewById(R.id.endRideButton);
        endRideButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

if(endRideButton.getText().toString().equals(getString(R.string.start_ride))){
                    endRideButton.setText(getString(R.string.end_ride));
                    isLogging = true;
                    startRide();
                    currentType = TYPE_START;
```

```
                initPiData();
            }else {
                endRideButton.setText(getString(R.string.start_ride));
                currentType = TYPE_FINISH;
                endRide();
                initPiData();
            }
        }
    });

    final Button changeStateButton = (Button)
findViewById(R.id.pauseRideButton);
    changeStateButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            if(currentType.equals(TYPE_PATH)){ //Pause
                currentType = TYPE_FINISH_P;
                initPiData();
                changeStateButton.setText(getString(R.string.resume_ride));
            }else if (currentType.equals(TYPE_PAUSE)){ //Unpause
                currentType = TYPE_START_P;
                initPiData();
                changeStateButton.setText(getString(R.string.pause_ride));
            }
        }
    });

}


/**
 * Creates and initializes all of the objects needed for the
 * Google Maps API to function correctly.
 */
protected void initializeMap(){
    // Construct a FusedLocationProviderClient.
    mFusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(context);

    // Build the map.
    SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
            .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
}


/**
 * Begins the ride sequence by activating a
```

```
     * GPS logger if there was no destination
     * selected.
     */
    private void startRide(){
        updateRideNumberAndTime();
        isLogging = true;
        runLoggingThread();
    }


    /**
     * Begins the end ride sequence including
     */
    private void endRide(){
        isLogging = false;
    }

    /**
     * Saves the state of the map when the activity is paused.
     */
    @Override
    protected void onSaveInstanceState(Bundle outState) {
        if (mMap != null) {
            outState.putParcelable(KEY_CAMERA_POSITION,
mMap.getCameraPosition());
            outState.putParcelable(KEY_LOCATION, mLastKnownLocation);
            super.onSaveInstanceState(outState);
        }
    }

    /**
     * Sets up the options menu.
     * @param menu The options menu.
     * @return Boolean.
     */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.current_place_menu, menu);
        return true;
    }

    /**
     * Handles a click on the menu option to get a place.
     * @param item The menu item to handle.
     * @return Boolean.
     */
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()){
```

```
            case R.id.go_home:
                goHome();
                break;

            case R.id.change_home:
                updateHomeAddress();
                break;

            case R.id.call_contact:
                callContact();
                break;

            case R.id.change_phone:
                updatePhoneNumber();
                break;

            case R.id.new_dest:
                makeNewRoute();
                break;

            case R.id.upload_ride:
                uploadLogToServer();
                break;
            default:
                return false; //Failed
        }
        return true;
    }

    /**
     * Immediately begins a route from
     * the current location back to
     * HOME_ADDRESS
     */
    private void goHome(){
        startLocation = ""; //Will default to use current location
        endLocation = HOME_ADDRESS;

        extractDesiredRoute();
    }

    /**
     * Generates a dialog into which users can
     * update their home address
     */
    private void updateHomeAddress(){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Update Home Address");

        // Set up the input
```

```
        final EditText input = new EditText(context);
        // Specify the type of input expected; this, for example, sets the input
as a password, and will mask the text
        input.setInputType(InputType.TYPE_CLASS_TEXT);
        input.setHint(getString(R.string.update_field));
        builder.setView(input);

        // Set up the buttons
        builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                HOME_ADDRESS = input.getText().toString().trim();
                Toast.makeText(context, "Home address successfully updated",
Toast.LENGTH_LONG).show();
            }
        });
        builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener()
{
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });

        builder.show();
    }

    /**
     * Calls the emergency contact
     */
    private void callContact(){
        Intent callIntent = new Intent(Intent.ACTION_CALL);
        callIntent.setData(Uri.parse("tel:"+ PHONE_CONTACT));
        try {
            startActivity(callIntent);
        }catch (SecurityException e){
            Log.e(TAG, e.getMessage());
        }

    }

    /**
     * Generates a dialog into which users can
     * update their emergency cell number
     */
    private void updatePhoneNumber(){

        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Update Emergency Contact");
```

```
        // Set up the input
        final EditText input = new EditText(context);
        // Specify the type of input expected; this, for example, sets the input
as a password, and will mask the text
        input.setInputType(InputType.TYPE_CLASS_TEXT);
        input.setHint(getString(R.string.update_field));
        builder.setView(input);

        // Set up the buttons
        builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                PHONE_CONTACT = input.getText().toString().trim();
                PHONE_CONTACT = PHONE_CONTACT.replaceAll("-", "");
                PHONE_CONTACT = PHONE_CONTACT.replaceAll("\\.", "");
                PHONE_CONTACT = PHONE_CONTACT.replaceAll("\\(", "");
                PHONE_CONTACT = PHONE_CONTACT.replaceAll("\\)", "");
                Toast.makeText(context, "Emergency contact successfully updated",
Toast.LENGTH_LONG).show();
            }
        });
        builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener()
{
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });

        builder.show();

    }


    /**
     * Opens a dialog for the user to input
     * a new start and end destination
     */
    private void makeNewRoute(){

        final AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Plan Route");

        LinearLayout layout = new LinearLayout(context);
        View child = getLayoutInflater().inflate(R.layout.new_route_dialog,
null);
        layout.addView(child);


        // Add a TextView here for the startLocation
```

```
        final EditText origin = (EditText) child.findViewById(R.id.origin_field);
        // Add another TextView here for the endLocation
        final EditText destination= (EditText)
child.findViewById(R.id.dest_field);

        builder.setView(layout); // Again this is a set method, not add

        builder.setPositiveButton("Go!", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                startLocation = origin.getText().toString();
                endLocation = destination.getText().toString();

                extractDesiredRoute();
            }
        });

        builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener()
{
            @Override
            public void onClick(DialogInterface dialog, int which) {
                //Do nothing :(
            }
        });

        builder.show();

    }


    /**
     * Manipulates the map when it's available.
     * This callback is triggered when the map is ready to be used.
     */
    @Override
    public void onMapReady(GoogleMap map) {
        mMap = map;

        // Use a custom info window adapter to handle multiple lines of text in
the
        // info window contents.
        mMap.setInfoWindowAdapter(new GoogleMap.InfoWindowAdapter() {

            @Override
            // Return null here, so that getInfoContents() is called next.
            public View getInfoWindow(Marker arg0) {
                return null;
            }

            @Override
```

```
            public View getInfoContents(Marker marker) {
                // Inflate the layouts for the info window, title and snippet.
                View infoWindow =
getLayoutInflater().inflate(R.layout.custom_info_contents,
                        (FrameLayout) findViewById(R.id.map), false);

                TextView title = ((TextView)
infoWindow.findViewById(R.id.title));
                title.setText(marker.getTitle());

                TextView snippet = ((TextView)
infoWindow.findViewById(R.id.snippet));
                snippet.setText(marker.getSnippet());

                return infoWindow;
            }
        });

        // Turn on the My Location layer and the related control on the map.
        updateLocationUI();

        // Get the current location of the device and set the position of the map.
        updateDeviceLocation();

        //Begin plotting a new route
        makeNewRoute();
    }


    /**
     * Gets the current location of the device,
     * and positions the map's camera.
                                    *
     */
    private void updateDeviceLocation(){

        try{
            Task<Location> locationResult =
mFusedLocationProviderClient.getLastLocation();
            locationResult.addOnCompleteListener(this, new
OnCompleteListener<Location>() {
                @Override
                public void onComplete(@NonNull Task<Location> task) {
                    if(task.isSuccessful()){
                        if (task.isSuccessful()) {
                            // Set the map's camera position to the current
location of the device.
                           Location location = task.getResult();
                            currentLatLng = new LatLng(location.getLatitude(),
location.getLongitude());
```

```
mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(currentLatLng, DEFAULT_ZOOM));

                        } else {
                            Log.d(TAG, "Current location is null. Using
defaults.");
                            Log.e(TAG, "Exception: %s", task.getException());
                            mMap.moveCamera(CameraUpdateFactory
                                    .newLatLngZoom(mDefaultLocation,
DEFAULT_ZOOM));

mMap.getUiSettings().setMyLocationButtonEnabled(false);
                            currentLatLng = mDefaultLocation;

                        }
                    }
                }
            });
        }catch (SecurityException e){
            Log.e(TAG, e.getMessage());
        }

    }


    /**
     * Makes the URL based on the start and end points and
     * calls to send off the URL via HTTPS
     *
     * Requires:
     *              startLocation and endLocation have been updated
     *              to the desired string representation of the
     *              origin/destination
     *
     */
    private void extractDesiredRoute() {

        //If any of the fields are blank, use the current
        //location to fill them

        if(startLocation.length() < 1) {
            startLocation = currentLatLng.toString(); //In the form of (lat,lng)
            startLocation = startLocation.substring(startLocation.indexOf('(') +
1, startLocation.lastIndexOf(')'));
        }

        if(endLocation.length() < 1) {
            endLocation = currentLatLng.toString(); //In the form of (lat,lng)
            endLocation = endLocation.substring(endLocation.indexOf('(') + 1,
endLocation.lastIndexOf(')'));
```

```
    }

    //Make the URL for the API call
    String url = makeURL(startLocation, endLocation);
    Log.d(TAG, url);

    if(SEND_REQS)
        new connectAsyncTask(url).execute(); //Make the request
}


/**
 * Class that will make the HTTPS requests to Google Directions
 */

private class connectAsyncTask extends AsyncTask<Void, Void, String> {
    private ProgressDialog progressDialog;
    String url;

    connectAsyncTask(String urlPass) {
        url = urlPass;
    }

    @Override
    protected void onPreExecute() {
        // TODO Auto-generated method stub
        super.onPreExecute();
        progressDialog = new ProgressDialog(context);
        progressDialog.setMessage("Fetching route, Please wait...");
        progressDialog.setIndeterminate(true);
        progressDialog.show();
    }

    @Override
    protected String doInBackground(Void... params) {
        JSONParser jParser = new JSONParser();
        String json = jParser.getJsonStringFromUrl(url);
        return json;
    }

    @Override
    protected void onPostExecute(String result) {
        super.onPostExecute(result);
        progressDialog.hide();
        if (result != null) {
            drawPath(result);
        }
    }
}
```

```
    /**
     * Draws a line on the map representing the best route
     * to take to get from origin to destination.
     *
     * Begins the data logger
     *
     * @param result - the String representation of the line
     */
    public void drawPath(String result) {
        if(SEND_REQS) {
            if (polyline != null) {
                mMap.clear();
            }

            startRide();

            try {
                // Tranform the string into a json object
                final JSONObject json = new JSONObject(result);
                JSONArray routeArray = json.getJSONArray("routes");
                JSONObject routes = routeArray.getJSONObject(0);
                JSONObject overviewPolylines =
routes.getJSONObject("overview_polyline");
                String encodedString = overviewPolylines.getString("points");

                List<LatLng> list = decodePoly(encodedString); //Decode the line

                PolylineOptions options = new
PolylineOptions().color(Color.BLUE);
                for (int i = 0; i < list.size(); i++) {
                    LatLng point = list.get(i);
                    options.add(point);
                }

                polyline = mMap.addPolyline(options);

                //Get the exact start and end locations
                JSONObject legsArray =
routes.getJSONArray("legs").getJSONObject(0);

                JSONObject endLoca = legsArray.getJSONObject("end_location");
                String endAddress = legsArray.getString("end_address");
                endAddress = endAddress.substring(0, endAddress.indexOf(','));
                String endLat = endLoca.getString("lat");
                String endLng = endLoca.getString("lng");

                JSONObject startLoc = legsArray.getJSONObject("start_location");
                String startAddress = legsArray.getString("start_address");
```

```
                startAddress = startAddress.substring(0,
startAddress.indexOf(','));
                String startLat = startLoc.getString("lat");
                String startLng = startLoc.getString("lng");

                LatLng originLatLng = new LatLng(Double.valueOf(startLat),
Double.valueOf(startLng));
                LatLng destLatLng = new LatLng(Double.valueOf(endLat),
Double.valueOf(endLng));

                plotLocationOnMap(originLatLng, startAddress);
                plotLocationOnMap(destLatLng, endAddress);

            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }



    /**
     * Adds a marker on the map at the desired location
     *
     * @param loc - Location on the map where the marker will be placed
     * @param title - Title for the marker when clicked on
     */
    private void plotLocationOnMap(LatLng loc, String title){
        if(mMap == null)
            return;


        if(SEND_REQS) {
            // Add a marker for the selected place, with an info window
            // showing information about that place.
            mMap.addMarker(new MarkerOptions()
                    .title(title)
                    .position(loc));
        }
    }


    /**
     * Updates the map's UI settings based on whether the user has granted location
permission.
     */
    private void updateLocationUI() {
        if (mMap == null) {
            return;
        }
```

```
    if(SEND_REQS) {
        try {
            mMap.setMyLocationEnabled(true);
            mMap.getUiSettings().setMyLocationButtonEnabled(true);
        } catch (SecurityException e) {
            Log.e("Exception: %s", e.getMessage());
        }
    }
}


/**
 *
 * @param start - the starting location of the ride
 * @param end - the end location of the ride
 * @return the String representation of the URL that needs to be send to
 *              determine the best route to take
 */
private String makeURL(String start, String end){
    StringBuilder url = new StringBuilder();
    start = start.trim().replaceAll("\\s+", "+"); //Format for google API
    end = end.trim().replaceAll("\\s+", "+");

    url.append("https://maps.googleapis.com/maps/api/directions/json");
    url.append("?origin=");// from
    url.append(start);
    url.append("&destination=");// to
    url.append(end);
    url.append("&mode=bicycling"); //We are biking
    url.append("&key="); //Add our API key
    url.append(getString(R.string.API_KEY));

    return url.toString();
}


/**
 *Bluetooth listener that will write the retrieved
 * data from the Pi back into the log file
 */
SppService.Listener sppListener = new SppService.Listener() {
    @Override
    public void onStateChanged(final SppService.ConnectionState state) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if(state.equals(SppService.ConnectionState.STATE_NONE)){
                    finish();
                }
```

```
                }
            });
        }

        @Override
        public void onRemoteDeviceConnected(final String deviceName) {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    if(btDialog != null && btDialog.isShowing()) {
                        btDialog.dismiss();
                    }
                    initializeMap();
                }
            });
        }

        @Override
        public void onDataReceived(final byte[] data, final int length) {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {

                    String rideData = new String(data, 0, length);
                    /*
                        TEXT file format (spaces added for clarity, there will be
none in the file)

                        rideNum | "TYPE" | COOR_lat | COOR_lng | time | COOR_alt |
speed | distance | acceleration |

|_____________________________________________|_____________________________________
___________|
                                                |
|
                                            Supplied by App
Supplied by Pi
                     */
                    String appData = String.valueOf(rideNumber) + "|" +
currentType + "|" + getLatitude()
                            + "|" + getLongitude() + "|" +
String.valueOf(System.currentTimeMillis()/1000);

                    rideData = appData + rideData;
                    Log.d("Data update... ", rideData);
                    //Write it into a file

                    if(!currentType.equals(TYPE_PAUSE)) { //If paused do not log
                        writeDataToLog(rideData);
                    }
```

```
                    if(currentType.equals(TYPE_START) ||
currentType.equals(TYPE_START_P)){ //Next will always be a normal path
                        currentType = TYPE_PATH;
                    }

                    if(currentType.equals(TYPE_FINISH_P)){ //We are going into a
pause
                        currentType = TYPE_PAUSE;
                    }


                }
            });

        }

        @Override
        public void onConnectionLost() {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(getApplicationContext(), "Connection Lost",
                            Toast.LENGTH_SHORT).show();
                    finish();
                }
            });
        }

        @Override
        public void onConnectionFailed() {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(getApplicationContext(),
                            "Connection Failed", Toast.LENGTH_SHORT).show();
                    finish();
                }
            });
        }
    };

    private void writeDataToLog(String data){
        FileOutputStream outputStream;
        try{
            outputStream = new FileOutputStream(logFile, true);
            outputStream.write((data + "\n").getBytes());
            outputStream.close();
        }catch(IOException e){
            e.printStackTrace();
```

```
        }

        Log.d(TAG, "Data write to file!");

    }



    /**
     * Will begin a separate thread to log GPS every LOG_DELAY ms.
     *
     * The log will contain Lat/Lng and a timestamp
     */
    private void runLoggingThread(){

        if(SEND_REQS) {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    while (isLogging) {
                        //Init the data request
                       initPiData();
                        new Thread(new Runnable() {
                            @Override
                            public void run() {
                                updateDeviceLocation(); //This is a really slow
thing...
                           }
                        }).start();

                        try {
                            Thread.sleep(LOG_DELAY);
                        } catch (InterruptedException e) {
                            Log.d(TAG, "Logging thread was interrupted");
                        }
                    }
                }
            }).start();
        }

    }

    /**
     * Initiate the echange of data between the
     * phone and the Pi
     */
    private void initPiData(){
        mSppService.send(("Data").getBytes());
    }
```

```
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // Stop the melody services
        if (mSppService != null) {
            mSppService.unregisterListener(sppListener);
            mSppService.stop();
            mSppService = null;
        }
    }


    /**
     * Reads the current ride number from a file
     * and increments it for the next ride
     */
    private void updateRideNumberAndTime(){
        FileOutputStream outputStream;
        FileInputStream inputStream;
        String contents = "";


        if(!rideNumberFile.exists()) { //Init the file if this is the first time
            Log.d("File did not exist", "Weird");
            try {
                outputStream = new FileOutputStream(rideNumberFile, false);
                Log.d("Writing back... ", (String.valueOf(1)));
                outputStream.write((String.valueOf(1)).getBytes());
                outputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }


        try{ //Read the current ride number
            inputStream = new FileInputStream(rideNumberFile);
            InputStreamReader reader = new InputStreamReader(inputStream);
            BufferedReader bufferedReader = new BufferedReader(reader);

            contents = bufferedReader.readLine(); //There will only be one line
for ride num
            Log.d(TAG, "Read number from file is... " + contents);
            rideNumber = Integer.valueOf(contents); //Set the ride number
            rideStartTime = System.currentTimeMillis() / 1000; //The start time of
the ride
        }catch(IOException e){
            e.printStackTrace();
        }
```

```
        //Log this as a ride (ride_id, time_in_epoch_seconds)
        try {
            outputStream = new FileOutputStream(rideHistoryFile, false);
            Log.d("Add ride to history...",(String.valueOf(rideNumber) + "|" +
String.valueOf(rideStartTime)));
            outputStream.write((String.valueOf(rideNumber) + "|" +
String.valueOf(rideStartTime)).getBytes());
        }catch (IOException e){
            e.printStackTrace();
        }

        try { //Update the ride number for the next one
            outputStream = new FileOutputStream(rideNumberFile, false);
            Log.d("Writing back... ", (String.valueOf(rideNumber + 1)));
            outputStream.write((String.valueOf(rideNumber + 1)).getBytes());
            outputStream.close();
            outputStream = new FileOutputStream(logFile, false);
            Log.d("Clearing log... ", "wow");
            outputStream.write(("").getBytes());
            outputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

        Log.d("Current ride number:", String.valueOf(rideNumber));
        Log.d("Current ride ST:", String.valueOf(rideStartTime));
    }


    /**     *
     * @return The string representation of the current
     * latitude
     */
    private String getLatitude(){
        return String.valueOf(currentLatLng.latitude);
    }


    /**     *
     * @return The string representation of the current
     * longitude
     */
    private String getLongitude(){
        return String.valueOf(currentLatLng.longitude);
    }


    /**
     * Pushes the log files to the server
     * and calls scripts to make the KML
```

```
     * based off of the log
     */
    private boolean uploadLogToServer (){

        String historyURL = TO_TABLE_URL + "history&table=ride_history";
        String logURL = TO_TABLE_URL + "log&table=logged_data";


        //Add all of the HTTP requests to be sent in sequential order
        ArrayList<ScriptExecutor> list = new ArrayList<>();
        list.add( new ScriptExecutor(UPLOAD_URL + "history", path +
"/ridehistory.txt", context)); //Send the ride history)
        list.add( new ScriptExecutor(UPLOAD_URL + "log", path + "/datalog.txt",
context)); //Send the log)
        list.add( new ScriptExecutor(historyURL, null, context)); //Upload the
history into a table);
        list.add( new ScriptExecutor(logURL, null, context)); //Upload the log
into a table);
        Log.d("RIDE NUMBER PRE ADD", String.valueOf(rideNumber));
        list.add( new ScriptExecutor(makeKMLURL + String.valueOf(rideNumber),
null, context)); //Make all the kmls);

        //Pause to allow all the objects to me made
        try {
            Thread.sleep(1000);
        }catch(InterruptedException e){
            e.printStackTrace();
        }

        //Build the caller
        ScriptExecutor.multiScript(list);

        try {
            Thread.sleep(1000);
        }catch(InterruptedException e){
            e.printStackTrace();
        }

        //Execute all of the scripts
        ScriptExecutor.run();

        return false; //return httpResult = 200;
    }
}
```

/**********End MapsActivity.java**********/

/**********ScriptExecutor.java**********/

```java
package ca.ubc.zachrivard.self.test;

import android.content.Context;
import android.os.AsyncTask;
import android.util.Log;
import android.widget.Toast;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.mime.HttpMultipartMode;
import org.apache.http.entity.mime.MultipartEntity;
import org.apache.http.entity.mime.content.FileBody;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.protocol.BasicHttpContext;
import org.apache.http.protocol.HttpContext;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;

/**
* Created by zachr on 2018-04-03.
*/

public class ScriptExecutor extends AsyncTask<Void, Void, String> {


    String url;
    String filePath;
    Context context;
    static ArrayList<ScriptExecutor> list;
    static int i = 0;

    public ScriptExecutor(String url, String filePath, Context context) {
        this.url = url;
        this.context = context;
        this.filePath = filePath;
    }

    /**
     * Allows the sequentialization of
     * multipple AsyncTasks
     */
    public static void multiScript(ArrayList<ScriptExecutor> da){
        list = da;
    }

    /**
     * Run the current AsyncTask
```

```
     */
    public static void run(){
        ScriptExecutor s = list.get(i);

        s.execute();
    }

    HttpResponse response;
    @Override
    protected String doInBackground(Void... params) {
        HttpClient httpClient = new DefaultHttpClient();
        HttpContext localContext = new BasicHttpContext();
        HttpPost httpPost = new HttpPost(url);
        Log.d("URL", url);
        try {

            if(filePath != null){ //Add in the file
                MultipartEntity entity = new
MultipartEntity(HttpMultipartMode.BROWSER_COMPATIBLE);
                entity.addPart("image", new FileBody(new File(filePath)));
                httpPost.setEntity(entity);
            }

            response = httpClient.execute(httpPost, localContext);

        } catch (IOException e) {
            e.printStackTrace();
        }
        if(response == null) return null;
        return response.getStatusLine().toString();
    }

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);

        if(s.contains("200")){
            Toast.makeText(context, "Script executed successfully!",
Toast.LENGTH_LONG).show();

            //If there are more tasks to run, run the next one
            if(++i < list.size()){
                run();
            }

        }else{
            Toast.makeText(context, "Error in execution - Please try in internet
range", Toast.LENGTH_LONG).show();
        }
```

```
    }
}
```

/**************************************End ScriptExecutor.java**************************************/

Appendix G – GitHub/Version Control

Brennan currently has a disproportionate number of commits on GitHub since he was constantly logged into git on the RPi, even if other people were working on it. Michael also pushed under 'Michal' a couple times and 'o5d1b' instead of 'Gwatcha'. In addition, Maya did not appear as a contributor (despite having commits) for the majority of the project. She was officially added when she fixed the incorrect git configuration on her computer the Sunday before the demo. Joel has a disproportionately high number of deletions as he was the one who cleaned out all of the test and temporary files before submission.

Appendix H – Other code

********** Start Arduino Code for Arduino_Safety_System..ino **********

```
 #define FR_LED 5
#define FL_LED 6
#define B_LED  7
#define F_LED 4
#define PHOTOCELL_PIN A1
#define FSR A0
#define R_SWITCH 3
#define L_SWITCH 2
#define DARKNESS_THRESHOLD 700 // light level at which the ambient lighting
will turn on
#define PRESSURE_THRESHOLD 500 // can adjust as needed
#define ON 1
#define OFF 0

//interrupt global variables
int rTurnState = OFF;
int lTurnState = OFF;
int RTURNBUFFER = OFF;
int LTURNBUFFER = OFF;


void setup() {
  // put your setup code here, to run once:
  pinMode(FR_LED, OUTPUT);
  pinMode(FL_LED, OUTPUT);
```

```
  pinMode(B_LED, OUTPUT);
  pinMode(F_LED, OUTPUT);
  //pinMode(PHOTOCELL_PIN, OUTPUT);

  pinMode(FSR, INPUT);
  pinMode(R_SWITCH, INPUT);
  pinMode(L_SWITCH, INPUT);

  Serial.begin(9600);

  attachInterrupt(1, RTurnISR, FALLING); //interrupt for right turn signal
  attachInterrupt(0, LTurnISR, FALLING); //interrupt for left turn signal

}

void loop() {

  if (checkPressure(FSR)) //check force sensitive resistor
      digitalWrite(B_LED, HIGH); // turn on brake light
  else
      digitalWrite(B_LED, LOW);

  if (isDark()) // check photocell
      digitalWrite(F_LED, HIGH); // turn on if darker than a certain threshold
  else
      digitalWrite(F_LED, LOW);

  if (rTurnState == ON) // if right turn signal on
  {
      //blink
      digitalWrite(FR_LED, HIGH);
      delay(250);
      digitalWrite(FR_LED, LOW);
      delay(250);
  }
  else
  {
      //turn off
      digitalWrite(FR_LED, LOW);
  }

  // if left turn signal on
  if (lTurnState == ON)
  {

      digitalWrite(FL_LED, HIGH);
      delay(250);
      digitalWrite(FL_LED, LOW);
```

```
      delay(250);
  }
  else
  {
      digitalWrite(FL_LED, LOW);
  }
}

/*
   Returns true if darker than a certain threshold ambient light value
*/
bool isDark()
{
  int lightLevel = analogRead(PHOTOCELL_PIN);
  Serial.print("light reading = ");
  Serial.print(lightLevel); // the raw analog reading
  Serial.print("\n");
  if (lightLevel < DARKNESS_THRESHOLD)
      return true;
  return false;
}

/*
   Returns true if pressure is higher than a certain threshold value
*/
bool checkPressure(int fsrNum)
{
  int fsrData;
  fsrData = analogRead(fsrNum);
  Serial.print("FSR reading = ");
  Serial.print(fsrData); // the raw analog reading
  Serial.print("\n");
  if (fsrData > PRESSURE_THRESHOLD)
      return true;
  else
      return false;
}

/*
   Right turn signal interrupt service routine
   changes state after switch pressed
*/
void RTurnISR() {

  //        Serial.print(icount);
  //        icount++;
  //        Serial.println(" right interrupt");
```

```
  //rising edge
  if (RTURNBUFFER == 0 && digitalRead(R_SWITCH) == ON) {
      RTURNBUFFER = 1;
  }

  //falling edge
  if (RTURNBUFFER == 1 && digitalRead(R_SWITCH) == OFF) {
      RTURNBUFFER = 0;
      lTurnState = OFF;
      //switch state
      if (rTurnState == OFF) {
      rTurnState = ON;
      }
      else {
      rTurnState = OFF;
      }
  }
  delay(2000);
}

/*
   Left turn signal interrupt service routine
   changes state after switch pressed
*/
void LTurnISR() {

  // rising edge
  if (LTURNBUFFER == 0 && digitalRead(L_SWITCH) == ON) {
    LTURNBUFFER = 1;
  }

  // falling edge
  if (LTURNBUFFER == 1 && digitalRead(L_SWITCH) == OFF) {
      LTURNBUFFER = 0;
      rTurnState = OFF;
      //switch state
      if (lTurnState == OFF) {
      lTurnState = ON;
      }
      else {
      lTurnState = OFF;
    }
  }
  delay(2000);
}
```

********** End Arduino Code for Arduino_Safety_System..ino **********

********** Start Arduino Code for Arduino_Slave.ino **********

```
#include <math.h>
#include <Wire.h>
#include <Adafruit_MPL3115A2.h>
#include <LiquidCrystal.h>
// OLED
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1305.h>

// load current sea level pressure in Pa for more accurate altitude reading -
from https://vancouver.weatherstats.ca
float CURRENT_SEA_LEVEL_PRESSURE =  101900.0; // last updated Apr. 3, 05:19

// LCD pins
#define EN 11
#define RS 13
#define D4 9
#define D5 8
#define D6 7
#define D7 6

// OLED Pins ~~~
// Used for software SPI
#define OLED_CLK 12
#define OLED_MOSI 4

// Used for software or hardware SPI
#define OLED_CS 5
#define OLED_DC 2

// Used for I2C or SPI
#define OLED_RESET 3
// ~~~

// Software SPI
Adafruit_SSD1305 display(OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS); //
create SSD1305 object (OLED display)
int OLED_ON = 0;

// Altitmeter pins for i2c communication - these pins cannot be changed
#define SCL A5
#define SDA A4

LiquidCrystal lcd(RS, EN, D4, D5, D6, D7); // creates lcd object
```

```
Adafruit_MPL3115A2 baro = Adafruit_MPL3115A2(); // creates MPL3115A2 object
(Altitude sensor)
String state = "0";
bool newInput = false;

// function headers
double readAlt(float seaLevelPressure);
double readTemp();

void setup()
{
  Serial.begin(9600);
  lcd.begin(16, 2); // initializes lcd
  lcd.noCursor(); // turns off display cursor
}

void loop()
{
  if (Serial.available()) // if there is new data in the serial input buffer
  {
       state = Serial.readStringUntil('\0'); // all input strings are terminated
by null characters
       newInput = true;
  }

  if (newInput == true) {
       char encodingChar = state.charAt(0); // extracts first character which
contains operation modes

       delay(1000);
       if (encodingChar == 'S') { //special inputs
       lcd.clear();
       state = state.substring(1);
       lcd.print(state);
       }

       /*
       FORM: 1,1,1,36.1123,-32.22, 23210,
             // On/Off?, Recording?, Paused?, Velocity, Acceleration, Distance
(km)
       */
       else if (encodingChar == 'O') { //OLED functions

       updateOLED(state.substring(1));
       }
       else if (encodingChar == 'D') { //altitude data
       String sendStr = "";
       int x = (int) readAlt(CURRENT_SEA_LEVEL_PRESSURE);
```

```
        sendStr = String(x);
        sendStr += "\n";
        Serial.print(sendStr);
        }
        else {//just print state
        lcd.print(state);
        }

        newInput = false;
    }
}

void bootOLED() {
  display.begin();
  display.clearDisplay();

  //Welcome logo
  display.setTextSize(3);
  display.setTextColor(BLACK); // 'inverted' text
  display.setCursor(0, 10);
  drawexpandingcircle();
  delay(750);
  display.println("Intelli");
  display.println("  Bike");
  display.display();

  delay(3000);
  display.clearDisplay();
  rectAnimation();
  display.clearDisplay();
  display.display();
}

// Writes frame for sensor data (seperating lines and units), but does not
display it yet.
void mainScreenFrame() {
  display.drawLine(0, 14, 105, 14, WHITE);
  display.drawLine(0, 15, 105, 15, WHITE);

  display.drawLine(105, 0, 105, display.height(), WHITE);

  display.setTextSize(1);
  display.setTextColor(WHITE);

  // km
  display.setCursor(51, 4);
  display.print("km");
```

```
  // KM/HR
  display.setCursor(85, 22);
  display.print("km");
  display.drawLine(85, 30, 95, 30, WHITE);
  display.setCursor(85, 32);
  display.print("hr");

  // *C
  display.fillCircle(92, 4, 1, WHITE);
  display.setCursor(94, 4);
  display.print("C");

  // m/s^2
  display.setCursor(88, 44);
  display.print("m");
  display.drawLine(85, 52, 95, 52, WHITE);
  display.setCursor(84, 54);
  display.print("s");
  display.setCursor(92, 54);
  display.print("2");

  display.fillCircle(90, 54, 1, WHITE);
}

// ~~~~~~Start of OLED Functions
void updateOLED(String stateStr) {

  char state[50];
  stateStr.toCharArray(state, 50);

  // Variables to update.
  bool rec, paused;
  double vel, acc, distance, temp;
  temp = readTemp();

  int control = atoi(strtok(state, ","));

  // Start parsing.
  // Means we are turning off
  if (control == 0) {
      display.command(174);
      OLED_ON = 0;
  }

  else {

      // Turn on in the case it isn't.
      if (control == 1 && OLED_ON == 0) {
```

```
      bootOLED();
        OLED_ON = 1;
      }

      // If we are currently recording.
      if (atoi(strtok(NULL, ",")) == 1) {
      rec = true;
      }
      else {
      rec = false;
      }

      // If we are currently paused in recording mode.
      if (atoi(strtok(NULL, ",")) == 1)
        paused = true;
      else
      paused = false;

      vel = atof(strtok(NULL, ","));
      acc = atof(strtok(NULL, ","));
      distance = atof(strtok(NULL, ","));

      updateScreen(rec, paused, vel, acc, distance, temp);

  }
}

// Clears and updates the screen according to variables passed.
void updateScreen(bool recording, bool paused, double velocity, double
acceleration, double distance, double temperature) {
  //T his might be too slow, we might want to do rectangles instead.
  display.clearDisplay();
  mainScreenFrame();

  display.setTextSize(1);
  display.setTextColor(WHITE);


  if (!recording) {
      display.fillRect(2, 3, 8, 8, WHITE);
  }
  // Recording track indicator
  else if (!paused) {
      display.fillCircle(6, 7, 4, WHITE);
  }
  else {
      display.fillRect(2, 3, 3, 8, WHITE);
      display.fillRect(7, 3, 3, 8, WHITE);
```

```
  }

  // Distance
  if (distance / 100 >= 1)
      display.setCursor(13, 4);
  else if (distance / 10 >= 1)
      display.setCursor(18, 4);
  else
      display.setCursor(25, 4);

  display.print(distance, 2);

  // Temperature.
  if (temperature / 10 >= 1)
      display.setCursor(67, 4);
  else
      display.setCursor(73, 4);
  display.print(temperature, 1);

  // Velocity Spot (Adjust according to length
  display.setTextSize(3);
  if (velocity / 10 >= 1)
      display.setCursor(7, 20);
  else
      display.setCursor(25, 20);
  display.print(velocity, 1);

  // Acceleration
  display.setTextSize(2);
  if (acceleration / 10 <= -1)
      display.setCursor(6, 46);
  else if (acceleration / 10 >= 1 || acceleration < 0)
      display.setCursor(18, 46);
  else
      display.setCursor(30, 46);
  display.print(acceleration, 2);

  displayBar(velocity);
  display.display();
}
// Fun OLED functions
// Creates a visual bar indicating speed, with a max value of 50 and a minimum
of 0.
void displayBar(double velocity) {
  for (int i = 63; i > 3 && ((62 - i) < (velocity * 6 / 5)); i -= 2) {
      display.drawLine(107, i, 127, i, WHITE);
  }
}
```

```
// Rectangle - circle animation
void rectAnimation(void) {
  uint8_t color = BLACK;
  for (uint8_t i = 0; i < display.height() / 2; i += 3) {
      display.drawCircle(display.width() / 2, display.height() / 2, i , color);
      if (color == WHITE) color = BLACK;
      else color = WHITE;
      display.display();

      display.fillRect(i, i, display.width() - i * 2, display.height() - i * 2,
color);
      display.display();
  }
}

// draws expanding circle
void drawexpandingcircle(void) {
  for (uint8_t i = 0; i < display.height() + 7; i += 2) {
      display.drawCircle(display.width() / 2, display.height() / 2, i, WHITE);
      display.display();
  }
}

// ~~~~~~~~~~~~~~~~End of OLED Functions

// retreives altitude from the sensor
double readAlt(float seaLevelPressure)
{
  baro.begin();
  baro.setSeaPressure(seaLevelPressure); // set baseline pressure for more
accurate reading

  return (double) baro.getAltitude();
}

// retreives temperature from sensor
double readTemp()
{
  baro.begin();
  return (double) baro.getTemperature();
}
```

********** End Arduino Code for Arduino_Slave.ino **********

Appendix I - Intended Features & Code

We did not include internet controlled Pi functionality because, since the platform we were mounting the Pi on was intended to be very mobile and therefore would not often have a wifi connection, and there was no way of manually interfacing with the Pi while using the bike to connect it to a network. This is why we decided to use the mobile phone as our networking device, as it would have a 3G connection and therefore be more reliably connected and able to interact with the server.

While we were not able to test the app-enabled hardware control functionality before the demo, we did have code that we are fairly certain would work. This Python code could be called by the mobile phone and enable/disable the alarm, as well as change the password.

```
(Top Level.py, line 560)

def setArmedOverride():
  ALARM_ARMED = 1;

def setDisarmedOverride():
  ALARM_ARMED = 0;

#change password
def changePass(digit0, digit1, digit2, digit3):
      passcode[0] = digit0
      passcode[1] = digit1
      passcode[2] = digit2
      passcode[3] = digit3

def alarmOff():
      ALARM_BREAK = 1
```

We set up the hardware box with the intention of making the wires flush against the breadboard and cleaning up the connections between breadboards, as well as securing a clear, waterproof lid on the box. However, some finicky connections required more time than expected to debug, and we did not have time to complete these tasks.