Deployable Prototype Documentation

Nurthin Aziz, Khalil Javed, Amarjit Singh, Paramvir Singh

SACRAMENTO STATE

Contents

List of Figures

List of Tables

Deployable Prototype Documentation

Khalil Javed, Nurthin Aziz, Amarjit Singh, and Paramvir Singh
California State University Sacramento
Sacramento, CA 95835

***Abstract*—A smart wheelchair is a powered wheelchair with assistive technology such as computers and sensors. Although automobiles have been using this assistive technology for collision avoidance, lane detection, and self-driving capabilities, these technologies have not been integrated on a large scale with powered wheelchairs. This paper aims to guide the user into our attempt at integrating these smart functionalities in the powered wheelchairs of the future. We begin by reintroducing our societal problem covering the Fall and Spring semester. Then move to our design idea and the plans to integrate four major functionalities in a working prototype: collision detection, collision avoidance, tilt detection, and event notifications. We continue by outlining the funding and budget leading into our work breakdown structure and the mitigation steps we needed to take. Lastly, we give a status report on our attempt at a deployable prototype as well as our forecasts for marketability.**

***Index Terms*—Smart wheelchair, power wheelchair, autonomous wheelchair**

I. Executive Summary

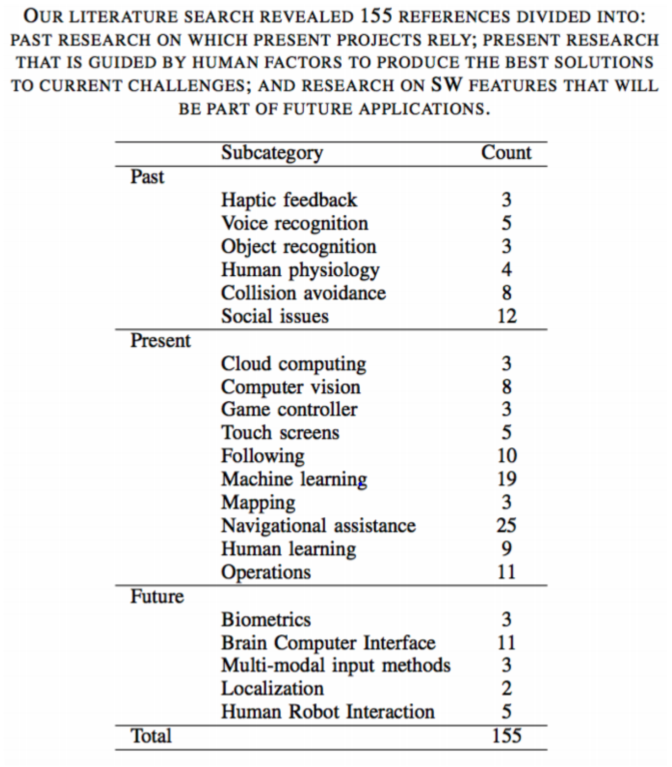
Wheelchair safety has become stagnant in terms of its technological advancements to the typical households. In other words, although there are safer wheelchairs out there, the design isn't marketed to the typical user. Our project idea proposes a smarter, safer design aimed at keeping costs affordable to the average income household. Our societal problem is such that wheelchairs are unsafe, many suffer from the same problems whether powered or manual. These problems are collision, tilting, falling, and lack of a notification to the caregiver. From the surveys we've conducted, these are what we believe to be the most desired by the wheelchair community.

This documentation guides the user into our approach as well as the cost breakdown and task breakdown to go from an idea to implementation and finally testing for deployment. This document also outlines the wiring required between each device and finally the pseudocode to get started with programming the controllers and sensors.

II. Introduction

Powered wheelchairs have expanded the range of possibilities for many that were unable to move using traditional wheelchairs. They have removed the need of another person pushing them, allowing them to live a life of new found independence. There is still however, a growing community of people with motor and sensory impairments that need additional assistance navigating through daily life.

While engineering advances, the realm of transportation has been increasing at a rapid pace, the application of these technologies aimed certain populations have been much more stagnant in terms of commercial application. The powered wheelchair has not undergone much change from when it was first revealed, though there have been some promising research advances. Jesse Leaman and Hung M. La conducted searches into these advances and produced the data in Figure 1 which outlines the possibility and future of the smart powered wheelchair [1].



Our literature search revealed 155 references divided into: past research on which present projects rely; present research that is guided by human factors to produce the best solutions to current challenges; and research on SW features that will be part of future applications.

	Subcategory	Count
Past		
	Haptic feedback	3
	Voice recognition	5
	Object recognition	3
	Human physiology	4
	Collision avoidance	8
	Social issues	12
Present		
	Cloud computing	3
	Computer vision	8
	Game controller	3
	Touch screens	5
	Following	10
	Machine learning	19
	Mapping	3
	Navigational assistance	25
	Human learning	9
	Operations	11
Future		
	Biometrics	3
	Brain Computer Interface	11
	Multi-modal input methods	3
	Localization	2
	Human Robot Interaction	5
Total		155

Fig. 1. Possibility and Future of Smart Powered Wheelchairs

III. Societal Problem

In the U.S. alone between 1.6 and 2.2 million Americans rely on wheelchairs to assist with mobility impairment on a daily basis [2]. A survey done in a disabled community shows us that wheelchair users are among the most visible in public streets., The roughly 2 million Americans utilizing manual devices and approximately 155,000 using powered wheelchairs often have functional limitations and activity restrictions. Different disability types may need different environmental supports to be successful and productive. Furthermore, wheelchair users face more barriers and obstacles in their struggle to claim back a normal lifestyle that some of us can take for granted.

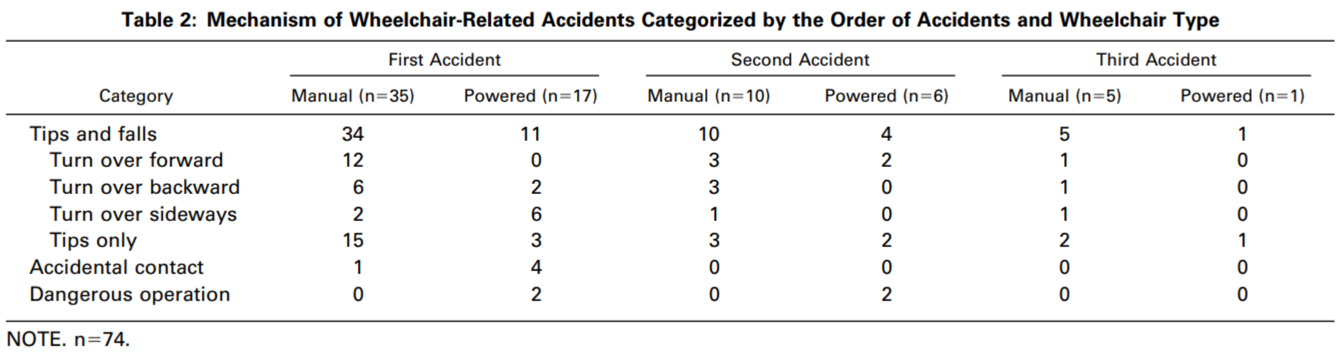


Table 2: Mechanism of Wheelchair-Related Accidents Categorized by the Order of Accidents and Wheelchair Type

	First Accident		Second Accident		Third Accident	
Category	Manual (n=35)	Powered (n=17)	Manual (n=10)	Powered (n=6)	Manual (n=5)	Powered (n=1)
Tips and falls	34	11	10	4	5	1
Turn over forward	12	0	3	2	1	0
Turn over backward	6	2	3	0	1	0
Turn over sideways	2	6	1	0	1	0
Tips only	15	3	3	2	2	1
Accidental contact	1	4	0	0	0	0
Dangerous operation	0	2	0	2	0	0

NOTE. n=74.

Fig. 2. Possibility and Future of Smart Powered Wheelchairs

Out of the approximately 2.2 million wheelchair users in America, more than 100,000 reports were made in 2003 related to injuries treated in emergency departments, doubling the number reported in 1991. The causes of wheelchair-related accidents reported by the participants are summarized in Figure 2 [4] Out of those, the most common activities reported where an incident occurred was during transfer, operations on ground level, driving up or down a hill, and reaching too far forward causing the wheelchair to lean forward with them all together. Accidents that happened during transfer of a patient were usually caused by forgetting to set the brakes or caused by a malfunction in the wheelchair; accidents caused while navigating on hilly and ground levels were caused by uneven or steep to climb surfaces or being hit by others as well as hitting other obstacles [3, 4].

Among the 95 participants, 52 participants reported at least one accident whether caused by the same fault or something else. Of the 74 accidents reported by the 52 participants, it was reported that tips and falls accounted for 87.8% accidents, accidental contact 6.8%, and 5.4% dangerous operation.

Other wheelchair related problems experienced by our target audience can range from muscular impairments causing trouble with navigation to hypo-activity causing slow reaction times. People suffering from cerebral palsy or sluggish cognitive tempo (SCT) will experience this regularly, making it nearly impossible to become independent on their own. There are alternative solutions that have been tested to help with these issues such as thought control, pupil control, or autonomy but the approach to the issue remains imperfect or impractical and expensive for the average household budget.

Different disabilities require different care, with a few requiring mobility aids for the disabled to get through their daily tasks. Take for example someone with Parkinsons, their general mobility and coordination is very limited due to their condition. To give a very rough explanation, Parkinsons is a type of movement disorder where symptoms can include: trembling of hands, arms and legs; stiffness in their muscles; slowness of movement and poor balance and coordination (or a combination of it all)[5]. As these symptoms become worse, patients may have difficulty walking, talking or completing other daily tasks and may require a caregiver. There are multiple disabilities that require a person to use a wheelchair or any mobility aid. [6] Yet these aids are still the same as once when they were invented. With todays technologies, there are many ways we can make it easier for someone suffering with these symptoms to be able to navigate without the worry of the incidents mentioned in the societal problem.

A simple wheelchair is limited to its function, even a powered wheelchair can only do so much. This is where smart technologies need to come into play. With the integration of smarter safety features, we can make these devices more effective while keeping costs low for the average household income.

IV. Design Idea Contract

It used to be the case that safety provided by proximity sensors and collision avoidance algorithms was only available in luxury vehicles, yet we have seen these innovations trickle down to more affordable cars in the past couple of years [7]. While engineering advances in the realm of transportation at high speeds, the application of these technologies to special populations have been much more stagnant in terms of commercial application.

Our solution involves introducing proximity sensors, a gyroscopic sensor, a GPS/GSM module, and the microcontroller and microcomputer to provide the safety functionality and communication to hardware. The core idea is to connect a microcontroller in between the wheelchair controls and the computer. With the microcontroller acting as a middleman, we can send/block the appropriate controls to the motor depending on the state of the wheelchair. For our design idea, we focused on making each individual feature to be modular, easy to use and seamless to negate a single feature or multiple among all features and functions. These key features include: collision avoidance and detection, tilt and step prevention, event notification and a user friendly mobile application.

A. *Feature I: Collision Detection*

The first feature on the list as a requirement to implement what we believe to be the best solution towards making wheelchairs safer is collision avoidance and collision detection. In relation to the project the collision detection feature will directly interact with collision avoidance/motor control.

This is required to send the distance data to the motor controller and the motor control can determine how hard to hit the breaks depending on the distance of the object detected. Our approach to implementing this feature into a powered wheelchair requires us to do the following:

1) Have Priority control of navigation
2) Implement Kinect sensors to detect the distance between the surrounding wheelchair and another object
3) Trigger an event call to a handler to either stop or slow down the wheelchair in time before colliding with object
4) Re-enable the controls to the motor from the joystick once the unsafe state is left

B. Feature II: Collection Avoidance

Feature two, STOP is a simple label for Collision Avoidance. The purpose of this is to test the results when an animate or inanimate object is within sight. The idea is that a minimum threshold is set to give the user a wide enough circumference for safety. In addition to this, it also provides the chair enough time to avoidance imminent danger. In this case scenario if the nearest point were to exceed this threshold an event handler should be called.

C. Feature III: Tilt Awareness

Tilt awareness detects the uneven pavement of the ground and when climbing up a steep pathway, ramps and curbs, the sensor alerts the user if he or she is in danger of falling backwards or tipping over to the side. Using the IMU we can use both the gyroscope and accelerator to determine certain slopes, which will help us map out the angle that the wheelchair. If the angle is recorded to be in an unsafe state, then we would log it as so and use those data points within a conditional statement in software to handle the event when triggered. If the wheelchair does come to an unsafe state, then an alarm will be fired off notifying the user that the wheelchair is on the edge of tilting or falling over. The level of danger will be indicated on LCD display along with an audio for the user to be cautious.

In greater detail, if the user surpasses certain threshold that have been set proprietary, the sensor will send a signal to the GUI, which will then display an alert symbol for the user. For extreme cases, if the user happens to fall or gets tipped over to the side, the sensor will send a signal to another Raspberry Pi, which will then send a SMS to a caregiver.

1) Constantly check if user is in unsafe state
2) If Wheelchair tilts over proprietary threshold for roll in either direction warn the user by displaying a notification.
3) If the wheelchair tilts over completely in X or Y direction, send notification.

D. Feature IV: Step Detection

Step detection is another feature that is for the safety of the user. This feature will determine if there is a step or steep decline in the path of the user and alert them and stop the wheelchair, using the STOP function. This feature is however a simple implementation, using only ultrasonic sensors. This feature involves being able to detect and act upon an immediate decline in the ground. This is from which ground the motors are acting upon. If a decline is seen to be far greater than allowable for the wheelchair to move forward, a middle man should be able to act upon the travel of the wheelchair. We can either warn the user and stop the wheelchair completely or we could let them proceed on their own. This should all happen though the Arduino which is already connected to the motor controls.

1) Check for Step or steep decline
2) If Step is detected alert the user and stop the wheelchair.

E. Feature V: Event Notification

Event notification is one of the most critical features to our project for a variety of reasons. The goal of our project was to make a powered wheelchair safer and situationally aware. In the unfortunate case that the user tips beyond the fall threshold, it is imperative that someone is immediately notified. Additionally, if the user is in trouble and is alone, they should be able to send an alert to someone with the press of a button. Lastly, we felt it was important to post a live stream of GPS data so that the location of the powered wheelchair user can be queried on demand.

Implementation of event notifications will require many modules working together to seamlessly notify the guardian of any accidents that may have occurred, this again can be thrown into the handler routine function definition or called as another subroutine giving the state the decision on choosing whether to send a notification or not. The different sensors and modules involve that have been accounted for are as follows: To send out the notification to a guardian from far away we would have to send out an SMS to them.

F. Feature VI: Mobile Application

The mobile application is another important feature to the project as well because the user needs to be able to use it to monitor the live GPS data that is posted by the wheelchair, and to configure different options on the wheelchair (such as the phone number to alert). The application would be used by both the powered wheelchair user and possibly the caregiver and family members as well to monitor the location of the wheelchair. This is very important when locating the user and enabling/disabling features according to the users specific needs.

The one core element that weve seen with the latest wheelchairs being manufactured today for the masses is that comfort is the key. Wheelchairs have become an essential device with many individuals seeing them as a mobile assistant, and to the many extreme cases theyve become the main source of mobility. Our project tackles the problem of safety and concern for all wheelchair users, whether they have cerebral palsy, Parkinsons disease or have gotten into an accident. This means putting in place features such as collision avoidance, collision detection, step detection, tilt awareness and event notification integrated in an app. Keeping this in mind also

TABLE I
MATERIAL COSTS

Date	Item	Cost
9/16/2017	FONA 808 GPS GSM module	$50
9/16/2017	IoT Sim card	$0.00
9/18/2017	2x 12v 35Ah Batteries	$114.12
9/18/2017	Dual 25A Motor Driver	$125.00
2/5/2018	BNO055	$35.00
9/30/2017	2x Kinect v1	$25.00
11/21/2017	28dB GPS Antenna (High power)	$14.07
11/21/2017	SMA Female to uFL cable (antennas)	$6.99
11/21/2017	12V to 5V Dual	$7.99
TOTAL		**$378.17**

means keeping costs reasonable and parts readily available to the open market. Viable solutions are always readily available today, but no realistic solution considers the costs and how it affects its users. With the safety of our users being our biggest priority, the implementation of remote navigation or extreme comfort becomes secondary and not a functional requirement.

V. Funding and Budget

Our team did not receive any additional funding from outside resources making us stick to the goal of keeping things affordable. Fortunately, a neighbor was kind enough to give us theirs. The only issue was the powered wheelchair needed new batteries. The batteries, rated 12V 35Ah, set us back $114 roughly, in other words we got a working powered wheelchair for $114 and a few hours of our time to test and go through any repairs. Considering the cost of old and used, this saved us hundreds. Considering the cost of new, this saves us thousands.

With the center piece to our project now in our hands the only other materials needed were a motor controller to overhaul the current controller, the microcontroller to communicate with it, the various sensors as required for our safety features, and the micro computer to control it all. Parts for each of the features, and the cost breakdown can be found in Table 1.

Though, major savings for this project came from already owned equipment such as the microcontroller, wires, and low cost ultrasonics; the overall cost of the project taking everything into account whether purchased or pre-owned still came down to about $378 as mentioned in the table.

VI. Project Milestones

With consumer level products made to market, comes extra work involved. In other words, you work for what you don't pay for. Example, some powered wheelchairs utilize the CAN-Bus protocol allowing us to easily take control of the motors without having needing to program. Taking a more cost effective approach meant needing to program it, this became the first major feat our team overcame. A list of all the major features can be seen below:

1) Taking control of the motors
2) Integrating a Joystick
3) Event Notification Through GUI and SMS
4) Hardware Integration between all features
5) Wheelchair running on startup with all features
6) Software Stabilization
7) Remote configuration between App and GUI
8) Mitigation Steps

VII. Work Breakdown Structure

Though the concept of our features are simple to understand, the implementation as always is the challenge. This section illustrates the work breakdown structure (WBS) as seen in Figure 3. The WBS guides our team to how each user will be responsible for their feature as well as how to approach their feature from researching, implementing, testing, and deploying.

A. Task Assignments

1) Collision Detection:

a) Detect Object: With the Microsoft Kinect as our sensor of choice for collision detection, the next step was to actually attempt to connect the device to the connect. Ultimately, this meant downloading libfreenect, an open source driver for the Kinect, run an example code and begin the learning process.

We learn from example, and from examples we can expand on it with our own ideas. This was the biggest task to Collision Detection, and every feature in fact. Because we're not modifying any registers or overwriting the current firmware the programming language of choice was Python. Though, if the reader is so eagerly inclined the libfreenect libraries are also available in C, Java, and ROS (not a programming language but a development environment for robots).

At a high level to test the base functionality and determine if this truly is the route the project will be taking we needed to be able to do one important thing, start a depth stream. Libfreenect supports this.

Since libfreenect can support reading the depth stream from the Kinect we can read those raw values, the Kinect has a 640x480 resolution. It can also detect objects up to 20 feet away, but realistically we will care for objects about 5 feet in front of us.

Not a simple task as we'll have to traverse the 2-D array of rows and columns and determine what's an object or not. The approach to this in theory for our project will be to simple say everything is a square and if that square object is close enough to use, we call our collision avoidance handler.

Responsible: Nurthin Aziz

b) Tie into Collision Avoidance: With a working theory in place testing needs to be done to tackle any issues not accounted for. In order to test we'll need to first integrate this into collision avoidance. Collision avoidance and collision detection are on two separate controllers due to the capabilities of the Kinect. The Kinect will be talking from a Raspberry Pi to the Arduino using I2C.

I2C is a serial protocol that allows up to communicate between devices, the downfall to this is that because its serial it only supports half-duplex. Half-duplex, means it can only do one at a time, meaning reading and writing.

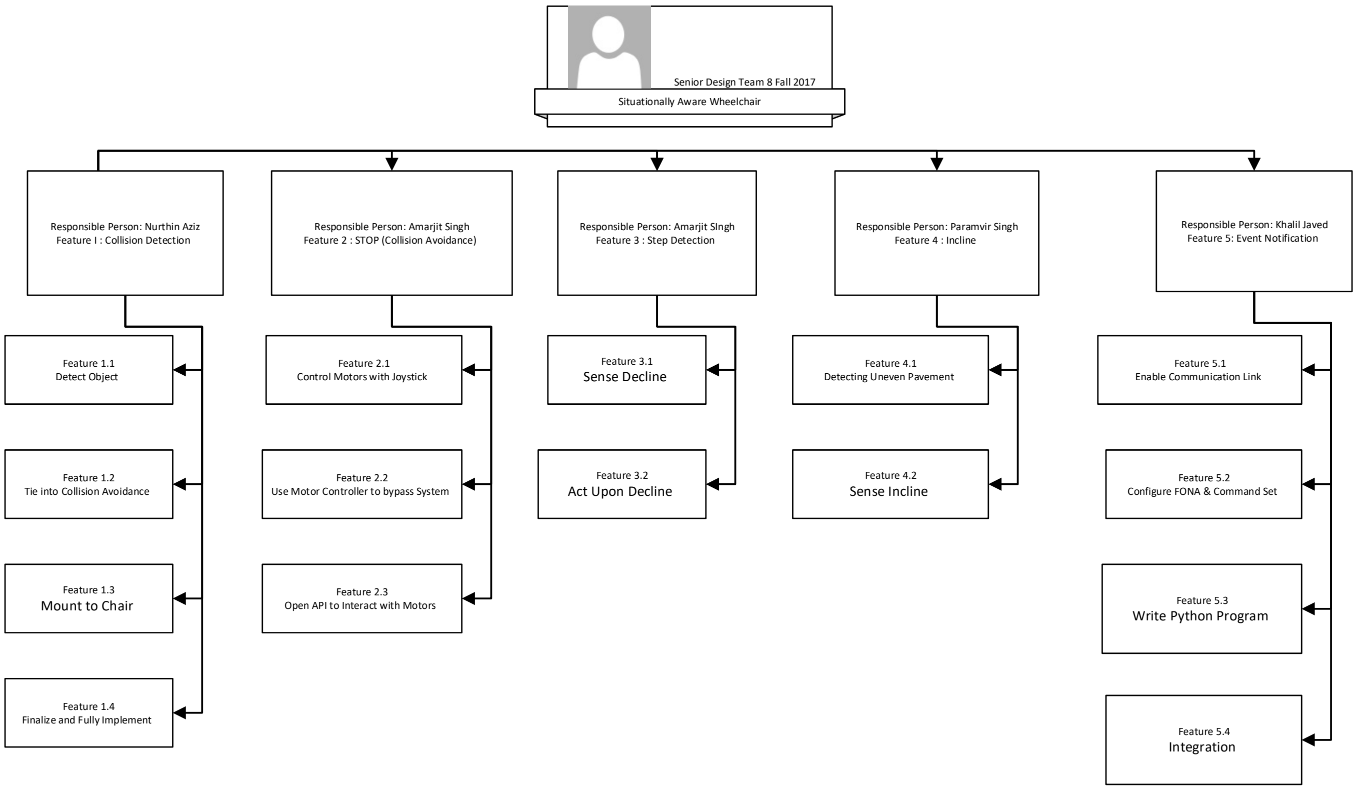

Senior Design Team 8 Fall 2017
Situationally Aware Wheelchair
Responsible Person: Nurthin Aziz
Feature I : Collision Detection
Responsible Person: Amarjit Singh
Feature 2 : STOP (Collision Avoidance)
Responsible Person: Amarjit SIngh
Feature 3 : Step Detection
Responsible Person: Paramvir Singh
Feature 4 : Incline
Responsible Person: Khalil Javed
Feature 5: Event Notification
Feature 1.1
Detect Object
Feature 1.2
Tie into Collision Avoidance
Feature 1.3
Mount to Chair
Feature 1.4
Finalize and Fully Implement
Feature 2.1
Control Motors with Joystick
Feature 2.2
Use Motor Controller to bypass System
Feature 2.3
Open API to Interact with Motors
Feature 3.1
Sense Decline
Feature 3.2
Act Upon Decline
Feature 4.1
Detecting Uneven Pavement
Feature 4.2
Sense Incline
Feature 5.1
Enable Communication Link
Feature 5.2
Configure FONA & Command Set
Feature 5.3
Write Python Program
Feature 5.4
Integration

Fig. 3. High Level Work Breakdown Structure

Responsible: Nurthin Aziz

c) Mount To Chair: Nothing special to the mounting, though the spot of choice had to be overhead, unless and armrest was to be implemented.

Responsible: Nurthin Aziz

d) Finalize and Integrate: This task is an ongoing task set for the remainder of our time to prototype and deploy. The tasks required for this involved everything was testing, implementing, testing, and more implementing. A continuous cycle until satisfied with the reliability of the product.

Responsible: Nurthin Aziz

2) Collision Avoidance:

a) Control Motors with Joystick: To be able to stop or avoid an obstacle we need to be able to control the motors. Bypassing the main system and using our own motoro controller will alow such functionality. This controller should allow interfacing using a microcontroller. Which in turn will allow us to run the motors using code.

The work will be straight forward but time consuming. Intial work is to get the motors to do any movement using a simple code then work on directions. A sabretooth 2x25A v2 mtoro controller will be used to accomplish this. This controller is well documented and easy to find resources on. Arduino microcontroller will be used to maker the motors move.

Responsible: Amarjit Singh

b) Use Motor Controller to bypass other systems: Most important implementation in all this project is getting the motors to run. If the wheelchair isn't moving the user is disabled whether medically or not. This feature should and must work first. Using a motor controller that can directly function with the motors and thus allowing us to control the wheelchair with our own joystick.

Responsible: Amarjit Singh

c) Allow other features to interact with motors: This function will give other features the ability to control aspects of the motors to their needs. For example, when the collision detection feature sends an alert to Arduino that its detected an object, the motors should stop.

The code written needs to be open to allow other features to interact with the motors. The other feature must send a signal that can then be converted to something understandable by the Arduino.

Responsible: Amarjit Singh

3) Step Detection:

a) Sense Decline: Implementing a sensor that can calculate the distance between the bottom of the wheelchair and the ground. This data should open at all times and communicating the with motor controllers. The sensors used are ultrasonics and does a transmit and receive to ping the distance of an object in front of it. The distance is calculated based on the round trip time taken.

Responsible: Amarjit Singh

b) Act On Decline:: If a decline is seen to be far greater than allowable for the wheelchair to move forward, a middle man should be able to act on the navigation of the wheelchair. Since the Arduino will also house the step detection, communication was seamless.

Responsible: Amarjit Singh

4) Tilt Detection (Incline):

a) Detection Uneven Pavement: Can detect any sense of danger in the pavement and able to send the differences between surfaces in real time to notify the user.

To get the best data result, this sensor will be in the center of the chair to take data points for the wheelchair to be in unstable angle. This will ensure that whenever there is a tilt on the wheelchair, the sensor will be able to find the distance between ground and the wheelchairs wheels.

Responsible: Paramvir Singh

b) Sensing the incline: Once on an incline street or pathway, the sensor will pick up if it is safe to move forward or send signal to the microcontroller for being in an unsafe state.

Responsible: Paramvir Singh

5) Event Notification:

a) Enable Communication Link: The goal of this step is to ensure the Raspberry Pi 3 and FONA 808 can successfully send/receive data to each other. This involved soldering pins and sending data over a serial to USB connector.

This feature begins by researching the required materials for the Raspberry Pi 3 and FONA 808 to perform their tasks. There are battery packs, GSM antennas, GPS antennas, and USB to serial cables that need to be ordered; however, there are many optiosn for these parts and the goal is to determine the equipment that's cost effective and performs the best.

Responsible: Khalil Javed

b) Configure FONA & Learn Command Set: The goal of this step is to get the SIM card registered and working with the FONA. Following this we will learn about the command set the FONA uses to communicate.

Responsible: Khalil Javed

c) Writing Python Program: This step builds the foundation for our feature by creating a Python scrip that sends a text message to a pre-programmed phone number. This will involve the AT commands learned in the previous task to set up a serial connection with the FONA. Following this we can then begin to parse GPS data and play with SMS messaging.

Responsible: Khalil Javed

d) Integration: Finally, the integration of event notification as it stands with all other features, namely tilt awareness. Because the two features are tied together they must have a communication link of their own. From their we can determine the best place, with little noise, and determine the power source required when integrating into the wheelchair.

The rest of this task carries on to the end of the project until satisfied with the results.

Responsible: Khalil Javed

VIII. Risk Assessment and Mitigation

A. Risk One - Detection Objects

When dealing with safety features the impact of a risk is high therefore mitigation plans need to be put in place to ensure the likelihood of that event is small.

This risk begins with collision detection. The biggest challenge to collision detection has been the simplest concept to understand. Detect an object and calculate the distance between it and you. Concept is simple but against implementation is the challenge.

The likelihood of this risk as it stands now is likely, and the risk can be major. The reason for its likelihood is due to the complexity of needing to traverse the depths stream when read from the connect. A simple for-loop to traverse the rows and columns cause too much overhead.

The mitigation plan for this risk will be to set the threshold between the object and the wheelchair user. Furthermore, the ability to turn on and off the event notification during times of unreliability will be a must or simply just wanting to freely roam or push boxes around.

B. Risk Two - Loss of Power to Motors

This risk doesn't involve the batteries dying the the user not capable of moving around, it should always be the responsibility of the user to charge the batteries.

This risk is involved with a disconnection to the batteries. The likelihood that this risk can happen is high, a major risk that must be handled with importance. A way to handle this is to make sure that all connections are secure and reliable. Test the motors before installation, protect the motor controller and have it covered if possible.

C. Risk Three - Step Detection Sensors Blocked

In the case of step detection, if any of the sensors are blocked due to a rock or a small object we can quickly imagine the wheelchair coming to an abrupt stop wondering why. The likelihood of this risk is likely, but the impact is very minor. No mitigation plan is necessary, as the user can just backup.

D. Risk Four - Angle Calibration For Tilt Awareness

Needing to calibrate the gyroscope was a big issue we needed to overcome. The likelihood was likely and the impact or severity can be serious. There was a simple solution to this, get a better gyroscope. The MPU6050 was a cheap solution at first to test the waters, and now with our feet wet we needed an upgrade to the BNO055.

E. Risk Five - Notification Delay

This risk can occur if the GPS antennas have not been intialized correctly, so there needs to be safe measured put in place. Programmatically, we plan to implement these safe measures whic include 2 GPS query retires followed by a toggling of the GPS module and another set of retries. If the system is still unable to grab the coordinates, the caregiver will still be notified of the event.

IX. Design Overview

A wheelchair that is aware of its situation. This statement is what drove us into creating a system in which we can use various sensors to detect and react to an event that has occurred. The use of cost effective sensors was a big objective of ours, as it would allow us to produce a product cheaper than many alternatives available in the market with similar features. The answer to why would one want a situationally aware wheelchair is where our societal problem comes in play. There are Powered Wheelchair (PW) users that are older or have conditions such as cerebral palsy or multiple sclerosis which affect reaction time and motor control. Keeping this in mind, features from our system would be welcomed in the wheelchair community as there is a strong need for this functionality. The powered wheelchair has not undergone much change since it was revealed decades ago. To overcome this stagnant progression in PW safety, our design idea focused on collision avoidance and detection, step detection, tilt awareness, GPS tracking, SMS for event notifications, and a mobile application to personalize the wheelchair to the user.

Having a smart system on the wheelchair could help these user in their daily routine. As most users suffering from the conditions mentioned need a caregiver most of the day, this make it hard for these users to be able to travel alone leaving them less inclined to enjoy life outside. Our system features would work together to act in the situation where the caregiver normally would, but for reasons X, Y, or Z can't.

Our design was derived from the need in the current PW and what it was lacking in standalone safety features. We looked at the areas in which the PW was found in an accident and from these issues we sought out technologies that could be used to avoid or mitigate the risks involved.

Alternatively, there are other ways people have been tackling this issue, but fall short. For example, during our research we saw products such as exoskeletons and bionic helper limbs. These products are meant to bring full autonomous navigation or full support to joints and limbs to the user but fall short due to the limited production and cost per unit. Therefore, our team has picked a design and sought out to create a product that would be an attachment to the current users powered wheelchair rather than needing to make another purchase.

So, you might be asking, what are these features and what do they do? Let us begin with the key feature on which every other feature is dependent, motor controls. Wait doesn't the PW already have motor controls, isn't that how it works? Yeah, the PW does have motor controls but do the motor controls that are already implemented work with our feature sets, nope. That is why for our design the team had to figure out how we would tap into the motors so that we can imply our own controls on the entire system. By making use of a programmable motor controller we could use the wheelchair motors and have them operate as we would prefer.

1) *Control wheelchair movement by external joystick.*
2) *Start and Stop wheelchair by will of other features.*
3) *Maneuver the wheelchair separate from user joystick input.*

Now with the understanding of the change in the motor controller is mind let us focus briefly on the main features. Starting with collision avoidance and collision detection, this feature is designed as the eye for the visual situation around the wheelchair. By taking the advance sensor sets of the Xbox 360 Kinect, we can cut out costs and receive data in accordance to depth in the field of view of the wheelchair. This depth data lets us map out the moves and events that can be triggered is necessary at a given threshold. Additionally, to the eye of the wheelchair we have step and curb detection, as the Kinect is only able to map out the area in the foreground of the wheelchair we need something that can look below the wheelchair. Using ultrasonic sensor, as we only need small data sets and short distances, we mount these on the bottom and sides of the wheelchair. From the bottom ultrasonic we map out the data directly beneath the wheelchair, whereas from the sides it is from curb detection.

Now that we have a control of the environment in the form of eyes, though the Kinect and the ultrasonic. We know need to have a positioning understanding of the wheelchair, this could be in the form of GPS and in the form of orientation/tilt. A tilt sensor implementation, a sensor that knows the orientation and the angle on which the wheelchair is operation at. Our implementation is using a BNO055, this is a small compact sensor that is more powerful than we need in our application. An absolute orientation sensor, this can keep its orientation even when not powered so calibration for this, on a basic level, is not needed. Additional to the orientation system we have a module that can determine the current location of the wheelchair using a GPS system. The FONA module is a cellular connective module that has on board GPS service, this feature can be used to record the location of the wheelchair and send its coordinates when a distress signal is sent. The event notification system alerts the user of a triggered event on the LCD screen of the wheelchair and if such an event where to occur the system will use a FONA sensor to send a text message, with recorded location from the FONA onboard GPS, to a registered caregiver phone number.

Finally, to complete the needs of a smart wheelchair we have the integration of a Mobile application platform and a LCD with a user-friendly GUI on the wheelchair. With the on-display GUI, from which the user can see a speed needle, see their current speed setting, see a battery level, and most importantly change the settings. This GUI lets the user interact with which feature should be on and active or which feature should be off as not needed. The screen also has a SOS button if in any case the PW user needs to alert their caregiver in any circumstance. The mobile application can be used to set caregiver phone number, turn on or off settings as on the GUI, and most importantly be able to queried the location of the wheelchair when it is out and about thus allowing a caregiver to check in and find out where the PW user is.

X. Deployable Prototype Status

Our device is in a state that is super close to be able to be deployed yet there are a few things that need work. As this device is running from opensource microcontroller and microcomputer that shuns us away from having a device that can be launched with. Although these devices are meant for rapid prototyping and if we can easily offload out code ontop of a close knit system then we are ready to deploy. All feature sets are in correct functional order and have been tested to be working to their fullest, a few edge cases might be reaming to knock out. Additionally the look of the system must be considered before deployment, we can't have wires all over the place. So before a actually deployment the wires must be routed and the devices such as the raspberry pi and the arduino must be put into a closed housing so that they look clean and non obtrusive.

A. Current Status

With Senior Design coming to a close, we need to assess how much of our product is working according to expectation, and how well we stuck to our problem statement. As stated before the main goal of our design was to make the current PW safer than what is already was. By adding sensors and modules that would function around the wheelchair we could monitor and react to possible collision events as needed.

Revisiting the features we have: Collision Detection and Avoidance, Step and Curb detection, Tilt awareness, Event notification and a Mobile application. These features have been discussed and they work as expected yet, as nothing is perfect we do have a few issues when it comes down to the Collision Detection system. The avoidance feature is simple and smooth, as it only needed a trigger event to make its action prominent. As for the detection system it uses the Kinect module to scan the frame and give back an alert if necessary this is where the current issue lies.

This feature is complex, it requires taking in the entire field of view of the Kinect, scanning the entire frame and determining if an object is too close for safety. As of now we are receiving the depth data and are able to use the frame for reference, but the issues are the edge cases in which the depths of some objects skews the entire depth being sent to the motor controls causing misfire. The mitigation plan for this is still in the works as a simple for-loop to scan everything will require too much overhead, causing a delay in what is read by the Arduino and what is seen by the Kinect. This is still being worked on and will hopefully be mitigated, additionally backup sensors such that of an ultra-sonics can be used to test against for added data points.

The remaining features are working as indicated and they have minimal to no edge cases or issues that can cause fault in their current functionality. The tilt sensor is mounted on a center point of the wheelchair to obtain a better and more reliable degree of tilt. The ultrasonic are mounted in the front, back, and sides. Furthermore, the GUI is fully functional and intractable by the user, the alert events show up on the screen as they are triggered, and audio alert system works as expected.

B. Marketability Forecast

Marketing is important when we are developing a product or application for a group of people. We are not doing this as a fun project for around the house use, in developing a system to solve a societal problem we must understand the market and how we can provide the product to our customers. In our case as we are dealing with PW users who could use the added safety measure to their current system, we must design a plug and play system that the user can easily and readily use.

As of now the system is in a state in which it can not be brought to market due to some minor prototyping concerns. These concerns are simply how the system is built up on a set of a microcontroller and microcomputer, Arduino and Raspberry Pi respectively. These systems are good to allow us to rapidly prototype our desing idea but must surely be changed in allowing real world deployment. This is where an embedded system comes into play, although we have our system designed with a raspberry pi in mind when we deploy we must move the entire codebase over to an embedded system.

With an embedded system we have full rights to our product and the parts on it, such that we can take full ownership of the design. Additionally, the embedded system will have no additional parts or functions that are not needed to the design specification, as we see with the Raspbian software. This will make boot times faster and thus make the entire system enclosed and easy to use and install. But most importantly this will knock out the remaining factor that is truly halting our design to be launched and marketed.

XI. Conclusion

A wheelchair that is safer is a wheelchair that is trusted, for decades the wheelchair has been a simple and straightforward device. It helps the disabled, those that need it, get around with ease. The powered wheelchair came along and made getting around even easier but introduced potential dangers. In our design we challenged these faults prone to a powered wheelchair. The design of our safer wheelchair gives the user Collision detection warnings and actions, it alerts and stop them from going over a curb and most importantly send their caregiver a SMS notification with the GPS coordinates if a dangerous fault has occurred. Using multiple sensors, we can determine faults and take actions upon them. By adding a microcontroller, we setup states in which the wheelchair will act if a certain data is detected. Thus, the wheelchair can now successfully stop from a fall or collision and send an alert message if the user does endure any harm. By adding graphical user interface, we can alert the user of any reason of the stop or show them of potential dangers.

The wheelchair is much safer than it would have been on its own. Using these sensors, we have made the user of the wheelchair feel safer when going out and experiencing the world by themselves, not having to worry about faults from the power wheelchair.

XII. References

[1] Jesse Leaman, and Hung Manh La with a comprehensive review and criteria for what would should be considered in the future of Smart Wheelchair innovations and technology. In A Comprehensive Review of Smart Wheelchairs: Past Present, and Future, pages

[2] ”2012 Disability Status Report: United States.” Disabilitystatistics.org, Disabilitystatistics, 2012, www.disabilitystatistics.org/reports/2012/English/HTML/report 2012.cfm?fips=2000000&html_year$\bar{2}$012&subButton$\bar{G}$et %2BHTML.

[3] H XIANG-AM CHANY-G SMITH, WHEELCHAIR RELATED INJURIES TREATED IN US EMERGENCY DEPARTMENTS, 2006 FEB - HTTPS://WWW.NCBI.NLM.NIH.GOV/PMC/ARTICLES /PMC2563507/

[4] W. CHEN, Y. JANG, J. WANG AND Y. WANG, ”WHEELCHAIR-RELATED ACCIDENTS: RELATIONSHIP WITH WHEELCHAIR- SING BEHAVIOR IN ACTIVE COMMUNITY WHEELCHAIR USERS”, PUBMED, P. 1, 2011.

[5] ”Parkinson’s Disease — PD — Medline-Plus”, Medlineplus.gov, 2018. [Online]. Available: https://medlineplus.gov/parkinsonsdisease.html.

[6] 2012 Disability Status Report: United States. Disabilitystatistics.org, Disabilitystatistics, 2012, www.disabilitystatistics.org/reports/2012/English/HTML/report 2012.cfm?fips=2000000&html_year$\bar{2}$012&subButton$\bar{G}$et %2BHTML.

[7] I. Fletcher, B. J. B. Arden and C. S. Cox, ”Automatic braking system control,” Proceedings of the 2003 IEEE International Symposium on Intelligent Control, Houston, TX, USA, 2003, pp. 411-414.

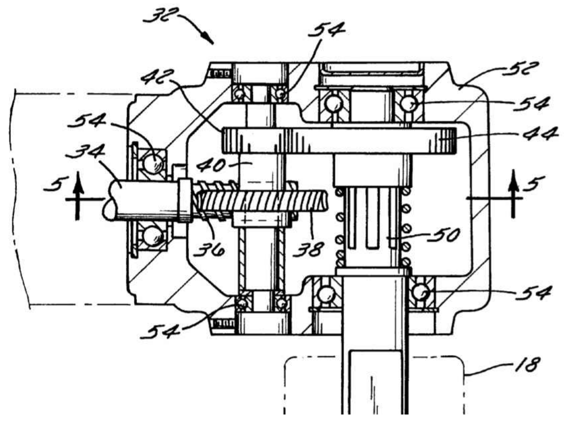


Fig. 4. Inner Workings of the Motor that will be used [7]

XIII. Glossary

Appendix A
User Manual

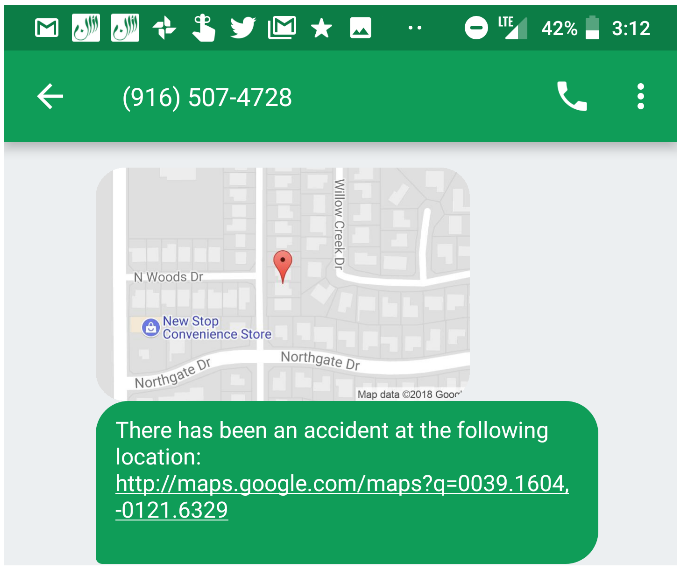
When considering the additional features we implemented in our safer powered wheelchair, it was imperative to make the entire system easy to learn and operate. Although the joystick mechanism has been replaced, its function is similar to any standard powered wheelchair. The additional features, however, will be unfamiliar to new users. In this user guide, we will be going over the touchscreen layout and the mobile application, using both to highlight all features of our safer, smarter wheelchair.

A. Wheelchair

1) Power Button: The wheelchair itself functions similarly to a standard powered wheelchair in that it has a manual power toggle located near the right armrest. This is used to turn the wheelchair on or off. All smart functionality will turn on with this button and it will take up to 30 seconds for a complete boot up. During this time, the wheelchair is operational but smart functionality will not be available until the user interface is displayed on the screen.

2) Joystick Operation: The joystick also operates similarly to a standard powered wheelchair. Moving the joystick forwards and backwards will ramp up acceleration and deceleration respectively. Moving the joystick right and left will spin the wheelchair right or left. The joystick is multidirectional and can operate between the front, back, left, and right zones.

3) Automatic Braking: When using the powered wheelchair, you may experience automatic braking in certain scenarios in the default configuration. For example, when the user approaches an obstacle, the wheelchair will slow down to prevent a collision. Additionally, when the user approaches a curb or a step, the wheelchair will also halt the



There has been an accident at the following location: http://maps.google.com/maps?q=0039.1604,-0121.6329

Fig. 5. Emergency SMS sent with longitude and latitude

motors before continuing. This behavior is expected and is used to prevent accidents.

4) Automatic Alerts to Caregiver: Alerts to a caregiver or any phone number configured are sent in the case of an accident. The message is an SMS with an Google Maps embedded link which smartphones will automatically convert to a geographical location as seen in figure 5. Additionally, this message can also be sent on demand if the user hits the "help" button on the touchscreen gui.

B. Touchscreen Control Panel

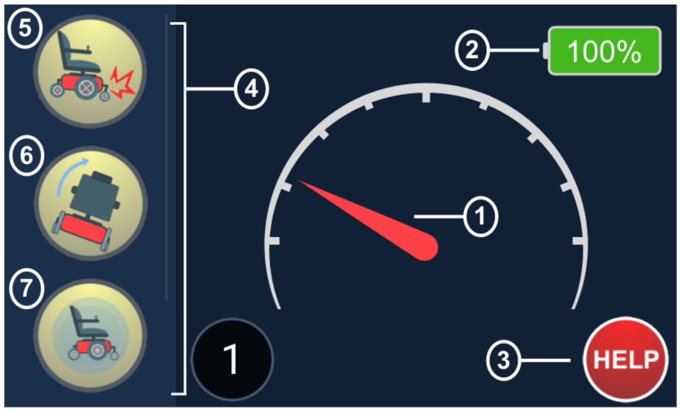


Fig. 6. Touchscreen GUI on Powered Wheelchair

After the wheelchair boots up, you will be presented with the control panel as seen in the top screenshot of figure 6. See the key below for the items highlighted in the bottom screenshot of figure 6:

1) Speedometer
2) Battery Indicator
3) Emergency Button
4) Sidebar
5) Collision Detection
6) Tilt Detection
7) GPS Tracking
8) Step Detection (off screen)

1) Indicators: The **Speedometer** shows the users speed relative to the maximum output speed of the wheelchair. The **Battery Indicator** shows the battery level of the powered wheelchair and broken up into 5 states: 100%, 75%, 50%, 25%, and 10%. The wheelchair MUST be charged if under 10% as the motors will slow or stop soon afterwards.

2) Toggling Features: The **Sidebar** shows the entire feature set for the powered wheelchair. The icons top to bottom are Collision Detection, Tilt Detection, GPS Tracking, and Step Detection. If the icon is bright, the feature is on, and if the icon is dark, the feature is off. Tapping the icon turns the selected feature on or off. If the selected feature is turned off, the feature will be completely disabled and the user will not receive any audio or video alert.

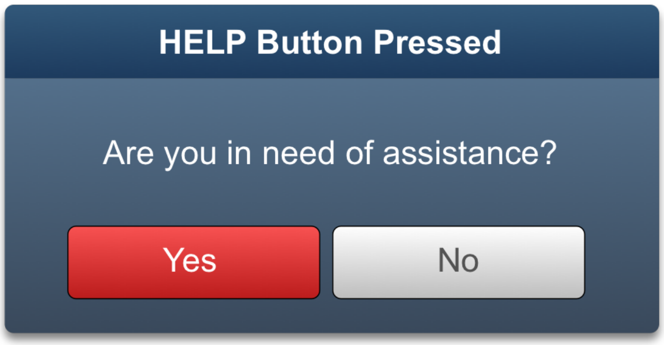


Fig. 7. Prompt message when emergency button is pressed

Fig. 8. Message shown upon delivery of SMS

3) Using the Emergency Button: The **Emergency Button** located on the right should be used in case of an emergency. When the button is pressed, the screen shown in figure 7 is displayed, and prompts the user if he or she would like to send an SMS to the emergency phone number configured. The user will be notified when the SMS is sent as seen in figure 8. See the user guide section for the Mobile Application to learn how to configure an emergency phone number.

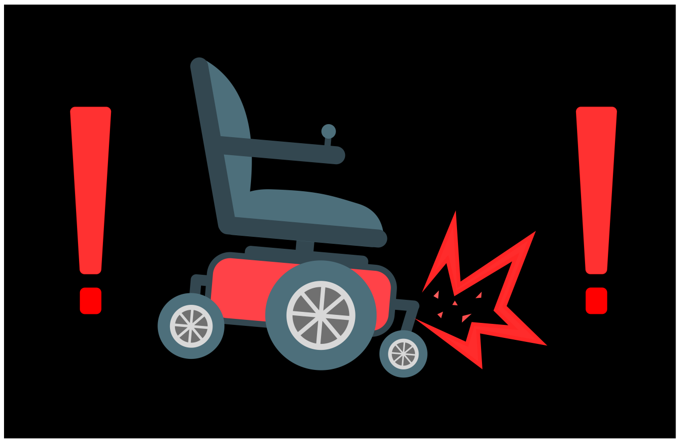


Fig. 9. Collision detection alert screen

4) Sounds & Alerts: This powered wheelchair is designed to send the user visual and audio alerts in specific circumstances. These include: when a potential collision is detected, when the user is about to drive over a curb or step, and when the user is at risk of tipping over due to being at an unsafe tilt angle. In these scenarios, the touchscreen interface will flash with an alert such as the one seen in figure 9. The alert is accompanied by a brief audio message notifying the user of the event that occurred, or the action that was taken by the wheelchair.

C. Mobile Application

Fig. 10. Mobile Application Icon

Along with the touchscreen the wheelchair user, or a caregiver can remotely configure the powered wheelchair via an Android application. This allows greater customization and a user interface design that is similar to the touchscreen on the wheelchair itself. All changes made on the application are immediately updated on the wheelchairs control panel and the wheelchair will play a contextual audio message when a setting is updated.

1) Configuring the Wheelchair: The Smart Wheelchair mobile application begins with a splash screen after which the user is on the Configure Wheelchair page by default. Upon pressing the button to connect, the application will take the user to their settings where they will connect to the wheelchair network and return to the application.

2) Connecting to the Network: Our powered wheelchair emits a secure wireless network that is used to connect to the wheelchair and apply any configuration changes. The network is available as soon as the control panel is loaded after booting. The WiFi SSID is in the format SmartWheelchair-X with X denoting the unique identifier of the wheelchair. The default password for the wheelchair is smart123. Upon connecting to the network and returning to the application, the user can now press the button to connect to their powered wheelchair and will be presented with the options seen in figure 11 upon successful connection.

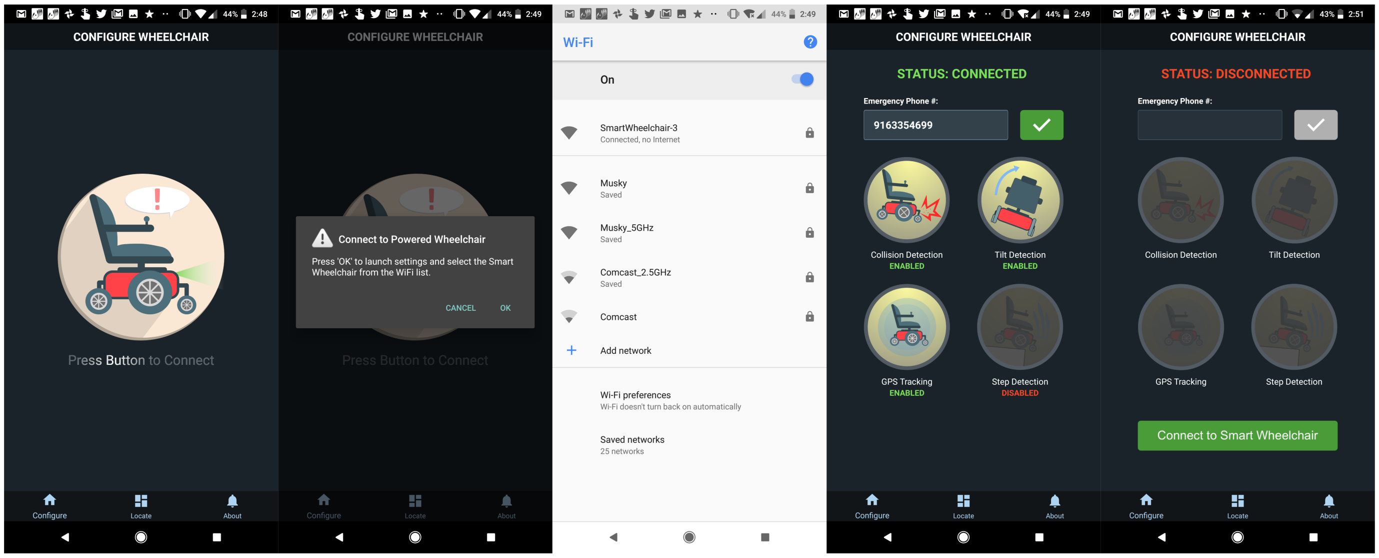
3) Connection Status Indicator: The Connection Status Indicator is shown at the top of the screen, showing green when connected, and red when disconnected. If the connection is interrupted, the user can press the green Connect button that appears at the bottom of the screen to restart the connection process.

4) Enabling & Disabling Features: After connecting, the status of each feature of the wheelchair will be displayed as well as the phone number that is configured to receive alerts in case of an accident or emergency. Enabling and disabling features works identically as it does on the wheelchairs touchscreen control panel. Each of the icons is labeled with the feature it controls and is also labeled as Enabled or Disabled. Pressing any one of the buttons will immediately update the wheelchair and play an audio message after the configuration is applied. The icon will also be updated in the mobile application to reflect the change.

5) Updating Emergency Phone Number: Below the **Connection Status Indicator** is the **Emergency Phone Number Entry** field. Tapping in this field allows the application user to change the configured phone number and this setting is sent to the wheelchair when the green check button is pressed. Similarly to the enabling and disabling of features, an audio message plays on the wheelchair when the configuration is successfully applied.

6) Locating the Wheelchair:

7) Disconnect from the Wheelchair Network: If you are still connected to the powered wheelchair, the message shown in figure 12 will appear, notifying you to disconnect from the wheelchair. Upon disconnecting from the wheelchair, the internet connection is restored to the phone and the application will query the last reported location of the wheelchair. This step is mandatory to track the wheelchair.

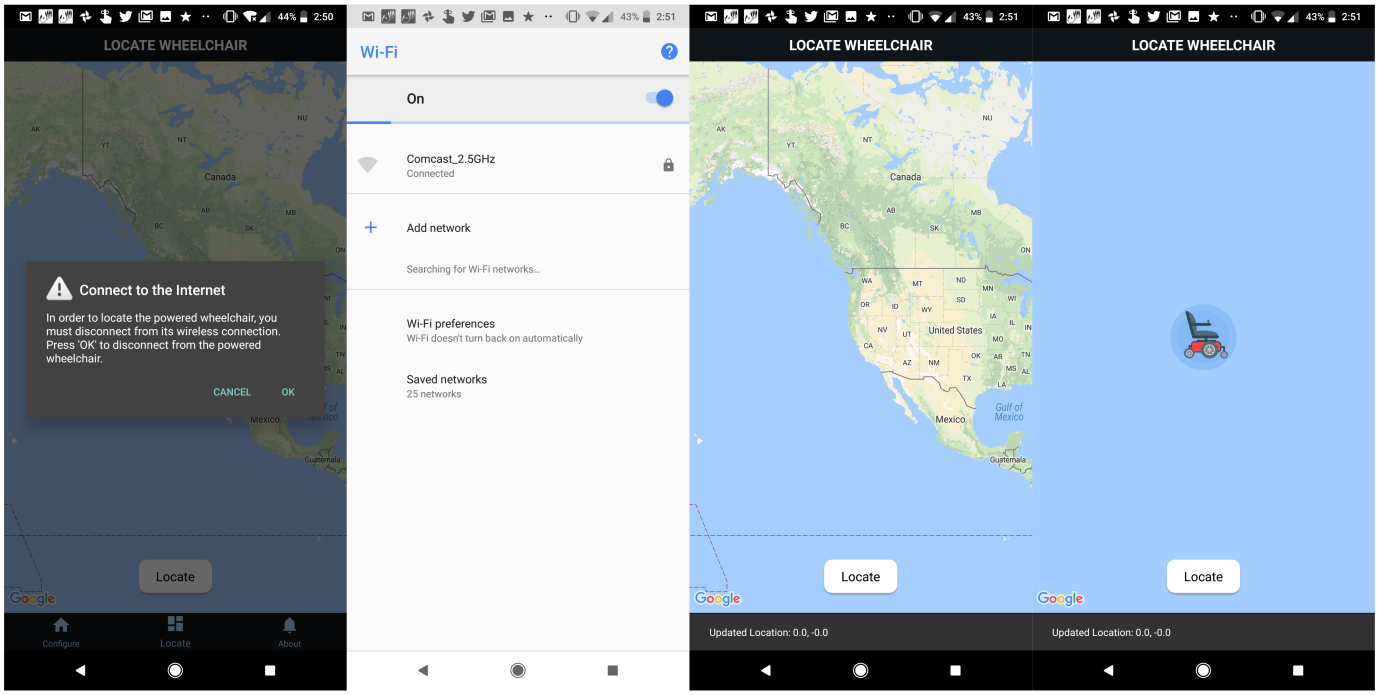


Connect to Powered Wheelchair

Press 'OK' to launch settings and select the Smart Wheelchair from the WiFi list.

CANCEL OK

Fig. 11. Screens shown when configuring the wheelchair



Connect to the Internet

In order to locate the powered wheelchair, you must disconnect from its wireless connection. Press 'OK' to disconnect from the powered wheelchair.

CANCEL OK

Updated Location: 0.0, -0.0

Fig. 12. Screens shown when locating the wheelchair

8) Locate the Wheelchair: Upon disconnecting from the wheelchair, the application will immediately begin querying the last reported location of the wheelchair, notifying you when the location is updated and displaying the gps dot as shown in figure 12. To center in on the location of the wheelchair, you must press the Locate button at the bottom of the map. This zooms in the current view on the last reported location of the wheelchair.

D. Troubleshooting

1) I Cannot Locate the Wheelchair: If you cannot locate the powered wheelchair from the mobile application, try the following:

1) Disconnect from the Powered Wheelchair Network
2) Reboot the Powered Wheelchair
3) Ensure the GPS Tracking Feature is On
4) Wait 2 Minutes for GPS Signal Lock
5) Relaunch Mobile Application

2) The Location is Not Updating:

1) Ensure the GPS Module is Connected via USB
2) Ensure the Wheelchair Battery is Not Below 10%
3) Ensure the GPS Tracking Feature is On
4) Motor Controller

The goal of our team was to implement our features and their control in a way that is easy to use. The configured features simply work without the user having to think about them when turning the wheelchair on or off. The large toggle buttons and audio queues make it easy for all populations to understand the controls. The rich graphics make our product seamlessly fit in to the modern smartphone applications that the general population is accustomed to. In the next section, we will cover the hardware that makes all of this possible.

Appendix B
Hardware

See figure 13 for the system wiring diagram. This diagram illustrates the proper connections made in accordance to the way the software is designed as well as well as how the devices should interact with each other.

Appendix C
Software

With the complexity of software, we give you a high level understanding of how the devices are communicating with each other and what the data being sent is used for as well as what the data is. An illustration of this can be found in figure 14.

The pseudocode can be found in figures 15 to through 59.

Appendix D
Mechanical

All mechanical drawings can be found in figures 60 to 65. The boxes were 3-d printed where necessary, therefore these are the sketches.

Appendix E
Vendor Contacts

N/A

Appendix F
Resumes

See end of document for resumes.

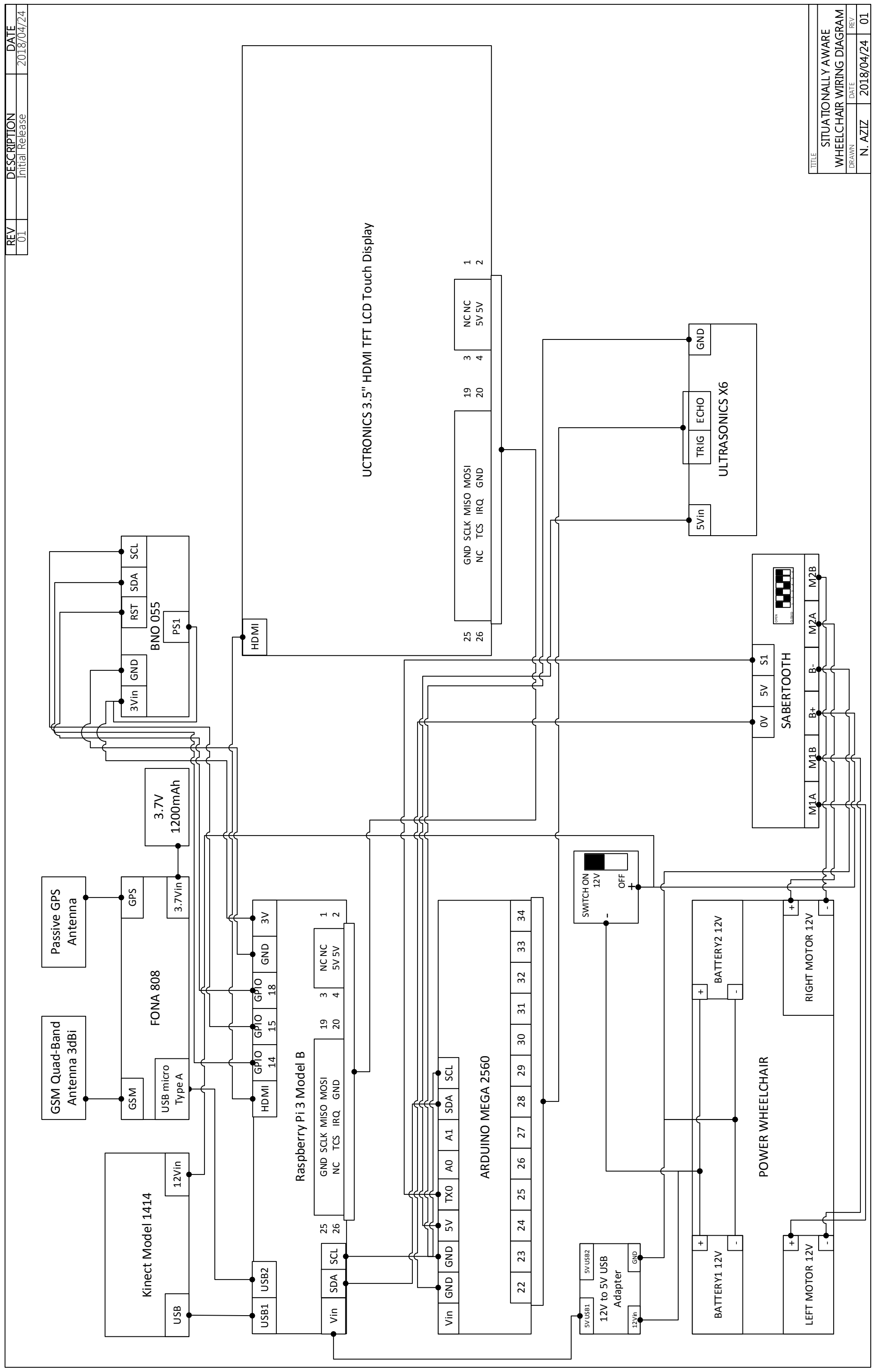


Fig. 13. System Wiring Diagram

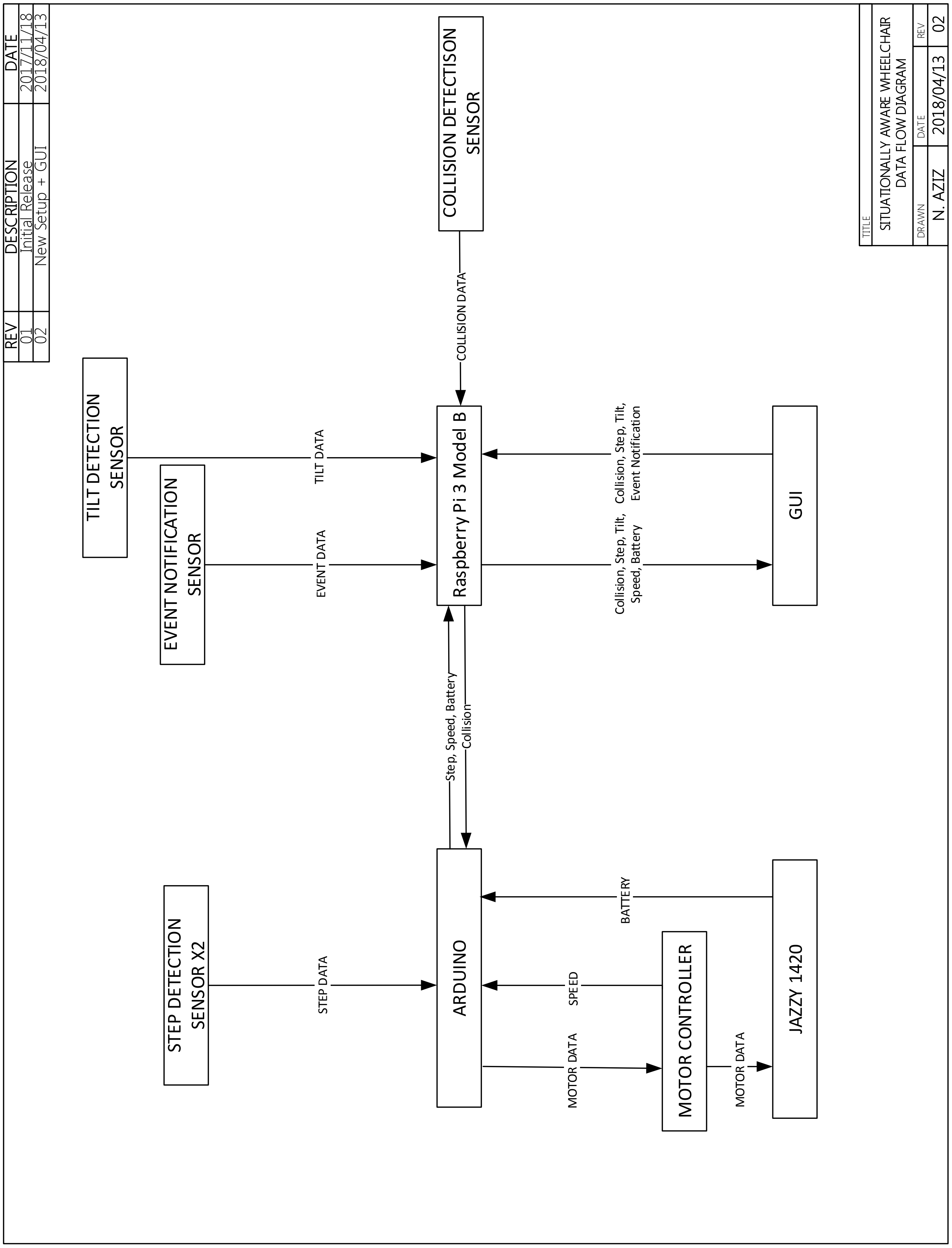


Fig. 14. System Software Diagram

```
#!/usr/bin/env/python
# -*- coding: utf-8-*-

import freenect
import cv2
import numpy as np
import smbus2 as smbus
import subprocess
from threading import Thread
from collections import deque
from time import sleep, gmtime, strftime

# Arduino Connection
bus = smbus.SMBus(1)
address = 0x04

# Used for storing the Kinect filepath
camerapath = ''
motorpath = ''
audiopath = ''

def send_depth(depth):
    for i in depth:
        bus.write_byte(address, i)
    return -1

# Calls lsusb to get the filepath for usb_reset(), output is:
# # Bus 001 Device 014: ID 045e:02ae Microsoft Corp. Xbox NUI Camera
# The devfs path to these devices is:
# # /dev/bus/usb/<busnum>/<devnum>
# So for the above device, it would be:
# # /dev/bus/usb/001/014


def get_kinect():
    proc = subprocess.Popen(['lsusb'], stdout=subprocess.PIPE)
    out = proc.communicate()[0]
    lines = out.split('\n')
    for line in lines:
        if 'Xbox NUI Camera' in line:
            parts = line.split()
            bus = parts[1]
            dev = parts[3][:3]
            global camerapath
            camerapath = '/dev/bus/usb/%s/%s' % (bus, dev)
        if 'Xbox NUI Motor' in line:
            parts = line.split()
            bus = parts[1]
            dev = parts[3][:3]
            global motorpath
            motorpath = '/dev/bus/usb/%s/%s' % (bus, dev)
        if 'Xbox NUI Audio' in line:
            parts = line.split()
            bus = parts[1]
            dev = parts[3][:3]
            global audiopath
            audiopath = '/dev/bus/usb/%s/%s' % (bus, dev)


# https://gist.github.com/x2q/5124616
# Calls ./usbreset on camerapath, motorpath, and audiopath
def usb_reset():
    get_kinect()
    global camerapath
```

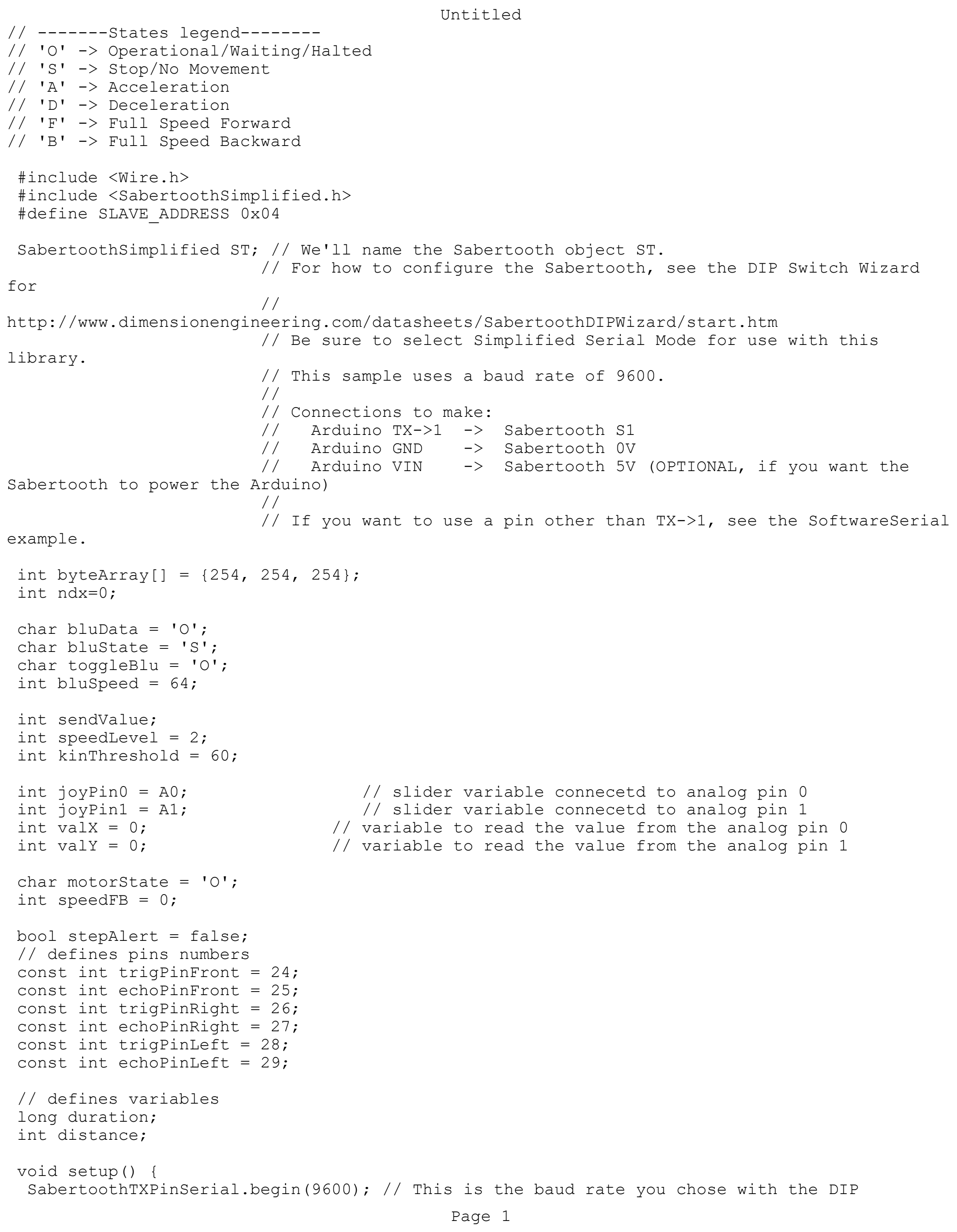
Fig. 15. Pseudocode For Collision Detection Page 1

```
    global motorpath
    global audiopath

    process1 = subprocess.Popen(
        ['./usbreset', camerapath], stdout=subprocess.PIPE)
    stdout1 = process1.communicate()
    process2 = subprocess.Popen(
        ['./usbreset', motorpath], stdout=subprocess.PIPE)
    stdout2 = process2.communicate()
    process3 = subprocess.Popen(
        ['./usbreset', audiopath], stdout=subprocess.PIPE)
    stdout3 = process3.communicate()

    print camerapath
    print motorpath
    print audiopath
    print stdout1
    print stdout2
    print stdout3
    sleep(3)


def greater_avg(deptharray):
    print(deptharray)
    deptharray_np = np.array(deptharray)
    deptharray_np = np.average(deptharray_np)
    return deptharray_np

class KinectDepth(Thread):
    # Prototype the daemon service
    def __init__(self, *largs, **kwargs):
        super(KinectDepth, self).__init__(*largs, **kwargs)
        self.daemon = True
        self.queue = deque()
        self.quit = False
        self.index = 0

    # Our daemon service (Kinect)
    def run(self):
        q = self.queue
        while not self.quit:
            # Open the File To Read if Collision Detection Feature On/Off
            collisionfile = open(
                "/home/pi/Desktop/everyone/vals/collision.txt", "r")
            powerOn = collisionfile.readline(1)
            collisionfile.close()   # Done Reading, close file

            # Grab the depth array from Kinect
            array = freenect.sync_get_depth(
                index=self.index, format=freenect.DEPTH_REGISTERED)
            # If None then either:
            # # 1) Kinect Failed to connect (funny).
            # # 2) Kinect Device has been claimed. (Possibly a hard kill? Ctrl-C)
            if array is None:
                print("Resetting usb, wait two seconds..")
                usb_reset()
                sleep(2)
                continue
            else:
                depthsArr, timestamp = array

                # Define Regions of Interest Out of: 480 x 640
                # roiright = int((np.average(depths[:,:213])/1000) * 36)
                # roimid = int((np.average(depths[:, 213:426])/1000) * 36)
```

Fig. 16. Pseudocode For Collision Detection Page 2

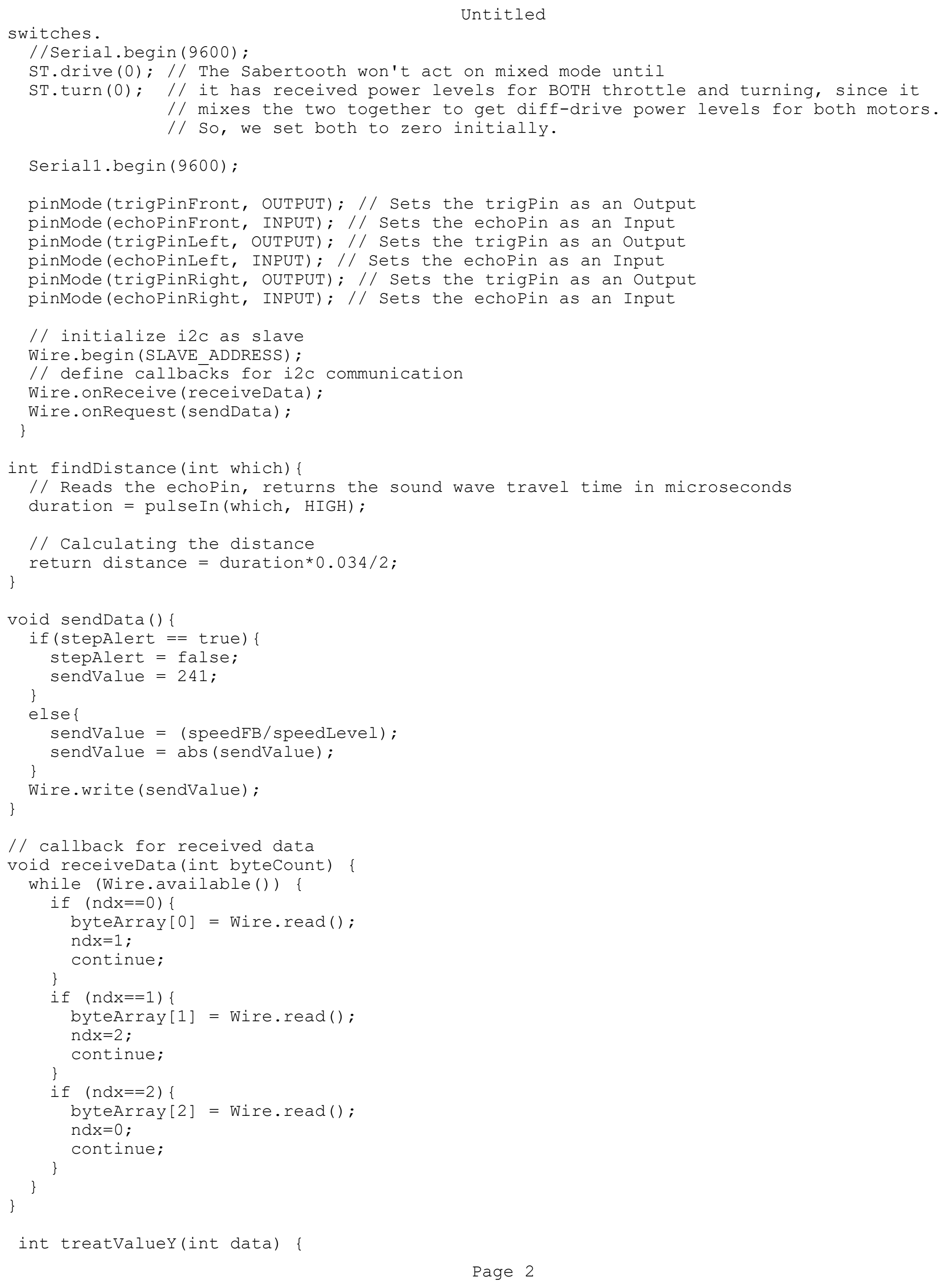
```
                # roileft = int((np.average(depths[:,426:640])/1000) * 36)
                # depths = [roileft, roimid, roiright]

                # Create a mask with our threshold value
                threshold = 60
                mask = depthsArr > threshold

                # Slice the regions into seperate 2d arrays
                roiright = depthsArr[:,:213]<60
                roimid = depthsArr[:,:213:426]<60
                roileft = depthsArr[:,:426:640]<60

                # Build a list of indices with the mask and check length
                objOnRight = 50 if len(roiright[mask]) > 0 else 255
                objInMid = 50 if len(roimid[mask]) > 0 else 255
                objOnLeft = 50 if len(roileft[mask]) > 0 else 255

                depths = [objOnRight, objInMid, objOnLeft]
                # IF ON: Send Depth Data to Motor Controllers
                if powerOn == '1':
                    if objOnRight or objInMid or objOnLeft:
                        collisionafile = open(
                            "/home/pi/Desktop/everyone/vals/collisiona.txt", "w")
                        collisionafile.write("1")
                        collisionafile.close()
                    else:
                        collisionafile = open(
                            "/home/pi/Desktop/everyone/vals/collisiona.txt", "w")
                        collisionafile.write("0")
                        collisionafile.close()
                    send_depth(depths)

                    # Define Current Time
                    currenttime = strftime("%Y-%m-%d %H:%M:%S", gmtime())
                    print "{0}       AT: {1}".format(depths, currenttime)

                # IF OFF: Send Const Data to Motor Controller
                else:
                    print "Feature Off {0}".format(powerOn)
                    # 253 is used and RESERVED when the feature is off.
                    send_depth([253, 253, 253])
                continue
            q.appendleft(array)

    def pop(self):
        return self.dequeue.pop()


if __name__ == "__main__":
    while (1):
        KinectDepth().run()
```

Fig. 17. Pseudocode For Collision Detection Page 3



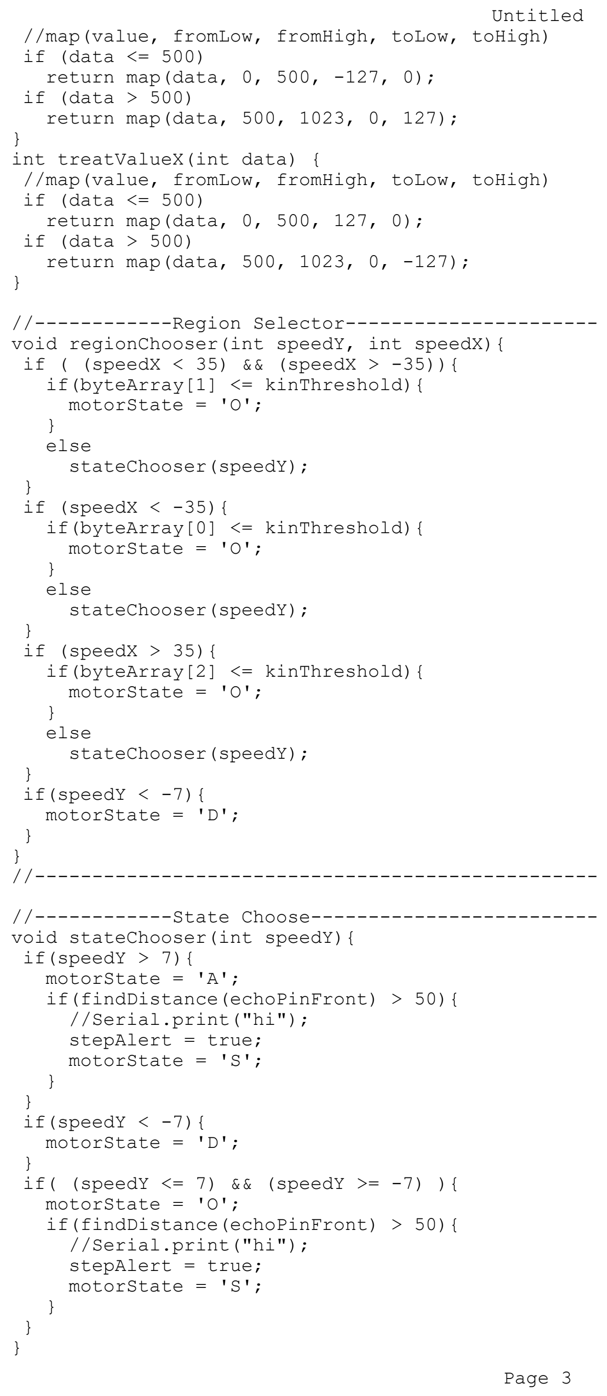
```
                                            Untitled
// -------States legend--------
// 'O' -> Operational/Waiting/Halted
// 'S' -> Stop/No Movement
// 'A' -> Acceleration
// 'D' -> Deceleration
// 'F' -> Full Speed Forward
// 'B' -> Full Speed Backward

 #include <Wire.h>
 #include <SabertoothSimplified.h>
 #define SLAVE_ADDRESS 0x04

 SabertoothSimplified ST; // We'll name the Sabertooth object ST.
                         // For how to configure the Sabertooth, see the DIP Switch Wizard
for
                         //
http://www.dimensionengineering.com/datasheets/SabertoothDIPWizard/start.htm
                         // Be sure to select Simplified Serial Mode for use with this
library.
                         // This sample uses a baud rate of 9600.
                         //
                         // Connections to make:
                         //   Arduino TX->1  ->  Sabertooth S1
                         //   Arduino GND    ->  Sabertooth 0V
                         //   Arduino VIN    ->  Sabertooth 5V (OPTIONAL, if you want the
Sabertooth to power the Arduino)
                         //
                         // If you want to use a pin other than TX->1, see the SoftwareSerial
example.

 int byteArray[] = {254, 254, 254};
 int ndx=0;

 char bluData = 'O';
 char bluState = 'S';
 char toggleBlu = 'O';
 int bluSpeed = 64;

 int sendValue;
 int speedLevel = 2;
 int kinThreshold = 60;

 int joyPin0 = A0;                 // slider variable connecetd to analog pin 0
 int joyPin1 = A1;                 // slider variable connecetd to analog pin 1
 int valX = 0;                  // variable to read the value from the analog pin 0
 int valY = 0;                  // variable to read the value from the analog pin 1

 char motorState = 'O';
 int speedFB = 0;

 bool stepAlert = false;
 // defines pins numbers
 const int trigPinFront = 24;
 const int echoPinFront = 25;
 const int trigPinRight = 26;
 const int echoPinRight = 27;
 const int trigPinLeft = 28;
 const int echoPinLeft = 29;

 // defines variables
 long duration;
 int distance;

 void setup() {
  SabertoothTXPinSerial.begin(9600); // This is the baud rate you chose with the DIP
                                             Page 1
```

Fig. 18. Pseudocode For Motor Controls Page 1



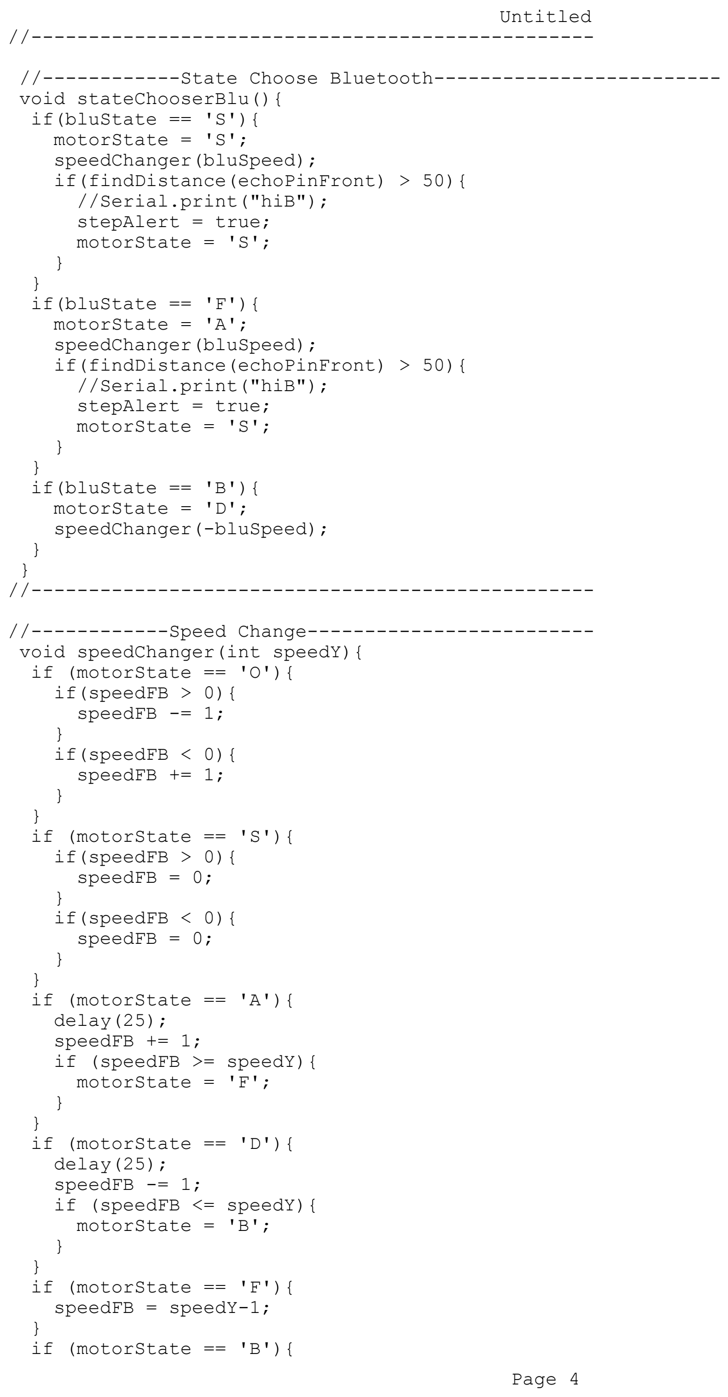
```
                                        Untitled
switches.
  //Serial.begin(9600);
  ST.drive(0); // The Sabertooth won't act on mixed mode until
  ST.turn(0);  // it has received power levels for BOTH throttle and turning, since it
               // mixes the two together to get diff-drive power levels for both motors.
               // So, we set both to zero initially.

  Serial1.begin(9600);

  pinMode(trigPinFront, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPinFront, INPUT); // Sets the echoPin as an Input
  pinMode(trigPinLeft, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPinLeft, INPUT); // Sets the echoPin as an Input
  pinMode(trigPinRight, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPinRight, INPUT); // Sets the echoPin as an Input

  // initialize i2c as slave
  Wire.begin(SLAVE_ADDRESS);
  // define callbacks for i2c communication
  Wire.onReceive(receiveData);
  Wire.onRequest(sendData);
 }

int findDistance(int which){
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(which, HIGH);

  // Calculating the distance
  return distance = duration*0.034/2;
}

void sendData(){
  if(stepAlert == true){
    stepAlert = false;
    sendValue = 241;
  }
  else{
    sendValue = (speedFB/speedLevel);
    sendValue = abs(sendValue);
  }
  Wire.write(sendValue);
}

// callback for received data
void receiveData(int byteCount) {
  while (Wire.available()) {
    if (ndx==0){
      byteArray[0] = Wire.read();
      ndx=1;
      continue;
    }
    if (ndx==1){
      byteArray[1] = Wire.read();
      ndx=2;
      continue;
    }
    if (ndx==2){
      byteArray[2] = Wire.read();
      ndx=0;
      continue;
    }
  }
}

 int treatValueY(int data) {
                                         Page 2
```

Fig. 19. Pseudocode For Motor Controls Page 2



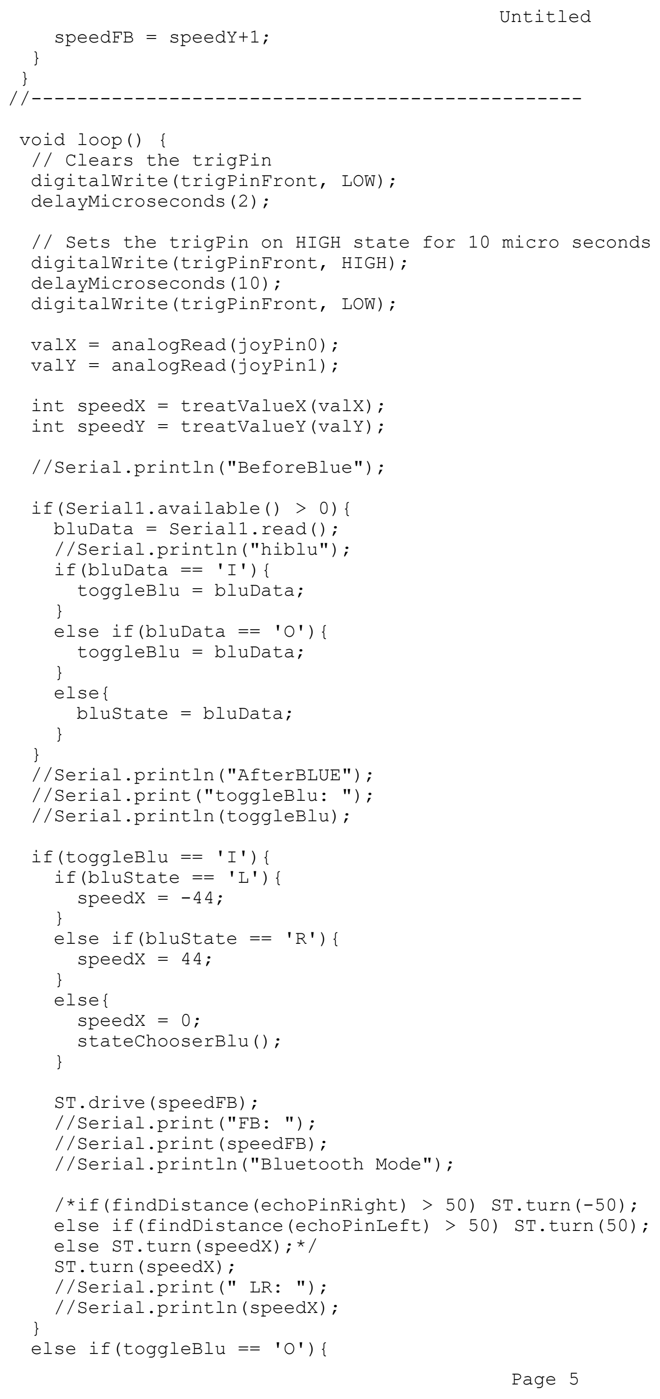
```
                                              Untitled
 //map(value, fromLow, fromHigh, toLow, toHigh)
 if (data <= 500)
   return map(data, 0, 500, -127, 0);
 if (data > 500)
   return map(data, 500, 1023, 0, 127);
}
int treatValueX(int data) {
 //map(value, fromLow, fromHigh, toLow, toHigh)
 if (data <= 500)
   return map(data, 0, 500, 127, 0);
 if (data > 500)
   return map(data, 500, 1023, 0, -127);
}

//------------Region Selector----------------------
void regionChooser(int speedY, int speedX){
 if ( (speedX < 35) && (speedX > -35)){
   if(byteArray[1] <= kinThreshold){
     motorState = 'O';
   }
   else
     stateChooser(speedY);
 }
 if (speedX < -35){
   if(byteArray[0] <= kinThreshold){
     motorState = 'O';
   }
   else
     stateChooser(speedY);
 }
 if (speedX > 35){
   if(byteArray[2] <= kinThreshold){
     motorState = 'O';
   }
   else
     stateChooser(speedY);
 }
 if(speedY < -7){
   motorState = 'D';
 }
}
//-----------------------------------------------

//------------State Choose-------------------------
void stateChooser(int speedY){
 if(speedY > 7){
   motorState = 'A';
   if(findDistance(echoPinFront) > 50){
     //Serial.print("hi");
     stepAlert = true;
     motorState = 'S';
   }
 }
 if(speedY < -7){
   motorState = 'D';
 }
 if( (speedY <= 7) && (speedY >= -7) ){
   motorState = 'O';
   if(findDistance(echoPinFront) > 50){
     //Serial.print("hi");
     stepAlert = true;
     motorState = 'S';
   }
 }
}
                                               Page 3
```

Fig. 20. Pseudocode For Motor Controls Page 3



```
                                          Untitled
//--------------------------------------------------

 //------------State Choose Bluetooth-------------------------
 void stateChooserBlu(){
  if(bluState == 'S'){
    motorState = 'S';
    speedChanger(bluSpeed);
    if(findDistance(echoPinFront) > 50){
      //Serial.print("hiB");
      stepAlert = true;
      motorState = 'S';
    }
  }
  if(bluState == 'F'){
    motorState = 'A';
    speedChanger(bluSpeed);
    if(findDistance(echoPinFront) > 50){
      //Serial.print("hiB");
      stepAlert = true;
      motorState = 'S';
    }
  }
  if(bluState == 'B'){
    motorState = 'D';
    speedChanger(-bluSpeed);
  }
 }
//--------------------------------------------------

//------------Speed Change-------------------------
 void speedChanger(int speedY){
  if (motorState == 'O'){
    if(speedFB > 0){
      speedFB -= 1;
    }
    if(speedFB < 0){
      speedFB += 1;
    }
  }
  if (motorState == 'S'){
    if(speedFB > 0){
      speedFB = 0;
    }
    if(speedFB < 0){
      speedFB = 0;
    }
  }
  if (motorState == 'A'){
    delay(25);
    speedFB += 1;
    if (speedFB >= speedY){
      motorState = 'F';
    }
  }
  if (motorState == 'D'){
    delay(25);
    speedFB -= 1;
    if (speedFB <= speedY){
      motorState = 'B';
    }
  }
  if (motorState == 'F'){
    speedFB = speedY-1;
  }
  if (motorState == 'B'){
                                           Page 4
```

Fig. 21. Pseudocode For Motor Controls Page 4



```
                                          Untitled
    speedFB = speedY+1;
  }
 }
//-----------------------------------------------

 void loop() {
  // Clears the trigPin
  digitalWrite(trigPinFront, LOW);
  delayMicroseconds(2);

  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPinFront, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPinFront, LOW);

  valX = analogRead(joyPin0);
  valY = analogRead(joyPin1);

  int speedX = treatValueX(valX);
  int speedY = treatValueY(valY);

  //Serial.println("BeforeBlue");

  if(Serial1.available() > 0){
    bluData = Serial1.read();
    //Serial.println("hiblu");
    if(bluData == 'I'){
      toggleBlu = bluData;
    }
    else if(bluData == 'O'){
      toggleBlu = bluData;
    }
    else{
      bluState = bluData;
    }
  }
  //Serial.println("AfterBLUE");
  //Serial.print("toggleBlu: ");
  //Serial.println(toggleBlu);

  if(toggleBlu == 'I'){
    if(bluState == 'L'){
      speedX = -44;
    }
    else if(bluState == 'R'){
      speedX = 44;
    }
    else{
      speedX = 0;
      stateChooserBlu();
    }

    ST.drive(speedFB);
    //Serial.print("FB: ");
    //Serial.print(speedFB);
    //Serial.println("Bluetooth Mode");

    /*if(findDistance(echoPinRight) > 50) ST.turn(-50);
    else if(findDistance(echoPinLeft) > 50) ST.turn(50);
    else ST.turn(speedX);*/
    ST.turn(speedX);
    //Serial.print(" LR: ");
    //Serial.println(speedX);
  }
  else if(toggleBlu == 'O'){
                                           Page 5
```

Fig. 22. Pseudocode For Motor Controls Page 5

```
                                          Untitled
    regionChooser(speedY, speedX);
    speedChanger(speedY);

    ST.drive(speedFB/speedLevel);
    //Serial.print("FB: ");
    //Serial.print(speedFB/speedLevel);
    //Serial.println("Joystick Mode");

    /*if(findDistance(echoPinRight) > 50) ST.turn(-50);
    else if(findDistance(echoPinLeft) > 50) ST.turn(50);
    else ST.turn(speedX/3);*/
    ST.turn(speedX/3);
    //Serial.print(" LR: ");
    //Serial.println(speedX/3);
  }
}
```

Page 6

Fig. 23. Pseudocode For Motor Controls Page 6

```
# Simple Adafruit BNO055 sensor reading example.  Will print the orientation
# and calibration data every second.
#
# Copyright (c) 2015 Adafruit Industries
# Author: Tony DiCola
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
# THE SOFTWARE.

import time
import subprocess

def main():
      import logging
      import sys
      import time

      from Adafruit_BNO055 import BNO055


      # Create and configure the BNO sensor connection.  Make sure only ONE of the
      # below 'bno = ...' lines is uncommented:
      # Raspberry Pi configuration with serial UART and RST connected to GPIO 18:

      bno = BNO055.BNO055(serial_port='/dev/serial0', rst=18)


      # Enable verbose debug logging if -v is passed as a parameter.
      if len(sys.argv) == 2 and sys.argv[1].lower() == '-v':
            logging.basicConfig(level=logging.DEBUG)

      # Initialize the BNO055 and stop if something went wrong.
      if not bno.begin():
            raise RuntimeError('Failed to initialize BNO055! Is the sensor connected?')

      # Print system status and self test result.
      status, self_test, error = bno.get_system_status()
      print('System status: {0}'.format(status))
      print('Self test result (0x0F is normal): 0x{0:02X}'.format(self_test))
      # Print out an error if system status is in error mode.
      if status == 0x01:
            print('System error: {0}'.format(error))
            print('See datasheet section 4.3.59 for the meaning.')

      # Print BNO055 software revision and other diagnostic data.
      sw, bl, accel, mag, gyro = bno.get_revision()
      print('Software version:   {0}'.format(sw))
      print('Bootloader version: {0}'.format(bl))
      print('Accelerometer ID:   0x{0:02X}'.format(accel))
```

Fig. 24. Pseudocode For Tilt Page 1

```
        print('Magnetometer ID:    0x{0:02X}'.format(mag))
        print('Gyroscope ID:       0x{0:02X}\n'.format(gyro))

        print('Reading BNO055 data, press Ctrl-C to quit...')

        sent = False
        while True:
              # Read the Euler angles for heading, roll, pitch (all in degrees).
              heading, roll, pitch = bno.read_euler()
              # Read the calibration status, 0=uncalibrated and 3=fully calibrated.
              sys, gyro, accel, mag = bno.get_calibration_status()
              # Print everything out.
              print('Heading={0:0.2F} Roll={1:0.2F} Pitch={2:0.2F}\tSys_cal={3} Gyro_cal={4}
Accel_cal={5} Mag_cal={6}'.format(
                heading, roll, pitch, sys, gyro, accel, mag))

              if roll > 50 or roll < -50 or pitch > 50 or pitch < -50:
                    if(not sent):
                        print('Sending SMS to FONA')
                        crashfile = open("/home/pi/Desktop/everyone/vals/crash.txt", "w")
                        crashfile.write("1")
                        crashfile.close()
                         sent = True
              elif roll > 5: # or roll < -25:
                    tiltfile = open("/home/pi/Desktop/everyone/vals/tiltR.txt", "w")
                    tiltfile.write("2")
                    tiltfile.close()
                    print('is sending right tilt alert to GUI')
                 elif roll < -5:
                    tiltfile = open("/home/pi/Desktop/everyone/vals/tiltR.txt", "w")
                    tiltfile.write("1")
                    tiltfile.close()
                    print('is sending left tilt alert to GUI')

              else:
                    print("here")
                        tiltfile = open("/home/pi/Desktop/everyone/vals/tiltR.txt", "w")
                    tiltfile.write("0")
                    tiltfile.close()
                    sent = False # Reset the sent variable after wheelchair is upright
                    print('you good')
              ###########################################################################
              ##SMS FONA###
              ##########################################################################


              #x,y,z = bno.read_linear_acceleration()

              # Sleep for a second until the next reading.
              time.sleep(.5)
def start():
    while(True):
        main()
start()

def getVals():
    tiltState = open("/home/pi/Desktop/everyone/vals.tilt.txt", "r")
    tiltVal = tiltState.readLine(1)
    tiltState.close()

# def tiltAlertFunc():
#      global tiltState
#      global tiltAlert
#
```

Fig. 25. Pseudocode For Tilt Page 2

```
#       if tiltState == 0:
#         tiltAlert = True
#         print("tiltState 0")
#     if ((tiltState == 1) & (tiltAlert == True)):

#if __name__ == "__main__":
#    start()
```

Fig. 26. Pseudocode For Tilt Page 3

```
==============================================================================
 MOBILE APPLICATION CODE (ANDROID)
==============================================================================


================================================
================================================
===========      SPLASHSCREEN      ============
================================================
================================================
================================================


package khalil.smartwheelchair;

import android.app.Activity;
import android.content.Intent;
import android.graphics.PixelFormat;
import android.media.Image;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.Window;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

public class Splashscreen extends Activity {
    public Button splash_button;

    public void onAttachedToWindow() {
        super.onAttachedToWindow();
        Window window = getWindow();
        window.setFormat(PixelFormat.RGBA_8888);
    }
    /** Called when the activity is first created. */

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splashscreen);
        StartAnimations();
        splash_button = (Button) findViewById(R.id.splash_button);
        splash_button.setBackgroundResource(R.drawable.selector);
        splash_button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                final Intent intent = new Intent(Splashscreen.this, MainActivity.class);
                startActivity(intent);
            }
        });
    }

    private void StartAnimations() {
        Animation fade_in = AnimationUtils.loadAnimation(this, R.anim.alpha);

        ImageView iv = (ImageView) findViewById(R.id.splash_text);
        iv.startAnimation(fade_in);
    }
}
```

Fig. 27. Pseudocode For Mobile App & Event Notification Page 1

```
================================================
================================================
=======   CUSTOM PAGER FOR FRAGMENTS   =========
================================================
================================================
================================================


package khalil.smartwheelchair;

import android.content.Context;
import android.support.v4.view.ViewPager;
import android.util.AttributeSet;
import android.view.MotionEvent;


public class CustomViewPager extends ViewPager {
        public CustomViewPager(Context context, AttributeSet attrs) {
            super(context, attrs);
        }

        @Override
        public boolean onTouchEvent(MotionEvent event) {
            return false;
        }

        @Override
        public boolean onInterceptTouchEvent(MotionEvent event) {
            return false;
        }
}

================================================
================================================
======     PAGER ADAPTER FOR FRAGMENTS     =====
================================================
================================================
================================================

package khalil.smartwheelchair;

import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentStatePagerAdapter;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by khalil on 12/22/17.
 */

public class SectionsStatePagerAdapter extends FragmentStatePagerAdapter {


    private final List<Fragment> mFragmentList = new ArrayList<>();
    private final List<String> mFragmentTitleList = new ArrayList<>();

    public SectionsStatePagerAdapter(FragmentManager fm) {
        super(fm);
    }

    public void addFragment(Fragment fragment, String title){
        mFragmentList.add(fragment);
        mFragmentTitleList.add(title);
    }

    @Override
```

Fig. 28. Pseudocode For Mobile App & Event Notification Page 2

```
    public Fragment getItem(int position) {
        return mFragmentList.get(position);
    }

    @Override
    public int getCount() {
        return mFragmentList.size();
    }
}


================================================
================================================
===  MAIN ACTIVITY TO HOUSE ALL FRAGMENTS   ====
================================================
================================================
================================================

package khalil.smartwheelchair;

import android.content.DialogInterface;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v4.app.Fragment;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.view.MenuItem;

public class MainActivity extends AppCompatActivity {

    private SectionsStatePagerAdapter mSectionsStatePagerAdapter;
    private CustomViewPager mViewPager;
    private static final int REQUEST_SMS = 0;

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

        mSectionsStatePagerAdapter = new SectionsStatePagerAdapter(getSupportFragmentManager());
        mViewPager = (CustomViewPager) findViewById(R.id.container);
        setupViewPager(mViewPager);

        BottomNavigationView bottomNavigationView = (BottomNavigationView)
                findViewById(R.id.navigation);

        bottomNavigationView.setOnNavigationItemSelectedListener
                (new BottomNavigationView.OnNavigationItemSelectedListener() {
                    @Override
                    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                        Fragment selectedFragment = null;
                        switch (item.getItemId()) {
                            case R.id.navigation_home:
                                setViewPager(0);
                                break;
                            case R.id.navigation_dashboard:
                                setViewPager(1);
                                break;
                            case R.id.navigation_notifications:
                                setViewPager(2);
                                break;
                        }
                        return true;
                    }
                });
```

Fig. 29. Pseudocode For Mobile App & Event Notification Page 3

```
        requestSMSPermission();

    }

    private void setupViewPager(ViewPager viewPager){
        SectionsStatePagerAdapter sectionsStatePagerAdapter = new
SectionsStatePagerAdapter(getSupportFragmentManager());
        sectionsStatePagerAdapter.addFragment(new Fragment2(), "Order History");
        sectionsStatePagerAdapter.addFragment(new Fragment1(), "Market Overview");
        sectionsStatePagerAdapter.addFragment(new Fragment3(), "Order History");

        viewPager.setAdapter(sectionsStatePagerAdapter);
    }

    public void setViewPager(int fragmentNum){
        mViewPager.setCurrentItem(fragmentNum);
    }

    public void requestSMSPermission() {
        int hasSMSPermission = checkSelfPermission(android.Manifest.permission.SEND_SMS);
        if (hasSMSPermission != PackageManager.PERMISSION_GRANTED) {
            if (!shouldShowRequestPermissionRationale(android.Manifest.permission.SEND_SMS)) {
                showMessageOKCancel("This application requires some permissions to operate properly.",
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                                    requestPermissions(new String[]{android.Manifest.permission.SEND_SMS,
                                                    android.Manifest.permission.READ_PHONE_STATE,
                                                    android.Manifest.permission.INTERNET},
                                            REQUEST_SMS);
                                }
                            }
                        });
                return;
            }
            requestPermissions(new String[]{android.Manifest.permission.SEND_SMS,
                            android.Manifest.permission.READ_PHONE_STATE,
                            android.Manifest.permission.INTERNET},
                    REQUEST_SMS);
        }

    }

    private void showMessageOKCancel(String message, DialogInterface.OnClickListener okListener) {
        new android.support.v7.app.AlertDialog.Builder(MainActivity.this)
                .setMessage(message)
                .setPositiveButton("OK", okListener)
                .setNegativeButton("Cancel", null)
                .create()
                .show();
    }
}


================================================
================================================
=======   FRAGMENT 1: LOCATE WHEELCHAIR  =======
================================================
================================================
================================================


package khalil.smartwheelchair;


import android.*;
```

Fig. 30. Pseudocode For Mobile App & Event Notification Page 4

```
import android.app.AlertDialog;
import android.app.DownloadManager;
import android.app.ProgressDialog;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.location.Address;
import android.location.Geocoder;
import android.net.wifi.SupplicantState;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.provider.Settings;
import android.support.design.widget.Snackbar;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.ubidots.ApiClient;
import com.ubidots.Value;
import com.ubidots.Variable;

import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.ubidots.ApiClient;
import com.ubidots.Variable;

import org.json.JSONObject;

import java.io.IOException;
import java.text.DateFormat;
import java.util.Calendar;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Timer;
import java.util.TimerTask;

/**
```

Fig. 31. Pseudocode For Mobile App & Event Notification Page 5

```
 * Created by khalil on 12/22/17.
 */

public class Fragment1 extends Fragment implements OnMapReadyCallback{
    private GoogleMap mMap;
    private Button mLocateBtn;
    private TextView mAddress;
    private TextView mStatus;
    private ImageView mStatusCircle;
    private Toast mToast;
    private ImageView mGPSPin;
    private RequestQueue mRequestQueue;
    public JsonObjectRequest stringRequest;
    private static final String TAG = MainActivity.class.getName();
    private static final String REQUESTTAG = "Price Request";
    Double longitude = 0d;
    Double latitude = 0d;
    String address;
    Timer timer;
    LatLng location;
    Boolean pause;
    Boolean locationUpdated = true;
    Boolean isInitialized = false;
    double[] latlng = {0d,0d};


    @Override
    public void setUserVisibleHint(boolean isVisibleToUser) {
        super.setUserVisibleHint(isVisibleToUser);
        if (!isVisibleToUser) {
            if (timer != null) {
                System.out.println("STOPPING FRAGMENT 1 TIMER");
                timer.purge();
                timer.cancel();
                timer = null;
            }
        } else {
            if(isConnected()){
                displayDialog();
            }
            if (timer == null && isInitialized) {
                System.out.println("STARTING FRAGMENT 1 TIMER");
                startTimer();
            }
        }
    }

    @Override
    public void onViewCreated(View view, Bundle savedInstanceState){
        SupportMapFragment mapFragment = (SupportMapFragment)
getChildFragmentManager().findFragmentById(R.id.map);
        mLocateBtn = getActivity().findViewById(R.id.locatebutton);
        mAddress = getActivity().findViewById(R.id.address);
        mStatus = getActivity().findViewById(R.id.status_text);
        mStatusCircle = getActivity().findViewById(R.id.status_circle);
        mapFragment.getMapAsync(this);
        mLocateBtn.setBackgroundResource(R.drawable.selector_locate);
        mGPSPin = getActivity().findViewById(R.id.gps_pin);
        mGPSPin.setImageResource(R.drawable.gps_pin_disabled);
        mLocateBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                geoLocate(getActivity().findViewById(R.id.map));
            }
        });
    }

    public boolean isConnected(){
```

Fig. 32. Pseudocode For Mobile App & Event Notification Page 6

```
        String ssid;

        WifiManager wifiManager = (WifiManager)
getActivity().getApplicationContext().getSystemService(Context.WIFI_SERVICE);
        WifiInfo wifiInfo = wifiManager.getConnectionInfo();

        if (wifiInfo.getSupplicantState() == SupplicantState.COMPLETED) {
            ssid = wifiInfo.getSSID();
            System.out.println("SSID " + ssid);
            if(ssid.toLowerCase().contains("smartwheelchair")) {
                    return true;
            } else {
                return false;
            }
        }
        return false;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState){
        View v = inflater.inflate(R.layout.fragment1_layout, container, false);
        return v;
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        String str = "38.5816,-121.4944";
        String[] coordinates = str.split(",");

        goToLocation(Double.parseDouble(coordinates[0]), Double.parseDouble(coordinates[1]));
        geoLocate(getActivity().findViewById(R.id.map));
        isInitialized = true;
    }

    public void goToLocation(double lat, double lng) {
        LatLng ll = new LatLng(lat, lng);
        CameraUpdate update = CameraUpdateFactory.newLatLng(ll);
        mMap.moveCamera(update);
    }

    public void goToLocationZoom(double lat, double lng, float zoom) {
        LatLng ll = new LatLng(lat, lng);
        CameraUpdate update = CameraUpdateFactory.newLatLngZoom(ll, zoom);
        mMap.moveCamera(update);
    }

    public void geoLocate(final View view) {
        Button button = (Button) getActivity().findViewById(R.id.Locatebutton);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                if(!isConnected() && !latitude.equals(0d) && !longitude.equals(0d)) {
                    moveCameratoLocation();
                }
            }
        });
    }

    private void startTimer() {
        timer = new Timer();
        timer.scheduleAtFixedRate(new TimerTask() {
            public void run() {
                        if(!isConnected()) {
                            getActivity().runOnUiThread(new Runnable() {
                                @Override
                                public void run() {
                                    new ApiUbidots().execute();
```

Fig. 33. Pseudocode For Mobile App & Event Notification Page 7

```
                                }
                            });
                        } else {
                            getActivity().runOnUiThread(new Runnable() {
                                @Override
                                public void run() {
                                    mAddress.setText("N/A\nN/A");
                                    mStatus.setText("Status: No Location\nUpdated: N/A");
                                    mGPSPin.setImageResource(R.drawable.gps_pin_disabled);
                                    mStatusCircle.setImageResource(R.drawable.status_red);
                                }
                            });
                        }
                    }
                }, 1000, 3000); //Length of how often to update location
            }

            public void displayDialog(){
                getActivity().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        AlertDialog.Builder builder;
                        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                            builder = new AlertDialog.Builder(getContext(),
        android.R.style.Theme_Material_Dialog_Alert);
                        } else {
                            builder = new AlertDialog.Builder(getContext());
                        }
                        builder.setTitle("Connect to the Internet")
                                .setMessage("In order to locate the powered wheelchair, you must disconnect from its
        wireless connection. Press 'OK' to disconnect from the powered wheelchair.")
                                .setPositiveButton("Ok", new DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface dialog, int which) {

                                        startActivityForResult(new Intent(Settings.ACTION_WIFI_SETTINGS),0);
                                        //moveCameratoLocation();
                                    }
                                })
                                .setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface dialog, int which) {
                                        toastMessage("Cancelling Wheelchair Locate");
                                    }
                                })
                                .setIcon(android.R.drawable.ic_dialog_alert)
                                .setCancelable(false)
                                .show();
                    }
                });
            }

            private void toastMessage(final String message){
                getActivity().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        if(mToast!=null)
                            mToast.cancel();
                        mToast = Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT);
                        mToast.setGravity(Gravity.CENTER_HORIZONTAL,0,700);
                        mToast.show();
                    }
                });
            }

            public void getLocation(Long timestamp) {
                View view = getActivity().findViewById(R.id.map);

                latlng[0] = longitude;
                latlng[1] = latitude;
```

Fig. 34. Pseudocode For Mobile App & Event Notification Page 8

```
        Geocoder geocoder;
        List<Address> addresses = null;

        geocoder = new Geocoder(view.getContext(), Locale.getDefault());

        try {
            addresses = geocoder.getFromLocation(latlng[0], latlng[1], 1); // Here 1 represent max location
result to returned, by documents it recommended 1 to 5
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            address = addresses.get(0).getAddressLine(0); // If any additional address line present than
only, check with max available address lines by getMaxAddressLineIndex()
            String city = addresses.get(0).getLocality();
            String state = addresses.get(0).getAdminArea();
            String country = addresses.get(0).getCountryName();
            String postalCode = addresses.get(0).getPostalCode();
            String knownName = addresses.get(0).getFeatureName(); // Only if available else return NULL
            mAddress.setText(address.split(",")[0] + "\n" +
                    address.split(",")[1].trim() + ", " +
                    address.split(",")[2].trim());
            getDate(timestamp);
            mGPSPin.setImageResource(R.drawable.gps_pin);
        } catch (Exception e) {
            address = "Unknown Location";
        }

        location = new LatLng(latlng[0], latlng[1]);
        if(locationUpdated) {
            mMap.clear();
            mMap.addMarker(new MarkerOptions()
                    .position(location)
                    .title(address)
                    .icon(BitmapDescriptorFactory.fromResource(R.drawable.gps_dot))
                    .anchor(0.5f, 0.5f));
        }
    }

    public void moveCameratoLocation(){
        mMap.moveCamera(CameraUpdateFactory.newLatLng(location));
        goToLocationZoom(latlng[0], latlng[1], 16);
    }

    public void sendMySMS() {

        String phone = "9165739798";
        String message = "LOCATE";

    }

    @Override
    public void onDestroy(){
        super.onDestroy();
        pause = true;
    }

    @Override
    public void onPause(){
        super.onPause();
        pause = true;
    }

    @Override
    public void onStop(){
        super.onStop();
    }
```

Fig. 35. Pseudocode For Mobile App & Event Notification Page 9

```
    @Override
    public void onResume(){
        super.onResume();
    }

    public class ApiUbidots extends AsyncTask<Integer, Void, com.ubidots.Value[]> {
        private final String API_KEY = "A1E-84aa9fd314060bd8013989f6dbd8176b6f4e";
        private final String Lat_ID = "59e1c8e6c03f977288035309";
        private final String Long_ID = "59e1c4b4c03f976d8614041e";

        @Override
        protected com.ubidots.Value[] doInBackground(Integer... params) {
            try {
                ApiClient apiClient = new ApiClient(API_KEY);
                Variable longitude = apiClient.getVariable(Long_ID);
                com.ubidots.Value[] lon = longitude.getValues();
                return (lon);
            } catch (Exception e) {
                System.out.println("ERRORORORORORO");
                e.printStackTrace();
            }
            return null;
        }

        @Override
        protected void onPostExecute(com.ubidots.Value[] lon){
            if(lon != null) {
                String latlng_string = String.format("%.0f", lon[0].getValue());
                Double lng = Double.parseDouble(latlng_string.substring(2, 9).replaceFirst("^0+(?!$)", "")) /
10000;

                Double lat = Double.parseDouble(latlng_string.substring(10, 17).replaceFirst("^0+(?!$)", ""))
/ -10000;
                //Toast.makeText(getContext(), Double.toString(lng) + ", " +
Double.toString(lat),Toast.LENGTH_SHORT).show();
                if (!longitude.equals(lng) && !latitude.equals(lat)) {
                    longitude = lng;
                    latitude = lat;
                    locationUpdated = true;
                } else
                    locationUpdated = false;
                getLocation(lon[0].getTimestamp());
            }
        }
    }

    private String getDate(final long time) {
        Calendar cal = Calendar.getInstance(Locale.ENGLISH);
        cal.setTimeInMillis(time);
        final String date = android.text.format.DateFormat.format("MM/dd HH:mm a", cal).toString();
        getActivity().runOnUiThread(new Runnable() {
            @Override
            public void run() {
                String status_line;
                long tenAgo = System.currentTimeMillis() - 10 * 60 * 1000;
                if(time < tenAgo){
                    status_line = "Status: Not Live Data\n";
                    mStatusCircle.setImageResource(R.drawable.status_yellow);
                } else {
                    status_line = "Status: Live Data\n";
                    mStatusCircle.setImageResource(R.drawable.status_green);
                }
                mStatus.setText(status_line + "Updated: " + date);
            }
        });
        return date;
    }
}
```

Fig. 36. Pseudocode For Mobile App & Event Notification Page 10

```
================================================
================================================
====  FRAGMENT 2: CONFIGURE WHEELCHAIR  ========
================================================
================================================
================================================

package khalil.smartwheelchair;


import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.media.MediaCas;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.wifi.SupplicantState;
import android.net.wifi.WifiConfiguration;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.provider.Settings;
import android.support.design.widget.Snackbar;
import android.support.v4.app.Fragment;
import android.telephony.PhoneNumberFormattingTextWatcher;
import android.telephony.PhoneNumberUtils;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.CheckedTextView;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.facebook.shimmer.ShimmerFrameLayout;
import com.jcraft.jsch.ChannelExec;
import com.jcraft.jsch.JSch;
import com.jcraft.jsch.Session;

import org.json.JSONObject;
import org.w3c.dom.Text;

import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Properties;
import java.util.Timer;
import java.util.TimerTask;
import java.util.concurrent.ExecutionException;

import fr.castorflex.android.smoothprogressbar.SmoothProgressBar;
```

Fig. 37. Pseudocode For Mobile App & Event Notification Page 11

```
import fr.castorflex.android.smoothprogressbar.SmoothProgressDrawable;

/**
 * Created by khalil on 12/22/17.
 */

public class Fragment2 extends Fragment {
    private RelativeLayout mSettings;
    private Button mButton;
    private Button mCollisionButton;
    private Button mTiltButton;
    private Button mGPSButton;
    private Button mStepButton;
    private Button mConnectButton;
    private Button mIconButton;
    private Toast mToast;
    private EditText mPhonenum;
    private EditText mCommand;
    private ShimmerFrameLayout mShimmerFrameLayout;

    private String command;
    private boolean isInitialized = false;
    private boolean firstRun = true;

    private ProgressDialog mProgress;
    private ProgressBar mConnectionProgress;
    private ProgressBar mProgressBar;
    private ProgressBar mCollisionProgress;
    private ProgressBar mTiltProgress;
    private ProgressBar mGPSProgress;
    private ProgressBar mStepProgress;

    private Timer mTimer;
    private TextView mConnection;
    private TextView mCollisionStatus;
    private TextView mTiltStatus;
    private TextView mGPSStatus;
    private TextView mStepStatus;
    private ShimmerFrameLayout mPressConnect;
    private String mPhoneFile = "/home/pi/Desktop/everyone/vals/phone.txt";
    private String mCollisionFile = "/home/pi/Desktop/everyone/vals/collision.txt";
    private String mTiltFile = "/home/pi/Desktop/everyone/vals/tilt.txt";
    private String mGPSFile = "/home/pi/Desktop/everyone/vals/gps.txt";
    private String mStepFile = "/home/pi/Desktop/everyone/vals/step.txt";
    private String mSoundEnabled = "mpg123 /home/pi/Desktop/everyone/audio/sound.mp3";
    private String mCollisionEnabled = "mpg123 /home/pi/Desktop/everyone/audio/collision_enabled.mp3";
    private String mCollisionDisabled = "mpg123 /home/pi/Desktop/everyone/audio/collision_disabled.mp3";
    private String mTiltEnabled = "mpg123 /home/pi/Desktop/everyone/audio/tilt_enabled.mp3";
    private String mTiltDisabled = "mpg123 /home/pi/Desktop/everyone/audio/tilt_disabled.mp3";
    private String mGPSEnabled = "mpg123 /home/pi/Desktop/everyone/audio/gps_enabled.mp3";
    private String mGPSDisabled = "mpg123 /home/pi/Desktop/everyone/audio/gps_disabled.mp3";
    private String mStepEnabled = "mpg123 /home/pi/Desktop/everyone/audio/step_enabled.mp3";
    private String mStepDisabled = "mpg123 /home/pi/Desktop/everyone/audio/step_disabled.mp3";
    private String mPhoneUpdated = "mpg123 /home/pi/Desktop/everyone/audio/phone_updated.mp3";

    @Override
    public void setUserVisibleHint(boolean isVisibleToUser) {
        super.setUserVisibleHint(isVisibleToUser);
        if (isVisibleToUser){

            if (mTimer == null && isInitialized) {
                System.out.println("STARTING FRAGMENT 2 TIMER");
                connectionTimer();
            }
        }else{
            if (mTimer != null) {
                System.out.println("STOPPING FRAGMENT 2 TIMER");
                mTimer.purge();
```

Fig. 38. Pseudocode For Mobile App & Event Notification Page 12

```
                mTimer.cancel();
                mTimer = null;
                firstRun = true;
            }
        }
    }

    private void fadeOut() {
        Animation bottomDown = AnimationUtils.loadAnimation(getContext(),
                R.anim.alpha_r);
        bottomDown.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {

            }

            @Override
            public void onAnimationEnd(Animation animation) {
                animateButtonUp();
            }

            @Override
            public void onAnimationRepeat(Animation animation) {

            }
        });
        mSettings.startAnimation(bottomDown);
        mSettings.setVisibility(View.INVISIBLE);
    }

    private void animateButtonUp() {
        Animation bottomUp = AnimationUtils.loadAnimation(getContext(),
                R.anim.alpha);
        bottomUp.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {

            }

            @Override
            public void onAnimationEnd(Animation animation) {
            }

            @Override
            public void onAnimationRepeat(Animation animation) {

            }
        });
        mIconButton.startAnimation(bottomUp);
        mPressConnect.startAnimation(bottomUp);
        mPressConnect.setVisibility(View.VISIBLE);
        mIconButton.setVisibility(View.VISIBLE);
    }

    private void pullSettings(){
        List<String> settings = new ArrayList<String>();

    }


    private void connectionTimer() {
        mTimer = new Timer();

        mTimer.scheduleAtFixedRate(new TimerTask() {
            public void run() {
                if(isVisible()) {
                    new Thread(new Runnable() {
                        @Override
```

Fig. 39. Pseudocode For Mobile App & Event Notification Page 13

```
                        public void run() {
                            if (!isConnected()) {
                                getActivity().runOnUiThread(new Runnable() {
                                    @Override
                                    public void run() {
                                        mConnection.setText("STATUS: DISCONNECTED");
                                        mConnectButton.setTag("off");
                                        mPhonenum.setFocusableInTouchMode(false);
                                        mPhonenum.setText("");
                                        mPhonenum.setAlpha(0.5f);

mCollisionButton.setBackgroundResource(R.drawable.collision_button_off);
                                        mCollisionStatus.setText("");
                                        mTiltButton.setBackgroundResource(R.drawable.tilt_button_off);
                                        mTiltStatus.setText("");
                                        mGPSButton.setBackgroundResource(R.drawable.gps_button_off);
                                        mGPSStatus.setText("");
                                        mStepButton.setBackgroundResource(R.drawable.step_button_off);
                                        mStepStatus.setText("");
                                        //mConnectButton.setBackgroundResource(R.drawable.selector_connect);
                                        mConnectButton.setAlpha(1f);
                                        mConnection.setTextColor(getResources().getColor(R.color.redColor));
                                        mButton.setBackgroundResource(R.drawable.check_button_gray);
                                    }
                                });
                            }
                        }
                    }).start();
                }
            }
        }, 1000, 5000); //Length of how often to update location
    }

    @Override
    public void onViewCreated(View view, Bundle savedInstanceState){
        mSettings = getActivity().findViewById(R.id.settings_view);
        mSettings.setVisibility(View.INVISIBLE);
        mIconButton = getActivity().findViewById(R.id.icon_button);
        mButton = getActivity().findViewById(R.id.phone_check);
        mCollisionButton = getActivity().findViewById(R.id.collision_button);
        mTiltButton = getActivity().findViewById(R.id.tilt_button);
        mGPSButton = getActivity().findViewById(R.id.gps_button);
        mStepButton = getActivity().findViewById(R.id.step_button);
        mConnectButton = getActivity().findViewById(R.id.connect_button);
        mPhonenum = getActivity().findViewById(R.id.phone_number);
        mPhonenum.addTextChangedListener(new PhoneNumberFormattingTextWatcher());
        mPhonenum.setBackgroundResource(R.drawable.selector_textwindow);
        mConnection = getActivity().findViewById(R.id.link_status);
        mConnectionProgress = getActivity().findViewById(R.id.connection_progress);
        mProgressBar = getActivity().findViewById(R.id.connection);
        mCollisionProgress = getActivity().findViewById(R.id.collision_progress);
        mTiltProgress = getActivity().findViewById(R.id.tilt_progress);
        mGPSProgress = getActivity().findViewById(R.id.gps_progress);
        mStepProgress = getActivity().findViewById(R.id.step_progress);
        mCollisionStatus = getActivity().findViewById(R.id.collision_status);
        mTiltStatus = getActivity().findViewById(R.id.tilt_status);
        mGPSStatus = getActivity().findViewById(R.id.gps_status);
        mStepStatus = getActivity().findViewById(R.id.step_status);
        mShimmerFrameLayout = getActivity().findViewById(R.id.shimmer_icon);
        mPressConnect = getActivity().findViewById(R.id.shimmer_text);
        mPressConnect.startShimmerAnimation();


        mProgressBar.setVisibility(View.INVISIBLE);
        mConnectionProgress.setVisibility(View.INVISIBLE);
        mCollisionProgress.setVisibility(View.INVISIBLE);
        mTiltProgress.setVisibility(View.INVISIBLE);
        mGPSProgress.setVisibility(View.INVISIBLE);
```

Fig. 40. Pseudocode For Mobile App & Event Notification Page 14

```
        mStepProgress.setVisibility(View.INVISIBLE);

        mIconButton.setBackgroundResource(R.drawable.selector_icon_button);
        mButton.setBackgroundResource(R.drawable.selector_save_config);
        mCollisionButton.setBackgroundResource(R.drawable.collision_button_off);
        mTiltButton.setBackgroundResource(R.drawable.tilt_button_off);
        mGPSButton.setBackgroundResource(R.drawable.gps_button_off);
        mStepButton.setBackgroundResource(R.drawable.step_button_off);
        mConnectButton.setBackgroundResource(R.drawable.selector_connect);
        mConnection.setText("STATUS: DISCONNECTED");
        mPhonenum.setFocusableInTouchMode(false);
        mConnection.setTextColor(getResources().getColor(R.color.redColor));

        mCollisionButton.setTag("on");
        mTiltButton.setTag("on");
        mGPSButton.setTag("on");
        mStepButton.setTag("on");
        mConnectButton.setTag("off");


        mProgress = new ProgressDialog(getActivity());
        mProgress.setMessage("Updating settings...");

        mIconButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(!isConnected()){
                    displayDialog();
                } else {
                    animateDown();
                }

            }
        });
        mButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String connection = mConnection.getText().toString();
                String phoneNum = mPhonenum.getText().toString();
                if(!connection.contains("DISCONNECTED")){
                    if(phoneNum.length() != 10) {
                        mPhonenum.setText("");
                        toastMessage("Invalid Phone Number Length!");
                    }else {
                        sendCommand("echo '" + phoneNum + "' > " + mPhoneFile + "; " + mPhoneUpdated, true,
"update_setting", "mPhonenum", false);
                    }
                } else {
                    notConnectedAlert();
                }
            }
        });

        mCollisionButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String connection = mConnection.getText().toString();
                if(!connection.contains("DISCONNECTED")) {
                    String result;

                    if (mCollisionButton.getTag().equals("on")) {
                        sendCommand("echo '0' > " + mCollisionFile + "; " + mCollisionDisabled, true,
"update_setting", "mCollision", false);
                    }
                    else {
                        sendCommand("echo '1' > " + mCollisionFile + "; " + mCollisionEnabled, true,
"update_setting", "mCollision", false);
                    }
                } else {
```

Fig. 41. Pseudocode For Mobile App & Event Notification Page 15

```
                    notConnectedAlert();
                }
            }
        });

        mTiltButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String connection = mConnection.getText().toString();
                if(!connection.contains("DISCONNECTED")) {
                    String result;

                    if (mTiltButton.getTag().equals("on")) {
                        sendCommand("echo '0' > " + mTiltFile + "; " + mTiltDisabled, true, "update_setting",
"mTilt", false);
                    }
                    else {
                        sendCommand("echo '1' > " + mTiltFile + "; " + mTiltEnabled, true, "update_setting",
"mTilt", false);
                    }
                } else {
                    notConnectedAlert();
                }
            }
        });

        mGPSButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String connection = mConnection.getText().toString();
                if(!connection.contains("DISCONNECTED")) {
                    String result;

                    if (mGPSButton.getTag().equals("on")) {
                        sendCommand("echo '0' > " + mGPSFile + "; " + mGPSDisabled, true, "update_setting",
"mRealtime", false);
                    }
                    else {
                        sendCommand("echo '1' > " + mGPSFile + "; " + mGPSEnabled, true, "update_setting",
"mRealtime", false);
                    }
                } else {
                    notConnectedAlert();
                }
            }
        });

        mStepButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String connection = mConnection.getText().toString();
                if(!connection.contains("DISCONNECTED")) {
                    String result;

                    if (mStepButton.getTag().equals("on")) {
                        sendCommand("echo '0' > " + mStepFile + "; " + mStepDisabled, true, "update_setting",
"mStep", false);
                    }
                    else {
                        sendCommand("echo '1' > " + mStepFile + "; " + mStepEnabled, true, "update_setting",
"mStep", false);
                    }
                } else {
                    notConnectedAlert();
                }
            }
        });
```

Fig. 42. Pseudocode For Mobile App & Event Notification Page 16

```
        mConnectButton.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                if(mConnectButton.getTag().equals("off")){
                    displayDialog();
                } else {
                    Snackbar.make(getView(), "Already Connected!", Snackbar.LENGTH_SHORT).show();
                }
            }
        });

        mPhonenum.setOnFocusChangeListener(new View.OnFocusChangeListener() {
            @Override
            public void onFocusChange(View v, boolean hasFocus) {
                if (!hasFocus) {
                    hideKeyboard(v);
                }
            }

        });



        mPhonenum.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                if(mConnectButton.getTag().equals("off"))
                    notConnectedAlert();
            }
        });

        isInitialized = true;
        //if(!isConnected())
        //    displayDialog();
    }

    public void animateUp(){
        Animation bottomUp = AnimationUtils.loadAnimation(getContext(),
                R.anim.alpha);
        bottomUp.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {

            }

            @Override
            public void onAnimationEnd(Animation animation) {
                if (mTimer == null && isInitialized) {
                    System.out.println("STARTING FRAGMENT 2 TIMER");
                    connectionTimer();
                }
            }

            @Override
            public void onAnimationRepeat(Animation animation) {

            }
        });
        mSettings.startAnimation(bottomUp);
        mSettings.setVisibility(View.VISIBLE);
    }

    public void animateDown(){
        Animation bottomDown = AnimationUtils.loadAnimation(getContext(),
                R.anim.alpha_r);
        bottomDown.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {
```

Fig. 43. Pseudocode For Mobile App & Event Notification Page 1

```
            }

            @Override
            public void onAnimationEnd(Animation animation) {
                mIconButton.setVisibility(View.INVISIBLE);
                mPressConnect.setVisibility(View.INVISIBLE);
                establishConnection();
            }

            @Override
            public void onAnimationRepeat(Animation animation) {

            }
        });
        mIconButton.startAnimation(bottomDown);
        mPressConnect.startAnimation(bottomDown);
    }

    private void establishConnection() {
        mConnectionProgress.setVisibility(View.VISIBLE);
        mConnectionProgress.setTag("Connecting");
        connectionTimer();
    }

    private void notConnectedAlert() {
        AlertDialog.Builder alert = new AlertDialog.Builder(getContext(),
android.R.style.Theme_Material_Dialog_Alert);
        alert.setTitle("Not Connected to Powered Wheelchair");
        alert.setMessage("You are not connected to the powered wheelchair. Dismiss this dialog and tap the
'Connect to Wheelchair' button below to connect.");
        alert.setPositiveButton("OK",null);
        alert.show();
    }

    public void hideKeyboard(View view) {
        InputMethodManager inputMethodManager
=(InputMethodManager)getActivity().getSystemService(MainActivity.INPUT_METHOD_SERVICE);
        inputMethodManager.hideSoftInputFromWindow(view.getWindowToken(), 0);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState){
        View v = inflater.inflate(R.layout.fragment2_layout, container, false);

        //Initialize the views
        //Pull Bitcoin-USD Data
        //Pull Ticker Data
        //Convert Ticker share to bitcoin shares, then bitcoin shares to USD
        return v;
    }

    public String executeRemoteCommand(
            String username,
            String password,
            String hostname,
            int port, boolean getOutput) throws Exception {

        String output = "";
        final String output2;

        JSch jsch = new JSch();
        Session session = jsch.getSession(username, hostname, port);
        session.setPassword(password);
        session.setConfig("StrictHostKeyChecking", "no");
        session.setTimeout(5000);
        session.connect();
```

Fig. 44. Pseudocode For Mobile App & Event Notification Page 18

```
        ChannelExec channel = (ChannelExec)session.openChannel("exec");
        channel.setCommand(command);
        channel.connect();
        if(getOutput) {
            //Thread.sleep(1000);
            InputStream in = channel.getInputStream();
            byte[] tmp = new byte[1024];
            while (true) {
                while (in.available() > 0) {
                    mProgress.setMessage("Receiving data ...");
                    int i = in.read(tmp, 0, 1024);
                    if (i < 0) break;
                    output += (new String(tmp, 0, i));
                }
                if (channel.isClosed()) {
                    if (in.available() > 0) continue;
                    output2 = output;
                    getActivity().runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            //toastMessage(output2);
                        }
                    });
                    System.out.println("exit-status: " + channel.getExitStatus());
                    break;
                }
                try {
                    Thread.sleep(100);
                } catch (Exception ee) {
                }
            }
            channel.disconnect();
        } else {
            channel.disconnect();
            return "Success!";
        }

        return output;
        // show success in UI with a snackbar alternatively use a toast
    }

    private void toastMessage(final String message){
        getActivity().runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if(mToast!=null)
                    mToast.cancel();
                mToast = Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT);
                mToast.setGravity(Gravity.CENTER_HORIZONTAL,0,900);
                mToast.show();
            }
        });
    }

    private void sendCommand(final String cmd, final Boolean display, final String purpose, final String
setting, final boolean getOutput){
        String result;
                new AsyncTask<String, Void, String>(){
                    String result;
                    @Override
                    protected String doInBackground(String... params) {
                        try {
                            command = params[0];
                            result = executeRemoteCommand("pi", "Dvdboxset890", "192.168.42.1", 22,
getOutput);
                            //toastMessage(executeRemoteCommand("pi", "raspberry", "192.168.42.1", 22));
                        } catch (Exception e) {
                            e.printStackTrace();
```

Fig. 45. Pseudocode For Mobile App & Event Notification Page 19

```
                result = "Failed!";
                System.out.println("Failed!!!!!");
            }
            return result;
        }

        @Override
        protected void onPostExecute(String result) {
            super.onPostExecute(result);
            mProgressBar.setVisibility(View.INVISIBLE);

            if (purpose.equals("get_settings")) {
                if(firstRun){
                    mProgress.hide();
                    firstRun = false;
                }
                if(result != null)
                    System.out.println("RESULT IS : " + result);
                if((result == null || result.contains("Failed")) && isVisible()) {
                    mConnection.setText("STATUS: DISCONNECTED");
                    mConnectButton.setTag("off");
                    mPhonenum.setFocusableInTouchMode(false);
                    mPhonenum.setText("");
                    mPhonenum.setAlpha(0.5f);
                    mCollisionButton.setBackgroundResource(R.drawable.collision_button_off);
                    mCollisionStatus.setText("");
                    mTiltButton.setBackgroundResource(R.drawable.tilt_button_off);
                    mTiltStatus.setText("");
                    mGPSButton.setBackgroundResource(R.drawable.gps_button_off);
                    mGPSStatus.setText("");
                    mStepButton.setBackgroundResource(R.drawable.step_button_off);
                    mStepStatus.setText("");
                    //mConnectButton.setBackgroundResource(R.drawable.selector_connect);
                    mConnectButton.setAlpha(1f);
                    mConnection.setTextColor(getResources().getColor(R.color.redColor));
                    mButton.setBackgroundResource(R.drawable.check_button_gray);
                }
                else if (isVisible()) {
                    mConnection.setText("STATUS: CONNECTED");
                    mConnectionProgress.setVisibility(View.INVISIBLE);
                    mConnectButton.setTag("on");
                    mPhonenum.setFocusableInTouchMode(true);
                    mPhonenum.setAlpha(1f);
                    //mConnectButton.setBackgroundResource(R.drawable.connect_button_gray);
                    mConnectButton.setAlpha(0f);
                    mConnection.setTextColor(getResources().getColor(R.color.green));
                    mButton.setBackgroundResource(R.drawable.selector_save_config);
                    updateSettings(result);
                    if(mConnectionProgress.getTag() != null &&
mConnectionProgress.getTag().equals("Connecting"))
                        animateUp();
                    mConnectionProgress.setTag("Connected");
                }
            } else if (!result.contains("Failed")){
                if(setting.equals("mTilt")){
                    if (mTiltButton.getTag().equals("on")) {
                        mTiltButton.setBackgroundResource(R.drawable.tilt_button_off);
                        mTiltButton.setTag("off");
                        mTiltStatus.setText("DISABLED");
                        mTiltStatus.setTextColor(getResources().getColor(R.color.redColor));
                    }
                    else {
                        mTiltButton.setBackgroundResource(R.drawable.selector_tilt);
                        mTiltButton.setTag("on");
                        mTiltStatus.setText("ENABLED");
                        mTiltStatus.setTextColor(getResources().getColor(R.color.green));
                    }
                    mTiltProgress.setVisibility(View.INVISIBLE);
```

Fig. 46. Pseudocode For Mobile App & Event Notification Page 20

```
        } else if (setting.equals("mCollision")){
            if (mCollisionButton.getTag().equals("on")) {
                mCollisionButton.setBackgroundResource(R.drawable.collision_button_off);
                mCollisionButton.setTag("off");
                mCollisionStatus.setText("DISABLED");
                mCollisionStatus.setTextColor(getResources().getColor(R.color.redColor));
            }
            else {
                mCollisionButton.setBackgroundResource(R.drawable.selector_collision);
                mCollisionButton.setTag("on");
                mCollisionStatus.setText("ENABLED");
                mCollisionStatus.setTextColor(getResources().getColor(R.color.green));
            }
            mCollisionProgress.setVisibility(View.INVISIBLE);
        } else if (setting.equals("mRealtime")){
            if (mGPSButton.getTag().equals("on")) {
                mGPSButton.setBackgroundResource(R.drawable.gps_button_off);
                mGPSButton.setTag("off");
                mGPSStatus.setText("DISABLED");
                mGPSStatus.setTextColor(getResources().getColor(R.color.redColor));
            }
            else {
                mGPSButton.setBackgroundResource(R.drawable.selector_gps);
                mGPSButton.setTag("on");
                mGPSStatus.setText("ENABLED");
                mGPSStatus.setTextColor(getResources().getColor(R.color.green));
            }
            mGPSProgress.setVisibility(View.INVISIBLE);
        } else if (setting.equals("mStep")){
            if (mStepButton.getTag().equals("on")) {
                mStepButton.setBackgroundResource(R.drawable.step_button_off);
                mStepButton.setTag("off");
                mStepStatus.setText("DISABLED");
                mStepStatus.setTextColor(getResources().getColor(R.color.redColor));
            }
            else {
                mStepButton.setBackgroundResource(R.drawable.selector_step);
                mStepButton.setTag("on");
                mStepStatus.setText("ENABLED");
                mStepStatus.setTextColor(getResources().getColor(R.color.green));
            }
            mStepProgress.setVisibility(View.INVISIBLE);
        } else if(setting.equals("mPhonenum")){
            mProgress.hide();
            toastMessage("Updated Phone Number");
        }
    } else {
        toastMessage("Failed to update setting!");
        if(setting.equals("mTilt")){
            mTiltProgress.setVisibility(View.INVISIBLE);
        } else if (setting.equals("mCollision")){
            mCollisionProgress.setVisibility(View.INVISIBLE);
        } else if (setting.equals("mRealtime")){
            mGPSProgress.setVisibility(View.INVISIBLE);
        } else if (setting.equals("mStep")){
            mStepProgress.setVisibility(View.INVISIBLE);
        } else if(setting.equals("mPhonenum")){
            mProgress.hide();
        }

    }
    if(display) {
        mProgress.setMessage("Retrieving settings...");
        getActivity().runOnUiThread(new Runnable() {
            @Override
            public void run() {
                mProgress.hide();
            }
```

Fig. 47. Pseudocode For Mobile App & Event Notification Page 21

```
                    });
                }
            }

            @Override
            protected void onPreExecute() {
                super.onPreExecute();
                if(display){
                    if (purpose.equals("get_settings"))
                        mProgress.setMessage("Updating settings...");
                    else
                        mProgress.setMessage("Sending to Wheelchair...");
                    getActivity().runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            if(purpose.equals("get_settings")){
                                if(firstRun){
                                    mProgress.setMessage("Checking connection status...");
                                    mProgress.show();
                                }
                            }
                            if(setting.equals("mCollision")){
                                mCollisionProgress.setVisibility(View.VISIBLE);
                            } else if (setting.equals("mTilt")){
                                mTiltProgress.setVisibility(View.VISIBLE);
                            } else if (setting.equals("mRealtime")){
                                mGPSProgress.setVisibility(View.VISIBLE);
                            } else if (setting.equals("mStep")){
                                mStepProgress.setVisibility(View.VISIBLE);
                            } else if (setting.equals("mPhonenum")){
                                mProgress.setMessage("Updating Phone Number...");
                                mProgress.show();
                            }
                        }
                    });
                }
            }

        }.execute(cmd);
    }

    public boolean isConnected(){
        String ssid;

        WifiManager wifiManager = (WifiManager)
getActivity().getApplicationContext().getSystemService(Context.WIFI_SERVICE);
        WifiInfo wifiInfo = wifiManager.getConnectionInfo();

        if (wifiInfo.getSupplicantState() == SupplicantState.COMPLETED) {
            ssid = wifiInfo.getSSID();
            System.out.println("SSID " + ssid);
            if(ssid.toLowerCase().contains("smartwheelchair")) {
                //updateSettings();
                command = "echo \"$(</home/pi/Desktop/everyone/vals/collision.txt)\"; echo
\"$(</home/pi/Desktop/everyone/vals/tilt.txt)\"; echo \"$(</home/pi/Desktop/everyone/vals/gps.txt)\"; echo
\"$(</home/pi/Desktop/everyone/vals/step.txt)\"; echo \"$(</home/pi/Desktop/everyone/vals/phone.txt)\"";
                sendCommand(command, false, "get_settings", "null", true);
                return true;
            }
        }
        return false;
    }

    public class spinProgressBar extends AsyncTask<Void, Void, Void> {

        /**
         * Background task to sleep a thread for 1 second while the
         * progress circle spins.
```

Fig. 48. Pseudocode For Mobile App & Event Notification Page 22

```
         * upon successful check-in.
         * @param args no parameters needed for this task.
         * @return null
         */
        @Override
        protected Void doInBackground(Void... args) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            return null;
        }

        /**
         * After waiting one second, hide the progress circle.
         */
        @Override
        protected void onPostExecute(Void result) {
            mProgressBar.setVisibility(View.INVISIBLE);
            super.onPostExecute(result);
        }

        /**
         * Make the progress circle animation visible to
         * the user.
         */
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            mProgressBar.setVisibility(View.VISIBLE);
        }
    }

    private void updateSettings(String result) {
        String[] settings = new String[6];
        settings[0] = (result.split("\n")[0]);
        settings[1] = (result.split("\n")[1]);
        settings[2] = (result.split("\n")[2]);
        settings[3] = (result.split("\n")[3]);
        settings[4] = (result.split("\n")[4]);

        if (settings[0].equals("0")) {
            mCollisionButton.setBackgroundResource(R.drawable.collision_button_off);
            mCollisionButton.setTag("off");
            mCollisionStatus.setText("DISABLED");
            mCollisionStatus.setTextColor(getResources().getColor(R.color.redColor));
        } else {
            mCollisionButton.setBackgroundResource(R.drawable.selector_collision);
            mCollisionButton.setTag("on");
            mCollisionStatus.setText("ENABLED");
            mCollisionStatus.setTextColor(getResources().getColor(R.color.green));
        }
        if (settings[1].equals("0")) {
            mTiltButton.setBackgroundResource(R.drawable.tilt_button_off);
            mTiltButton.setTag("off");
            mTiltStatus.setText("DISABLED");
            mTiltStatus.setTextColor(getResources().getColor(R.color.redColor));
        } else {
            mTiltButton.setBackgroundResource(R.drawable.selector_tilt);
            mTiltButton.setTag("on");
            mTiltStatus.setText("ENABLED");
            mTiltStatus.setTextColor(getResources().getColor(R.color.green));
        }
        if (settings[2].equals("0")) {
            mGPSButton.setBackgroundResource(R.drawable.gps_button_off);
            mGPSButton.setTag("off");
            mGPSStatus.setText("DISABLED");
```

Fig. 49. Pseudocode For Mobile App & Event Notification Page 23

```
            mGPSStatus.setTextColor(getResources().getColor(R.color.redColor));
        } else {
            mGPSButton.setBackgroundResource(R.drawable.selector_gps);
            mGPSButton.setTag("on");
            mGPSStatus.setText("ENABLED");
            mGPSStatus.setTextColor(getResources().getColor(R.color.green));
        }

        if (settings[3].equals("0")) {
            mStepButton.setBackgroundResource(R.drawable.step_button_off);
            mStepButton.setTag("off");
            mStepStatus.setText("DISABLED");
            mStepStatus.setTextColor(getResources().getColor(R.color.redColor));
        } else {
            mStepButton.setBackgroundResource(R.drawable.selector_step);
            mStepButton.setTag("on");
            mStepStatus.setText("ENABLED");
        }

        if(!mPhonenum.hasFocus())
            mPhonenum.setText(settings[4]);
    }

    public void connectWifi(){
        //final String CAPTIVE_PORTAL_DETECTION_ENABLED = "captive_portal_detection_enabled";
        //Settings.Global.putInt(getActivity().getContentResolver(), CAPTIVE_PORTAL_DETECTION_ENABLED, 0);

        String ssid = "SmartWheelchair-3";
        String key = "smart123";
        toastMessage("HERE");

        WifiConfiguration wifiConfig = new WifiConfiguration();
        wifiConfig.allowedAuthAlgorithms.set(WifiConfiguration.AuthAlgorithm.OPEN);
        wifiConfig.SSID = String.format("\"%s\"", ssid);
        wifiConfig.preSharedKey = String.format("\"%s\"", key);

        WifiManager wifiManager = (WifiManager)
getActivity().getApplicationContext().getSystemService(Context.WIFI_SERVICE);
        if(!wifiManager.isWifiEnabled()) {
            wifiManager.setWifiEnabled(true);
        }
        wifiManager.disconnect();
        displayDialog();
//remember id
        //int netId = wifiManager.addNetwork(wifiConfig);
        //wifiManager.enableNetwork(netId, false);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent intent)
    {
        if(requestCode==0)
        {
            WifiManager wifiManager = (WifiManager)
getActivity().getApplicationContext().getSystemService(Context.WIFI_SERVICE);
            if(wifiManager.isWifiEnabled() &&
wifiManager.getConnectionInfo().getSSID().toString().toLowerCase().contains("smartwheelchair")){
                final Intent intent2 = new Intent(getContext(), MainActivity.class);
                startActivity(intent2);
            }
            //restart Application here
        }
    }

    public void displayDialog(){
        AlertDialog.Builder builder;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            builder = new AlertDialog.Builder(getContext(), android.R.style.Theme_Material_Dialog_Alert);
```

Fig. 50. Pseudocode For Mobile App & Event Notification Page 24

```
        } else {
            builder = new AlertDialog.Builder(getContext());
        }
        builder.setTitle("Connect to Powered Wheelchair")
                .setMessage("Press 'OK' to launch settings and select the Smart Wheelchair from the WiFi
list.")
                .setPositiveButton("Ok", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        startActivityForResult(new Intent(Settings.ACTION_WIFI_SETTINGS),0);
                    }
                })
                .setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        toastMessage("Not Connecting to wheelchair");
                    }
                })
                .setIcon(android.R.drawable.ic_dialog_alert)
                .setCancelable(false)
                .show();
    }

}

================================================
================================================
======= FRAGMENT 3: CONTROL WHEELCHAIR =========
================================================
================================================
================================================

package khalil.smartwheelchair;

import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;

import com.google.android.gms.maps.SupportMapFragment;

import java.io.IOException;
import java.io.OutputStream;
import java.util.Set;
import java.util.UUID;

import io.github.controlwear.virtual.joystick.android.JoystickView;

public class Fragment3 extends Fragment {
    private final String DEVICE_ADDRESS = "00:14:03:06:65:3B"; //MAC Address of Bluetooth Module
    private final UUID PORT_UUID = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");
    private Toast mToast;
    private BluetoothDevice device;
    private JoystickView mJoystick;
    private BluetoothSocket socket;
    private OutputStream outputStream;
    private ProgressDialog mProgress;
    private Boolean btControl = false;
```

Fig. 51. Pseudocode For Mobile App & Event Notification Page 25

```
Button forward_btn, forward_left_btn, forward_right_btn,
        reverse_btn, reverse_left_btn, reverse_right_btn,
        bluetooth_connect_btn, toggle_btn;

String command = "S"; //string variable that will store value to be transmitted to the bluetooth module
String lastcommand;

@Override
public void setUserVisibleHint(boolean isVisibleToUser) {
    super.setUserVisibleHint(isVisibleToUser);
    if (isVisibleToUser){
        //connectWifi();
    }
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState){
    View v = inflater.inflate(R.layout.fragment3_layout, container, false);

    return v;
}



@Override
public void onViewCreated(View view, Bundle savedInstanceState){
    //declaration of button variables
    bluetooth_connect_btn = (Button) getActivity().findViewById(R.id.bluetooth_connect_btn);
    bluetooth_connect_btn.setBackgroundResource(R.drawable.selector_bt_connect);
    toggle_btn = (Button) getActivity().findViewById(R.id.toggle_btn);
    toggle_btn.setBackgroundResource(R.drawable.selector_bt_control_off);
    toggle_btn.setAlpha(0f);
    mProgress = new ProgressDialog(getActivity());
    mJoystick = (JoystickView) getActivity().findViewById(R.id.joystick);
    mJoystick.setAlpha(0.1f);


    //OnTouchListener code for the forward button (button long press)


    //Button that connects the device to the bluetooth module when pressed
    bluetooth_connect_btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mProgress.setMessage("Checking bluetooth status...");
            mProgress.show();
            new Thread(new Runnable() {
                @Override
                public void run() {
                    if(BTinit())
                    {
                        BTconnect();
                        //mProgress.hide();
                    }
                }
            }).start();

        }
    });

    mJoystick.setOnMoveListener(new JoystickView.OnMoveListener() {
        @Override
        public void onMove(int angle, int strength) {
            // do whatever you want
            if (btControl) {
                if (strength > 50) {
                    if (45 < angle && angle < 135) {
```

Fig. 52. Pseudocode For Mobile App & Event Notification Page 26

```
                    //FORWARD
                    if (!command.equals("F")) {
                        command = "F";
                        System.out.println("FORWARD");
                        sendBtCommand(command);
                    }
                } else if (angle > 135 && angle < 225) {
                    //LEFT
                    if (!command.equals("L")) {
                        command = "L";
                        System.out.println("LEFT");
                        sendBtCommand(command);
                    }

                } else if (angle > 225 && angle < 315) {
                    //DOWN
                    if (!command.equals("B")) {
                        command = "B";
                        System.out.println("BACK");
                        sendBtCommand(command);
                    }

                } else if (angle > 315 || angle < 45) {
                    //RIGHT
                    if (!command.equals("R")) {
                        command = "R";
                        System.out.println("RIGHT");
                        sendBtCommand(command);
                    }

                }
            } else {
                if (!command.equals("S")) {
                    command = "S";
                    System.out.println("STOP");
                    sendBtCommand(command);
                }
            }
        }
    }
});

toggle_btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(btControl)
        {
            command = "O";
            try
            {
                if(outputStream != null) {
                    outputStream.write(command.getBytes());
                    toggle_btn.setBackgroundResource(R.drawable.selector_bt_control_off);
                    btControl = false;
                }
                else
                    toastMessage("Please pair the device first");
            }
            catch (IOException e)
            {
                e.printStackTrace();
            }
        }
        else {
            command = "I";
            try {
                if (outputStream != null) {
                    outputStream.write(command.getBytes());
```

Fig. 53. Pseudocode For Mobile App & Event Notification Page 27

```
                            toggle_btn.setBackgroundResource(R.drawable.selector_bt_control_on);
                            btControl = true;
                            mJoystick.setAlpha(1f);
                        } else
                            toastMessage("Please pair the device first");
                    } catch (IOException e) {
                        e.printStackTrace();
                    }

                }
            }
        });

    }

    private void toastMessage(final String message){
        getActivity().runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if(mToast!=null)
                    mToast.cancel();
                mToast = Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT);
                mToast.setGravity(Gravity.CENTER_HORIZONTAL,0,430);
                mToast.show();
            }
        });
    }

    //Initializes bluetooth module
    public boolean BTinit()
    {
        boolean found = false;

        BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

        if(bluetoothAdapter == null) //Checks if the device supports bluetooth
        {
            getActivity().runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    mProgress.hide();
                }
            });
            toastMessage("Device doesn't support bluetooth");

        }

        if(!bluetoothAdapter.isEnabled()) //Checks if bluetooth is enabled. If not, the program will ask
permission from the user to enable it
        {
            //mProgress.hide();
            Intent enableAdapter = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableAdapter,0);

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

        Set<BluetoothDevice> bondedDevices = bluetoothAdapter.getBondedDevices();

        if(bondedDevices.isEmpty()) //Checks for paired bluetooth devices
        {
```

Fig. 54. Pseudocode For Mobile App & Event Notification Page 28

```
            toastMessage("Please pair the device first");
        }
        else
        {
            getActivity().runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    mProgress.setMessage("Looking for wheelchair...");
                }
            });
            for(BluetoothDevice iterator : bondedDevices)
            {
                if(iterator.getAddress().equals(DEVICE_ADDRESS))
                {
                    getActivity().runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            mProgress.setMessage("Connecting to bluetooth controller...");
                        }
                    });
                    device = iterator;
                    found = true;
                    break;
                }
            }
        }
        if(!found) {
            toastMessage("Device not found!");
        }
        return found;
    }

    public boolean BTconnect()
    {
        boolean connected = true;

        try
        {
            socket = device.createRfcommSocketToServiceRecord(PORT_UUID); //Creates a socket to handle the
outgoing connection
            socket.connect();
            getActivity().runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    mProgress.hide();
                    toastMessage("Connection to wheelchair successful!");
                    toggle_btn.setAlpha(1f);
                }
            });

        }
        catch(IOException e)
        {
            getActivity().runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    mProgress.hide();
                    toggle_btn.setAlpha(0f);
                    toastMessage("Connection to wheelchair failed, are you within range?");
                }
            });
            e.printStackTrace();
            connected = false;
        }

        if(connected)
        {
            try
```

Fig. 55. Pseudocode For Mobile App & Event Notification Page 29

```
            {
                outputStream = socket.getOutputStream(); //gets the output stream of the socket
            }
            catch(IOException e)
            {
                e.printStackTrace();
            }
        }

        return connected;
    }

    @Override
    public void onStart()
    {
        super.onStart();
    }

    public void sendBtCommand(String command){
        try
        {
            if(outputStream != null)
                outputStream.write(command.getBytes());
            else {
                toastMessage("Please pair the device first");
            }
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }

}


===============================================================================
 FONA SCRIPT CODE (RASPBERRY PI)
===============================================================================

from time import sleep
from decimal import Decimal
from geopy.geocoders import Nominatim
from ubidots import ApiClient
import serial
import time
import threading
import os

geolocator = Nominatim()

GPSON = 'AT+CGPSPWR=1\n'
GPSOFF = 'AT+CGPSPWR=0\n'
CELLON = 'AT+CMGF=1\n'
CELLOFF = 'AT+CMGF=0\n'
GETGPS = 'AT+CGPSINF=2\n'
INITSMS = 'AT+CMGF=1\n'
SENDSMS = 'AT+CMGS="%s"\n'
HOMEDIR = '/home/pi/Desktop/everyone/vals/'
EVENTOCCURRED = 0


def sendSMS(ser, coordinates):
    phoneNum = queryFile("phone.txt")
    sendToFona(INITSMS,'OK')
    print("\nSending SMS...\n")
    sendToFona(SENDSMS % phoneNum, '>') # Initiate the first message
```

Fig. 56. Pseudocode For Mobile App & Event Notification Page 30

```
    if(queryFile("crash.txt")):
        msg = ("There has been an accident at the following location:"
               "\nhttp://maps.google.com/maps?q={},{}\n".format(coordinates[0],coordinates[1]) + chr(26)) #
Send the first message
    else:
        msg = ("HELP button pressed at the following location:"
               "\nhttp://maps.google.com/maps?q={}.{}\n".format(coordinates[0],coordinates[1]) + chr(26))
    sendToFona(msg, 'OK')
    sendToFona(CELLOFF, 'OK')
    if(queryFile("help.txt")):
        resetFile("help.txt")
    print("SMS Sent!\n")

def queryGPS(ser):
    get_coordinates = False
    while(not get_coordinates):
        gps_output = sendToFona(GETGPS, '+CGPSINF:')
        print(gps_output)
        gps_data = gps_output.split(",")
        try:
            gps_list = [gps_data[2],gps_data[4]]
            #gps_list = ["0038.5605","-0121.4231"]
            get_coordinates = True
        except IndexError:
            print("Index error! Trying to get GPS again...")
    return(gps_list)

def httpPOST(ser, long, lat, var_key):
        long = long.replace("-","").replace(".", "")
        lat = lat.replace("-","").replace(".", "")
        val = "1" + long + lat
        coordinate = Decimal(val)
        print(coordinate)
        if(coordinate == 10000000000000000):
            return
        cont_len = len("%d" % coordinate) + 10
        print(cont_len)
        #ser.write('AT+CIPSTART="tcp","things.ubidots.com","80"\n')
        sendToFona('AT+CIPSEND\n', '>')
        response = ser.readline()
        http = ('POST /api/v1.6/variables/{}/values HTTP/1.1\n'.format(var_key)
                 + 'Content-Type: application/json\n'
                 + 'Content-Length: {}\n'.format(cont_len)
                 + 'X-Auth-Token: A1E-03qioilTFhKUVYB2G6nDcXHrVKioHt\n'
                 + 'Host: things.ubidots.com\n'
                 + '\n'
                 + '{{"value":{}}}\n'.format(coordinate)
                 + '\n'
                 + chr(26))
        sendToFona(http, 'SEND OK')
        sendToFona("AT\n",'OK')

        #print("Buffer Issue!")

def flush():
    global ser
    while(ser.inWaiting() > 0):
        print(ser.readline())

def handle(pin):
    GPIO.remove_event_detect(33)
    print("Fall Detected")
    crash = True

def sendToFona(command, expected):
    global ser
    ser.write(command)
    result = ser.readline()
```

Fig. 57. Pseudocode For Mobile App & Event Notification Page 31

```
    print(result)
    while(not (expected in result)):
        result = ser.readline()
        print(result)
        if(("CLOSED" in result) or ("CONNECT FAIL" in result)):
            sendToFona('AT+CIPSHUT\n', 'SHUT OK')
            sendToFona('AT+CIPSTART="tcp","things.ubidots.com","80"\n','CONNECT OK')
            break
    print("Matched expected response: " + result)
    return(result)


def queryFile(filename):
    file = open(os.path.join(HOMEDIR, filename), "r")
    val = int(file.read())
    print("READ VALUE FROM FILE: " + str(val))
    file.close()
    return val


def resetFile(filename):
    file = open(os.path.join(HOMEDIR, filename), "w")
    file.write("0")
    file.close()

def start():
    global ser
    crash = False
    connected = False
    while(not connected):
        print("Looking for FONA...")
        try:
            ser = serial.Serial('/dev/ttyUSB0', 115200, timeout=1)
            connected = True
        except serial.SerialException:
            print("Serial Device Not Found!")
            sleep(2)

    ser.close()
    ser.open()
    ser.write(chr(26))
    sendToFona(GPSON,'OK')
    sendToFona('AT+CSQ\n','OK')

    sendToFona('AT+CIPSHUT\n', 'SHUT OK')
    sleep(1)

    sendToFona('AT+CIPSTART="tcp","things.ubidots.com","80"\n','CONNECT OK')

    long_key = "59e1c4b4c03f976d8614041e"
    lat_key = '59e1c8e6c03f977288035309'
    notSent = True

    print("Getting GPS Coordinates...\n")
    while True:
        coordinates = queryGPS(ser)
        crash = queryFile("crash.txt")
        button = queryFile("help.txt")
        tiltEnabled = queryFile("tilt.txt")
        gpsEnabled = queryFile("gps.txt")

        if(gpsEnabled):
            httpPOST(ser, coordinates[0], coordinates[1], long_key)
        else:
            sleep(1)
        print("Coordinates: {}, {}".format(coordinates[0],coordinates[1]))
        # Get location from coordinates
        crash = queryFile("crash.txt")
```

Fig. 58. Pseudocode For Mobile App & Event Notification Page 32

```
        if ((crash and tiltEnabled) or button):
            print("###############################################")
            print("########### SENDING SMS MESSAGE ###############")
            print ("###############################################")
            sendSMS(ser, coordinates)
            resetFile("crash.txt")
            #break
    ser.close()
start()
```

Fig. 59. Pseudocode For Mobile App & Event Notification Page 33

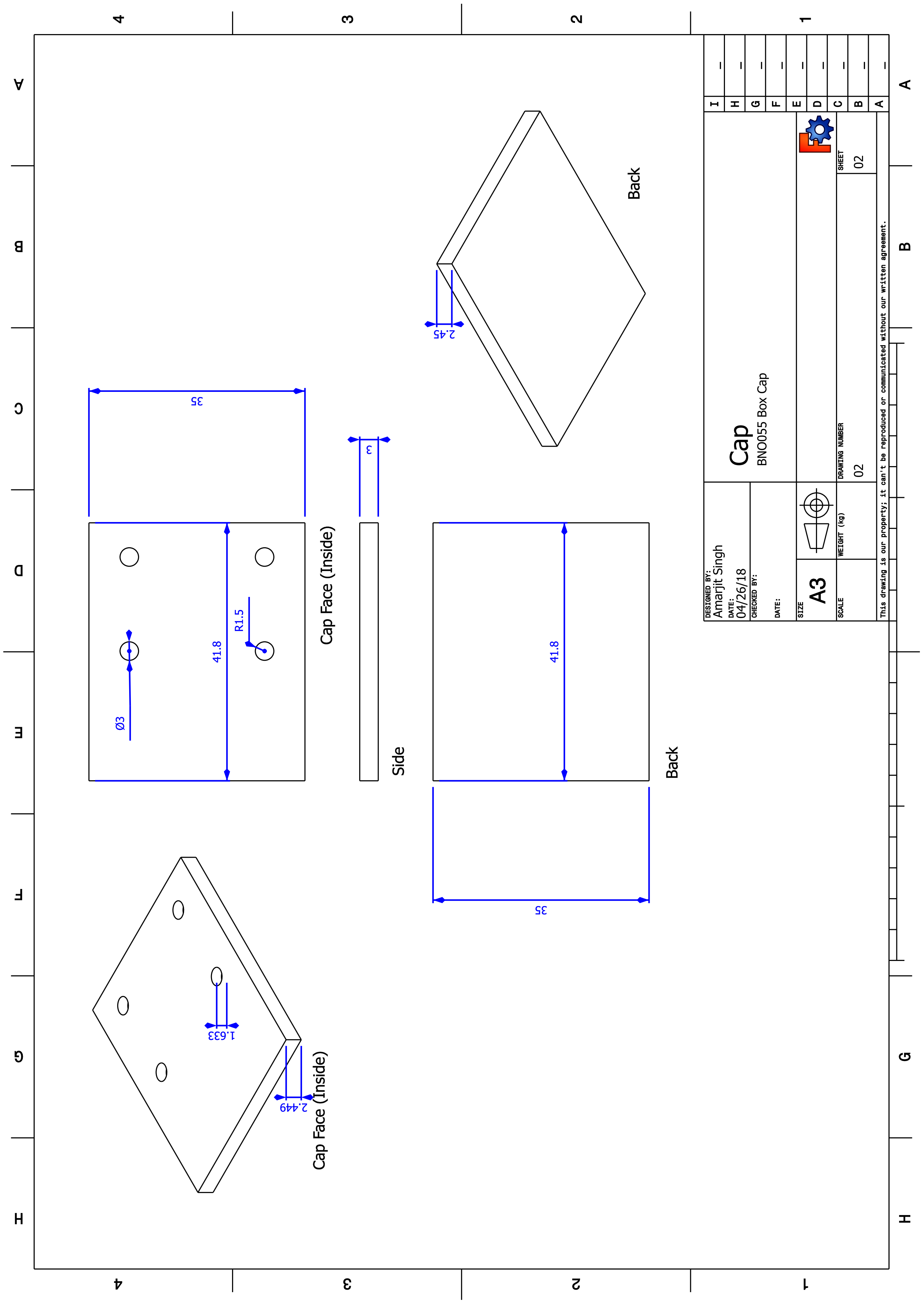


Fig. 60. Mechanical Drawing For Box Cap

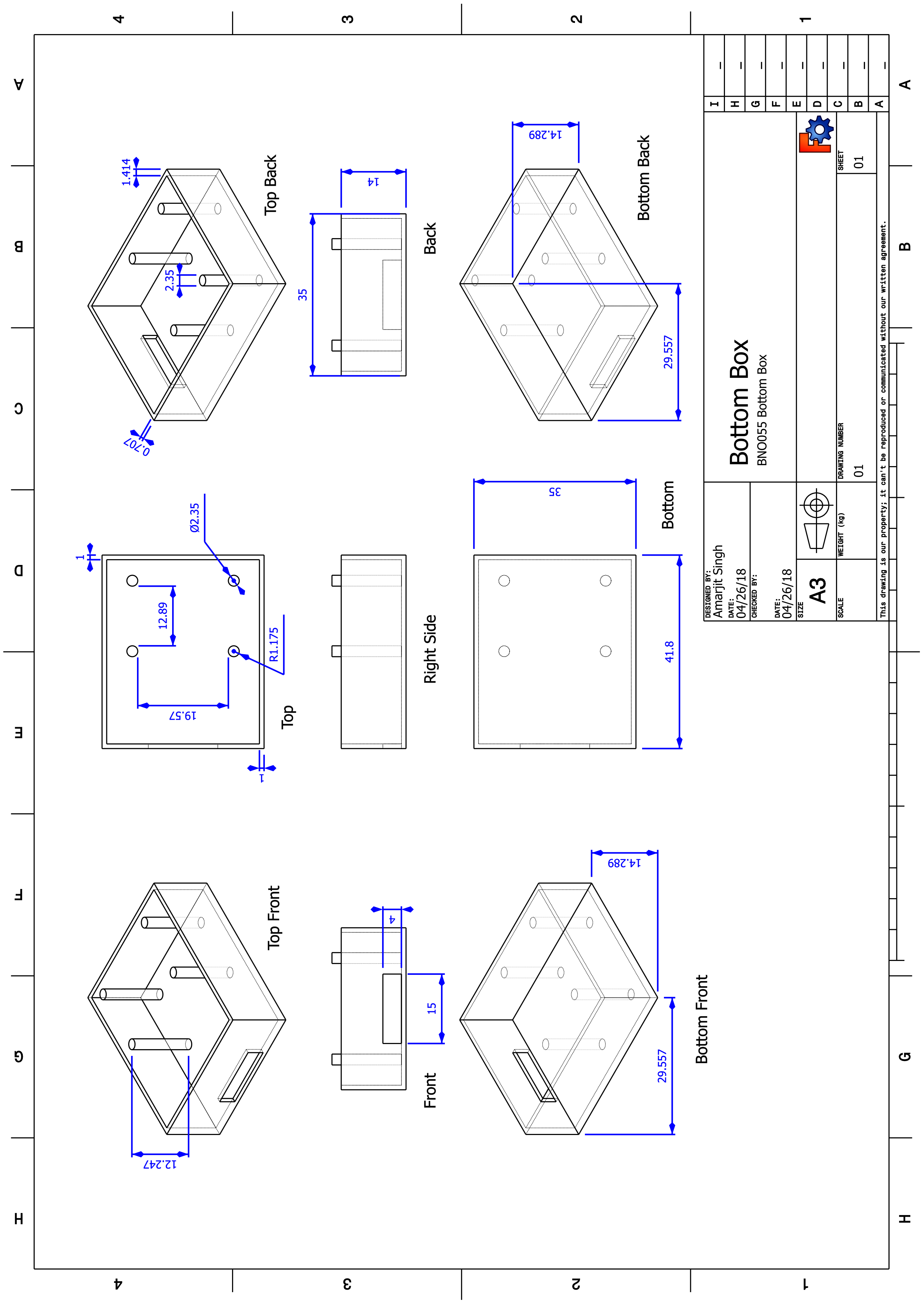


Fig. 61. Mechanical Drawing For Bottom of Box Cap

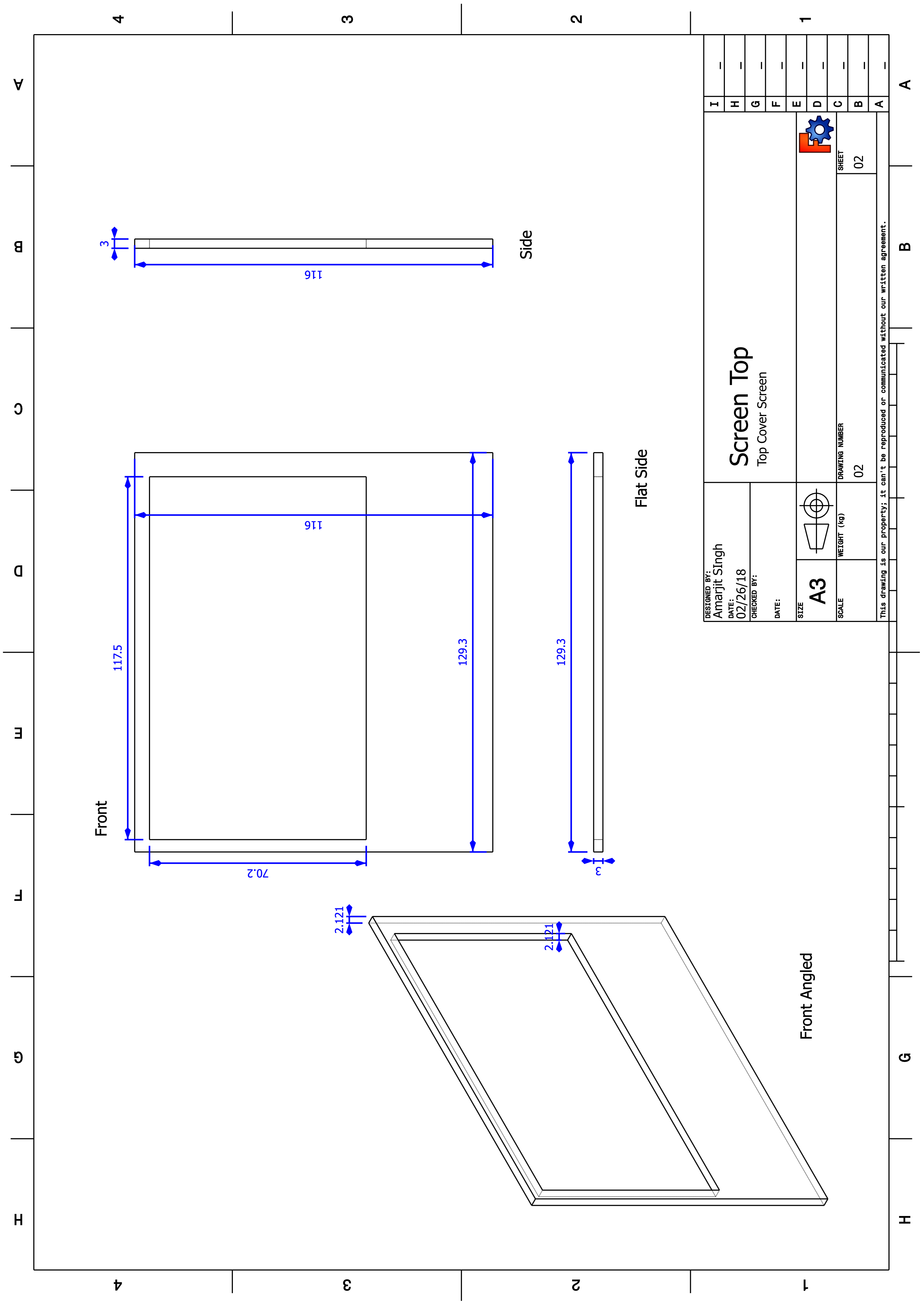


Fig. 62. Mechanical Drawing For Top of Screen Housing

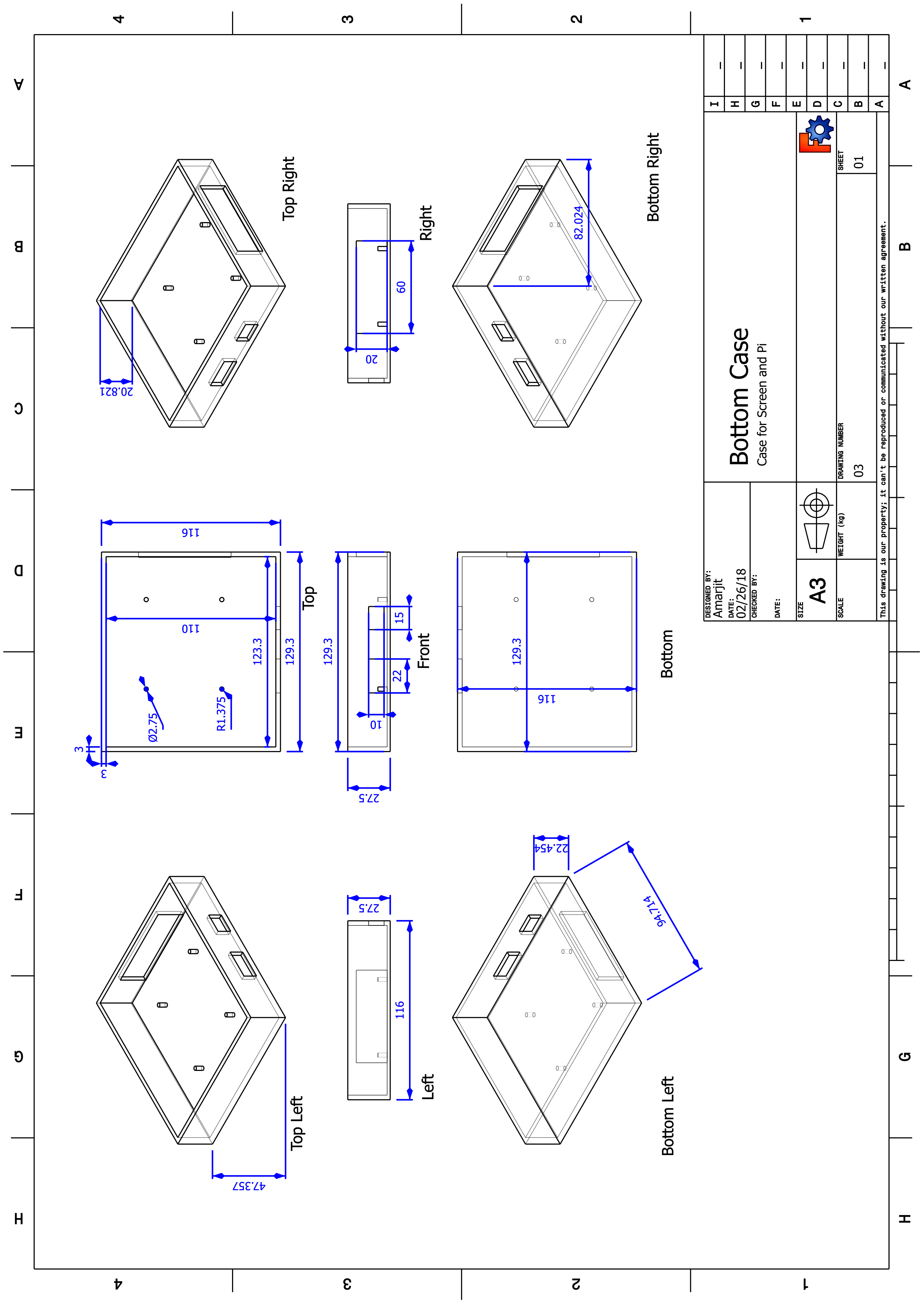


Fig. 63. Mechanical Drawing For Bottom of Screen Housing

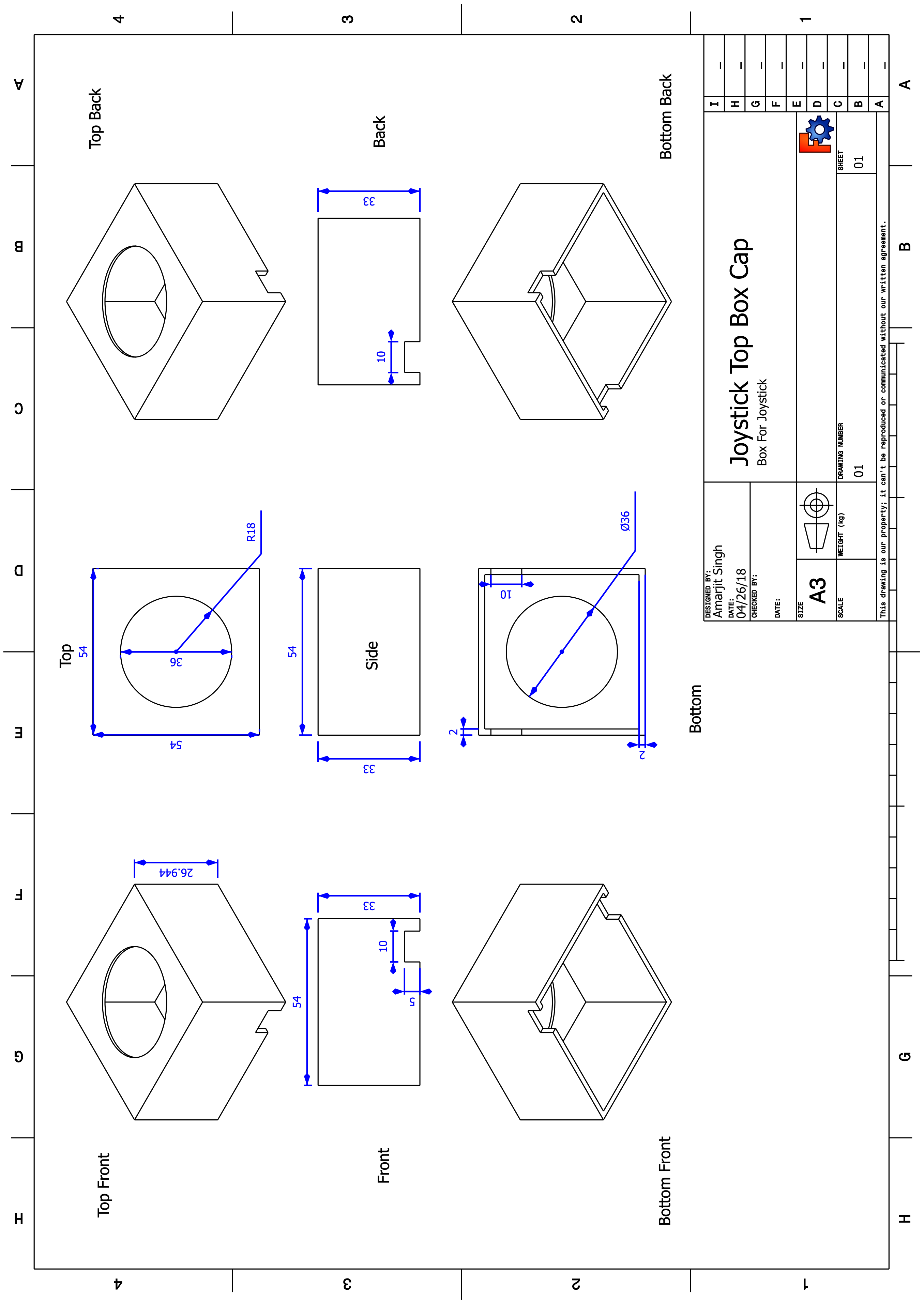


Fig. 64. Top Housing For Joystick

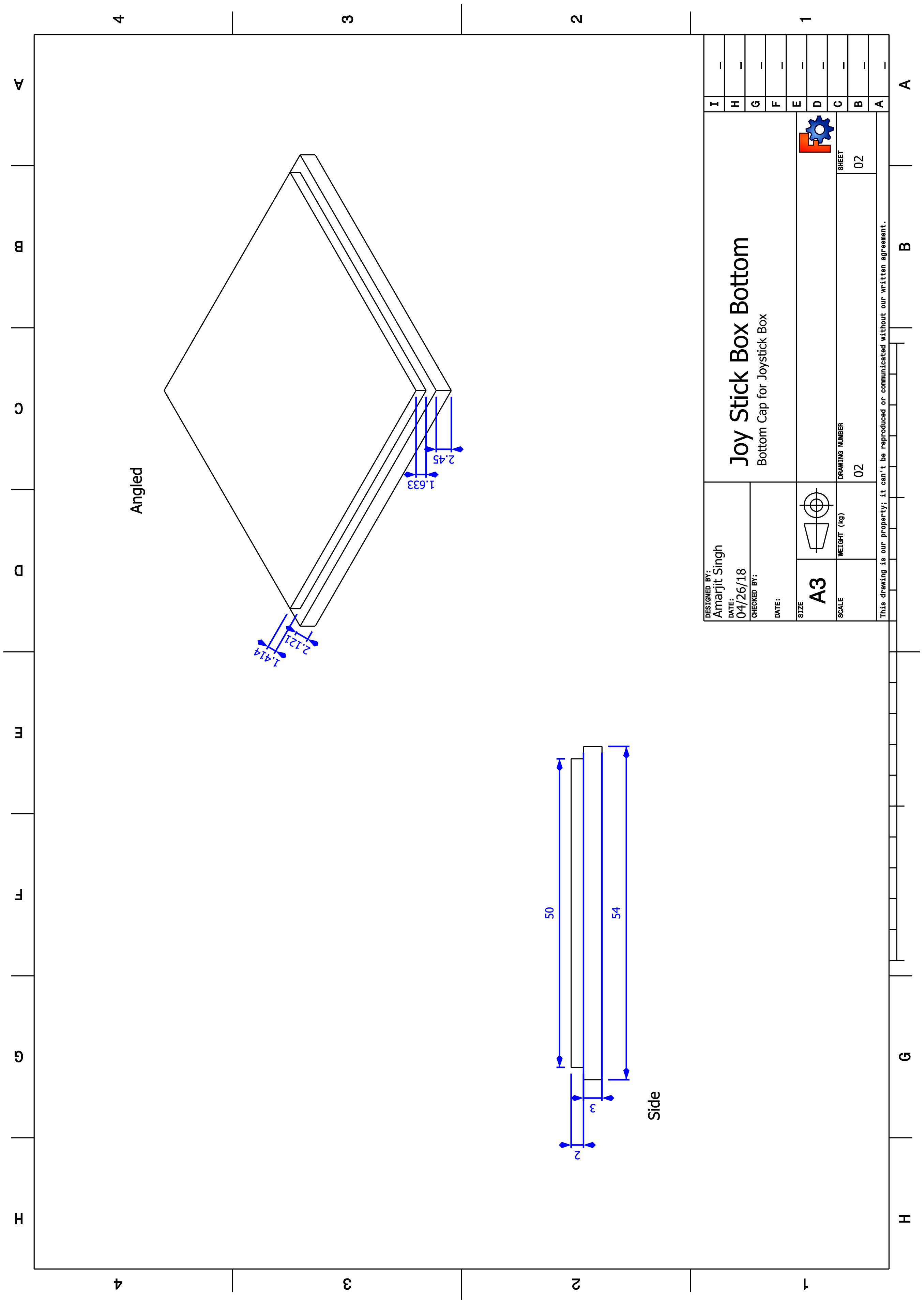


Fig. 65. Mechanical Drawing For Bottom of Screen Housing