Remote Mic Stand for Pogo Studio

Design Review

Zachary Newell – Alexander Lincoln – Tyler Harrington

TA: Justine Fortier

01 October 2012

ECE ILLINOIS

Department of Electrical and Computer Engineering

I.) Introduction

The project was chosen to solve a problem for a local recording studio, Pogo Studio. The motivation for the project is the quality of sound varies with the position and orientation of the microphone (mic). The problem is forgetting the old sound after walking into the next room and repositioning the mic. The wireless mic stand will solve this by enabling the sound recorder to move the mic from the studio wirelessly. It is exciting to work on a project that will directly solve a customer's problem, and also learning that there is a large market for this product.

Objectives

The overall scope of the project is to be able to move a microphone (mic) stand from inside the recording room wirelessly such that sound-checks can be performed efficiently. Adjusting the mic stand wirelessly eliminates the hassle of walking back and forth between rooms to adjust the stand, as well as not forgetting the previous sound quality at the microphone's previous position.

- Functionality
  - Move forward/backwards and left/right
  - Adjust height up/down
  - Adjust pan/tilt
- Goals
  - Perform adjustments and movements wirelessly via smart phone app
  - Store positions as presets
  - Recall preset positions by location and orientation of mic
- Benefits
  - Allows re-positioning of mic stand without leaving the booth
  - Eliminates using an equalizer to "fix" the sound opposed to just moving the mic
- Features
  - Opposed to a robotic arm, the user is not limited to a spherical range of motion; that is, the range of motion is enhance by being able to move the entire mic stand

- The user can save preset locations and positions to be recalled for future use

II.) Design

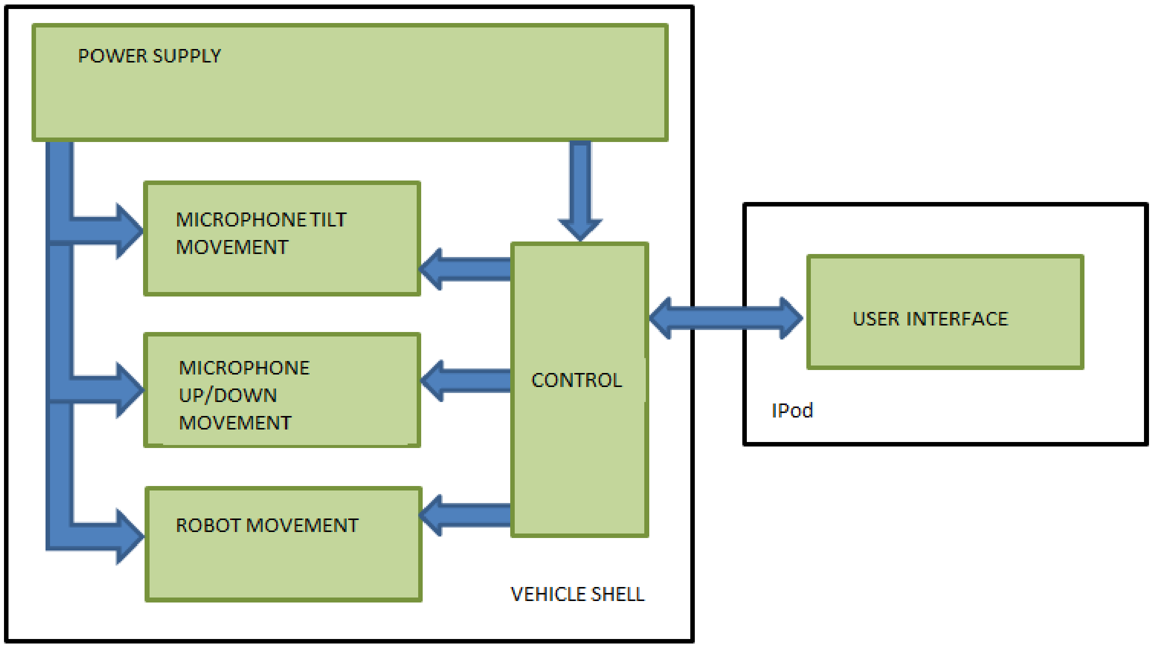

Figure 1 Block Diagram

HardwareOverview

Table 1: Hardware Overview

Component(s)	Function	Block
5 DC Brush Motors	Run 4 wheels and up/down movement	Mic up/down, Robot movement
Servo Motor	Tilt mic stand	Mic Tilt Movement
PIC Microcontroller	"Brains" of Control	Control
Buck Converter	Steps 11.1V to 5V	Power Supply
Linear Regulator	Steps 5V from converter to 3.3V	Power Supply
Motor Drive	Switches provide positive/negative potential across motor for forward/reverse motor operation	Power Supply
Battery	High energy density, but lightweight	Power Supply
WiFly Module	Provides wireless connectivity between mic stand and user	Control
Encoders	Provides feedback on position of motors	Control

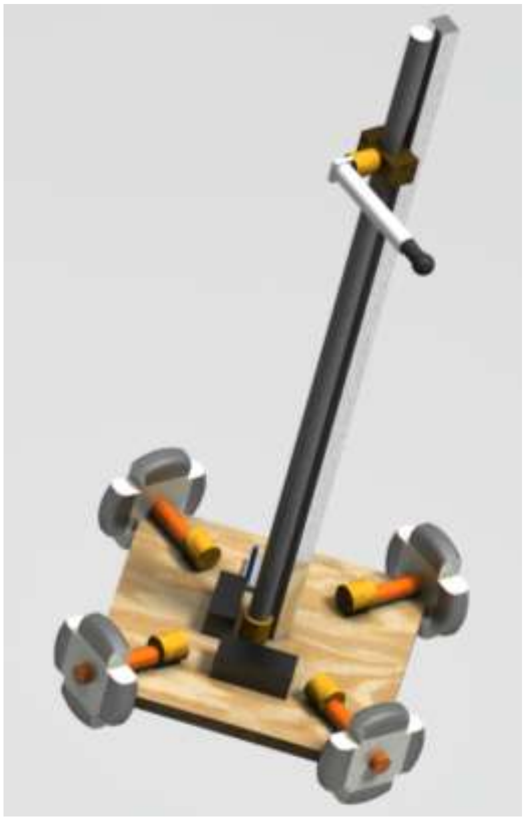

Figure 2: 3D sketch of finished vehicle

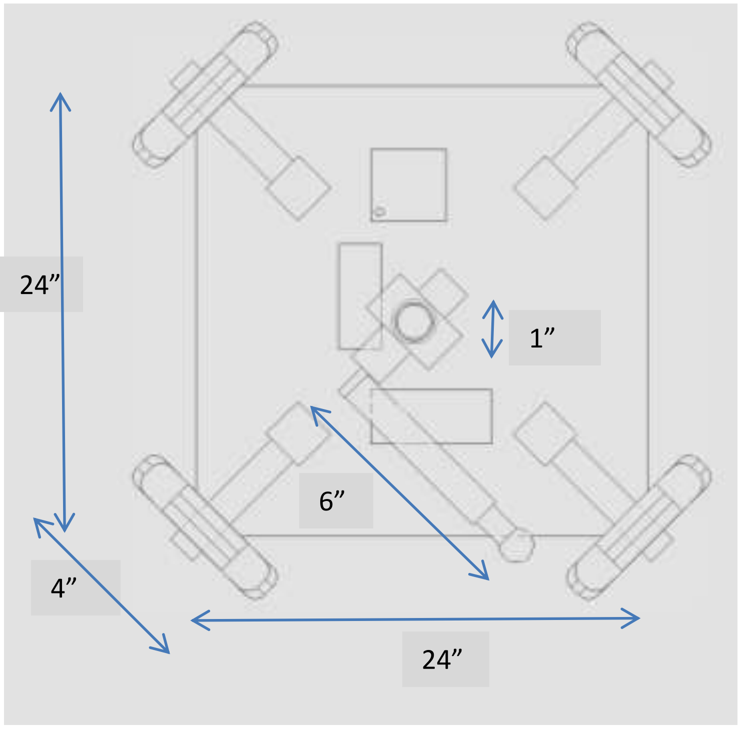

Figure 3: Top view of finished vehicle

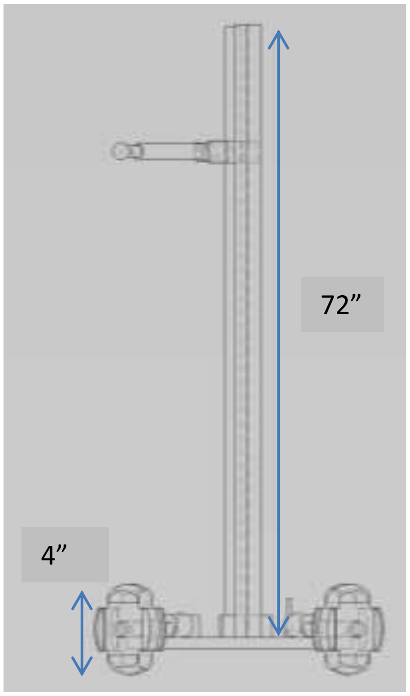

Figure 4: Side-view of finished vehicle

The block diagram in Figure 1is vital to the completion of the project. By separating the device into different blocks, this will enable the progress of the project to be monitored, divide the work amongst the group, and run tests to help determine the functionality of the robot.

Power Supply

An 11.1V 3000mAh LiPo battery will supply the robotic microphone. The 11.1V battery will need to be stepped down to 5V and 3.3V. This battery was chosen due to its high energy density and ability to supply the system well over an hour. The largest load the battery will need to supply are two DC motors running simultaneously. This will be a load of 1.6A; given that the battery can supply 3A continuously for one hour, the battery will have sufficient power to last through a demo.

The five volts will be used to run the servo motor that moves the arm, the PIC microcontroller, encoders, and the h-bridge motor drives. The 3.3V will power the WiFly module. The five DC motors that move the robot will be using the 11.1V from the battery via an h-bridge motor drive. Each wheel(x4) will have its own motor and the fifth motor will be used to raise and lower the stand; note that there will be at most two motors running at one time.

Vehicle Shell

The vehicle shell is defined as the platform or base for which all gear is mounted or originates from. The main platform will be a 0.5 inch thick, 24-by-24”square plywood board. The corners will be chamfered at a 45 degree angle so that there is room for the wheels to be mounted at a 45 degree angle to the edge of the platform. The chamfer will be a 2 inch by 2 inch triangle cut away on each corner. The plywood was chosen because of its durability as well as the cost benefits associated with wood. The wood will be able to bear a large load, much greater than the load of around 35 pounds for the entire device. Attached to the vehicle shell will be the motors and axles for the robot movement, the power supply, control hardware, the shaft and motor for the up/down movement, and ties to keep the wires connecting the system tidy. Also part of the vehicle shell will be two handles attached on opposite sides of the base to enable the user to safely transport the robot.

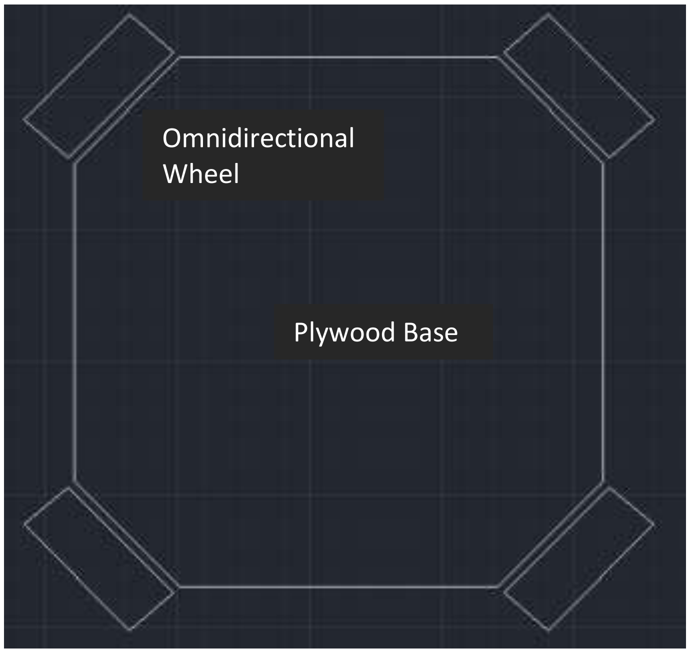

Figure 5: Vehicle Platform

Robot Movement

This block consists of four motors, four axles, four omnidirectional wheels, and mounting gear for the axles. The robotic mic stand will be able to make small adjustments forwards, backwards, left, right, and pan. The wheel configuration shown in Figure 5will allow for these motions without having to raise and lower a pair of axels. Another added aspect of the robot movement is its ability to spin. By turning two opposite wheels in opposite directions, the robot will be able to spin on its axis; this provides the panning movement of the microphone. As shown in Figure 5, the wheels are to be mounted on all four corners of the platform at 45 degree angles so that small movement can be made in any direction. The omnidirectional wheels act as regular wheels, but allow for sliding at a perpendicular angle to the wheels. The omnidirectional wheels contain rollers on the wheel surface to ease the friction and enable smooth movement in any direction. The wheels are 4 inches in diameter and each will be able to support 25 pounds. This will have ample support for the 35 pound device.

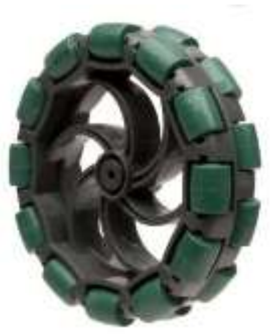

Figure 6: Omni-directional wheel [1]

Figure 6provides a visual of the desired omnidirectional wheels. The motors that will be used for the robot movement will each be 12 VDC, 127.7:1 gear ratio Pittman DC motor. The maximum continuous torque for this motor is 500 oz-in. An axle will be attached to each motor shaft with mounted two-axle brackets attached to the main platform. To limit the torque on the motors, the length of the axels will be minimized. In other words, for a wheel to properly operate, it must bear less than 31.25 lbs (500/16 = 31.25). In most cases two motors will be running simultaneously, therefore they are able to move ~62.50 lbs. This is far more than the estimated weight of 35 pounds.

Up/Down Movement

The parts in this section include a six-foot threaded pipe, a 6-foot aluminum shaft, a 12V DC Pittman motor, and an internally threaded mount. The operation of the up/down movement consists of a motor that will be mounted on the vehicle's platform such that the rotor shaft is perpendicular with the platform. Attached onto the rotor will be a threaded rod and is parallel to an aluminum shaft. A metal mount with internal threads will be screwed onto the threaded rod and attached to the aluminum shaft. The user will then control the height of the mic by turning the motor such that the threaded pipe will spin. This causes the mounted hardware to move vertically along the aluminum shaft. Attached to the mount will be a motor which controls the tilt movement; this can be better visualized by inspecting Figure 2.

Microphone Tilt Movement

The tilt movement will enable the microphone to tilt to the desired angle for recording. The parts necessary for this module is a 343oz-in servo motor, a hollow PVC pipe, and microphone mounting equipment. The servo motor will be mounted onto the movable mount as described in the up/down movement section. The servo motor will control the angle in which the arm is located. The PVC pipe will be five inches long and be used as the arm that holds the microphone. Calculating from the 343oz-in torque requirement, the motor will support a weight of 3 pounds up to 7 inches away; that is, 3*16*7 = 336oz-in of torque.

Control

The control's main component is the PIC Microcontroller (PIC), PIC16F887. It receives data communicated through WiFi from the user on an iOS device (iPhone or iPad). The data received is an opcode. This opcode is parsed into movements that are programmed into the PIC. The movements and their respective opcodes are located in Table 2. Once the movement is known, a signal is output from the PIC to two motor drives. Each motor drive is a DRV8837 by Texas Instruments. It has two inputs, IN1 and IN2; IN1 will drive the motor forwards and IN2 will drive it in reverse. The motor drive is shown in Figure 10; note that each DC motor has its own motor drive. When the motor drive is not receiving a signal (nothing on its input) the motor will not turn.

The component that allows wireless connectivity between the iOS device and the PIC is Roving Network's RN-XV WiFly Module – Wire Antenna (WiFly). This module uses 802.11 b/g protocol and has a built-in wire antenna. This protocol allows connectivity to Pogo Studio's existing WiFi network. The WiFly module is being used strictly as a pass-through device that allows data to be transferred over an existing WiFi network between the PIC and the user's commands on the iOS device.

User Interface

The user interface will be an important aspect of the overall design. Having a logical interface will prevent frustration on part of the end-user. The iPad will provide plenty of screen real-estate. The application will be a tabbed application with one window for movement and a separate window containing a table of saved locations. A rough copy of the user-interface is shown in Figure 7. When a user taps or holds a button, a packet will be instantiated and sent across the TCP connection on the network. The end-user will easily be able to save a location and view/navigate to old locations by clicking the "Saved Locations" tab in the bottom right. The interface will also inform the user that data is being sent to the device. For example, if the user taps the forward button, the button will turn to a green color for the duration of the tap. This interaction is critical for the user to be confident that the operation has occurred.

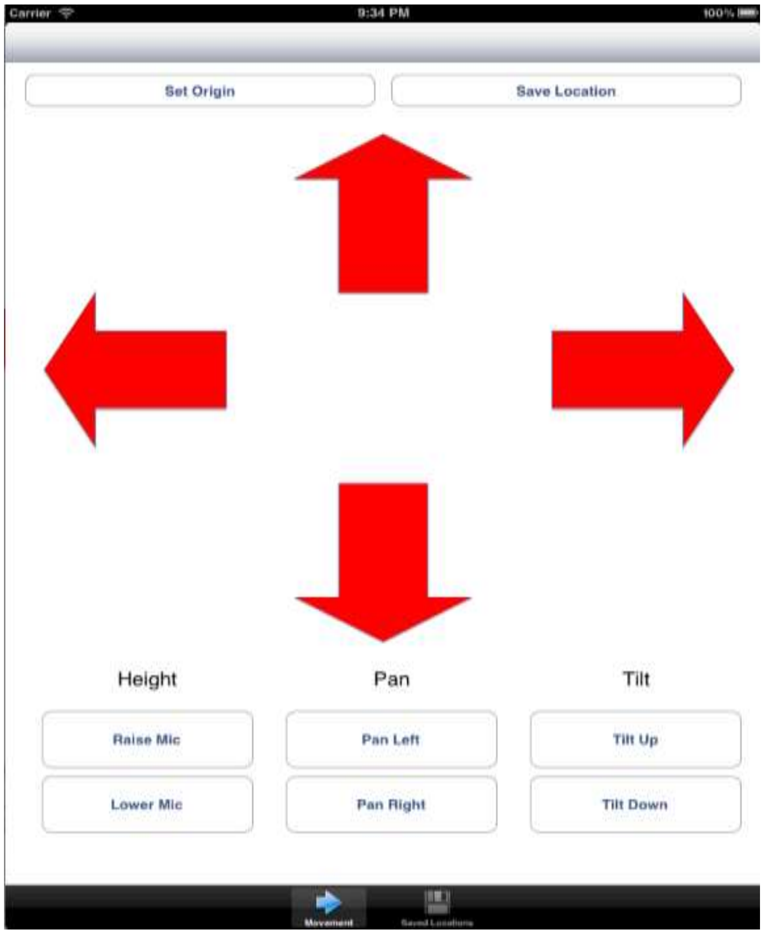

Figure 7: User interface

Schematics

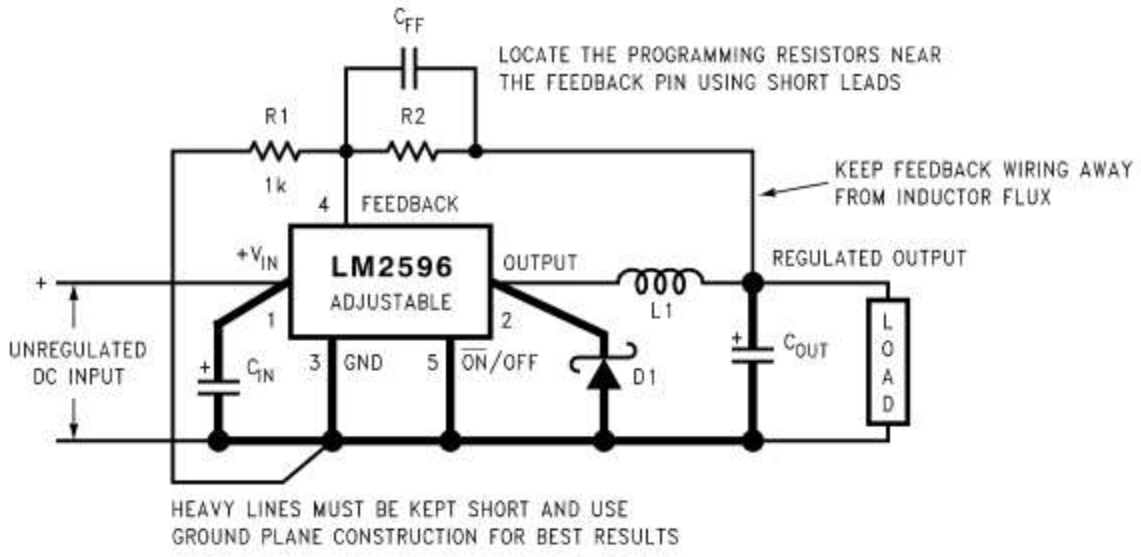
C_FF
LOCATE THE PROGRAMMING RESISTORS NEAR THE FEEDBACK PIN USING SHORT LEADS
R1
R2
1k
4
FEEDBACK
KEEP FEEDBACK WIRING AWAY FROM INDUCTOR FLUX
+V_IN
LM2596
ADJUSTABLE
OUTPUT
REGULATED OUTPUT
L1
1
2
UNREGULATED DC INPUT
C_IN
3
GND
5
ON/OFF
D1
C_OUT
LOAD
HEAVY LINES MUST BE KEPT SHORT AND USE GROUND PLANE CONSTRUCTION FOR BEST RESULTS

Figure 8: 11.1V to 5V buck converter; [2]

V_{in} = 11.1V

V_{out} = 5V

C_{in} - 470µF, 50V, Electrolytic

C_{out} - 220µF, 35V Electrolytic

D1 – 5A, 40V Schottky Rectifier, 1N5825

L1 - 68µH, L38

R1 - 1kΩ

$$R2 = R1\left(\frac{V_{out}}{V_{ref}} - 1\right) = \left(\frac{5}{1.23} - 1\right) = 3k\Omega$$

$$C_{FF} = \frac{1}{31 \times 10^3 \times R_2} = \frac{1}{31 \times 10^3 \times 3 \times 10^3} = 10nF$$

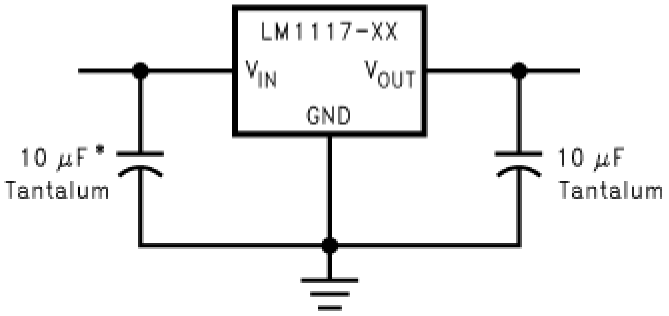

Figure 9: LM1117-3.3 regulator; [3]

V_{in} = 5V from buck converter

V_{out} = 3.3V specified by the “XX” in Figure 2.

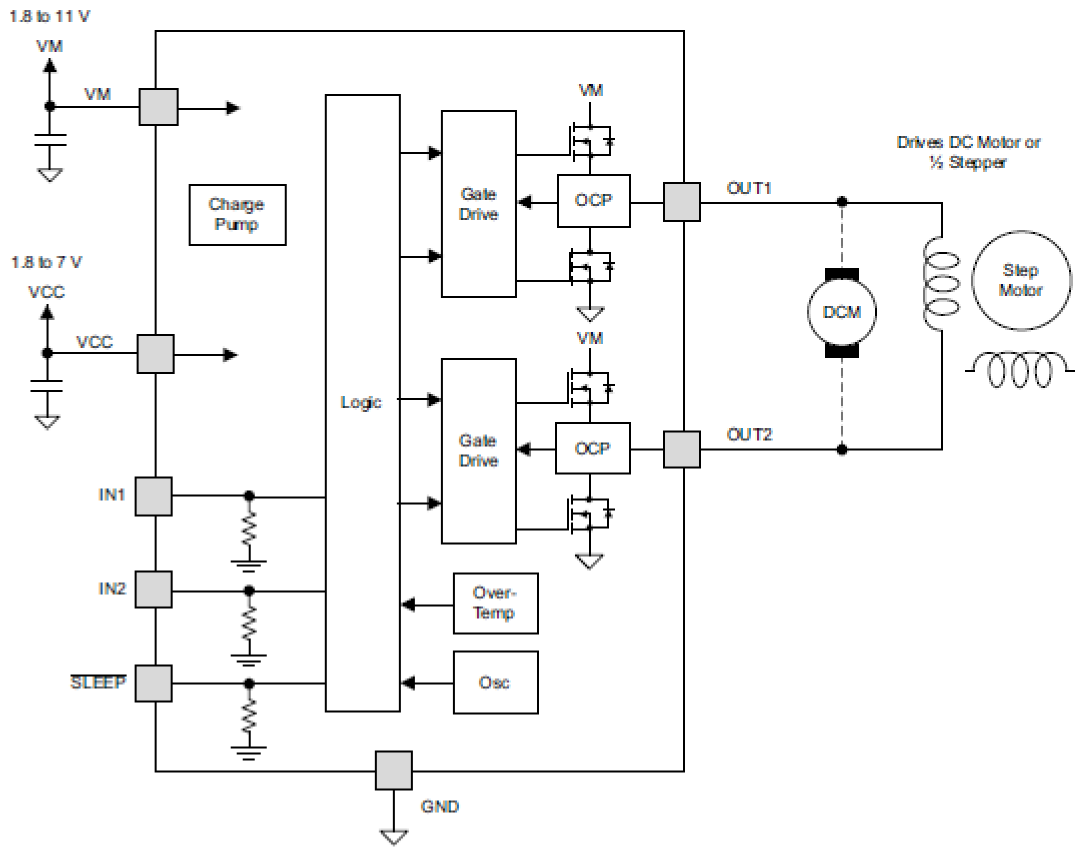

Figure 10: H-bridge motor drive’s block diagram [4]

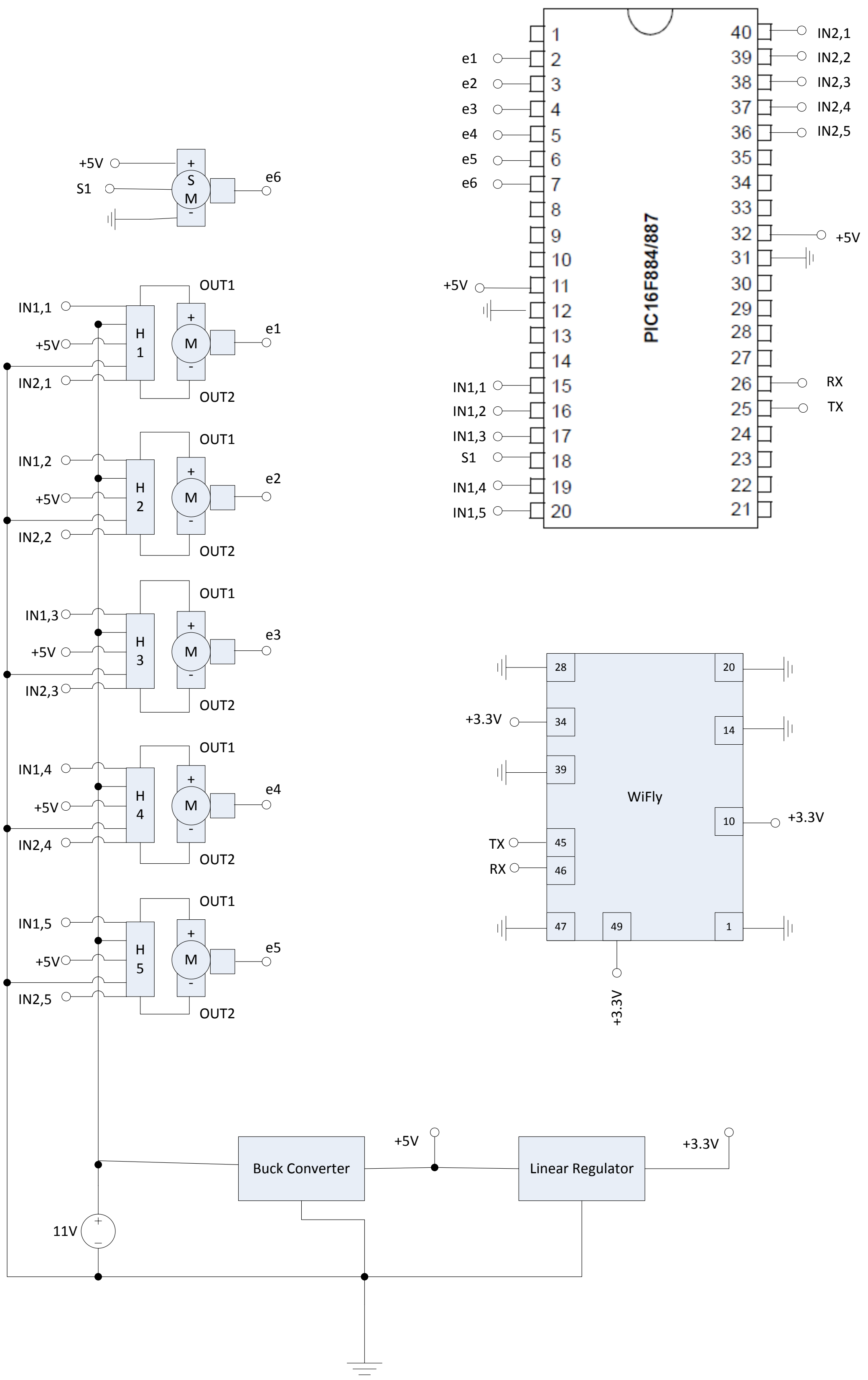

Figure 11: Schematic of all electrical components. Note that the encoders require power and ground but are not displayed. [5]

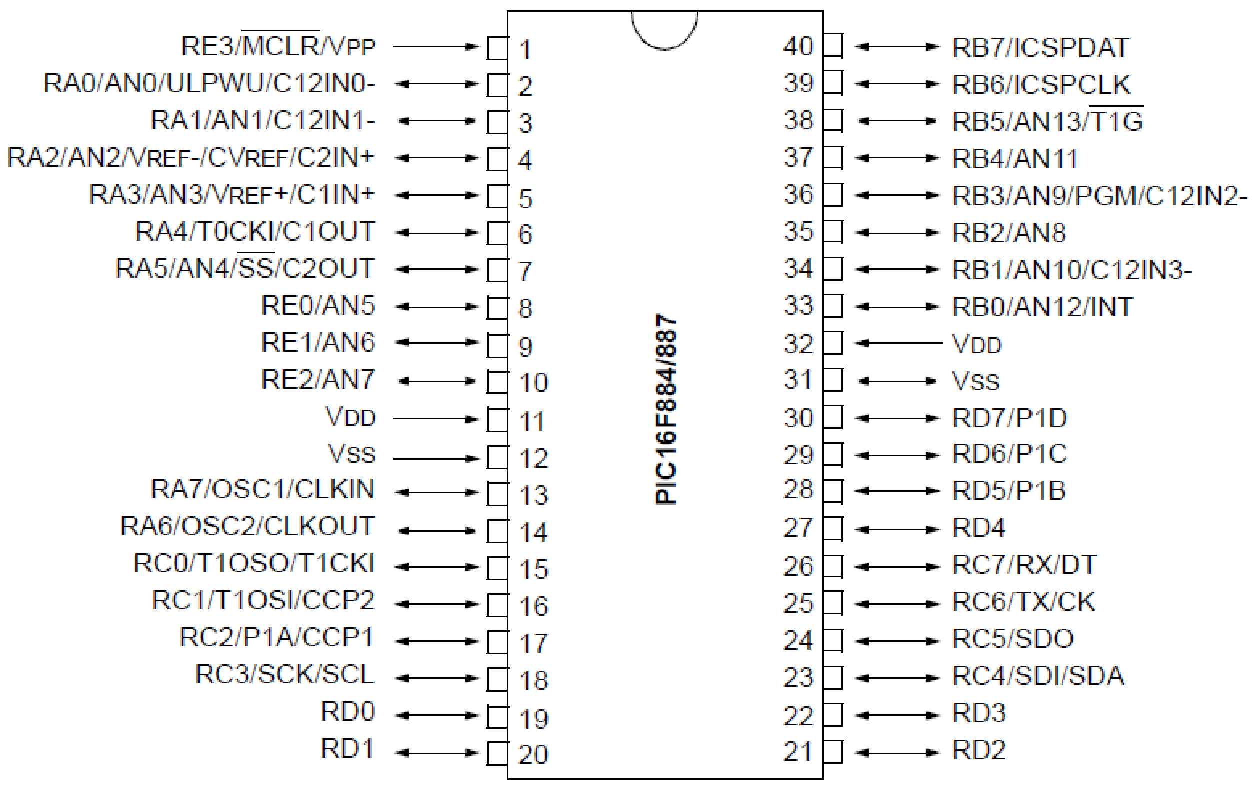

Figure 12: PIC16F887 pinout[5]

III.) Software Design

The software portion of our project can be broken into two logical units: the embedded microcontroller and the iOS client. The iOS client will relay a message through a TCP connection via existing WiFi. The WiFly module receives the packet and passes through to the PIC. Based on this signal, the microcontroller will determine the desired action and control the respective motor appropriately. Figure 13 and Figure 14 demonstrate the iOS client and PIC microcontroller software operations respectively.

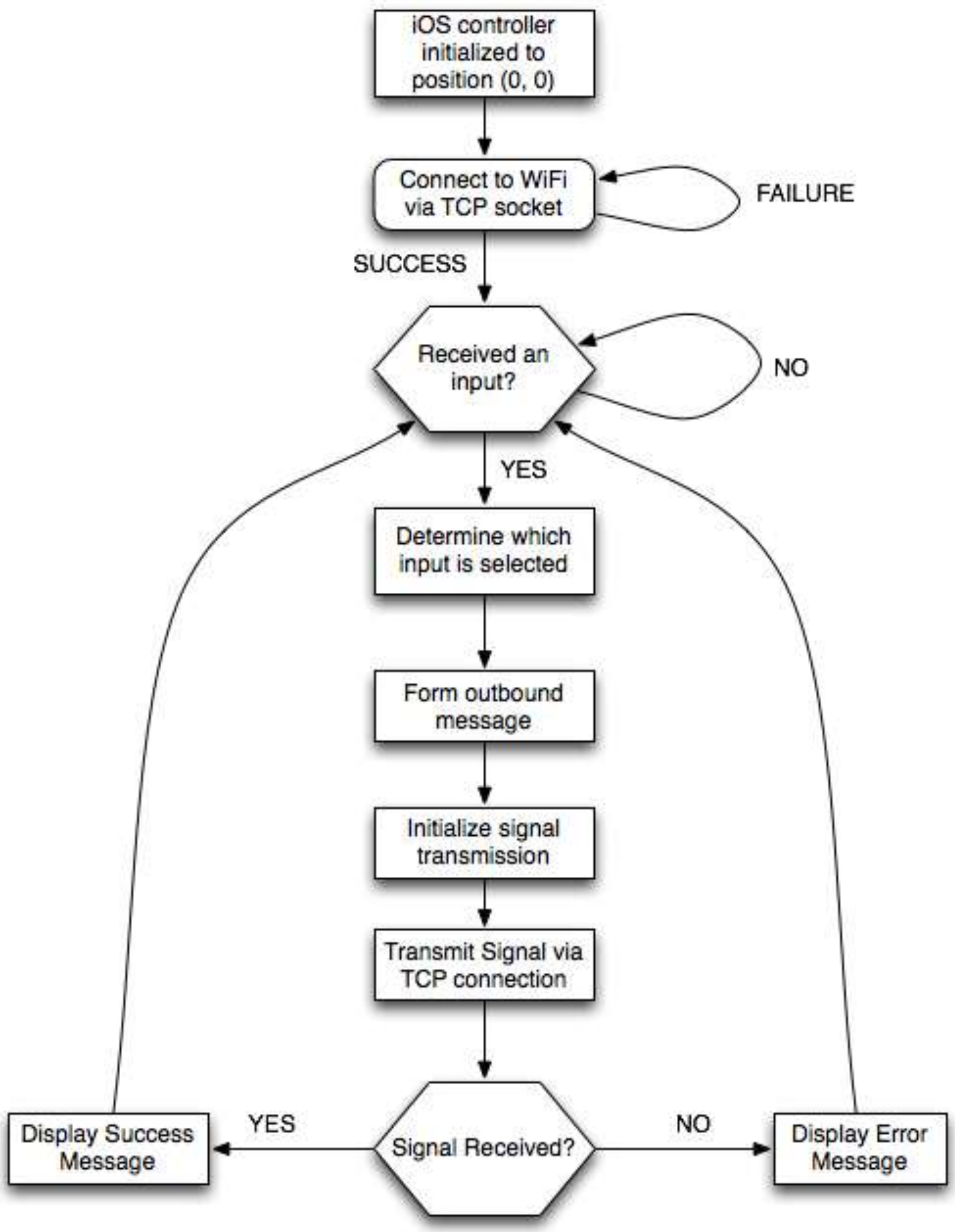
iOS controller initialized to position (0, 0)
Connect to WiFi via TCP socket
FAILURE
SUCCESS
Received an input?
NO
YES
Determine which input is selected
Form outbound message
Initialize signal transmission
Transmit Signal via TCP connection
Signal Received?
YES
NO
Display Success Message
Display Error Message

Figure 13iOS Client flow chart

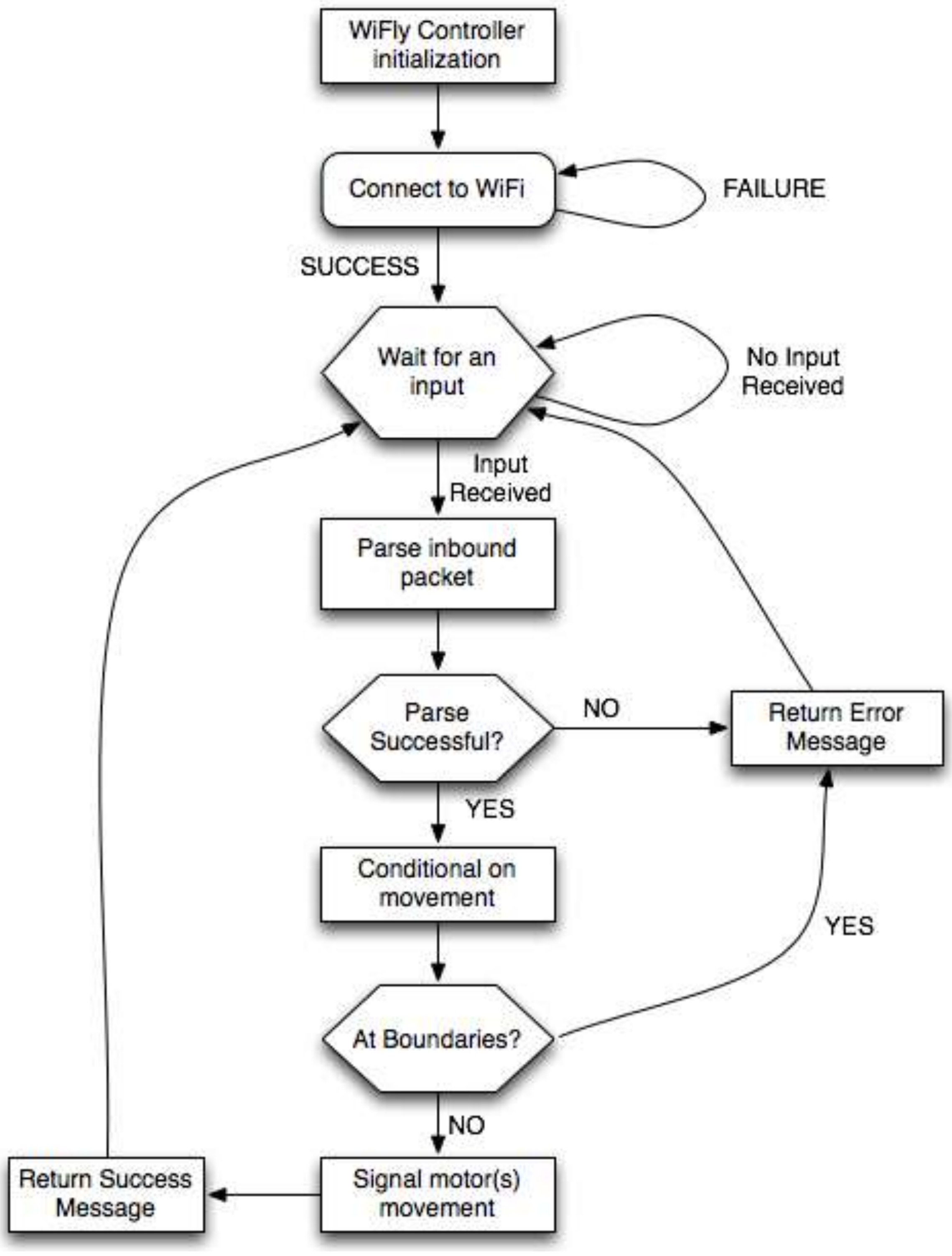

Figure 14: PIC Microcontroller software operations flow chart

Wireless Communication:

As stated, a TCP connection will be made between the iOS device, the Wi-Fi connection, and the WiFly module through a specified port. After the socket connection is created via the iOS device, the WiFly TCP listener will instantiate and begin waiting for a signal.

Movement:

Each press of a movement will generate a specific opcode, i.e. a string, corresponding to a movement. Each received packet will rotate the respective motors a specified distance. For example, each forward motion will rotate the forward motors one revolution thus enabling simplified location tracking. Table 2shows the list of potential movements and corresponding opcodes:

Table 2: Opcodes for movement

Movement	Opcode
Forward	'F'
Backward	'B'
Left	'L'
Right	'R'
Pan Right	'PR'
Pan Left	'PL'
Tilt Up	'TU'
Tilt Down	'TD'
Raise Mic	'RM'
Lower Mic	'LM'
Get Current Location	'GL'

The data will be received as binary. After parsing, the desired movement will be determined and the PIC microcontroller will then control the respective motor for a specified amount of time. This amount of time is determined by how much movement the motor should make per control operation.

Location Tracking:

As requested by Pogo Studio, it would be desirable to remember old locations. This will be done via the iOS device and the Core Data framework. As the user executes a direction, the five-dimensional grid of locations will be update. This grid will be in the format:

$$(x - position, y - position, height, tilt, pan)$$

The positions and movements will be recorded by the number of presses by the user. The number of presses is not a physical count of presses by the user, but by how many times a code is sent from the iOS device. For example, if the user holds the forward button for one second, and there are 10 ‘forward movement’ codes sent per second, this would be recorded as 10 forward movements. If the user would like to navigate to a saved location, it will calculate the distance between the current location and the saved location. Once the differences are calculated, movements will be instantiated. For this to be possible, the iOS client will need to be able to know its current location, which will be determined instantaneously by sending a ‘getCurrentLocation’ packet.

The getCurrentLocation method will send a call to the PIC. The PIC will recall the current number of pulses that each encoder has sent and subsequently send a packet back to the iOS device containing the position of the device in the format shown above.

Movement Boundaries:

The boundaries will be set and controlled via the PIC microcontroller. For example, if the iOS client attempts to raise the microphone beyond the vertical boundary, the PIC will prevent the motion and return an error message to the client. These boundaries will be checked prior to each movement being carried out.

Preset locations:

The preset will be stored by the user using the iOS device. Shown in Figure 7, the button labeled "Save Location" will be pressed to store a preset. The user will be prompted to optionally describe the location such that when attempting to recall a preset location, it is easily distinguishable from other presets. To recall a preset, the user must use the "Saved Locations" page. The saved presets will be listed in chronological order.

IV.) Requirements and Verification

Each component of our project will be developed and tested individually so that full functionality is present. Finally, we will assemble all of the pieces so that our device meets our expectations along with the customer's. Table 2 contains the list which details the testing process we will take.

Table 3: Requirements and verifications

Block	Sub-block	Req #	Requirement	Verification
1.) Power Supply	1.1) Battery	1.1a	Device must run for 1 hour at full load	Run two DC motors (highest current draw) at the same time (~1.6A total) until the battery dies. Record the duration the battery ran and verify it was greater than 1 hour.
	1.2) Buck Converter (LM2956-ADJ)	1.2a	Load voltage is at 5V +-0.2V	Use a voltmeter to measure across output capacitor.
	1.3) Linear Regulator (LM1117-3.3)	1.3a	Load voltage is at 3.3V +- 0.3V with 5V supply from kit	Use a voltmeter to measure voltage across Vout and Ground.
		1.3b	Load voltage is at 3.3V +- 0.3V with 5V supply from LM2596-ADJ	Use a voltmeter to measure voltage across Vout and Ground
2.) Control	2.1) WiFly	2.1a	Power up the WiFly and enable it to be recognized by the iOS device	Use iOS's WiFi radio and scan for WiFly's default SSID.
		2.1b	Receive data from iOS device	Place LED on RX (Pin 45) to verify data is being received.

	2.2) PIC	2.2a	Complete PIC tutorial	Light up an LED within the tutorial.
		2.2b	Receive data from iOS device and output signal from the PIC	Program the PIC to receive a signal from the WiFly module and output a signal to power an LED on the output.
		2.2c	Transmit data from PIC to iOS device	Program a GPIO on the PIC to output data to the WiFly module's TX (Pin 46).
	2.3) H-Bridge (DRV8837)	2.3a	Run motor forward through the H-bridge	Power H-bridge and attach motor. Place a 'high' signal on IN1 and verify motor is turning forward. If unsure of direction, check for +11V across OUT1 and OUT2.
		2.3b	Run motor in reverse through the H-bridge	Power H-bridge and attach motor. Place a 'high' signal on IN2 and verify motor is turning in reverse. If unsure of direction, check for -11V across OUT1 and OUT2.
		2.3c	Use iOS device to run motor forward	Use iOS device to output a forward signal to the WiFly module. Program the PIC to send this signal to IN1 of the H-bridge and verify motor is rotating forward.
		2.3d	Use iOS device to run motor in reverse	Use iOS device to output a reverse signal to the WiFly module. Program the PIC to send this signal to IN2 of the H-bridge and verify motor is rotating in reverse.
	2.4) Encoder (62a22-02-P)	2.4a	Receive pulses from the encoder when the motor turns a known amount (x) of revolutions	Program PIC to count the pulses received. Compare with encoder's specified pulses per revolution (PPR).
		2.4b	Have the motor turn x revolutions using data gathered from the encoder	From 2.4a, use stored count of pulses and output signal to the h-bridge to turn the motor x revolutions

3.) Vehicle Shell	3.1) Weight	3.1a	The platform must be able to support 35lbs of weight	Confirm all joints are still attached and no warping or bending has occurred after leaving 35lbs on the platform overnight
	3.2) Balance	3.2a	The platform must be stable enough to support the arm and servo motor	After the pivoting arm and servo motor is attached to the base, extend the arm at 90° at peak height. Next, push on the top of the device and measure how much force it requires to push over the device using a scale. Verify it does not tip with a force less than 5lbs.
4.) Up/Down Movement	4.1) Up	4.1a	The motor must be able to move up the shaft without the platform rotating	After the wheels and vertical shaft has been mounted, apply 11V across the motor terminals and verify there is no rotation.
		4.1b	The motor must stop before it reaches the top of the rod	1st, program the PIC to receive and count the pulses from the encoder on the vertical motor. Next, place the mount at the lowest point on the vertical shaft. Apply power to the motor moving the mount up the shaft to the top. Once it's at the top, recall the counter's value and program the PIC to never allow up-movement past the newfound distance. Verify this works by attempting to move the mount past the top of the shaft.
		4.1c	Use the iOS device to perform the up movement	Program the PIC to receive the 'RM' opcode from the WiFly module. Next, output the ‘up’ signal to IN1 of the h-bridge drive on the vertical motor. Verify when the up button is pressed by the user, the motor moves the mount up and stops at the top.
	4.2) Down	4.2a	The motor must be able to move down the shaft without the platform rotating	After the wheels and vertical shaft has been mounted, apply -11V across the motor terminals and verify there is no rotation.

		4.2b	The motor must stop before it reaches the bottom of the rod	Using the data gathered in 4.1b, place the motor halfway between the top and bottom of the shaft. Verify this location by moving the (already working) motor up and confirm it takes half the pulses to get to the top. Next Program the PIC to never let the motor lower further than the bottom of the shaft. Verify it will stop rotating when it reaches the bottom.
		4.2c	Use the iOS device to perform the up movement	Program the PIC to receive the 'LM' opcode from the WiFly module. Next, output the 'down' signal to IN2 of the h-bridge drive on the vertical motor. Verify when the down button is pressed by the user, the motor moves down and stops at the bottom.
Robot Movement	5.1) Right/Left	5.1a	The robot must move right/left in a straight line via the iOS device	Program the PIC to receive the 'R' and 'L' opcodes from the WiFly module. Next, program the PIC to output a 'right' signal to IN1 of the two h-bride drives that move the robot right, and output a 'left' signal to IN2 of h-bridge drives to the same motors. Verify this works by lying tracks parallel to the wheels with 1.5" of room on each side to allow for error; confirm by moving the vehicle and it remains in-between the two parallel tracks.

		5.1b	The robot must be able to recall its right/left location from an initial point	Program the PIC to receive and count the pulses from the encoders on the right and left motors. Use the right #pulses as positive integers and left #pulses as negative. When the user saves the location, the integers will be saved in a 2-D array. Verify that the robot returns to the saved location by first marking a reference point on the iOS and on the floor. Next, the user will move from the initial spot and save its location. Next move the robot to a new location. Have the user restore the saved location and verify all four wheels are within a 2" radius of the saved location.
	5.2) Forward/ Reverse	5.2a	The robot must move forward/ backward in a straight line via the iOS device	Program the PIC to receive the 'F' and 'B' opcodes from the WiFly module. Next, program the PIC to output a 'forward' signal to IN1 of the two h-bride drives that move the robot forward, and output a 'reverse' signal to IN2 of the h-bridges that drive the same motors. Verify this works by lying tracks with 1.5" parallel with each wheel and confirm it is within these tracks after the movements have been made.

		5.2b	The robot must be able to recall its right/left location from an initial point	Program the PIC to receive and count the pulses from the encoders on the forward and reverse motors. Use the forward #pulses as positive integers and reverse #pulses as negative. When the user saves the location, the integers will be saved in a 2-D array. Verify that the robot returns to the saved location by first marking a reference point on the iOS and on the floor. Next, the user will move from the initial spot and save its location. Next move the robot to a new location. Have the user restore the saved location and verify all four wheels are within a 2" radius of the saved location.
	5.3) Pan right/left	5.3a	Right/CW	Program the PIC to receive the 'PR' opcode from the WiFly module. Next, program the PIC to output a ‘pan-right’ to IN2 of the front motor and IN1 of the opposite motor. Verify this works by drawing a circle on the floor with a diameter of 27" and confirm the robot is within the circle after 2 revolutions.
		5.3b	Left/CCW	Program the PIC to receive the 'PL' opcode from the WiFly module. Next, program the PIC to output a ‘pan-left’ signal to IN1 of the front motor and IN2 of the opposite motor. Verify this works by drawing a circle on the floor with a diameter of 27" and confirm the robot is within the circle after 2 revolutions.

		5.3c	Recall home location after panning	Program the PIC such that when in "pan mode", the robot cannot leave pan mode until it returns to its home/0° rotations. Verify that the robot returns to pan mode by placing a mark at its home/0° location. Next, pan left/right and try to *move* left/right/ forward/ backward. The robot should recall to its 0° mark before it performs a movement.
	5.4) All movements	5.4a	Perform all left/right/ forwards/ backwards movements and recall them	Place the robot in an initial position and set it as the initial position on the iOS device. Using this device, move the robot left and right, and forwards and backwards with different combinations. Next save the location on the iOS device and physically by marking the wheels. Continue to randomly move the robot and recall the saved location. Verify all the wheels are within a 2" radius of the saved location.
Tilt Movement	6.) Up/down	6.1a	The servo must move via PWM	Program the PWM control to receive a signal from the PIC and output a square wave to the servo motor's 'signal' input. Verify that the arm is moving via the iOS device.

		6.1b	The servo must be able to store and recall its position	Program the PIC to receive and count the pulses from the encoder on the servo motor. Use the 'up' #pulses as positive integers and 'down' #pulses as negative. When the user saves the location, the integers will be saved in a 1-D array. Verify that the robot returns to the saved location by first marking a reference point (parallel to the ground). Next, the user will move from the initial spot and save its location. Next move the robot to a new location. Have the user restore the saved location and verify that the arm has returned to the saved location.
		6.1c	The servo must be able to support 3lbs at 6in.	Measure 3lbs of weight and place on the end of the arm. Verify the servo motor can still rotate.

If every requirement is met in Table 3, the project can be officially labeled as complete.

V.) Cost and Schedule

Table 4: Cost

Part	Quantity	Cost	Total
Pittman 12 VDC Brush Motor - #GM9213C177-R1	5	$158.00	$790.00
343 oz-in Servo Motor #HS-805BB	1	$40.00	$40.00
11.1V LiPo Rechargeable Battery	1	$40.00	$40.00
PIC Microcontroler #PIC16F887	1	$2.50	$2.50
Linear Regulator #LM1117	1	$1.00	$1.00
Buck Converter #LM2596	1	$1.00	$1.00
WiFly	1	$85.00	$85.00
62a22-02-P	6	$45.00	$270.00

Motor Drive #DRV8837	8	$1.75	$14.00
Omnidirectional Wheel	4	$12.50	$50.00
4 ft. x 4 ft. Plywood	1	$3.00	$3.00
Axle	4	$3.00	$12.00
Mounting Screws	40	$0.10	$4.00
6 ft. threaded pole	1	$3.00	$3.00
6 ft. aluminum shaft	1	$3.00	$3.00
Steel Arm Mount	1	$4.00	$4.00
4 in. PVC pipe	1	$0.40	$0.40
Wire	100 ft.	$10.00	$10.00
Labor	3*2.5* 195 hours	$35.00	$51,187.50
Total			$52,520.40

Table 5: Schedule

	Duties			
Week of (Sun):	**Note(s)**	**Zach**	**Alex**	**Tyler**
16-Sep	Proposal Due 9/19 Meet with Mark Rubel	Intro and Schedule. Order Parts .	Design and Requirements/Verifications. Speak with Shop Workers on mech design.	Cost Analysis. Research/ Choose Transceiver.
23-Sep	DR Sign-up Opens	Program PIC to send/ receive data through transceiver	Coordinate wheels (mechanical design)/consult with shop workers for mounting	Program basic app functionality
30-Sep	Design Reviews	Design electrical schematic	Design mechanical schematic. Verify vehicle shell platform is capable of supporting load.	software layout and piece everything together
7-Oct		Coordinate with Tyler on Wifly receive message from iPad. Verify Wifly correctly receives and outputs all signals.	Bring together all hardware. Create AutoCad model of device. Submit layout/plan for shop-workers.	Finish app for Ipad. Coordinate and begin testing transmission signals with Wifly

14-Oct		Test H-bridge functionality in coherence with Tyler. Test all signals and the H-bridge response to those signals.	Receive finished device from machine shop. Perform all physical testing and verification on finished device. Test loads on wheels, platform, mount, shafts, axles. Verify all up to requirements.	Program Pic. Test and verify that all signals are inputted and outputted properly.
21-Oct	Ind. Progress Reports Due 10/24	Progress Report and testing motor functionality. Verify that the H-bridge accurately controls all motors.	Test loads and torque requirements on all motors. Verify every motor meets every requirement.	Finish programming the store location capability for the app.
28-Oct		Continue Testing and begin to introduce store/restore positions logic into PIC. Verify entire control system, using 5 trial frame.	Verify entire control system from the motor standpoint. Verify each motor's response to the input signals. Begin to record the movement that each signal gives to the motors for precision notes	Help to test and verify the entire control system from the application side. Make sure that the signals transmit properly. Verify user friendliness of app.
4-Nov	Mock-up Demos and Presentation Sign-ups	Test and validate position store/restore functionality. Test all 5 trials and verify results.	Test and verify the store/restore results from a motor standpoint. Test and verify all aspects of the power supply.	Test and verify the store/restore functionality of the store/restore functions in the app.
11-Nov	Mock-up Presentations	Mock-up Presentation planning and incorporate store/restore functionality with pan/tilt	Mock-up Presentation planning. Test and verify the pan/tilt functionality.	Mock-up Presentation planning. Completely debug app. Make sure app presentation is useable.
18-Nov	Thanksgiving Break			
25-Nov	Demo and Presentation Sign-ups	Validate proper functionality on all counts of control and power supply and prepare for final demo	Validate proper functionality on all counts of vehicle hardware and motors and prepare for final demo.	Validate proper functionality on all counts of user interface and prepare for final demo
2-Dec	Demos	Play with toy and give demo	Play with toy and give demo	Play with toy and give demo
9-Dec	Presentations, Final Papers Due 12/12	Write control and power supply final paper portion.	Write all movements and vehicle shell final paper portion.	Write user interface final paper portion.

VI.) Ethical Analysis

While creating the robotic mic stand, there are several ethical issues that must be adhered to in order to give the customer, Pogo Studios, the best possible product. One of the main policies from the IEEE Code of Ethics, policy number 7, states “to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others.”[6] This policy stands out particularly because the team members involved with this project has had little experience designing robotic devices in the past; therefore, it is necessary to conduct thorough research into the design of the device. This is a reminder that sources used while developing the design of this device needs to be properly credited throughout the entirety of this project.

Another policy within the IEEE code of ethics that has stood out is policy number 3 which states “to be honest and realistic in stating claims or estimates based on available data.”[6] When the device is presented before peers and the customer, honesty and clarity in the capabilities of the device must be communicated. For example, the team members must be responsible in providing the error associated with the movements and the device’s mechanical limitations.

A moral consideration that may be unique to our project is that the power supply contains a Lithium-Polymer battery. These batteries have very high energy densities and are able to explode if proper care is not taken. The team members must inform the customer of its risks and provide instructions on proper use and handling of the battery.

VII.) References

[1] *4" Omni-Directional Wheel (2-pack)* [Online]. Available: http://www.vexrobotics.com/276-2185.html

[2] *LM2596 SIMPLE SWITCHER Power Converter 150 kHz3A Step-Down Voltage Regulator*, 1st ed., Texas Instruments, Dallas, TX, 2011, pp. 07-14.

[3]*LM1117/LM1117 800mA Low-Dropout Linear Regulator*, 1st ed., National Semiconductor, Dallas, TX, 2004, pp. 02-09.

[4] *Low-Voltage H-Bridge IC*, 1st ed., Texas Instruments, Dallas, TX, 2012, pp. 02-16.

[5] *PIC16F882/883/884/886/887 Data Sheet*, 1st ed., Microchip, Chicago, IL, 2009, pp. 07-14.

[6] *IEEE Code of Ethics,* IEEE Standard 7.8, 2012