REMOTE MIC STAND FOR POGO STUDIO

By

Tyler Harrington

Alex Lincoln

Zach Newell

Final Report for ECE 445, Senior Design,Fall 2012

TA: Justine Fortier

12 December 2012

Project No. 19

Abstract

This report will discuss in detail the design, construction, testing and final result of our Fall 2012 senior design project. The device that was constructed was a remote controlled robotic microphone stand. Our project is capable of moving in ten different directions: forward, reverse, left, right, pan left, pan right, up, down, tilt up, and tilt down. An iOS application is used as the main control interface. The user connects to the device using a wireless internet connection generated by the robot. Another feature available to the user from the iPad app is the use of position recall. The user can choose to store certain positions around the room into the iOS app. This will allow position recall, where the robotic microphone stand is able of automatically returning to any saved position in three dimensional space. The following pages help to further detail our work.

Contents

1. Introduction

1.1 Overview

The goal of this project is to enable Pogo Studio in Champaign to have greater control over the quality and effects of sound in their recording studio. Since the positioning and orientation of the microphone has a great effect of the recording, a wirelessly remote controlled microphone stand will be created that is capable of moving in three dimensions (x, y, z), panning, and tilting.

Furthermore, if a desirable sound is found, the recording engineers at Pogo Studio wish to be able to save the location and orientation of the microphone for later use.

1.2 Objectives

The scope of the project is to be able to move a microphone 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 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 microphone
- Benefits
- Allows re-positioning of the device without leaving the booth
- Eliminates using an equalizer to "fix" the sound opposed to just moving the microphone
- Features
- The range of motion is nearly limitless by being able to move the entire microphone stand in any direction and by any distance, excluding the height limitation
- The user can save preset locations and positions to be recalled for later use

1.3 Block Diagram

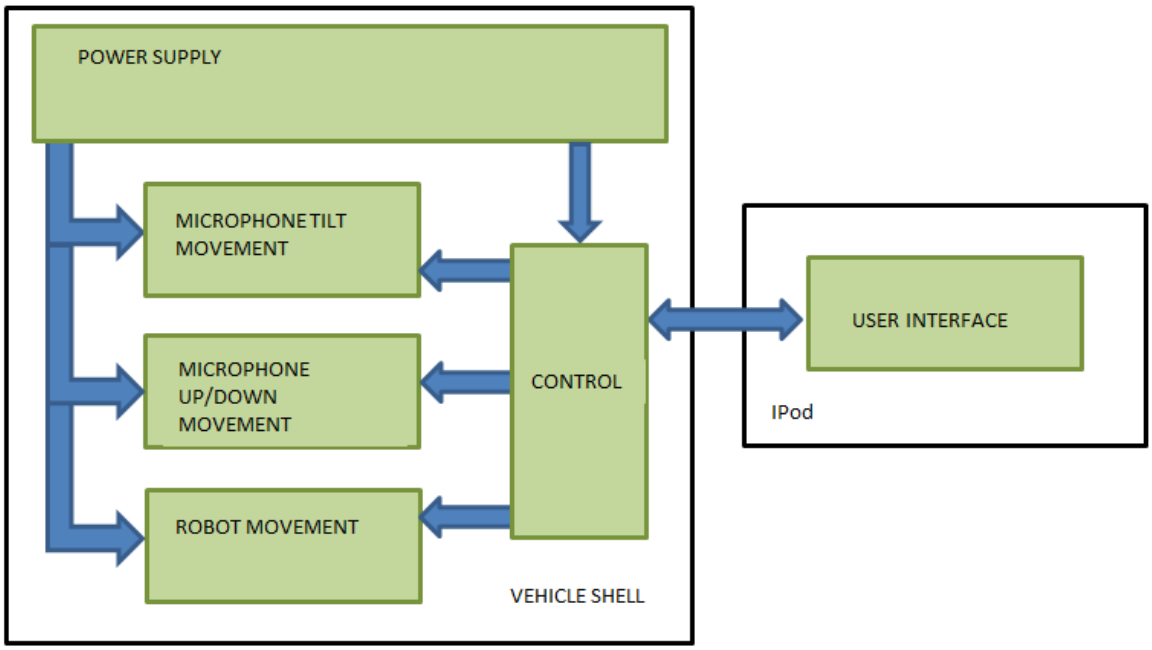

Figure 1 Design Block Diagram

2 Design

As discussed in the introduction, our project can be divided into six main sub blocks. Each component contains a unique function and design process which will be discussed within this section. It is also important to understand how each one of these blocks or pieces interfaces with the others. Figure 2 below illustrates our device using a three dimensional cad drawing so that one can understand how each unit corresponds to the final product.

Figure 2 Device CAD Drawing

2.1 Power Supply

The power supply circuit is an integral part of our design. We wanted to allow the robot to traverse anywhere throughout the room without being limited by a cord or connector. Thus, we chose to use a battery powered device. Another important component of our design process was in determining the different voltage levels needed within the circuit. Due to two different types of motors, as well as the WiFly communication device, we needed to create three different voltage buses: 11.1V, 5V, and 3.3V. In order to implement these voltage levels, we used a rechargeable battery, a buck converter, a linear regulator, and several passive components.

2.1.1 Battery

Within our design, we chose to use 12VDC Pittman brush motors for the majority of the movements. Thus, we needed a battery capable of supplying a voltage relatively close to 12V. After making sure the batteries would turn sufficiently under 11.1V, we decided upon an 11.1V 3000mAh LiPo battery. The battery was chosen due to its high energy density and ability to supply the system for well over an hour. The largest load that the battery would need to supply is four DC motors running simultaneously. This will give a load of 3A even and given that the battery can supply 3A continuously for one hour, the battery will more than be able to handle such power requirements. We were able to connect the battery

leads to a pluggable terminal block that could be quickly and safely removed from the PCB so as to allow for an easy way to kill power to the circuit and charge the battery.

2.1.2 Buck Converter

In order to run the servo motor, the Arduino microcontroller, the encoders, and the h-bridge motor drives, a 5V bus was needed within the circuit. In order to achieve this requirement, a buck converter chip and circuit was created. In order to step the 11.1V down to 5V, the circuit in Figure 3 was created, using the LM2596 buck converter chip. The circuit values are displayed below the figure.

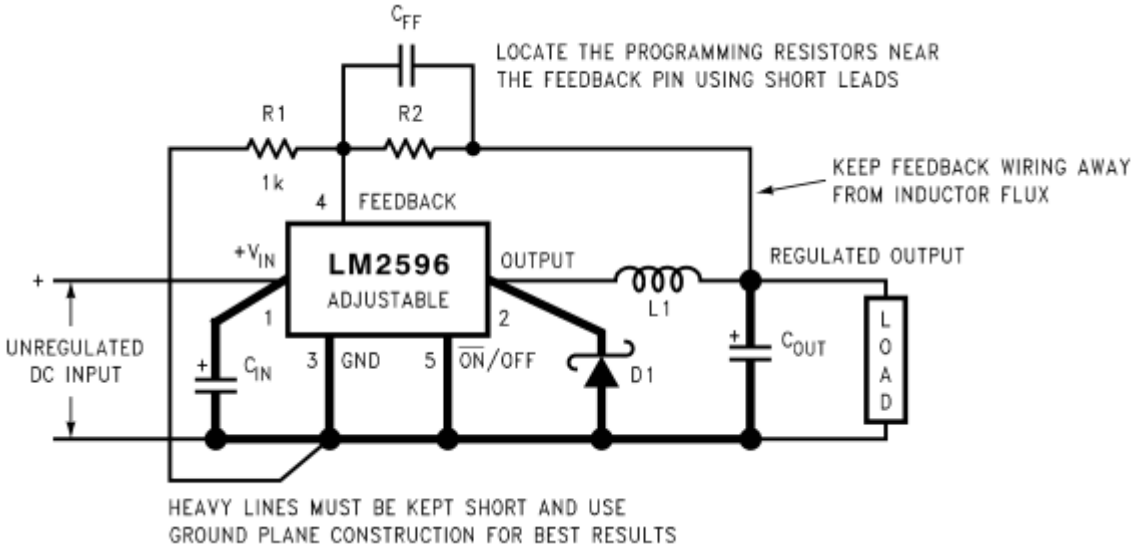

Figure 3 Buck Converter [2]

Table 1 Converter Specifications

V_{in} (V)	V_{out} (V)	C_{in} (uF)	C_{out} (uF)	D1 (A)	L1 (uH)	R1 (kΩ)1
11.1	5	470, Electrolytic	220, Electrolytic	5, Schottky	68	1

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

$$C_{FF} = \frac{1}{31 \times 10^3 \times R_2} = \frac{1}{31 \times 10^3 \times 3 \times 10^3} = 10nF \quad (2)$$

The resistors R1 and R2 were calculated so as to create the 5V on the output capacitor, as displayed in equation 1 above.

2.1.3 Linear Regulator

Finally, in order to achieve a 3.3VDC bus for the WiFly, a linear regulator circuit was used. The LM1117 worked best for our circuit, as it was a small three pin chip capable of the converting the 5V into 3.3V. The circuit used can be seen in Figure 4 below.

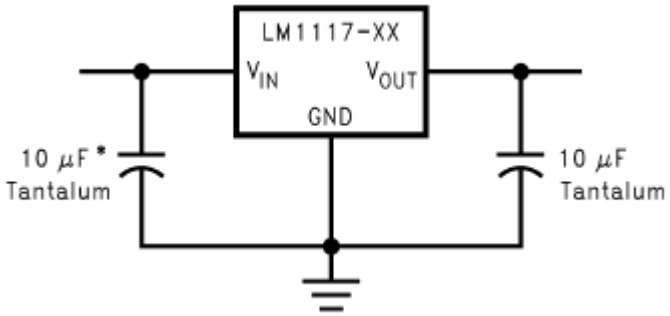

Figure 4 Linear Regulator [3]

2.2 Control

The control unit of the robotic microphone stand entails the WiFly device, the Arduino microcontroller, the h-bridge motor drives, and the encoders. The basic flow of communication within the project starts from the user interface (iPad). Using the adhoc WiFi network created by the WiFly, an opcode is sent to the WiFly. This opcode is then passed along to the microcontroller which parses this code into movements. The microcontroller will then send appropriate signals to the correct motor drives, so as to turn the corresponding motors. When the motor drives receive this input signal, they pass the 11.1V through to the motors. The encoders monitor the rotations of each motor and send the pulse signals back to the microcontroller. The Arduino then receives those signals and passes them on to the iOS device via the WiFly. Figure 5 helps to further illustrate the entire control process.

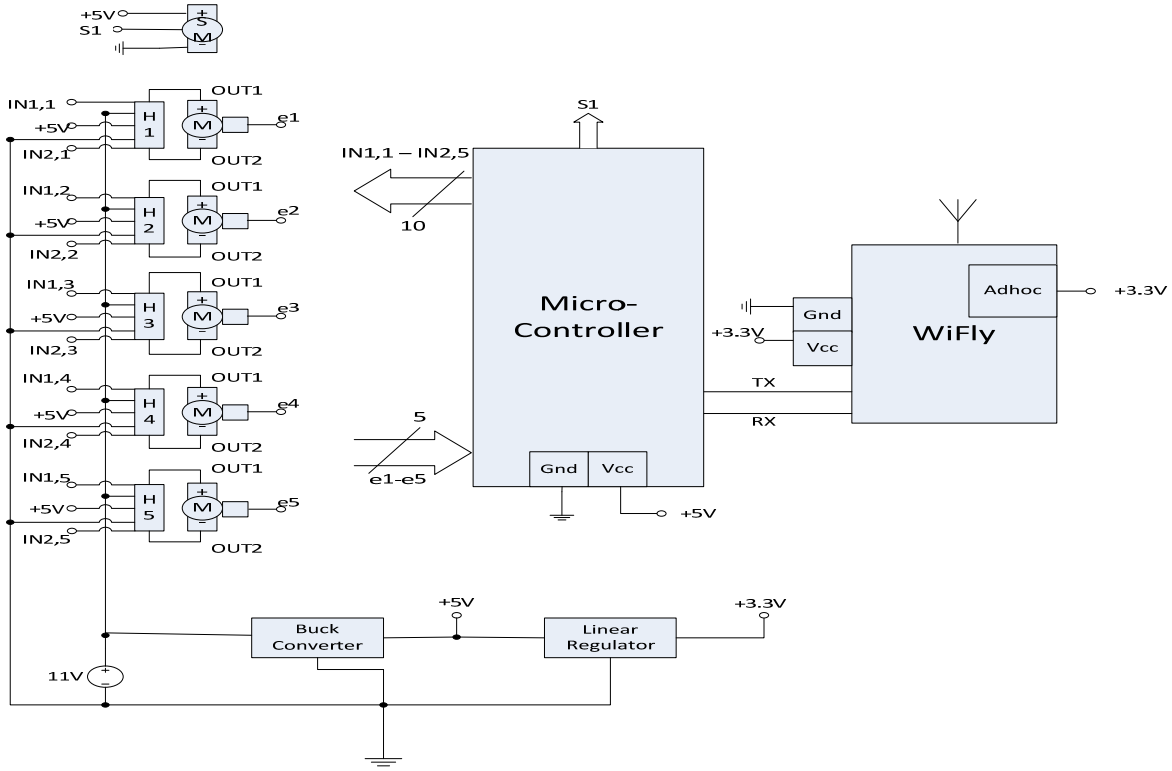

Figure 5 Control Unit [5]

2.2.1 WiFly

The component used to pass data through a wireless network was the Roving Network's RN-XV WiFly Module – Wire Antenna. This module uses 802.11 b/g protocol and has a built in wire antenna. The WiFly module was used to strictly pass data through to the microcontroller from the iOS device. There was no programming or code required within the WiFly. By setting the eighth pin to high when in use, the WiFly creates an adhoc WiFi network that the iOS device is able to connect to. Thus, when using the iPad app, one must first connect to this WiFi network and then begin use.

2.2.2 Microcontroller

The final design required the use of the Arduino microcontroller. The benefits behind using an Arduino as opposed to a PIC microcontroller were that writing code and debugging for errors was made exponentially easier as a group member had previous experience using the Arduino. The Arduino was able to attach to a board to allow for quick programming. The Arduino design also gave us more freedom for use of input pins, as it contained an out for a PWM signal, which the servo motor needs. A design alternative was to use the PIC microcontroller. The PIC would have allowed for the same functionality, although is overall an inefficient design due to the man hours needed to correctly and accurately program and test the chip.

The program code located within the microcontroller was quite simple. The code would basically parse the incoming opcode and use that data to determine which motors to turn on. After the movement

finished, the Arduino would send the updated location back to the iOS device. Figure 6 below helps to illustrate the logic.

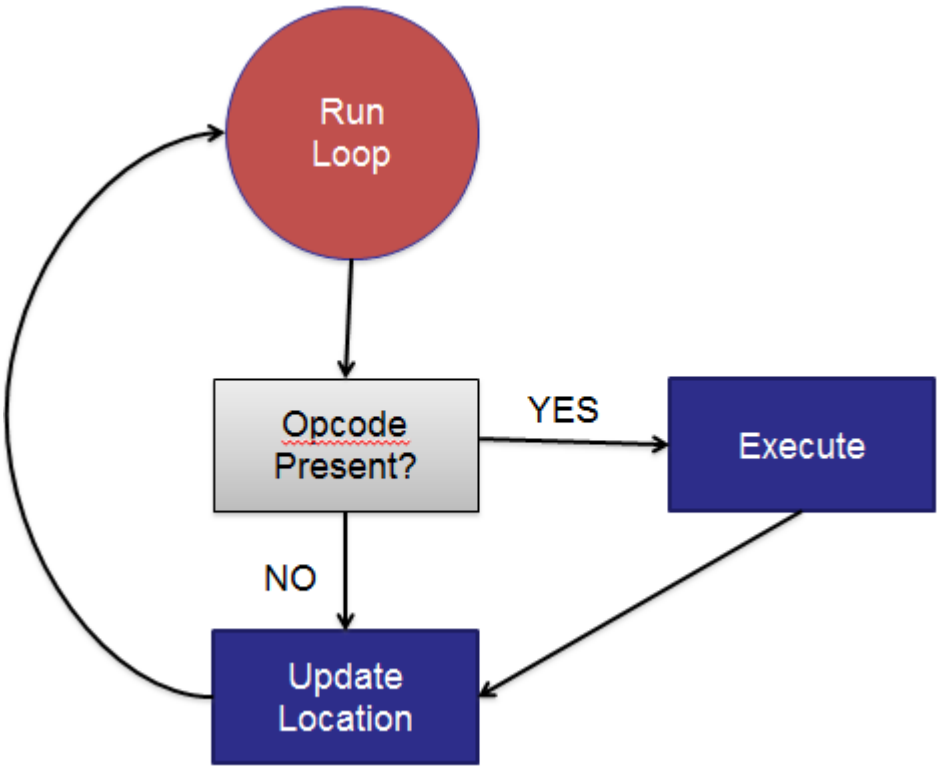

Figure 6 Microcontroller Logic

2.2.3 H-Bridge Motor Drives

The motor drives used on our PCB were the DRV8837 by Texas Instruments. These incredibly small components were surface mount pieces and were chosen due to the fact that they were cheap and functional. Each motor drive was able to take in two inputs. Depending on which input was 5V, the switch opened to pass 11.1V to one of the outputs. Changing which input was high allowed for the reversal of the motor polarities. Motor drives were used for the five 12VDC Pittman motors. The most challenging part in the process was accurately testing and soldering the motor drives due to their small size and requirement that they pass no more than 12V. An alternative design to the motor drives would be to create an H-bridge circuit ourselves using MOSFET components and resistors. The quick reversal of motor polarity is a vital part of our design, so the reliance on working motor drives was heavy. Figure 7 helps to illustrate the H-bridge motor drive circuit in its entirety.

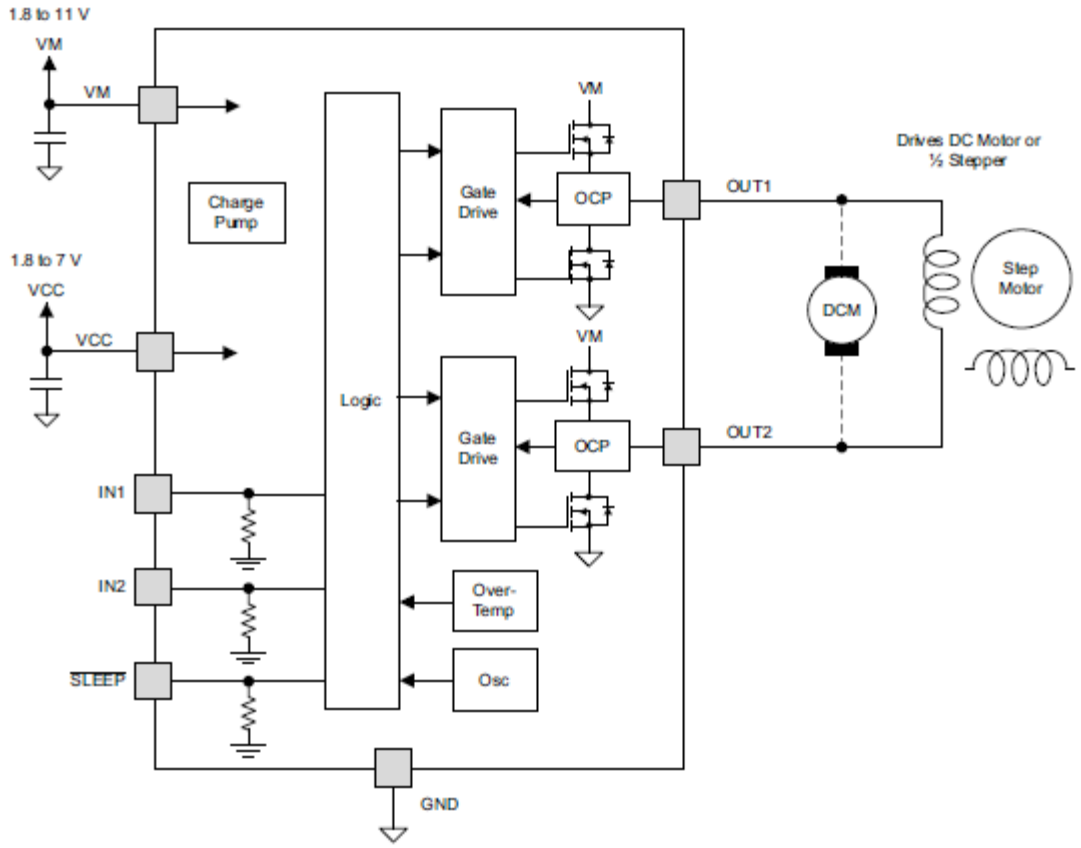

Figure 7 H-bridge Motor Drive [4]

2.2.4 Encoder

The encoders we used in our design were Panasonic EVEG 50mN*m encoders. We used five encoders in our design so as to track the rotations of each of the five 12VDC Pittman motors. These encoders served many purposes in our device in order to accurately keep track of the exact position of every direction of the microphone at all times. One such use of this feature is for the position recall function. The iOS device is capable of storing locations of the microphone so that the user can return to previous setups with one click. Thus, the wave form that each encoder outputs is tracked by the Arduino, sent to the iOS device, and recorded so that the x,y,z coordinates of the microphone are known at all times. Another feature that is allowed with the use of encoders is the ability to stop the device at certain boundaries. The main boundary that we have limited right now is the up/down movement. The microphone needs to stop when it reaches the top of the shaft or the battery will quickly deplete. Thus, the encoder pulses were recorded for the complete span of the shaft. The encoders help to monitor the up/down location so as to know when to stop the motor before the mount reaches the top. Figure 8 illustrates the circuitry required within our circuit for each encoder.

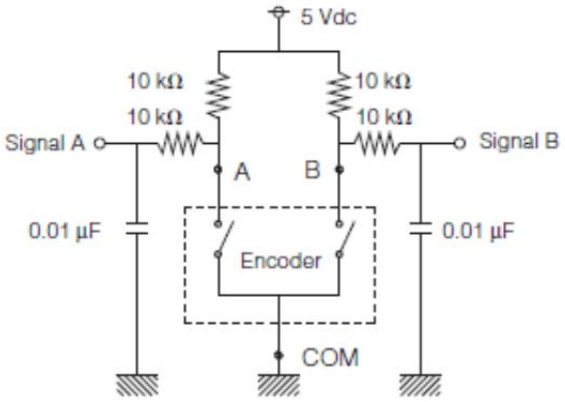

Figure 8 Encoder Circuit

Also, one can see in Figure 9 below the output that each encoder emits while its corresponding motor is rotating. As one can see, the two signals are 90® out of phase. This allows the user to know which direction the encoder is turning. In our design, just knowing the number of pulses was enough, since the iOS device kept track of the direction for us. The iOS device sampled the below pulse every 20 ms so as to make sure to count the total number of peaks. Each peak represents 1/20th of a full revolution.

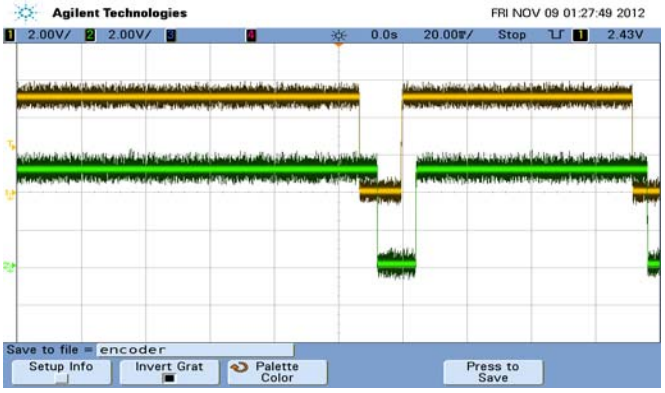

Figure 9 Encoder Output Pulses

2.3 Vehicle Shell

The vehicle shell is defined as the platform or base for which all gear is mounted or originates from. The main platform is a 24”-by-24” metal platform. The corners are chamfered at 45® so that the wheels can be mounted at the horizontals. The base needed to be wide enough to support the potentially top-heavy stand. Mounted onto the platform was a 6’ tall shaft that contains the up/down movement equipment. The PCB and battery were also mounted onto the main platform using stilts and screws and Velcro respectively. A possible design alternative would be to create the platform out of plywood or light plastic instead. This would decrease the overall weight of the device and possibly allow the wheels to turn with less friction. However, implementing this design would cause the possibility of the platform not being strong enough to handle the tall shaft above it.

2.4 Up/Down Movement

The up/down movement consists of a 6.5’ aluminum shaft, a 6’ threaded pipe, an internally threaded metal mount, and a 12VDC Pittman brush motor. The shaft is mounted to the platform in a sturdy fashion so as to be able to withstand the possible torque applied to it. The motor shaft is connected to the threaded pipe. The internally threaded mount is connected to the threaded pipe and slid into the aluminum shaft. Thus, when the motor turns, the threaded pipe rotates and the mount will move up and down the shaft. This design helped to guarantee a straight line movement, as the threaded pipe is locked into place. Using a heavily threaded pipe, we were able to move the motor with exact precision, albeit a slow ascent/decent. An alternative design to this approach would be to create an elbow joint that would be able to move the mic out and up at the same time. The problem with this design is that there would be several instances in which the user wants to move straight up or down, instead of out at the same time. Our current design makes efficient use of the space given as well as allows accurate movement up and down the shaft.

2.5 Robot Movement

The movements entailed in this section consist of right/left, forward/reverse, and pan right/left. The components that help to make this design work consist of four 12VDC Pittman brush motors, four wheel axels, four omnidirectional wheels, and mounting gear for the axels. Using a different motor for each wheel allows for small directional movements, as well as the ability to spin the device on its axis by turning each wheel in the same direction. The wheels will be mounted at 45® angles so as to allow for movements straight sideways. Depending on which motors were turning, the device could move in any possible direction. In order to coordinate the pan, each motor must be turning in the same direction. This will allow the device to pivot around its center. The leads of each motor were connected to the motor drive outputs. Thus, the control section was used to coordinate the robotic movements. The best part of this design is implementing the use of omnidirectional wheels. These wheels bear rollers on the tread so that the device can move with low friction perpendicular to the tread. This is advantageous compared to an alternative design such as that of a typical car. By having four wheels facing the same direction, small left and right movements would prove impossible. The design of a typical car would cause rounded turns and slow sideways movements. Using omnidirectional wheels really added a needed complexity to the design that would be lacking in other designs. Figure 10 contains a picture of an omnidirectional wheel to help illustrate the functionality of such a device.

Figure 10 Omnidirectional Wheel [1]

Many calculations were needed to make sure that the wheels and motors would be able to support the weight and torque of the device during movement. The wheels bore the entire weight of the device, so we needed to be sure our chosen design could handle such a load. The specifications of our chosen wheels remarked that each wheel could hold 25 lbs. Calculation number 3 below helps to illustrate that we were well below the max load, as our device weighed about 35 pounds.

$$Max\ Wheel\ Load = 4\ x\ 25\ lbs. = 100\ lbs. \tag{3}$$

Next, we needed to make sure that we had picked the appropriate sized motors for our design. Again, assuming our device was 35 lbs, the max torque of each motor was 500 oz-in, the axle length was as small as possible, and that the minimum motors turning would be two at a time, we can use the following calculation:

$$2\ motors\ x\ 500\ oz - in\ x\ \left(\frac{1}{16oz}\right) = 62.50\ lbs. \tag{4}$$

Thus, the max weight that our device could be was 62.60 lbs. This shows that our motors would more than be able to handle the torque of moving our device.

2.6 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 343 oz-in servo motor, a metal arm, and microphone mounting equipment. The servo motor will be mounted onto the movable shaft of the up/down movement unit. The servo motor will rotate the metal arm so that the microphone can tilt to all angles from 0® to 180®. The servo motor was chosen so that fine tune adjustments in the tilt angel can be made. By sending a PWM signal from the Arduino to the servo, the device is able to move at corresponding angles. The total number of pulses dictates the degrees that the servo motor needs to rotate. A possible alternative solution to the rotation design would be to use a large DC motor similar to the other motors used in our design. However, it would be very tough to allow for precise movements in the rotation using this method. The torque of the servo motor is important to know when designing this aspect. Assuming that the arm and microphone combination weighs about 3 pounds, and the arm and microphone is about 7 inches in length from the servo, the following calculation helps to prove that the servo can withstand the designated torque:

$$3\ lbs.\ x\ 7\ inches\ x\ 16\ oz = 336\ oz - in\ torque \tag{5}$$

2.7 User Interface

The user interface is an integral part of the overall design. Having a logical, clear interface will prevent frustration on part of the user. The iPad will provide plenty of screen real-estate. The application will consist of a screen to control all movements of the device. Another screen will be able to slide over in order to store or recall certain locations. This design was chosen as the motivator for this project, Mark Rubel, uses an iPad in many parts of his daily activities. This will allow for quick and easy control of the

device, as the iPad will be able to painlessly connect to the WiFly's adhoc network. A screen shot of the user interface can be seen in Figure 11 below. Note that the arrows and buttons will change colors when clicked so that the user knows when they have selected a certain movement.

Figure 11 User Interface

2.8 iOS Client

The iOS client will relay the interface selection through the adhoc connection to the WiFly, which then passes that data onto the Arduino. The iOS software will generate a different opcode with the selection and release of each command. Thus, the microcontroller will know when to turn on and turn off each specified motor.

Location tracking was a large requirement brought to us by Pogo Studio. 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 updated. The grid format will be as follows: (x –position, y –position, height, tilt, pan). The positions and movements will be recorded using the number of pulses sent from the encoders. During recall, the iOS client will send commands for a certain amount of time back to the microcontroller, one direction at a time, so that the device is able to move back to its appropriate position. During this process, the pan is negated first and then retrieved at the end.

Finally, movement boundaries are controlled using the iOS software. By first measuring the number of encoder pulses sent from top to bottom of the up/down movement, we were able to set the full distance that the mount was able to travel. Thus, we were able to detect within the iOS client when the mount was getting close to the edges so that it would stop the mechanism in time.

3. Design Verification

The table of the design and verifications is located in Table 1 of the appendix. The design is split up in the following blocks: Power Supply, Control, Vehicle Shell, Up/Down Movement, Robot Movement, and Tilt Movement. Each block and sub-block's design verifications will be examined.

3.1 Power Supply

The testing procedures and verifications of the 11.1 V LiPo battery, 5 V buck converter, and 3.3 V linear regulator are examined.

3.1.1 Battery

Requirement 1.1a was verified by running two DC motors constantly for an hour and verifying the battery still had greater than 11 V at the terminals

3.1.2 Buck Converter

Requirement 1.2a was verified by measuring the output voltage of the buck converter across its output capacitor yielding 4.98 V.

3.1.3 Linear Regulator

Requirements 1.3a and 1.3b were verified by placing a voltmeter across the linear regulator's output and verifying that the output yielded 3.27 V with 5 V from an independent power supply, and 3.28 V from the output of the buck converter.

3.2 Control

The control consists of the WiFly module, microcontroller, motor drives, and encoders. Their verification procedures will be examined.

3.2.1 WiFly

Requirement 2.1a was verified by supplying the WiFly module with 3.3 V. While the device is powered up, an iPad was used to search available WiFi networks. Once the SSID "WiFly-GSX" was located, this verified the WiFly was broadcasting correctly.

Requirement 2.1b was verified by first establishing an Adhoc network by placing 3.3 V on pad 8. Next, data was sent to the WiFly via Telnet, and the LED was lit when data was sent from a terminal window.

3.2.2 Microcontroller

Requirement 2.2a was verified by setting digital pin 13 to output high with a 1 s delay to pulse an LED in an infinite loop.

Requirement 2.2b was verified by first wiring the microcontroller's RX pin to the WiFly's TX pin. Next, the iPad sent the opcode '2' by pressing the forward button on the iPad App. The microcontroller parsed this opcode and sent a high signal to digital pin 13. This was verified by observing the LED on digital pin 13 was lit after the forward button was pressed on the iPad.

Requirement 2.2c was verified by first wiring the microcontroller's TX pin to the WiFly's RX pin. Next, an encoder was wired onto a proto board and its output was connected to digital pin 13. As the encoder was manually turned, the microcontroller was programmed to not only receive the pulses, but to count the pulses. This was verified by counting 24 pulses with one revolution of the encoder.

3.2.3 H-Bridge Motor Drives

Requirements 2.3a and 2.3c were both verified by pressing forward on the iPad and having digital pin 13 output a high signal to IN1 of a motor drive – yielding the motor to turn clockwise.

Requirements 2.3b and 2.3d were both verified by pressing backward on the iPad and having digital pin 12 output a high signal to IN2 of the same motor drive – yielding the motor to turn counter-clockwise.

3.2.4 Encoder

Requirement 2.4a is synonymous with requirement 2.2c due to the microcontroller needing to be able to send the data to the iPad via the WiFly module. See 2.2c for verification procedure.

Requirement 2.4b was verified by having the microcontroller receive and store the number of pulses generated as the motor turned. It was observed that the motors turning produced 104 pulses from an initial point. Next, the recall function was executed running the motor in reverse using the 104 pulses that were stored. Finally, it was observed that the device was back to the initial point after the recall function was executed.

3.3 Vehicle Shell

The Vehicle Shell consists of the wheels, platform, and shaft. The verifications used to determine its stability are examined.

3.3.1 Weight

Requirement 3.1a was verified by placing a 45lb lead-acid battery on the platform. The device was driven with the battery and it was left on the platform overnight. The following day, there was no damage done to the device.

3.3.2 Balance

Requirement 3.2a was verified by attaching a microphone to the end of a pivoting arm and attaching the opposite end to the servo. The device was never able to be pushed over without using excessive force.

3.4 Up/Down Movement

The verifications of the Up/Down movement of the device is examined.

3.4.1 Up

Requirements 4.1c and 4.1a were both verified when the iPad sent the opcode move the motor up. The microcontroller correctly parsed the opcode and sent a high signal to IN1 of the motor drive responsible for driving the up/down motor. No rotation of the base was observed during execution.

Requirement 4.1b was verified by counting the number of pulses it took for the platform to move from the lowest point on the shaft to the highest. Next the microcontroller was programmed to keep track of the platform's height in a 1-D array. Finally, the platform was moved to an arbitrary location. The platform was then sent to the top. It was observed that the platform stopped before it reached the top.

3.4.2 Down

Requirements 4.2c and 4.2a were both verified when the iPad sent the opcode to move the motor down. The microcontroller correctly parsed the opcode and sent a high signal to IN2 of the motor drive responsible for driving the up/down motor. No rotation of the base was observed during execution.

Requirement 4.2b was verified in the same manner as 4.1b except it was tested as the platform was sent to the bottom of the shaft.

3.5 Robot Movement

The Robot Movement consists of the robot moving on the XY plane, including the robot rotating about its axis to allow the user to pan. The verifications of these movements are examined.

3.5.1 Right/Left

Requirement 5.1a was verified by centering the robot with the tiles in the lab – parallel with motors 1 and 3. Next, the forward movement was executed on the iPad. The microcontroller output a CW signal to motor 1 and a CCW signal to motor 3. This caused the device to move in the forward direction. It was observed the motors turned at the same speed causing the robot to move in a straight line using the tiles as a reference.

Requirement 5.1b was verified by aligning the robot in the same manner as requirement 5.1a. Next, the robot was moved to an arbitrary location on said plane. The recall function was then executed by the iPad for the device to return to the initial location. It was observed that the robot returned to the initial point using the tiles as reference. The number of pulses output to the iPad GUI also verified that the device has returned to its initial location.

3.5.2 Forward/Reverse

Requirement 5.2a was verified in the same manner as 5.1a except the device used motors 2 and 4.

Requirement 5.2b was verified in the same manner as 5.1b except the device used motors 2 and 4.

3.5.3 Pan Right/Left

Requirement 5.3a was verified by programming the microcontroller to send a high signal to IN1 of motors 1 through 4 when pan right is executed on the iPad. This turns all 4 motors clockwise causing the robot to rotate about its axis, yielding the panning effect. The robot was observed to stay on its axis while rotating.

Requirement 5.3b was verified in the same manner as 5.3a except the high signal was sent to IN2 of the four motors yielding the panning effect in the counter-clockwise direction.

Requirement 5.3c was verified by first programming the microcontroller to “un-pan” if the device is currently panned and a forward/reverse/left/right movement is received from the iPad, and to return back to its pan after the forward/reverse/left/right movement is executed. Next, the robot panned was panned right (clockwise). Finally, a forward movement was executed. It was observed that the robot reversed the pan, moved forward, and panned right again - to its initial state.

3.5.4 Recall XY Movements

Requirement 5.4 was verified by first establishing an initial point and saving it. Next, the device was moved forward for 305 pulses and right for 260. The location was saved on the iPad. Then, the device was moved forward again for 112 pulses and to the left 207 pulses. Finally, the iPad recalled the location at (260, 305). The robot was observed to move in reverse 112 pulses and to the right 207 pulses. Also using the tiles as reference, the robot had recalled to the correct location.

3.6 Tilt Movement

The verification of the tilt movement supplied by the servo motor is examined

3.6.1 Up/Down

Requirement 6.1a is verified by sending a pulse length of xxx which equates to 45°. It was observed that the arm as moved halfway between 0° and 90°.

Requirement 6.1b was verified by first saving the length of the pulse for 45° . Next, after moving the arm to an arbitrary location, the position is recalled on the iPad. It was observed that the arm had returned to the 45° mark.

Requirement 6.1c was verified by attaching the microphone to the end of the 6in. arm provided by the machine shop. The iPad moved the microphone to 90° and it was observed that the servo supported the weight of the microphone.

4. Costs

By keeping a detailed account of all the parts and money that we spent throughout the project, we were able to create a suitable bill of materials. Not only is this an important part of the project and design, but it allows us to determine if this would be a marketable project. The total sum of all parts and labor can be noted in Table 2 below.

Table 2 Cost of Labor and Parts

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
Arduino Uno	1	$23.00	$23.00
Linear Regulator #LM1117	1	$1.00	$1.00
Buck Converter #LM2596	1	$1.00	$1.00
Encoder #EVE-GA1F2024B	6	$1.46	$8.76
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			$52,529.16

5. Conclusion

5.1 Accomplishments

This semester, we were able to build a fully functioning, remote controlled microphone stand that met all of our requirements namely: movement in three dimensions (x, y, z), tilting upwards/downwards, panning left/right, and position recall. While the final result met expectations, there were many incremental milestones that paved the way to a successful project including: establishing communication between the iOS device and Wifly module, accurate operation of motor drives on the PCB, receiving and interpreting data from the rotary encoders.

5.2 Uncertainties

The primary goals of the project were met with accuracy and provide for consistent operation of the device. However, there will always be minor uncertainties for which the team cannot account. First, reading data from the given encoders and the inability to account for an imperfect floor can result in a flawless current position. Therefore, there is a likelihood that the position recall function will result in minor error.

5.3 Ethical considerations

While creating the robotic microphone stand, several ethical issues were 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 had little experience designing robotic devices; therefore, it was necessary to conduct thorough research into the design of the device. The final design was clearly documented for future revision and third-party examination.

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.

5.4 Future work

There are some improvements to be made to make the product ready for commercial use. The device currently has wires and an unenclosed PCB. Connecting all the wires into a ribbon cable and providing and enclosure with a port for the ribbon cable connection, would help to legitimize the look and feel of the product. Furthermore, voltage regulation on the battery would be helpful in protecting and decreasing the likelihood of circuit failure.

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

Appendix A Requirement and Verification Table

Block	Sub-block	Req #	Requirement	Verification	Yes/No?
1.) Power Supply	1.1) Battery	1.1a	Device must run for 1 hour at majority load	Run two DC motors (majority 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.	YES
	1.2) Buck Converter (LM2956-ADJ)	1.2a	Load voltage is at 5V +- 0.2V	Use a voltmeter to measure across output capacitor.	YES
	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.	YES
		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	YES
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.	YES
		2.1b	Receive data from iOS device	Place LED on RX (Pin 45) to verify data is being received.	YES
	2.2) Micro	2.2a	Complete Micro tutorial	Light up an LED within the tutorial.	YES
		2.2b	Receive data from iOS device and output signal from the Micro	Program the Micro to receive a signal from the WiFly module and output a signal to power an LED on the output.	YES
		2.2c	Transmit data from Micro to iOS device	Program a GPIO on the Micro to output data to the WiFly module's TX (Pin 46).	YES

	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.	YES
		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.	YES
		2.3c	Use iOS device to run motor forward	Use iOS device to output a forward signal to the WiFly module. Program the Micro to send this signal to IN1 of the H-bridge and verify motor is rotating forward.	YES
		2.3d	Use iOS device to run motor in reverse	Use iOS device to output a reverse signal to the WiFly module. Program the Micro to send this signal to IN2 of the H-bridge and verify motor is rotating in reverse.	YES
	2.4) Encoder (62a22-02-P)	2.4a	Receive pulses from the encoder when the motor turns a known amount (x) of revolutions	Program Micro to count the pulses received. Compare with encoder's specified pulses per revolution (PPR).	YES
		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	YES
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	YES
	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.	YES

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.	YES
		4.1b	The motor must stop before it reaches the top of the rod	1st, program the Micro 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 Micro to never allow up-movement past the newfound distance. Verify this works by attempting to move the mount past the top of the shaft.	YES
		4.1c	Use the iOS device to perform the up movement	Program the Micro 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.	YES
	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.	YES
		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 Micro to never let the motor lower further than the bottom of the shaft. Verify it will stop rotating when it reaches the bottom.	YES

		4.2c	Use the iOS device to perform the up movement	Program the Micro 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.	YES
Robot Movement	5.1) Right/Left	5.1a	The robot must move right/left in a straight line via the iOS device	Program the Micro to receive the 'R' and 'L' opcodes from the WiFly module. Next, program the Micro 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.	YES
		5.1b	The robot must be able to recall its right/left location from an initial point	Program the Micro 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.	YES
	5.2) Forward/ Reverse	5.2a	The robot must move forward/ backward in a straight line via the iOS device	Program the Micro to receive the 'F' and 'B' opcodes from the WiFly module. Next, program the Micro 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.	YES

		5.2b	The robot must be able to recall its right/left location from an initial point	Program the Micro 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.	YES
	5.3) Pan right/left	5.3a	Right/CW	Program the Micro to receive the 'PR' opcode from the WiFly module. Next, program the Micro to output a 'pan-right' to IN1 of motors M1 - M4. 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.	YES
		5.3b	Left/CCW	Program the Micro to receive the 'PL' opcode from the WiFly module. Next, program the Micro to output a 'pan-left' to IN2 of motors M1 - M4. 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.	YES
		5.3c	Recall home location after panning	Program the Micro 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.	YES

	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.	YES
Tilt Movement	6.) Up/down	6.1a	The servo must move via PWM	Program the PWM control to receive a signal from the Micro and output a square wave to the servo motor's 'signal' input. Verify that the arm is moving via the iOS device.	YES
		6.1b	The servo must be able to store and recall its position	Program the Micro 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.	YES
		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.	YES