

Presents the

**Autonomous Payload Delivery Challenge**

**Engineering Team:**

Alex Egg

Andrew Goria

Alvin Lacdan

Jeff O’Brien

Kevin Shinkle

Daniel Tarantino

Submitted to:

John Kennedy and Lal Tummala

**Design Co. Ltd, San Diego, CA**

Table of Contents

[*Abstract* 3](#_Toc387943705)

[*Introduction* 4](#_Toc387943706)

[*Locomotion* 5](#_Toc387943707)

[*Chassis Design* 7](#_Toc387943708)

[*Navigation* 8](#_Toc387943709)

[*Beacon Detection Utilizing Antenna* 11](#_Toc387943710)

[*Obstacle Avoidance* 14](#_Toc387943711)

[*PCB Design* 15](#_Toc387943712)

[*PCB Shield* 15](#_Toc387943713)

[*Antenna & Ultrasonic PCBs* 17](#_Toc387943714)

[*Payload Delivery System* 18](#_Toc387943715)

[*Budget:* 19](#_Toc387943716)

[*Conclusion and Recommendations* 19](#_Toc387943717)

[*Appendix* 21](#_Toc387943718)

# *Abstract*

Sharknado is charged with the task of designing and building an autonomous payload delivery robot that will be able to locate different beacons by which of traveling to programmed GPS locations and an RF antenna to deliver a payload (in the case of the challenge, golf balls), and return to the starting location in under two minutes. In the past engineering teams have bought their chassis and used the motors and wheels that had come with it. Sharknado has decided to design their own chassis made from aluminum and buy their own motors and wheels and integrate them on their own. Our goal is design and build the autonomous robot that will fall under the given requirements in the most efficient way possible.

# *Introduction*

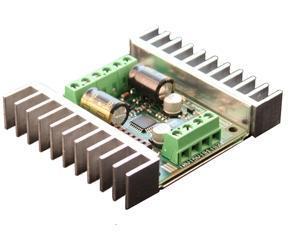
Team Sharknado was tasked in participating in the Autonomous Payload Delivery Challenge. The purpose of this challenge was to simulate military robots in the battlefield. Soldiers die in the battlefield every day to explosive traps from the enemy army but this problem can be prevented using robots. Military Robots are designed and implemented to survey the battlefield, find areas where explosive traps might be, drop payloads that will activate those traps, and drive away. Our team has designed our own model of this operation which will autonomously travel to GPS locations that we have set, drop a golf ball which simulates explosive activating devices. In the future this vehicle can one day go out into the battlefield and save real soldiers from IEDs.

Not only is this a project, it is also a competition among four other teams. These teams will have their car travel to 3 GPS locations chosen by our adviser, drop a payload at each location, and come back to the start. Every 1 foot off the golf ball is away from the target, 1 point will be taken away from the final score. If the car takes more than 2 minutes to complete the course, 1 point for every 10 seconds after will be deducted. If the car does not cross the finish line at the end, the car will get no points. Throughout the course, there will be obstacles in the way of the car which it would have to avoid. Also, at each of the GPS locations, there will be an 80 kHz signal emitting from it so our car will use an antenna which will pick up the signal to get a more accurate location of the target. Our car must do all these things in order to complete the course and this report will discuss how we took on this challenge and what we did to overcome issues.

# *Locomotion*

In order to make the autonomous payload delivery challenge to possible, we need the vehicle to move. There were two options that we were deciding on in the beginning of this semester that were going to dictate how this project would go. We could go RC which would give us the speed we would need to complete the course fast or differential drive which would give the car accuracy. In the end we decided that differential drive would be best because the way the points were distributed, it seemed like not a lot of points would be deducted for passing the 2 minute mark. Since we were going differential drive, we now had to choose between buying a kit or going full custom. It seemed that buying the kit was not cost effective because the motors were not very good and would have to buy them anyways. With that said we went full custom so we would have more freedom with parts placement and saw it a fun and unique challenge.

Getting the parts needed to get this car moving was the most important thing early on in this project. Our first step was figuring out what kind of motors we needed to get this car moving fast enough on grass. We needed the robot to travel at least 2.5meters per second and need enough torque as well to roll on grass. We have chosen the Polulu motor with a 19:1 gear ratio that allows 500 rpms with 84 oz.-in torque. This gear ratio was good because the larger the gear ratio increase the difference between torque and velocity. This would move our car over grass and be enough to push the car with all the parts we need it to have. We bought two motors with encoders and two without so that we would have at least 1 wheel on both sides of the car that we can track rotation and speed data from. Once we got the motors decided we needed to choose a motor controller that had enough power to make the motors move at the speeds we wanted to. For this we chose the Sabertooth 2x12 dual 12A motor driver which can supply up to 24V. Our motors that we chose were able to run between 7V and 12V and any more would have done some damage to those motors. Our solution was to buy a battery than can supply 24V and then power up the motor controller which can use the 24V and supply the motors which are connected in parallel. To ensure our motors do not burn out, we made sure that in the code, we would never run the motors over 50% duty cycle. This would power the motors to less than 12V which was still a decent speed to run the car. We would make changes to the code using the Arm Cortex M3 which we powered through the 5V output of the Sabertooth.



***19:1 metal gear motor w/ encoder Sabertooth 2x12 dual 12A motor driver***

Now that we had all of the parts to get this car moving, we needed to test the way the car moves. Using our microcontroller, we were able to send Serial packets to our motors get them moving. We began testing our car by driving forward, backward, turning on a dime to the left or the right and stopping. We tested the many modes that the Sabertooth can support and decided on using the packetized serial mode. For full forward speed, we set the code to have a power of 127 but we never did that because it would blow our motors. We tested this car with lower powers so that it wouldn't go to fast and hit something. When testing on the street, we had a power of 20 and on grass our power was 40. When our car turns, one side of the vehicle moves faster than the other which makes it move in the direction of the slower side of the car. If we set the turn to a power of 40 with a drive forward power of zero, one side of the motors would move at a power of 40 and the other side would move at power of -40 causing it to move at a very sharp turn.

The next step was to test our encoders so that we are able to calculate distance traveled by our vehicle and to make our vehicle turn when we wanted it to. This is where we faced many of our problems. When we tested our encoders, we saw that there was a lot of noise. The numbers spiked up way too much and were not consistent each run so we knew there was a problem. We tried to reduce the noise by changing the grounds and using different power supply for the microcontroller. Changing the power supply helped a little but in the end a decision had to be made to stop spending time on the encoders and focus on the navigation part which included the GPS and magnetometer. Using the magnetometer as a compass, our vehicle will drive in the direction of the GPS location and if the vehicle would get off course, the magnetometer will keep it on track. For further information on how this works, please read the Navigation section.

# *Chassis Design*

Sharknado is a fully customized robot to meet the specifications of the project. The main body of the chassis was water jet cut from a piece of .1 thick aluminum metal. This allowed for a stable and sturdy chassis that was still light weight. The design of the chassis was created using CorelCAD to create a .dxf file that was used to cut the aluminum. The final layout of the chassis can be seen in the picture to the right. Due to the chassis being a flat surface, custom brackets were created to hold the motors in place. The designs of the motor brackets are based off a bracket we previously purchased for the motors that were used. In order to mount a second layer on the chassis, bolts were used with nylon standoffs to ensure that it provided an even surface for additional sensors. The chassis had strategically placed mounting holes for the motor controller, motor brackets, and microcontroller with additional holes to allow for easy wiring. Additional mounting holes were hand drilled to mount printed circuit boards designed by engineers in the group. Our aluminum chassis is conductive so all of our sensors and boards had to be placed on standoffs to prevent any shorts with the exception of our motor controller that came with a heat sink attached. The picture to the left displays our initial design for the chassis which was used throughout the project. The main changes to the final design included bumpers for wheel protection. The custom design allowed for maximum space utilization and a lightweight design that led to the success of Sharknado.

Bottom Layer

Top Layer

# *Navigation*

Sharknado’s software is designed around a state machine architecture. All of Sharknado’s navigation routines take place in the Search state. Figure 2 shows a diagram of Sharknado’s state machine in the Appendix.

Sharknado’s navigation system is based around two simple controlling interfaces to the motors: turning and speed. With these controls we can navigate to a GPS point and stop. Turning and speed are both controlled by PID control systems. The turning PID takes an input from the magnetometer and the speed PID takes an input from the wheel encoders.

**Vincenty's Algorithm**

The first step in the GPS navigation is to get the compass heading and distance to the target. The heading to the target and the distance to the target is calculated using Vincenty's Inverse Algorithm. Vincenty’s algorithm is based on the assumption that the figure of the Earth is an spheroid, and hence are more accurate than methods such as great-circle distance which assume a spherical Earth and much more accurate than the naive arc tan method employed by most teams in the competition.

Vicenty’s Algo. documented here: https://www.ngs.noaa.gov/PUBS\_LIB/inverse.pdf

**PID Controller**

If the difference between the angles of Sharknado’s heading and the heading of the target is 0, then Sharknado does not do any turning. However, if there is a difference between the headings, the PID output will cause the vehicle to turn accordingly.

Tuning the PID was difficult because of the relative slow-feedback nature of our system -- it takes relatively long time for the vehicle to turn as the instruction has to pass through the UART to the motor-controller and then passed as a pulse-width signal to the motors where it is converted into physical motion by the motors. Because of this we had to turn down the integrator weighting of the PID or else it causes wilding “fish-tailing” in the vehicle.

The turning PID was very advantageous feature of our system, as it allowed us to treat our motors as a black-box. The alternative is to -- though experimentation -- to know that pulse-width X to the motors goes 3 m/s and pulse-width Y to the motors goes 4 m/s or that pulse-width Z to the motors for 10 seconds (or w/ encoder input) causes the vehicle to turn 180 degrees. This approach is only half-baked, because, among other things if the battery starts to die, the pulse X may not give the desired results anymore. This is where the advantage of the PID controller really shines: advanced control over the turning and speed of our vehicle.

**Sensor Inputs**

The other half of the PID equations is sensor inputs. The PID controller will send the motor interfaces adjustments based on the sensor readings; so good sensor samples are vital to the success of the project.

*Wheel Encoders*

One of the original goals was to take input from the wheel encoders to help feed the speed PID and also use it for distance tracking and more advanced maneuvers. However, we were unable to correctly implement the encoders due to noise interference from out DC motors in the system.

*Magnetometer*

The magnetometer is one of the most important sensors we are using. It gives us our heading relative to magnetic north. This input is fed directly into the PID every system clock cycle. Samples from the compass sensor have degree of noise/error associated with them. For example: leaving the sensor stationary on a desk does not produce a constant reading, but rather a good amount of jitter. To combat this we tried implementing a sliding window median, however, this turned out to have a slow response. We also tried a FIR filter (see Appendix figure 3), but it turns out the characteristics of the PID system work as a good-enough filter since the vehicle has such a slow response, fast fluctuations in the heading are not observed, but rather averaged out numerically.

*GPS*

The GPS sensor was very high-quality. A test on a stationary desk, produces very little noise or reading jitter. Also, on average, we are able to get accuracy of about 2 meters to the target. No filtering was necessary for the GPS unit and samples are fed directly into Vincenty's algorithm, where the target heading is then set as the new turning PID set point.

A sample of the GPS headings can be seen in Appendix Figure 4.

Sharknado has full-feedback system which reliably drives it to the GPS targets. Feedback from the sensors is passed into the PID and causes Sharknado steer or accelerate. Also Sharknado has a sophisticated distance and heading algorithm which allows it to navigate across the SDSU field, but also, in theory across large spheroid geographies.

# *Beacon Detection Utilizing Antenna*

At the drop location is a Beacon emitting electromagnetic sine waves with a frequency of 80 kHz. A loop antenna was created to receive the Beacon signal. Normally the size of the loop antenna is directly proportional to the wavelength.

**λ =c/f** = 3x10^8(m/s)/ 80(kHz) = 3750 meters.

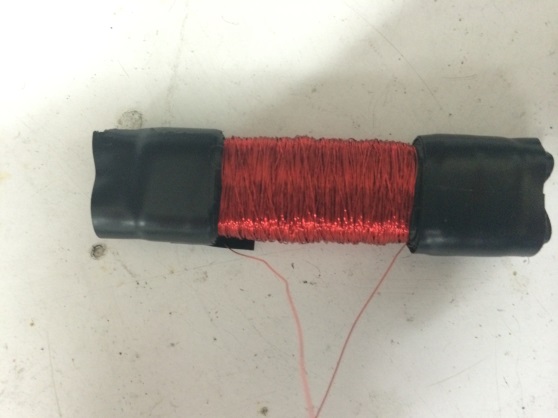
Since the signal is only 80 kHz the wavelength of the signal is very large. We cannot have an antenna with a loop length of 3,750 meters; therefore, we wrapped the loop antenna about a ferrite rod. The ferrite material absorbs the magnetic portion of the signal allowing us to use a much shorter length.

The loop antenna uses 700 turns of magnetic wire about a ferrite core. The turns of the antenna create an inductance, L, of 28.3mH which needs to be tuned with capacitors, based on the equation below, to resonate at 80 kHz.

**FResonate=1/(2\*π\*Sqrt[LC])** 🡪 80kHz = 1/(2\*π\*Sqrt[(28.3x10^-3)C]) 🡪 C= 139.85pF

Based on testing, the actual capacitance needed was only 86pF.

The Beacon and our wrapped antenna can be seen below.

Receiver Antenna Transmitting Beacon

As our vehicle gets closer to the beacon, the antenna will receive and provide a sine wave with larger voltages. This signal will be sent to receiver circuitry to determine the location of the drop.

Our receiver circuit has 4 different stages. For our entire receiver schematic see Appendix Figure 6.

Stage one provides gain to receive the Beacon from a greater distance seen in Figure 7 of the Appendix. It is an operation amplifier, the MC30378, in a non-inverting configuration. Its gain is decided by its two resistors: R1 and R2. We chose R1= 11kΩ and R2=100kΩ.

**Gain = R2/R1 +1** = 100k/11k + 1 = 10.091 v/v

The reason we did not have the gain be very large is because operational amplifiers have limits on their frequency response. Our input signal to the op amp has a frequency of 80 kHz.

One extra feature of this stage is an added filter. R2 is in parallel with a capacitor to create a RC filter that removes noise above 100 kHz. It is a low pass filter with 100 kHz cut off frequency.

**FCutoff=1/(2\*π\*R\*C)**

The second stage takes the received sine wave and half-wave rectifies it. This is a process where an input signal is turned completely positive. Its design, with inputs and outputs, see Appendix Figure 8.

It is in an inverting configuration so the negative cycles are turned positive. We used a precision rectifier because it allows us to bypass the voltage drop of a simple diode rectifier.

The third stage makes the signal constant. A resistor and capacitor are connected at the end of the rectifier to make a DC voltage with ripples from the positive peaks. The capacitor and resistor are in parallel leading to ground.

The fourth stage is a protection circuit to reduce the voltage to useable levels for the Microcontroller. See Appendix Figure 8.

The zener diode prevents the voltage from exceeding 3.3v by shunting it to ground. The resistor is placed to protect the diode from large currents.

The payload will be dropped when the controller received 2 Volts which is about 1-2 feet. The Microcontroller will read the voltage through analog to digital conversion.

# *Obstacle Avoidance*

In order to avoid obstacles on the competition field, Sharknado utilizes three ultrasonic sensors. Each of the sensors has a range of approximately 200 centimeters or around 6 meet. This range allowed for ample amount of time to react to the obstacle before running into it. One ultrasonic sensor was placed directly in the middle of the chassis and the other two were placed on either side. The ultrasonic sensors the each side were placed at a 30 degree angle to allow for maximum vision on Sharknado. By using two separate timers, a trigger pulse of 50 microseconds is sent every 250 milliseconds. One timer is used to set the signal to high and the second timer is used to make the signal go low again. When receiving data from the echo signal, the amount of time between signals was calculated to a distance which was used to flag whether a collision is close or not. Once an obstacle comes within range of the ultrasonic sensors, the state machine of Sharknado will switch to the escape state where it executes the escape algorithm. Based upon which ultrasonic sensors were triggered, the code intelligently decides whether to slightly turn left or right, or to execute a full pivot in order to find a clear path to travel through. The ultrasonic sensors were placed on custom made printed circuit boards which regulated the echo voltage down to 3.3 volts in order to ensure it is compatible with the microcontroller. One recommendation for future projects would be to invest in more effective and higher quality ultrasonic sensors. The sensors used in this project were greatly affected from noise within the system and became more inconsistent as the complexity of the code increased. Also the response time between triggering a signal and executing the escape signal was slower than anticipated.ltrasonic sensors. to nd a clear ppa

# *PCB Design*

## *PCB Shield*

In order to cut down on unorganized wires going in and out ARM Cortex M3 and potentially damage multiple sensors and well as ugly breadboards, a PCB was required in order to fix that problem. This PCB was to be mounted directly on to the ARM Cortex M3. Also the use of wiring harnesses was needed in order to keep the wires organized and have a much cleaner look. Different color wires were used to distinguish which were ground, power, output, as well as input.

Figure 10 in the Appendix represents the first prototype PCB shield for the ARM Cortex M3. It was given the designation “The Shark Shield” by its designer. This prototype was a hub for all the sensors to connect to. In addition, the motor controller, Sabertooth 2x12, also connected to the Shark Shield. The 5 V output from the motor controller, ground, and the input S1 were all to be connected to the Shark Shield. In the beginning, the input and outputs of the 2 encoders as well as 2 ultrasonic sensors were to be connected to the top left of figure 1.1 respectively. Below the ultrasonic sensors housed the servo payload motor PWM input, 5V input, and ground respectively. Near the center of figure 1.1 housed the 9-pin GPS was to sit on top of the Shark Shield by use of an 8-Pin female header. Just below that shows where the 4-pin magnetometer input and output connection was housed. The magnetometer was not to sit on top of the Shark Shield like the GPS but was to be raised a few more inches above the second layer of the chassis by a PVC pipe. However, multiple issues on the PCB were soon realized once full assembly was complete. Testing resulted in complete failure. Many connections below and above the Shark Shield were extremely difficult to solder as others were impossible to solder correctly. As a result, some pins were never connected to receive any signals and/or voltage what so ever. Leaving some sensors completely unpowered and no data could be received or transmitted. Furthermore, a new 9-pin magnetometer was added, the addition of 3rd ultrasonic sensor, and the addition of a push button switch, the first Shark Shield was never used.

Figure 11 in the Appendix represents the 2nd and final revision of the Shark Shield. The differences can easily be seen when comparing figures 10 and 11 The traces on top that are connected to the header through holes are to be connected to the ARM Cortex M3. Labels were also printed on the Shark Shield to designate which sensor goes where. For example, U1 is the right ultrasonic sensor, Eright is the right encoder, and PB stands for push button switch. Another change is that both the GPS and Magnetometer are now on the left side of the Shark Shield. In addition, since the magnetometer is unaffected by the magnetic interference from the motors, the magnetometer will now sit on top of the Shark Shield just like the GPS and next to the GPS. The traces that are connected to the though holes for the sensors, antennas, and LEDs are underneath the Shark Shield to give the PCB a nice clean look on the top. In addition, surface mount capacitor pads were added to each sensor to help with decrease noise coming from the ground. In addition, the ultrasonic placement and the motor controller placement were moved from the left side to the front side for easy connection. A push button switch was added to start and stop the program from the ARM Cortex M3. The encoders and the servo were moved to the rear of the Shark Shield. In addition, the left side of the Shark Shield was shortened while the right side was lengthened to accommodate added traces. Also, the 5 V output from the motor controller is no longer powering the ARM Cortex M3, therefore it was not connected to anything. Furthermore, 2 LEDs were added to show when the GPS and magnetometer and GPS got a fix.

## *Antenna & Ultrasonic PCBs*

A necessary task of integrating every part of Sharknado was designing and building printed circuit boards (PCB). As discussed earlier, PCB’s significantly reduced the amount of loose wires needed on the chassis. Any complex circuitry with multiple parts i.e. resistors, capacitors were sent to be implemented onto a PCB. One of those complex circuitry includes the gain stage and precision rectifier stage of the antenna.

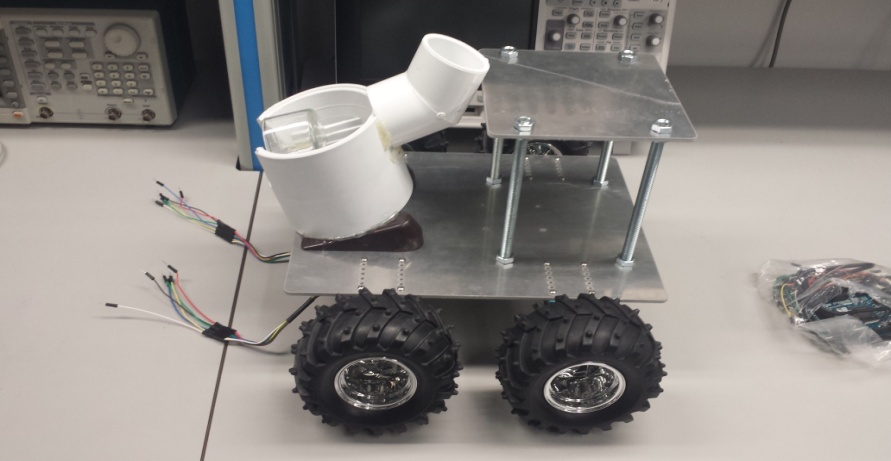
Two pieces of information were involved in the planning stages of designing the PCBs. The first being the location of the actual antenna. Since the tuning capacitors of the antenna work best when physically close to the antenna, the antenna PCB was designed with that in mind. The PCB would be bolted down near the front with the antenna. The second piece of information needed was which parts were going to be through-hole or surface mount. In this case, all resistors and capacitors were surface mount and all op-amps and diodes were through-hole so as cut down on costs so all stated parts were freely provided by the school.

The first version of the antenna PCB (Appendix Figure 12) was unnecessarily small since at the time of designing, the amount of space allowed on the chassis was relatively unknown. This made things difficult when soldering the parts to the board in the lab. Antenna PCB Rev. A is now referred to as a test PCB. Although there were problems encountered when soldering the board, the end result worked well enough with the antenna to get results in testing.

Three key changes were made in the final design of the Antenna PCB (Appendix Figures 13). The first was the size. The overall design of Sharknado was finalized meaning all locations of components were known. The new chassis was bigger allowing more room for the PCB which in turn allowed a bigger PCB. Soldering parts to Rev. B was significantly easier. More room for the PCB led to the second key change which was the addition of mounting holes. The PCB could be mounted to the chassis with standoffs without the worry of any shorts. The addition of a ground plane to the bottom of the board was the third difference which allowed for surface mount parts to be soldered on both sides of the board when space was needed.

Other PCBs were needed for the protection circuits between the ultrasonic sensors and the microcontroller. The ultrasonic sensors were fed with 5V but also outputted the 5V back to the microcontroller. Since the microcontroller is not able to take the 5V but only a max of 3.3V, there needed to be a circuit between the two. The circuit was designed to only allow a max of 3.3V to the microcontroller in the form a simple zener diode circuit with a current limiting resistor (Appendix Figure 9). The 3.3V zener diode allowed any voltage above 3.3V shunted to ground which fit the requirements perfectly. Each PCB was designed differently for a specific ultrasonic based on its location of right, center, and left.

# *Payload Delivery System*

The Payload Delivery system is constructed out of a four inch PVC pipe with a slightly smaller PVC pipe within it. The outer PVC pipe has openings on both sides to allow for a ball to fall through each side. The inner PVC pipe has only one opening to ensure that a single balls falls into the custom payload holder. A standard 180 degree servo motor is attached to the center PVC pipe allowing it to turn and distribute the golf ball from the holder. The Servo motor is programmed to turn the full 180 degrees, shake back and forth to ensure the ball falls out, and then return to the starting position so a new ball can fall into the delivery mechanism. A ball storage device holds a total of three additional golf balls and utilizes gravity to ensure the next ball is dropped into delivery mechanism. Due to the high amount of current draw, the servo motor was powered by a five volt output from the motor controller. This allowed for the main 24 volt battery to supply all the current required to operate the device. It was a necessity to ensure the servo motor was exactly centered in the middle of the inner PVC pipe. When the motor was not centered, the amount of friction created from dropping a golf ball was greater than the available torque. This prevented the balls from being consistently dropped at the necessary locations. Also in order to prevent the motor from turning to due bouncing while driving, the program was required to constantly have a current degree location for the servo motor to be at. This also created tension on the motor that eventually wore down the gears making decreasing the amount of available torque from the motor. Despite all of the challenges, this mechanism was very efficient and allowed for an effective method of dispensing payloads at the necessary locations.

# *Budget:*

Sharknado received a $750 budget to build this autonomous payload robot. The pie chart found in the Appendix shows how the $750 budget was distributed. The project was kept under budget since the team was careful with the more expensive parts including the motor controller and motors. Thankfully no expensive parts burned out or broke.

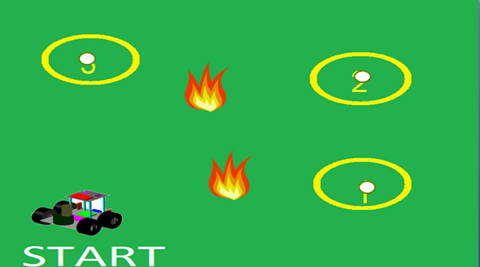
# *Conclusion and Recommendations*

Designing and building an autonomous robot came with many challenges that the team at Sharknado had to overcome. As a first project for Sharknado there were mistakes that were made by the team that might have been avoided if the team had previous experience. However, these mistakes prove to be great lessons for the team in the future. There are a few recommendations Sharknado would give if another team were to attempt the same project. One of those recommendations is to start designing the PCBs as soon as possible since errors in deigning them are relatively easy to make. A Shark Shield 1.3 should have been made to fix the multiple issues discovered from Shark Shield 1.2. One issue is that a 12mm push button did not fit in the PB through-hole placement because the holes needed to be spaced wider and further apart from each other. However, a smaller push button switch was acquired and added to the Shark Shield. In addition, 2 traces attached to the through holes to connect 2 different pins on the ARM Cortex M3 were underneath the Shark Shield and impossible to connect to the traces with solder, so wires were soldered on the top to connect these pins to the ultrasonic sensor and the servo.

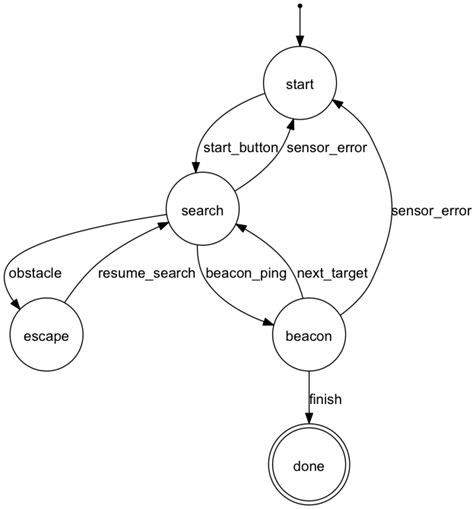
A good recommendation for future teams is to fine tune a good antenna so that more gain can be obtained since Op Amps could not provide much gain at the 80 kHz frequency.

A major regret the team collectively has is the purchase of cheap ultrasonic sensors. The sensors were noisy and unstable and cause the team to abandon all use of obstacle avoidance on the day of the competition. If the team had gone with more expensive sensors, it is likely Sharknado would have completed the challenge.

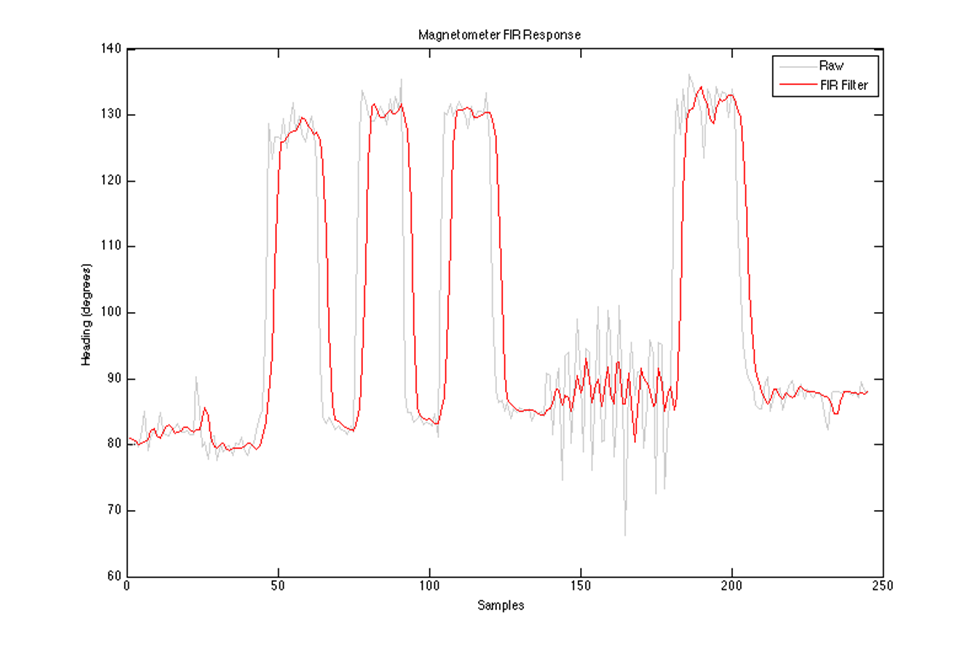
# *Appendix*



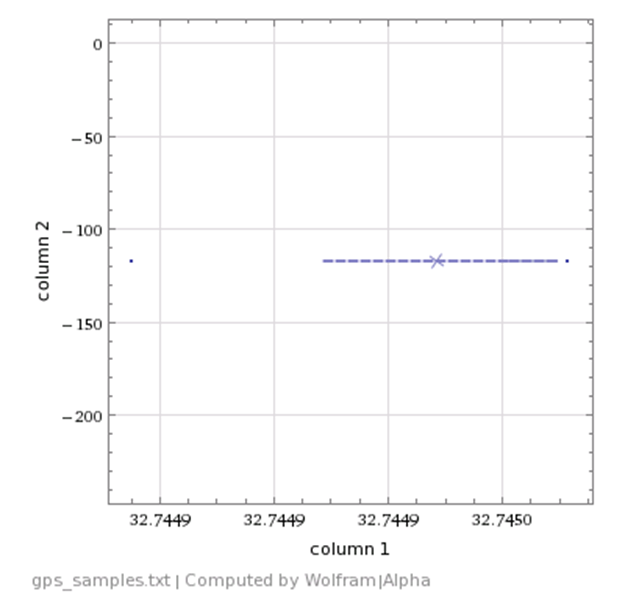
*Figure 1 Task Layout*



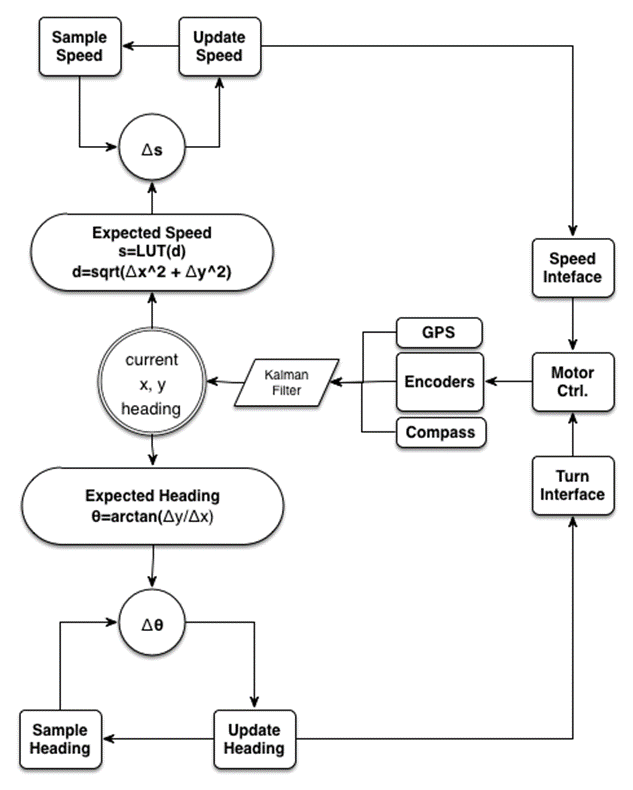
*Figure 2 Sharknado State Machine*



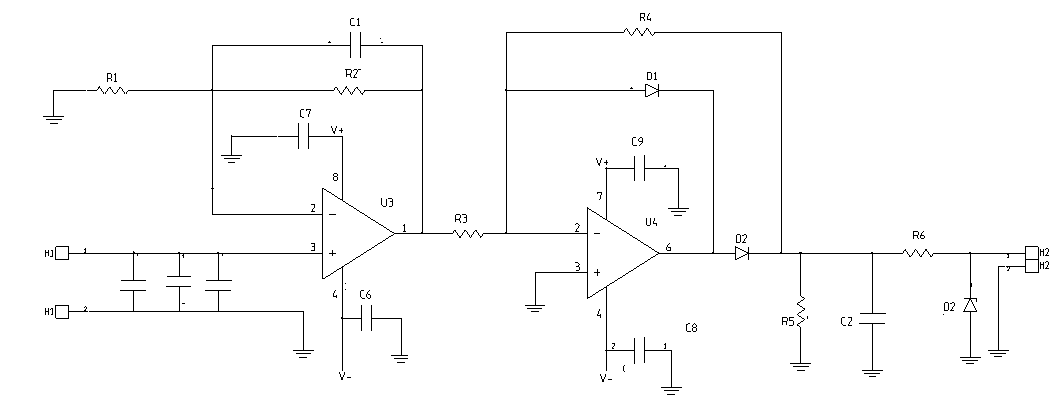
*Figure 3 FIR Filter*



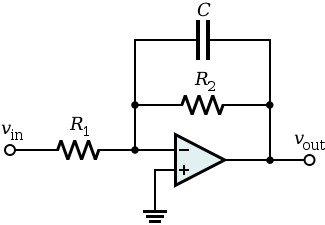
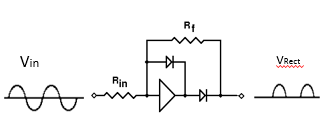
*Figure 4 the above plot has > 400 samples, however, the values of the longitude are all constant and the latitude only changes 3 times*



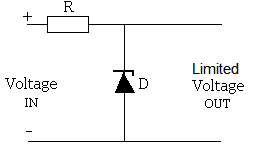
*Figure 5 Total Navigation Block Diagram*



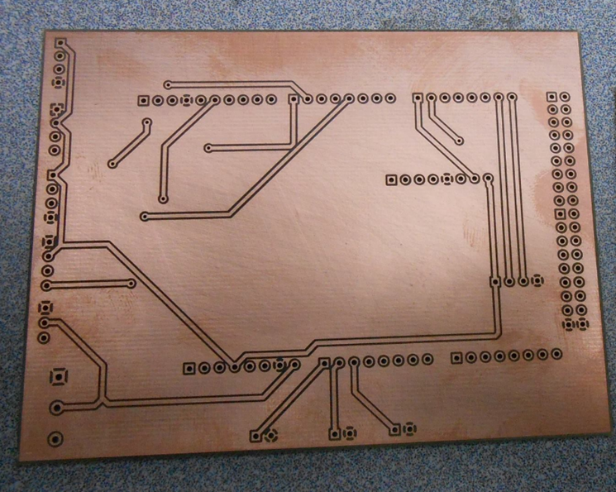
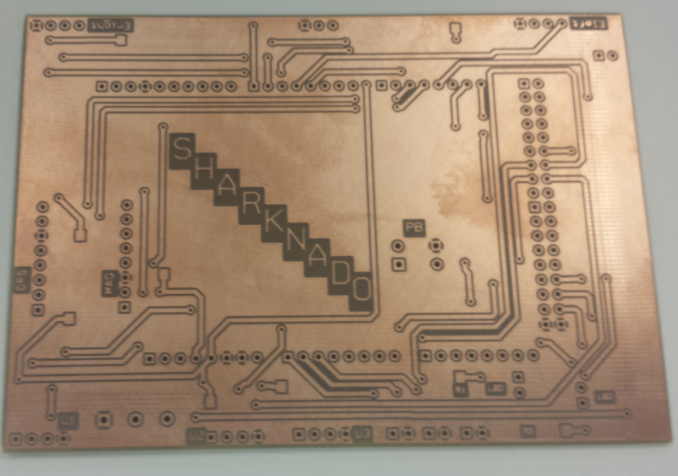
*Figure 6 Beacon Receiver Circuitry*

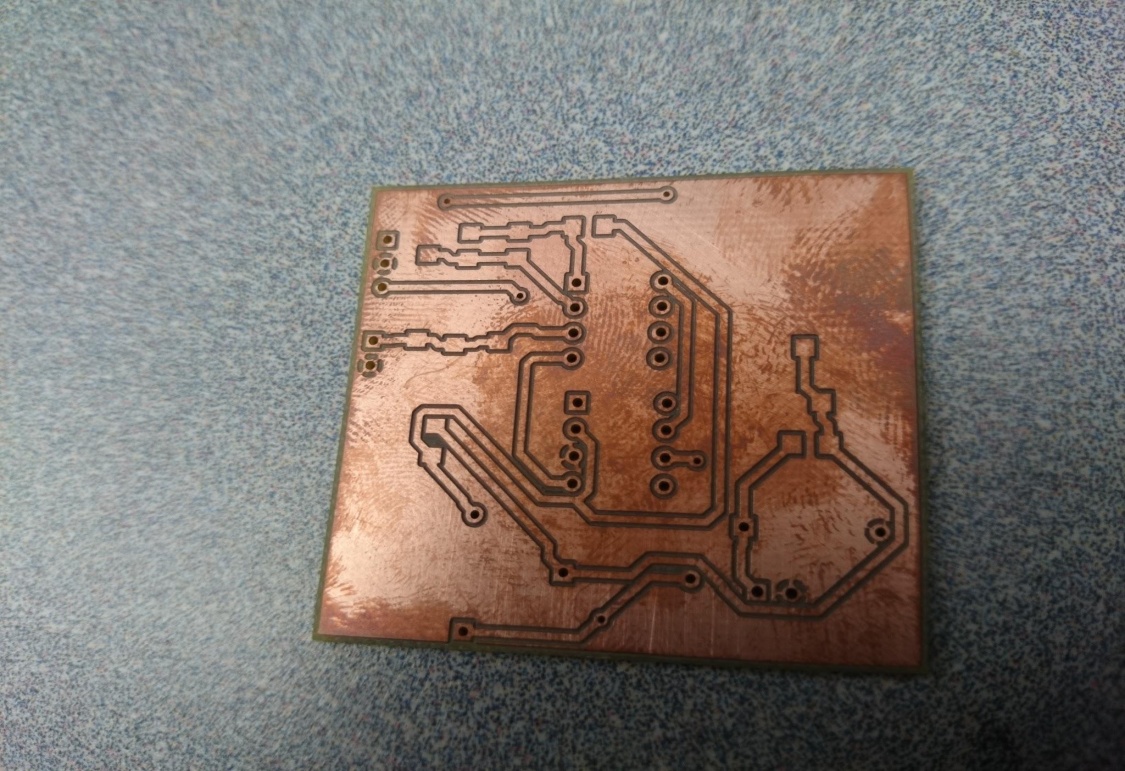
*Figure 7 Non-inverting Gain Stage Figure 8 Half-Wave Rectifier*



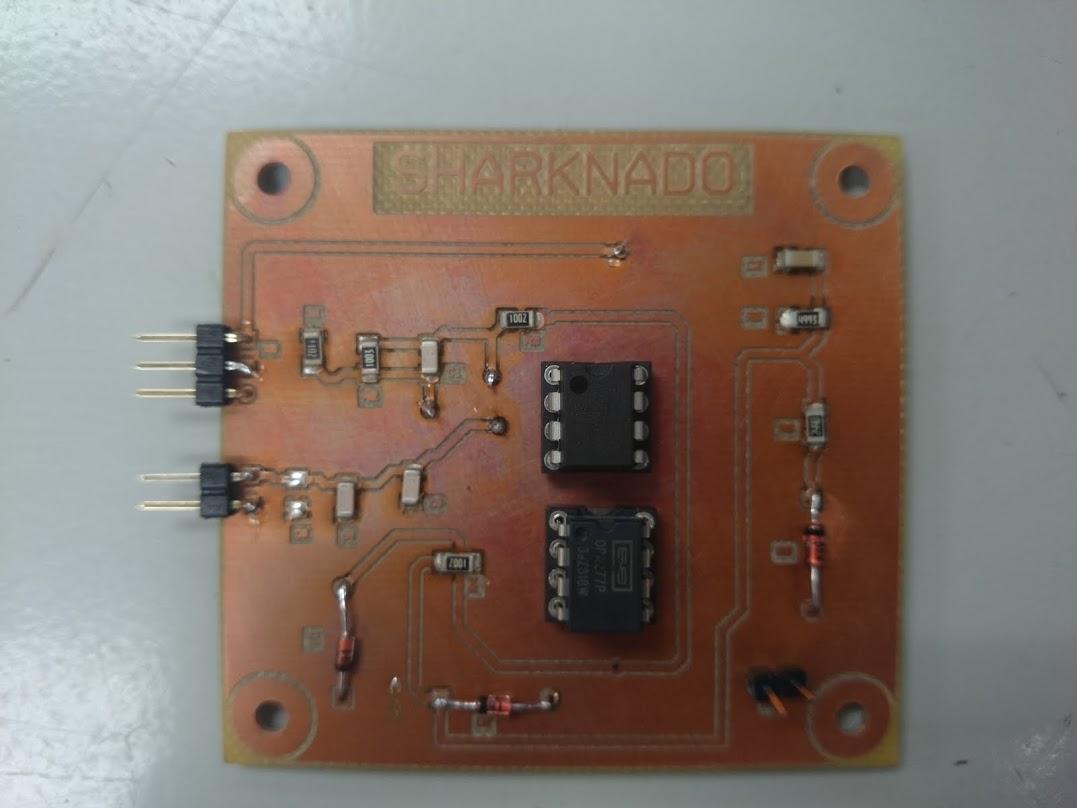
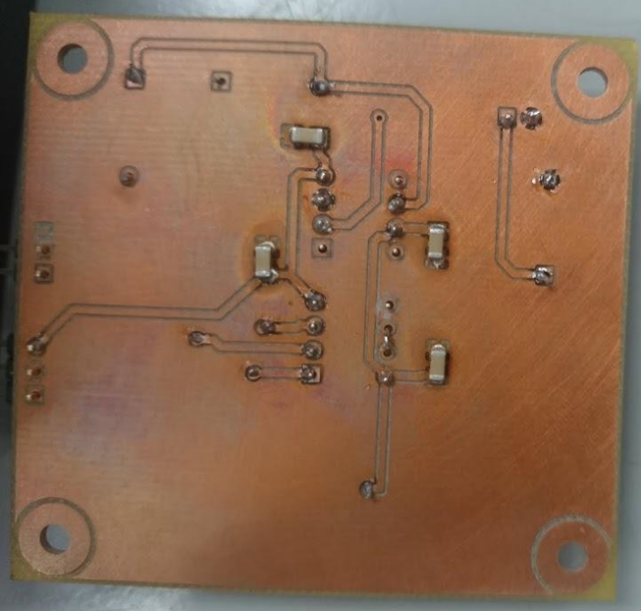
*Figure 9 Protection Circuit*

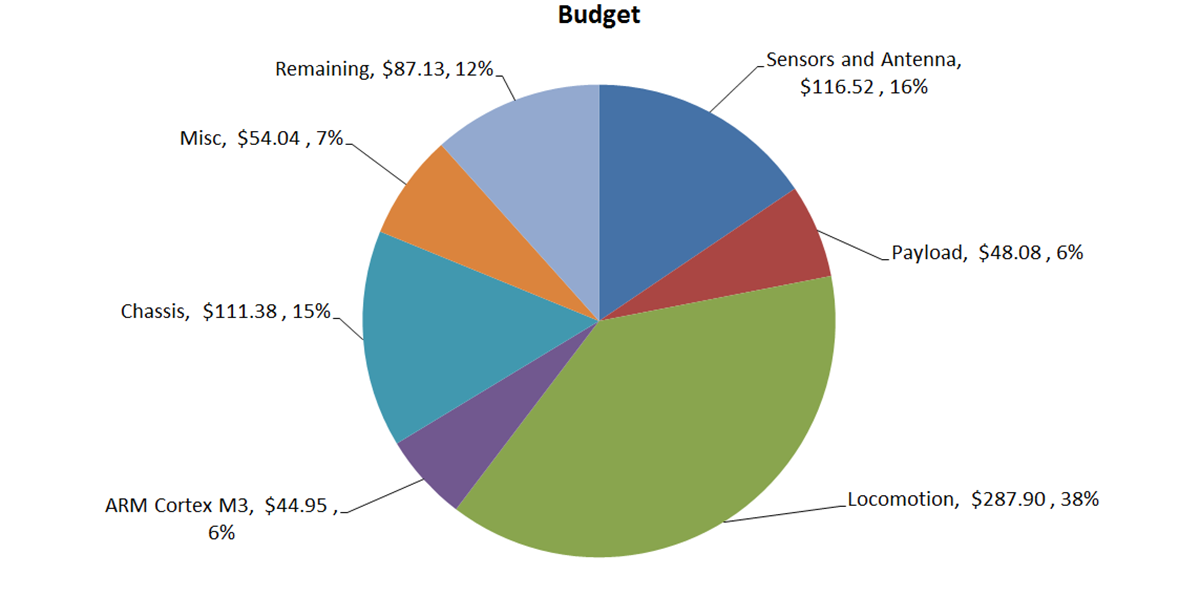
*Figure 10 Microcontroller Shield Rev A Figure 11 Microcontroller Shield Rev B*



*Figure 12 Antenna PCB Rev A*

* *

*Figure 13 Antenna PCB Rev B Top layer (left) Bottom Layer (right)*



*Figure 14*