Soteria:
Beacon-Finding Robot

Final Design Report

Lynnae Logan, Tracy Sjaaderma
Christopher Dang, Ronald Hruban

New Mexico Institute of Mining and Technology
Electrical Engineering Junior Design – EE 382

TABLE OF CONTENTS

LIST OF FIGURES

1.0 ABSTRACT

Locating a missing person or missing object can be a difficult task, especially when searching a large area. The search must be thorough and care must be taken not to overlook a single spot. However, if the missing person or object were transmitting a signal, locating would be made much simpler, and an autonomous robot could conduct the search. This paper details the designing and testing of a beacon-finding robot. This robot is required to be capable of autonomously navigating an unobstructed outdoor area. It must also be able to identify a message encoded by Frequency-Shift Keying modulation that is transmitted by a beacon. Lastly, it must remotely find and return the GPS coordinates of this beacon within an accuracy of 10 meters.

Each of the subsystems and individual components of this project are described and explained, as well as the overall integration. Due to component failures in the last week of testing, complete integration was not achieved, however, each of the subsystems are completely functional. With more time, the robot could be completely integrated, fully functional, and would successfully meet all of the design requirements.

2.0 INTRODUCTION

In this project, an autonomous robot was designed to remotely locate and identify a beacon as well as return its GPS coordinates. If the transmitting beacon was attached to a person, this robot could be used for Search and Rescue purposes, and was therefore given the name Soteria – the Greek Goddess of safety and of deliverance and preservation from harm. During this semester-long project, components were combined to form subsystems that aid the robot in mobility, navigation, location, and RF processing. With each of these subsystems being integrated together using an Arduino Due, Soteria was equipped with the memory, sensors, and information needed to drive in a straight line at a specified compass heading, determine her own GPS coordinates, receive and identify an FSK modulated identifier transmitted by a beacon, and then determine the relative direction of that beacon. Through MATLAB post-processing, the latitude and longitude of the beacon would then be determined within an accuracy of 10m.

Due to component failure, the Arduino Due was replaced with an Arduino Uno in the final stages of the project. Despite this setback, Soteria still performed as expected, although navigation and beacon location programs had to be run independently. With autonomous robots such as Soteria, missing people can be more easily found in remote and even dangerous locations where human rescue teams cannot safely search.

This report explains each subsystem in detail before describing the final integration of the project. Results from testing each subsystem, as well as final overall results, are also discussed.

3.0 OBJECTIVE

The purpose of the design project was to design and build an autonomous robot that could identify an RF beacon and return its GPS location. The robot had to meet several requirements. It had to autonomously navigate an unobstructed outdoor area. The robot had to correctly identify an RF beacon via its FSK identifier. The FSK Identifier was transmitted with a baud rate of 600. The high bit frequency was 2.2GHz, and the low bit frequency was 2.15GHz. The identifier had to be checked against that of the requested beacon. In addition, the robot needed to gather enough data to remotely locate the position of the beacon. The beacon's GPS coordinates were then to be returned with an accuracy of 10m. These requirements were to be met within the timeframe of a single semester. A budget of $325.00 was given for the project, as were the chassis, motors, batteries, and several RF components.

4.0 SUBSYSTEMS

In order to complete these required tasks, the project was divided up into the following subsystems: Antenna Design and Antenna Rotation, RF Signal Processing, Navigation, and Beacon Location. Simulated data was used to test each of these subsystems individually to ensure they were fully functional before integration was attempted.

4.0.1 SYSTEM OVERVIEW

A unique and original robot was designed to meet the given challenge. The robot uses a repetitive search method to gather data on the beacon to be used in post processing. The robot begins by running a short calibration routine for the compass. This routine doubles as a delay to allow GPS satellite locks to be obtained. Once calibrated, the robot travels in a predetermined heading using the onboard compass to make course corrections. After travelling a distance of about 7 feet, the robot stops. The servo then begins to rotate the antenna, measuring the max power angle at each angle as determined by the peak detector. Once all angles have been searched, the antenna is returned to the position of highest received signal strength. The signal being received is then compared to the identifier of the desired beacon. If the identifier is a match, the GPS, compass, and servo position information of the robot is saved to the onboard SD card. The robot then resumes travel along the previous heading until it has traveled another 7 feet and repeats the searching process. The robot makes a total of 15 stops to gather the data needed to locate the beacon accurately through post-processing performed in MATLAB.

4.1 ANTENNA DESIGN & ROTATION

The Antenna is a very important part of the overall robot design. It is responsible for receiving the beacon signal with high fidelity at a distance. The antenna needed to be able to receive a 2.2GHz signal and also be very directional.

4.1.1 ANTENNA DESIGN

An antenna was provided for the project, however, the provided antenna was very large and heavy. In order to save on weight and increase the stability of the robot, a custom antenna was designed and built. After examining many antenna types, the Yagi antenna design was chosen for the project.

As a proof of concept, a small 3 element Yagi antenna was built using wood as the boom and thick copper wire as the parasitic antennas, which act as reflectors and

directors for the incoming signal. The design in Figure 4.1.1 was used with a wavelength of 13.6cm to design the initial antenna.

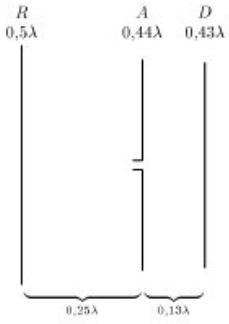

Figure 4.1.1: 3 Element Yagi Antenna [4]

This antenna was tested using a spectrum analyzer. Measurements of the power output of the antenna were taken at 5° intervals of azimuthal rotation. The gain of the antenna was found to be 7.4dBd, dBi gain above a tested isotropic antenna at the same distance. The half power beam width of the antenna was found to be 45°.

After it was discovered that it was possible to build a functioning Yagi antenna, a higher power Yagi antenna was designed and built. The antenna elements were calculated using the Yagi Design Calculator on the K7MEM amateur radio website. The design specifications entered into the calculator were: a frequency of 2.15GHz, a non-metallic boom with a length of 560mm, and a 2mm element width. The antenna design specifications output is in Figure 4.1.2.

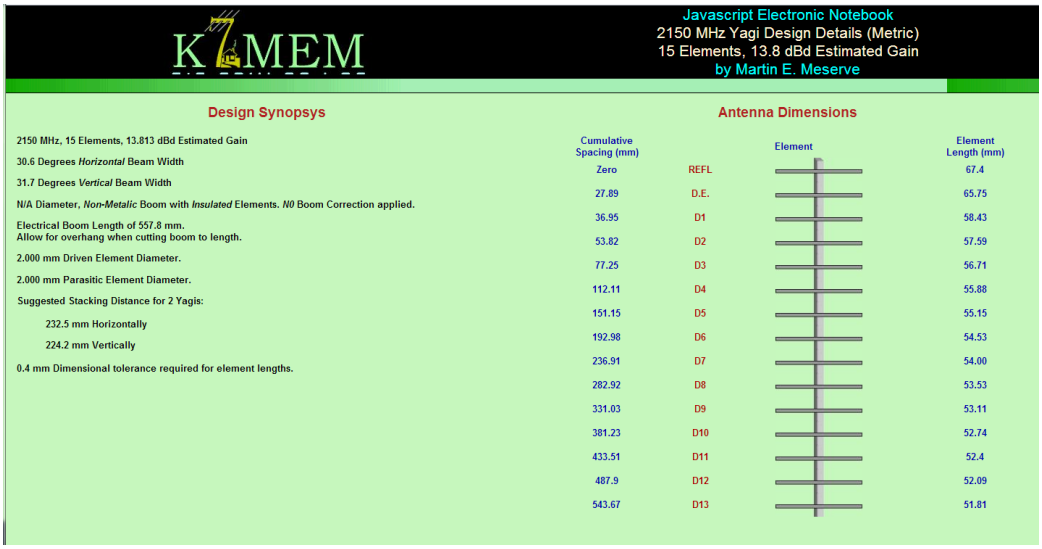

Figure 4.1.2: 15 Element Yagi Antenna design

The antenna design specified 0.4mm dimensional tolerances. With the tools available, these tolerances could not be met. This created errors in the antenna and the final functionality of the antenna was less than the theoretical functionality. The 15 element Yagi antenna had a 12.3dBd gain, which was 1.5dBd less than expected. The produced antenna had a half power beam width of 60°, which was far greater than the expected half power beam width of 30.6°. Due to the extremely low directionality of the 15 element antenna, the 3 element antenna was used in subsystem tests. The finished two antennas can be seen in Figure 4.1.3.

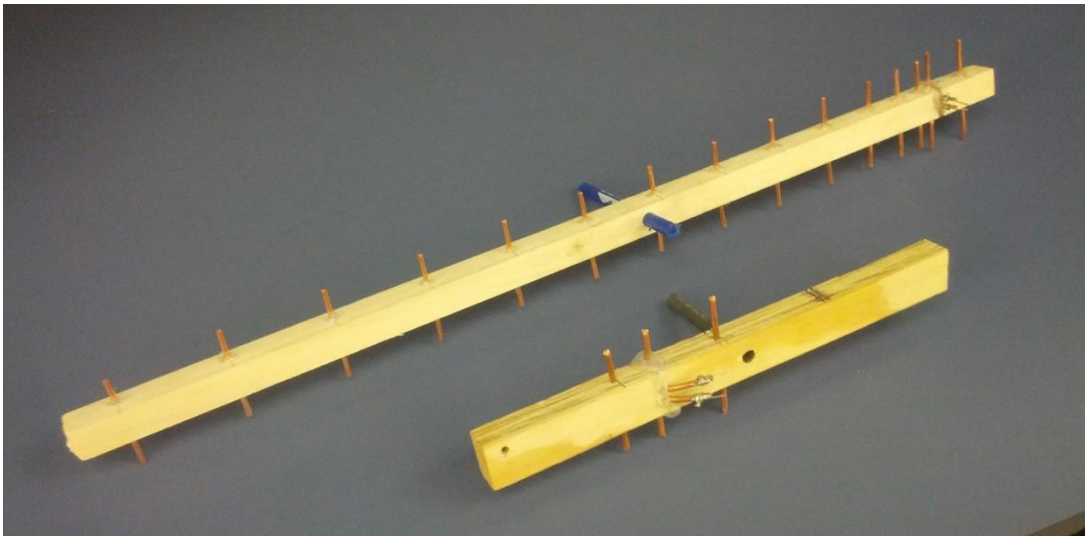

Figure 4.1.3: 15 Element Yagi Antenna (top) and 3 Element Yagi Antenna (bottom)

4.1.2 Antenna Control Servo

In order to more accurately search for the direction in which the antenna found the strongest signal, it was decided that the antenna would rotate on top of the robot. A servo was the mechanism chosen to rotate the antenna. Servos can consistently turn to specified angle positions. This made the servo the optimum choice for rotating the antenna in small degree increments. To save on costs, a donated 180° Parallax 900-00005 servo was used. In order to obtain 360° of antenna rotation, a custom chain drive was built to connect the servo to the antenna. The chain drive, in Figure 4.1.4, has a bracket which keeps the servo from rotating.

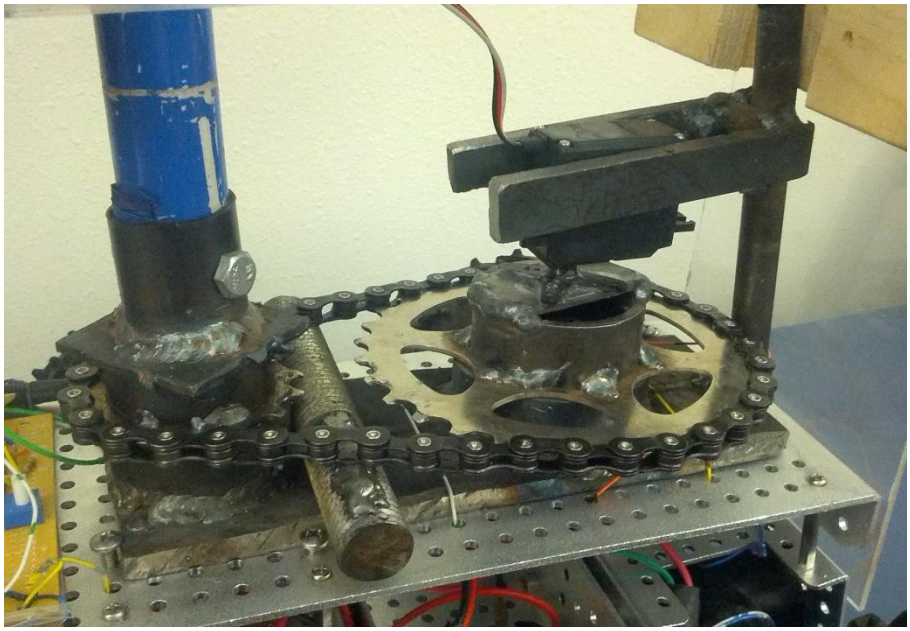

Figure 4.1.4: 2:1 Antenna Chain Drive System

The servo horn is fastened to a 32 tooth gear which drives a 16 tooth gear via a guided chain. The antenna mast is mounted atop the 16 tooth gear and braced by the plexiglass body of the robot. The 2:1 gear ratio allows for 360° of rotation to be achieved by the antenna when the servo has only rotated 180°. The set up successfully rotates the antenna 360° and always returns the antenna to its starting position when used with the Arduino Servo Library.

4.2 RF Signal Processing

The RF signal processing portion of the Soteria Robot allows for proper identification of the beacon transmitter. After down converting and demodulating the incoming signal, verification of the correct beacon can be made.

4.2.1 Down-Conversion

The beacon that the Soteria Robot is required to locate transmits an identifier encoded by Frequency Shift Keying (FSK) modulation. FSK modulation turns a digital binary signal into a sinusoidal signal. For every high bit – digital 1 – a sinusoid with a high frequency is transmitted. For every low bit – digital 0 – a sinusoid with a low frequency is transmitted. This is illustrated by Figure 4.2.1.

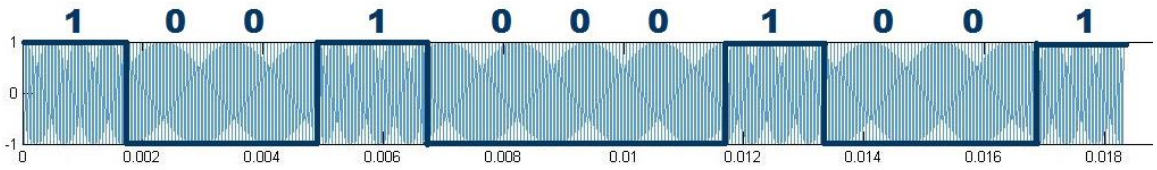

Figure 4.2.1: FSK Identifier

In the case of the beacon being searched for by the Soteria Robot, a high bit is transmitted by a sinusoidal signal with a frequency of 2.2GHz, and a low bit is transmitted by a sinusoidal signal with a frequency of 2.15GHz. Figure 4.2.2 shows these two signals represented in frequency-space on the Spectrum Analyzer.

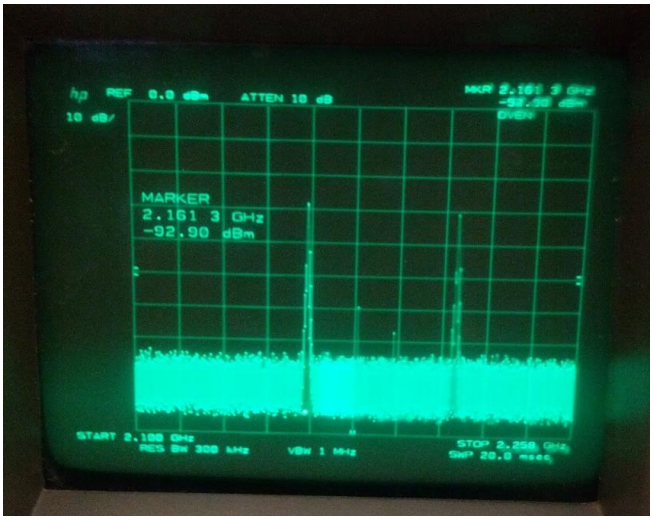

Figure 4.2.2: 2.2GHz and 2.15GHz Frequencies

Many components and microcontrollers are unable to work with such high-frequency signals, and so the signals must be lowered to slower frequencies. This is done through a process called “down-conversion”. In order to down-convert the signal from the beacon, three main components were used.

The first component used was an amplifier, ZFL-2500+, which strengthened the very weak signal received by the antenna. The second component used was a ZX95-2400A+ Voltage-Controlled Oscillator (VCO). A VCO outputs a sinusoidal signal with a frequency determined by the input DC voltage. The signals from the antenna and the VCO are both sent into the RF and Local Oscillator inputs of the Mixer. A mixer takes two sinusoidal inputs and returns two signals: the first has a frequency that is the sum of the input frequencies; the second has a frequency that is the difference of the input frequencies.

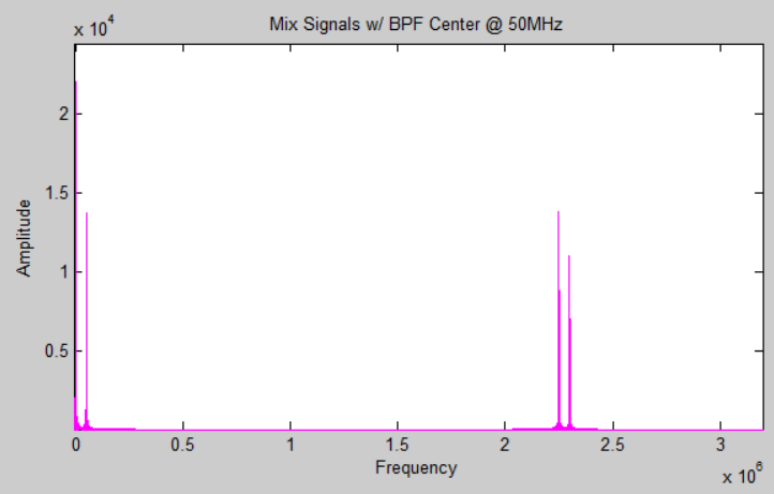

Figure 4.2.3: Simulated Frequencies before Down-Conversion (right) and After (left)

Since the signals transmitted by the beacon differ in frequency by 50 MHz, the lowest this pair of frequencies can be down-converted to is 0 and 50MHz. Figure 4.2.3 shows this. In frequency space, every sinusoid is represented by two impulses, one at the positive frequency and one at the negative (Figure 4.2.4). When the positive impulses are shifted down, the negative impulses are shifted up. When the lower positive impulse is shifted down past 0Hz, its negative component is shifted up past 0Hz. This gives the possibility of aliasing – two signals interfering and thus becoming indistinguishable. Therefore, 0Hz and 50MHz are the lowest frequencies to which the 2.15GHz and 2.2GHz sinusoids can be down-converted to, respectively.

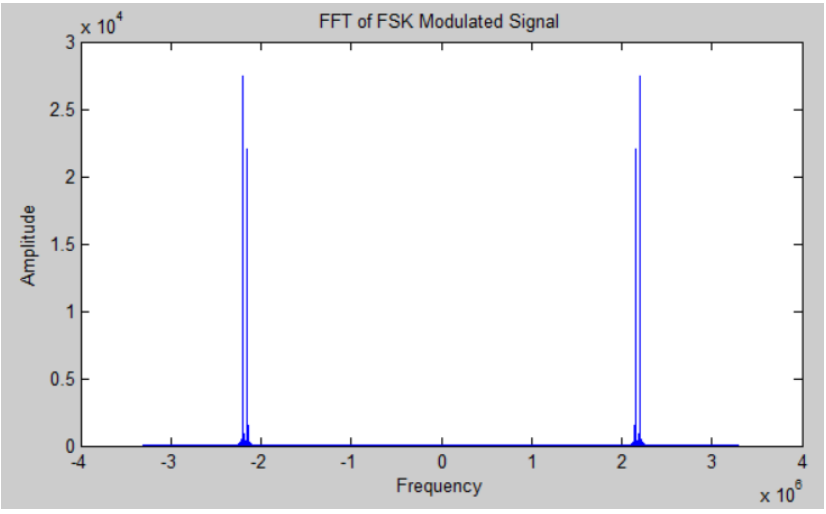

Figure 4.2.4: Impulses of a Frequency

In order for the beacon signals to be down-converted to these frequencies, the antenna signal had to be mixed with a sinusoid that had the same frequency as the lower beacon signal – 2.15GHz. The datasheet for the VCO used indicates that the tuning voltage – V_{tune} – must be approximately 9.07V. The circuit built to supply this voltage contained a voltage divider with a potentiometer for fine-tuning. A spectrum analyzer was used to observe the signal coming from the VCO, and thus V_{tune} was adjusted to be the exact voltage needed to create a 2.15GHz sinusoidal signal.

Once these three components were connected together, the output of the Mixer was analyzed on the Spectrum Analyzer. V_{tune} of the VCO was slightly adjusted again to give a Mixer output of exactly 50MHz. (Although the output was only 50MHz when a high bit was being transmitted, the identifier was transmitted so quickly that only a slight flicker indicated that the 50MHz signal was not always present.)

From here, the output of the mixer was sent through a series of four ZX60-3018G-S+ amplifiers. Without these amplifiers, the signal was too weak to later be demodulated. These four amplifiers, the amplifier before the Mixer, and the VCO were all powered by a 14.6V battery. This battery was first run through a 7812 voltage regulator to get a regulated 12V source. This regulated 12V was used to power the four amplifiers after the Mixer. This 12V was also supplied to the aforementioned voltage divider containing the potentiometer in order to supply the V_{tune} for the VCO. Finally, the regulated 12V was then run through a 7805 to supply a regulated 5V source. This 5V source was used to power all of the amplifiers and the VCO.

A 50MHz center frequency bandpass filter was placed at the end of the chain of amplifiers. A bandpass filter was used in order to attenuate or even remove any noise that would interfere with the desired signal. The chosen filter had a passband from 41MHz to 58MHz. Any frequencies within this range are left essentially unaffected (that is, the loss for frequencies in this range is less than 1dB [1]. At 11.5MHz and 200MHz, the loss is greater than 10dB, and further out at 4.1MHz and 450MHz, the loss is greater than 20dB.

The signal at the output of the bandpass filter was viewed on an oscilloscope. When the settings on the oscilloscope were changed such that the horizontal axis was on a time scale of 10ms/div, the beacon identifier could be easily recognized, as shown in Figure 4.2.5. Although the identifier was recognizable at this point, it was still impossible for the microcontroller to read the correct identifier from this signal. As a result, it must first be demodulated.

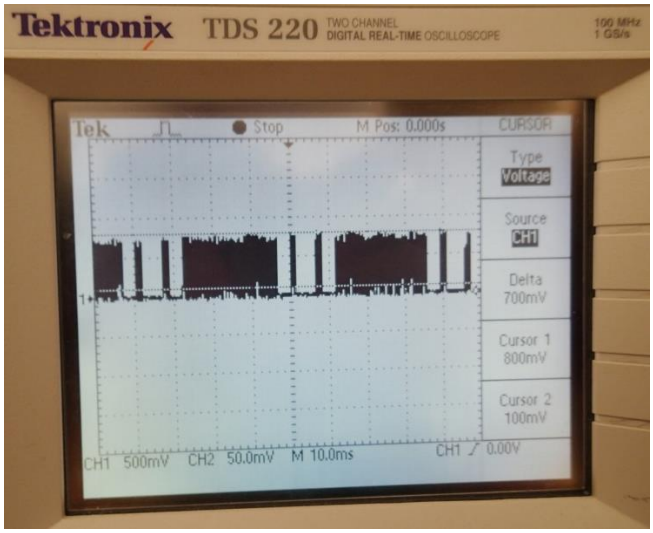

Figure 4.2.5: FSK Sinusoid of Identifier

4.2.2 DEMODULATION

The signal coming out of the bandpass filter is sent through a demodulation circuit. This circuit is made from a few simple components: a rectifying circuit consisting of a resistor and a switching Schottky diode, an operational amplifier, and a comparator. This circuit can be seen in Figure 4.2.6. The purpose of the demodulation circuit is to essentially reverse the FSK modulation now that the signal has been down-converted to a workable frequency.

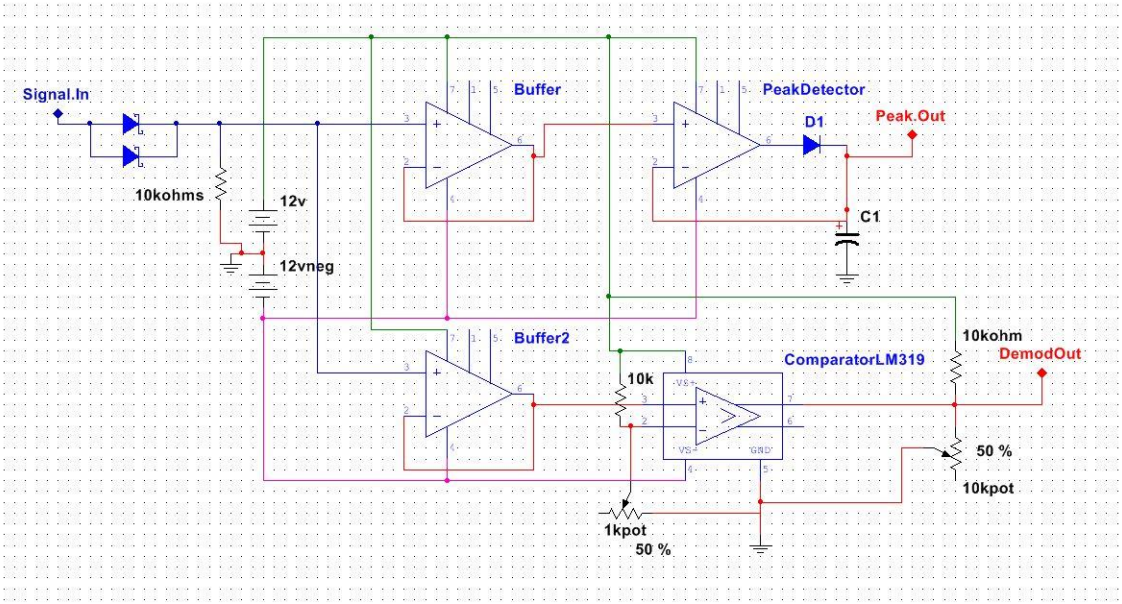

Figure 4.2.6: Demodulation and Power Detection Circuit

The rectifying circuit is a simple half-wave rectifier. The positive parts of the sinusoid remain untouched whereas the negative parts of the sinusoid are blocked, and thus are instead seen as 0V. At this point, the signal is split into two parts. The first part continues through the rest of the demodulation circuit. The other part of the signal is sent through a peak detector circuit which will be discussed later. The rectified signal for demodulation is then sent through a voltage follower using the LM741 operational amplifier. The LM741 has a slow slew rate of .5V/us [2], which means it responds slowly to quick voltage changes in a circuit. Resultantly, this causes the amplifier to work as a low-pass filter and smoothes out the rectified wave, making it more similar to a square wave. Figure 4.2.7 illustrates this. The brown waveform is the half-rectified wave, and the blue signal on top is the smoothed-out signal.

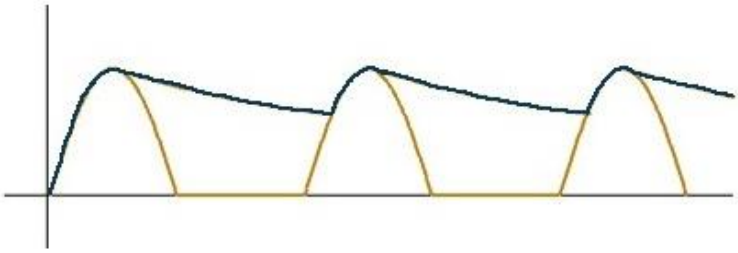

Figure 4.2.7: Half-Wave Rectified Waveform

After the buffer, the last stage in the demodulation circuit is a comparator, the LM319. A comparator is a circuit component that returns a high voltage when the input voltage is above a certain threshold and returns a low voltage when the input voltage is below a certain threshold. On the Soteria Robot, the threshold is set to nominally 600mV. This threshold is tuned with a potentiometer to get the value with the best performance. The low voltage is returned as 0V and the high voltage returned can be set by a potentiometer to be any voltage between 0V and 5V. The high output level adjustment is useful when changing between different logic level devices, like the Arduino Uno(5V) and the Arduino Due(3.3V).

When the output of the comparator is viewed on an oscilloscope, the squarewave is nearly perfect and exactly matches the binary value set to be transmitted by the beacon. After outputting 15 high bits, the beacon then outputs a low start bit, one nibble representing the decimal number “2” repeated, and finally an even parity bit. This pattern then repeats indefinitely while the beacon is on. As can be seen from Figure 4.2.8, this is exactly what is read by the oscilloscope at the output of the comparator.

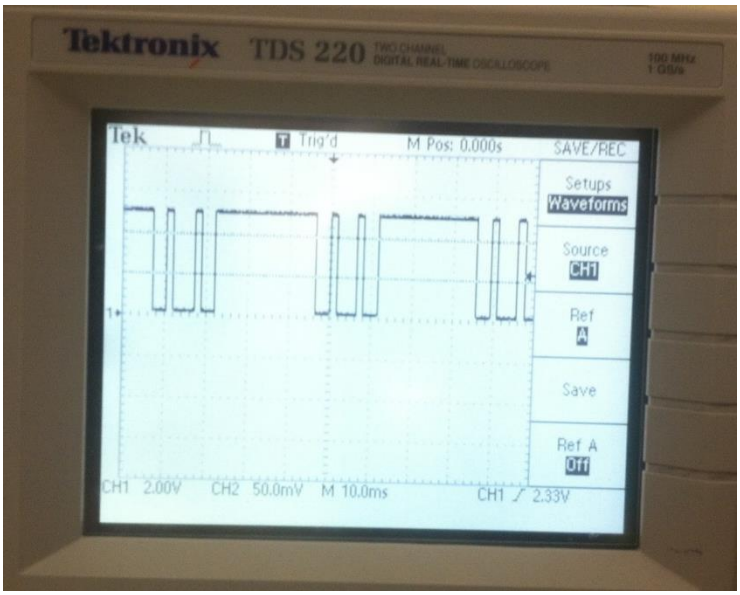

Figure 4.2.8: Demodulator Comparator Output

4.2.3 Beacon Identification

The output of the comparator is then sent into a digital port on the microcontroller. The digital ports on the Arduino Due can only take 3.3V, thus the output of the comparator had to be shifted down from 5V to 3.3V. This was done using the previously mentioned potentiometer.
The microcontroller then takes 72 samples of the comparator output at exactly 600Hz. This will yield exactly 1 number per bit of information transmitted. The 72 bits of data is enough to completely contain the identifier message 3 times. This will ensure at least one uninterrupted identifier sequence no matter when the Arduino begins reading the data coming in. The 72 bits of data is then turned into a string. The Arduino takes the data string and searches it for the identifier using the ".indexOf" command. 50 72 bit data strings are read and compared in order to develop a metric for quality of identifier matches. Some may question the reliability of this method of checking because the sampling rate is below the Nyquist frequency, however, during testing, whenever a good signal was being received, the 72 bit strings always perfectly contained the identifier message. This identification method has also been demonstrated to work effectively at 30m with the 3 element Yagi antenna.

4.2.4 Max Power Determination

The other half of the circuit in Figure 4.2.6 is the power detection circuit. The circuit takes a second buffered copy of the down-converted signal and sends it into a simple peak detector circuit. This specific peak detector holds the highest input voltage with a

time constant of .3s. This time constant was chosen to be very slow so that the zeroes transmitted in the identifier are not enough to noticeably drop the peak detector output.

This circuit is used when searching for the direction of maximum power received. The servo controlling the antenna stops after each degree change and waits 0.75s to allow for settling with the slow time constant. The peak detector output is then read using the built in analog to digital converter on the Arduino. If the peak detector output is higher than the previous maximum, the new value is saved as the new maximum and the position of the servo is saved as the new position of maximum power. After the servo has completed a search of 360°, the antenna rotates back to the position of maximum power and the FSK identifier is checked.

This part of the system proved quite difficult to test due to the amount of reflections caused by the nearby Workman building, inside and outside. However, with careful tuning of the VCO, the correct direction of the beacon was found repeatedly by this system. It was successful up to a distance of 30m.

4.3 NAVIGATION

The Navigation subsystem controls the movement of the robot as well as records the GPS location and compass heading of the robot.

4.3.1 CHASSIS

The structure that encompasses the robot, the motors, and the wheels were all provided by the New Mexico Tech Electrical Engineering department. The set came together as Polulu's Dagu Wild Thumper 6WD All-Terrain Chassis with 75:1 ratio motors (Figure 4.3.1). The chassis has independent suspensions for each of its wheels, allowing it to handle uneven outdoor surfaces. The chassis itself is big and bulky with a payload capacity of up to 11lb.

Figure 4.3.1: The Polulu Dagu Wild Thumper All Terrain Chassis [5]

4.3.2 H-BRIDGE

Responsible for the control of the motors are the Pololu high-power motor drivers (Figure 4.3.2), which were also provided by the New Mexico Tech Electrical Engineering department. These are MOSFET H-bridges designed to be able to handle high power motors up to 30V and 35A of continuous current. The H-bridge allows for bidirectional control of the motors, thus allowing for controlled turning capabilities. This H-bridge is designed for 5V systems and the minimum allowed logic voltage is 3.5V.

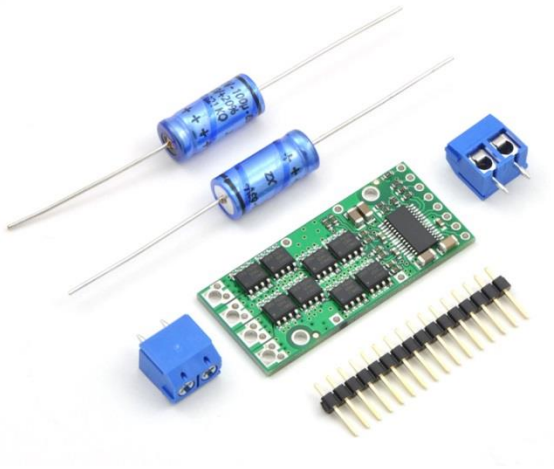

Figure 4.3.2: The Polulu High-Power Motor Drivers and Accessories [6]

4.3.3 COMPASS

One of the two major components of the navigation subsystem is the compass. The Honeywell HMC6352 digital compass was chosen for this design due to its heritage with

small robot designs, including a user calibration routine that was reviewed to be quite effective for compensating against electromagnetic interference due to DC motors, ferric chassis, and etc. for small robot projects. The HMC6352 digital compass was also chosen due to ease of use with Arduino microcontrollers boards.

Figure 4.3.3: The Honeywell HMC6352 Compass

The compass communicates with the microcontroller through I2C and reads the heading according to where the North arrow points on the compass as seen in Figure 4.3.3. The compass was used for navigating by programming the microcontroller to travel in a specific heading. The microcontroller would read the compass and compare the true heading of the robot with the programmed heading followed by adjusting the motors accordingly to set the robot on track with the heading. By this method, the robot could be programmed to travel in any desired heading, to which it was programmed to travel along the diagonal path in the Workman courtyard at a heading of about 121°. Before using the compass for navigation, however, the compass is always calibrated first so as to get accurate readings. By sending a command from the microcontroller to the compass, the compass enters into the user calibration mode. Afterwards, as recommended by the datasheet, the robot would do two full rotations over a period of 20 seconds. After running this calibration routine, the robot then continues with the navigation routine.

4.3.4 GPS

The other major component of the navigation subsystem was the GPS. The GPS choice for the design was the Venus638FLPx (Figure 4.3.4). The Sparkfun board with included SMA connector was bought for easy use with an active GPS antenna. This GPS was chosen due to heritage with past junior design projects as well as for being the best value for functionality versus cost when compared to alternatives GPS units. It has a cold start of about 29s, meaning that the GPS will start being reliable and start getting readings from multiple satellites after that amount of time. The GPS has picked up a maximum of 10 satellite locks in the Workman courtyard area where it was tested and generally reads accurate coordinates.

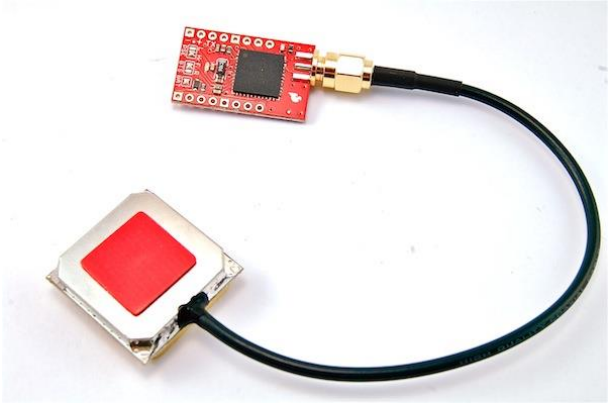

Figure 4.3.4: Venus638FLPx GPS with SMA Connector and Antenna [8]

The GPS communicates with the microcontroller through UART and sends NMEA sentences as strings to the microcontroller. The accuracy of the GPS was cited as 2.5m, however, actual testing showed that the inaccuracies of the GPS could go as far as up to 7m. As can be seen in Figure 4.3.5, the measured GPS coordinates of the robot were always in the vicinity of the actual path that the robot travelled, but never accurate enough to give a true position on the path.

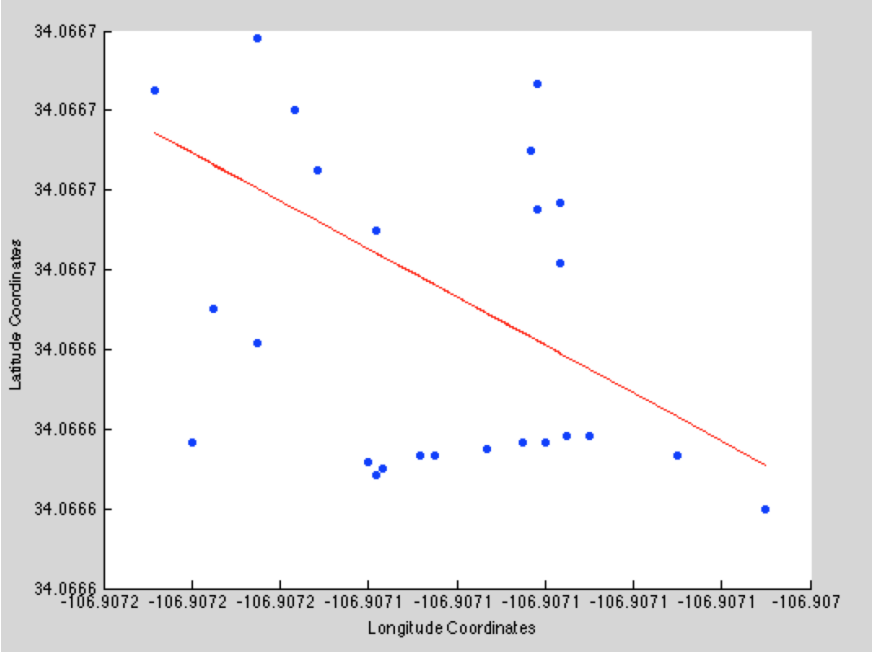

Figure 4.3.5: GPS Coordinates vs. Actual Path

Compensation for inaccuracies of the GPS is discussed next in the navigation routine.

4.3.5 DATA STORAGE

The robot's heading, GPS coordinates, and angle at which the strongest signal was found are all necessary components for determining the location of the beacon. All of this data needs to be stored in an appropriate storage medium that both has the space to hold the large volumes of data as well as allow portability between the robot and a computer from where the data can be extracted. The choice for the storage medium was a microSD card, from which we would write to using a SD card writer shield from Sparkfun (Figure 4.3.6).

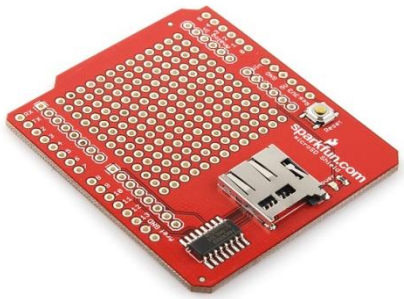

Figure 4.3.6: Sparkfun SD Card Shield [9]

The shield was made for Arduinos, thus allowing for the possibility of stacking and a clean setup. Also onboard the shield was a level shifter so as to protect the SD card from breaking under too high of voltage. The SD card writer communicated with the microcontroller through SPI and saved all data onto a text file.

4.3.6 NAVIGATION ROUTINE

The goal of the navigation subsystem was to have the robot travel in a straight line according to a set heading while periodically stopping to read and record the robot's heading and GPS coordinates. This was done by integrating the chassis, H-bridge, compass, GPS, and the microcontroller to work together. The microcontroller, being a 3.3V logic device, had to feed it's outputs through a logic level shifter before being able to command the H-bridge and control the motors. The bjt levelshifter circuit in Figure 4.3.7 was developed in a previous project. If 3.3V is applied to the input, the output of the circuit is connected the supplied voltage, in this case 5V.

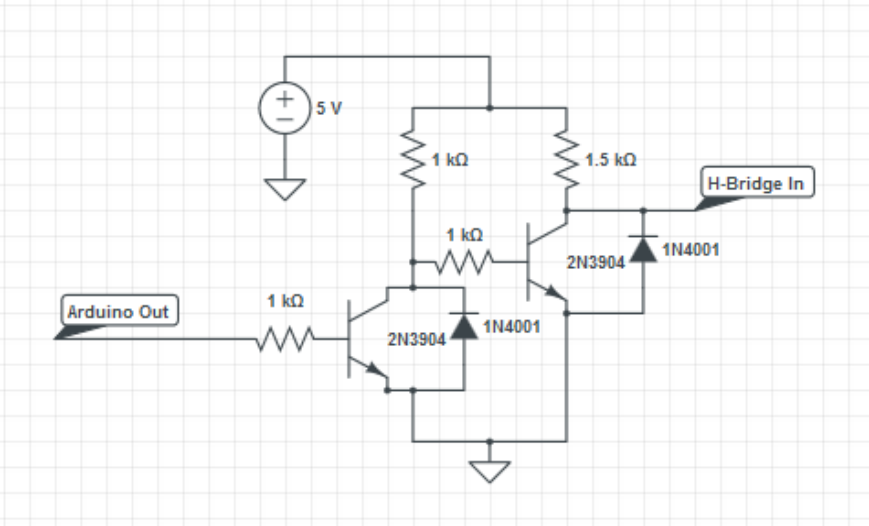

Figure 4.3.7: 3.3V to 5V Levelshifter

The compass, GPS, and SD card reader all communicates with the microcontroller via I2C, UART, and SPI respectively. The navigation routine always starts with the compass calibration routine as outlined in Section 4.3.3. Afterwards, the robot travels according to the set heading for a set programmed distance. The robot then stops and records multiple samples of headings from the compass and multiple coordinate readings from the GPS and saves them all onto the SD card. Multiple readings are taken so as to reduce the errors induced from the inaccuracies of the devices; being able to average multiple readings gives a more accurate reading of the actual heading and position of the robot. After taking a set amount of readings, the robot then continues the loop of travelling and stopping to take readings until a set amount of stops. The robot then stops and the navigation routine is considered complete.

4.4 Beacon Location

After the robot has completed its data collection run, the data is taken from the SD card and processed using MATLAB. The data is read into vectors and averaged for each measurement location. The GPS longitude and latitude values are by default read from the GPS in decimal minute format. In-house MATLAB code is used to converted decimal minute format to decimal degrees, which is the same format that is used by Google Maps and Mac OS Maps. The antenna servo position and the compass heading of the robot at each measurement location are added together to give the azimuthal direction of the beacon with reference to North. Using the point-slope formula, a line is generated for each sample location. The intersections between the 15 lines are calculated and averaged to develop a final coordinate location of the beacon. Figure 4.4.1 shows the position calculation program run with simulated data.

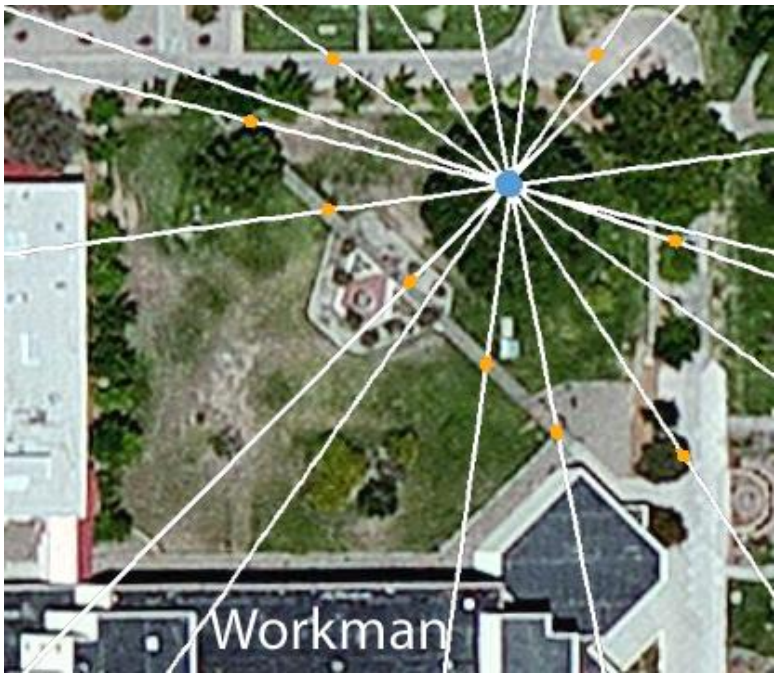

Figure 4.4.1: Location Program Output Overlaid onto Map of Course

The orange dots are GPS coordinates of 9 measurement locations. The white lines represent the direction in which the beacon was determined to be at each measurement location. The blue dot represents the actual location of the beacon. As can be clearly seen, all the white lines intersect near the beacon, and their average would definitely reside within 10m of the beacon.

This program was repeatedly run with ±10° standard deviation of Gaussian error added to the calculated azimuthal directions of the beacon. With nine data points and added error, the program calculated the coordinates within 10m of the beacon 60% of the time and was within 15m 90% of the time. It was determined that a metric of error could be calculated from the standard deviation of the intersection points that were calculated. However, it was never implemented due to the lack of live test data.

5.0 INTEGRATION

After the subsystems were built, tested, and shown to be fully functional, the components were combined together on the robot platform. During this stage of the project major setbacks were encountered. These problems ultimately prevented total system integration before the May 1st deadline.

5.1 HARDWARE INTEGRATION

Figure 5.1.1 displays the final robot with all subsystems attached and connected.

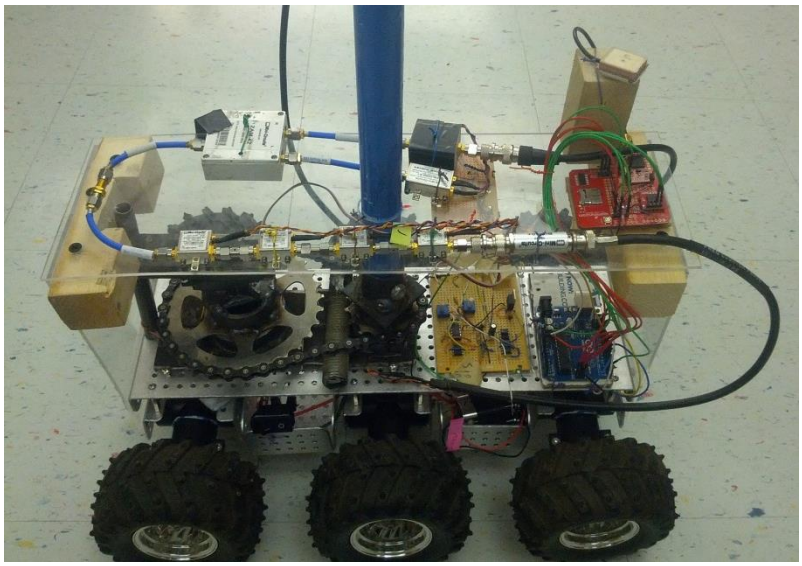

Figure 5.1.1: Populated Robot

Due to the large amount of real-estate required by all the subsystems, a platform was built above the robot chassis. The platform was made of plexiglass and wood. These materials were chosen because they did not conduct and the components could be mounted directly to the surface without risk of shorting. The plexiglass was also chosen because it can be see-through. This made trouble shooting and connecting wires much easier.

On the top tier of the robot, the down-conversion circuit was mounted along with the red circuit board which contained the compass, GPS, and SD card reader. The GPS antenna was raised above the top tier on a small wood block to reduce possible interference. The power regulation circuit for the down-conversion circuit was mounted on the underside of the plexiglass. Mounted directly on top of the chassis were the antenna chain drive, the demodulation and peak detection circuit, and the microcontroller. The batteries and h-bridges were housed within the chassis. The robot, while very stable and hardly affected by wind, was too heavy for the suspension to maintain unloaded ride height. The suspension was hardened by adding a wire link between the pairs of axels at the front and the back. The link prevented the axle separation which had allowed maximum suspension droop.

5.2 SOFTWARE INTEGRATION

The first step in software integration was to combine the functions for each subsystem into one program setup to run the search routine. Minutes before running the combined code and four days before May 1st, the Arduino power cable was dropped onto the Arduino Due. This resulted in a short which disabled the processor.

Due to the limited time remaining in the project, it was decided to use an on hand Arduino Uno the run the robot. The Uno had exactly 2 more digital pin than was required by the robot. Also, after the elimination of some global variables, the full program used only 86% of the total memory of the Uno.

Unfortunately, it was quickly discovered that there was overlapping pin use between the SPI and Servo libraries being used by the final program. An attempt was made to right a new servo library that did not use pin 10, SPI chip select. The new library turn out to be too much of a time sync. Instead, two programs were written in order to demonstrate the functionality of all the subsystems. The first program had the robot calibrate, navigate with the compass, and stop save GPS and compass data to the SD card. The second program would rotate the antenna 360°, return to the direction of maximum power, print the servo position, and print the number out of 50 identifier checks which were matches.

5.3 FINAL TESTING

By running the first program – navigation program – and marking the position and orientation of the robot at each stop, then returning to each position and running the second program – RF program – one run of live test data was created. The average GPS coordinates at each of the 15 stops are plotted in Figure 5.3.1.

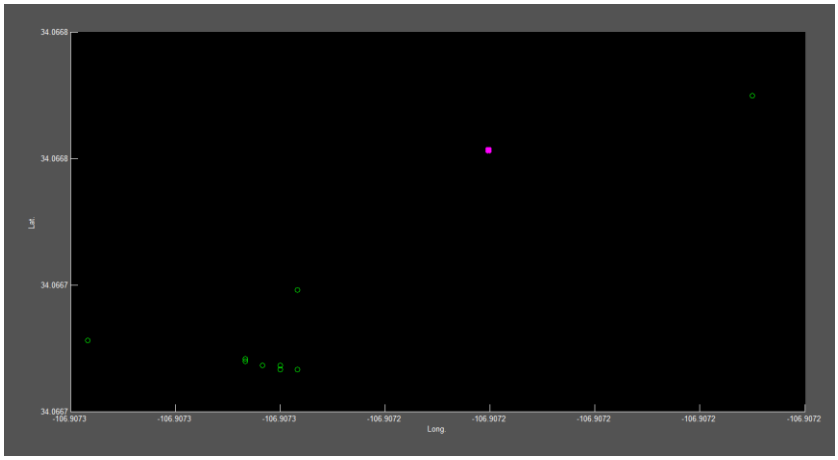

Figure 5.3.1: Live Test GPS Data

The green circles represent the coordinates of each data collection point. The pink dot represents the beacon location. As can be seen in Figure 5.3.1, there was a lot of overlap of GPS coordinates. This means there was a lot of error in the GPS data from the run because each data collection point was approximately 7ft apart. 7ft is about the minimum resolution of the GPS. It is believe that if the space between the collection points is increased, the data will improve.

The direction of maximum power was added to the GPS data. Then the data was run through the location program described in Section 4.4. The output plot is Figure 5.3.2.

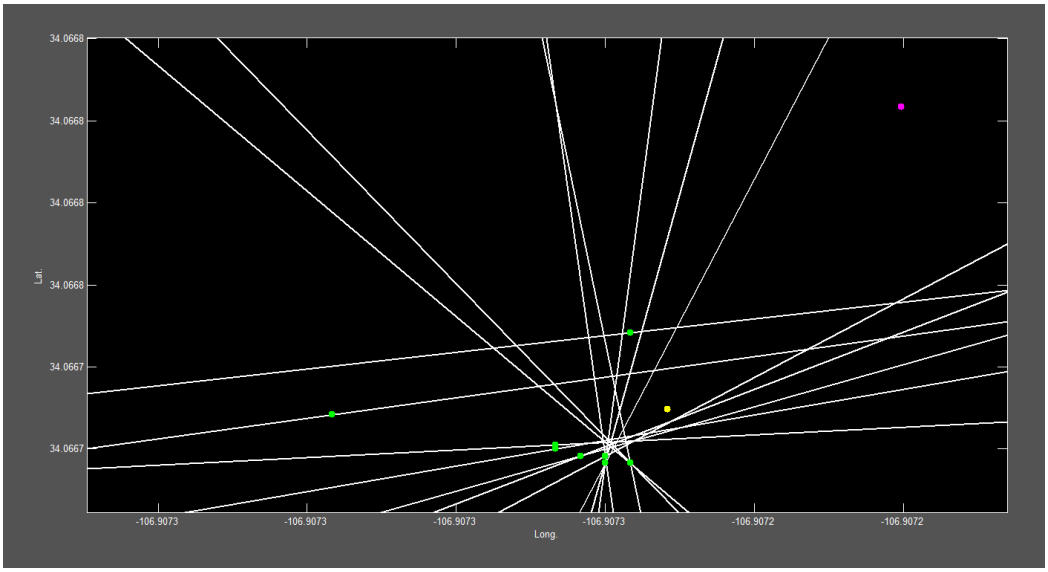

Figure 5.3.2: Live Test Beacon Location

The yellow dot represents the calculated beacon location. The separation between the beacon and its calculated position was 15m. This is 5m outside the desired 10m radius. If the GPS coordinates had been more appropriately spaced apart the line intersections would have been much closer to the actual beacon location and may have resulted in better calculated coordinates. With only one set of live test data, GPS accuracy appears to be the limiting factor.

6.0 CONCLUSION

The Soteria Beacon-Finding Robot was designed for the purpose of locating and identifying a RF beacon emitting an FSK modulated signal. Due to the destruction of the microcontroller before the integration process, a fully autonomous robot capable of finding the beacon was not completed, however, each of the subsystems were able to fully function separately and accomplish the following: sucessfully identify the beacon up to 30m, determine the direction of the beacon at a distance up to 30m, autonomously traverse an unobstructed area using a compass, save test data from the GPS and compass, and caclulate the location of the beacon within 15m. These results prove the validity and feasablity of the presented design solution.

With additional time and a new Arduino Due, it is strongly believed that a fully-functioning autonomous beacon-finding robot could be completed. Also, even with the current state of the project, an increase in the distance between data collection locations should improve GPS data. In combination with this improved data, a standard deviation based correction factor could also be quickly added to the location calculation program. These added measures would increase the accuracy of the calculated beacon location and could be fairly easily implemented.

For anyone continuing this design, it is recommended that these design additions be implemented.

Figure 6.0.1: Soteria Robot

7.0 REFERENCES

1. Mini-Circuits. *Coaxial Bandpass Filter, BIF-50+.* Retrieved from: http://www.minicircuits.com/pdfs/BIF-50+.pdf
2. http://www.ti.com/lit/ds/symlink/lm741.pdf
3. Rectifiers and Filters. *Kshitij School.* Retrieved from: http://www.kshitij-school.com/Study-Material/Class-12/Physics/Alternating-current-circuits/Rectifiers-and-filters.aspx
4. Short Antenna Figure. *Wikipedia.* Retrieved from: http://en.wikipedia.org/wiki/File:A8-8.jpg
5. Dagu Wild Thumper 6WD all-terrain chassis, silver. *Polulu.* Retrieved from: http://www.pololu.com/product/1561
6. Pololu high-power motor driver with included hardware. *Polulu*. Retrieved from: http://www.pololu.com/product/758
7. Compass Module - HMC6352. *Sparkfun.* Retrieved from: https://www.sparkfun.com/products/7915
8. Venus GPS with active antenna. *Doctormonk.* Retrieved from: http://www.doctormonk.com/2012/05/sparkfun-venus-gps-and-arduino.html
9. microSD Shield. Sparkfun. Retrieved from: https://www.sparkfun.com/products/9802

8.0 APPENDIX

8.1 FINAL BUDGET

Purchased Parts

Part	Quantity	Price per Unit	Total
HMC6352 (Digital Compass)	1	$34.95	$34.95 + $14.17 shipping
Antenna GPS Embedded SMA	1	$11.95	$11.95
Venus GPS with SMA Connector	1	$49.95	$49.95
Arduino Due 32bit ARM Microcontroller	1	$49.95	$49.95
microSD Shield	1	$14.95	$14.95
Bandpass Filter 50 MHz Center Freq.	1	$56.95	$56.95
Schottky Switching Diodes Chip	2	$0.67	$1.34 + $3.21 shipping
7ns Ultra-Fast Comparator	2	$4.56	$9.12 + $3.21 shipping
SOT-363 Breakout Board	1	$1.92	$1.92 + $3.21 shipping
Misc.	N/A	N/A	15
Total	**11**		**$269.88**

Given Parts

Part	quantity	Price per unit	Total
Dagu Wild Thumper 6WD All-Terrain Chassis	1	$249.95	$249.95
Pololu High-Power Motor Driver 18v25	2	$49.95	$99.9
2S1P 6.6V 2300mAh Receiver Pack	2	$35.95	$71.9
ZFL-2500+ Mini-Circuits 500-2500 MHz amplifier	2	$99.95	$199.9
ZAM-42 Mini-Circuits 1500-4200 MHz frequency mixer	2	$54.95	$109.9
BBP-10.7+ Mini-Circuits 9.5-11.5 MHz bandpass filter	2	$40.95	$81.9
ZX95-2400A+ Mini-Circuits 2000-2400 MHz voltage controlled oscillator	2	$40.95	$81.9
Total	**13**		**$895.35**

8.2 PROGRAMS

All of the code written for the project is saved to the accompanying USB drive.