A.L.L.E.N.

Autonomous Logical Land-Based Electronic Navigator

Author: Camille Arnn

camillearnn@yahoo.com

Trinidad State Junior College

Faculty Advisor: Ms. Cindy Clements

Team Members: Hayden Alworth, Camille Arnn,

Mary Carpenter, Andrez Leyva, and Eric Perry

April 7th, 2014

Abstract

Our main goal in creating our robot for the Robotics Challenge was to create a small, enclosed robot to successfully roam through any course consisting of various obstacles that could be avoided or climbed using a whisker/bumper sensor and a flexi force sensor. We wanted to utilize our own original ideas, but incorporated some past ideas that we believed we could improve upon.

For the robot's shape we created a design consisting of a dome sitting upon a trapezoidal box. The dome enabled us to fulfill our goal of an enclosed robot. The robot's enclosure offered protection from sand intrusion in the breadboards, compass, beacon receiver, motors, and other small parts that needed safeguarding. In the context of climbing, we found the trapezoidal box to be superior to the previous rectangular boxes of past years.

With our chosen design, some obstacles pose greater risks for us than others. Without much clearance on the bottom of the robot, we had to compensate for rocks that could possibly cause our robot to high center through different sensors and programming. We decided to incorporate a flexi force sensor along the bottom of the robot body in hopes of avoiding any obstacles that pose a possible high centering threat. This small sensor that lies against the front of the robot body measures the amount of force being exerted on it.

Improving upon previous wheel designs and utilizing 3D technology, we created a wheel with a cleat-like design. After testing, we added pieces of sensory balls with small, rubber knobs to the arches of our wheels for the purpose of traction on different surfaces to allow for the climbing of small obstacles and even some obstacles comparable to its own size.

After deciding to recycle the design of the previous year's bump sensor, we began looking for areas of improvement. We recognized their error in creating a sensor platform that extended too far past the front of the robot; this proved to be a critical issue

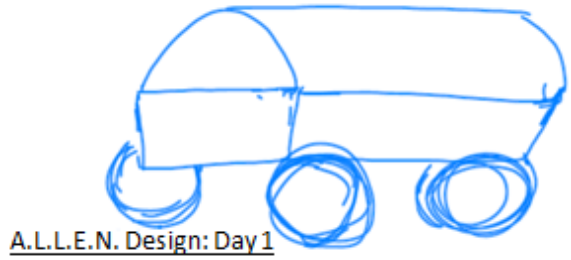
in downhill climbing. We recreated their design using three trapezoidal shapes to act as the bumpers that trigger the micro switches in place of their four shapes. Our sensor platform is much closer to the robot's body and the platform is vertically adjustable to allow for modification during the testing process and in the field.

Although pit avoidance was still something we wanted to accomplish, we had tested and determined we were pleased with A.L.L.E.N.'s navigation and ability to avoid obstacles. The remaining time we had prior to the challenge was spent determining what it would take to develop a working infrared navigation. Though it is no simple task, A.L.L.E.N. is now capable of recognizing holes in its path. Upon recognition of these holes, the robot then begins navigating in a way that allows for obstacle avoidance.

From the beginning of the design process, and throughout the building and programming process, we decided that instead of starting from square one, we would take what past teams have done, and build upon it, to create a robot more technologically and mechanically advanced as to outperform and outlast other robots.

Introduction

Initially our design included four wheels, a case of some sort used for sealing our robot from the sand and other outdoor environmental issues that may arise, and a dome to house our breadboard, compass, switch and other components. This design was partially altered, one step at a time as we continued to work toward achieving the best robot possible. We discontinued use of a wagon type design and opted for a more functional design consisting of the dome and trapezoidal box, figuring this would offer us the most accommodation and flexibility to achieve our goals in the least amount of material.



A.L.L.E.N. Design: Day 1

Design Process

Wheels

Although we decided upon use of the same motors, (the Vex 2 wire Motor 393), last year's robot design, C.A.T., used we hoped to have fixed the problem the previous team had encountered with internal overheating of the motors and a complete shutdown by incorporating a design that did not have tank like treads. We felt (because a wheel would require less work to rotate than a tread needed) recreating the original wheel design from two years ago would prove to be more beneficial and eliminate the problem of the internal temperature regulation setting these motors are preprogrammed with that caused the shutdown of the motors.

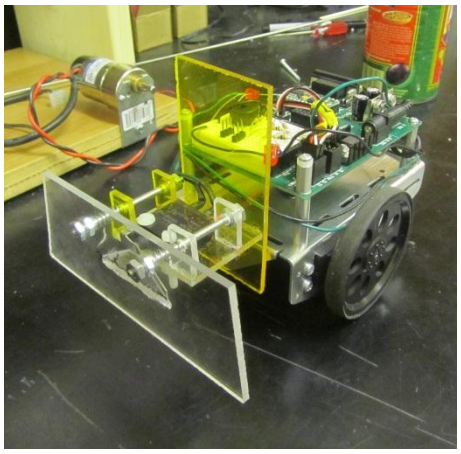
Using the CAD software, Solidworks, we were able to, initially, draw up what had been used in previous years, and from this point find weaknesses in the design and improve them to allow for the wheel to conquer more obstacles. Our team discontinued the use of hexagonal shaped wheels; we added two more lobes to the center, not only increasing the surface area contact with the ground, but also providing for a much more stability.

After the wheels were designed and printed with the application of 3D printing technology, in order to achieve our design of a cleat-like wheel, we cut and glued pieces of sensory balls that provided the small rubber knobs we envisioned. The addition of these pieces of rubber allowed for a great difference in the traction and ability to climb of our robot, compared to only the use of the slick ABS printed material.

Whiskers

The previous team faced obstacles with their design of a whisker using microswitches; this malfunction crippled their ability to descend downhill. The whiskers were designed in a way that had them protruding out much too far in front of the robot, causing them to trigger unnecessarily at undesirable times. We incorporated a shorter whisker design and improved upon sensitivity of each plate.

Once the design was finalized, a small prototype was created on a Boebot (see below) and run to test whether the new, more compact design would function properly and as sensitively as desired. Once this prototype worked accordingly, it was then expanded into three plates that would later be connected to the front of our robot.



Had last year's whiskers not sat as low as they did on the front of their design, the unnecessary triggering on downhill descents could have been avoided. This mistake led us to incorporate adjustability in the design of our whiskers. We found early in the testing process that high centering would prove to be an issue for A.L.L.E.N. If we so chose, the sensors are capable of placement along the bottom of the robot. We found this placement would be problematic and counterproductive because lower placement makes the robot susceptible to unnecessary triggering during downhill descent, which was the issue we were trying to avoid originally. Along with the lowest position possible, there are two other potential placements of the whiskers.

Flexiforce

A flexiforce sensor was placed along the bottom edge of our robot because, after much testing, we found this sensor was the best fix for high centering. When a certain amount of force is applied to the acrylic strip that lies in front of the flexiforce senor, the robot then assumes it has encountered an obstacle and begins finding an alternative route. This small sensor allows for the robot to successfully roam, descend downhill, and prevents high centering, without having to unnecessarily place the whiskers along the bottom of the robot.

Sharp Infrared Sensors

In an effort to create a robot that successfully detects pits or holes in the sand, infrared sensors were crucial additions for the sensor platform on A.L.L.E.N. Sharp IRs, we found, were the only sensor that was able to give us accurate distance measurements in both sand and sunlight. After circuit

adjustments, the Sharp IRs were tested using a test program that allowed us to determine if holes existed in the robot's path using Sharp IR values. Our team was challenged with much debugging before the robot was able to successfully navigate based upon the values it received.

Programming

While there are a number of microcontrollers, such as the Arduino and Basic Stamp that can be used, our previous experience and the partnership shared by Trinidad State and Parallax, led us to use of the Parallax Propeller chip. We decided that this propeller chip's multifunction capabilities and reliability was more suitable for our robot's brain. The Parallax Propeller chip is capable of doing eight things simultaneously with its eight cogs; this was a vital aspect our brain needed due to the structure and complexity of our program.

The initial steps writing the program that would allow A.L.L.E.N.'s many features to work properly and simultaneously, started with learning the Propeller C language and writing a working whisker sensor program. We then progressed toward adding corner detection, compass, and accelerometer. These were all added to the program separately and each was tested until working properly. Accurate beacon navigation was added next; many issues arose in the process of a robot that, not only receives and reads values from the beacon, but also properly reacts and navigates systematically toward the designated destination. In the final steps of putting together a working program, sharp IR sensors were added to the many features the program offers.

The navigational goal of the program is to successfully check all of the sensors; if all of the sensors are undisturbed, the goal then becomes to head north or toward the beacon. The robot will do this until any of the sensors register that an obstacle has been detected, in which case A.L.L.E.N. will maneuver in whichever manner is indicated by the sensor that detected the obstacle.

With all of the sensors on the robot: the whiskers, the Sharp Infrareds, the flexiforce sensor, and the different navigational tools: Northern navigation with the compass and navigation using the beacon receiver, the program required the ability to process all of this information quickly. Originally, the program called on many different functions every time values from the sensors were received. The manner in which the robot performed during all of this was not as fast as it needed to be. Once the entire sensor reading code was placed in one of the Propeller chip's cogs, this portion of the code was then allowed to run continuously. Placement of this portion of the program in a separate cog allowed for the main program to have these values readily available as opposed to waiting on a function to initialize the sensor then retrieve the reading, allowing for access to up-to-date sensor readings whenever necessary.

There are a number of features that set this program apart from past programs used by previous teams. While the propeller chip was used last year, the C language software was not fully developed. Along with a more developed C language software, we also had a programmer capable of writing his own libraries.

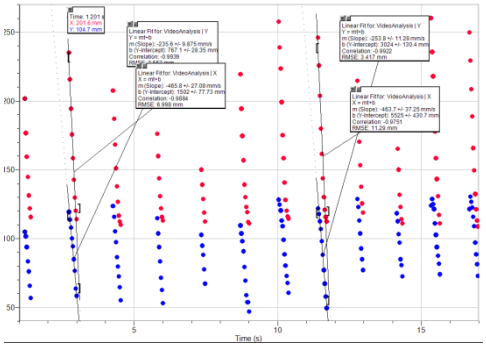
One of the main features this program offers is accessibility. Each of the many different features of this robot can be enabled and disabled using the whiskers and flexiforce sensor to communicate with

the program. Also, each portion of the program is written so that variables can be adjusted simply and quickly. This can be seen in the structure of the program as a whole. After it was written and working, the program was restructured in a way that separated the navigational portion of the program from the main event handling portion. This proved to allow for quick changes of any of the values that needed adjustment during the testing process.

Use of Technology

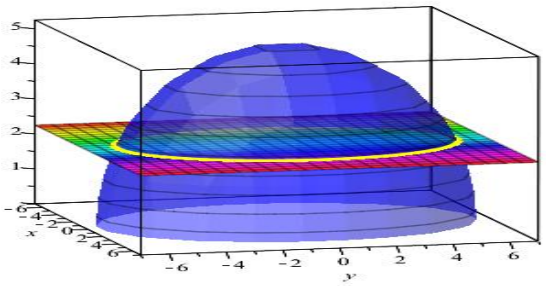
Logger Pro

In an attempt to fully optimize our robot's rotations per minute and speed, and also to compare these results when the robot is climbing rocks and maneuvering around other obstacles, we recorded the robot's movements while it was in a mock testing sand pit, and loaded these videos into Logger Pro to get exact readings. Logger Pro is a computer software program that is generally used for data compilation and analysis of this data. These computations and much testing were then used to determine the best value to be entered into our programs for the robot's RPMs. Since our programmer was writing the new libraries for the motor drivers, Logger Pro also allowed him to see if his program was sending the right signals to the motor controller.



Maple

Because our original dome wasn't the right size for our robot, we used Maple, the computer algebra software, to graph a model of the elliptical dome we had bought, and the space curve created when the dome was cut. Considering our tallest object, the beacon receiver, we were able to plot the intersecting plane with the dome that would represent the cut that would be made on the dome to ensure that a precise slice was made.



SolidWorks

Much of our designing process took place on the computer, using the CAD software, Solidworks. As well as quick and accurate designing, everything printed using the 3D printer was drawn using this software, so it was crucial for designing.

Because of the more complicated design of our whiskers, it was easier, and more accurate to draw them using 3D software to find exactly where parts would line up. Our wheels, as previously mentioned, were also designed and drawn using this program. Solidworks allowed us to design and print quickly and efficiently.

3D Printing

With the new addition of our 3D printer, we were able to create parts of our robot that in past

years we had not been capable of creating. For example, we had to rely on others to manufacture the wheels and couplers we were able to produce ourselves this year.

Although it helped immensely, we did encounter issues with the printer. The printing process for each wheel required 23 hours, increasing the possibility of loss of power, and also occupied the printer for an extended amount of time. When the printer did lose power, it did not resume and complete the print and the final product was what had thus been printed up until power was lost.

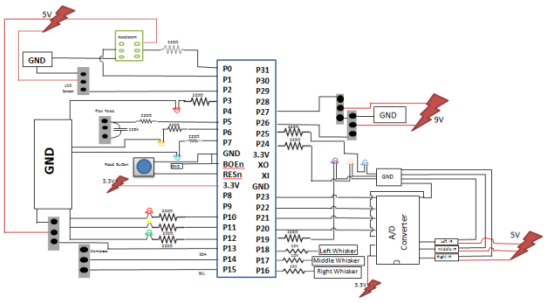
We turned to the printer to help with small measurements that needed to be very precise when it came to aspects that needed to be exactly level. All of the adjustability we had with our sensors was made possible with brackets designed on the computer, printed off, and attached to the side of the robot's body. This precision can also be seen in the small parts that hold the pistons at their correct height on the whiskers.

Every part of our program gives off readings and values that allows us to see how the program is interpreting the data it receives. In an effort to make reading the LCD screen more accessible while it's located inside our enclosed robot, we designed an LCD screen mount that form fits the screen and allows for the screen to be held at the angle of our choosing.

Pitfalls found during testing

Upon original placement of the wheels, we found that the axles weren't exactly aligned; this caused instability in the wheels that foreshadowed the possibility of more trouble later in the field. In an effort to fix this defect, bushings were added inside the main body of the robot to each of the axles, creating a stable robot.

Testing was delayed greatly due to the complicated circuit of our H Bridge Motor Driver circuit where we found much of the wiring was coming unplugged. This motor driver was replaced with a Vex corporation model 29 and this Vex motor controller was incorporated into the wiring; these motor controllers helped to clean up the board greatly and enhance our ability to keep the wiring tight and the circuits complete. The inclusion of the motor controllers proved to be crucial with the addition of the IR sensors later in the design process. Had they not been present on the board, there would not have been room to include the necessary ADC chip for the Sharp IR circuit.



The original ramping of speeds in our program was too slow to allow the robot to gain enough momentum to conquer obstacles. It wasn't until we completely eliminated this feature that we were able to achieve our full potential and full speed in the field. Logger Pro was used to track these various speeds, and find the best speed for our desired maneuverability.

We had a big issue with high centering because the radius of our wheel allowed for a small amount of clearance underneath the robot. Along

with the inclusion of the flexiforce sensor, a keel was added to the bottom of the robot's body to allow for the robot to lean itself back on its wheels if the issue of high centering did arise in the field.

We tested a different sensor in an attempt to fix our high centering problem. Prior to adding the keel, a small push sensor made by Vex robotics, was added to the bottom of the robot body. This did not perform to our expectations. It's bulky design and added weight did not seem worth what it offered so it was not included in our final design.

From the first day of designing, it was clear the only route we wanted to go in our robot's creation was a completely enclosed design. When we received the domes we had ordered from a company in Canada, we were glumly displeased with the fit of the dome on the top of the robot's trapezoidal body. Reordering the dome wasn't an option due to time constraints; we decided to alter what we had. Maple was used to find the cut needed to have a dome that fit our robot in the way we had imagined. After cutting the dome, a rectangular flange was added to the dome to allow for the dome to be removable.

When the 3D printer we had was in a long print the chance for losing power was more likely, also the printer wipes its memory when this happens. We learned this when, in the 18th of a 23 hour wheel print, the weather got unruly, and the building the printer resides in lost power and we had a partial wheel. This "skinnier wheel" was used as a test wheel, and we were forced to begin the print later when the weather wasn't an issue.

After adding the attachments to make the whisker sensor adjustable, we found the sensor was susceptible to getting stuck at an angle not parallel to the ground. We printed off triangular supports to be added to the top and bottom of the plate to keep this issue from occurring in the field when the robot was roaming.

After evaluating the values given by the flexiforce sensor, we found the sensor was being triggered by obstacles that could have been conquered by the wheels alone. Small rectangular pieces were added to the left and right sides of the acrylic used to push the sensor to lessen sensitivity.

In the development of getting the IRs working mechanically as a pit sensor, we found when they were facing directly forward, the IRs located on the left and right sides were not picking up obstacles in the way we wanted them to. We printed off a detailed and tailored mount after finding which angles the IRs needed be set at to work optimally. This eliminated the problem we saw with the side IRs not detecting pits.

The circuitry of the Sharp IRs alone proved to have many pitfalls. First off, the circuit had to be adjusted from an RCTIME circuit for the test program to one of our own making. This RCTIME circuit would not have worked in the field due to the inability it gave us to read any of the values received below the Propeller chip's threshold voltage of 1.65. Not being able to access the full range of output voltages offered by the Sharp IRs (from 3.3 V to .4) would have kept us from being able to detect pits, which was our entire reasoning for adding the IRs.

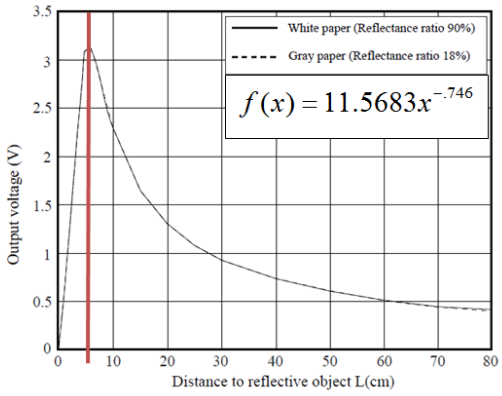
Because we could not find an ADC circuit that was able to read voltages above and below the Propeller chip's threshold voltage, we decided to build one that began with wiring the Sharp IR sensor's output wire directly into a Propeller I/O pin

and slowly adjusting the applied voltage from another of the Propeller's I/O pins to find the amount of voltage coming from the IRs. This worked when tested in the field but did not provide for values stable enough for detecting holes.

After a few failed attempts, it was decided an ADC chip was necessary for hole-detection. Before its addition on the board, the chip needed to be tested. Experienced to the difficulty of developing a working IR, the first test program written with this ADC chip did not function. It wasn't until the program was compared to one written in PBASIC language were we able to find that a change in the bit transfer modes would give us a working program. Finally, we were able to simply read the voltages from the Sharp IRs over their entire range and with high accuracy.

We ran into the obstacle of not being able to retrieve actual distances from the IR's values. We solved this issue through the observation of a Voltage to Distance graph found in the Sharp IR documentation (see below). Plotting the graph from its peak and running a power regression modeled the exact function that would allow for us to successfully retrieve distances.

Fig. 2 Example of distance measuring characteristics(output)



During the process of testing our IRs, the 5-volt regulator on our circuit board began overheating. Assuming a wiring issue, the board was quickly turned off and the circuits were checked; everything was right. After considering the amount of current each of the three IR sensors required, 30 mA, we found too much current must have been drawn through the 5V regulator. Although the total 90 mA of current drawn by the three IR sensors wouldn't have caused this overheating issue alone, we concluded the total amount of current being drawn from the circuit as a whole was too much for the regulator. Overheating was a new issue for us mainly because past years have never created a robot possessing so many sensors. After adding another 5V regulator for the sharp IRs, this issue did not reoccur.

With two regulators, no overheating, successful distance readings and IR values, the program should have been working flawlessly, but it was still encountering issues. The program was running normally and then it would freeze. This strange behavior in the program occurred a number of times. When each part of the program was tested individually, it worked properly. The problem was narrowed down to an issue residing in the portion of the code that converted the Sharp IR voltages to distances. After testing if the problem was a function in the programming and not getting any behavioral changes in the readings, we noticed the compass also started giving off very suspicious readings.

We discovered the problem we had with the program freezing and the compass giving off a strange reading only occurred when the compass and IR reading code were in different cogs. This occurrence led us to believe these two floating point

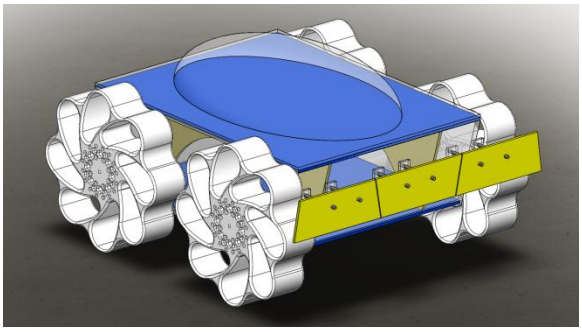
operations could not be done simultaneously. After our programmer developed two test programs to observe the way different floating point operations are able to perform while in different cogs. He found when one cog caught up to the other cog, they both corrupted the values and froze.

After Andy Lindsay, the author of the Propeller C libraries for Parallax, was notified about what was occurring within our program, he instructed our programmer, Hayden Alworth, on how to fix the problem within Lindsay's own library source code. Since fixing this code and placing both portions of our code in one cog, the program has been free from freezing and the compass has read properly.

A.L.L.E.N. has gone through very much testing in the process of his creation. In our final days leading up the Challenge at the Great Sand Dunes, we found our rubber knobs added to the lobes of the robot's wheels had not held up as we as we had hoped they would have; essentially, they had been sanded smooth by the sand pit. Comparing A.L.L.E.N.'s performance in videos taken at the beginning of his tests proved we had lost traction and grip through so much use in the sandpit. All of the test runs had taken their toll on the wheels. We printed off new wheels at the last minute before the challenge in hopes this late change would give us back the grip and ability to climb originally possessed early in our tests.

Final Design

In all, using past teams' ideas and mistakes, along with our own original designs, allowed us to create a robot that successfully navigated around the sandpit and avoided obstacles while staying true to what we had originally hoped to achieve in the creation of our robot, A.L.L.E.N.



Conclusion

The Robotics Challenge at the Great Sand Dunes offered us the opportunity to face challenges we had yet to even consider and also a starting place for next year's team to grow from. We quickly learned clearance on the bottom of the robot is crucial for success. Without a keel that kept the robot's weight maintained on its wheels when it got stuck on an obstacle and also a program that compensated for this issue when it arose in the field, A.L.L.E.N.'s success would have been much more limited.

To start the challenge off, I n course 1 we began with a beacon navigation that after properly working for about ten feet, startlingly took a right turn toward a path we had not anticipated. We changed our battery in hopes the issue was a small error in which we had overlooked an overused battery. Although the quick change of the battery did slightly improve our robot's behavior, it did not eliminate problems we had with navigating solely using the beacon. After changing the method of navigation, this course was completed without any other malfunctions.

In order to successfully complete most courses, the program was set to use the compass as its navigator. It was decided at this course, after many trials using the beacon, to traverse using the compass navigation. Our robot maneuvered successfully with the beacon navigation, until an undetermined point where the beacon was transmitting inaccurate values (typically, 90˚ or 270˚). These readings are what caused the strange behavior in course 1.

Course two was our team's opportunity to learn that not every problem we had thus far faced would be the only problems we could possibly face. Completely bewildered when one side of our wheels stopped working in the middle of a run, we found a small pebble had been lodged in between the wheel and the robot's body. Due to the fact that we did not have any small pebbles in our testing facility, we had not had an experience like this. Thankfully, this was a small issue in which we could simply remove the pebble, replace the robot and begin maneuvering through the field as we had been before. Had the robot's design allowed for more room where the pebble had gotten stuck, it's possible this problem could have been avoided altogether.

The third course offered us a great deal of trouble. The wall of rocks strategically placed directly in front of the beginning of the course posed a threat to our robot initially. Our program was equipped with a tilt sensor that kept the robot from mounting rocks that put A.L.L.E.N. at risk of flipping the robot on its dome. Without the interference of the tilt sensor, the robot was able to overcome the wall by maneuvering directly at the wall and climbing over it. Once past that obstacle, direct navigation was possible. Although typically helpful, we found the tilt sensor to be unnecessary for this particular course.

Due to the fact course 4 was very similar to course 5 (these were not performed chronologically) a simple change to the direction of our swiveling compass allowed us to maneuver through this course unscathed.

A certain congregation of rocks on the fifth course repetitively triggered A.L.L.E.N.'s flexiforce sensor. This continuous activation kept the robot moving in a loop for quite some time. Eventually, A.L.L.E.N. was able to move through a gap and out of the circle of rocks to complete this course.

Course 6, being the most densely populated with rocks, we anticipated much more trouble. A.L.L.E.N. was able to successfully find a path directly at the beacon, only pausing once on a mound of sand.

In summary, A.L.L.E.N.'s whiskers, flexi force, and Sharp IR sensors, along with its programming performed as planned. The robot's ability to read the beacon continuously proved to be our greatest obstacle. We also concluded the oversensitivity of the tilt sensor was unnecessary. While we found much success in the field, the various obstacle courses set up at the Sand Dunes offered us visual evidence of the importance of a well-rounded robot capable of using various sensors for obstacle detection.