AutoBot

Project Manager
Marissa Hintz
Documentation Chief
Phil Miller
Financial Manager
Derek Chopp

Project Engineers
Jake Kubisiak
Nikko Kolean
Brian Wilder
Myder Vang

The goal of this project was to design, build, and program an autonomous vehicle which can successfully compete in the Intelligent Ground Vehicle Competition.

Table of Contents

The Autobot team of Michigan Technological University would like to introduce the location and obstacle aware robot, “Bishop”. Though there have been improvements, this is the fourth and final year that the general design for this robot will be used. The mechanical design and hardware used has stayed fairly consistent throughout the lifetime of Bishop. Various small electrical elements have been added and the general electrical design has been improved. Most of the improvements have been made in the software, specifically in the navigation, avoidance, and line detection code. Though much of this robot comes from previous years and teammates, the robot that we are now presenting has been changed and improved to much better meet the competition requirements.

Bishop was designed for safety, reliability, ease of maintenance, to allow additional improvements, and to meet the competition requirements. To reach these goals, the robot was designed modularly. This allows each part of the robot to be tested and developed separately from the rest of the robot so that there is less possibility for error when the entire robot is running together.

There could be an entire book written on the adventures of Bishop. There have been many complications and changes throughout the development of this robot. Each new problem presented the team with a new challenge and forced the team to think of creative solutions using everything from duct tape to 3D printed models to software changes. For the sake of the reader, we will restrict the adventures to a summary. We hope that you enjoy learning about a robot that everyone on the team has come to hate and love concurrently.

The team was organized with one project manager, document chief, and a financial manager. The remaining members acted as project engineers. The project manager assumed the responsibility of assuring that the project would be completed on time by allocating resources and keeping the team accountable. The document chief organized the documentation and took meeting minutes. Finally, the financial manager maintained the budget and finances for the team. Each member of the team functioned as an engineer by designing, building, and testing the robot. The divisions of the roles can be seen below in Table 1.

Position	Team Member	Major	Year
Project Manager	Marissa Hintz	CPE	5
Documentation Chief	Phil Miller	EE	3
Financial Manager	Derek Chopp	CS/CPE/EE	4
Project Engineer	Jake Kubisiak	EE/CPE	4
Project Engineer	Nikko Kolean	CPE	2
Project Engineer	Brian Wilder	CPE	3
Project Engineer	Myder Vang	EE	4

Table 1: Member Information

Each member of the team worked on different aspects of the robot. Some focused on one specific facet of the robot while others with more versatile skills worked on many different parts of the robot. Jake Kubisiak, Derek Chopp, Phil Miller, and Myder Vang worked on the mechanical and electrical aspects of the robot. Marissa Hintz, Phil Miller, Derek Chopp, Jake Kubisiak, Nikko Kolean, and Brian Wilder worked on the software for the robot.

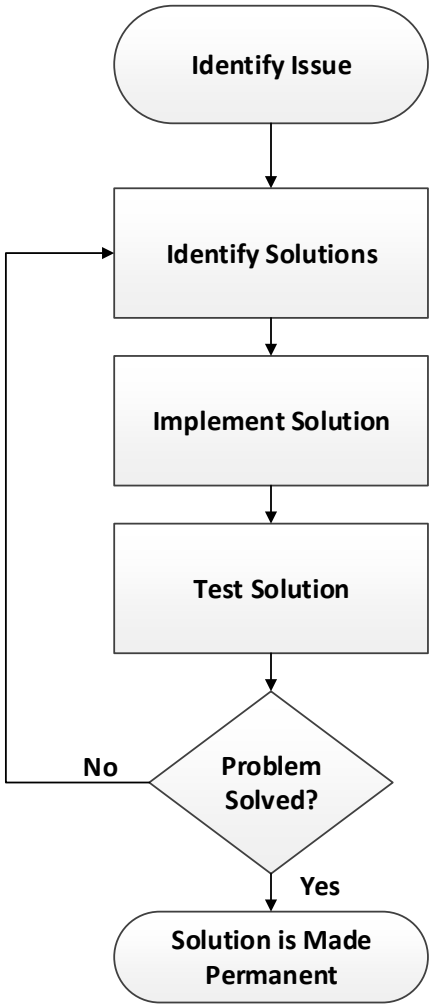
The team worked each week of the fall semester to complete the robot. Individually the members worked throughout the week on specific projects. Each weekend the team would meet, combine all of the parts of the robot, and test it together. This amounted to approximately 6 hours a week per team member or 42 hours a week for the entire team.

Bishop is made up of many different parts mechanically, electrically, and in software. Each of these parts needs to work together in order to achieve the final goal of a working robot. The mechanical body was built first to accommodate the electrical elements. It consists of a peninsula which holds the majority of the bulky electronic components and a mast which extends to allow components like the cameras to have a better view or better signal. Bishop has three wheels: a small one in the front and two larger wheels on the sides. A laptop was connected to the electronics to allow the software to read and analyze the data. The software takes all of the received data and processes it.

Most of the software design is focused on the navigation. Bishop's navigation is based off of its current GPS location, Inertial Measurement Unit (IMU) data, and what it reads from a map. This map is constantly being constructed using information from the Laser Range Finder and the cameras. Bishop will calculate a heading from the GPS and IMU data but before moving it will check the map. If an obstacle or line is shown in the map, Bishop will move to avoid it. When the x and y offset from the robot to the given GPS waypoint has reached a sufficiently low value, the waypoint has been reached and Bishop moves onto the next waypoint.

This is the third year that Bishop has been entered into the IGVC so it may seem that not much more design work was necessary. Although much of the main design was done by previous students, there was still much to be done by the current team of students. The robot that we began to work on had many significant design flaws which were identified at the competition in the summer of 2014. This team of students worked to correct the identified issues.

The team followed an iterative design process. First an issue would be identified. The team members who were assigned the issue would work together to come up with solutions. The solution that the team members considered the most effective would be implemented and tested. Following testing if the issue was successfully solved and no new issues were raised by the change, it would become permanent. If the solution failed to correct the issue or broke another aspect of the robot the process would begin again.



Design Overview

The robot has a height, width, and length of 72x29x43 inches. The central structure of the robot is the peninsula, which acts as the housing for the electrical components, two batteries, and the laser range finder positioned at the front. An 8020 rail acts as a mast extending upwards from the peninsula about 3 feet, providing a higher perspective for the 2 cameras, and bringing the e-stop in compliance with IGVC height regulations.

The majority of the material used for the construction of this autonomous robot is aluminum, with many of the parts being 8020 extrusion rails. Aluminum was used because of its high strength-to-weight ratio, relative inexpensiveness, and its ability to be easily machined. The 8020 extrusion rails provide a rugged structure, while allowing parts to be easily mounted or connected. The current design consists of a chassis containing two independent motors, with a peninsula in front containing electrical components and supporting a caster wheel. The peninsula is constructed of lengths of right angle aluminum. Directly behind the peninsula are the battery powered motors, which power the two 29 inch driving wheels that are mounted directly to the shaft via a hub assembly. Near the front of the robot is a smaller pivoting wheel, acting as a guide for the changing of direction. In the very rear of the robot is a dock on which the connected laptop can securely rest. As for the transportation of the payload, there is a hanging carriage below the peninsula where the payload will be carried during the competition.

Safety

Being that the current chassis design is one that has been reused from previous years, and has been used in competition during the 2014 IGVC; a number of issues had been identified, mainly revolving around the structural integrity of the chassis as a whole. It was shown that continued vibration of the chassis resulted in its weakening and eventual loss of various components.

To ensure that the mechanical design would not hinder the team's progress during testing or during the competition, numerous lock washers were added to the entirety of the robot. Testing has shown that the addition of a few dozen lock washers has effectively eliminated any structural concerns that were previously had.

Design Overview

The electrical design of the robot is based on a 12V bus. The robot takes power from two 12V-batteries in parallel to power the motors. Each motor, both left and right, is protected by a 40A fuse and a relay that opens when the current rises above that rating or a fault is detected. From the same 12V source, a boost converter is incorporated to convert 12V to 20V to power the laptop. Likewise, there is another spur off the 12V bus that is converted to 24V to power the laser range finder. There is also a voltage regulator feeding off of the 12V bus and converting it into 5V for the wireless e-stop receiver.

During the 2014 IGVC, a number of issues were identified. Taking the problematic features of the previous electrical design into consideration, a number of changes were made to increase the robustness of the system and to bring it into compliance with electrical standards. One of the issues identified was that a number of the wires were undersized and not terminated correctly. Another issue was that electrical components were soldered in line with the wiring harness and were unshielded making it possible for shorts to occur. There was also no apparent logical plan to the layout of the electronics or the wiring scheme. Additionally, the warning light showing that the robot was operating in autonomous mode was not bright enough to be seen outdoors. Finally, there was no way to tell if the batteries on the robot had reached a voltage too low to allow the robot to function correctly.

To address many of these issues the robot was dismantled and the electrical components were reassembled in an orderly fashion. This allowed for ease of access for maintenance and provided the team with the ability to expand if necessary. Terminal blocks were installed to distribute the 12V bus to the various components, which in turn provided proper termination points. Likewise, a grounding bus was implemented in the same manner. Since the motors can draw upwards of 40A, it was necessary to install 10-gauge wire to meet the current requirements. The remaining components required 12-guage wiring. Cable management was a priority with the new revision, so wiring was laid out and secured with zip ties.

A light tower was installed to replace the original warning light. This new tower provides much more visibility in the daylight, but required a custom designed board so that the software could interface with it. With the new light, the robot now has the ability to provide feedback of its health and operational status. In autonomous mode, a green light blinks continuously whereas in manual mode the light is solid. A yellow and red light are also incorporated to provide warnings of potential issues and errors.

In addition, a voltage regulating circuit was designed and fabricated to take the place of the components that were wired in line with the wiring harness. The combination of these design changes created an environment that was much more robust, and easier to maintain.

Electrical Components

A basic high level architecture can be seen in Figure 1 This highlights the sensors and controls currently used on the robot.

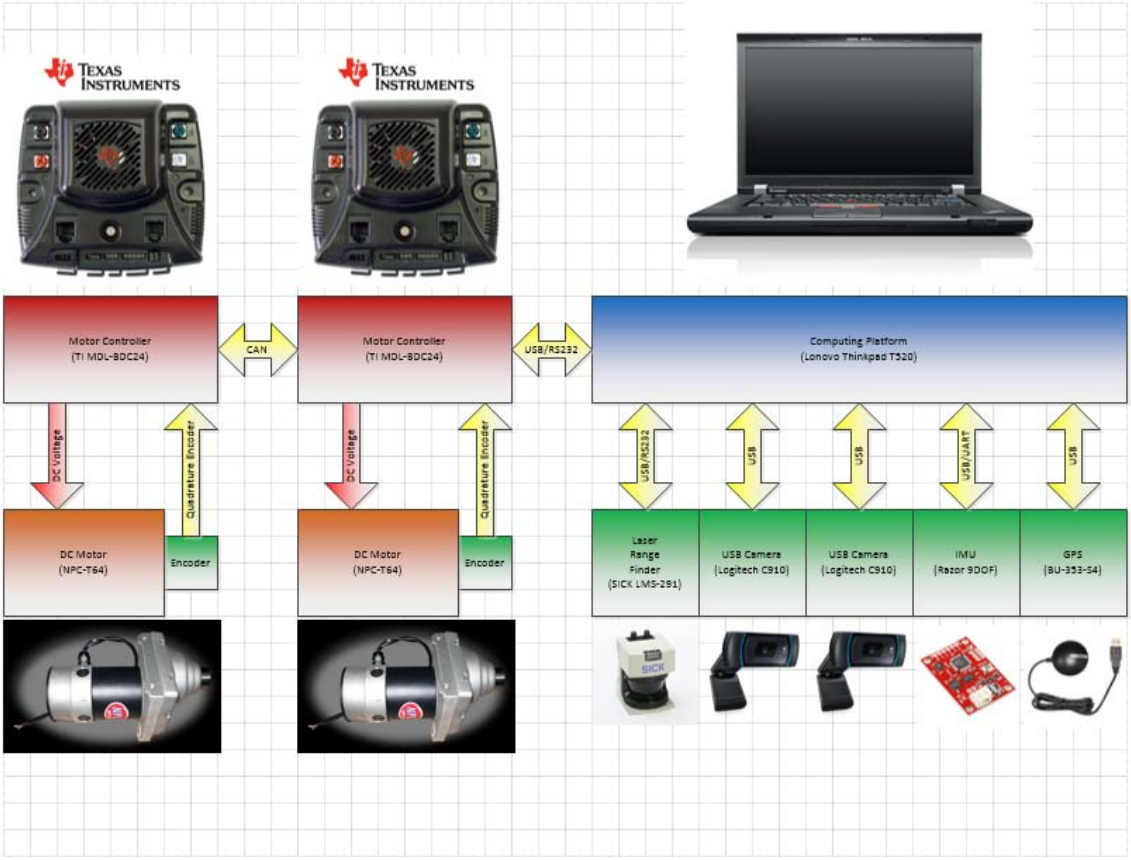


Figure 1: Electrical Layout

Computation Platform

The computation platform is a Lonovo ThinkPad T520 laptop. It features a quad core Intel Core i7-2720QM CPU running at 2.2 Ghz and 8 Gigabytes of RAM and runs 64-bit Ubuntu 12.04 LTS. All sensors data is collected by the laptop and then processed for localization, mapping, lane tracking, path planning, and object detection data. All the sensors are converted to USB to allow easy interfacing with the laptop.

Motor Controllers

The motor controllers are Texas Instruments MDL-BDC24. They are single channel, 40A continuous, 12-24 Volt motor controllers with a RS232/CAN interface allowing control from a computer. They can be networked to allow control of all modules from a single data line. We utilize the built in RS232 to CAN bridge to interface with a USB to RS232 converter to allow control from our laptop over a virtual serial port. The motor controllers also have quadrature encoder inputs to allow for closed loop control of the position as well as the ability to report the raw tick count back to the host. We also utilize the controller's ability to read the bus voltage for a low voltage alarm.

Motors

The motors used are NPC-T64 made by National Power Chair Robotics. They are 24 Volt DC motors with a 20:1 gearbox resulting in a 230 RPM output. Encoders are attached to the rear of the motors allowing positioning information to be sent back to the motor controllers.

Laser Range Finder

We currently use a SICK LMS-291 laser range finder which allows for accurate distance measurements up to 30 meters over 180° in 0.5° intervals and is positioned to scan parallel to the ground. Due to the voltage requirements of the LMS-291 it must be supplied by a 12 Volt to 24 Volt boost converter. It communicates back to the laptop through an USB to RS232 converter.

USB Cameras

Our current vision system relies on 2 Logitech C910 USB cameras mounted high to allow for a birds eye view after software perspective mapping. To get the maximum amount of coverage from the camera's without overlap, they are angled to the left or right with respect to the robot.

Inertial Measurement Unit

The Inertial Measurement Unit is a Sparkfun Razor 9 Degree of Freedom. It uses a kalman filter to combine data from a 3-axis gyroscope, 3-axis accelerometer, and digital compass to give us the robot's orientation with respect to the magnetic poles of the earth.

Global Positioning System

We use a GlobalSat BU-353-S4 USB GPS receiver. It contains the SiRF Star IV chipset and returns NMEA sentences at a rate of 1 Hz over USB. The robot speed is slow enough that the 1 Hz update rate is adequate to support the software.

Safety Light

We designed and built a custom board to handle the safety light. It allows control over 3 separate channels with separate timers for each channel. Each channel can handle up to 2 amps via a low side MOSFET and is upgradeable to 4 amps. It communicates over an onboard USB to UART bridge to allow emulation as a virtual communication port within our computing platform.

Efficiency

Due to the limited amp hour ratings of the batteries, power efficiency was a high priority in the design of the electrical system. Therefore, boost converters were used to convert the 12V bus to a 20V and 24V output due to their high efficiency ratings. To aid in power consumption when testing or during transportation, switches were terminated in line with the LRF and motors to disable them when they are not needed

Safety

Many precautions were made in regard to safety. With regard to the previous iteration of the robot, the exposed wires presented a shock hazard that needed to be addressed. By properly terminating the wiring into terminal blocks, this hazard was eliminated. IGVC regulations required both a physical and wireless e-stop that will disable all movement in the event of an emergency. The wireless e-stop will disable movement of the robot when a switch on a wireless remote is flipped or else if communication with said remote is lost. Movement is disabled by removing an enable bit to the motors themselves. The physical e-stop is mounted on the rear of the robot, approximately 3 feet off the ground, which provides easy access. This switch cuts power to the relays, which in turn shuts down the motor controllers entirely.

The biggest safety concern found at the previous competition was loose wiring which was causing shorts throughout the robot. Once again, terminal blocks increased the reliability and durability of the electrical system by eliminating the loose termination points. Overall, the system is much more sound and has a higher tolerance for vibration.

Software Strategy

With the type of autonomous robot required by this competition, software must be intentionally designed and well organized. The software running Bishop is organized into different modules and then called from a main file. The main file allows the user to specify which portions (cameras, LRF, GPS, etc…) of the robot they wish to run. This allows work and testing to be done concurrently on different aspects of the robot.

Since the organization of this project at Michigan Tech enforces continual turnover of engineers, one of the greatest challenges faced by the team was understanding and working with the code provided by previous students. To overcome this challenge, the team spent a lot of time working to understand the already written code. This was done primarily through flow charts and testing of functions individually. The flow charts provided an image of what was happening in the code which allowed the students to identify issues quicker and come to a more complete solution.

Computers

Before any code could be written, the platform, operating system, and programming language had to be decided upon. A couple constraints that were considered were the flexibility, reliability, speed, and availability of prewritten software.

The first decision to be made was the type of computing platform. There were two main options available: a distributed system, or a centralized computing system. A distributed system would include many different coded components that each managed one aspect of the project while a centralized system would have only one component that would manage the entire project's software. Our team decided upon the centralized system as it is much easier to modify and update. The distributed system may introduce more non-software related bugs, or malfunctions. In essence, the code for the robot is included in one laptop that controls each aspect of the robot.

Next, the operating system was chosen. One of the greatest debates for PC users is whether a distribution of Linux or Windows should be selected. Linux offers more flexibility when creating a software platform and it is often much more reliable than Windows. In light of this, Linux has been chosen instead of Windows. However, there are many different distributions of Linux. In the past the Autobot team has had issues with the reliability of a selected distribution of Linux, so when choosing the operating system for this project, a very reliable, well tested, and supported system was picked: Ubuntu 12.04 Long Term Service Package.

There are many different programming languages to choose from when developing. Each language has different strengths and weaknesses. The C programming language is a procedural program that is designed for efficient execution. As our robot will be in competition to finish a course quickly, an efficient language was preferred. C, however, is older and does not include many of the features that would make the coding for the robot easier. One of the defining factors for our selection of a programming language is the availability of libraries. C++ was the language that was decided upon as it contained two libraries that our code is very heavily based upon: the Mobile Robot Programming Toolkit (MRPT) and OpenCV.

To increase the ease of development, flexibility of the code, and the ease of testing the robot, the code was designed to be modular. In a coding sense, this means that the code is divided into smaller segments that don't depend on each other or depend minimally on each other. This allows testing and development of the code to occur on one part of the robot even if another part of the robot is not functioning. For instance, the cameras can be tested independentally of the rest of the robot's systems. This allows them to be easily added and removed as needed.

As the final goal of this project is to enter the robot into a competition, speed was a very important consideration. There were some components in the code that slowed down the completion of the calculations. To amend this, some of the calculations are run concurrently by using threading. Threading is launching different calculations at one time and having them run simultaneously. Although this does complicate the programming, it greatly increases the speed of the robot.

System Integration

There were many different aspects of the robot that were needed to allow Bishop to make good decisions. In general, the robot will take the images from the cameras and the scan from the LRF and put the results into a map. The robot will then read that map to determine if it can head straight for the given waypoint or if it will need to turn. The process is detailed below.

GPS

Part of the competition is to design a robot that can travel autonomously from one location to another. In order to do that, the robot must know where it is. A GPS unit was used to find the robot's current location. The GPS is constantly pulling data in a separate thread from the rest of the code. This allows the robot to have an updated current location as soon as it is available.

Navigation

In order to move from the robot's current position to the desired GPS location a navigation function was used. Bishop takes all the GPS coordinates from a file and creates a stack. The waypoints from the stack and from our GPS are used to calculate an x and y offset using an equation for equi-rectangular projection. The equations used can be seen below.

$$xOffset = \Delta lon * \cos\left(\frac{\Delta lat}{2}\right)$$

$$yOffset = \Delta lat$$

$$distance = \sqrt{x^2 + y^2} * radius$$

Each time Bishop moves, he recalculates the x and y offset. The waypoint is reached when the x and y offset have reached a sufficiently low value. Once a waypoint has been reached, that co-ordinate will be popped off of the stack and the next co-ordinate becomes Bishop's new target.

Mapping

Bishop is designed to create maps based on images it pulls from two mounted cameras and a LRF. When the raw data is pulled from either of the cameras or the LRF, a map is created for that data. After the map has been made, it is sent through a series of filters in order to remove unnecessary data as well as to emphasize important details. The two maps are then combined to form one map that Bishop uses to navigate

The cameras are used to conduct line following or line avoidance. There are two cameras positioned at the top of the mast and facing downwards, which allows the robot to see lines on either side of him. Image processing is handled using the Open Computer Vision (OpenCV) set of tools. Functions such as the Hough Line detection, inverse perspective transform, threshold, and Canny filters are handled through this library. Each camera is calibrated using a precision map. This precision map is used to obtain the transformation matrix to perform an Inverse Perspective Transform on the images received from the camera. This transform generates a top-down view of the camera image. Line detection combines edge and color detection to locate and avoid the white lines. The process for this can be seen in Figure 2 below.

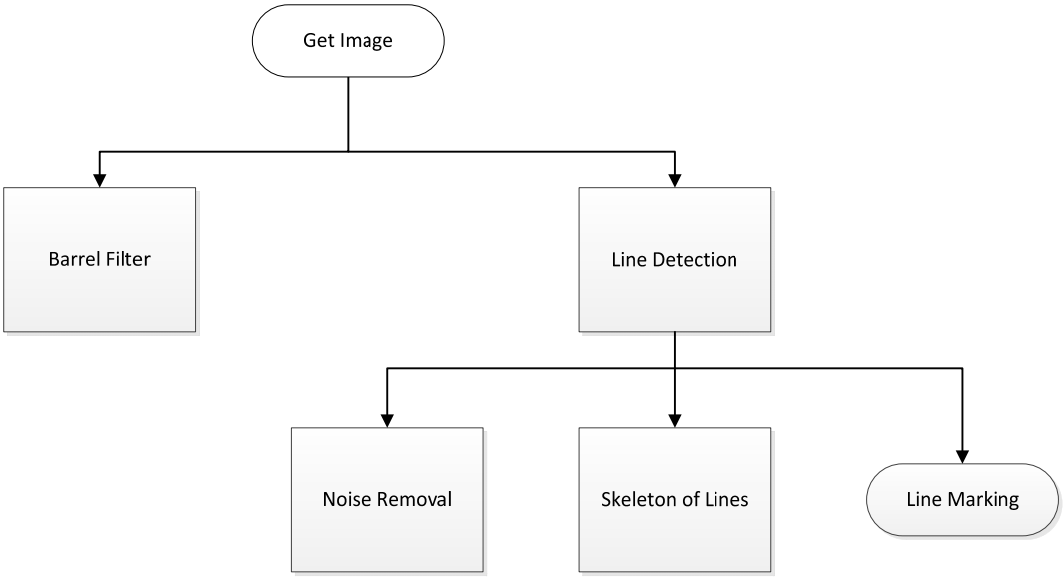


Figure 2: Line Detection

The LRF is used to detect obstacles. In order to find obstacles, the laser range finder sends out a laser pulse and determines the range (max 65 meters) to an object. The algorithm utilized determines a probability of an obstacle's presence, from 0-100%, where 0% means there is no obstacle present. This is done to prevent false positives from being saved into the map. The LRF map is then marked with the probabilities that were found with white being 0% and black being 100% probability. An example of this can be seen in Figure 3 and Figure 4. Figure 3 shows what the LRF map produces of the location shown in Figure 4. When the map in Figure 3 was made however there were construction barrels scattered throughout the courtyard.

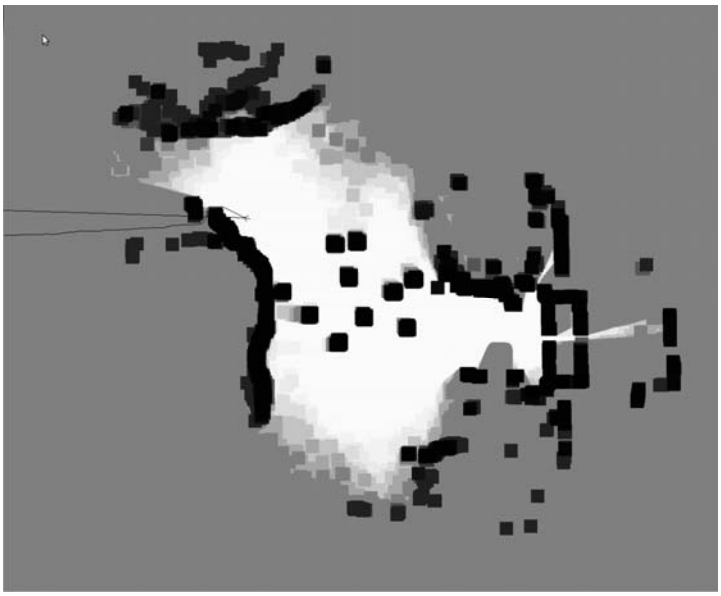


Figure 3: LRF Map of Dow

Figure 4:Picture of Dow

Pathing Algorithm

While previous versions of this robot have relied on a path finding algorithm, this version uses a simpler method of navigation that opts to change the robot's path as it approaches an obstacle. Essentially, the navigation is done by determining the distance and the direction it needs to go, and that direction is then changed as it approaches an obstacle according to the obstacle avoidance system.

Avoidance

Lane Following

Lane detection is mainly implemented through the use of two cameras. The cameras constantly provide a raw image of the area ahead of Bishop. Bishop's map building code then takes this image and starts line detection by filtering out any light that was reflected off of the grass in order to prevent any false positives. Next the map building code filters the image such that lines appear white while everything else appears black (with the exception of obstacles). From here Bishop's pathfinding algorithm takes over. The algorithm combines the filtered camera map with the map from the laser range finder. Any position on the map with a color value less than the accepted threshold has an obstacle cost added to it in order to flag that position as an undesirable path. Finally, Bishop navigates the created path.

Obstacle Avoidance

At the core of Bishop's navigation code is the obstacle avoidance system. This system relies on data acquired from the LRF and the cameras. The navigation code is constantly checking if an obstacle is within a given distance from the LRF. If an obstacle is detected, it finds a clearing off of the combined map and turns until that angle matches. The flowchart for this can be seen in Figure 5.

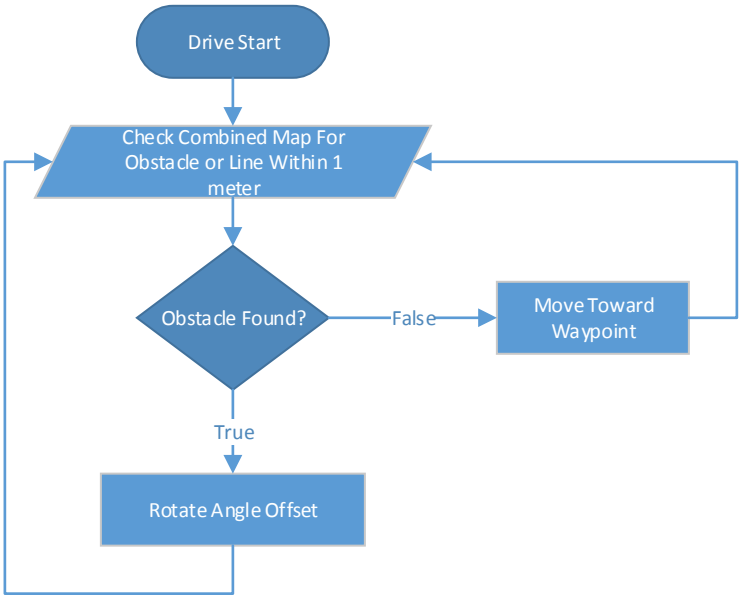


Figure 5:General Flowchart for Avoidance

In order to interface the expanded map into the avoidance system, we rely heavily on a function that returns the occlusion returned from any given pixel on the expanded map. In order to determine if an obstacle or line is detected, the value returned is compared to a set threshold. If it less than this threshold, it is deemed an obstacle. This function is embedded in a loop that continues to run until it has determined that there are no obstacles or lines within one meter from the robot. In the event that it does find an obstacle or line, the robot must correct its path. In order to do this, it must find a path it can take without colliding into an obstacle or crossing a line. The flowchart for this can be seen in Figure 6. In order to achieve this, the line of sight—that is, the imaginary line that the avoidance system uses on the expanded map to scan for lines and obstacles—is rotated and the loop that compares each pixel on this line is once again executed. The diagram of the robot turning is shown in Figure 7. Once the avoidance system detects a line of sight in which no objects or lines intersect, the robot then turns until that line is normal to the LRF.

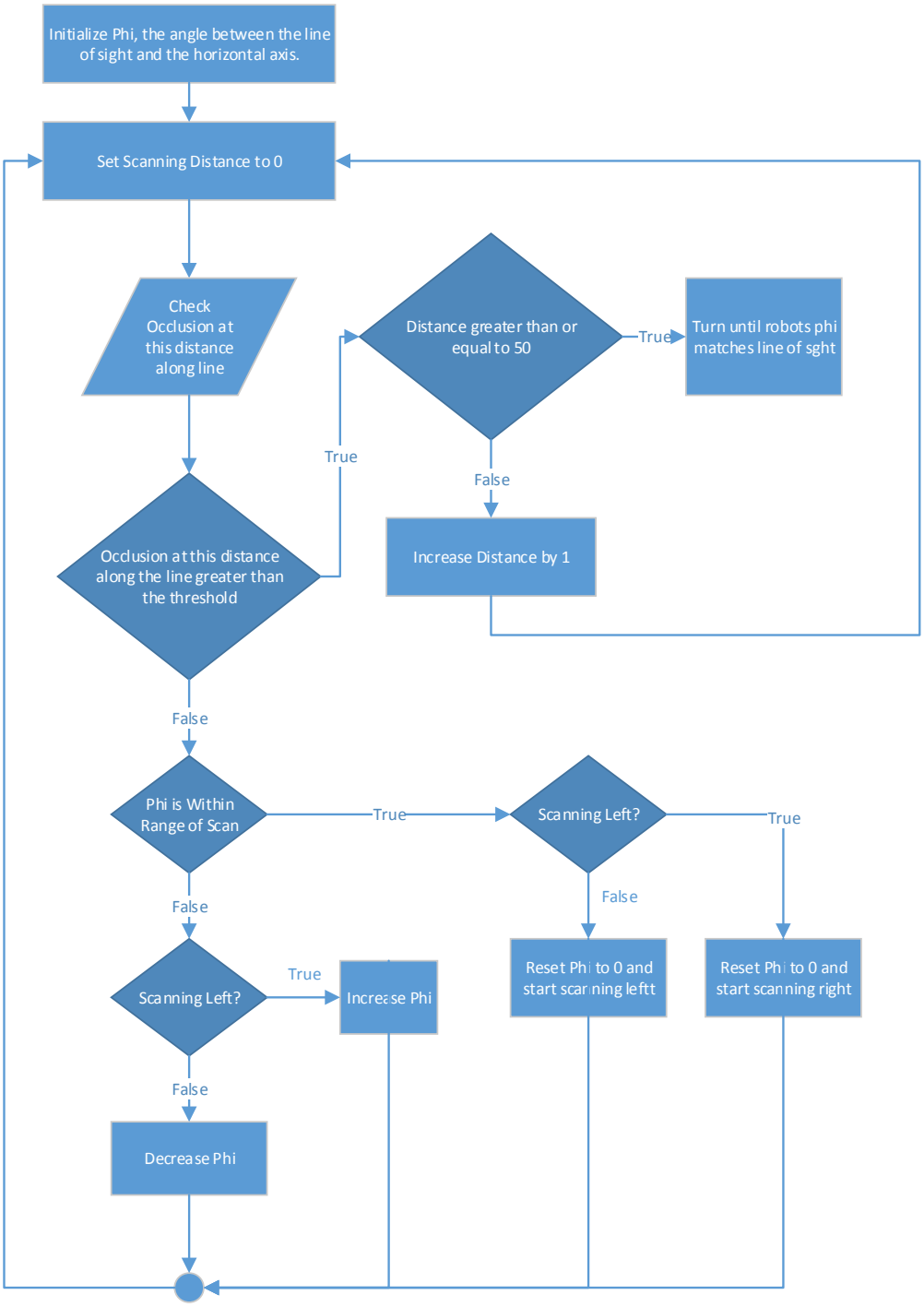


Figure 6:Phi Correction

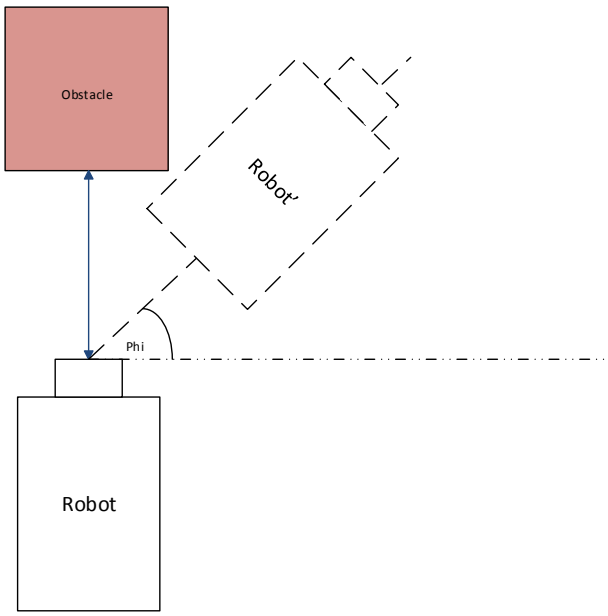


Figure 7:Robot Turning Diagram

Test Results

The majority of testing that was done with Bishop was to verify which aspects needed improvement. As such, most aspects were tested separately as well as together in a final systems test. Through these tests several things were determined. First Bishop's battery life can last between two to four hours depending on which set of batteries are being used. When Bishop was last tested it averaged a speed of 1.25 miles per hour (this can be easily altered in the software). Bishop set to detect obstacles when they are 75 inches away, however due to reaction time Bishop responds to them at about 70 inches. Bishop's reaction time was calculated based off of when obstacles are seen and when Bishop responds to them. Bishop responded to objects 5 inches after they were detected so at 1.25 miles per hour this calculates to be roughly a .5 second response time. Bishop did not seem to have trouble with the simple ramps he encountered and his speed was seen to stay constant if Bishop was required to move up a ramp. The specifications for the GPS that Bishop utilizes state that it is accurate to within a four meter radius. This was also confirmed when testing how accurately waypoints are found. Bishop's high speed operations were determined by the amount of time it takes to iterate through the drive process. On average this was found to be 95 milliseconds. Unfortunately, due to time constraints, there is little to no data as to how well Bishop deals with complex obstacles.

Predicted Performance

After all of the testing that was done it is safe to conclude that Bishop will be able to pass the qualifying course fairly easily. Bishop is also very likely to pass the basic course. Unfortunately, since there is little data as to how Bishop responds to complex obstacles, there is not enough evidence to say with certainty that Bishop will be able to pass the advanced course.

Issues Encountered Over the Semester

Detailed tests were taken of the robot regularly to identify issues. There were multiple design issues that our team encountered. These issues included cold affecting the LRF, failing USB ports, incorrect data from sensors, and incorrect navigation and avoidance code.

Whilst testing we encountered an issue that the LRF couldn't operate in low temperatures which hindered our ability to continue testing obstacle detection and avoidance. To solve this issue we created insulation around the SICK laser range scan finder and used Hot Hands to heat the component while we tested.

We also had issues with the laptop ports as a USB cable had been removed improperly when Bishop ran over a cord and it got caught on the wheel. This created issues connecting to that USB port. To solve this we reconnected the USB port but lost our track pad and other laptop functionality. This led us to acquire a new laptop as a safety backup in case our current one stopped working. This did eventually happen, so we moved all testing to the new laptop.

Another issue encountered was incorrect data being reported from the IMU. The IMU was incorrectly calibrated and faced the incorrect direction on the robot. This caused the robot to believe that it was moving in one direction when it was actually moving in a different direction. After recalibrating the IMU and placing it in the correct position the data received was correct.

The navigation code from the previous semester was relying on faulty calculations. This caused the robot to move in ways that the team had a hard time understanding. The solution to this problem was to go through each of the calculations and rewrite them if necessary.

The map made from the cameras was also incorrect. This did not result from incorrect data from the cameras, but rather incorrect calibration of the cameras. Much of the code used for the cameras was legacy code and difficult to understand and support. To fix this issue, the camera code was rewritten following more modern examples and in a way that the team could understand.

Table 1 shows a cost estimate for all of the parts used on the robot throughout its lifetime.

Part	Cost/unit	# of units	Total Cost
SICK LMS-291	$5,500.00	1	$5,500.00
Logitch C910	$70.00	2	$140.00
Razor 9DOF	$75.00	1	$75.00
GlobalSat BU-353-S4	$42.00	1	$42.00
Texas Instruments MDL-BDC24	$80.00	2	$160.00
NPC-T64	$331.00	2	$662.00
Lonovo T520	$430.00	1	$430.00
US Digital Encoder	$97.00	2	$194.00
CUI VHK50W-Q24-S24	$121.00	1	$121.00
Light Stack	$25.00	1	$25.00
Light Controller	$20.00	1	$20.00
E-Stop	$20.00	1	$20.00
Misc Electronics/Wires/Connectors	$100.00	1	$100.00
Wheel Hub	$25.00	2	$50.00
Tire	$22.00	2	$44.00
Front Caster	$32.00	1	$32.00
Mechanical	$200.00	1	$200.00
		Total:	$7,815.00

The robot that we presenting has had sufficient change and innovation to allow it to be entered into the design competition. Much of the robot that was used in the 2014 competition has been changed to make a more efficient, safe, and effective robot. We look forward to competing with it in the 2015 Auto-Navigation completion. Thank you for considering our robot.