Project Number: MQP GSF - M122

GUEST ORIENTATION, ASSISTANCE, & TELEPRESENCE (G.O.A.T) ROBOT

A Major Qualifying Project Report:

Submitted to the Faculty

Of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

_____

_____

Noah LeBlanc

Kush Patel

Sabbir Rashid

Date: August 29th 2012

Approved:

_____

Prof. Gregory S. Fischer, Major Advisor

_____

Prof. Taskin Padir, Co-Advisor

TABLE OF CONTENTS

AUTHORSHIP PAGE

ACKNOWLEDGEMENTS……SR
ABSTRACT……SR
EXECUTIVE SUMMARY……SR
INTRODUCTION……KP
LITERATURE REVIEW……SR/NL
PROJECT APPROACH……NL
ALTERNATIVE DESIGNS……NL/KP/SR
FINAL DESIGN METHODOLOGY……NL/KP/SR
FINAL DESIGN VALIDATION……NL
DISCUSSION……NL
FUTURE WORK……NL/SR

ACKNOWLEDGEMENTS

We would like to thank our advisors Professor Gregory Fischer, for his invaluable advice and commitment to providing the resources we needed to make this MQP possible, and Professor Taskin Padir for his unwavering support during this project. We would also like to acknowledge everyone working at the AIM Lab, especially Kevin Harrington and Alex Camilo, for their invaluable advice. Furthermore, we would like to thank Tom Angelotti from the ECE Shop for providing us with last minute parts, Professor Stephen Bitar for his assistance in soldering, and Tracy Coetzee, for helping with our part orders. Finally, we would like to thank Segway Inc. for the use of the Segway Robotic Mobility Platform, upon which we built our robotic system.

ABSTRACT

The project was focused on a mobile research platform for autonomous navigation components and sensors vital to its autonomous interaction with its environment. The goal of this project was to create such a mobile robotic platform, which would in turn be capable of acting as a fully autonomous tour guide for the WPI campus. The project combined the robust capabilities of a Segway Robotic Mobility Platform with the cutting edge adaptability of the Robot Operating System software framework. The robot will work in conjunction with school staff to provide video tour information as part of an enhanced tour experience. The project is a highly visible representation of WPI's unique MQP program and its ability to prepare engineers capable of solving real world problems.

EXECUTIVE SUMMARY

Like all great endeavors, this project began with a simple notion. The thought that arose in the mind of a WPI Robotics Engineering student resembled the rhetorical question: *Why not create a robot designed to help people?* Of course, there are many different ways of helping people – and even more ways of designing a robot to do so. As one RBE student joined with another, the original idea that followed was the design of an indoor escort robot, which would use RFID technology to guide the user to the appropriate location in the building they needed to reach. The thinking process preceding the project was at this stage, when the idea was discussed with an Electrical Engineering student, who enthusiastically joined the team completing the three member project group.

As the project began, the team continued to conduct research on robots designed for helping humans. Several articles that were found focused on tours given by robots. Included were both indoor robots that gave tours inside museums and an outdoor tour guide robot at the National Taiwan University. Of the original idea, the only aspect that remained was the concept of guiding the user. An outdoor robot that gave tours around a university campus had not yet been accomplished in the United States. With this in mind, the team set out to make history.

The goal of this project was to create a mobile robotic platform capable of acting as a fully autonomous tour guide for the WPI campus. By including components and sensors vital to autonomous interaction with its environment, the robot will serve as mobile research platform for autonomous navigation. Several types of sensors were used, including GPS, ultrasonic, and orientation sensors for localization and obstacle detection. Two identical cameras were mounted to provide the robot with stereo vision, and speakers were installed. A control computer was built from individual parts, and a touchscreen computer was purchased to serve as a graphical user interface.

For each of the parts used, validation of functionality needed to be tested, for the both the hardware and software comprising the system. The software for both the control computer and the user interface computer was developed within the Robot Operating System (ROS) software framework. ROS is a graph based software architecture which consists of various nodes, or separately running executable programs, which communicate through special ROS messages and services. The GPS sensor was tested by logging GPS readings once per second for a period of one hour. Testing of the odometry data was accomplished by manually driving the robot around a circuitous route in the basement hallways of Higgins laboratories. The tests for navigation without obstacles were implemented by modifying an example ROS node, downloaded from the ROS website, used for sending a single goal to the navigation stack. Testing of power of the overall system was conducted, to make sure the batteries would be able to provide the necessary energy over the maximum allotted time period. Furthermore, testing of the localization and point cloud imaging for obstacle detection was successfully completed.

Though many of the tests for validation ran successfully, such as those listed above, not all of the tests worked out as planned. For example, while the robot was able to navigate without the presence of obstacles, once obstacles were introduced errors arose from artifacts present in the point clouds caused by imperfect and noisy images from the stereo cameras, which in turn caused the image processing algorithms to improperly locate objects. Also, numerous attempts were made to use the orientation sensor's included software to calibrate the magnetic compass sensors in its position atop the robot's sensor mast. However, the calibration was unsuccessful before an unexpected failure of the sensor.

Furthermore, several difficulties were faced (many of which were overcome) throughout the integration of the ultrasonic sensors.

Both the achievements and the complications faced contributed to the learning experience of the project. Despite such difficulties, the project was an overall success, and the goal of creating a mobile robotic platform was accomplished. To summarize, the team succeeded in designing and building mounts and framing for sensors and electronics, achieving operation of the robot from both plug-in AC and battery DC power with operation of the robot on DC power for at least two hours. Furthermore, the robot was capable of travelling two miles between battery charges, travelling at more than 2 m/sec and accelerating at more than 1m/sec^2, and reaching all handicap accessible outdoor areas. Both manual and wireless e-stop systems were created, as well as a breakout board for interfacing sensors & computers with the PIC32 microcontroller, and the ability to manually control the robot using an Xbox360 controller. Finally, a ROS package for interfacing with Centralized Controller based RMP; stereo vision to generate point cloud data, which in turn was used to detect obstacles; localization using IMU, odometry, compass, and GPS data; and Autonomous Navigation to consecutive waypoints in the absence of obstacles were achieved.

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

The General Problem

Robotic assistance has become predominant in modern society as a means of helping people to complete certain repetitive, mundane, or simple tasks and to do so safely and efficiently. Robots are designed generally for tasks that are very repetitive such as building cars, assembling parts, vacuuming one's home—but there are still many other tasks that can become automated to help benefit society. Of the many tasks, human assistance is one that is simple enough in some cases to be automated. One such example is RI-MAN, shown below in Figure 1, the first robot designed to aid the elderly by lifting and carrying humans to provide care (RIKEN, BMC). Since then, more robots have been created for similar tasks, such as RIBA, "the friendly robot nurse" (Salton). The emphasis that these robots were designed for the purpose of human assistance can be inferred by spelling out their acronyms; RI-MAN: Robot Interacting with huMAN; RIBA: Robot for Interactive Body Assistance.

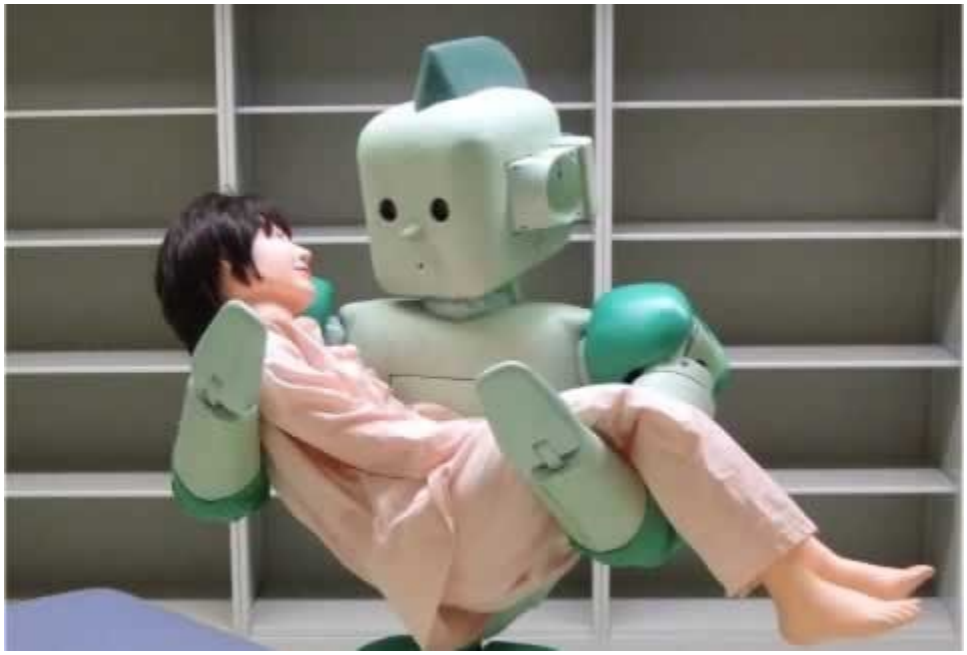


Figure 1: RI-MAN - First robot designed to lift and carry humans to provide care in an elderly home

With the development of transportation technologies in the past century, such as the automobile and airplane, long distance travel has become common. People are able to travel from home to work a few miles away, or across the country for education or business with relative ease. In many cases involving travel, people enter very unfamiliar places where they may or may not receive assistance from people, maps, or escort services to provide them with information and a tour of their new environment. Automating the task of public assistance in places like college campuses, hospitals, or offices can enhance the lifestyles of people now as well as in the future. An example of such a robot is the autonomous receptionist MechaDroid Type C3, shown in Figure 2, which has the ability to answer simple questions through voice recognition (TechnoGadget), as well as create a variety of facial expressions (Teruaki Ando).

Figure 2: MechaDroid Type C3 - Autonomous Receptionist

In order to design these public assistance robots, there is significant research, testing, and development that goes into creating a platform capable of these tasks. Starting research or designing a platform from scratch is very difficult as well as time consuming, since much of the time is needed to physically design the different portions of the system and interfacing them before the logic and algorithms can be developed to produce an intelligent system. Furthermore, in order for such a process to be successful, extensive planning and insight into the future possible uses of the system are necessary.

Overall Goals

As reflected in *The General Problem* section above, the use of autonomous robots in modern society will increase the amount of aid and assistance available to those who need it. With this is mind, this design project attempted the engineering of a mobile research platform for autonomous navigation and human-robot interaction which included sensors and intelligent systems vital to its autonomous interaction with its environment and humans. In creating such a platform, future modifications to the platform must be considered. Therefore, the robot must be built with a simple and light base that would facilitate modifications and include several features and tools to allow for easy adaption into a future project or integration of this platform into society.

The goal of this design project was to engineer a mobile platform that is simple to use, operate, and modify such that people could easily integrate it into their research projects or modify the robot to complete the tasks they desire. One such modification of the platform that was considered and implemented in a feasibility study was the use of the platform as an autonomous tour guide.

Figure 3: Tour guide implementation of the GOAT robot

Figure 3 shows a possible implementation of the GOAT platform as a tool to assist WPI's Crimson Key in the task of providing campus tours. For this purpose, the platform can be used to enhance the tour experience as its mobile ability allows it to travel to any of the possible tour locations and, through visual and audio interaction capability, provide detailed information about WPI and showcase the WPI MQP program and as an example of a real-world engineering accomplishment. While the use of the platform as a tour guide robot greatly influenced its design, the goal of this project ensures that a much greater scope of assistance could be achieved with simple modifications.

General Procedure

In developing this system, the team divided the major portions of this project into smaller, simpler tasks such that each individual team member was able to handle a subset of tasks. The completion of smaller tasks in turn allowed the team to complete the major portions of this project. First, the ideas were discussed and finalized resulting in a concrete set of goals with a strong focus on the overall goal of the project. Along with the development of the smaller tasks, alternative methods and implementations were discussed, as possible means of achieving some of the major portions or simpler tasks of this project. Upon completion of the ideas, designs began to take form, which were analyzed for their strengths and weaknesses, after which the final design methods were chosen and the components of the system were constructed and integrated. The platform's functionality was tested and validated upon completion, to be followed by future enhancements, development, and implementations.

This document provides detailed information about this project, the steps taken, the research conducted, and the final results and analysis. The following chapter is the literature review intended to provide the reader with a background of similar research and designs that are currently in existence,

which had served to provide the team with a foundation for the design decisions that were made. Following the literature review is the project strategy which discusses the methods used to arrive at the key goals and specifications required to achieve the overall project goal. The alternative designs section entails the other design options that were considered in development of this platform and the reasons for rejecting or approving them. Subsequently, the final design methodology goes into details of all the tasks and steps that were actually taken to implement the approved designs. The design validation section provides details on the different tests conducted to verify the capabilities of the robotic platform. Following in the discussion are some discoveries made and difficulties overcome during this project. The future work section provides possible enhancements and developments for any successors who wish to build on the achievements of this project.

2. LITERATURE REVIEW

Two types of tour guide robots, indoor and outdoor, were investigated as background. Indoor robots include Rhino and Minerva. The outdoor robot is the NTU-1. The following is a brief overview of the robots researched, providing examples for each kind of robot.

RHINO Indoor Tour Guide Robot

An example of an indoor tour guide robot was developed by students from Germany and the United States and deployed at the Deutsches Museum Bonn in Bonn Germany over a 6-day period (Hähnel, Schulz and Burgard). The robot was based on the RHINO platform developed at University of Bonn (Joachim Buhmann) which is equipped with laser, sonar, infrared, and tactile sensors for mapping and navigation as shown in Figure 4.

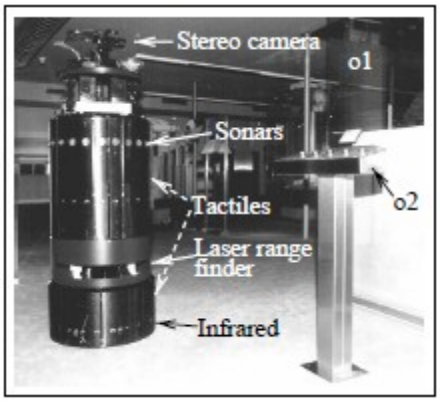


Figure 4: RHINO Platform-based Tour Guide Robot

Localization

The robot's localization routine utilized Markov Localization (Maria Isabel Ribeiro) modified with an entropy filter. The system relies on an assumed initial pose which is updated by sensor readings. Sensor readings which increase the robot's certainty of pose are retained and those which are "corrupted" by the presence of people, and therefore decrease its certainty of pose, are discarded, making it suitable for a dynamic environment where an absolute certainty of pose is not possible (Dieter Fox). However, to account for random events, a random sampling of corrupted readings was incorporated to account for unexpected changes in robot pose.

Mapping and Navigation

An accurate map of the museum layout was used for reference. A probabilistic occupancy grid algorithm was used to update the map with unexpected changes in the environment such as people or moved furniture. Paths were generated around obstacles, and when the destination was reached, the map was reset to the preloaded museum layout so that outdated obstacles did not persist. Hard and soft velocity constraints were used to avoid obstacles (Thrun, Bennewitz and Burgard, Experiences with two Deployed Interactive Tour-Guide Robots). Hard constraints set velocity limits that would inevitably result in a collision. There were two soft velocity constraints; one which preferred velocities that brought the robot closer to its goal, and a second that preferred the maximum allowable velocities, i.e. those that brought it as far from obstacles as possible. The soft constraints caused the robot to prefer

uncluttered areas while bringing it closer to its goal, all while working within the hard constraints. Path planning was accomplished by value iteration, which values each unoccupied grid cell with the value of its best choice unoccupied neighbor, plus the cost of moving to that cell (Bram Bakker). The steepest descent to the goal cell valued at zero became the robot's path, which was then passed to the collision avoidance routine to generate "target locations" along the path.

Task Planning

The robot's task planning software generated a series of symbolic actions, using the GOLOG programming language (Hector J. Levesque), which had a high probability of fulfilling a user's request. The linear set of actions was processed to generate sets of hierarchical and conditional plans using a custom set of commands called GOLEX (Hähnel, Schulz and Burgard). Hierarchical plans simplified the task planning process by breaking the task down into specific low-level actions for the robot, such as turn, drive a certain distance, orient camera, etc. Conditional plans were pre-specified variations based on the current conditions, such as the amount of detail provided to the user when explaining exhibits based on the level of explanation the user requests. The execution of the commands was monitored by GOLEX which made further adjustments to the plan when necessary.

User Interface

The on-board interface was a mixed-media interface that integrated text, graphics, pre-recorded speech, and sound. When visitors approached the robot, they were able to choose a tour or, alternatively, listen to a brief, pre-recorded explanation. They indicated their choice by pressing one of four colored buttons. There was also a web-based interface which allowed a user to monitor the robot's actions and state through cameras and an environment map, and receive information pertaining to the tour.

Minerva Indoor Tour Guide Robot

Another example of an indoor tour guide robot was developed as a project at Carnegie Mellon University. The robot, shown in Figure 5, called Minerva, was installed for a total period of 14 days from August 24 through September 5, 1998, in the Smithsonian's National Museum of American History, where it gave tours of the museum (Thrun, Bennewitz and Burgard, Experiences with two Deployed Interactive Tour-Guide Robots).

Figure 5: Minerva Robot.

Localization

Minerva was equipped with special-purpose localization equipment, including radio beacons, cameras, and laser range finders, and sonar and tactile sensors. Since the environment in which Minerva was deployed was filled with moving objects such as people, Minerva used a high resolution Markov localization algorithm, as in the case of the RHINO tour guide robot, to maintain a sense of orientation. Periodically, camera images and laser range scans were compared with preloaded ceiling and occupancy maps to increase the certainty of the robot's pose.

Mapping and Navigation

While inertia and torque limits imposed constraints in the robot's motion, Minerva was able to generate collision free motion using a collision avoidance module. This module controlled the motion direction and the speed of the robot to avoid collisions while maximizing on the robot's progress to the goal. Minerva was able to detect people by applying a filter for recognizing when sensor data is corrupted, indicating that a person is present. People that Minerva was trying to attract for purposes of providing tours were treated differently from dynamic obstacles (Thrun, Bennewitz and Burgard, Experiences with two Deployed Interactive Tour-Guide Robots).

An occupancy map (Thomas Collins) was created by Minerva from range data, images, and odometry. Figure 6 is an image of a map created by Minerva through manual maneuvering. It represents an area of 65 by 45 meters. Dark regions are most likely occupied, lighter regions represent areas that Minerva was free to travel, and purple regions are unexplored. Minerva used a statistical approach, combining mapping and sensor data to travel in areas that have the greatest probability of being unoccupied (Thrun, Bennewitz and Burgard, MINERVA: A second generation mobile tour-guide robot).

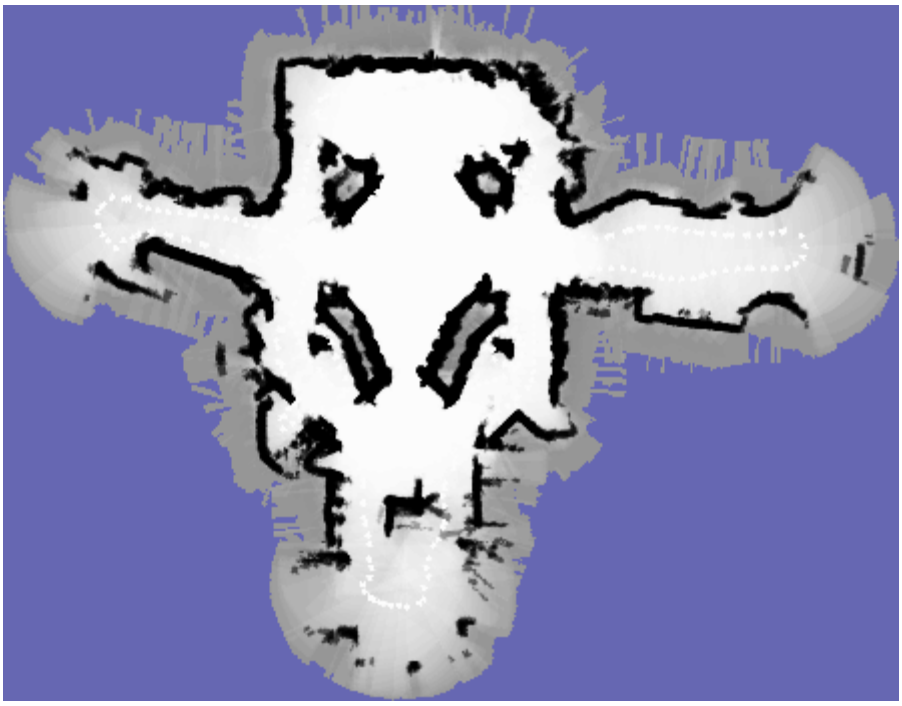


Figure 6: National Museum of American History occupancy map

Human Interactions

Compared with the RHINO tour guide robot, Minerva is much better known amongst tour guide robots. This is due to Minerva's memory-based reinforcement learning approach (Natalia Hernandez-Gardiol), which the robot used to learn how to attract people. By using different head motions, different facial expressions, and different speech acts, it could determine which strategies of interaction work best. Minerva expressed emotion using human-like expressions, while providing guidance with its synthesized voice and buttons. Nevertheless, the messages voiced by Minerva were prerecorded, and Minerva's functions were essentially limited to explaining exhibitions, approaching visitors, and suggesting tours (Thrun, Bennewitz and Burgard, Experiences with two Deployed Interactive Tour-Guide Robots).

National Taiwan University Outdoor Tour Guide Robot

An example of an outdoor tour guide robot is the NTU-1 from National Taiwan University, shown in Figure 7. The robot consists of a custom-built platform which users interact with through a touch-screen interface and voice commands. The robot guides users to their chosen destinations where it provides a video presentation. Table 1 provides a list of specifications for the NTU-1 Robot.

Figure 7: NTU-1, the National Taiwan University tour guide robot

Table 1: NTU-1 Platform Specifications

Attribute	Specification
Size	80cm x 80cm x 140cm
Weight	80kg
Shape	Fiber glass reinforced plastics
Skeleton	Aluminum
Driving Type	Two-wheel differential type
Driving Motor	DC brushless motor
Max. Speed	1 m/s
Batteries	Two lead-acid batteries 12V 26Ah series
Computers	Intel core 2 duo
Display	15" touch panel
Continuous Operation Time	1~2 hours

Architecture

NTU-1 relies on dual Intel Core 2 Duo based computers (Intel), named PC-1 and PC-2, for the bulk of its computational tasks. The NTU-1 system architecture is shown in Figure 8. Low-level tasks such as navigation and collision avoidance are handled by PC-1 and high-level tasks such as human robot interaction and image processing are handled by PC-2. The computers communicate through a LAN router which also provides outside LAN access to the system. All system components communicate directly with the computers through USB, RS-232, or bluetooth, with the exception of the ultrasonic sensors and a head control module which interface through an FPGA (Chiang, Tseng and Wu).

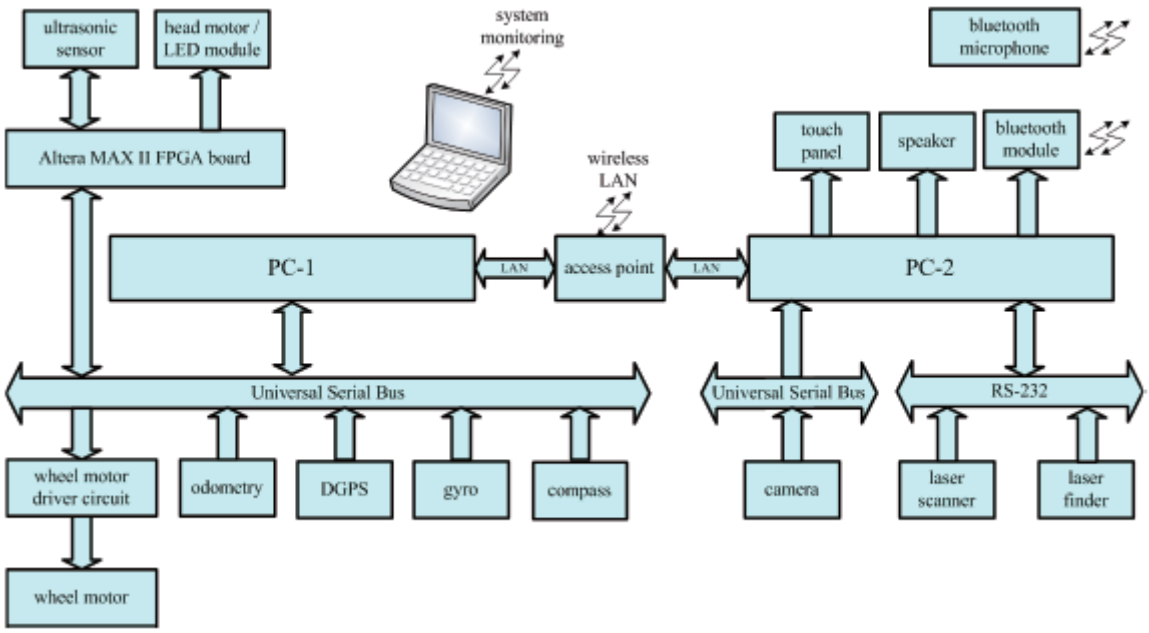


Figure 8: NTU-1 Robot system architecture

Localization

The robot contains a preloaded 2D map of the university campus with a coordinate system based on an arbitrarily chosen location where the x-axis and y-axis are aligned with the north and east compass directions respectively. The current pose is determined using GPS, a compass module, and wheel

encoders with data fusion performed using an Extended Kalman Filter technique (Greg Welch). If the GPS signal is lost, the robot will rely on information from the compass module and wheel encoders, until the signal is restored (Chiang, Tseng and Wu).

Path Planning and Obstacle Avoidance

The campus map contains 66 checkpoints which all robot paths are based upon. The system determines the path to its destination by choosing the sequence of checkpoints with the shortest distance using a Floyd-Warshall algorithm (Department of Computer Science and Engineering, HKUST). The paths between the nodes are then divided into sub checkpoints approximately 6 meters apart. The robot navigates to each checkpoint along the generated path while deviating to avoid obstacles detected by 12 ultrasonic sensors place around the robot's body (Chiang, Tseng and Wu).

3. PROJECT APPROACH

Initial Client Statement

"The Guest Orientation, Assistance, and Telepresence (GOAT) Robot will act as a tour-guide or visitor escort at the WPI Campus. GOAT will provide live or telepresence assistance to prospective students as well as academic, corporate, official, and other guests. Users will interact with the robot through a combination of a touch-screen interface, voice, and gesture commands. The robot will guide visitors to destinations and be capable of providing video tour information."

Project Objectives

Much of the appeal of the project came from the wide variety of possible features for the platform, a variety reflected in the initial client statement above. Due to limited time and resources, the students were required to choose a subset of the possible features. Furthermore, a fully autonomous robot capable of safely interacting with visitors on a crowded college campus is a daunting challenge by any measure. As a result, the primary goal of the project was defined as producing a mobile robotic platform with the hardware and software systems in place necessary for autonomous navigation and interaction with the campus visitors. The capabilities of the system would be tested by a simple implementation of autonomous navigation and some preliminary interactions with campus visitors. With the scope of the project reduced from the original idea and vision, the next step in the planning process was to define the project objectives and specifications necessary to achieve the new overall goal of the project. The following objectives tree in Figure 9 represents the decisions made after many discussions with our advisor, and some preliminary investigations into the feasibility of the possible features.

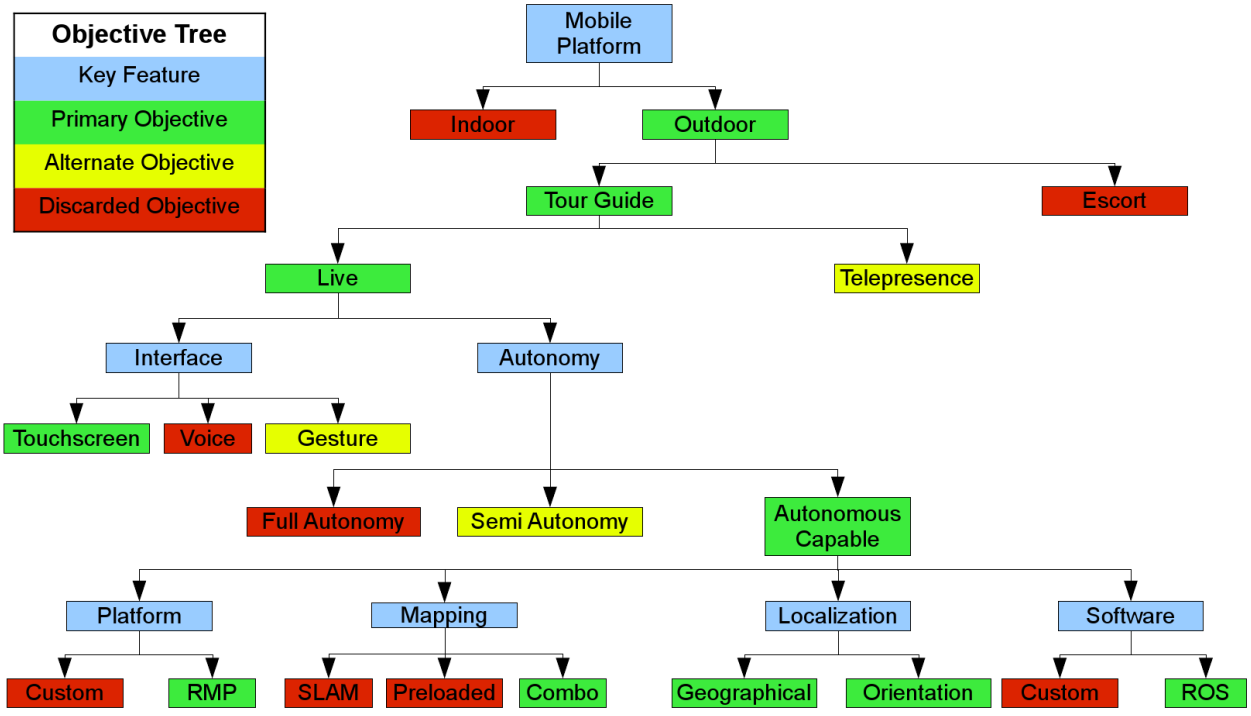


Figure 9: Initial project objectives decision tree

With the combinations of these selected primary features, the robot is a representation of an autonomous assistance robot. Many of these primary branches, such as autonomous capable, mapping, and interface, are involved enough to have been individual projects. Therefore, considering the limited time and resources available, if each of the alternative and discarded objectives were included the project may have drifted away from the primary objective of designing an autonomous capable platform. Like the field of robotics, this project took several components of different complex systems and combined them into a single autonomous assistance robot. The following is an explanation for the classification of each objective into primary, alternative, or discarded.

Primary Objectives

1. Outdoor – Outdoor operation was chosen over indoor to provide the greatest visibility to the public and the widest variety of tour locations. Also, making the robot capable of travelling outdoor to various campus buildings would allow it to access multiple indoor environments in the future, rather than just one isolated campus building.
2. Tour Guide – Designing the robot as a tour guide simplified the scope of its operation by allowing it to travel predesigned and existing tour routes rather than unpredictable routes required by a visitor escort. In addition, the tour guide function offers the greatest opportunity for interacting with a wide variety of campus visitors and attracting interest from prospective students and their families.
3. Live Interaction – Concentrating on live interaction reduced the scope of the project to the systems and user interface of the platform itself as opposed to the separate and additional networking and interface requirements of a telepresence system. Furthermore, true telepresence, i.e. allowing remote users to gain partial control of the platform, would require an additional layer of software control between the normal autonomous and manual control modes resulting in a more complex system.
4. Touch Screen Interface – A user-friendly graphical interface with easily understood prompts and inputs was considered the minimum for a robot designed to interact with the public. A touch-screen interface was considered to be the most intuitive form of graphical interface since it combines the prompts and user inputs into virtual graphical buttons. The graphical buttons also allow for great flexibility in designing and modifying a wide variety of interfaces for a given situation. Finally, the low cost of modern touchscreen interfaces, which are comparable to normal computer monitors, made this option a clear winner for user interface devices.
5. Autonomous Capability – Developing an autonomous tour guide robot is a project with a very large scope, especially for an undergraduate project. In order to make the goals of the project more realistic and reachable, it was decided that the minimum requirement be creating a robot that is capable of acting as a fully autonomous tour guide, rather than one that fulfills that goal. This lesser requirement still demanded that all the systems required for full autonomy be in place and their operation verified by limited autonomous operation under controlled test conditions.
6. Off the Shelf Platform – Developing a mobile robotic platform is a daunting challenge and would require the full attention of the project team for most, if not all of, the project period. In

addition, developing a tour guide robot would inevitably require additional project teams to carry on the work in the future. Capturing the interest of future teams would be more likely if an autonomous capable platform was already available, rather than a simple platform with little or no software development that would be a tour guide robot in name only. Beginning the project with an existing mobile platform would allow the team to concentrate on higher level development and leave the school with at least a partially capable tour guide robot.

7. Combination Map – Autonomous, mobile robots generally require a map of their environment to allow navigation from their current location to one or more goal locations. If a robot is working in an unknown environment it must employ Simultaneous Localization and Mapping (SLAM) whereby it builds up a map of its surroundings, fitting new data with previously collected data to create a picture of its surroundings and determine its position relative to those surroundings. If a robot is working in a known and static environment it can utilize a preloaded map stored in permanent memory and use sensor data to determine its location relative to the map. Since the tour guide robot was to operate in a known environment, i.e. the school campus, it could navigate between sets of preloaded GPS waypoints that would lead it to goal destinations while confining it to allowable areas of travel such as sidewalks. However, due to the limited accuracy of GPS data and the presence of dynamically changing obstacles on the campus, such as people, maintenance equipment, etc., the robot was required to combine the preloaded map with additional sensor data to refine its localization and update the map with any changes and obstacles it perceived.
8. Geographical Localization – At a minimum, the robot was required to determine its 2D location on the campus map in order to navigate to goal destinations. The simplest way of achieving this was to use a GPS sensor which senses the robot's location in a geographical coordinate system of latitude and longitude. A GPS sensor offered two main advantages: the sensor uses existing signals from satellites so it does not require the robot to sense its local environment, and locations on the campus map could be defined in geographical coordinates using existing maps such as those from Google Earth (Google).
9. 3D Orientation – A simple 2D location on the map was considered enough to navigate to a given location on campus. However, other factors had to be considered in addition to its geographical location. In order for the robot to sense the position of objects around it in a dynamic environment, it had to know its position relative to those objects. Since the outdoor campus is not a simple 2D plane like the floor of a building, the robot had to know its 3D orientation to keep track of obstacle locations when traveling on inclines and other uneven terrain. Therefore, the robot required sensors to determine its 3D orientation, including the three angles of rotation which affect the apparent location of sensed objects relative to the robot's body and its X and Y translations in reference to a global coordinate frame.
10. Existing Software Framework – An autonomous mobile robot intended to interact with humans requires a complex software system with several software modules designed to handle different tasks such as navigation, user interface, and overall system monitoring. There exist software frameworks that provide much of the groundwork required for developing these modules and allowing them to communicate as a cohesive system. To ease the task of creating a software system for the robot and allowing the maximum amount of focus on developing higher level

capabilities, the team decided to use and existing software framework in the form of the Robot Operating System or ROS.

Alternate Objectives

1. Telepresence – Telepresence would allow remote users to view a tour through the "eyes" of the robot or to take partial control of the robot for exploring the campus or interacting with people on campus. This technology offers a wide range of uses and is an attractive tool for capturing a wider audience interested in the school. However, due to the increased complexity of a telepresence system in addition to the overall tour guide robot system, it was identified as an alternative objective to be pursued only if basic live presence tour guide capabilities were achieved.
2. Gesture Interface – Computer vision was considered from an early point in project development as a tool for mapping and obstacle detection. The existing and free software packages capable of providing computer vision include features for recognizing human forms and faces. The ability to recognize the human form allow for the recognition of common gestures that could be used for commanding the robot. While potentially quite useful, this capability was considered more of a "gimmick" to be pursued only if time allowed.
3. Semi-Autonomy – Developing the systems needed for an autonomous mobile tour guide robot was considered a minimum objective for establishing enough capabilities for the project to be carried on by future teams. Once the systems were in place, actual testing of the autonomous modes and deployment of the platform as an autonomous robot could proceed provided enough time remained before project completion. Semi-autonomous modes of operation, such as those allowing the robot to navigate predetermined routes, was considered a logical first step towards eventual, fully-autonomous operation.

Discarded Objectives

1. Indoor – Operation of the robot for indoor environments was considered beyond the achievable scope of the project as it would require modified mapping and navigation systems, as well as enabling the robot to directly interact with its environment to operate doors and elevators.
2. Escort – Allowing the robot to escort individual campus visitors to specific locations would require that the ability to fully navigate the campus be implemented as well as more complex human robot interaction, and was therefore considered beyond the scope of the project.
3. Speech Recognition – The ability to recognize spoken commands was considered as an option for user input. However, since this capability did not relate to any other systems being developed and would pose significant challenges of its own, it was considered an unjustified complexity for the project.
4. Full Autonomy – Fully autonomous operation was defined as allowing the robot to plan its own paths and navigate to destinations based on locations chosen by a user. While highly desirable for the final iterations of the system, such operations require extensive development and testing and were considered outside the scope of the initial project.
5. Custom Platform – Developing a mobile robotic platform capable of acting as a campus tour guide was would have been necessary without the ability to acquire and existing platform. Once

an existing platform was acquired by the team, there was no reason to develop one from scratch and so this was discarded as an objective.

6. Custom Software Framework – The idea of developing a software system from the ground up was initially considered by the team. However, once the complexities of developing a system for an autonomous mobile robot capable of human interaction were realized, the idea of a custom software system was discarded in favor of using an existing software framework in the interest of time and the ability to focus on higher level capabilities.

Design Specifications

With the primary objectives for the project identified, the next step was to define design specifications required for meeting those objectives. Below is a list of specifications divided into groups for several primary objectives, and sub-groups of minimum specifications or must haves, and desired specifications or should haves.

1. **Outdoor Operation/Platform**

   The robot must…

   a. Be capable of reaching all handicap accessible outdoor areas on campus during daylight hours and in the absence of precipitation
   b. Be capable of operating in areas of shade or direct sunlight
   c. Be capable of operating in temperatures of 15°C to 35°C
   d. Be capable of two hours of continuous operation between battery charges
   e. Be capable of travelling two miles between battery charges
   f. Be capable of travelling at 2 m/sec and accelerating at 1m/sec^2
   g. Be equipped with a wireless emergency stop system that will cut power to the motor drives
   h. Allow for charging from a standard 120VAC power outlet while the machine is powered down
   i. Must have a manual control mode for moving the platform under power when not in autonomous mode

   The robot should…

   j. Be capable of operating in minimal precipitation such as light rain or snow showers
   k. Be capable of operating in temperatures of 0°C to 35°C
   l. Be capable of four hours of continuous operation between battery charges
   m. Be capable of travelling four miles between battery charges
   n. Be equipped with a wireless emergency stop system that will bring the platform to a controlled stop before cutting power to the motor drives
   o. Allow for charging from a standard 120VAC power outlet while the machine is power up for non-mobile operation to allow for testing and development while charging

2. **Tour Guide/Live Interaction**

   The robot must…

   a. Allow a school staff member to initiate or stop a predetermined tour experience
   b. Be able to lead visitors to locations of interest on campus by a predetermined route

c. Stop at each location of interest and provide video and audio tour information pertaining to each location

The robot should…

d. Allow users to initiate tour experiences without the need for guidance from school staff
e. Allow visitors to choose from a set of locations on campus and lead them to the chosen locations based on a path planning algorithm
f. Allow users to start, stop, and repeat video and audio tour information at each location of interest
g. Provide video and audio tour information relevant to the robot's surroundings while travelling between locations of interest

3. **Autonomous Capability**

The robot must…

a. Be capable of determining its three dimensional location on campus, including translation and rotation within a 2D plane, in reference to a static global coordinate frame using GPS data
b. Update its location while in motion
c. Determine the required heading angle and linear distance to reach a goal position from its current position and use this data to navigate to the goal location
d. Be capable of detecting obstacles at a distance greater than 3 feet and less than 15 feet
e. Adjust its path while travelling to a goal location to avoid colliding with obstacles
f. Be capable of staying within safe areas of travel, i.e. sidewalks and pathways, while navigating to a goal location and avoiding obstacles

The robot should…

g. Be capable of planning a path to a goal location using a search algorithm which will construct a path using the edges between a set of predetermined waypoints on the campus map

Revised Client Statement

With the set of possible objectives reduced to a subset of objectives reachable during the term of the project and designed to leave a platform for enhancement by future project teams, a revised client statement was written to encompass the new set of objectives. The revised statement was as follows:

"The Guest Orientation, Assistance, and Telepresence Robot will be a platform for autonomous mobile robotic research. The platform will provide a software framework for inputting velocity commands and receiving feedback pertaining to the kinematic and operational state of the platform. Sensors will determine 2D geographic localization, 3D orientation, and a 2D picture of the platform's surroundings for navigation and obstacle avoidance purposes. A touchscreen interface will allow for live, human interaction and cameras will allow for telepresence capability. The platform will be capable of fulfilling a variety of roles on the WPI campus involving human interaction with a mobile robot."

4. ALTERNATIVE DESIGNS

System Overview

An outdoor mobile platform robot would comprise several key features, including a user interface, mapping, localization, and autonomous navigation ability, a base platform, and the necessary hardware and software, some of which can be used “off the shelf” and some of which must be designed. To achieve autonomy, the use of sensors and computers systems to process their data would be necessary. Furthermore, the robot would need one or more power sources, and the construction of an overall power system. While there are many ways to achieve these features, numerous design methods have been considered over the course of the project. Several of the design methods have been used for the final design of the robot, and are briefly discussed in this section, with detailed discussions the next section, “Final Design Methodology.” Also discussed in this section are the alternate designs that had been considered but not constructed.

Power System

As shown in the initial power system diagrams below in Figure 10 and Figure 11, an alternate design for charging the GOAT robot that had been considered was the use of a voltage detector (Intersil) to autonomously switch from Battery Power to Power Supply Power when plugged in for charging.

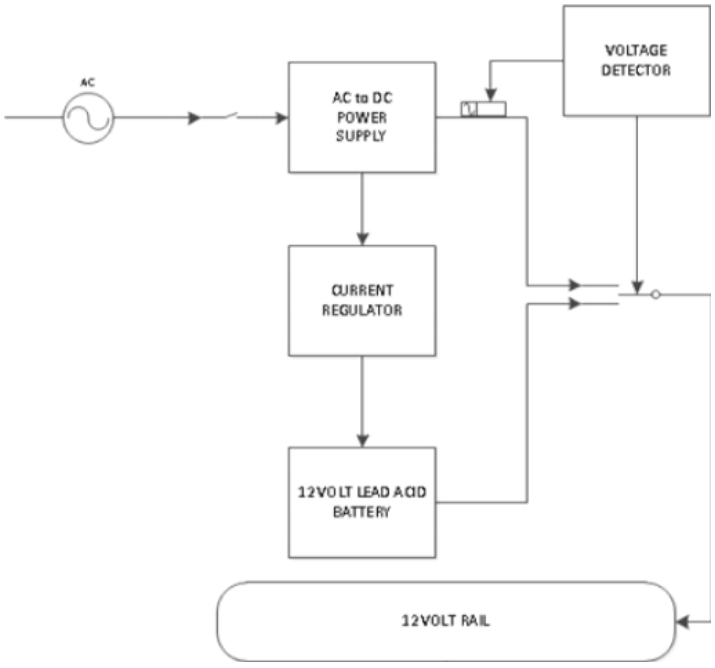


Figure 10: Initial power diagram (1), voltage detector for charging

The use of a microcontroller for smart switching had also been considered. Since it would be necessary to power lower voltage components, the construction of a buck converter had been considered (Donald Schelle).

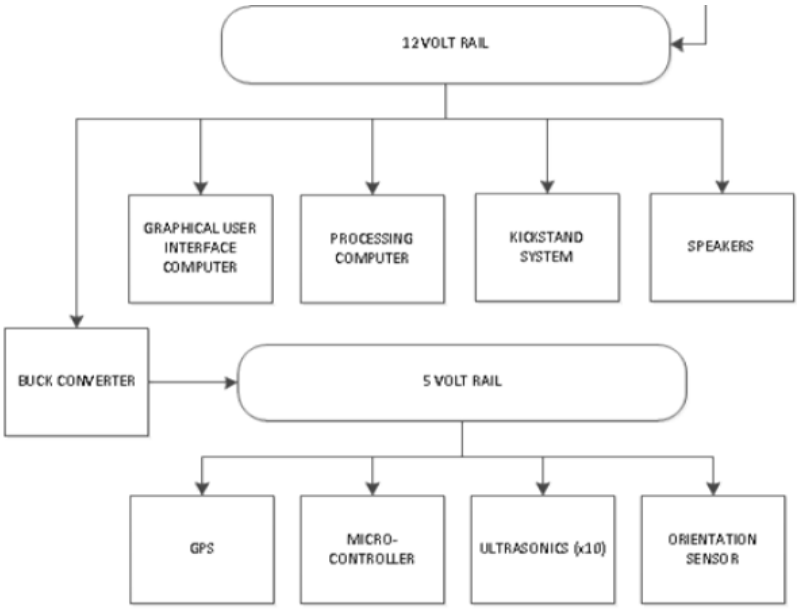


Figure 11: Initial power diagram (2)

In a buck convertor circuit (Figure 12), the transistor acts as a switch, consistently turning on and off such that the DC input voltage (Vi) appears as a square wave with frequency corresponding to that of the switch. The inductor and capacitor then act as a low pass filter (Starck), only allowing the DC component of the square wave to enter the load. The diode is necessary for consistent current flow while the switch is open. By varying the duty cycle (D) of the switch, the output voltage can be controlled; $V_o = D*V_i$.

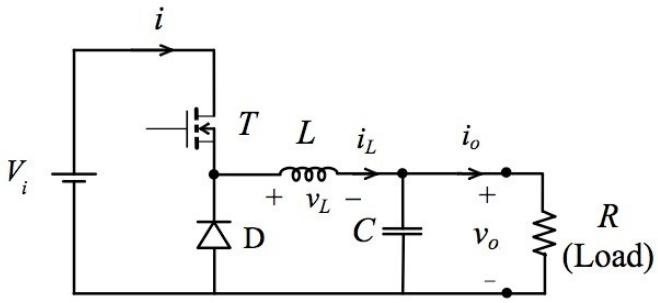


Figure 12: Buck converter circuit

A simulation of this circuit, created using PSPICE, revealed that a 45% duty cycle using a 100mH Inductor, 10µF Capacitor, and 50Ω Resistor would step a 12V source down to a 5V output, as shown in the graph below (Figure 13).

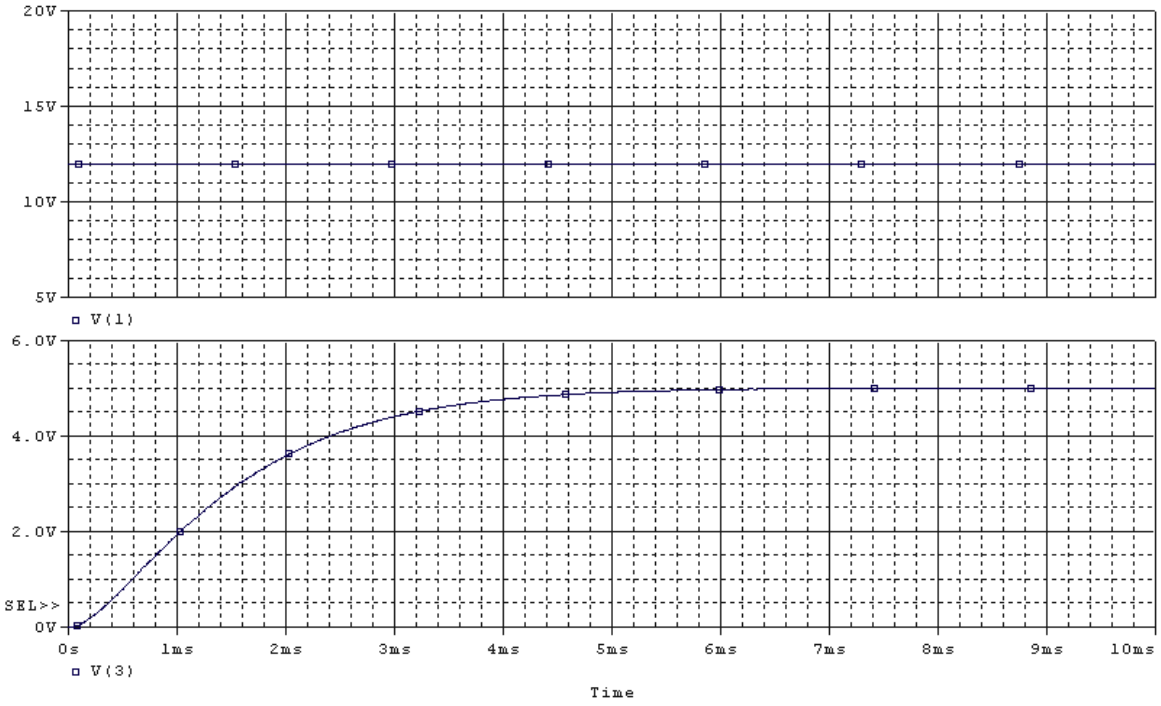


Figure 13: PSPICE buck convertor simulation

To control the duty cycle, the use of a 555 Timer had been considered, which outputs a square wave with frequency and duty cycle depending on chosen resistor and capacitor values. However, the minimum duty cycle a 555 timer can theoretically output is 50%. The required duty cycle to convert a 12V input into a 5V output using the buck converter is 41.66% (from ratio 5/12) to 45% (from simulation). However, by using an inverter or by wiring the transistor backwards to invert logic levels, the problem could be solved. The duty cycle of the 555 timer would then need to be from 55% to 58.33% (7/12), which is achievable by using the appropriate resistor and capacitor values.

Another alternate design for powering the system would be the use of solar panels. Since the GOAT Robot will spend considerable amounts of time outdoors, solar panels are an optimal choice of obtaining energy. However, since weather conditions vary, the solar panels will not output a constant voltage level. To insure that the voltage does not vary too greatly, a voltage regulator would be required. This can be done using a buck converter, described above, with an additional feedback loop for current regulation. The buck current is useful since the reduction in voltage will result in an increase in current. To attain the necessary voltage, multiple panels can be arranged in series.

Platform

The team decided early on to take advantage of a Robotic Mobility Platform (RMP) from Segway Inc. (Segway Inc.). These are robust, capable, and proven platforms designed for use as general purpose mobile robotics research platforms. An example of an RMP, the RMP 200 model, is shown in Figure 14.

Figure 14: The RMP 200 platform from Segway Inc.

RMP200

The RMP 200 model was identified as the most suitable model of RMP. Several factors contributed to this decision:

- Size and standup profile suited to interaction with humans
- Zero turn radius for maneuvering in tight spaces and crowds
- Long range and operation time suited for leading several tour groups in a given day
- Top mounting plate suitable for sensors and touchscreen interface
- Large payload area below top mounting plate for computer and other electronics
- Robust design made from thoroughly tested and proven Segway components
- Simple USB or CAN interface

One potential disadvantage to the RMP 200 platform was its balancing capability, which would have interfered with the use of computer vision due to a constantly changing platform pitch while in motion and while stationary. This could be avoided by putting the platform in “tractor” mode which disables the balancing feature but which requires the use of a castor wheel for stability. Additionally, the RMP 200, in its base configuration, does not offer a payload power system which would require all other additional platform electronics to be powered by a separate battery supply.

Prototype RMP

After extensive talks with the RMP team at Segway, it was decided that the project team would utilize and test a prototype platform based on a new Centralized Controller Unit (CCU) control architecture. This platform was statically stable by design and equipped with a payload power system by default. In addition, the new control architecture allowed for easier velocity control through normalized commands.

Platform Stability

The RMP 200 is a two wheel balancing platform that requires motor power to stay upright. With the assumption that the project would utilize an RMP 200, platform stability was a prime concern and several scenarios were considered for addressing the issue.

Dynamically Balanced Platform

Initially, the scenario of an unmodified, self-balancing RMP 200 platform was considered for its implications. In its default configuration, there is no mechanism to prevent it from falling over if power is lost, thereby damaging the machine and its payload and also posing a risk of injury to nearby people. Since the intent was for the robot to interact with campus visitors and operate within crowded public spaces, the risks posed by an unmodified platform were considered far too great. In addition, the platform and the hardware which would be mounted on it to create an autonomous system represented a significant financial investment which could be lost if the robot were to suffer a loss of stability. For these reasons, a dynamically balanced platform that was stable only under its own power was considered not feasible.

Landing Gear System

Preliminary designs were made for a landing gear system that could be deployed prior to a normal, controlled shutdown of the platform. The landing gear would deploy prior to the loss of motor power, hold the platform upright in the absence of motor power without using power of its own, and retract when motor power was restored.

The landing gear system was to consist of three main components, the mechanical landing gear, a motor driver and momentary double-pole, double-throw (DPDT) switch for operating the landing gear actuator, and a PIC32 microcontroller (Microchip Technology Inc.) for controlling the motor driver. The sketch in Figure 15 shows the basic concept for the mechanical landing gear system. A linear actuator would retract and pull on the cables which would extend the landing gear, creating tension in the cables by winding up torsion springs attached to the gear. When the actuator extends, the springs would retract the gear and take up slack in the cables. The linear actuator is not back drive-able so the system would not require power to stay deployed.

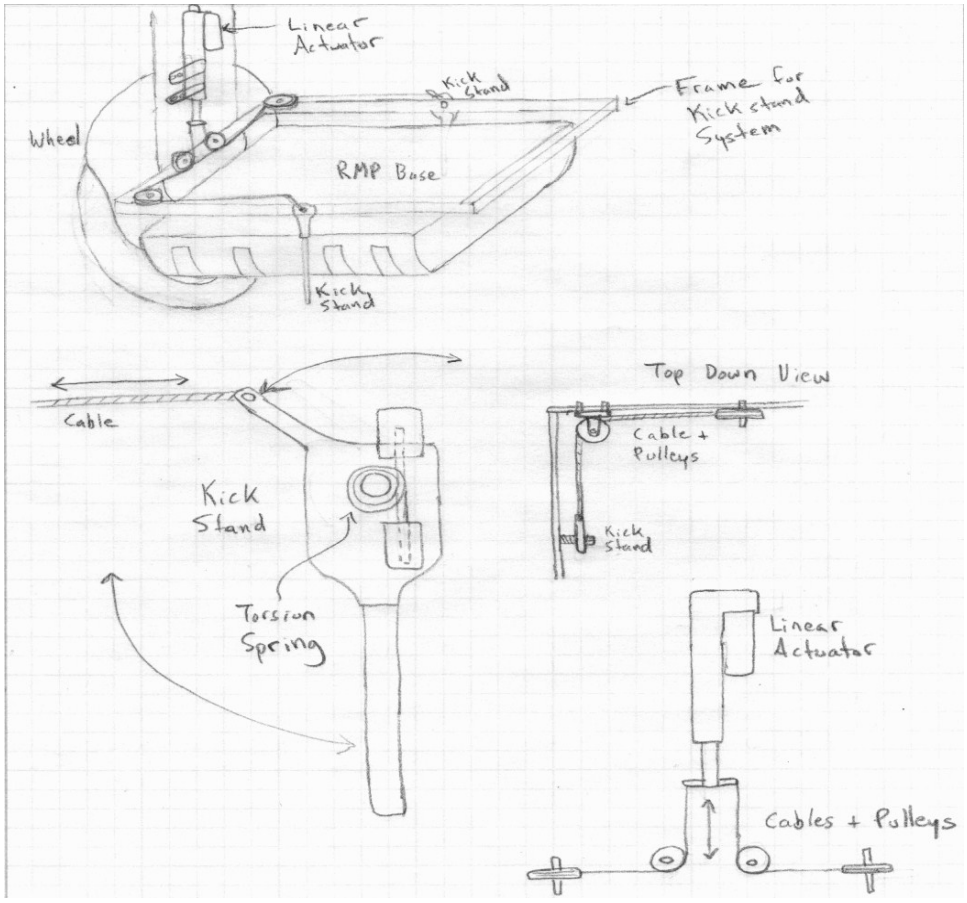


Figure 15: Proposed landing gear mechanism

The linear actuator was to be driven by an off-the-shelf motor driver module (Pololu) controlled by a PIC32 microcontroller. The PIC32 would receive commands from the main system PC, and would also send a stop command to the RMP platform and deploy the gear in the event of a communications loss by the PC. The actuator could also be driven by a DPDT switch which would connect the motor directly to the main system power to actuate the landing gear without the PIC32 or motor driver. The overall proposed landing gear control system architecture is shown in Figure 16.

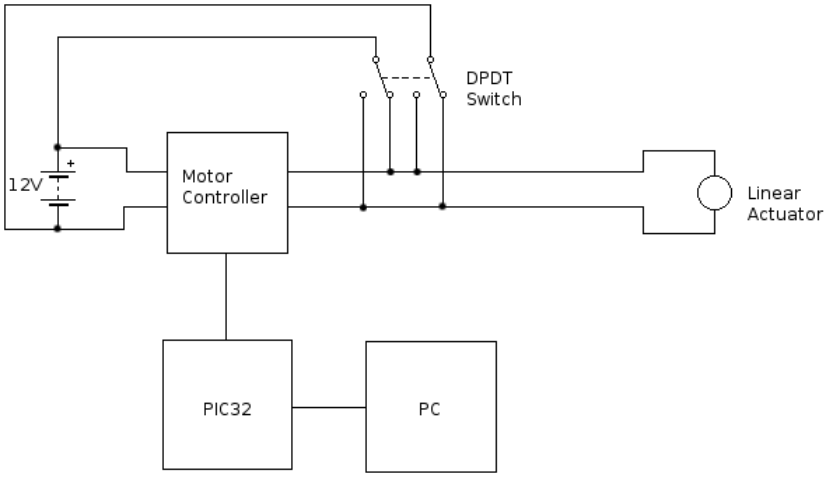


Figure 16: System diagram for landing mechanism control

Statically Stable Platform

Though the landing gear system was considered a reasonable method of stabilizing a dynamically balanced platform, there remained a significant risk of stability loss if the platform suffered a catastrophic failure and the landing gear could not deploy in time to prevent damage to the robot or injury to anyone in close proximity. Therefore, when the team was given choice of RMP platforms which included a statically stable option, the choice became clear; a prototype, 2-wheel RMP was selected with a rear castor wheel for stability. To increase the stability, the heaviest component of the system, a 12V lead gel cell battery, was mounted above the castor to shift the platform's center of gravity toward the rear. The castor with the battery mounted above it can be seen in Figure 17.

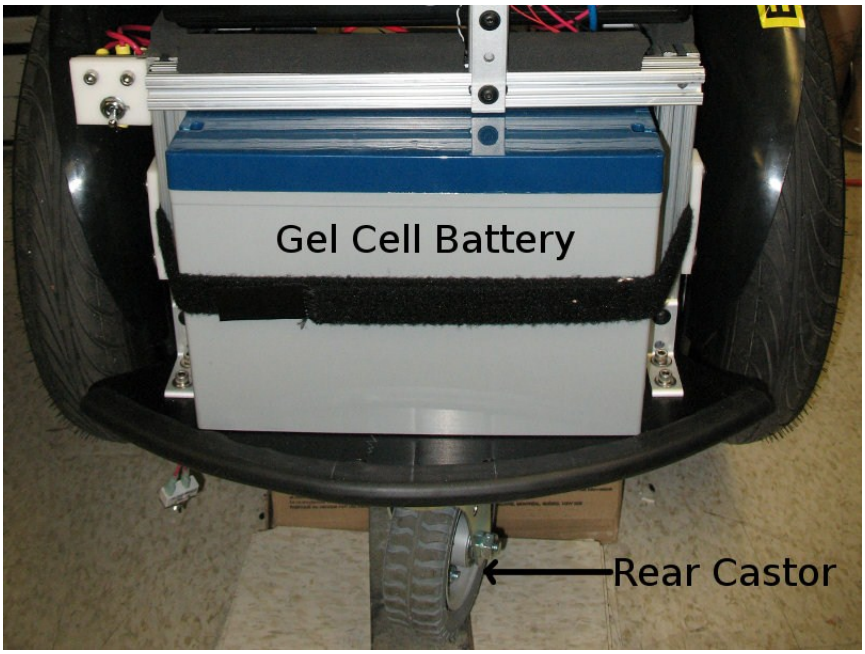


Figure 17: Gel cell battery positioned above rear castor for stability

User Interaction

Of great importance for a tour guide robot is the ability to interact with users to assist them and provide information. Several methods of interaction were considered and planned for in the robot's design. Some were implemented, others were provided for in the final system design, and yet others were discarded.

Campus Map

For interacting with visitors, the initial plan was to display a 3D campus map on the touch screen interface for the dual purpose of indicating a user's location and allowing them to choose destinations for the robot. A method was devised of creating a 3D campus map in Trimble Sketchup (Trimble Navigation Limited) using building models imported from Google Earth and ones produces locally. This would allow building models to be automatically placed relative to each other based on their geographical locations, and building and landscape textures could be added using Google Earth and Google Maps Street View. All 3D model data would be exported to a COLLADA file which is a universal 3D exchange format (Khronos). COLLADA is an XML format which means all necessary geometry and

visual data can be extracted using an XML parser. This data can be used to produce a 3D interactive map for the user interface. To demonstrate this process, building models for Gordon Library, East Hall, and East Hall Garage were imported into Trimble Sketchup. The models were then exported to a COLLADA file and geometry data was extracted and plotted using Matlab (The MathWorks Inc.). The results can be seen in Figure 18.

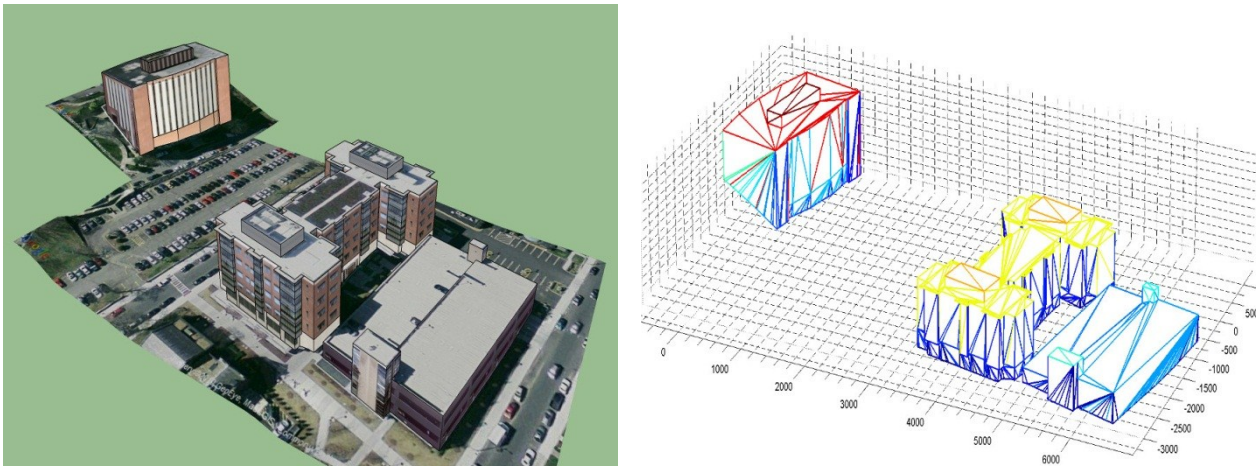


Figure 18: Google Sketchup models of Gordon Library and East Hall (left) and exported COLLADA data plotted with Matlab (right)

While a novel and attractive idea, the amount of effort required for creating a custom, interactive 3D map was considered far beyond the scope of the project. However, the method explored could be considered a valid idea for future development of user interfaces for the robot.

Gesture Recognition

Two types of sensors considered for the project were stereo cameras and a Microsoft Kinect (Microsoft). Both of these systems are capable of producing 3D images of their surroundings which can then be used to recognize familiar shapes such as the human body, otherwise known as skeleton tracking. Figure 19 depicts an example of skeleton tracking in use.

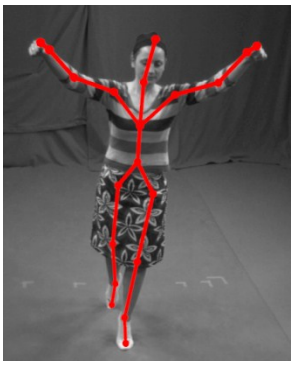


Figure 19: an example of skeleton tracking through computer vision

In fact, there exists a Kinect SDK (Microsoft Inc.) which provides skeleton tracking. Skeleton tracking was considered potentially useful since it would allow users to indicated locations or objects to the robot without a standard interface. As with the 3D campus map, the idea was considered novel and

attractive, though the extensive development time required for this type of user interaction precluded its use.

Speech Recognition

A second alternative to standard interfaces, speech recognition, was also considered. There exist open source API's for speech recognition which allow users to command computer systems using audible speech (Google). While certainly a possibility for future development, and once again a novel and attractive idea, the amount of development time required for a functioning and reliable speech recognition system was considered beyond the scope of the project and was left as a possibility for future development.

Synthesized Speech

Though not technically interactive, allowing the robot to speak to users in a monologue was considered as a viable option for providing information. This could have been accomplished by recording a human narrator reciting a script for what the robot would say. However, this method was considered impractical since it would require new recordings to be made whenever a corresponding change was made to the robot's script. Also, people tend to expect a robot to sound like a robot and could become disconcerted when hearing a human voice issuing from a machine. Instead, there are several methods available for converting text to synthesized speech, many of which do not require purchasing software. These methods can be used to produce audio files which can be played back for users of the robot. Due to its convenience, and impracticality of other interface methods due to limited project scope, it was decided that providing information to users using speech synthesis was a bare minimum requirement.

Localization

The preliminary plan was to perform initial robot localization on the campus map by combining data from a 3-axis orientation sensor, a GPS module, and platform wheel encoders. Since the absolute accuracy of this data in global coordinates would be limited by the accuracy of the GPS receiver, which is approximately 2 meters for most low cost models, the robot would also utilize stereo cameras to refine its pose.

The first step of the pose refinement process would be to use the cameras to obtain point cloud data along the robot's allowed navigation paths from which could be extracted land mark features using Point Cloud Library, or PCL. (Open Perception Foundation). This technique has been previously utilized for 6-DOD pose estimation (R. Huitl). The landmarks would come in the form of plane features located outside of the maximum GPS error distance from the robot, but close enough to provide a reasonably large change in perspective for a relatively small change in robot pose. Since this would take place prior to the actual operation of the robot, the data could be heavily processed to provide the most accurate plane features without concern for processing time. Once the landmark planes were generated, they would be permanently stored on the computer's hard drive as plane coefficients in the form of a, b, c, and d where ax+by+cz+d=0, along with the points that were used to create them. The Figure 20 depicts permanently stored planes with their accompanying points.

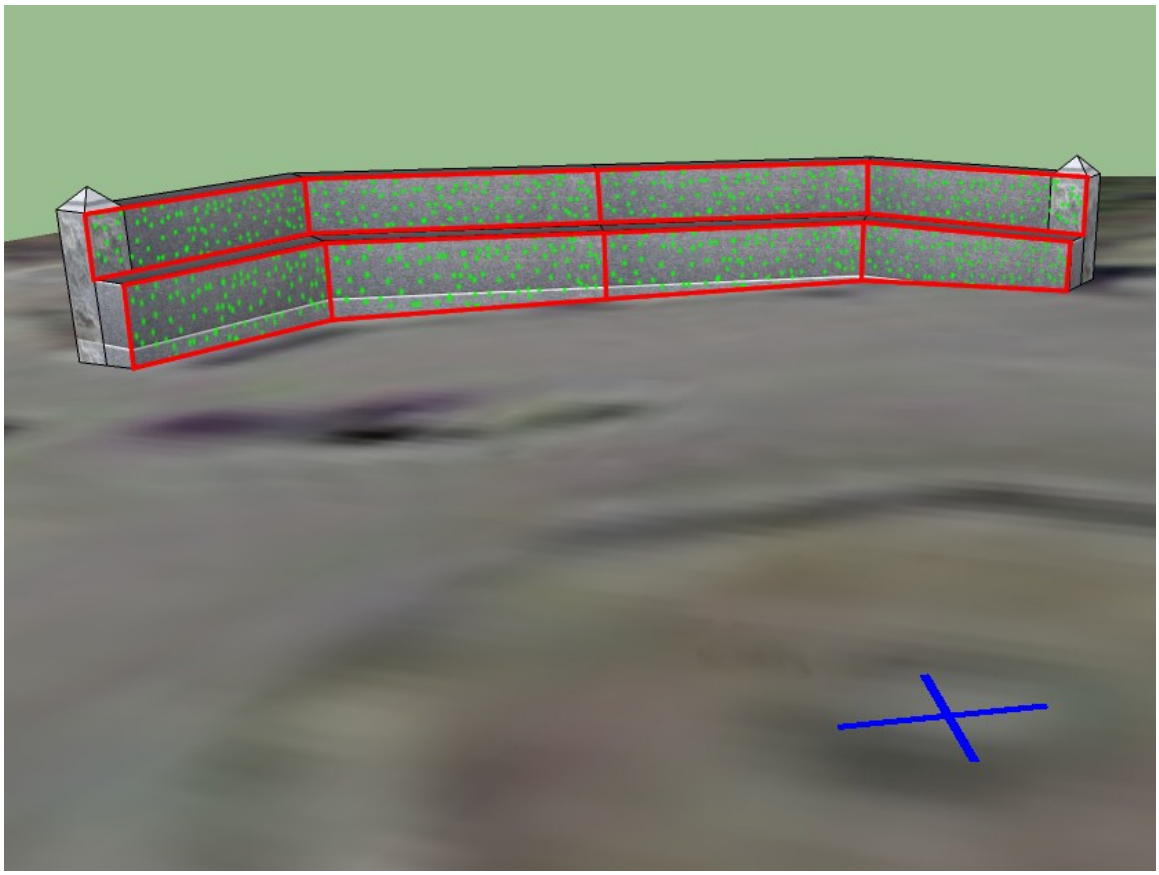


Figure 20: Plane features with points shown in green. The initially determined robot position is shown as a blue cross.

To refine its localization, the robot would take stereo camera images of its surroundings at predetermined checkpoints and use these to generate raw point clouds. The estimated location of the robot based on GPS, orientation sensor, and encoder data will be used to access the previously stored landmark planes the robot expects to see. The raw point clouds and the expected landmarks will be compared using PCL to produce a transformation matrix representing the error in the robot's initially determined pose. Figure 21 depicts the pose refinement process.

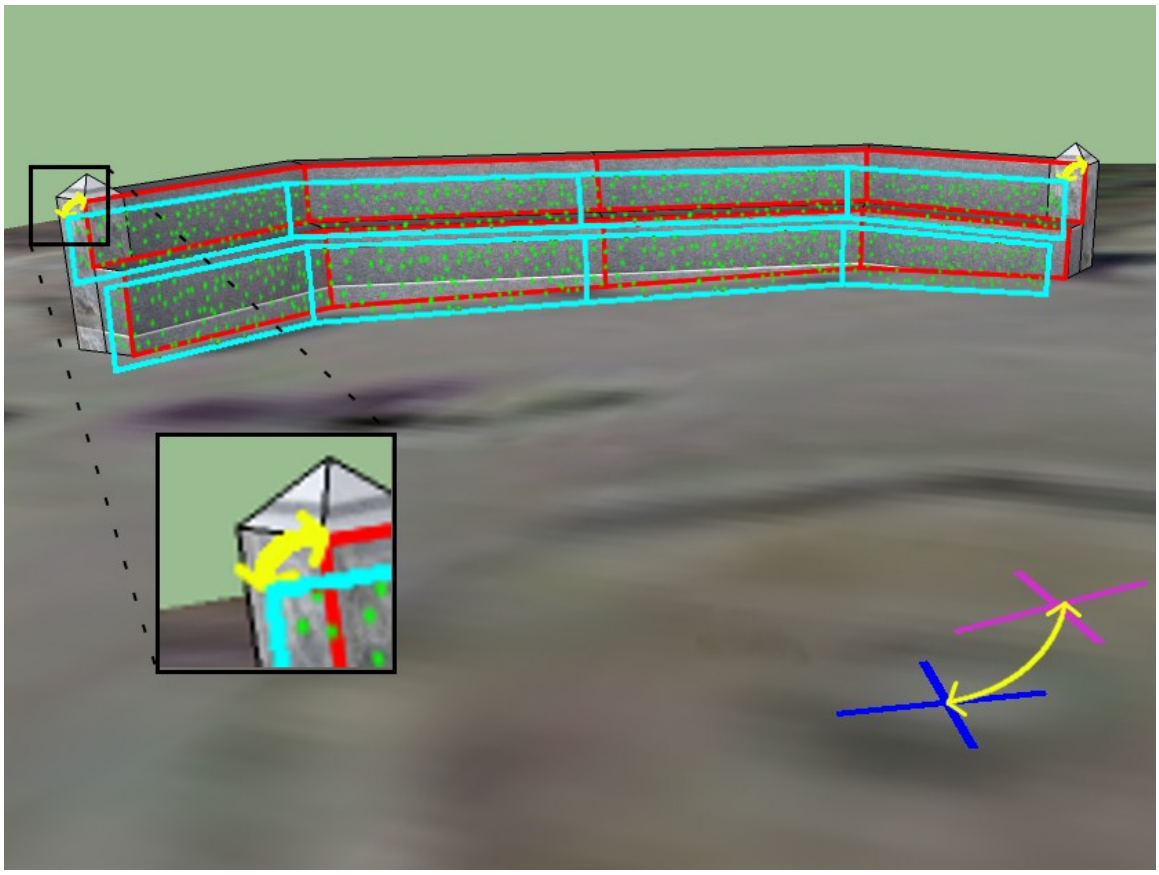


Figure 21: Transformation between expected position (blue cross) and actual position (purple cross) determined by comparing current perspective planes (light blue) to expected perspective planes (red).

Unfortunately, this type of localization refinement would require extensive effort in mapping the campus and refining the algorithms for registering the point cloud data from the stereo cameras with the preloaded map. Instead, it was decided to test and validate the system using only the GPS and orientation data. Though not sufficient for reliable navigation on the campus, it was considered enough to demonstrate the robot's autonomous capabilities.

Sensors

During the design process, many different sensors were considered for this system as they are some of the most vital components to an autonomous robot. The design of the platform needed to have certain key types of sensing in order to be a fully autonomous system and thus a list of sensor data types was generated. Because of its interactions with people and working in a changing environment, obstacle detection and avoidance was crucial and so some form of range sensing as well as point cloud

generation data was needed. Also, the platform must know its position and orientation and so some form of localization and accurate navigation was needed using a 3D coordinate system.

The use of range finders is vital to autonomous navigation as it allows for obstacle detection and avoidance. Of the many ultrasonic range sensors available, mainly two ultrasonic sensors were considered for the design and implementation: Parallax Ping, and MaxBotix MaxSonar (Figure 22). MaxBotix sensors have several different models of which the price increases as the benefits increase. These ultrasonic sensors also provide analog and digital outputs, but because the Parallax Ping sensor provides the best accuracy, implementation, and documentation for the price, it was chosen as the primary ultrasonic sensor.

Figure 22: MaxBotix LV-MaxSonar-EZ1 Ultrasonic Range Finder

Also because of budget limitations, a Light Detection And Ranging (LIDAR) sensor was not used as it would be prohibitively expensive and would require either 2 180^{o} LIDARs or 1 180^{o} LIDAR in the front and ultrasonic sensors in the rear as there is no good location on the platform to mount a LIDAR with greater than 180^{o} span. However, using a LIDAR, accuracy could increase drastically as well as maximum distance detection.

The Kinect Sensor (Figure 23), designed by Microsoft for the Xbox, was part of the initial design as it provided a full 2.5D point cloud, which would easily allow for mapping and navigation. Also, the ideas of gesture recognition as well as voice commands could have easily been implemented using the Kinect because it had software for human gesture tracking as well as a built in microphone which could take audio commands.

Figure 23: Microsoft Kinect Sensor For Xbox 360 (Microsoft)

The Kinect sensor idea was however scrapped because it was unable to process the information outside given that the sunlight interfered with its built-in infrared sensor. However, if the platform was modified to navigate indoors, the Kinect could be utilized as the sunlight would no longer interfere and thus enable indoor mapping and navigation easily with voice and gesture recognition.

Computer Architecture

Once the idea of using a Kinect sensor for environment mapping was scrapped, it became clear that computer vision was the most viable option. Computer vision requires a great deal of processing and so it was decided that significant attention should be paid to the computer or computers aboard the robot with regard to the load that would be put on them during robot operation. In addition to computer vision, the intent was to display video tour information and possibly an interactive map on a touch screen interface. Displaying video and a map would require yet more processing power and could possibly overload a single computer. As a result, the idea of using two computers was considered, one for processing data from sensors and controlling the platform and one for the user interface.

A system with two computers raised some concerns about power consumption. The question of whether or not to use two computers had to deciding factors: how much, if any, the power consumption would increase over one computer and a touch screen monitor, and could the main PC handle the processing load on its own. To help answer the second factor, a test was run, on the personal computer of an MQP team member, using stereo vision processing at a camera resolution of 640x480 pixels and a frame rate of 10Hz. The processor load was observed under these conditions to help gauge the system's ability to handle the most processor intensive task of computer vision. A screen shot of the processor load can be seen in Figure 24.

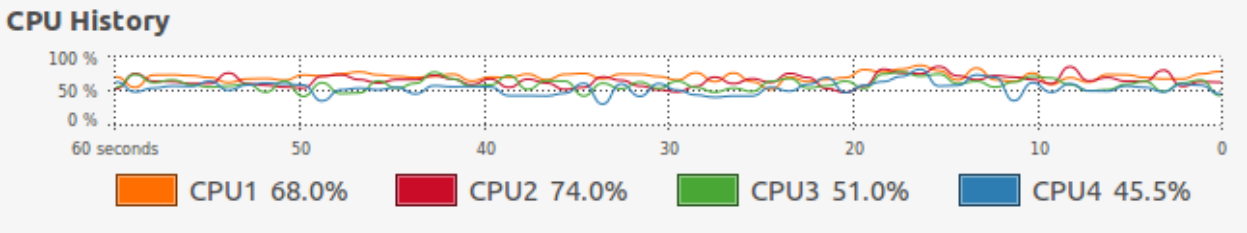


Figure 24: CPU load for main platform computer during computer vision test

The stereo vision test indicated that generating a disparity map and point clouds from the camera images required approximately 2/3 of the CPU resources for the main system computer. Considering all other tasks that would be required by a platform with full autonomous capabilities and a user interface, the remaining CPU resources were likely to be inadequate. For this reason alone, the decision leaned heavily in favor of a dual computer architecture.

Still to be considered was the power supply issue. After extensive research into the various options for touch screen monitors and all-in-one computers with a touch screen interface, it was found that all-in-one computers required very little additional power over comparatively sized touch screen monitors, and even less power than larger touch screen monitors. Furthermore, virtually all modern computer monitors contain their AC/DC power supplies within the body of the monitor itself which would require opening the monitor case to connect the monitor to the system power supply. Many all-in-one computers have external power supplies similar to a laptop PC which would require only that the DC wires be cut and spliced to the system power. Finally, a suitable all-in-one PC was found with adequate processing power, low power consumption of less than 45 watts, and an external power supply that was easily accessed (see Figure 25).

Figure 25: All-in-one PC chosen for the user interface computer

Considering the power supply issue and the CPU load for the entire system, the dual computer architecture with a main system computer and a touch screen all-in-one for the user interface became the clear choice.

5. FINAL DESIGN METHODOLOGY

System Overview

In order for the overall design to function as one intelligible system, the functional blocks that comprised of the robot had to be connected via the appropriate power and communication lines. An overall system block diagram labeling these connections is shown below in Figure 26, where red arrows are used to represent power while blue arrows are used to represent communication lines. In both cases, the arrowheads point toward to system block that is receiving power or being communicated to.

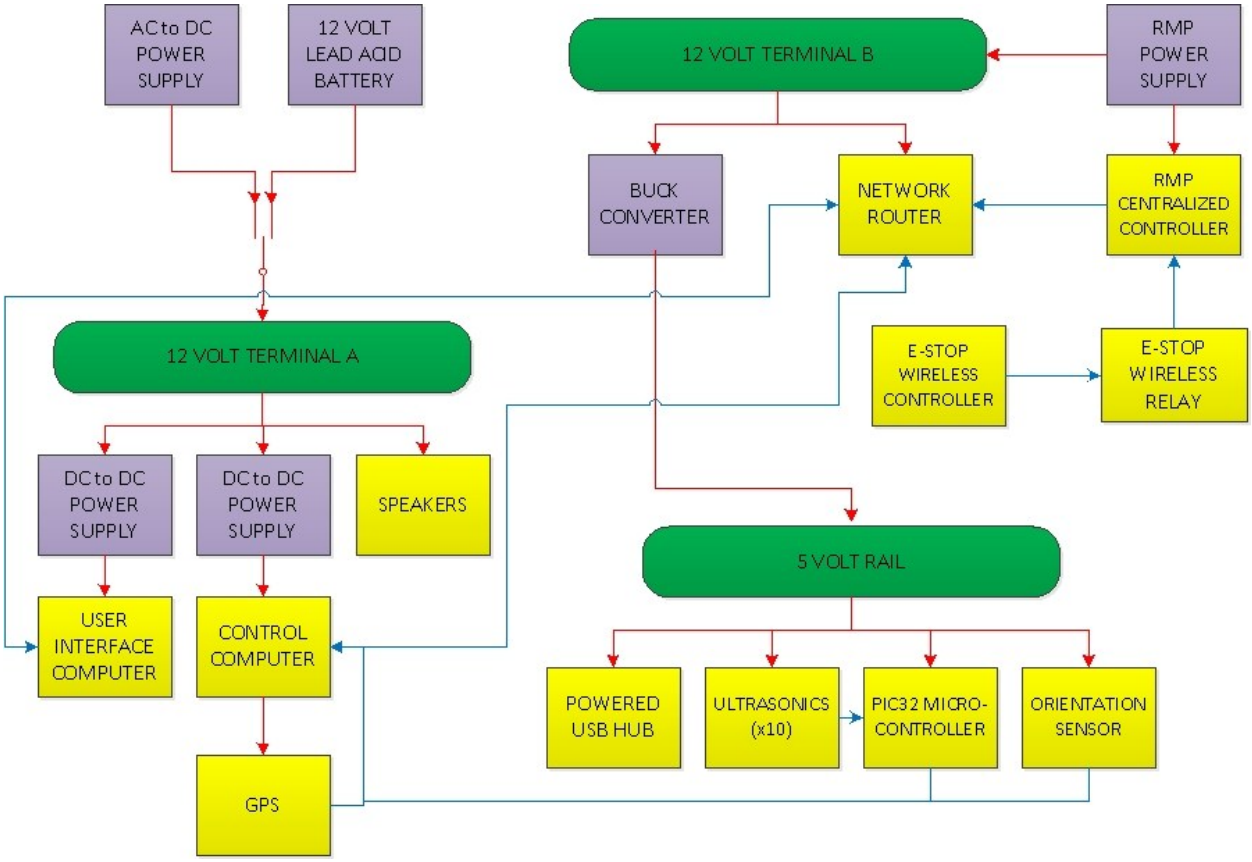


Figure 26: Overall system block diagram

Power System

The GOAT Robot was powered by two 12VDC rails, one of which could be alternately supplied by a gel-cell battery or an AC-DC power supply (Figure 27), while the other was supplied solely the RMP's built-in DC-DC power supply, which drew its power from one of the RMP's two 72VDC lithium-ion battery packs.

Figure 27: The gel-cell battery and AC-DC supply for the GOAT Robot.

The two 12VDC rails were accessed by two terminal blocks labeled Terminal A and Terminal B (Figure 28). A double-pole, double-throw (DPDT) switch (Figure 29) was used to alternate between the gel-cell and AC-DC supplies for Terminal A, while Terminal B provided access to the RMP power supply. When the robot was plugged into AC power, the DPDT switch was used to switch from battery power to AC-DC supply power, which would both recharge the gel-cell and RMP batteries as well as supply the system with power.

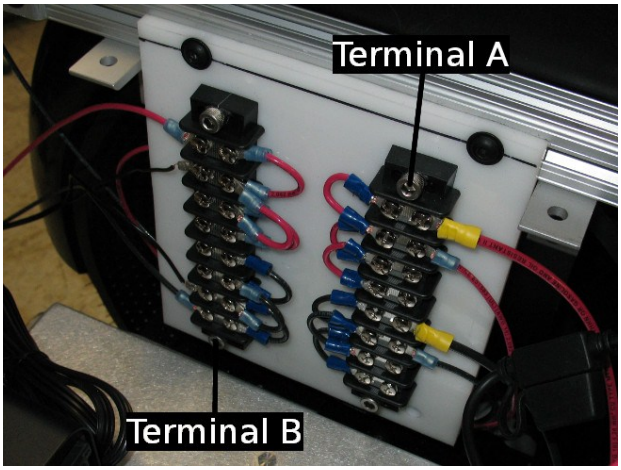


Figure 28: Power terminal blocks

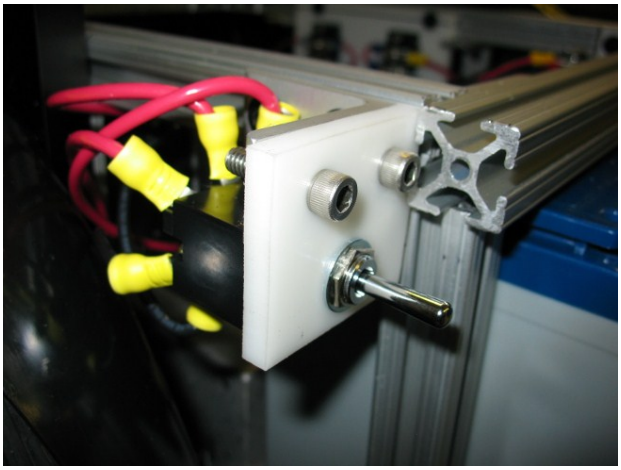


Figure 29: DPDT switch for terminal A

Terminal A was used to power the speakers, as well as both the User Interface and Control Computers. Terminal B was used to power the network switch, the wireless relay, and the 5-Volt buck converter board. In turn, the buck converter was used to power the 5 Volt components of the robot, including the GPS sensor, orientation sensor, powered USB hub, ultrasonic sensors, and the PIC32 microcontroller.

Collision Detection Sensing System

The collision detection system consisted of ten ultrasonic sensors, a PIC32 microcontroller for interfacing with the sensors, and a breakout board for the PIC32. The purpose of the system was to detect obstacles within two meters of the robot which were invisible to the stereo cameras and considered an imminent collision threat.

Breakout Board

In order to obtain access to all the necessary pins of the PIC32 microcontroller (Microchip Technology Inc.), including those that would be used for the platform design aspect of the project as well as future additions, a breakout board was designed and manufactured. The design of the board was completed using the software Altium Designer (Altium Limited), in which a schematic was created, as shown in Figure 30.

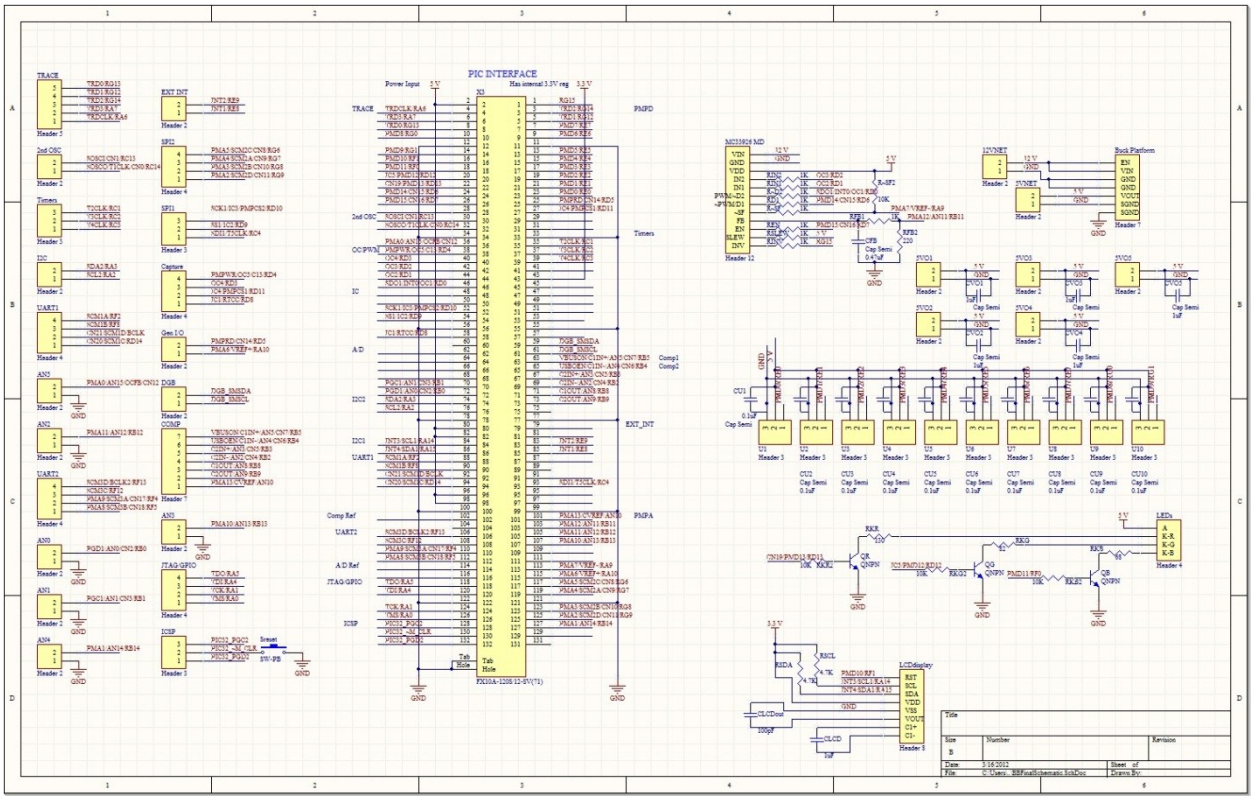


Figure 30: Breakout Board Schematic

The individual components shown on the right side of schematic are the buck platform connector, motor driver connector, and five 5 V output connectors above ten ultrasonic sensor, LED and LCD

display connectors. These components are connected to the appropriate pins of the microcontroller using Net labels. The 132-pin connector near the center is the PIC32 interface where a PIC32 development board was connected. The components arranged on the left of the schematic are the connectors allowing access to remaining unused pins of the PIC32 microcontroller.

Using this schematic, a PCB design was constructed with Altium. The use of Net labels on the schematic generated lines resembling wiring that signified where pins were to be routed together. The final PCB design, shown in Figure 31, was manually routed onto a 4 by 6 inch layout, with four layers.

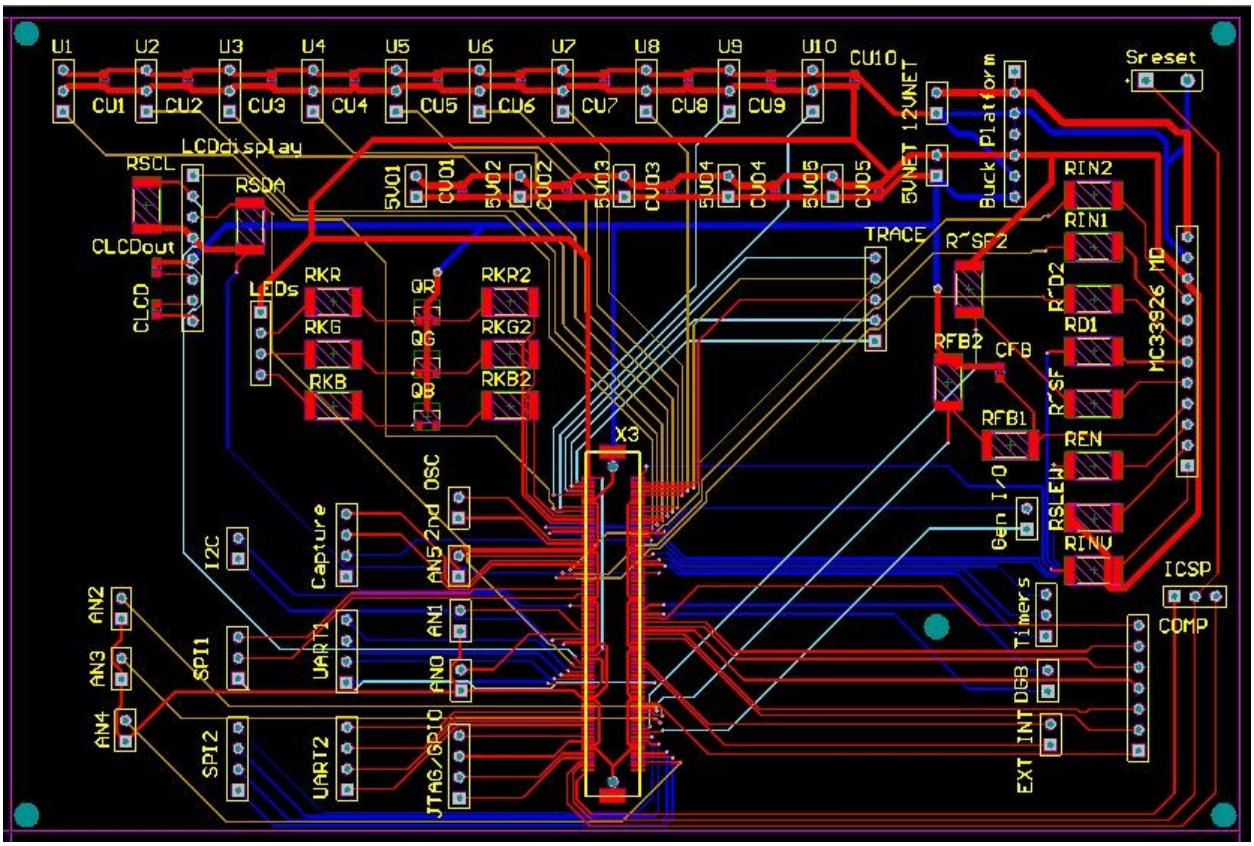


Figure 31: Breakout board PCB design

Upon completion of the PCB design, the necessary manufacturing “Gerber” files were generated. The board was manufactured by Advanced Circuits, and is shown in Figure 32.

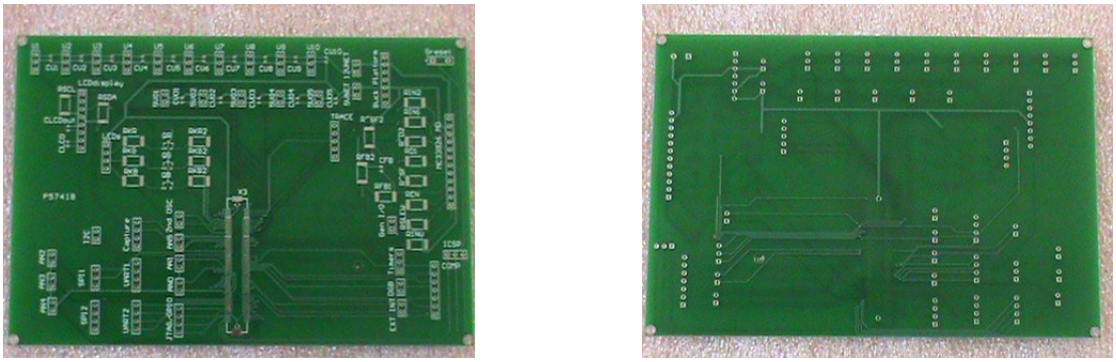


Figure 32: Breakout board manufactured PCB top view (left) and bottom view (right)

The breakout board was then assembled with the appropriate connectors and electrical components, the PIC32 development board was connected, and the breakout board was attached to a mounting plate for installation in the case for the robot’s control computer (Figure 33). To power the PIC32 microcontroller, a buck converter board was connected to the breakout board and mounted to the underside of the plate, allowing for the voltage from the 12V terminal block to be brought down to appropriate level of 5V.

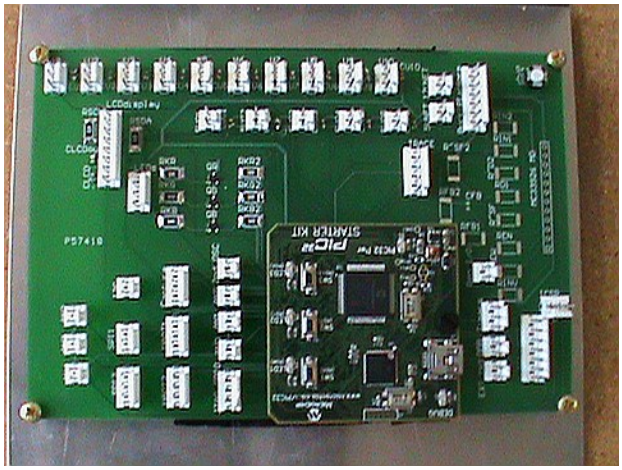


Figure 33: Breakout board assembly and mounting plate

Ultrasonic Sensors

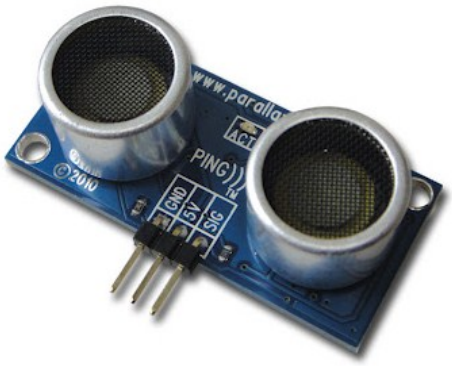


Figure 34: Parallax Ping Ultrasonic Sensor

The Parallax Ping Ultrasonic Sensors (Parallax), shown in Figure 34, use digital logic and are connected to the PIC32's PMD port which allows for digital I/O. Programmed using C-coding, a 5µs start pulse is sent to the signal pins of the ultrasonic sensors through the PMD ports, which are initialized as output pins. The pins are then configured as inputs to detect the echo pulse, which is generated by a sonar wave leaving the ultrasonic sensor, bouncing off an object, and then returning to be detected by the sensor. The echo pulse time is proportional to the distance of the object. Therefore, a counter is used to measure the pulse time, allowing for a corresponding integer count value to be recorded for each of the sensors, and stored into an integer array. The integer array is then converted into a character array, which is sent serially to the control computer, after which the computer converts the count values into ranges.

To mount the sensors to the robot, individual brackets for each sensor as well as front and rear mounting plates were designed using CAD software and cut from acrylic sheets using a laser cutter. Figure 35 shows the ultrasonic sensors assembled with the brackets and mounting plates.

Figure 35: Ultrasonic sensors attached to the acrylic brackets and mounting plates

The ultrasonic sensors are mounted around the robot as shown in Figure 36. The spread allows for overlap in the cones for the front sensors which increases the likelihood of obstacles being detected in the robot's path. The backside only has 3 sensors to track the distance at which the people are trailing the platform and to watch for anything else approaching from the rear, but the platform moves primarily in the forward direction, so monitoring the backside is not as critical as the front.

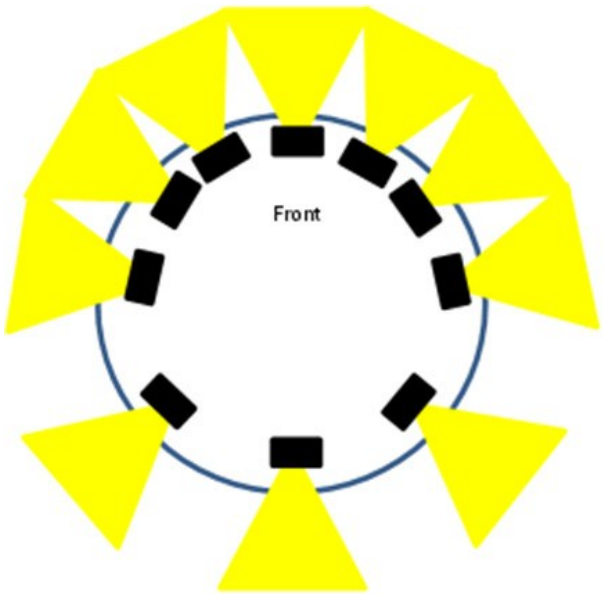


Figure 36: Top down view of ultrasonic sensors (black) mounting locations with sound projection width (yellow)

Platform

For construction of the platform, the Segway RMP was the initial base on which the robot's systems were built. The platform construction consisted of 3 phases: conceptual design, digital design, and physical construction. The conceptual phase involved considering the design and mounting possibilities and how they would allow for safety, security, accessibility, neatness, and durability. The gel-cell battery, due to its weight of approximately 60 lbs., required a placement at the bottom of the platform over the rear castor wheel in order to lower the robot's center of gravity and increase stability. The control computer, due to its larger size, was limited to a placement below the top plate of the RMP where it would not interfere with the mounting of the user interface computer, which required a position above the top plate for easy accessibility. The GPS sensor needed to be placed at the highest possible point in order to receive signals from the maximum number of satellites for optimum accuracy. The orientation sensor had to be centered in the horizontal plane of the robot for the most accurate yaw readings and could not be surrounded by metal or electronic components which could disrupt the readings from its compass module. The stereo cameras needed to be front and centered and high enough to view the area in front of the robot. The ultrasonic sensors needed to be on the outer-most perimeter of the RMP and high enough so as not to receive echoes from the ground.

Starting with several preliminary designs, the digital design phase began using existing CAD files for the RMP and mounting framework, as well as additional CAD models created to represent the system components. After several iterations, a final design configuration was created and further modifications were made in order to ensure the components fit correctly. The final configuration CAD design can be seen in Figure 37. Finally, the physical assembly of the system involved cutting the framing pieces according to the design, drilling and threading any necessary mounting holes, and mounting the system components onto the RMP.

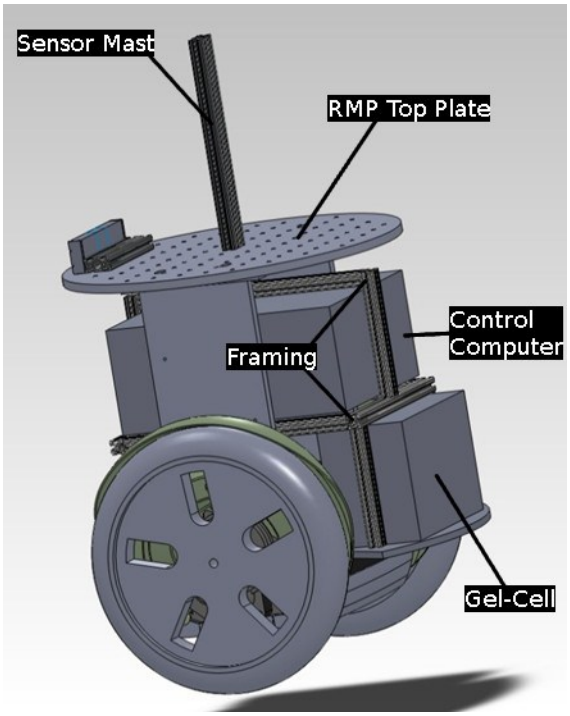


Figure 37: Solidworks Model of Platform Design

For the physical construction of the mounting frames, 80/20 aluminum framing was used to for the reasons that it's moderately light, strong, easily modifiable, and provides many easy attachment methods. Four-slotted, 1" square 80/20 aluminum was used for most of the framing except for the sensor mast which used 1" x 2" six-slot aluminum. The cameras and interface computer were mounted using other types of aluminum framing which were available to the MQP group. The stereo vision camera frame and the GPS and orientation sensor mount were designed in Solidworks and constructed using a thermoplastic 3D printer. The cover plate for the stereo vision cameras, as well as the brackets and mounting plates for the ultrasonic sensors were made using with laser cut acrylic sheets which were bolted and cemented together.

Other Sensors

The GOAT robot is an outdoor autonomous platform and so it must be able to sense its environment in order to adapt to and interact with it. The final platform includes the following sensors: 2 Logitech webcams to form a stereo camera module, 10 Parallax Ping ultrasonic sensors (Parallax), ND-100S GPS receiver (USGlobalSat Inc.), a CHR-UM6-LT orientation sensor (CH Robotics), and an Inertial Measurement Unit (IMU) and wheel encoders located in the RMP base. The sensors as they are located on and within the robot are shown in Figure 38.

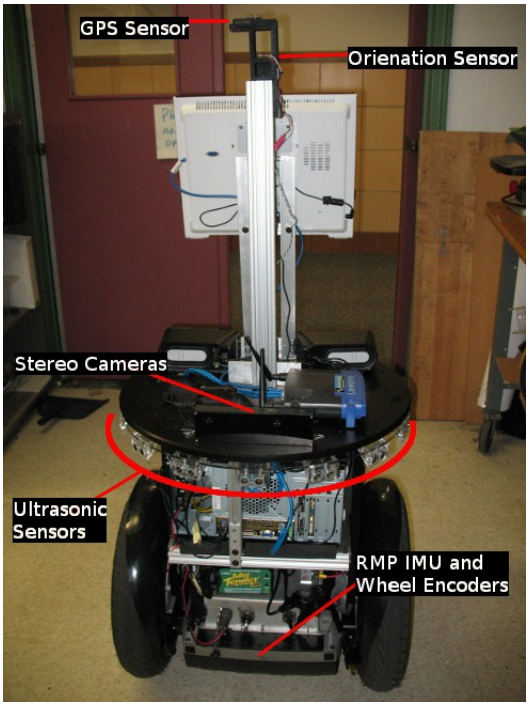


Figure 38: Overview of sensors for the GOAT Robot

The webcams were combined together into a stereo vision system to generate point cloud data which can be used for 3D mapping and obstacle detection and avoidance. Stereo vision operates by finding similar features in the images from the two cameras, from which is calculated a disparity, or an offset between the similarities. The closer an object is to the cameras, the greater the disparity. This creates a limitation for stereo vision when viewing closer objects for two reasons. First, the greater the disparity, the more processing is required to locate the similar regions in the two images. Second, if an object is too close to the cameras it will only be visible to one of them. The effect is that stereo cameras have a blind spot in an area immediately in front of them. To compensate, the ultrasonic sensors compliment the stereo vision system in that they collect range data on obstacles within close proximity to the robot, data which can be used for obstacle detection and avoidance.

The GPS, orientation, and IMU sensors allow for localization. The data received from the GPS is mapped to a position on the earth which is used to find an X and Y translation from a global coordinate frame located on the campus which is defined in the robot's localization software. The orientation sensor, with its built in compasses, can determine the heading of the robot, or yaw angle, and the RMP IMU provides pitch and roll angles. The pitch, roll, and yaw data is combined with the X and Y translation to create the initial 3D pose for the robot, with the Z translation always set to zero. This was considered

sufficient to properly determine the orientation of the 3D point clouds relative to the robot's body, after which the point clouds would be collapsed into 2D obstacle data, thus not requiring Z-axis translation information. Once an initial pose was determined, the X and Y translations, as well as the yaw angle, were updated while the robot was in motion using the RMP wheel encoder data.

Stereo Cameras

The stereo cameras were constructed using two low-cost webcams. The model chosen was the Logitech C260 (Logitech) shown in Figure 39 in their original casings.

Figure 39: Logitech C260 Webcams

The cameras were disassembled and the circuit boards placed into a new case to hold them securely with a fixed spacing of 120mm between the lenses, a spacing chosen as ideal for long range stereo vision which was desired for the possibility utilizing the cameras for pose refinement using relatively distant landmarks. Figure 40 shows the final casing of the webcam; comprised mainly of a 3D printed part, with a laser cut acrylic cover plate to protect the camera circuit boards and to shield the lenses from glare that could interfere with the stereo image processing.

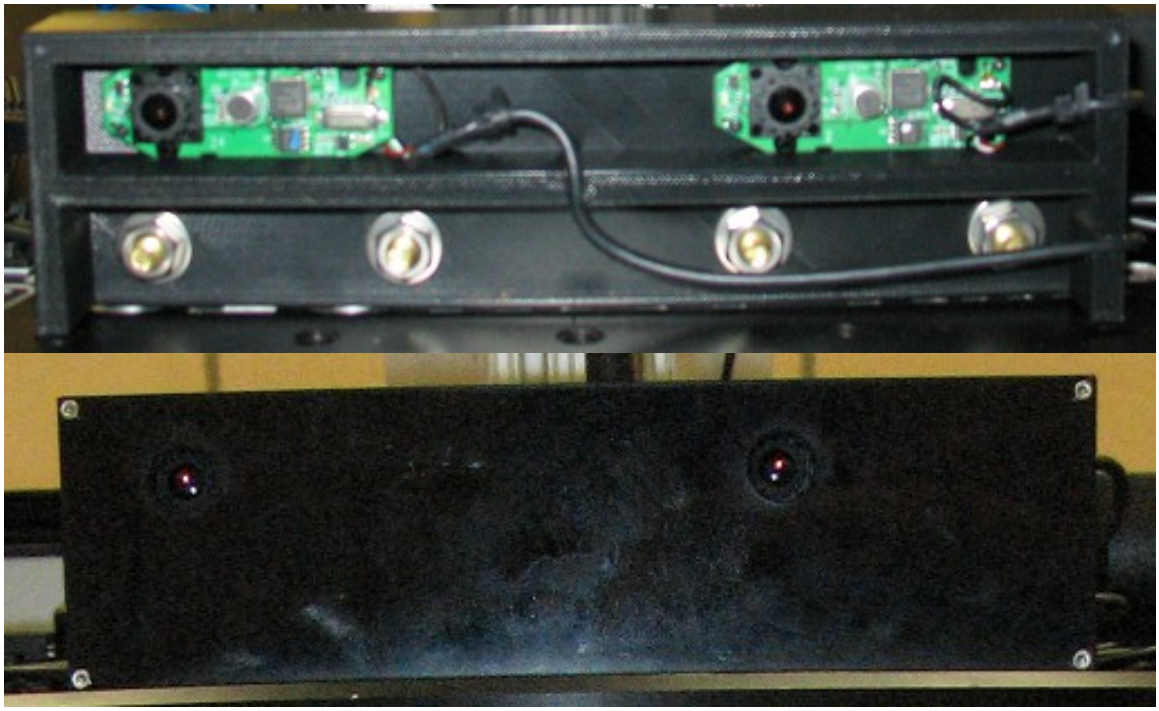


Figure 40: Mounted Stereo Vision System with and without protective cover plate

GPS Sensor

The GPS sensor shown in Figure 41 is designed by USGlobalSat Inc., and contains a built-in GPS Patch Antenna. Furthermore, it is designed with a built-in Serial to USB adapter for communication with several operating systems, including Windows, Mac, and Linux. The GPS provides accuracy to localize position within a minimum of 2 meters of accuracy. Since only a generic GPS sensor package had been developed for ROS, the packaged GPS sensor had to be partially modified and integrated with a new ROS packages designed to handle all the data and information available from this GPS.

Figure 41: ND-100S GPS receiver

Orientation Sensor

The UM6 orientation sensor, designed by CH Robotics, had 3-axis accelerometer, angular rate sensor, and compass sensor modules (see Figure 42). The sensor included an onboard 32-bit ARM processor which fused the data from the individual sensor modules using an Extended Kalman Filter algorithm (Greg Welch), and provided the processed data using serial transmission at a rate of up to 1KHz. The data included Euler angles and quaternions to provide 3D rotation information in reference to a geographical compass heading. With the automatic gyro bias calibration and the cross-axis

misalignment correction, the sensor could be easily calibrated using free open source software to correct for sensor biases, allowing for very accurate orientation data.

Figure 42: CHR-UM6-LT orientation sensor

Sensor Mast

The GPS sensor required placement at the highest point possible on the platform to allow it to receive signals from a maximum number of satellites, and the orientation sensor needed to be mounted away from electronics and metal to avoid magnetic interference; thus, a mast was created to mount the GPS and orientation sensors at the robot's highest point using a thermoplastic 3D printed mounting bracket (see Figure 43). Though the orientation sensor still suffers from some interference in this position, the software provided with the sensor can be used to calibrate the compass module to compensate for any constant ambient magnetic interference in its final position.

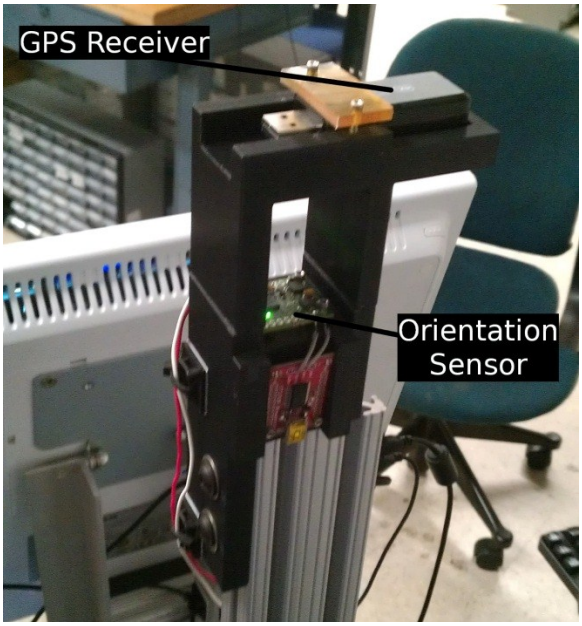


Figure 43: GPS and Orientation Sensor mounted on sensor mast

Software Architecture

The software for both the control computer and the user interface computer was developed within the Robot Operating System (ROS) software framework (Foote). ROS is a graph based software architecture which consists of various nodes, or separately running executable programs, which communicate through special ROS messages and services. Each node runs independently and can send and receive messages, or request and execute services without direct knowledge of the other nodes. Communication between the nodes is facilitated by the ROS master, which links nodes together when needed. ROS allows complex systems consisting of many separate functions to be easily integrated without the need for cumbersome loop structures and state machines, and its functionality has been demonstrated on the famous PR2 Robot from Willow Garage (Willow Garage).

The ROS structure for the GOAT platform consisted of several nodes, some were used “out of the box” from the standard ROS distribution, others were modified versions of those found in the ROS repositories (ROS Browsable Package Repositories), and others were custom built using ROS’s basic development API. The overall structure of the ROS system can be seen in Figure 44.

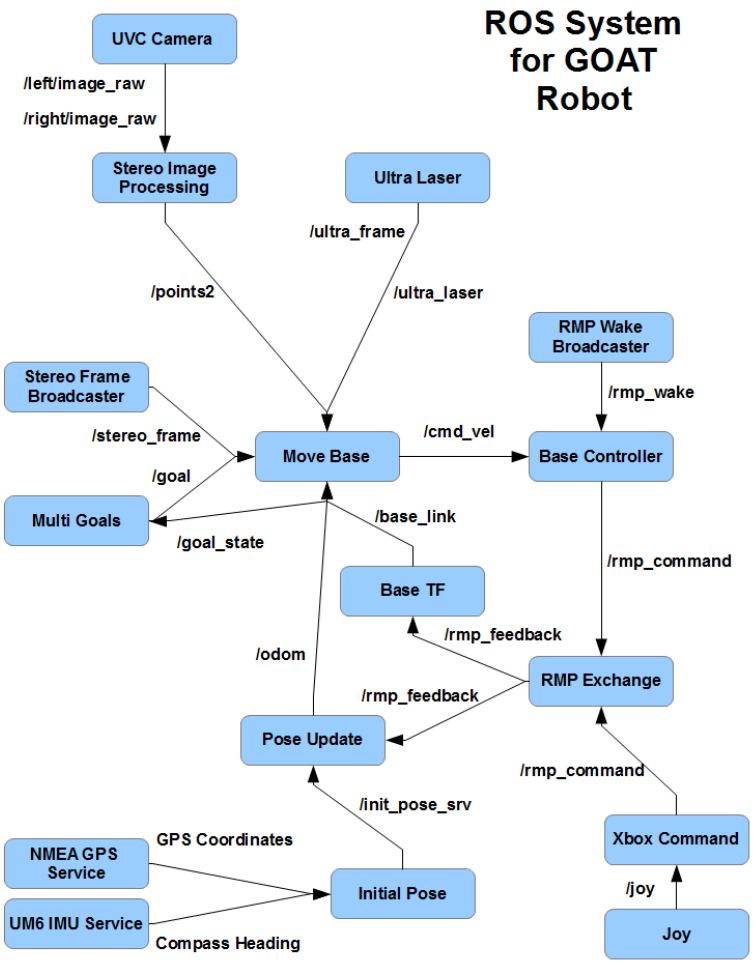


Figure 44: ROS system structure for the GOAT Robot

Sensor Nodes

There were several nodes used for interfacing with sensors and translating their data into ROS messages to be used by other nodes. The nodes for interfacing with sensors are described below.

UVC Camera Node

The UVC camera node is contained in an existing ROS package, called camera_umd, found in the ROS repositories. The node captures image streams from Universal Video Class (UVC) USB camera devices, and publishes them as ROS messages. UVC devices are controlled by parameters that share a common address across all devices in the UVC class. The parameters are addressed over a USB connection and assigned values. For example, the brightness level is a parameter that can be set for most UVC cameras and is accessed by the hex address value of 0x980900. The UVC Camera node as found in the repository was hard coded to set specific parameters used by the authors of the node. Initially, this caused issues since the Logitech webcams used for the stereo vision system did not support all of these parameters. However, the GUVCView program for the Linux operating system allows a user to save UVC device parameters to a configuration file (Assis). This program was used to create a configuration file for the webcams which was accessed with a text editor to determine which parameters the cameras supported. This information was used to edit the source file for the UVC Camera node to eliminate errors created by attempting to set unsupported parameters.

Stereo Image Processing Node

The standard ROS distribution comes with a package called stereo_image_proc. This package utilizes the OpenCV libraries (Bradski) for computer vision processing to process image streams from two cameras for creating disparity maps and point clouds. Disparity maps are “heat” maps where closer objects appear warmer and more distant objects appear cooler. The distance to objects is determined by comparing the images from two cameras, finding similar regions which represent common objects, and determining the position difference, or disparity, between the two regions. Closer objects will have a greater disparity while further objects will have less disparity as they converge to a common point at infinity. Once the disparity map is created, the pixel information from the left camera can be overlaid on the map to create a three dimensional point cloud where each point is a pixel. These clouds of 3D points are useful for a variety of tasks and can be used in much the same way as point readings from a laser scanner, i.e. for mapping an environment and detecting obstacles.

Before the stereo image processing node was utilized on the robot, it was tested with the stereo cameras to determine its functionality. As can be seen from the images below, the two, low-cost webcams performed quite well in creating a disparity map and 3D point cloud for a simple test case.

Figure 45: Stereo camera images for stereo image processing test

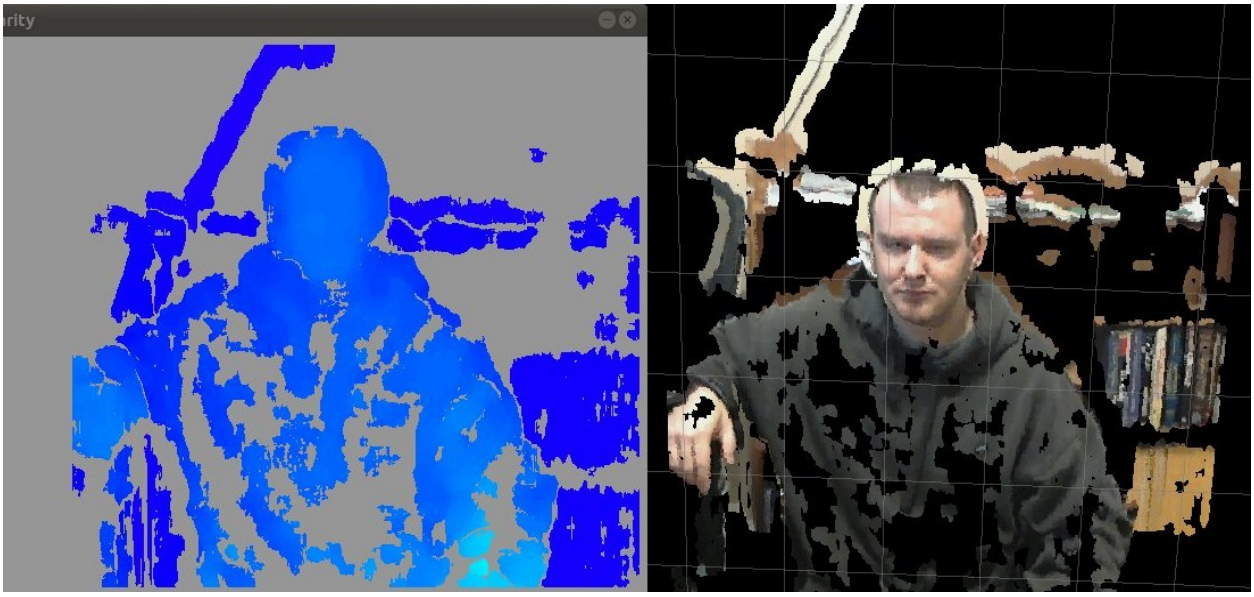


Figure 46: Disparity map (left) and point cloud (right) for stereo image processing test

NMEA GPS Service

Many GPS sensors use a common communications protocol known as the Nation Marine Electronics Association (NMEA) format (National Marine Electronics Association). Since this is a universal communications protocol for these sensors, there exists a ROS node found in the ROS repositories for communicating with GPS devices using this format. However, the existing NMEA GPS node simply runs in a loop which receives and processes the GPS data and publishes it in ROS messages as fast as possible. Since the robot was to use GPS data only for initial pose estimations at major navigation waypoints, this would result in unnecessary processing for the main computer. To alleviate this problem, the node was modified to act as a ROS service that would only be activated upon request, at which point it would take a single GPS reading and return the result.

UM6 IMU Service

There exists a ROS package for the UM6 inertial measurement unit from CH Robotics. This package contains a node which works with the sensor in "broadcast" mode which means the sensor continuously transmits all available data at a rate set in the sensor firmware, which can be no less than approximately 20Hz. Since the GOAT platform was only utilizing a small portion of the data, and since the data was only needed for initial pose estimations at major navigation waypoints, running the node without modification resulted in a significant amount of unnecessary processing for all the unused data. Instead, the existing node was modified to act as a service which returned only the Euler angles and Quaternion values for the sensor's pose estimation upon request.

Ultra Laser Node

The ultra_laser node was used to obtain data from the PIC32 microcontroller representing the readings from the ultrasonic sensors, and converting the readings into ROS laser_scan messages. The conversion was necessary since the ROS navigation stack requires either point cloud or laser scan data for obstacle detection. The node connected to the PIC32 using a serial connection, and data came in the form of timer counts which represented the time required for an echo return pulse from each sensor. The timer count was proportional to the range reading. i.e. a longer count meant a greater range to any detected obstacle. The sensors also waited a set amount of time to return an echo pulse to the PIC32, meaning that if this value was received there was no obstacle detected by the sensor. The data was transmitted in the form of a 21 byte array. The count from each sensor stored as a 2-byte integer, with the high and low byte of each integer transmitted separately. The final byte was a standard ASCII new line feed character '\n' (hex code 0x0A). This was utilized in the ultra_laser python code with the python serial library function "readline" which reads from a serial port until the new line character is received. This allowed the data to be automatically parsed into each 21 byte array. Each array was then converted back into integers using the python struct library function "unpack", and the integers values were converted to range readings, in meters, using a conversion factor. The range readings were used to produce a ROS laser scan message which covered a 360 degree arc, with range readings at every 0.5 degrees. The range reading from each sensor was repeated for the section of the arc it covered around the perimeter of the robot. The arc sections for each sensor are shown in Table 2, with the front sensor centered at zero degrees in the arc.

Table 2: Arc sections for converting ultrasonic sensor readings into laser scan messages

Sensor Number	Start Angle of Arc Section	End Angle of Arc Section
1	346	15
2	16	45
3	46	75
4	76	113
5	114	158
6	459	203
7	204	248
8	249	285
9	286	315
10	316	345

Repeating the range values in this way allowed the range readings from the ultrasonic sensors to be represented by an arc of laser scan points for purposes of obstacle detection.

Pose Estimation

Several ROS nodes were used to produce the necessary transformation frames which represented the various poses of the robot and its sensors. These poses were in reference to a world coordinate frame which could be defined as a set of GPS coordinates. ROS uses tf (transformation frame) messages to produce the necessary transformations frames which represent poses. Each tf message contains X, Y, and Z translation of the frame in meters, and a quaternion to represent the rotation of the frame. The tf messages also define the parent and child frame, allowing each frame to be defined in relation to a previous frame in a "transformation tree". The transformation tree for the GOAT robot is represented in Figure 47.

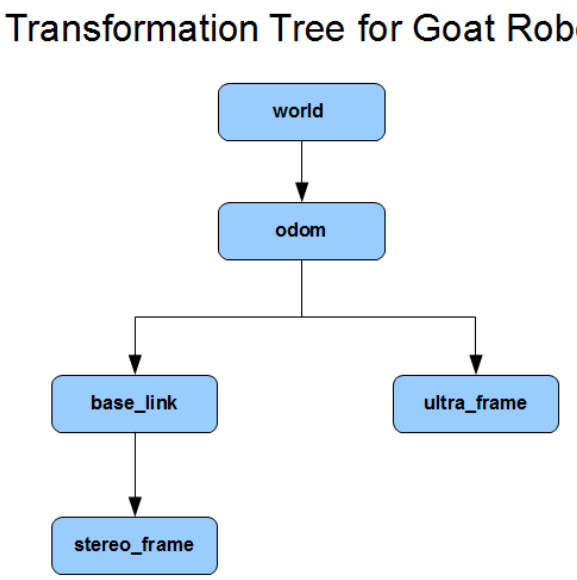


Figure 47: Transformation tree for the GOAT robot

Note that ROS uses right handed coordinate frames, with the positive X-axis forward, positive Y-axis left, and positive Z-axis up. Yaw is the rotation about the Z-axis, pitch is the rotation about the Y-axis, and roll is the rotation about the X-axis.

The odom frame represented the robot's X and Y translations and yaw angle in reference to the world frame, and was located at the center of the robot in the X/Y plane. The X and Y translations of the odom frame were initialized using GPS sensor readings in reference to the geographical latitude and longitude coordinates for the world frame, with the Z translation set to zero. The yaw rotation was initialized using a yaw reading from the compass sensor in reference to zero degrees north (the X-axis of the world frame). The odom frame was updated from the initial position using odometry data from the RMP wheel encoders. The base_link frame was centered on the odom frame but was rotated according to the pitch and roll angles of the robot as derived from the RMP IMU data. The stereo_frame represented the translations for the location of the left stereo camera in relation to the base_link frame. Finally, the ultra_frame represented the Z-axis translation from the odom frame to the center of the

ultrasonic sensor ring for the robot. Pitch and roll measurements were not used for the ultrasonic sensor frame since data from the ultrasonic sensors was converted to a 2D laser scan ROS message.

Init Pose Service

The init_pose node was a ROS service for determining the initial position of the robot at each major navigation waypoint. A request for the node's service would cause it to request the NMEA GPS and UM6 IMU services. The geographical coordinates from the GPS sensor contained in the response from the GPS service were used to calculate the robot's X and Y translations in reference to the geographical coordinates of the world coordinate frame. The compass heading from the UM6 orientation sensor was contained in the UM6 IMU service response and was used to determine the initial heading of the robot.

Pose Update Node

The pose_update ROS node utilized wheel encoder and yaw rate data from the RMP to update the robot's position in reference to the initial pose. The node was activated upon receipt of a feedback message from the RMP. Upon receiving each feedback message, the node would record the system time, as well as the previously recorded system time. The yaw rate of the RMP, as calculated by the RMP Centralized Controller and contained in the RMP feedback message, was used to determine the change in yaw from the previously calculated yaw by the following formula:

$$\boldsymbol{yaw = previous\ yaw + yaw\ rate * (current\ time - previous\ time)}$$

Also recorded upon receipt of an RMP feedback message was the total linear distance traveled as calculated by the RMP Centralized Controller. This value, along with the newly calculated yaw angle, was used to calculate the X and Y translations using the following formulas:

$$\boldsymbol{X = previous\ X + (linear\ distance - previous\ linear\ distance) * \cos(yaw)}$$
$$\boldsymbol{Y = previous\ Y + (linear\ distance - previous\ linear\ distance) * \sin(yaw)}$$

The calculated X and Y positions and the yaw angle were used to produce the odom transformation frame, with the Z translation and the roll and pitch angles set to zero.

Base TF Node

The base_tf ROS node was activated upon receipt of a feedback message from the RMP. The node used the pitch and roll angle of the RMP, as calculated by the RMP IMU and contained in the RMP feedback message, to produce the base_link transformation frame, with the translations and yaw angle set to zero.

Stereo Frame Broadcast Node

The stereo frame broadcast node was used to continuously produce a transformation frame for the stereo cameras at a rate of 20Hz. The stereo_frame represented the X, Y, and Z axis translations of the left stereo camera in relation to the base_link frame. The stereo frame was also rotated -90 degrees around the Z-axis and X-axis since the stereo_image_proc node calculated the positions of the point clouds with the Z-axis facing forward and the Y-axis facing down (see Figure 48).

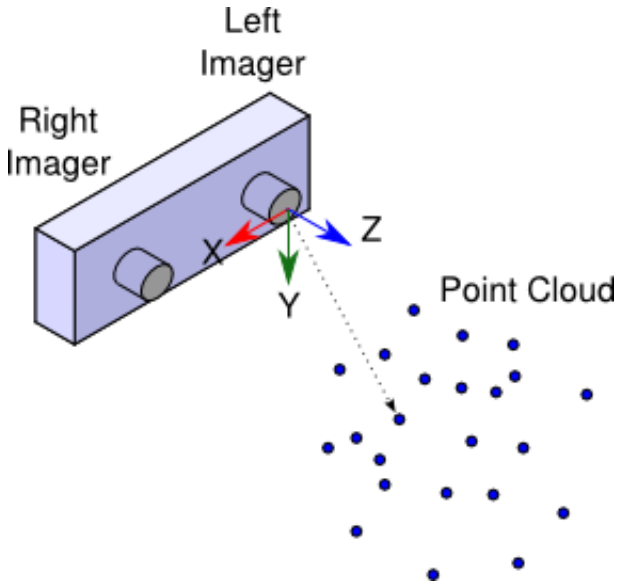


Figure 48: Coordinate frame used by the stereo_image_proc ROS node

RMP Control

The RMP platform contained an internal controller that handled low level control of the platform motors, transmitted feedback data representing sensor readings and system state, and allowed users to set parameters which controlled the platform motion and configuration. The controller could be addressed using Ethernet, CAN, or USB interfaces. Ethernet was chosen as the interface type since ROS allows computers to be networked with Ethernet so any ROS computer on the robot's local network could communicate with the platform. Several ROS nodes were written for interfacing with the RMP and are described below.

RMP Exchange Node

The RMP exchange node was responsible for sending all commands to and receiving all feedback from the RMP Centralized Controller Unit (CCU). The CCU, seen in Figure 49, was responsible for all internal control and monitoring of the RMP, including PID loops for control of the platform motors.

Figure 49: RMP Centralized Controller Unit (CCU)

Commands to be sent to the RMP were received by the exchange node in the form of a custom rmp_command ROS message type. Commands sent to the RMP platform contain three parts: a 16-bit header, two 32-bit variables, and a 16-bit cyclic redundancy check (CRC). The header can be one of two values, either 0x500 for motion commands or 0x501 for configuration commands.

For motion commands, the two 32-bit variables are the normalized velocity and yaw rates respectively, each in the form of an IEEE754 32-bit Floating Point value with a range of allowable values from -1.0 to 1.0. The RMP controller used these normalized values to scale the maximum velocity and yaw rate values, in units of meters per second and radians per second respectively, stored in the RMP controller's configuration. If the RMP exchange node received a command message with this header value, it would perform a check on the velocity and yaw rate values to determine if they were within the allowable ranges. If they were found to be less than -1.0 or greater than 1.0, they were automatically set to -1.0 or 1.0 and the command was sent to the platform.

For configuration commands, the first 32-bit variable was the command ID and the second was the configuration parameter value. The command ID sent to the RMP platform was a 32-bit integer value which identified the configuration parameter to be set. However, the ROS messages contained a string representing the name of the parameter itself. This string was used by the exchange node to lookup the corresponding command ID integer value which was then sent to the RMP. For example, the configuration parameter for setting the maximum turn rate would be identified in the ROS message as the string RMP_CMD_SET_MAXIMUM_TURN_RATE which is the name of the parameter given in the RMP interface documentation. This string was used by the exchange node to lookup the command ID value of 6 which was used by the RMP controller to identify the parameter to be set. This method of setting configuration parameters was used to make the ROS messages more human readable with the names of the configuration parameters rather than their numerical ID's.

Configuration parameter values were of three main types: 32-bit integer, 32-bit floating point, and an IP address string which was converted to a 32-bit integer value. Since ROS messages must define the data type being used, the RMP command ROS messages contained integer, float, and string variables for storing these values. The RMP exchange node would use the command ID to identify which

variable type was to be used and perform the appropriate actions such as converting the IP strings to integers. Also, many configuration values had allowable ranges and the command ID was used to verify that the value given was within this range. If the variable was not within the allowable range the command was not sent to the RMP and an error message was printed using the configuration name to identify the parameter that caused the error.

Once a motion or configuration command was sent to the RMP, the RMP would return a set of feedback values defined by the user via a set of configuration values. Based on the configuration set by the user, the exchange node would expect a certain number of values with each value being a certain type. The values were published as custom rmp_feedback ROS messages.

RMP Configuration Node

Setting of the platform configuration values was accomplished by means of a separate node. This node read the configuration values from a text file with each value identified by its string ID. The configuration node sent a RMP command message to the exchange node for every value to be set, allowing time between each command for the platform to set the value. This allowed all configuration values to be set using a human readable configuration file.

Xbox RMP Node

The xbox_rmp node was used for manual control of the platform. It worked in conjunction with a standard ROS joy node for reading values from a wide variety of video game controllers. The ROS joy node would read analog joystick values in ranges from -1.0 to 1.0, and button values as 0 or 1, and publish the values in a ROS message. The xbox_rmp node was used to receive the messages from the joy node when a wireless Xbox 360 game controller, shown in Figure 50, was plugged into the control computer.

Figure 50: Wireless Xbox 360 controller used for manual platform control

Since the joystick values were already normalized, they were sent directly to the platform as motion commands using the exchange node. One button was used as a deadman switch to prevent motion commands being sent from the joystick if the button was released. Other buttons were used to send configuration commands such as setting the platform into drive, standby, or shutdown mode. The Xbox

node was of great value for testing the platform outside of autonomous mode and was considered a necessary specification for allowing the end user to manually control the platform.

Robot Navigation

Autonomous navigation for the GOAT robot utilized components of the ROS Navigation Stack (Forouher), as well as custom ROS nodes required to interface with the navigation stack. The ROS Navigation Stack is considered stable, and has been demonstrated in use for the PR2 Robot from Willow Garage (Willow Garage).

Move Base Node

The move_base ROS node was included in the standard ROS distribution as part of the ROS navigation stack. The node performed two important tasks for autonomous navigation, producing a cost map using point cloud and laser scan data to represent obstacles in the robot's path, and producing velocity commands which allowed the robot to reach its goals while avoiding the obstacles. These task were performed by two subcomponents of the move_base node.

Cost Map Node

The costmap_2d node received point cloud and laser scan data to create a 2D cost map which represented obstacles in the robot's environment. Points from the point clouds and range readings from the laser scanner were marked as obstacles on the cost map's 2D grid. Each obstacle was also expanded by the radius of the robot to ensure the robot would not intersect with any obstacles while it was navigating. An example of the cost map using point cloud data to mark obstacles can be seen in Figure 51. The map cells marked as obstacles based on the point cloud data are shown in green, while cells marked as occupied by expanding the obstacles by the robot radius are marked in blue.

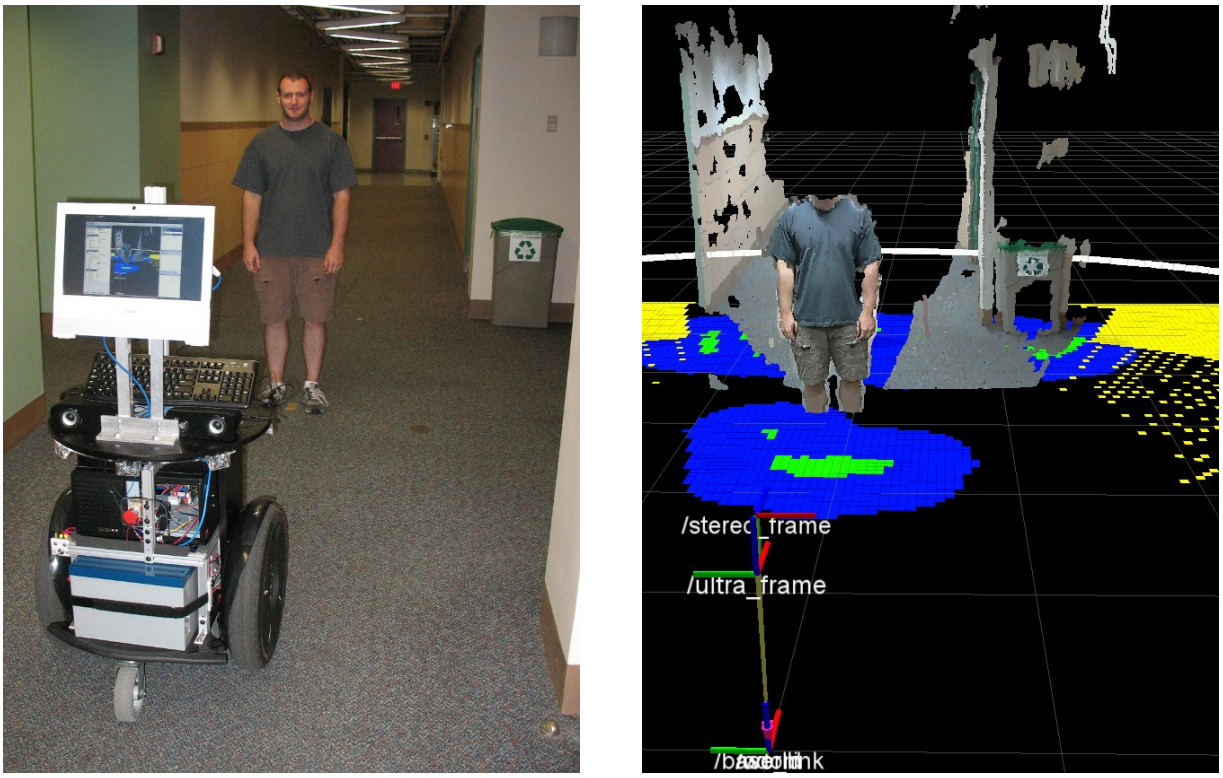


Figure 51: Cost map produced by the costmap_2d node. The point cloud pertaining to the individual at left are collapsed to a 2D cost map with occupied cells shown in green and obstacles expanded by the robot radius in blue

Base Planner Node

The base_planner node received goal messages representing the goal pose of the robot, and utilized the cost map produced by the costmap_2d node along with the transformation frames of the robot to produce velocities designed to move the robot to the goal pose while avoiding obstacles. The velocities were produced as cmd_vel messages which contained the linear and rotational velocities for the robot. The node would also produce goal_state messages with a value of success if the robot reached its goal pose and failure if it did not.

Multi Goal Node

The multi_navigation_goals node was a modified version of the simple_navigation_goals node which was an example node found on the ROS website used for sending goals to the base_planner node. The original node sent a single goal message, which contained X, Y, and yaw values to represent a goal pose for the robot. The original node also received goal state messages from the base_planner node and would print a success or failure message upon receipt. The node was modified to use an array of goal pose values. A loop was added which transmitted each successive goal upon receipt of a goal success message from the base_planner node. This allowed the robot to travel to several goals for testing purposes.

Base Controller Node

The base_controller ROS node served two purposes. First, it received cmd_vel messages from the base_planner node. It translated the rotational and linear velocities contained in the cmd_vel messages

to normalized velocity commands for the RMP. The normalized commands were put in an rmp_command message and published for use by the rmp_exchange node. The second purpose of the base_controller node was to continuously send rmp_command messages to the rmp_exchange node in the absence of cmd_vel messages. This was necessary to constantly obtain feedback from the RMP centralized controller which only provided feedback upon receipt of a command. The base_controller node accomplished this by receiving rmp_wake messages from the rmp_wake_broadcast node, which contained a simple loop to transmit the messages at a rate of 1Hz. These messages were empty and simply served to wake up the base_controller node. Upon receipt of an rmp_wake message, the base_controller node would record the system time, compare this to the time recorded when the previous cmd_vel message was received. If the difference in time was greater than 0.2 seconds, the node would send a CMD_NONE command to the RMP which triggered the RMP Centralized Controller to transmit feedback data to the rmp_exchange node.

Xbox Command Node

The xbox_command node was a simplified version of the xbox_rmp node. The node was not capable of sending velocity commands to the rmp_exchange node, and instead would send command messages to for putting the RMP in standby or drive mode, emergency stop mode, or power down mode. It purpose was to allow for some control of the RMP base while the navigation stack was running.

6. FINAL DESIGN VALIDATION

Upon completion of the various system components, each underwent a design validation to verify their correct operation. Furthermore, the overall system was evaluated at the completion of the project.

Sensor Validation

The sensor validation involved collecting data from the sensors and determining its precision and accuracy.

GPS Sensor

The GPS sensor was tested by logging GPS readings once per second for a period of one hour on the WPI quad. This location was chosen for its open area allowing for greater visibility of GPS satellites for the sensor. The readings were logged into a comma separated values (csv) file and imported into Matlab. The values were imported into a Matlab script with the same algorithms for calculating X and Y offsets as the init_pose ROS node. Before the offsets could be calculated, a reference point had to be determined from the mean position of all the readings, after which each reading was plotted in reference to the mean position. The results are shown in Figure 52.

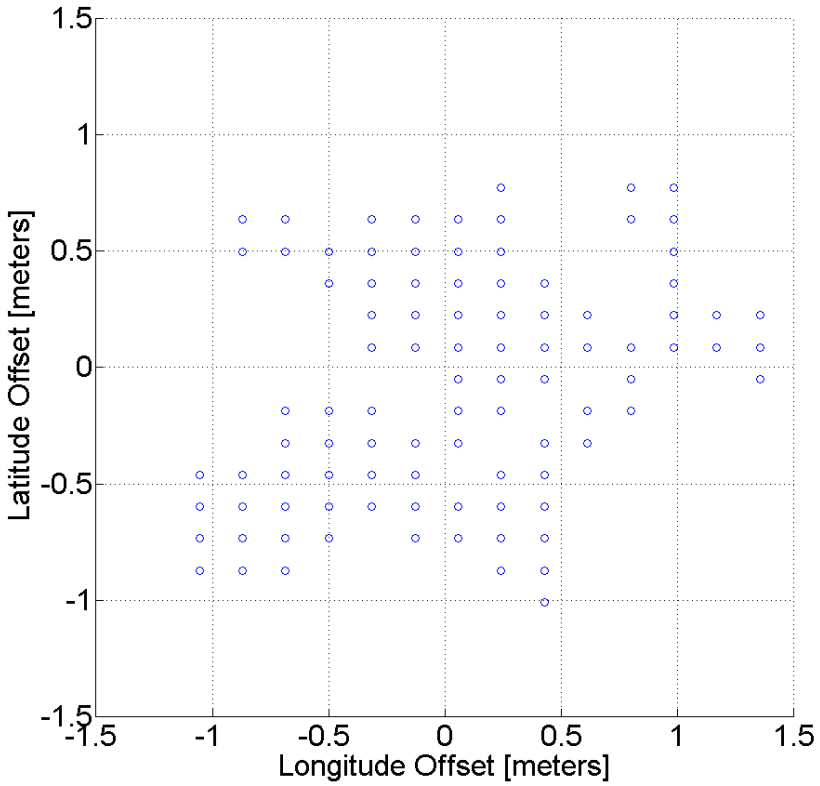


Figure 52: Plot of GPS coordinates taken over 1 hour in reference to mean of all coordinates

The minimum, maximum, and mean linear distance offsets from the average position of the GPS readings are shown in Table 3.

Table 3: GPS data collection results for minimum, maximum, and mean linear distances from mean position

	Minimum	Maximum	Mean
Linear Distance from Average Position	0.0763	1.3754	0.6620

The data shows variation in readings within an average radius of approximately 1.3 meters and a maximum radius of approximately 2.8 meters. These variations can be attributed to drift in the GPS readings as a result of the number of satellites in the GPS sensor's range, and their positions relative to the sensor. In essence, the more satellites, the better the accuracy of the position. Furthermore, the more satellites are positioned toward the horizons, and the fewer that are collinear, the better the horizontal precision of the position. The precision of the horizontal precision is transmitted as part of the NMEA format in a factor called the Horizontal Dilution of Precision or HDOP, with smaller values pertaining to greater precision (Langley). The precision of the horizontal position is of greatest concern to the GOAT Robot since it relies on the GPS data to determine its X and Y position on a 2D horizontal plane. Figure 53 shows a plot of the number of satellites in view and the HDOP during the period of data logging. Note that the numbers vary over time indicating an expected drift in the precision of the readings.

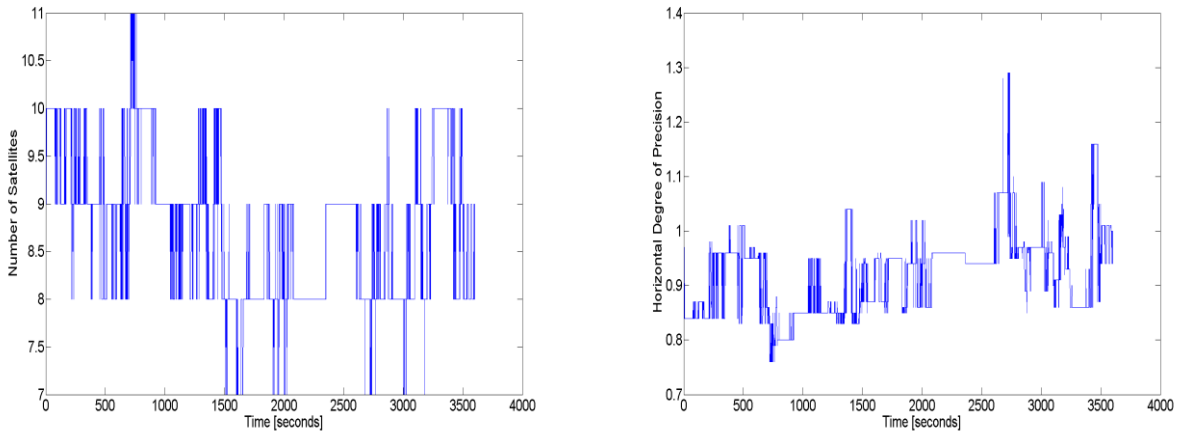


Figure 53: Plots of the number of satellites visible (left) and the HDOP of the GPS position (right) over the data logging period

While reasonably precise for determining an initial estimation of the robot's geographical position, the precision is not sufficient for ensuring the robot only navigates in safe lanes of travel such as sidewalks and crosswalks. The conclusion is that additional means of pose refinement will be required for the robot such as point cloud registration with a preloaded campus map. Alternatively, a method of image processing for recognizing safe lanes of travels using the stereo camera images could be used.

Orientation Sensor

Numerous attempts were made to use the orientation sensor's included software to calibrate the magnetic compass sensors in its position atop the robot's sensor mast. However, the calibration was unsuccessful before an unexpected failure of the sensor. While the reason for the failure of the orientation sensor is unknown, suffice to say it was not possible to validate its data after the failure.

Odometry Data

Despite which method of determining the robot's initial pose was used, the use of odometry data to update the X and Y translations and yaw angle of the robot's pose while in motion was necessary due to the slow rate of GPS updates which occurred at a frequency of 1Hz, and the possible loss of GPS signal and magnetic interference to the orientation sensor's compass sensor. As such, tests were performed to validate the accuracy of the pose_update ROS node which utilized odometry data to update the robot's pose. This was accomplished by manually driving the robot around a circuitous route in the basement hallways of Higgins laboratories using the Xbox controller. The circuit driven by the robot was a rectangle approximately 9m x 16m for a total linear distance of 50 meters and a total yaw angle of approximately 360 degrees. The robot's starting position was marked and the robot was driven around the circuit back to its starting position and orientation, after which the X and Y translation and yaw angle calculated by the pose_update node were recorded. Ideally, since the robot's starting position was set to the world coordinate frame at the start of each test, the three values would be zero if the odometry data was perfect. However, due to tire slippage, slight inaccuracies in the odometry calculations as a result of imperfect wheel sizes, and small errors by the robot driver, some amount of error will inevitably occur. The linear error was calculated from the recorded X and Y values, and the yaw error was taken directly from the yaw measurement with the assumption that the measured yaw should be zero. Finally, the averages of the linear and yaw errors were found, and the ratio of these errors to the assumed linear distance of 50 meters and assumed yaw of 360 degrees was calculated. **Table 4** shows the recorded data with the calculated errors.

Also, using the ROS visualization tool (RVIZ) the robot's coordinate frames were displayed on the user interface computer along with the world coordinate frame. Example screen shots of one test are depicted in Figure 54, with photographs of the robot and screen shots of the RVIZ displayed coordinate frames at the start and finish of the circuit.

Table 4: Measured pose data and calculated errors for five odometry tests

	Test 1	Test 2	Test 3	Test 4	Test 5
X Error [m]	-0.035979	0.848773	-0.639930	0.442384	0.266402
Y Error [m]	0.003637	1.203254	-0.442928	0.371344	0.063651
Linear Error [m]	0.036162	1.472493	0.778264	0.577581	0.273900
Average Linear Error [m]	0.627680				
Average Linear Error Ratio	0.012553				
Yaw Error [°]	-15.0456	-0.0879	-8.7623	-6.7207	-1.5747
Average Yaw Error [°]	-6.4382				
Average Yaw Error Ratio	0.017884				

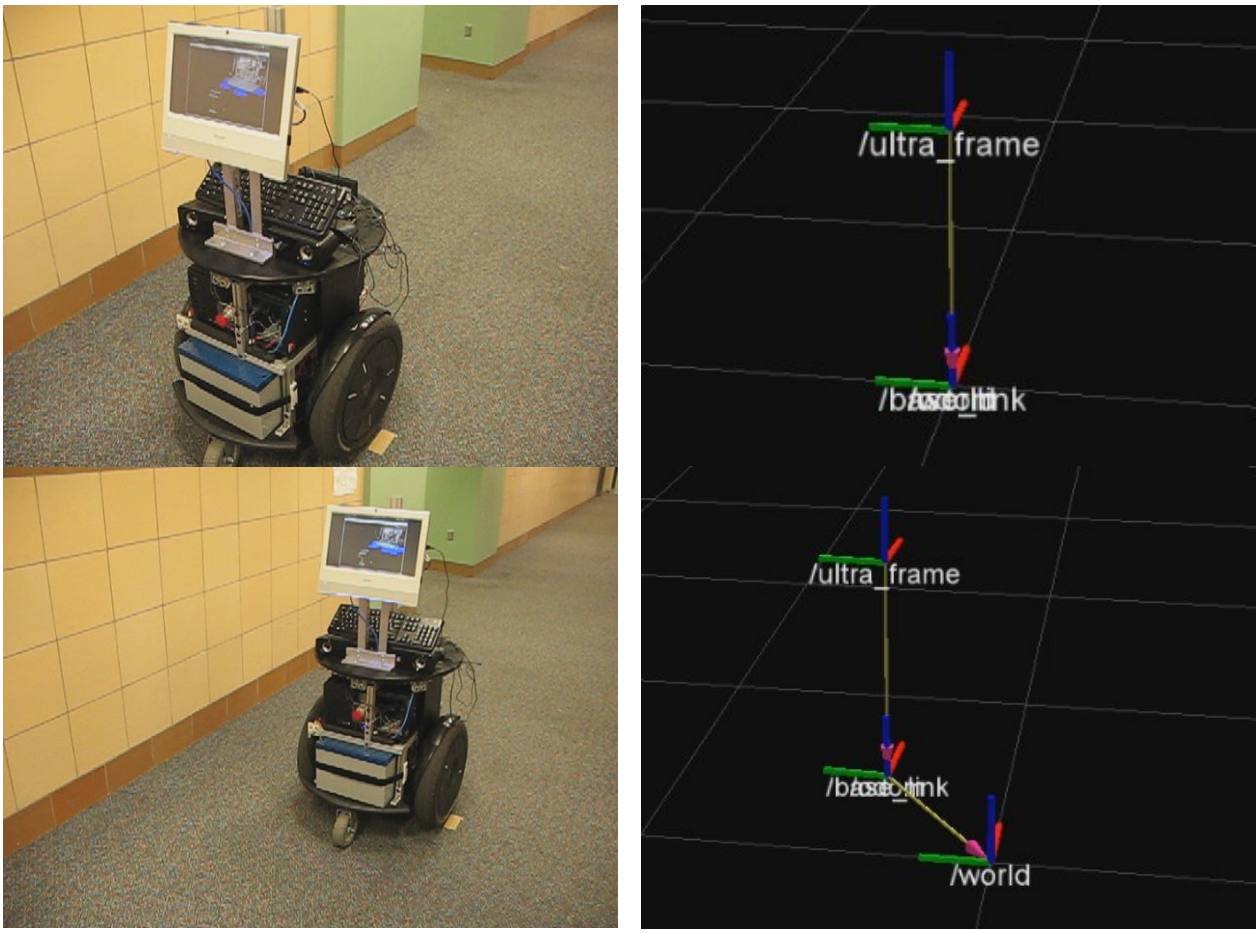


Figure 54: Robot and coordinate frames at start and end of odometry test. The world coordinate frame represents the robot's initial position. The remaining frames are for the robot and show an offset from the world frame at the final position.

The average linear distance error of approximately 1.3% and the average yaw error of approximately 1.8% are quite small which is encouraging for the use of odometry data in updating the pose of the robot while in motion. Though there were several significant yaw errors, this result can possibly be explained by a physical characteristic of the RMP, i.e. it was found that the robot tended to yaw to the left when commanded to drive straight. Most of this tendency was eliminated by adjusting the pressure in the RMP tires in order to increase the radius of the left tire relative to the right tire. However, some amount of drift remained for the test and as a result the robot driver tended to compensate with the Xbox controller by yawing the control stick to the right. It is possible that the consistently negative yaw error is the result of the RMP Centralized Controller measuring slightly more right yaw, or negative yaw, than was the reality. Regardless, the issue with the yaw drift discovered during the test serves to show the importance of properly inflated tires for the robot.

RMP IMU Data

In addition to the X and Y translations and yaw angle from the initial_pose and the pose_update ROS node, the base_tf node utilized pitch and roll angles from the RMP IMU to update the base link coordinate frame for the robot to create a full 3D pose. This was necessary since the stereo camera coordinate frame, which was linked to the base link frame, required a full 3D pose to properly orient the point clouds in relation to the robot's body even as the cameras, and therefore their images, rotated with the pitch and roll of the robot. To verify the base link frame was updated correctly, the robot was first tilted to the side and then to the front resulting in roll and pitch offsets, while the point clouds from the stereo cameras were displayed on the robot's UI computer. Through observation, it was verified that while the cameras and the stereo frame were significantly rotated during the test, the point clouds remained stationary relative to the robot's odometry frame. The tilting of the robot and the displayed coordinate frames and point clouds are shown in Figure 55.

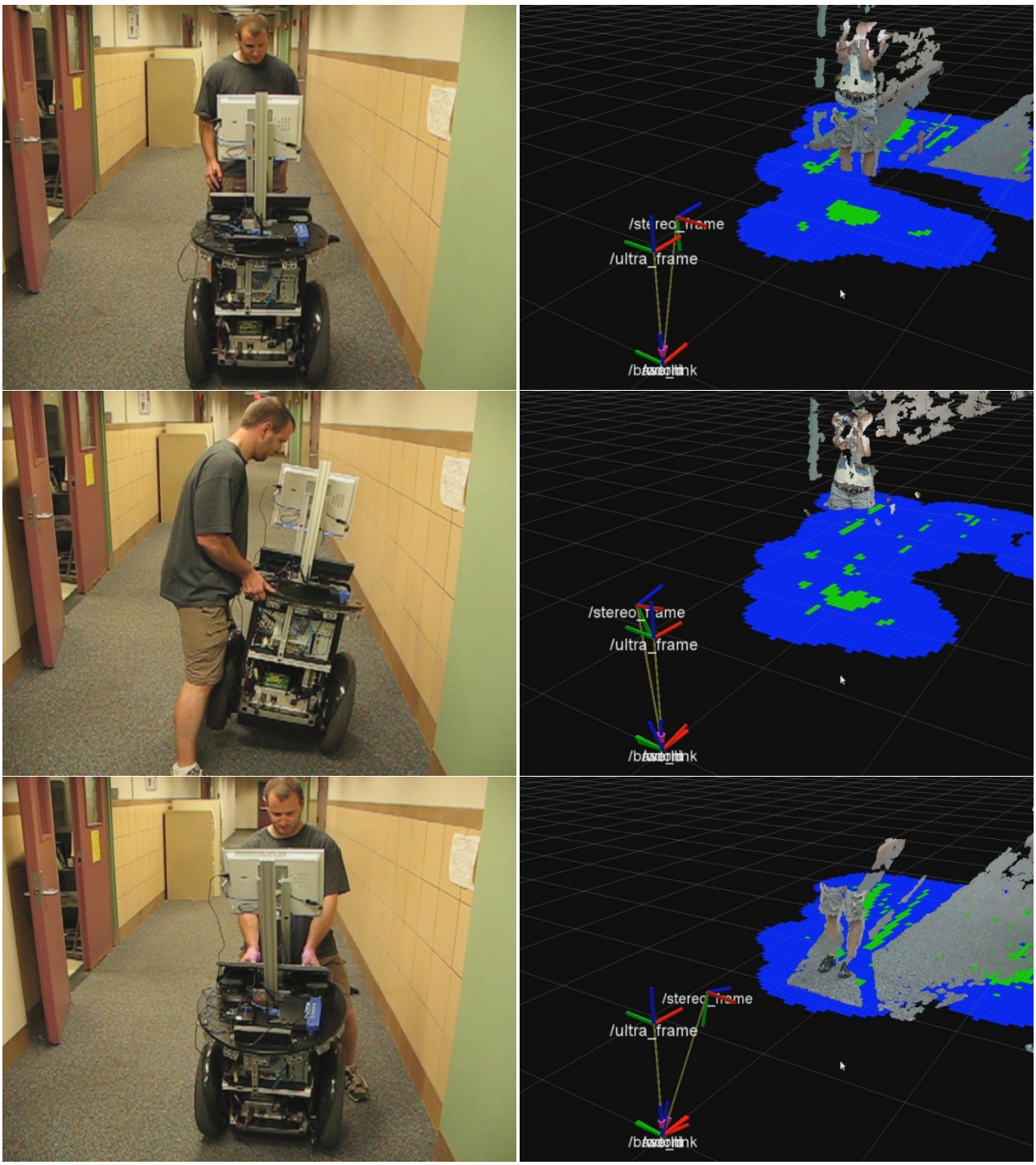


Figure 55: Images showing the robot (left images) with coordinate frames and point clouds (right images). The point clouds remain stationary relative to the robot when the robot is level (top), rolled left (middle), and pitched forward (bottom).

Point Cloud Range Test

In addition to the orientation, the range of objects represented by the point clouds should also be accurate. This is necessary for proper detection of obstacles, navigation, and the eventual possibility of using point cloud data for initial pose refinement. Though RVIZ allows points to be selected in a point cloud and will display their positions, the position information cannot be logged in this way. Furthermore, the points pertaining to an object are not perfect, i.e. their ranges vary, so an average of the ranges should be calculated to find the range of an object which is impractical for manually copying the range data from RVIZ. Since no simple method exists for automatically isolating the points pertaining to an object and logging their positions for analysis, the range accuracy of the point clouds was measured through observation by placing an object at a known distance in front of the robot and observing its location in the point clouds as displayed on the UI computer. The object, a cardboard box, was placed at 2 meters, 5 meters, and 10 meters in front of the robot (positive direction along the robot's X-axis) relative to the robot's base coordinate frame which was located at the center of the robot in the X/Y plane. The X/Y plane was displayed as a grid in RVIZ with each grid cell being 1 meter square. This allowed the apparent range to the cardboard box to be observed by counting the number of grid cells to where the box was located in the point cloud. The results of the observational test are depicted in Figure 56.

The results show that the range measurement for the 2 meter and 5 meter tests are quite accurate. However, the box appears to be located at 9 meters in the point cloud for the 10 meter test, indicating that the range accuracy begins to suffer at longer distances. While this effect is not detrimental to obstacle avoidance and navigation since the ROS navigation stack has been configured on the robot to ignore obstacles beyond 5 meters, the long range accuracy may be an issue if the point clouds beyond 5 meters are to be used for pose refinement by registering them with a preloaded point cloud map.

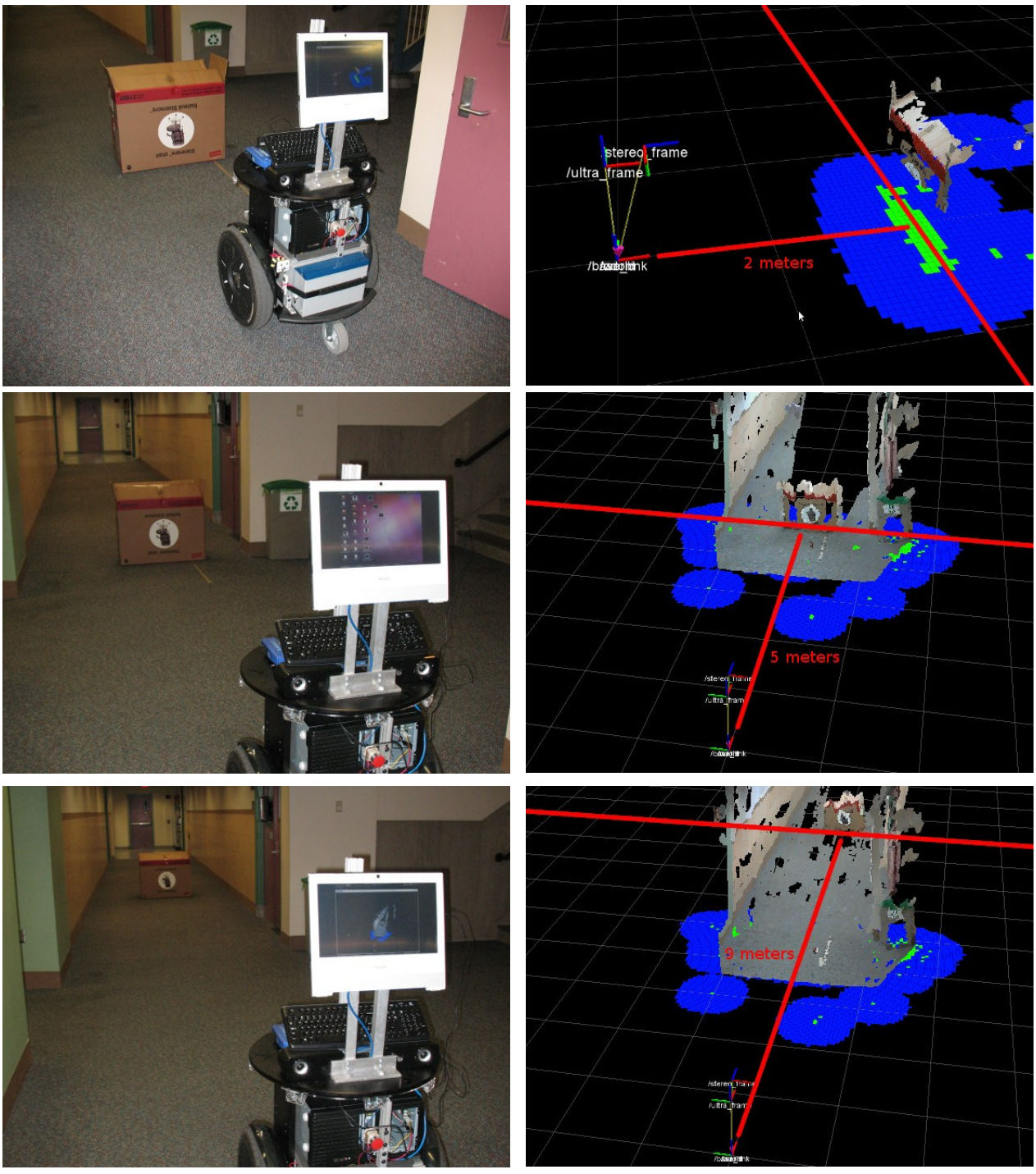


Figure 56: Robot with a box placed in front at 2 meters, 5 meters, and 10 meters (left images, top, middle, bottom respectively). Corresponding point clouds with observed distances for box are show at right.

Ultrasonic Sensors

Validation of the ultrasonic sensor system required obtaining echo return times from the sensors using the PIC32 microcontroller, transmitting the time values to the control computer via a serial connection, and converting the values to ranges for translation into ROS laser scan messages. While readings could be taken and transmitted, and data converted to laser scan messages, an accurate range conversion factor could not be determined. The inability to find a suitable conversion factor was the result of several unexpected problems which could not be resolved due to time constraints.

The first problem involved the breakout board, the design of which was concluded by converting the software files into machine language files, called Gerber Files, which the manufacturing equipment uses to create the board. It is possible that a drill Gerber file was missing when the Gerber files were sent to the manufacturer or a mistake was made by the manufacturer, with the result being missing drill holes connecting the top and bottom layers of the breakout board to the second conducting layer. Unfortunately, all of the signal pins designated for the ultrasonic sensors were connected to the second layer. While it was too late to send another manufacturing order, an alternate solution taken was to solder jumper wires from the designated signal pins of the ultrasonic sensors to unused pins on the top or bottom layers. Once this was accomplished, the PIC32 was programmed to take readings from the ultrasonic sensors.

The next step was to transfer the data received by the ultrasonic sensors to the control computer using a serial transmission routine designed by Kevin Harrington, mentioned in the Acknowledgements section. However, several problems arose when trying to implement the routine, the majority of which circled around the development environment being used, i.e. MPLAB-X, which only works in a Linux environment. Development of the code, including the transmission of data, was completed once the errors associated with learning a new operating system were overcome, and the code was tested using dummy data (see Figure 57).

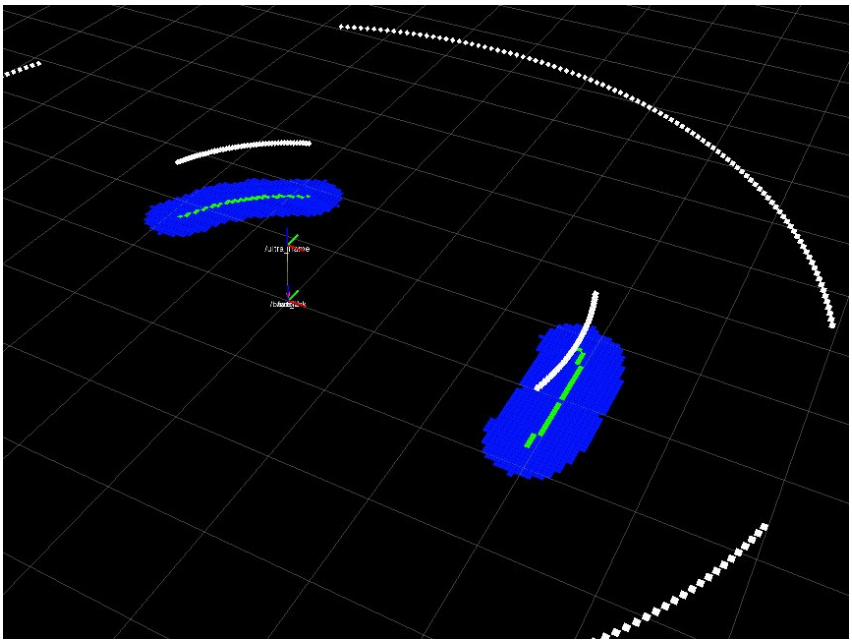


Figure 57: RVIZ display of dummy ultrasonic data converted to ROS laser scan messages. Dummy obstacles appear for the front and left-rear sensors.

Finally, issues arose when using actual ultrasonic sensor readings, in particular the timer counts used to measure the echo return time. The issues involved large, discontinuous jumps in count values for small changes in obstacle ranges. This led to an inability to find a consistent conversion factor of the count numbers into corresponding range values. This process could possibly be solved by revisiting the code of the PIC32 for taking ultrasonic readings. However, due to time constraints, this was not possible, resulting in a missing piece in the completion of the ultrasonic portion of the system.

System Validation

With all the sensor data available, both real and simulated, and the transformation frames in place, the time came to test the autonomous navigation and obstacle avoidance ability of the robot using the ROS navigation stack.

Navigation without Obstacles

The first test performed was commanding the robot to navigate to waypoints in the absence of obstacle data. The tests were implemented by modifying an example ROS node, downloaded from the ROS website, used for sending a single goal to the navigation stack. The code was modified to use an array of successive goals by adding a loop that would execute each time a goal success message was received from the navigation stack. The node was used to send two goals to the robot, one at 3 meters in the X direction (in front of the robot) and a second at 3 meters in the X direction and 2 meters in the Y direction, causing the robot to first drive 3 forward meters, turn left 90 degrees, and drive forward 2 meters. The test was performed using dummy data for the GPS and orientation sensor to initialize the robot's position to the world coordinate frame with a zero yaw angle. In addition, the navigation stack was run with dummy ultrasonic data converted to laser scan messages, with all ranges set outside the maximum obstacle range, a necessary addition since the navigation stack will not run in the absence of laser scan and point cloud data. This simple test verified that the pose data was being interpreted correctly by the navigation stack, and the velocity messages produced by the navigation stack were being properly converted to velocity commands for the RMP.

Navigation with Obstacles

While the robot could navigate to goals in the absence of obstacles, issues arose when obstacle data was included in the form of point clouds from the stereo cameras. The issues were the result of artifacts present in the point clouds caused by imperfect and noisy images from the stereo cameras, which in turn caused the image processing algorithms to improperly locate objects. In essence, clumps of points pertaining to walls or other objects would appear in the foreground which would be interpreted as false obstacles. An example of these false obstacles can be seen in Figure 58.

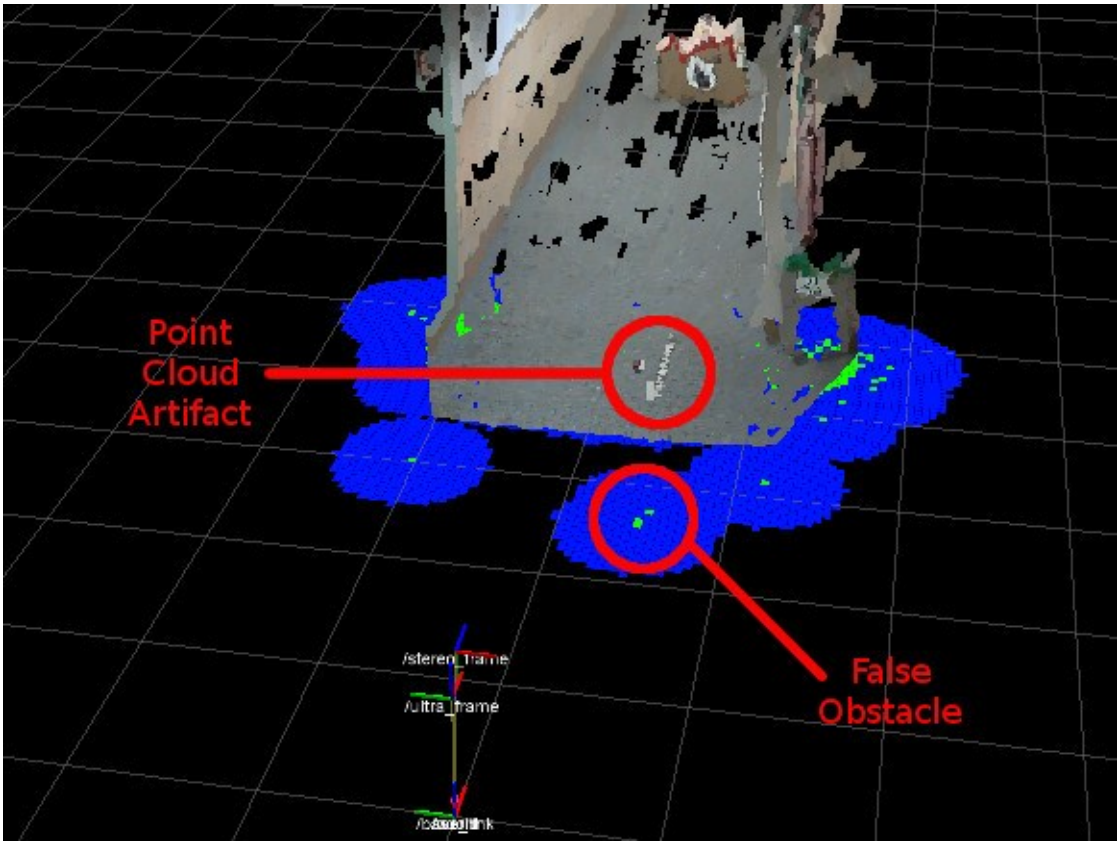


Figure 58: Example of a point cloud artifact and the resulting false obstacle

In an attempt to find a solution to the point cloud artifact problem, the navigation stack was initialized with a 3D voxel grid for storing the point cloud obstacle data. The meant the obstacles were initially represented as occupied cells in vertical columns of a three dimensional occupancy grid. The 3D grid was then collapsed to a 2D cost map for use by the navigation stack. The advantage to this technique is that a minimum number of occupied cells in a column would have to be reached for the corresponding 2D cell to be considered occupied after the column was collapsed. Since the artifacts were generally small, the height of each cell in the columns was set to the average height of the artifacts and the minimum number of occupied cells for the 3D map was set to 3. This meant that even if an artifact spanned two vertical cells in a column, the underlying 2D cell would not be marked as occupied. This technique was successful for eliminating most false obstacles; however, it led to an additional issue. The cost map used ray tracing to clear occupied cells, meaning that if an obstacle was seen, any cells on a direct line between the sensor and the obstacle were considered clear. This property was used in the purely two dimensional configuration to clear old obstacles by ray tracing with the ultrasonic readings. This technique failed for the 3D voxel grid since the ultrasonic readings could only be ray traced through vertical cells at the same height as the ultrasonic coordinate frame. Furthermore, the point clouds were imperfect, i.e. they contained many gaps, so ray tracing often failed for the point clouds even though they could be ray traced in three dimensions. The end result is that using the 3D voxel grid configuration resulted in obstacles persisting even after they were removed, and since the cost map coordinate frame rotated with the robot, a persistent obstacle in the robot's path would appear to remain in its path even as it turned to avoid it.

In summary, both the 3D voxel grid and strictly 2D cost map configurations resulted in false obstacle data which caused erratic behavior from the robot as it attempted to navigate to its goals. One possible solution to the issue is to constantly reset the cost map to clear persistent obstacles. However, this is not advisable since it could cause valid obstacles to be cleared once they are close enough to the robot to become invisible to the stereo cameras and possibly not detected by the ultrasonic sensors. Another possible solution is to use advanced filtering techniques to detect regions of point clouds that are small and only visible for one point cloud frame. While safer, point cloud filtering is a complex problem even when using the existing filtering algorithms provided by the Point Cloud Library. A third possible solution would be to use more precise sensors that are not as prone to noise and spurious readings, such as better stereo cameras or a laser scanner.

Though autonomous navigation with obstacle avoidance was not successful, much was learned about the underlying mechanisms used by the navigation stack and particularly the cost map, information that could be used to improve the system through hardware or software changes, or a combination of both, and complete the autonomous capabilities of the robot.

Remote Operation

A method of controlling the robot remotely was first tested in the basement hallways of Higgins Laboratories. The Xbox controller was connected to a laptop running the ROS joy stick node, and the laptop was wirelessly connected to the robot's network allowing the joystick messages to be received by the xbox_rmp node running on the robot's computer. The ROS image_view node, which allows for the viewing of camera images, was used on the laptop to view images from one of the robot's stereo cameras. This allowed someone using the laptop to view images from the robot's camera and control it using the Xbox controller. Videos of the test were taken from an outside perspective and also from the robot's perspective. Screen shots of both videos can be seen in Figure 59.

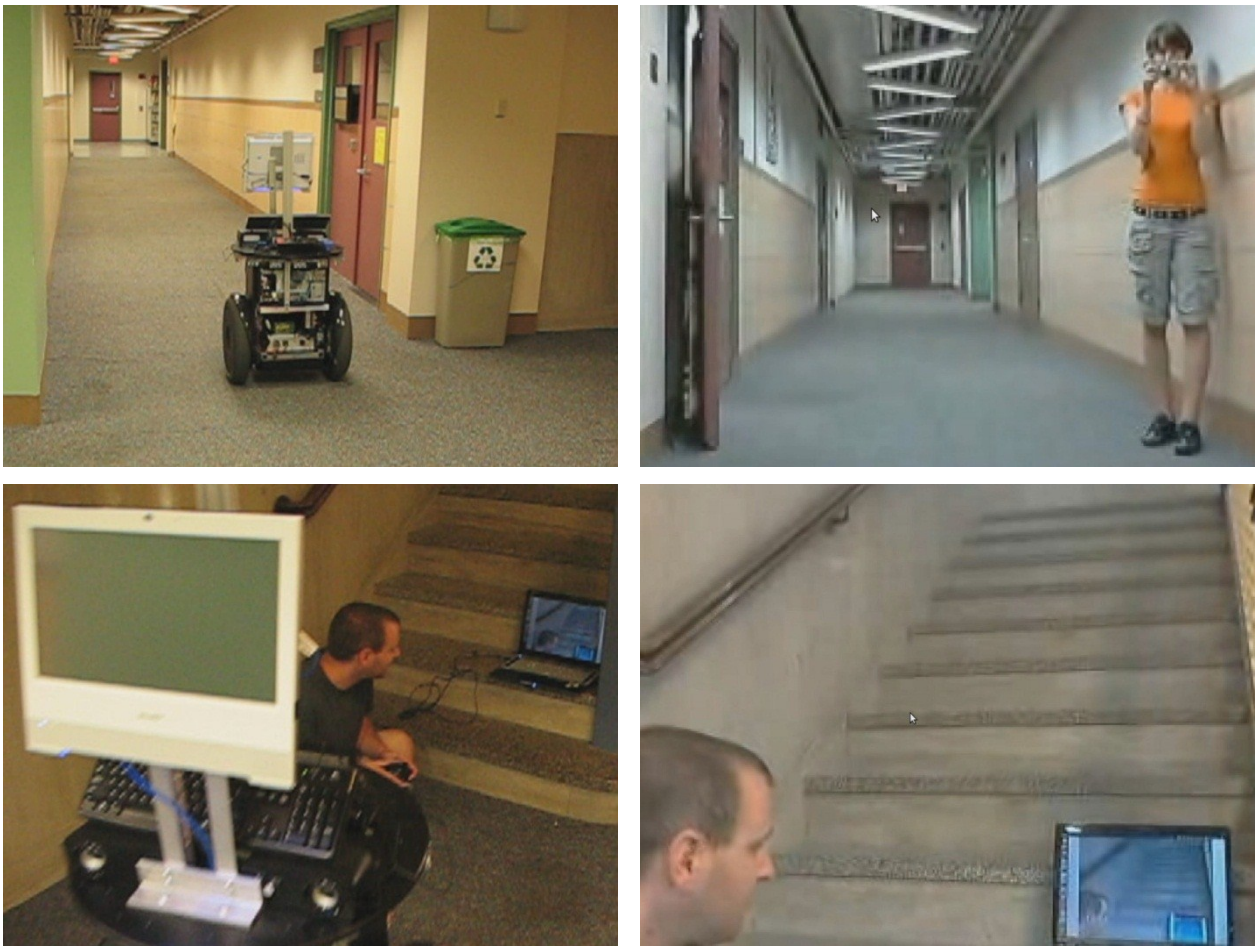


Figure 59: Video stills of remote operation test showing outside perspective (left) and robot perspective (right).

The remote control test was a preliminary step towards implementing telepresence capabilities for the robot, and was used in a feasibility study for robotic tours described in the following section.

Tour Study

As part of an associated IQP project, experimental tours were performed with the robot (LeBlanc). Volunteers were recruited after completing a normal, student-led tour of the WPI campus and participated in a short outdoor tour with the robot. The tour began at Harrington Auditorium, followed a route around the quad, and ended behind the Bartlett Center. The tour route is shown in Figure 60.

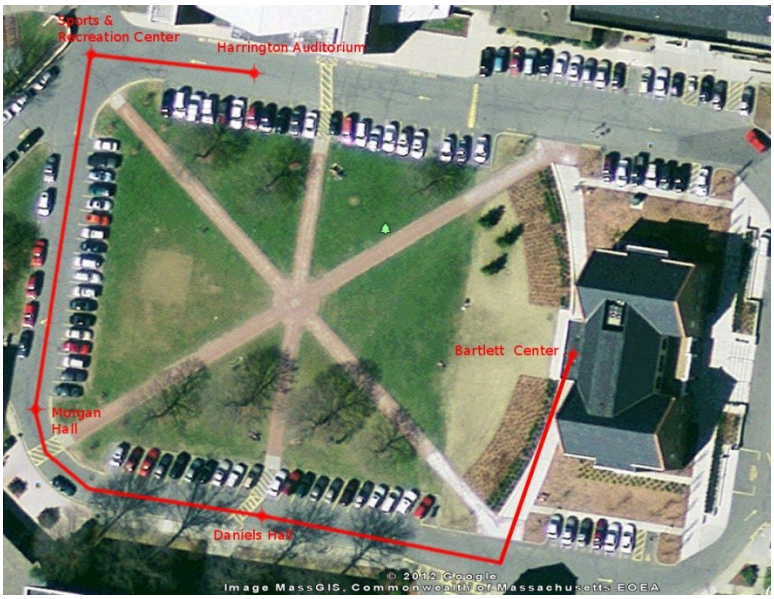


Figure 60: Route of experimental tour shown in red.

To make the robot more identifiable as a tour guide, it was covered in a body and head. The body was constructed from disposable ponchos and the head was a stuffed toy bought at a toy store. The robot can be seen before and after the addition of the body and head in Figure 61 and Figure 62. The head and body served to personify the robot and allow it to use the head to indicate tour locations by appearing to "look" in their direction. At each location, the robot played audio information through its speakers using the goat_sound ROS node.

A remote operator was stationed at the center of the quad and controlled the robot using the remote operation method described in the previous section. The tour volunteers were unaware of the remote operator making the tours a "Wizard of Oz" experiment. This allowed their reactions to what they perceived was an autonomous tour guide robot to be obtained through the use of questionnaires which they filled out after completing the robot guided tour. Overall, the volunteers were overwhelmingly pleased and enthusiastic about their robotic tour experiences leading to the conclusion that robot guided tours are a feasible practice for the WPI campus. Examples of the volunteers participating in the robot guided tour experiment can be seen in Figure 63.

In addition to the feasibility of a tour guide robot, the study found the ability of the robot to use gestures and gaze direction to indicate locations and its intended motions, interaction capabilities including a user interface and dialog with the robot, and the perception of safety around the robot were important factors for the volunteers' satisfaction with the tours. As such, the interactive and gesture capabilities of the robot should be a focus of future development, and the autonomous capability should not only be functional, but as smooth and reliable as possible so as not to cause concern over its safety.

Figure 61: Robot before addition of body

Figure 62: Robot after addition of body

Figure 63: Volunteers participating in the robot guided tour experiment

7. DISCUSSION

Throughout the course of the project, many tasks were accomplished allowing for the development of a platform for research in autonomous navigation and human robot interaction. First, a set of goals for a research platform were outlined and refined in scope to produce a set of specifications. Next, a hardware and software system architecture was designed that would be capable of meeting the specifications, with an RMP platform at its core, and custom components of the system were designed and constructed while others were procured after extensive research of commercially available hardware. When the RMP was obtained, the system was physically and electrically integrated with the platform and the basic functionality of the components was verified. Following this was the development of a software system utilizing existing applications, some of which were kept in their original form while others were modified to fit the needs of the system, as well as the development of new applications, all within the ROS software framework. The functionality of each software and hardware subsystem was tested in a set of validation exercises, each building on the previous until the final test of the overall system's autonomous capabilities, the results of which identified key areas of success to be built upon and areas where further development and design iterations were needed to realize the ultimate vision of the project. Concurrent to the validation exercises, a feasibility study of the system for use as an autonomous tour guide was carried out which produced valuable data to guide future development of the system, and proved its practicality as an autonomous and human-robot interaction research platform. Finally, the hard work of the students involved in all stages of the development and testing was documented in this report. The following is a discussion of what was accomplished, what is yet to be done, what this means for the project as a whole, and finally, suggestions for those who would take on the task of completing the development of a system worthy of future efforts, and putting it to use for the purpose it was intended; advancing the field of robotics and serving as a symbol of achievement for WPI students.

Achievements

Several goals that the team succeeded in accomplishing are listed below, and then described in further detail.

- Design and build mounts and framing for sensors and electronics
- Operation of robot from both plug-in AC and battery DC power
- Operation of the robot on DC power for at least two hours
- Capability of travelling two miles between battery charges
- Capability of travelling at 2 m/sec and accelerating at 1m/sec^2
- Operation in temperatures of 15°C to 35°C
- Capability of reaching all handicap accessible outdoor areas on campus during daylight hours and in the absence of precipitation
- Manual & wireless e-stop system
- Create a breakout board for interfacing sensors & computers with PIC32
- Develop ROS package for interfacing with Centralized Controller based RMP

- Manual control of robot
- Use stereo vision to generate point cloud data
- Use point cloud data to detect obstacles
- Localize using IMU, odometry, compass, and GPS data
- Autonomously navigate to consecutive waypoints in the absence of obstacles

Hardware

The team succeeded in designing and building mounts and framing for the sensors and electronics used in the project. Examples include the additional frames installed on the Segway RMP to hold the control computer, power supply, power terminals, and battery in place; framing for the stereo vision cameras as well as mounts for the ultrasonic sensors, GPS, and orientation sensors.

Operation of the robot from both plug-in AC and battery DC power was achieved, allowing for the robot to be used from multiple power sources. Therefore it became possible to recharge the robot while it is still in use, allowing for development to take place in a lab environment without concern for available power.

Through extensive use of the robot using battery power during testing and the tour study, it was consistently found that the robot could operate continuously using all systems for a period of at least four hours, thereby greatly exceeding the design specification of two hours. Furthermore, the testing verified the robot's ability to operate in temperatures of 15°C to 35°C, in full sunlight and shade, and its ability to reach all handicap accessible outdoor areas. Finally, thanks to the impressive capabilities of the RMP base, the robot was able to far exceed the range specification of 2 miles, the velocity specification of 2 m/sec, and the acceleration specification of $1 m/sec^2$.

An emergency stop system was installed which interfaced directly with the RMP Centralized Controller. This system was critical for ensuring safe testing and operation of the robot by adding an additional layer of safety independent of any systems added to the original RMP platform. The system operated by a switch on the robot body and by a wireless controller making it practical for shutting down the mobile platform in an emergency situation.

A breakout board was designed for interfacing the ultrasonic sensors with PIC32 microcontroller, and in turn sending messages over a serial connection to the control computer. Furthermore, the breakout board allows for easy access to any of the pins of the microcontroller that may be used in the future, connections to easily include an LCD Display or a motor driver, and access to relative GND, 3.3V and 5V levels.

Software

The software developed for the robot consisted of several ROS nodes linked together and interfaced with the existing ROS navigation stack to achieve a system capable of autonomous navigation. The software was capable of integrating sensor data for detecting obstacles and estimating the robot's pose, and was capable of interfacing with the RMP base.

The ROS nodes developed for the RMP allow for easy control, including manual control with generic game controllers, and configuration of the RMP using human readable commands and configuration

files. A great deal of work went into developing the node and making it as generic and user friendly as possible. Since the RMP was a prototype with a new set of interface protocols, the ROS package is the first of its type, and will serve to enhance the ROS repository and stand as a symbol of achievement by WPI students in the eyes of the rapidly growing ROS community.

A great deal of time and effort was also spent on developing the software for obtaining sensor data, processing it, and integrating it into the system for use in obstacle detection and localization. The development included overhauling existing ROS nodes to increase their efficiency and developing new nodes to receive data from the RMP and ultrasonic sensors. New nodes were also developed for processing the data to produce robot poses and obstacle data which could be used by the ROS navigation software. The ROS system was put to the test in a series of validation exercises to confirm the operation of the individual nodes, their compatibility with the ROS navigation software, and functionality of the overall system. The final results show a system capable of detecting obstacles, and autonomous navigation to consecutive waypoints which is a crucial step for a fully autonomous system.

Tasks to be Completed

The completed tasks described above were essential for the creation of a mobile robot platform. However, due to the time constraints and unexpected but inevitable complications, all the desired goals were not accomplished. Notably, the team was unable to accomplish the following:

- Use ultrasonic sensors to detect obstacles
- Autonomously navigate using obstacle avoidance
- User interface capabilities

The team was able to program the ultrasonic sensors to detect close range obstacles, outputting a numerical count value corresponding to distance of a nearby object. Moreover, the results of the ultrasonic sensors were successfully converted into the appropriate form to be able to be transferred serially to the control computer. However, due to unexplained discontinuities in the echo return pulse time counts from the sensors, a suitable conversion factor could not be found for translating the counts into range values. Moreover, the discontinuities make the ultrasonic data unreliable for use in obstacle avoidance by the navigation software. However, tests performed using dummy ultrasonic data served to verify the system of translating the ultrasonic range values into laser scan data useable by the ROS navigation stack for detecting obstacles. Though incomplete, the task of using the ultrasonic sensors for a close range collision detection system is nearly complete, pending the debugging of the sensors to resolve the discontinuity issue.

The use of real obstacle data from the stereo cameras and ultrasonic sensors for obstacles avoidance is considered by the MQP team to be a successful failure, in that the data was unreliable and resulted in false obstacles, but the navigation software interpreted the data correctly and attempted to avoid the obstacles while navigating. This verifies the overall system for autonomous navigation and obstacle avoidance, a system that can be put to real world use once the underlying technical issues of the subsystems have been resolved.

In addition to improvements to the robot's autonomous capabilities, improving the robot's capabilities for human robot interactions is an area which will require significant development. While the robot was shown to be capable of interacting with humans during the tour study, the interactions were limited and performed while under human control. The robot should be able to interact on its own using sound, video, and a user graphical user interface. While the hardware and some software is in place for these capabilities, i.e. the touch screen computer, the speaker system, and a ROS node for playing sound files, most of the underlying software to utilize the hardware for autonomous interaction has yet to be developed.

Conclusions

The end result of the project is a highly capable platform with great potential for use in autonomous and human robot interaction research. The ROS navigation software and the RMP base are tried and true systems, and the systems for integrating sensor data are in place. With the addition of more reliable sensors, the further development of the software to better deal with imperfect data from the existing sensors, and the development of user interface capabilities, the robot will certainly achieve full autonomy including interaction with human users. Furthermore, the success of the robotic tour study has proven the merit of the initial vision for the robot, as an autonomous tour guide and highly visible testament to the students and Robotics Engineering program at WPI. It is the hope of the MQP team that development work continues for the GOAT robot to further validate the hard work that has already been carried out. To that end, we have offered suggestions for future work to serve as a roadmap for continued development.

8. FUTURE WORK

The purpose of this project was to create a robotic platform with the capabilities required for autonomous, mobile navigation. To this end, the project was successful in that the platform is capable of receiving and executing motion commands, sensing its environment, processing the sensor data into obstacle and localization information, and autonomous navigation.

At issue is the reliability of the autonomous capabilities. For the robot to become a truly useful autonomous mobile platform, the problematic pieces of the system must further developed and debugged to reliability of the overall system. This could include upgrading or replacing sensors and improving the software to deal with the inevitably imperfect sensor data. Moreover, autonomous reliability in a test environment is insufficient for ensure safe and reliable operation while interacting with human users. The safety and reliability of the autonomous capabilities can only be verified through extensive testing of the platform under real world conditions.

Furthermore, simply reaching goal poses autonomously will not make the system useful in a practical sense as a platform capable of performing real world duties. The next step toward this goal is the implementation of a path planning algorithm which will utilize known waypoints to produce a series of goal poses that will allow the robot to reach several goals in sequence. This will require not only a ROS node capable of receiving multiple goal locations and finding an optimum path to each, but the careful choosing of all possible goal locations and waypoints between these locations on the campus map. Choosing the goal locations and waypoints will depend on several factors including efficiency in path planning, the ability of the platform to reach the intended targets, safety concerns for an autonomous platform navigating on a school campus, and input from school staff about the desired uses for the robot, i.e. where the robot must go to perform its duties.

The recording of goal locations can be done in many ways, but to be practical there should be a user friendly, graphical interface that allows a user to easily create itineraries for the robot to follow. The school staff that will be using the platform in the future cannot be expected to understand and modify the underlying system. Therefore, they should be able to choose amongst locations on a map and define the order in which the robot will travel to each location, and actions the robot will perform at the locations and while traveling between them. Finally, a user interface should be available to allow some amount of control over and monitoring of the robot's operation. Functions for this interface may include switching between autonomous and manual control modes, initiating autonomous itineraries, monitoring battery and system state, and putting the robot into shutdown or standby mode.

Though some amount of system monitoring should be available through the user interface, a truly autonomous robot should be capable of monitoring its own systems to ensure they are functioning properly, and should take appropriate action when any errors occur or conditions are detected that may compromise the proper functioning of the robot. Some pieces of this system are in place, such as the feedback data from the RMP which includes fault flags and battery state information. However, many other subsystems and functions require monitoring such as sensors and network communications, and the charge state of the gel cell battery. While monitoring all possible sources of faults is an extensive undertaking in itself, perhaps more so is identifying all possible failure modes and the system response to each. A complex system such as the GOAT robot with many interconnected subsystems can fail if any one component fails. When a given component fails, it can create a cascade of failures amongst other

components that rely on it. Therefore, the number possible failure scenarios is immense and cannot be practically predicted and accounted for. What must be accomplished is identifying the key types of failures, such as platform failure, control computer failure, or sensor failures. Once the key types of failures are identified, steps should be devised and programmed into the robot's control systems that will detect these failures bring the robot to a controlled stop using whatever means are available after the failure occurs. Identifying and accounting for key failure modes is lengthy process involving extensive testing but one that is critical for an autonomous, mobile platform that is intended for interaction with humans.

Once the platform is capable of fully autonomous operation and a user interface is in place to allow the end user to operate the platform, the next steps can be taken for extending the robot's capabilities and defining what roles it will fulfill and what duties it will perform on the school campus.

Of primary interest is enabling the robot to act as a campus tour guide. This role was identified early on as one which will provide a significant amount human robot interaction opportunities as well as public exposure for WPI's project program and robotics engineering department. To this end, the robot should also be enhanced with the capability of navigating indoor environments on campus. This ability could include enhanced environment sensors through the use of devices such as a Microsoft Kinect, and a serial manipulator for interacting with the environment.

Indoor capability will open up many other possible roles for the platform. These include but are not limited to:

- Hospital robot for patient assistance, interaction, monitoring, and telepresence for physicians and patient visitors
- Escort and guide robot for building visitors
- Telepresence robot for business and academic settings
- Robot for providing care and assisted for elderly living facilities
- Passive security robot for monitoring indoor spaces

The set of possible uses for a general-purpose, mobile autonomous robotic platform are far too many to list and many have yet to be conceived. For this reason, the platform created for this project should find many possible future uses and will potentially be the focal point of many projects and research studies to come. If even one practical use is found that allows the platform to fulfill its purpose, the project will have been a success and a testament to WPI's Robotics Engineering program.

References

Altium Limited. Altium. 2012. 29 August 2012 <http://www.altium.com/>.

Assis, Paulo. GTK+ UVC Viewer. 2008. 28 August 2012 <http://guvcview.sourceforge.net/>.

Bradski, Gary. OpenCV. 8 May 2012. 29 August 2012 <http://opencv.willowgarage.com/wiki/>.

Bram Bakker, Zoran Zivkovic, Ben Krose. Hierarchical Dynamic Programming for Robot Path Planning. Academic. Amsterdam, The Netherlands: Intelligent Autonomous Systems group, Informatics Institute, University of Amsterdam, n.d.

CH Robotics. "UM6-LT Orientation Sensor." 28 August 2012. CH Robotics. 29 August 2012 <http://www.chrobotics.com/docs/UM6_datasheet.pdf>.

Chiang, Kuo-Hung, et al. "Multisensor-based Outdoor Tour Guide Robot NTU-1." SICE Annual Conference. Tokyo, Japan: SICE, 2008.

"Department of Computer Science and Engineering, HKUST." 28 August 2012. The Department of Computer Science and Engineering. 29 August 2012 <http://www.cse.ust.hk/faculty/golin/COMP271Sp03/Notes/MyL15.pdf>.

Dieter Fox, Wolfram Burgard, Sebastian Thrun. "Markov Localization for Mobile Robots in Dynamic Environments." Journal of Artificial Intelligence Research (1999).

Donald Schelle, Jorge Castorena. "Buck-Converter Design Demystified." Power Electronics Technology June 2006: 46-53.

Foote, Tully. ROS.org. 27 January 2012. 28 August 2012 <http://www.ros.org/wiki/>.

Forouher, Dariush. navigation. 6 July 2012. 28 August 2012 <http://ros.org/wiki/navigation>.

Google. Google Earth. 2012. 29 August 2012 <http://www.google.com/earth/index.html>.

—. Speech Input API. 2012. 29 August 2012 <http://developer.chrome.com/trunk/extensions/experimental.speechInput.html>.

Greg Welch, Gary Bishop. An Introduction to the Kalman Filter. Academic. Chapel Hill, NC: Department of Computer Science, University of North Carolina at Chapel Hill, 1997.

Hähnel, D., D. Schulz and W. Burgard. "Mobile Robot Mapping in Populated Environments." Advanced Robotics (2003): 579-598.

Hector J. Levesque, Raymond Reiter, Yves Lesperance, Fangzhen Lin, Richard B. Scherl. "GOLOG: A Logic Programming Language for Dynamic Domains." The Journal of Logic Programming (1994).

Intel. Intel Core 2 Duo Processor. n.d. 29 August 2012 <http://www.intel.com/products/processor/core2duo/index.htm>.

Intersil. 2012. Datasheet Catalog.com. 29 August 2012 <http://www.datasheetcatalog.com/datasheets_pdf/I/C/L/7/ICL7665S.shtml>.

Joachim Buhmann, Wolfram Burgard, Armin B. Cremers, Dieter Fox, Thomas Hofmann, Frank E. Schneider, Jiannis Strikos, Sebastian Thrun. The Mobile Robot RHINO. Academic. Bonn, Germany: University of Bonn, 1995.

Khronos. COLLADA Digital Asset & FX Exchange Schema. 27 January 2011. 29 August 2012 <https://collada.org/mediawiki/index.php/COLLADA_-_Digital_Asset_and_FX_Exchange_Schema>.

Langley, Richard B. "Dilution of Precision." GPS World (1999): 57-58.

LeBlanc, Noah. Tour Guide Robot Interactions. Interactive Qualifying Project. Worcester, MA: Worcester Polytechnic Institute, 2012.

Logitech. Webcam C260. 2012. 29 August 2012 <http://www.logitech.com/en-us/support/webcams/7075>.

Maria Isabel Ribeiro, Pedro Lima. Markov Localization. Academic. Lisboa, Portugal: Instituto Superior Technico, 2002.

Microchip Technology Inc. 32-bit PIC Microcontrollers. 2012. 29 August 2012 <http://www.microchip.com/pagehandler/en-us/family/32bit/>.

Microsoft Inc. Kinect for Windows. 2012. 29 August 2012 <http://www.microsoft.com/en-us/kinectforwindows/develop/>.

Microsoft. Kinect - Xbox.com. 2012. 6 May 2012 <http://www.xbox.com/en-US/kinect>.

Natalia Hernandez-Gardiol, Sridhar Mahadevan. Hierarchical Memory-Based Reinforcement Learning. Academic. East Lansing, MI: Department of Computer Science, Michigan State University, n.d.

National Marine Electronics Association. National Marine Electronics Association. 2012. 29 August 2012 <http://www.nmea.org/>.

Open Perception Foundation. PCL. 2012. 29 August 2012 <http://pointclouds.org/>.

Parallax. PING))) Ultrasonic Distance Sensor. 2012. 29 August 2012 <http://www.parallax.com/tabid/768/ProductID/92/Default.aspx>.

Pololu. MC33926 Motor Driver Carrier. 2012. 29 August 2012 <http://www.pololu.com/catalog/product/1212>.

R. Huitl, G. Schroth, S. Hilsenbeck, F. Schweiger, E. Steinbach. TUMindoor: AN EXTENSIVE IMAGE AND POINT CLOUD DATASET FOR VISUAL INDOOR LOCALIZATION AND MAPPING. Academic. Munchen, Germany: Institute for Media Technology, Technische Universitat Munchen, Germany, n.d.

RIKEN, BMC. RI-MAN. n.d. 2012 <http://rtc.nagoya.riken.jp/RI-MAN/index_us.html>.

ROS Browsable Package Repositories. 2012. 28 August 2012 <http://www.ros.org/browse/repo_list.php>.

Salton, Jeff. "RIBA the friendly robot nurse." 3 September 2009. gizmag. 2012 <http://www.gizmag.com/riba-robot-nurse/12693/>.

Segway Inc. Segway Robotics. 2012. 29 August 2012 <http://rmp.segway.com/>.

Starck, Jason. All About Circuits. June 2000. 29 August 2012 <http://www.allaboutcircuits.com/vol_2/chpt_8/2.html>.

TechnoGadget. Trend Hunter. 11 October 2008. 2012 <http://www.trendhunter.com/trends/mechadroid-type-c3-not-your-ordinary-receptionist-robot>.

Teruaki Ando, Atsushi Araki, Masayoshi Kanoh, Yutaro Tomoto, Tsuyoshi Nakamura. "Relationship Between Mechadroid Type C3 and Human Beings." Journal of Advanced Computational Intelligence and Intelligent Informatics (2010): 869-867.

The MathWorks Inc. Matlab The Language of Technical Computing. 2012. 29 August 2012 <http://www.mathworks.com/products/matlab/>.

Thomas Collins, J.J. Collins, Conor Ryan. "Occupancy Grid Mapping: An Empirical Evaluation." Proceedings of the 15th Mediterranean Conference on Control & Automation. Athens, Greece: Computer Science and Information Systems Department, University of Limerick, Ireland, 2007.

Thrun, S., et al. "Experiences with two Deployed Interactive Tour-Guide Robots." International Conference on Robotics and Automation (ICRA '99). Pittsburgh, PA: Carnegie Mellon University Press, 1999.

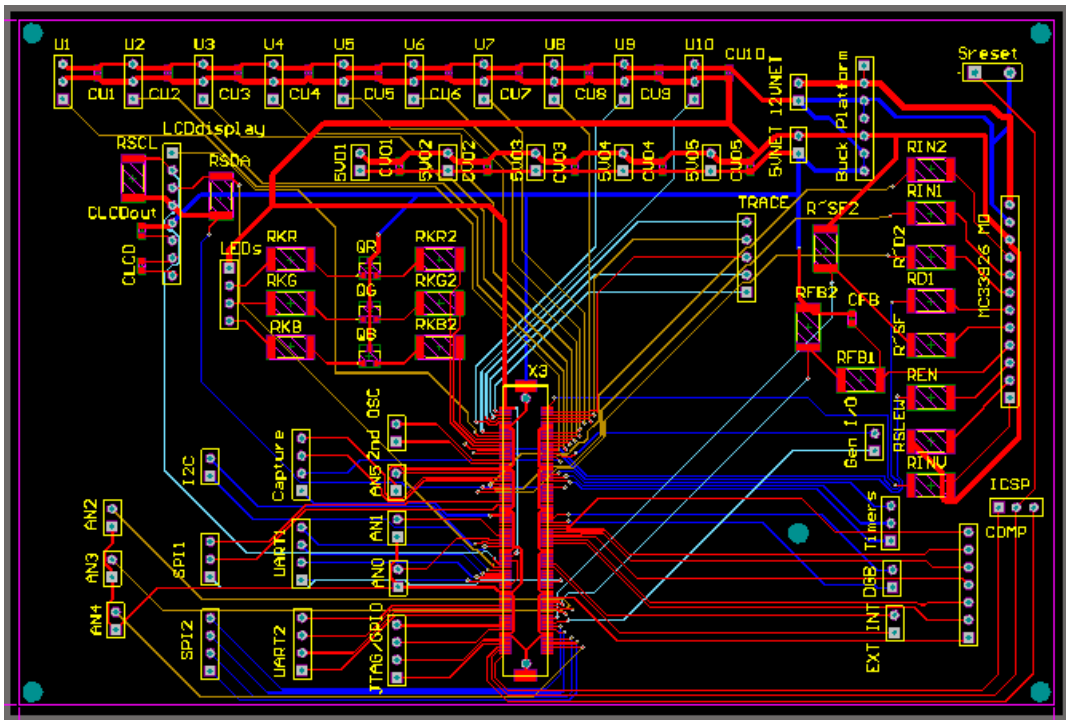
—. "MINERVA: A second generation mobile tour-guide robot." Proceedings of ICRA-99 (1999).

Trimble Navigation Limited. SketchUp. 2012. 29 August 2012 <http://www.sketchup.com/intl/en/index.html>.

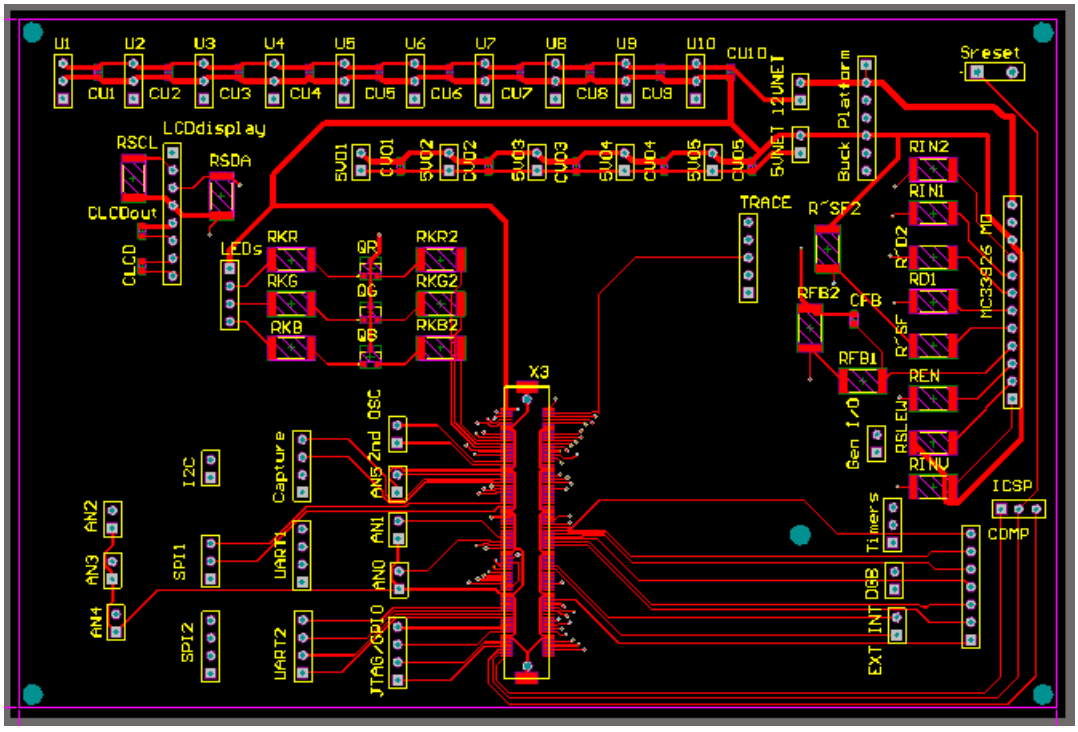
USGlobalSat Inc. "ND-100S." 2012. USGlobalSat Incorporated. 29 August 2012 <http://www.usglobalsat.com/store/download/590/nd100s_ds_ug.pdf>.

Willow Garage. PR2 Overview. 2011. 29 August 2012 <http://www.willowgarage.com/pages/pr2/overview>.

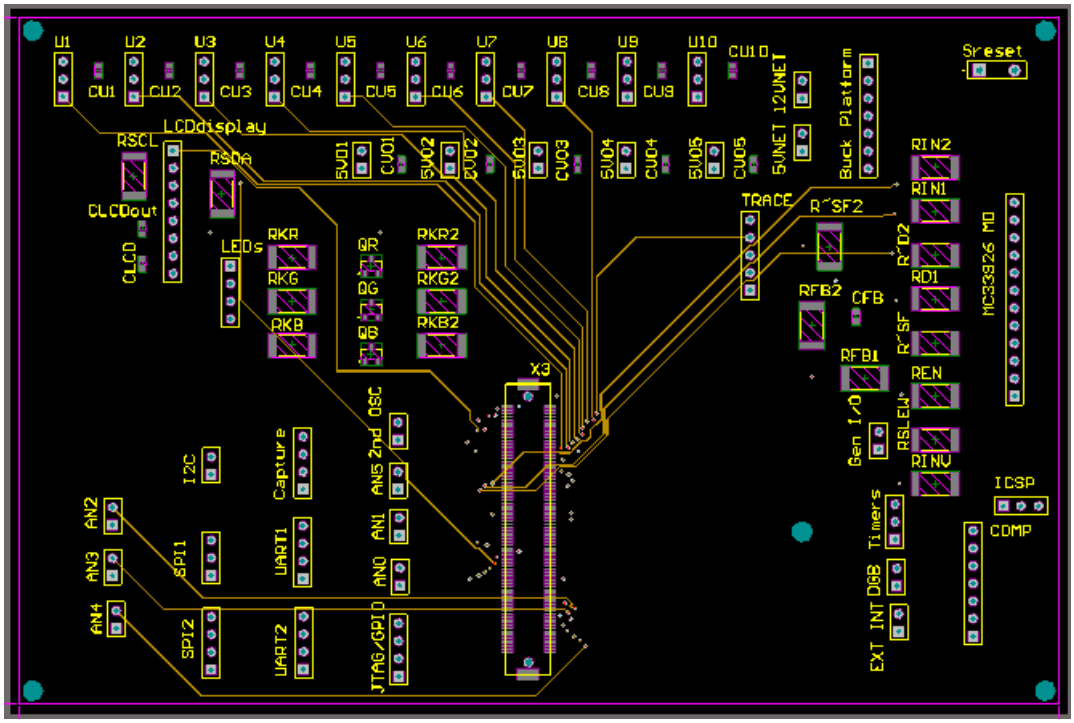
Appendix A: Breakout Board



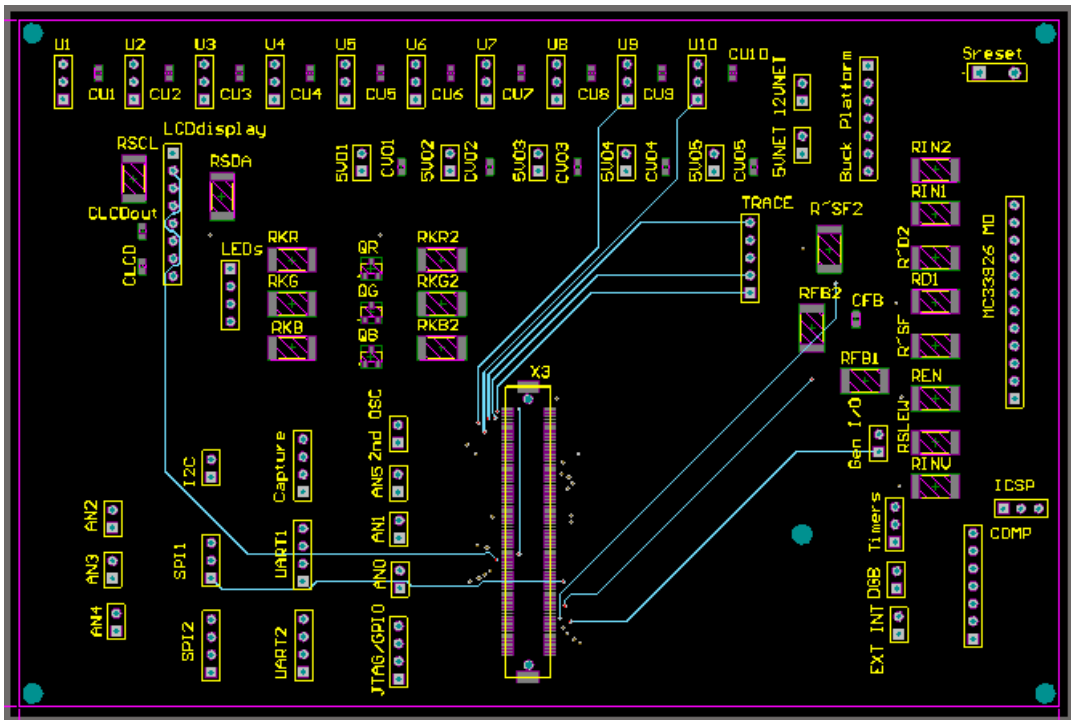
Final 4-layer 4x6" PCB Design Manually Routed



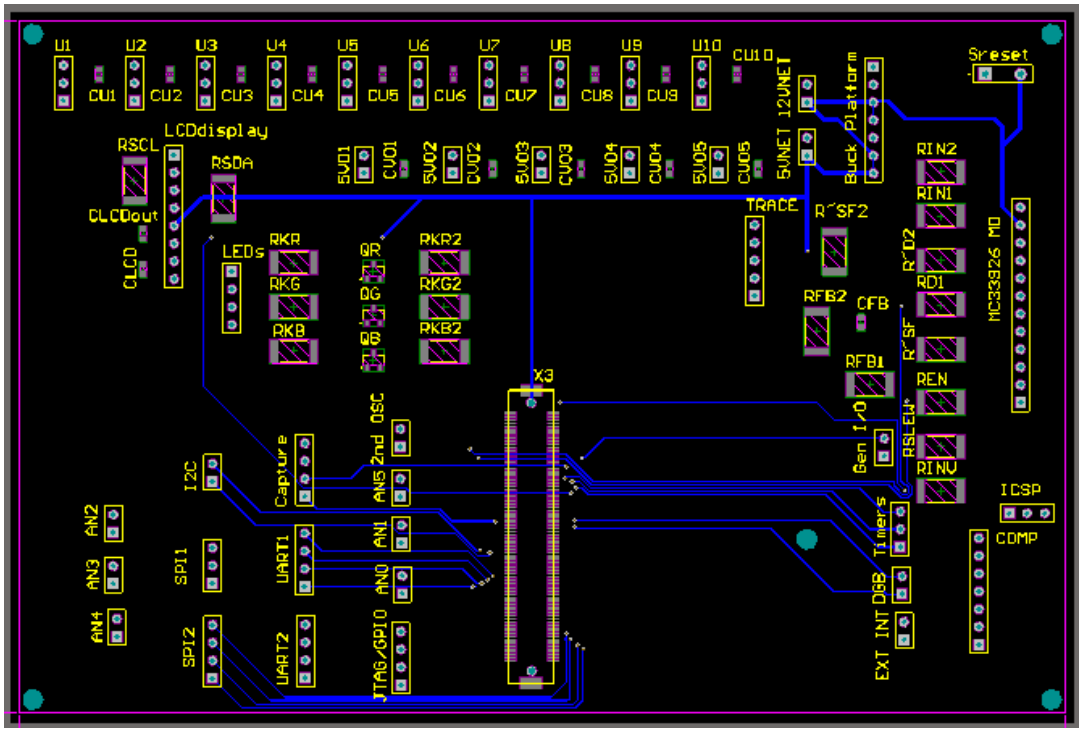
Final 4-layer 4x6" PCB Design Top Layer



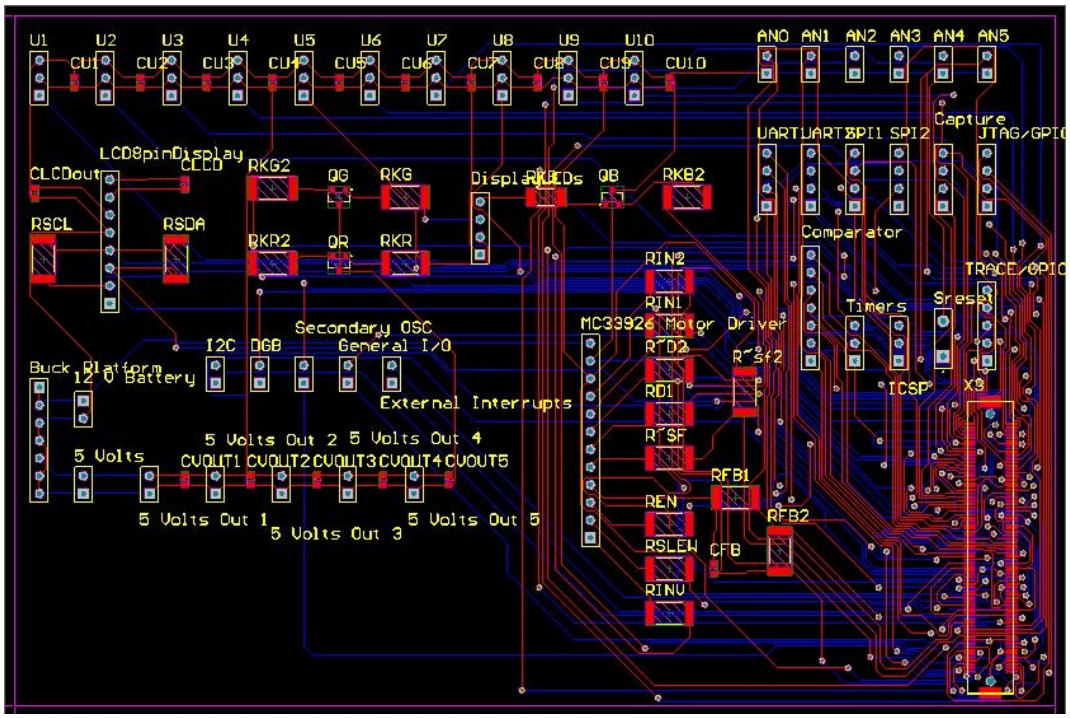
Final 4-layer 4x6" PCB Design Mid- Layer 1



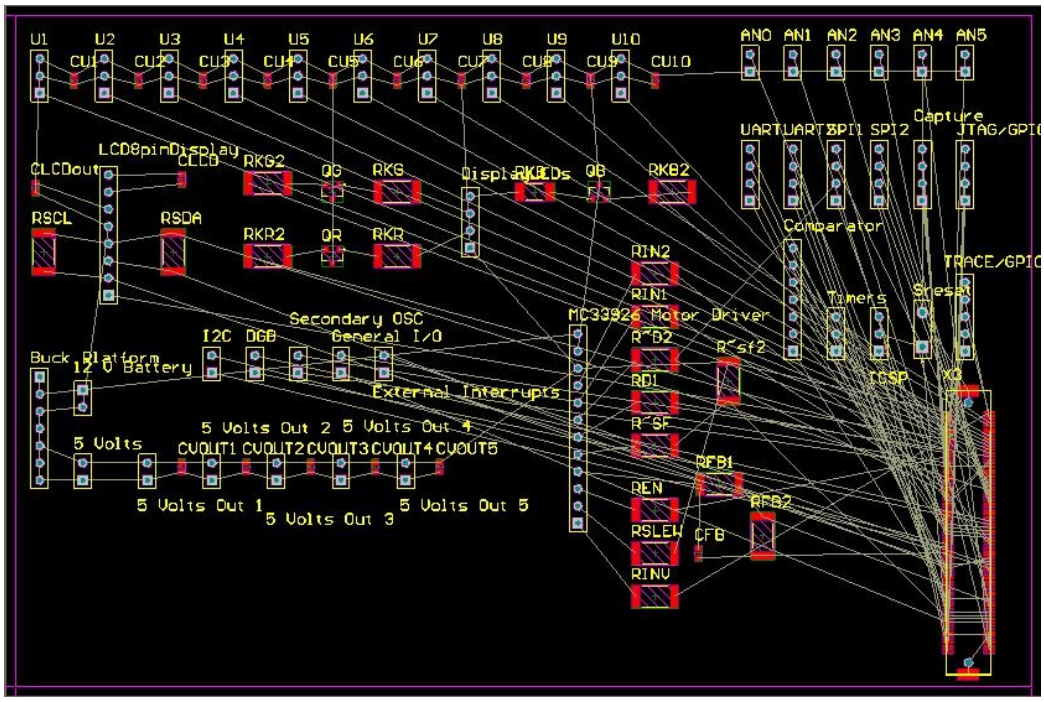
Final 4-layer 4x6" PCB Design Mid- Layer 2



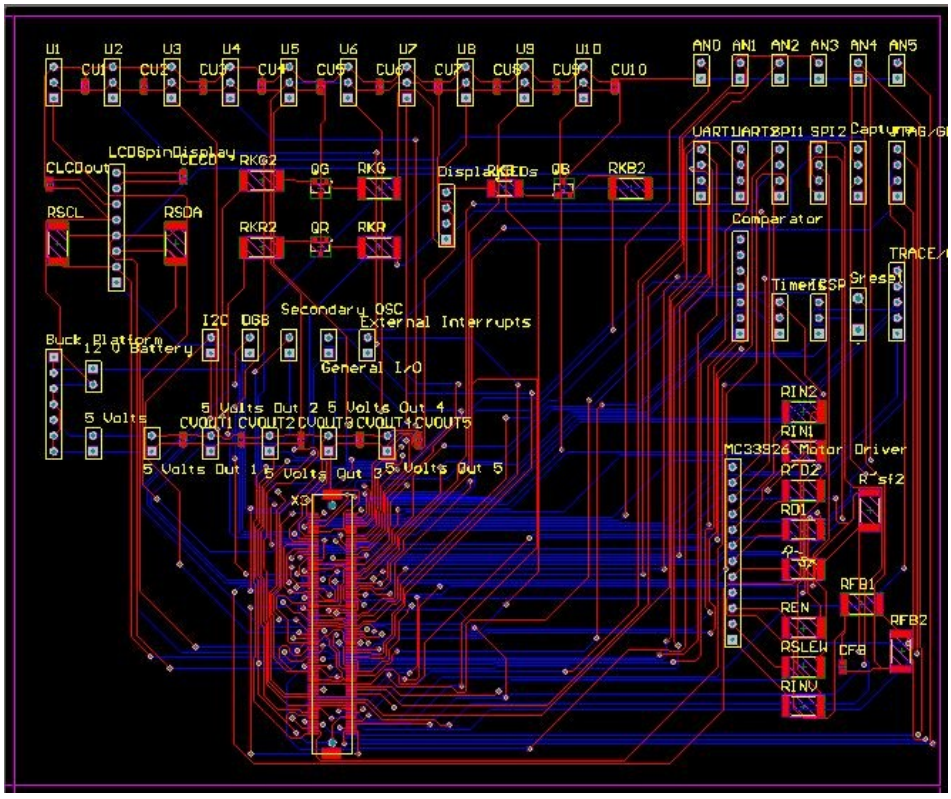
Final 4-layer 4x6" PCB Design Bottom Layer



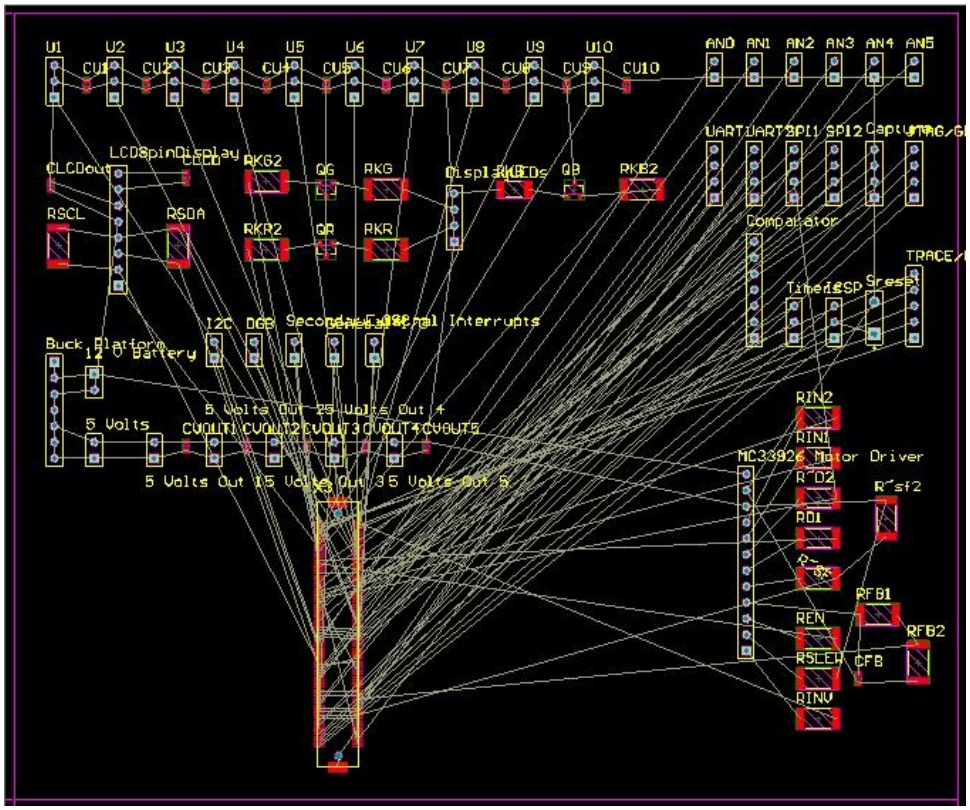
Discarded 2-layer 4x6" PCB Design AutoRoute



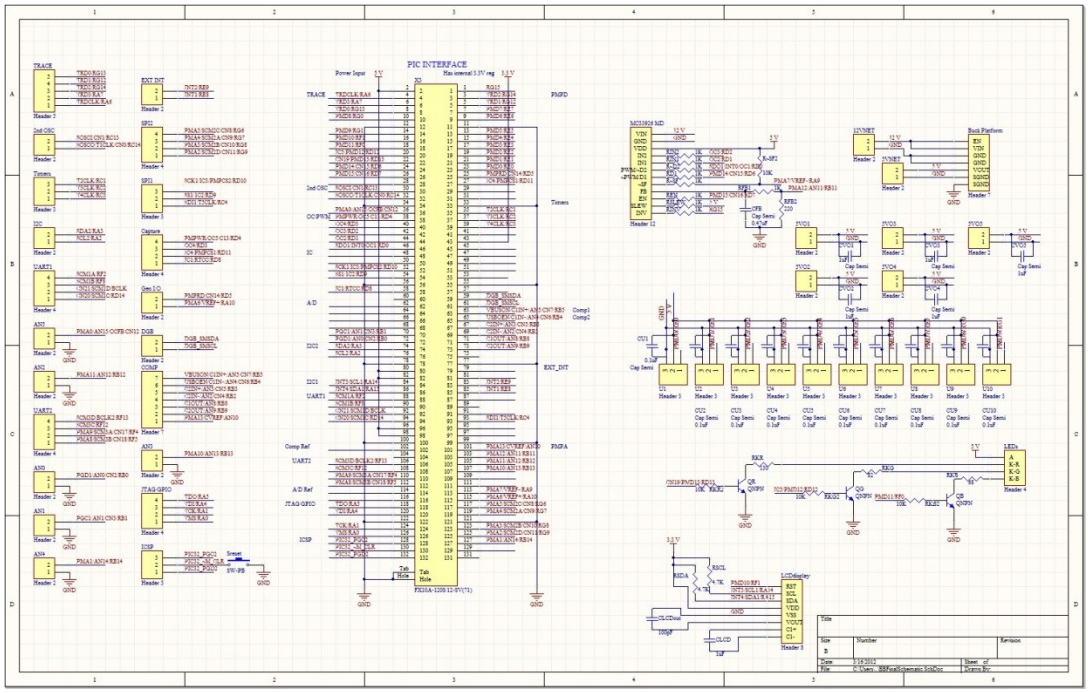
Discarded 4x6" PCB Design before Routing



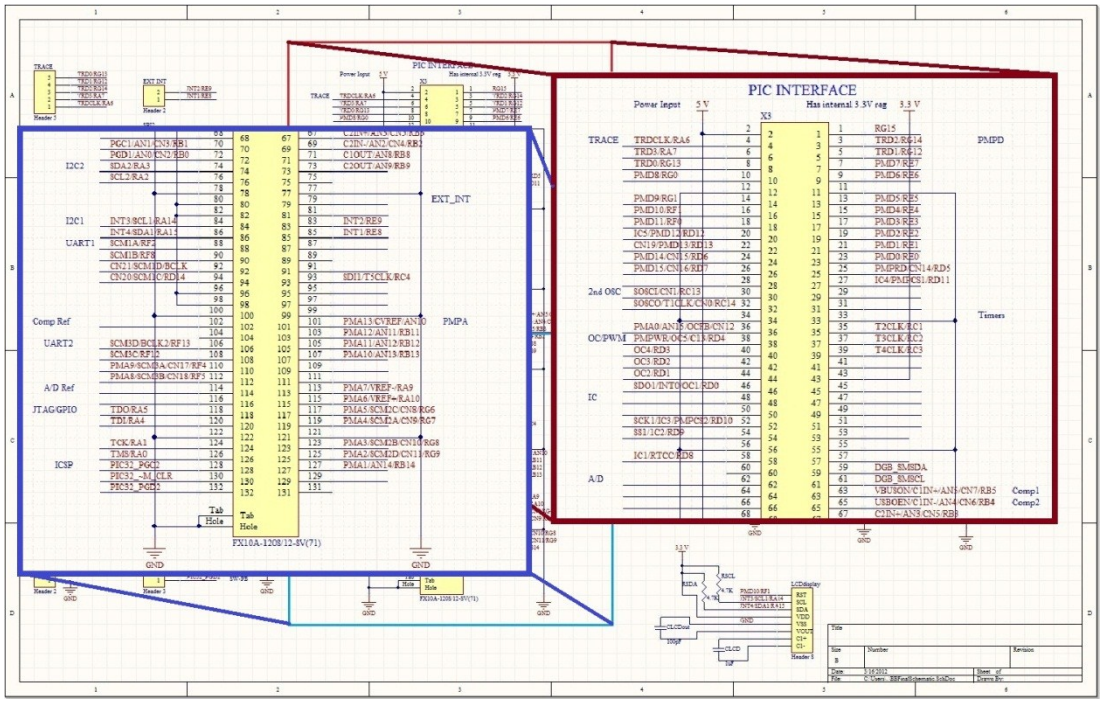
Discarded 2-layer 5x6" PCB Design AutoRoute



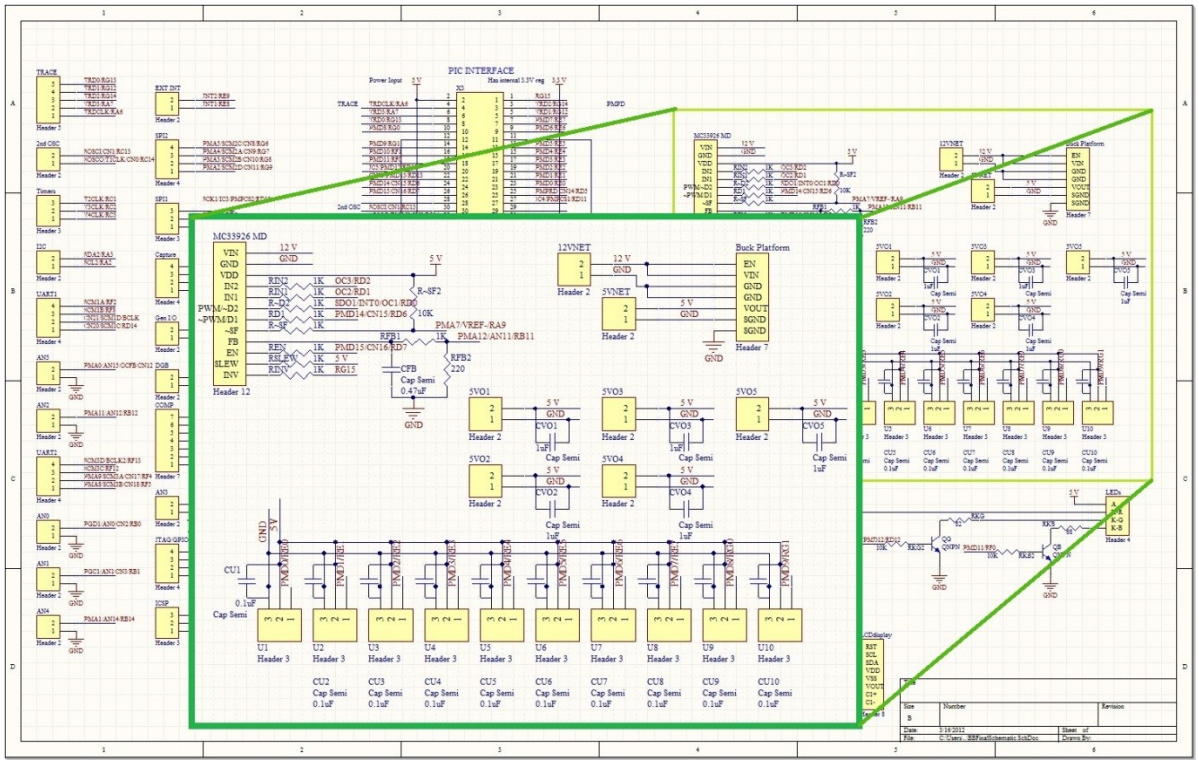
Discarded 5x6" PCB Design before Routing



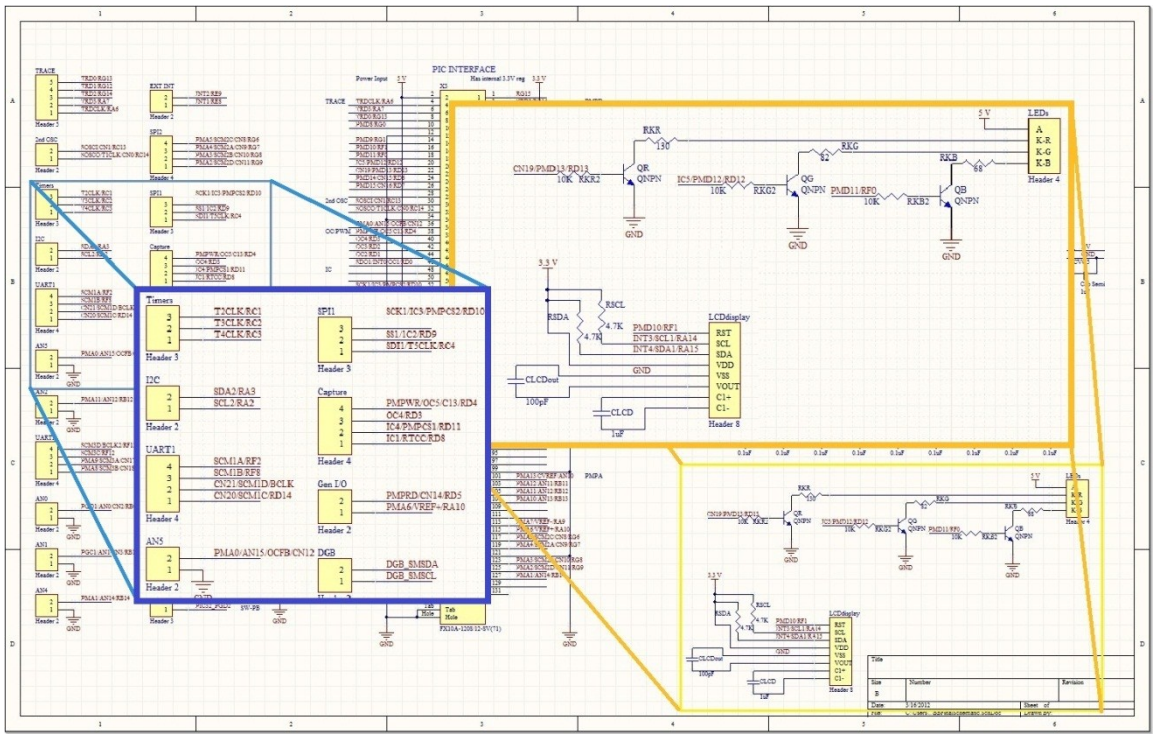
Final Breakout Board Schematic



Final Breakout Board Pic32 Interface Header Schematic



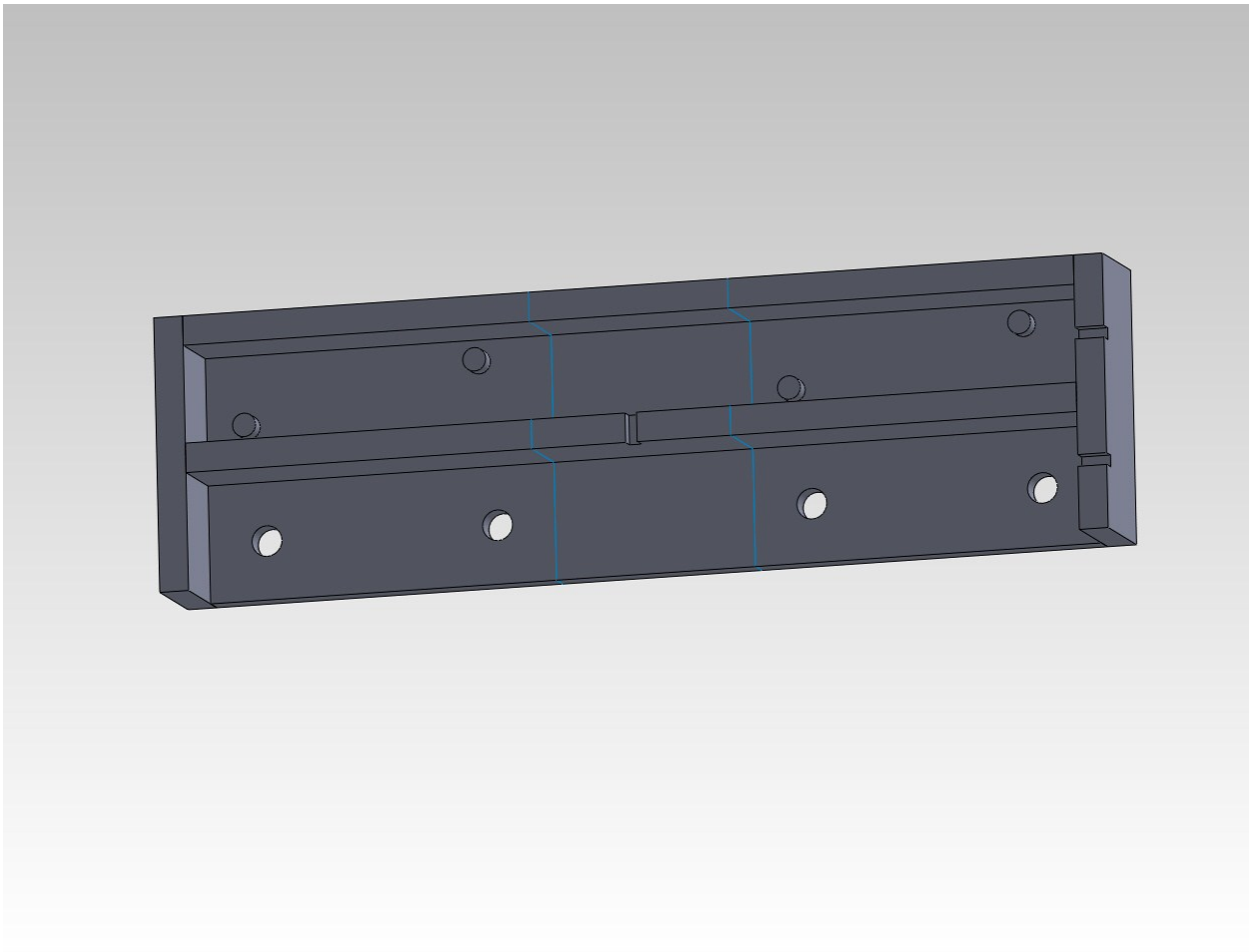
Final Breakout Board Ultrasonic Sensor Headers Schematic



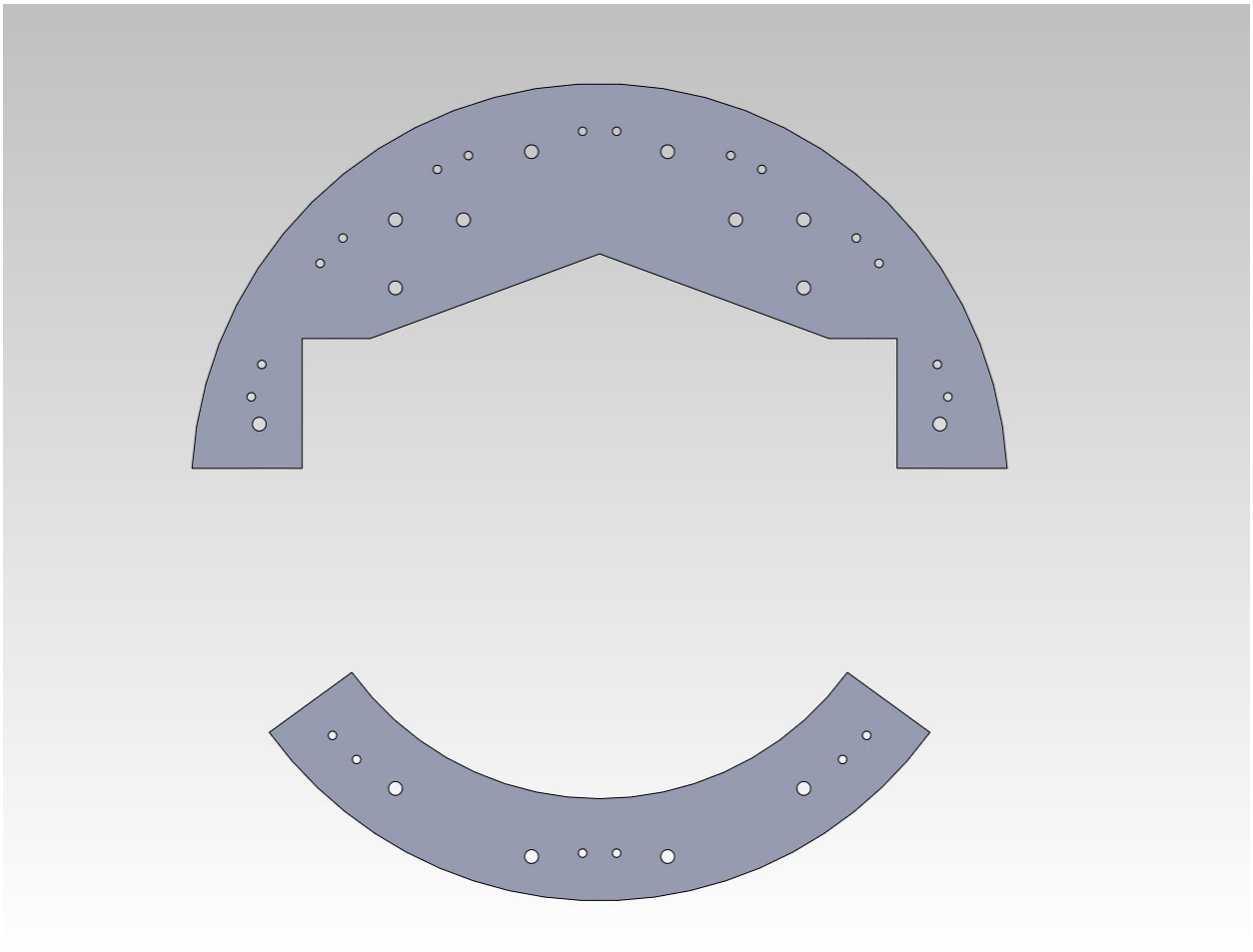
Final Breakout Board General I/O Pins and LCD Display Schematic

Appendix B: Mechanical Design

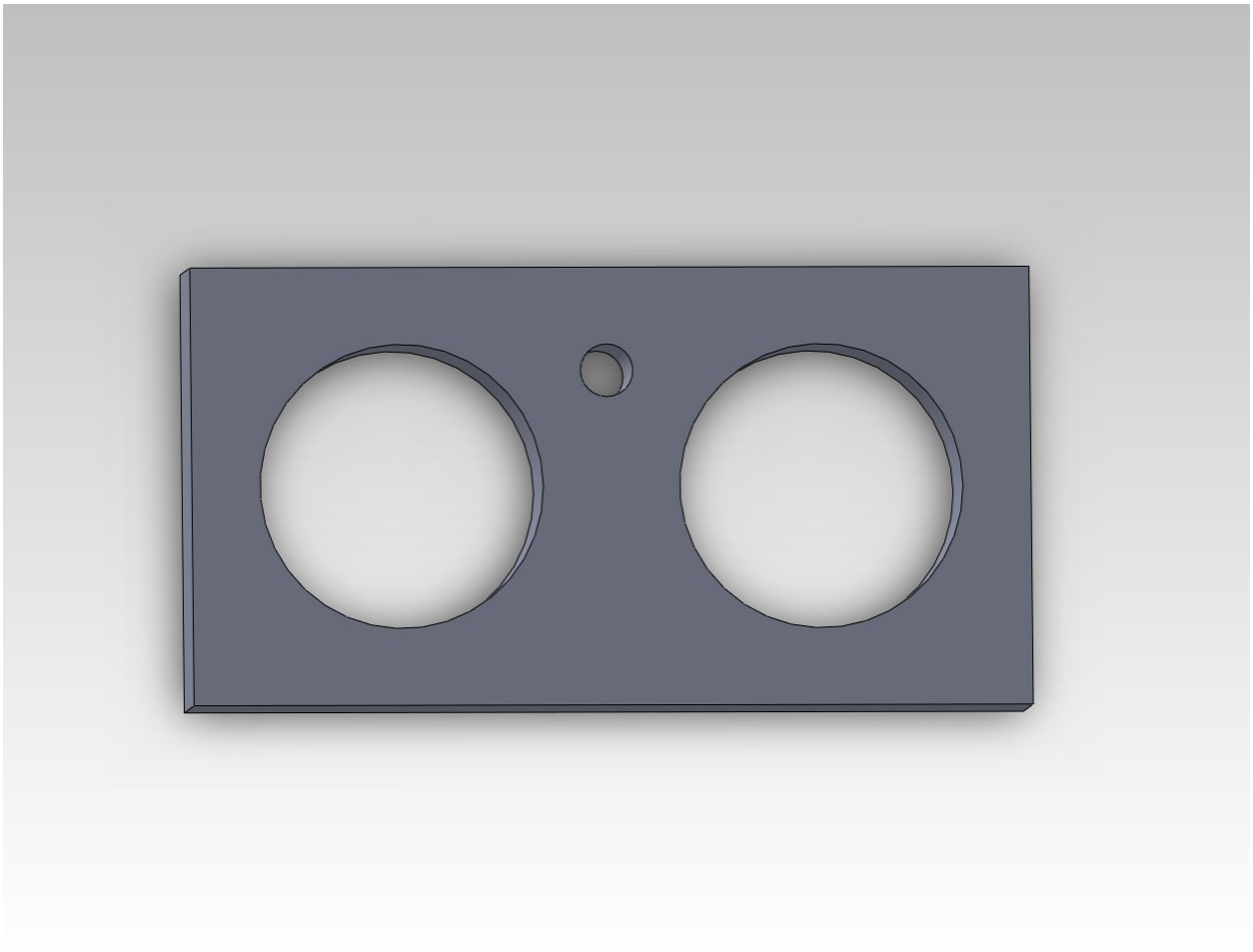
Stereo Camera Casing



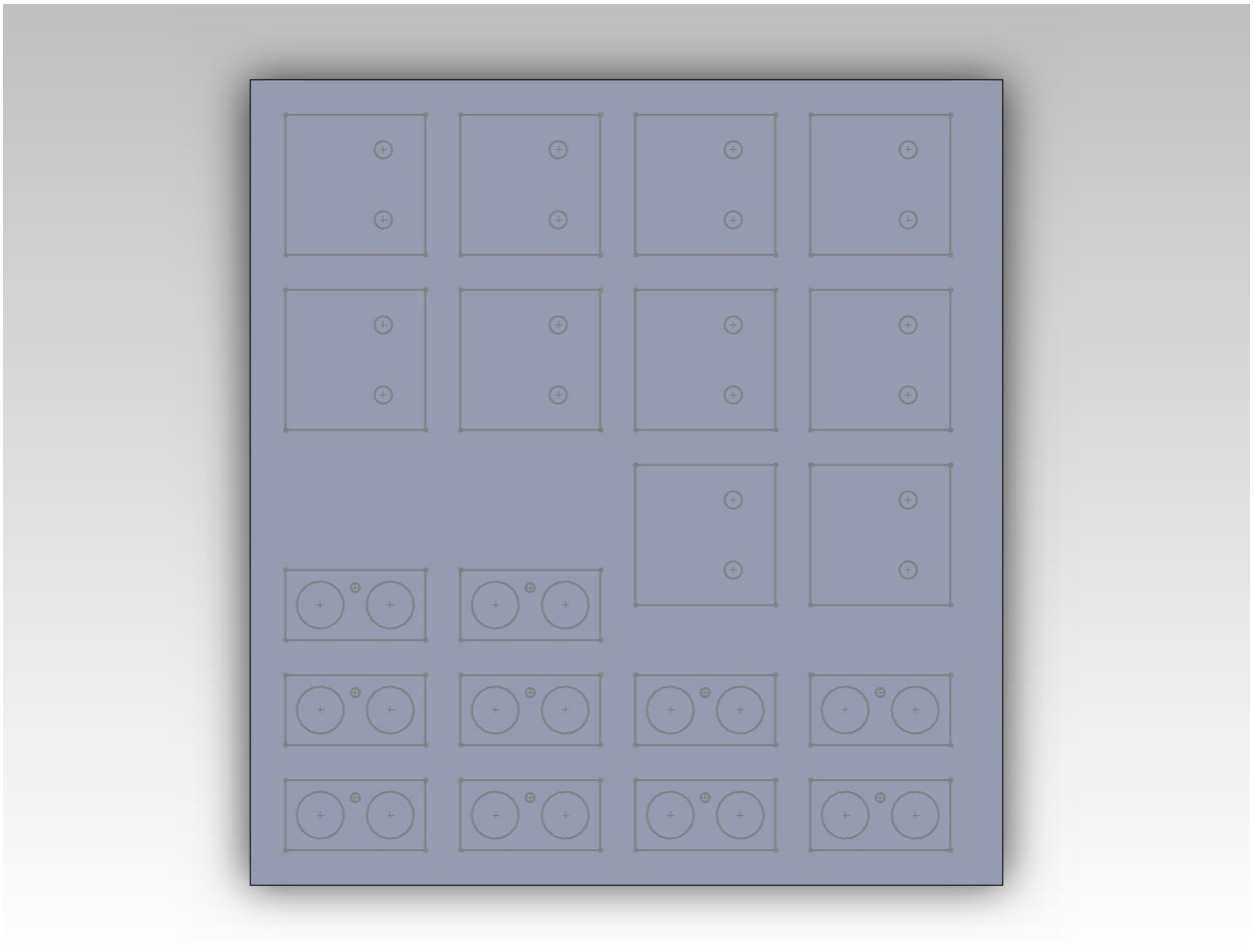
Ultrasonic Mounting Plates



Ultrasonic Bracket Faceplate



Ultrasonic Bracket Lasercut Pattern



Appendix C: Hardware

Breakout Board Bill of Materials					
Name	**Manufacturer**	**Manufacturer PN**	**Digikey PN**	**Quantity**	**Links**
Pic32 Interface	Hirose Electric Co Ltd	FX10A-120S/12-SV(71)	H11234-ND	1	http://search.digikey.com/us/en/products/FX10A-120S%2F12-SV%2871%29/H11234-ND/1036526
2 Pin Header 0.1" Pitch	TE Connectivity	640456-2	A1921-ND	18	http://search.digikey.com/us/en/products/640456-2/A1921-ND/109003
2 Pin Plug 0.1" Pitch	TE Connectivity	1375820-2	A99613-ND	18	http://search.digikey.com/us/en/products/1375820-2/A99613-ND/1864915
3 Pin Header 0.1" Pitch	TE Connectivity	640456-3	A19470-ND	12	http://search.digikey.com/us/en/products/640456-3/A19470-ND/259010
3 Pin Plug 0.1" Pitch	TE Connectivity	1375820-3	A99614-ND	22	http://search.digikey.com/us/en/products/1375820-3/A99614-ND/1864916
4 Pin Header 0.1" Pitch	TE Connectivity	640456-4	A1922-ND	7	http://search.digikey.com/scripts/dksearch/dksus.dll?FullDetail&name=A1922-ND#matingproducts
4 Pin Plug 0.1" Pitch	TE Connectivity	1375820-4	A99615-ND	7	http://search.digikey.com/us/en/products/1375820-4/A99615-ND/634994
5 Pin Header 0.1" Pitch	TE Connectivity	640456-5	A19471-ND	1	http://search.digikey.com/us/en/products/640456-5/A19471-ND/259011
5 Pin Plug 0.1" Pitch	TE Connectivity	1375820-5	A99616-ND	1	http://search.digikey.com/us/en/products/1375820-5/A99616-ND/1864917
7 Pin Header 0.1" Pitch	TE Connectivity	640456-7	A19472-ND	2	http://search.digikey.com/us/en/products/640456-7/A19472-ND/259012
7 Pin Plug 0.1" Pitch	TE Connectivity	1375820-7	1375820-7-ND	2	http://search.digikey.com/us/en/products/1375820-7/1375820-7-ND/1864919
8 Pin Header	TE Connec	640456-8	A1924-ND	1	http://search.digikey.com/us/en/products/640456-8/A1924-ND/

0.1" Pitch	tivity				
8 Pin Plug 0.1" Pitch	TE Connectivity	1375820-8	A99618-ND	1	http://search.digikey.com/us/en/products/1375820-8/A99618-ND/1864920
12 Pin Header 0.1" Pitch	TE Connectivity	1-640456-2	A19475-ND	1	http://search.digikey.com/us/en/products/1-640456-2/A19475-ND/259015
12 Pin Plug 0.1" Pitch	TE Connectivity	1-1375820-2	A99612-ND	1	http://search.digikey.com/us/en/products/1-1375820-2/A99612-ND/1864923
0.1" Pitch Plug Contacts	TE Connectivity	1375819-1	A100453CT-ND	200	http://search.digikey.com/us/en/products/1375819-1/A100453CT-ND/2233146
8 Pin Plug 0.079" Pitch	Sullins Connector Solutions	SWH201-NULN-S08-UU-WH	S9432-ND	1	http://search.digikey.com/us/en/products/SWH201-NULN-S08-UU-WH/S9432-ND/2411378
0.079" Pitch Plug Contacts	Sullins Connector Solutions	SWT201-UPTN-S01-UU-UU	S9475CT-ND	10	http://search.digikey.com/us/en/products/SWT201-UPTN-S01-UU-UU/S9475CT-ND/2769668
68 Ohm SM Resistor	Panasonic ECG	ERJ-12ZYJ680U	P68WCT-ND	1	http://search.digikey.com/us/en/products/ERJ-12ZYJ680U/P68WCT-ND/249683
130 Ohm SM Resistor	Panasonic ECG	ERJ-12ZYJ131U	P130WCT-ND	1	http://search.digikey.com/us/en/products/ERJ-12ZYJ131U/P130WCT-ND/249690
82 Ohm SM Resistor	Panasonic ECG	ERJ-12ZYJ820U	P82WCT-ND	1	http://search.digikey.com/us/en/products/ERJ-12ZYJ820U/P82WCT-ND/249685
10K Ohm SM Resistor	Panasonic ECG	ERJ-12ZYJ103U	P10KWCT-ND	4	http://search.digikey.com/us/en/products/ERJ-12ZYJ103U/P10KWCT-ND/249735
4.7K Ohm SM	Panasonic ECG	ERJ-12ZYJ472U	P4.7KWCT-ND	2	http://search.digikey.com/us/en/products/ERJ-12ZYJ472U/P4.7KWCT-ND/249727

Resistor					
1K Ohm SM Resistor	Panasonic ECG	ERJ-12ZYJ102U	P1.0KWCT-ND	9	http://search.digikey.com/us/en/products/ERJ-12ZYJ102U/P1.0KWCT-ND/249711
220 Ohm SM Resistor	Panasonic ECG	ERJ-12ZYJ221U	P220WCT-ND	1	http://search.digikey.com/us/en/products/ERJ-12ZYJ221U/P220WCT-ND/249695
Buck Converter Module	National Semiconductor	LMZ14203EVAL/NOPB	LMZ14203EVAL-ND	1	http://search.digikey.com/us/en/products/LMZ14203EVAL%2FNOPB/LMZ14203EVAL-ND/2194716
Heat Shrink Tubing 1/8"	Qualtek	Q2-F3X-1/8-01-QB48IN-25	Q2F3X018B-ND	12	http://search.digikey.com/us/en/products/Q2-F3X-1%2F8-01-QB48IN-25/Q2F3X018B-ND/1210323
0.1uF SM Capacitor	Murata Electronics NA	GRM188R71E104KA01D	490-1524-1-ND	10	http://search.digikey.com/us/en/products/GRM188R71E104KA01D/490-1524-1-ND/587865
1uF SM Capacitor	TDK Corporation	C1608Y5V1A105Z	445-1328-1-ND	6	http://search.digikey.com/us/en/products/C1608Y5V1A105Z/445-1328-1-ND/567635
100pF SM Capacitor	TDK Corporation	C1608C0G1H101J	445-1281-1-ND	1	http://search.digikey.com/us/en/products/C1608C0G1H101J/445-1281-1-ND/567662
0.47uF SM Capacitor	TDK Corporation	C1608Y5V1E474Z	445-3454-1-ND	1	http://search.digikey.com/us/en/products/C1608Y5V1E474Z/445-3454-1-ND/1801553
SMT NPN Transistors	Zetex	ZXTN08400BFFTA	ZXTN08400BFFCT-ND	4	http://search.digikey.com/us/en/products/ZXTN08400BFFTA/ZXTN08400BFFCT-ND/1557772
SPST-NO Switch	Panasonic ECG	EVQ-11U04M	P8082SCT-ND	1	http://search.digikey.com/us/en/products/EVQ-11U04M/P8082SCT-ND/259567
User Interface Computer Parts					
Item	**Cost**	**Shipping**	**Total**		**Link**
All-in-One PC	$ 349.99	$ 9.13	$ 359.12		http://www.tigerdirect.com/applications/SearchTools/item-details.asp?EdpNo=1422495&CatId=1888

USB Programmable DC-DC Converter	$ 59.95		$ 59.95		http://www.mini-box.com/DCDC-USB
Control Computer Bill of Materials					
Item	**Unit Price**	**Shipping**	**Quantity**	**Total**	**Link**
Micro-ATX AM3 Motherboard	$ 59.99	$ -	1	$ 59.99	http://www.newegg.com/Product/Product.aspx?Item=13-138-283
AMD Athlon II X4 CPU	$ 140.99	$ -	1	$ 140.99	http://www.newegg.com/Product/Product.aspx?Item=N82E16819103899
2GBx2 DDR3 RAM	$ 31.99	$ -	1	$ 31.99	http://www.newegg.com/Product/Product.aspx?Item=N82E16820231275
OCZ 32GB SSD	$ 54.99	$ -	1	$ 54.99	http://www.newegg.com/Product/Product.aspx?Item=N82E16820227510
GeForce GT 430 GPU	$ 69.99	$ -	1	$ 69.99	http://www.newegg.com/Product/Product.aspx?Item=N82E16814162067
Micro-ATX Case	$ 29.99	$ 9.99	1	$ 39.98	http://www.newegg.com/Product/Product.aspx?Item=N82E16811147112
ATX DC-DC Power Supply	$ 89.50	$ 9.41	1	$ 98.91	http://www.mini-box.com/M4-ATX
Sensors					
Logitech C260 Webcam	$ 19.99	$ 3.82	2	$ 47.62	http://www.amazon.com/dp/B003LVZO8I
Ultrasonic Sensors	$26.99		10	$269.90	www.parallax.com/Store/Sensors/ObjectDetection/tabid/176/ProductID/92/CategoryID/51/List/0/Level/a/Default.aspx
3-Axis Orientation Sensor	$ 149.00		1	$ 149.00	http://www.pololu.com/catalog/product/1256
Item	**Unit Cost**	**Quantity**	**Shipping**	**Total**	**Link**

Power System					
Atbatt					
12V Lead Gel Cell Battery	$197.09	1	$0.00	$197.09	http://www.atbatt.com/product/22548.asp
Battery Charger	$45.02	1	$0.00	$45.02	http://www.amazon.com/Battery-Tender-021-0156-Charger-Model/dp/B000NCOKQK/
Amazon					
Computer Speakers	$58.08	1	$0.00	$58.08	http://www.amazon.com/Creative-Inspire-Multimedia-Speaker-Technology/dp/B0028N6YH0/
Other					
Powered USB Hub	$21.95	1	$0.00	$21.95	http://www.amazon.com/Plugable-Port-Speed-Power-Adapter/dp/B003Z4G3I6/
USB Extension	$4.99	3	$0.00	$14.97	http://www.amazon.com/AmazonBasics--Male--Female-Extension-Cable/dp/B001TH7GV4/
Pololu					
Motor Driver	$17.95	1	$4.95	$22.90	http://www.pololu.com/catalog/product/1212