

**Project DZERV**

**Disaster Zone Emergency Response Vehicle**

Spring 2012-Fall 2012

Group 18

Group Members:

Marcial Rosario

Robert Smith

Michael Lopez

Table of Contents

1.0 Executive Summary.......................................................................................................1

2.0 Project Description.........................................................................................................2

2.1 Project Motivation and Goals...................................................................................2

2.2 Objectives.................................................................................................................3

2.3 Project Requirements and Specifications.................................................................4

2.3.1 Risks................................................................................................................6

2.3.2 Team Management..........................................................................................6

3.0 Research Related to Project Definition..........................................................................6

3.1 Existing Similar Projects and Products....................................................................6

3.1.1 Stevens Institute of Technology Search and Rescue.......................................7

3.1.2 OSU Search and Rescue Robot.......................................................................7

3.1.3 Autonomous Ball Collector (ABC).................................................................9

3.1.4 Other Available Products...............................................................................10

3.1.4.1 Unmanned Aerial Vehicles (UAVs)..................................................10

3.1.4.2 Kinect Robot......................................................................................11

3.2 Relevant Technologies...........................................................................................12

3.2.1 Wireless Protocols.........................................................................................12

3.2.1.1 Wi-Fi (IEEE Standard 802.11).........................................................12

3.2.1.2 Bluetooth..........................................................................................13

3.2.1.3 Infrared Communications.................................................................15

3.3 Strategic Components............................................................................................15

3.3.1 Mobile Base...................................................................................................16

3.3.2 Surface Mount vs. Through-Hole..................................................................17

3.3.3 Microcontrollers.............................................................................................17

3.3.4 Cameras..........................................................................................................18

3.3.5 Sensors............................................................................................................18

3.3.6 RF Transmitters and Receivers.......................................................................19

3.3.7 Power Supplies................................................................................................19

3.3.8 Base Computer.................................................................................................20

3.3.9 Graphical User Interface..................................................................................21

3.3.10 Voltage and Current Regulation....................................................................21

4.0 Project Hardware and Software Design.......................................................................21

4.1 Microcontroller......................................................................................................21

4.1.1 Research Phase.............................................................................................22

4.1.2 PIC16F886...................................................................................................24

4.1.3 Design Phase................................................................................................24

4.1.4 Programming................................................................................................26

4.1.5 Testing and Debugging.................................................................................28

4.2 Sensors/Instruments.....................................................................................................29

4.2.1 Pressure Sensor (MPLII5A1).............................................................................29

4.2.2 Temperature and Humidity Sensor (HTM1735LF)...........................................31

4.2.3 Ultrasonic Range Finder (LV-EZ1)...................................................................33

4.2.4 LED Bar..........................................................................................................35

4.3 Camera (AirSight Wireless IP Camera)...................................................................36

4.3.1 MAX232 Chip................................................................................................39

4.4 Software/Interface....................................................................................................41

4.4.1 GUI.................................................................................................................42

4.4.2 Robot Control.................................................................................................44

4.5 Voltage Regulators...................................................................................................45

4.5.1 Linear Regulators vs. Switching Regulators...................................................46

4.5.2 Linear Regulator Options................................................................................48

4.5.3 Switching Regulators......................................................................................51

5.0 Power and Communication..........................................................................................52

5.1 Chassis....................................................................................................................52

5.2 Motors.....................................................................................................................54

5.2.1 Continuous DC..............................................................................................54

5.2.2 Stepper Motor................................................................................................56

5.2.3 RC Servo Motors...........................................................................................57

5.2.4 Price Comparison...........................................................................................59

5.3 Communications...................................................................................................61

5.3.1 Infrared Communication..............................................................................61

5.3.2 RF (Radio Wave) Communication...............................................................62

5.3.3 Wireless Adapters ........................................................................................63

5.4 Power Supply........................................................................................................67

5.4.1 Electronics Power Supply.............................................................................67

5.4.2 Alkaline.........................................................................................................68

5.4.3 Lithium..........................................................................................................69

5.4.4 Lead Acid......................................................................................................70

5.4.5 Nickel Cadmium (NiCd)...............................................................................70

5.4.6 Nickel Metal Hydride (NiMH)......................................................................70

5.5 Choosing a Battery...............................................................................................71

5.5.1 Battery Charger.............................................................................................73

5.6 Antenna.................................................................................................................74

5.6.1 Ceramic Antenna...........................................................................................74

5.6.2 Planar Antenna...............................................................................................75

5.6.3 Whip Antenna.................................................................................................76

5.7 Motor Controller....................................................................................................77

6.0 Project Prototype Construction and Coding................................................................79

6.1 Strategic Locations................................................................................................79

6.1.1 EECS Senior Design Laboratory..................................................................80

6.1.2 Group Members’ Homes..............................................................................80

6.2 Parts Acquisition....................................................................................................81

6.2.1 Vendor Selection...........................................................................................81

6.3 Printed Circuit Boards............................................................................................82

6.3.1 ExpressPCB....................................................................................................82

6.3.2 Sunstone Circuits............................................................................................83

6.3.3 PCBFabExpress...............................................................................................84

6.3.4 Final Decision..................................................................................................85

7.0 Testing..........................................................................................................................85

7.1 Chassis...................................................................................................................85

7.2 Speed Test..............................................................................................................86

7.3 Battery Life............................................................................................................87

7.4 Battery Charging....................................................................................................88

7.5 Video Camera........................................................................................................88

7.6 PCB Testing...........................................................................................................89

7.7 Wireless Control Testing........................................................................................90

7.8 Sensor Testing........................................................................................................90

7.9 Microcontroller Testing..........................................................................................91

8.0 Bill of Materials and Responsibilities..........................................................................92

8.1 Bill of Materials......................................................................................................92

8.2 Responsibilities.......................................................................................................95

8.3 Project Milestones...................................................................................................96

Appendix A: Bibliography.................................................................................................97

Appendix B: Permissions.................................................................................................101

Appendix C: Index of Figures..........................................................................................108

Appendix D: Index of Tables...........................................................................................110

**1.0 Executive Summary**

The major focus for this project was to serve as a search and rescue robot which could be implemented in the field in dangerous, disaster-like situations. This robot serves as a means of making human involvement in dangerous search-and-rescue operations much more safe, thus enabling such operations to become much more successful.

To accomplish this task, we used a variety of sensors, such as temperature sensors, pressure sensors, as well as a camera and a rangefinder to help our robot “see” and assist with navigation. This data was continuously measured by the various sensors and streamed to and manipulated by the microcontroller. This data was then sent back to a computer so that it could be interpreted by the operator who will be able to see what is being captured by the camera, as well as a variety of other parameters about the area, such as the ambient temperature, atmospheric pressure, as well as distances to objects using the rangefinder.

We used an RF transmitter and receiver in order to communicate back and forth between the robot and the control unit, which happens to be the operator’s laptop computer. We also have measures in place such that the robot is able to be controlled using the laptop’s keyboard. This allows the user to not only guide the robot where it needs to go, but also enables the operator to determine whether it will be safe for a person to enter the area to complete the rescue operation using the data collected by the sensors on the robot.

The microcontroller, wireless communication assembly, sensors, and camera are all mounted onto a remotely controllable vehicle, which we purchased separately, and modified to suit our needs. This required us to completely gut the factory-built vehicle and essentially rebuild its electrical circuitry from the ground up, since we did not utilize the remote control that came with it. That means that in addition to programming the microcontroller to handle the data being fed to it by the sensors and camera, we also needed to interface it with the vehicle’s motors such that we could manage to control it remotely via a computer.

In summary, the goals of this project included designing a search and rescue vehicle capable of detecting whether the conditions in an area are conducive to human survivability and relaying this information back to a remotely located operator. We intended to have a working prototype accomplish this task as efficiently as possible, consuming as little power as possible, while minimizing costs and resources for our group members. After the successful completion of this prototype, such a robot is able to be used in an incredibly broad range of applications, from military reconnaissance operations, to aiding rescuers in areas ravaged by fires, floods, or other similar natural disasters.

**2.0 Project Description**

**2.1 Project Motivation and Goals**

When we first embarked on this mission to choose a project, we were rather indecisive in the beginning, as neither member of our group really had a solid idea of what exactly it was that they wished to accomplish in completing this senior design project. For a group as diverse as ours, with each member coming from different backgrounds with different areas of expertise, settling down and choosing one project on which to focus our efforts was a rather difficult task to accomplish. We had gone back and forth over many different projects, ranging from the simplistic to the overly complex. We discussed whether we wanted a mostly software-based project, or whether we wanted a more concrete, mostly hardware-based project.

Ultimately, we decided that we would like to have something that would be a split down the middle in terms of software and hardware, as we have two electrical engineers and one computer engineer. We therefore arrived at the conclusion that some type of robot would likely be the best fit for our group, as this would mean plenty of hardware and software work to go around for everyone, and we felt that this would be a great way for all of us to maximize our exposure to both sides of electrical and computer engineering.

Once we all descended upon the idea that a robot would be the best and most useful way for us to proceed, we needed to decide what type of robot we would build. After a great deal of research, we mulled through the seemingly endless possibilities. We could have moved forward with a quad-copter type project, which seemed to be quite popular at other universities. We also could have gone with something that could help automate some mundane everyday task around the home, such as retrieving a beverage from a refrigerator. We very well could have even designed a robot for entertainment purposes, such as for a game or something similar.

After many sessions of lengthy discussion on what we would be designing, we finally settled on the idea that we could design and implement something for the greater good: a search-and-rescue robot. We felt that proceeding with such a project allowed for a great deal of freedom in our design and implementation, while also exposing us to a vast array of design techniques, as well as the product development lifecycle as a whole. Designing something of this nature would gave us each the ability to focus on our interests and goals, while allowing us to work together to collectively design and build something that could one day prove to be helpful in real-world applications, especially given the extent to which robots are used today. This project not only provided us with the opportunity to test and hone our design and implementation abilities, but it also provided us with invaluable experience which is necessary for a successful career in engineering.

As two of our members currently have fairly close connections to the defense industry, it seemed logical that this was the direction that we would proceed in, as there was already a presence of these types of machines on the market, and thus gave a large pool from which to draw inspiration, as well as gain relevant, useful knowledge on how to design and construct this robot. We hoped that with this project, we would be able to accomplish the goals that we had set before us, including the successful interfacing of the microcontroller with the various sensors, as well as successfully managing to network all the wireless communication in such a way that the data will be able to travel in such a manner that the operator will be able to make use of real-time data and video being streamed from the robot. After a lot of time and effort put into this project, we succeeded.

Apart from the technical goals that we had in mind for this project, we truly wanted to create something that could benefit mankind. For example, whenever an earthquake strikes, or whenever an area is destroyed by a hurricane or tornado, people risk their lives to aid in the search and rescue operation. With the aid of something like our project, we can now assist in helping to speed up the operation by locating victims and survivors. We could also use our robot to locate potential safety hazards, such as downed power lines and fires so that people proceed with caution as necessary.

**2.2 Objectives**

Search and Rescue robots are a topic of both wide discussion and importance in today’s society. With catastrophes such as the tragedy of September 11th, 2011 and the recent tornadoes in the Midwest, search and rescue robots can be of vital and beneficial use in aiding and guiding rescue workers through disaster zones. Search and rescue robots might be deployed in disasters such as mining accidents, urban disasters, hostage situations, and explosions.

These robots can accomplish tasks on the rescue sites where either human or canine resources are rendered powerless, or where the situation might be too dangerous for a human to enter. Other benefits of such robots include reduced personnel usage, reduced fatigue of rescue workers, and gaining access to areas which may be unreachable by either human or canine.

The first major objective of DZERV was to be able to deploy it into an area where neither human nor canine could gain access to, whether it be due to severe conditions or scabrous terrain. Aside of simply gaining access to such areas, DZERV is able to retrieve some sort of useful information. Such information will come in the form of video feed locating live victims which need aid while sending back sensor data, informing rescue workers of issues such as extreme heat or fire, hazardous environments, or possible explosions.

The second major objective of DZERV was to transmit video data back to the rescue worker’s computer station. Such a feature could work for numerous other applications besides searching for live victims. The use of video feedback could be specifically useful for bomb detection in various situations. Take for example military checkpoints, while military officials are checking identifications, other officials could be operating the robot from a remote location checking under cars for explosives or other dangerous devices.

**2.3 Project Requirements and Specifications**

Our project was to consist of three major components, a mobile platform, an array of sensors, and a wireless communication link. DZERV is able to survey and gather information from its environment including temperature, humidity, and barometric pressure. Upon acquiring necessary sensor data, measured values are then wirelessly transmitted and displayed to the base computer in real time. Aside from transmitting sensor data, DZERV is also capable of transmitting real-time video feedback to the operator at the base computer. Movement of the robot is based on commands sent over a wireless link from the operator’s base computer. Essentially, DZERV sends both video and sensor data to the base computer, while the base computer sends signals to control the car. All of this has been implemented using wireless technology.

As previously mentioned, DZERV is able to send temperature, humidity, and barometric pressure values retrieved from its surroundings back to the user operating it from the base computer. Ideally, DZERV should be placed in areas where living human rescue workers cannot gain access to. Due to the conditions and environments which DZERV will be operating in through human control, it was necessary to implement some form of “eyes” on our robot. For our application, we have mounteda wireless camera onto the robot. This allows the user operating DZERV to know not only the area the robot is operating in, but also whether or not there is a potentially dangerous situation at hand. Aside of such benefits, mounting a wireless camera onto the robot also alleviates some pressures from the operator. The operator does not need a direct line of sight with the robot to guide it. Rather, he/she is able to maneuver the robot using the video feedback through the camera software installed on the base computer.

Another aspect of DZERV consists of the sensors mounted onto it. We had to carefully consider the wireless transmission of sensor readings back to the operator at the base computer. The operator had to be able to operate the robot form the base computer and have dependable real-time data retrieved from the sensors. Such information had to be real-time in order to allow the user to make active decisions about the environment that it was in. Such real-time information could lead to possible life or death situations in a harsh environment such as a fire or explosion, which the DZERV has no problem doing.

We also had to take into consideration obstacle avoidance and proper maneuvering. Due to this, we decided to mount an ultrasonic range finder onto our robot. By implementing a range finder, we were be able to properly avoid obstacles which could impede the movement of the robot. In the case of our applications an ultrasonic range finder was deemed more useful and beneficial than a bumper switch or limit switch. Unlike a bumper or limit switch which alerts you only when they have been hit, the range finder alerts us of an obstacle in its path prior to collision, thus not only giving the user enough time to properly and safely navigate around obstacles, but to also protect the robot itself.

Table 1 below provides a brief overview of the requirements for our project.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Requirement** | **Description** | **Testing** |
| 1 | Communication  range of 75 ft. | Communication  from TX should  send to RX from  100 ft. | Drive DZERV  until  communication  failure and  record distance. |
| 2 | Be able to  provide  audio/video | DZERV must  send real-time  audio/video to  base computer. | Check the live  feed of the robot  as it operates. |
| 3 | Battery life of  approximately  30 minutes |  | Run the motor  until it dies  while recording  the time. |
| 4 | Be able to  navigate  multiple types of  terrain  (pavement,  grass, etc.) | DZERV should  be prepared  for whatever  terrain it  encounters. | Drive the robot  over various  terrains to  ensure it can  move  efficiently. |
| 5 | Must weigh less  than 15 lbs. | DZERV should  be light-weight. | Measure the  robot on a scale. |
| 6 | Must maintain  low enough  temperature | There should be  no failure of  electronic  equipment. | Use a  temperature gun  to measure the  robot while it  operates. |
| 7 | Simple to use  GUI | The operator  should be able to  control DZERV  with ease. | Find someone  unfamiliar with  the project and  have them test it  and listen to  their feedback. |
| 8 | Portability | Aside of being  lightweight, it  should be easy  to transport. | Attempt to make  the project a one  piece robot. |
| 9 | Auto  notification | The system  should alert the  operator if a  condition that  needs attention  arises. | Place interrupts  within the GUI  code and test  dummy  conditions to see  if they pop up. |

Table 1: Table of Requirements

**2.3.1 Risks**

Just like any other project or task in life, we had to prepare to deal with possible risks. Since we are all full time students and two of us held demanding internships, preparing for risks was crucial. One risk separating us from other groups was the fact that we only had three group members. This was a positive aspect in the sense that our project did not become overcomplicated. However, we had to do more research and design work to accomplish this project than a group with four members. Another broad risk was the fact that none of us had ever even attempted to embark on such a mission or really held any experience designing, building, and testing a robot. More specific risks were involved with the robot itself. What if a sensor electronically failed? What if the user interface did not properly receive values? What if we miscalculated power and current, soldered a PCB together, and a component failed? These were all things that we had to prepare for by making sure we triple checked all of our work and calculations as we went. Of course, like any other project, we had some bugs and kinks that we had to work out, but we did the best we could to prepare for major risks.

**2.3.2 Team Management**

Since the original formation of our group, we all decided that this would be a collaborative effort, and that we would all take equal responsibility in our project by having no set leader. This was strongly influenced by the fact that none of us had ever been faced with such a feat, so we decided that tackling it together equally would be an optimal solution. Despite having no team leader, we all took initiative to make sure we were all kept well informed and up to date as to what everyone was working on. We scheduled regular weekly meetings where we would discuss what he had researched/worked on/etc. Doing this insured we were all kept up to date and that no two people were completing the same task.

For our documentation, we chose to utilize Dropbox, a virtual storage cloud, and have one main document, where each team member could write updates kept on personal machines to allow for the convenience of working on this project anywhere and anytime. We also chose to divide the documentation into small portions that each team member could individually manage. When splitting up the research and documentation, we decided that each team member had to choose what they were interested in working with, or had prior experience or knowledge of. If we were working on a section in which we expressed interest in or held knowledge in, we felt our productivity would be at an optimal level.

**3.0 Research Related to Project Definition**

**3.1 Existing Similar Projects and Products**

At the time we had sat down to discuss and finalize our project plans and goals, neither one of our group’s members had a great deal of experience working with much of the technologies which we used to accomplish our objectives for this project. It therefore became a necessity to spend a great deal of time researching in order for us to acquire the knowledge necessary to face the challenge before us. Much of the challenge in completing this senior design project came from absorbing and assimilating information from the work of groups and companies who have traveled this path previously. Fortunately, there were countless projects and products similar to what we planned to accomplish from which we were able to draw inspiration from.

**3.1.1 Stevens Institute of Technology Search and Rescue**

One such example was a project built by a group at the Stevens Institute of Technology in New Jersey. This particular group had come to nearly the same conclusion that we did, in that a robot of some sort would be a great way to combine the disciplines of computer, electrical, and in their case, mechanical engineering. Their particular design included some of the things that we ended up using, such as a camera, a rangefinder, and various other sensors. One of the major differences between that project and the one that we ended up implementing lied in that particular group’s decision on how to proceed with actual construction of the robot. They had contemplated constructing their own chassis for their robot. This was something that our group neither had the experience and technical know-how, nor the desire to do, as this would complicate our design greatly. It would also likely greatly increase the time and costs involved in prototyping and manufacturing. This was also something that could have lead to issues with scalability at a later time during the actual building of the working prototype, as this could have drained quite a bit of resources for our group members had we found that what we had constructed was not exactly something that we really, truly needed or wanted. However, the Stevens group members decided that purchasing a robot kit from a manufacturer called Vex would be the best way for them to proceed. We decided that this too would also complicate things for us due to some of the design constraints for our Senior Design class, and therefore, we did not be purchase a robotics kit in order to build our project. This group also intended to make their robot autonomous so that it could navigate on its own using some path prediction algorithms in conjunction with some limit and bump switches located on the vehicle, for collision detection. While this would have made our robot a far more effective search and rescue vehicle, we felt that this would have increased the complexity of our project immensely, since none of our members had much experience with programming for Artificial Intelligence. Furthermore, they also included a passive infrared sensor to detect changes in temperature to indicate the presence of a human or other animal in the area. While we had given this some thought at first, we decided that we would rather not include such a sensor in our design. Furthermore, they had also intended on building their own obstacle course on which to test their device. Time and budget permitting, such a requirement may have been something that we could have achieved, but we did not expect to do this for the testing of our final prototype. Overall, this project seemed like a great source off of which to base our design, but we still had quite a few unanswered questions, and so further research was required.

**3.1.2 OSU Search and Rescue Robot**

Another project that we had encountered during our research was for a search and rescue type robot made by a senior design group at Oklahoma State University. This project, from which we based much of our design, had quite a bit of detailed information, which has actually helped us a great deal for the theory behind the functionality of our project. This group had essentially the same idea as the group from Stevens, in that creating a search and rescue robot would be a great way to integrate the disciplines of electrical and computer engineering. Unlike the group from Stevens, however, these students decided against purchasing an actual robotics kit like the Vex robotics kit that the Stevens students purchased. However, they also did not venture to build their own robot either, as this would again greatly increase the complexity of the project with the given time constraints with which they were working. Therefore, they ended up purchasing an already working remote-controlled vehicle to serve as the mobile platform for their robot upon which everything else was added or built. This was the route which we chose to take, since this simplified our design quite a bit, and allowed us to focus more of our time on what really mattered, which was the successful interfacing of our hardware and software and getting it to work properly as one unit. In addition to using an already working mobile platform, the members of the OSU group went one step further and created a few inventive ways to control the vehicle. They not only created their own remote control for the vehicle, but they also created a program such that the vehicle could be controlled via user inputs on a laptop computer’s keyboard. Furthermore, they also came up with a very creative way to control the vehicle via voice commands. However, as the idea behind this vehicle was that it should be able to be operated remotely, it seemed rather infeasible to include such a feature, as this would require the operator to be near the vehicle at all times unless through the use of some peripheral device the operator could transmit their voice commands directly to the search and rescue module, which would also increase the complexity of the project. Apart from these features, this project was essentially quite similar to the Stevens group’s project in that there was a camera and various sensors, which all communicated wirelessly with the operator’s base computer in order to successfully transmit data remotely and receive commands. One major difference between this project and the Stevens project was that while the Stevens group built their robot for autonomous control, the Oklahoma State group did not, and actually intended for their robot to be controlled by an operator at all times. That being said, the technical design for this robot was actually a little complicated. Their PIC microcontroller allowed five ten-bit-wide channels for Analog to Digital Conversion. Because more than one timer would need to be used in order for each sensor to be able to handle measurements in real time simultaneously, they needed to utilize MOSFETs to act as switches between the different devices so that they could all interface with the microcontroller’s Analog to

Digital Converters in order to get the data back to the user in real time. The PIC processor that we intended to use only had three timers on-chip, and therefore we thought we needed to implement a similar design in order to ensure that everything worked as intended so that we could take measurements as accurately as possible. However, due to unforeseen circumstances, we vouched to go with the Atmel brand of chips to act as our main processor and implemented a second XBee module to successfully transmit all of our data. Radio Frequency (RF) communications played a major role in this group’s design, as this was how they were able to successfully have the data sent from the mobile robot platform and have the information received at the operator’s base computer. The communication aspect of a search and rescue robot is perhaps one of the most important factors of its completion because without a proper means of communication, there is no way for the data, video, or any other type of information to be sent back and forth between the base computer and the robot itself, which would not be very conducive to a successful search and rescue operation, or any other type of application where such a robot could prove to be useful. Our decision was to use RF as a means of communication, which will be discussed later in the Design Details section. It was loosely based on this project, as well as many others, and as we have discovered, was much simpler to implement than other types of communications protocols.

**3.1.3 Autonomous Ball Collector (A.B.C)**

Another project that our group members investigated briefly was a project done by a group of University of Central Florida students back in 2010. Their project essentially was an autonomous robot which was used for seeking and collecting tennis balls on a tennis court. Though this project was not necessarily the direction in which we planned to go, we were still able to gain some inspiration from the work that this group had done. This project was certainly on the more software-intensive side, as there was quite a bit of artificial intelligence to program into this robot, as it had to determine whether it was collecting tennis balls, while also knowing where to position itself to do so, along with a variety of other functions that were performed autonomously. However, as previously mentioned, autonomy is not the route which our group planned to take, due to the overall complexity that this additional requirement would bring to our project. Additionally, the A.B.C. team fabricated their robot entirely from scratch, which again, is something we chose not be do, since this would have added significant cost and additional man-hours to our overall design, which we felt could be better spent on mission-critical items, such as the microcontroller or the user interface. The main inspiration we’ve drawn from this project was their in-depth analysis of various types of wireless communications. Just as they had done, we also have investigated into the possibility of using Radio Frequency, Bluetooth, IEEE standard 802.11 Wi-Fi, and even infrared wireless, which we will again be discussing in our section on communications. To implement their design from an electronics standpoint, they utilized an Atmel microcontroller. This worked to their advantage for a variety of reasons. First, their members had very limited programming ability. The fact that this microcontroller could be programmed in C meant that they could use the limited knowledge they had with the C language as opposed to having to learn a completely different language, which would ultimately (and ideally), save them time in that they wouldn’t have to struggle with learning a new programming language and could get to work in the controller straightaway. Additionally, they were able to find a variety of pre-written libraries that would be of great assistance to them once they actually began working with the controller to interface all of their components. They also used two additional Atmel microcontrollers to control the various other functions of their

vehicle and to simplify the design for their group members. We found that such an arrangement was not necessary for our robot, as the controller we used was sufficient to handle the things that we need it to do. More on that will be discussed in our section on microcontroller selection. After reviewing this project, our group members decided that this was in reality, quite different from what we were attempting to accomplish, and the amount of inspiration we were able to draw from this machine was actually quite limited. While we did gain a bit of insight as to how we could get our communications for our own project working, much of what we found from the project by this UCF group was unfortunately, not applicable for our purposes. For this reason, we felt the need to continue investigating into other projects and products.

**3.1.4 Other Available Products**

When we sat down to begin performing research for our project, we knew that we would definitely be able to draw inspiration from things already available on the market since search and rescue robots are not a new technology and they have been employed by various military branches, police departments, fire departments, as well as other agencies, for quite some time. As such, we found a vast source of knowledge by looking into these various applications of real-world applications of search and rescue robots.

**3.1.4.1 Unmanned Aerial Vehicles (UAVs)**

We are seeing rapid growth in the field of unmanned aviation, especially for military purposes. More and more, we are hearing reports of strategic enemy targets being eliminated by these unmanned aerial vehicles. Similarly to other autonomous robots, these can be pre-programmed to fly along certain paths. However, they are not completely autonomous, as they are for the most part controlled by a human operator located in a remote command center.



Figure 1. An image of an MQ-9 Reaper

## (Reprinted from Wikipedia under the Creative Commons Attribution-ShareAlike 3.0 Unported License)

## Figure 1 above is an image of a General Atomics MQ-9 Reaper, or a Predator B, as it is most commonly referred to. This UAV is perhaps one of the most recognizable UAVs on the market, as these are the Predator drones that are often alluded to in the news stories about targets having been eliminated. However, these drones were primarily first developed to aid intelligence agencies in their operations in gathering intelligence and other information about certain areas. Because these types of aircraft were originally developed for the same reconnaissance / search and rescue operations, we decided that some research into these aircraft would have some merit. While we did not develop anything near the scale or magnitude of one of these types of aircraft, it, in theory, operates under the same basic fundamental principles of our own search and rescue robot. For example, the Predator is remotely operated, and is equipped with various sensors and cameras, and is able to process and transmit this information back to the operator at the control post on the fly. This was similar to our own robot in that it performs the same basic functions of reconnaissance, and the user is able to see this data being transmitted from our robot in real time as well. While the underlying principles for how the UAV and the operator communicate wirelessly may be substantially more complex than what we intended to do for our robot, the same idea of controlling the device remotely was still there, and the same considerations had to be accounted for. For example, what kind of range would we look to have on such a device? What types of frequencies would we be operating with? Questions such as these had to be answered for the development of these unmanned aerial vehicles just the same as they had to be answered for our project. The very fact that this technology is still relatively new, and the fact that it has revolutionized the way reconnaissance, and even combat instructions are carried out, makes an incredibly strong case that this type of machinery is most certainly paving the way for the future. It was reassuring for our group members that the type of robot we designed has some real-world use, and that inspired us to ensure that the utmost care was taken throughout this process so that our machine could have some semblance to these incredible machines.

## 3.1.4.2 Kinect Robot

Engineers at the University of Warwick in Coventry, UK have accomplished some great and very remarkable things in the field of search and rescue robotics. Using a Microsoft Xbox Kinect, they have managed to implement an incredible, powerful search and rescue robot capable of mapping its environment and giving the operators a greater amount of flexibility in what they can do with the robot. For instance, using the Kinect module, the robot can create a 3-D model of the area and allow its operators to get a better feel for the environment and also assist in pointing out possible locations for survivors. With this robot, the engineers from the university won first place in the European Robocup Rescue championship in 2010. The most remarkable thing about this vehicle is that despite the level of detail and sophistication that went into its construction, it is still remotely operated by a human being, and was fairly close to what we chose to do with our project. The Kinect module acts as a camera of sorts, and also allows this robot to map the area it is in. With an accurate model of the area, rescuers can use the robot to search out survivors from the safety of a remote location much more effectively without needing to worry about risking their lives by entering a dangerous building with the possibility that there may be nobody there. Similar to what we created, this vehicle was not really meant for actually assisting in the removal of survivors from the disaster area, but is meant to serve a pivotal role in actually finding survivors and helping rescuers in searching for them. While we will did not use the camera mounted on our vehicle to create a three-dimensional map of the environment, it still served as our primary means of exploration and our primary means of detecting the presence (or lack thereof) of any survivors/victims. Additionally, their unit is equipped with an infrared camera for the detection of body heat as well as a CO2 sensor to detect the presence of expelled CO2 gas, which would indicate the presence of a living, breathing human being. We actually did consider both of these as viable additions for our robot in our initial planning stages, but we felt that neither was truly necessary, and we therefore decided not to proceed in purchasing these parts for our robot.

Even after investigating all of these projects and inventions, we were still left with many, many unanswered questions. How were we going to get all of this working together? How would we bring all of the hardware and software together to create a fully-functional robot? How could we control this robot wirelessly without the use of a remote control? What could we do to ensure that our search and rescue vehicle would work properly? These questions, along with dozens of others, were all questions that needed to be answered before we could finalize our design plans and actually begin looking into the construction and testing stages of our project. Fortunately, with the help of many internet sources, as well as the help of many skilled individuals, we were able to have most of these questions answered, or at the very least, we were given a little bit more knowledge than we had previously, which served as a great place to start.

**3.2 Relevant Technologies**

When we first sat down to begin researching things for our project, one of the largest factors that helped us determine the direction in which we wanted to move was the availability of hardware and knowledge about the hardware/software that we would need in order to complete the task that has been laid before us. Among the first things that we researched were forms of wireless communication.

**3.2.1 Wireless Protocols**

When we first decided on what we wanted to do with our project, we had absolutely no idea what possibilities there were on the market for us to use in terms of different communications. We were aware of the existence of things like Wi-Fi, and RF communication, but we had no idea what they really were or how they actually worked. This portion of our paper serves to delve a little deeper into each of the technologies we researched, and how they worked or could be incorporated into our design. We’ll first begin with an investigation of Wi-Fi.

**3.2.1.1 Wi-Fi (IEE Standard 802.11)**

Much of the beginning of our research stemmed from the little bit of wireless communications that we did know. We knew about Wi-Fi, and were aware that it was used in the creation of things such as local area networks (LANs), but we were unsure as to how exactly this technology worked, or how we would be able to utilize it in our design. After a great deal of digging and research, we were able to solve this mystery and gain a much more detailed understanding of how Wi-Fi works.

Wireless Fidelity, as it was first called, much like many other forms of wireless communication, functions by the way of radio waves. Simply put, Wi-Fi works in a manner very similar to the way in which two-way radios or walkie-talkies communicate. Essentially, a computer converts digital information into analog radio waves and sends this data to a router via an antenna located inside the computer or laptop. This data must then undergo analog-to-digital conversion at the router level, which is then transmitted to the internet via a wired Ethernet link down on the router level of things. This also must happen when data is being moved back from the internet through the router to the computer, but in the reverse order, with the digital to analog conversion taking place at the router, and then the analog to digital conversion taking place at the computer.

There are many flavors of the IEEE 802.11 standard, all of which have very different applications, ranging from use inside buildings or inside moving vehicles. The one that seemed like best fit for our search and rescue robot was the 802.11b standard. It is important to note that for long time, this was the least expensive standard to use, but also is the slowest of the 802.11 family, operating in the 2.4 GHz frequency band with a baud rate of about 11 megabits per second. While this clearly would not be a very good choice for a user’s main internet connection, something like this was definitely more than able to satisfy our purposes in getting this project completed. After further investigation, we discovered that we could even implement a Wi-Fi network using a simple embedded device that we’d be able to place right on our printed circuit board, which would save us a great deal of space, as well as help us minimize our costs, as these units are actually relatively inexpensive. Additionally, such a unit would be able to communicate with a computer via the serial port, which in turn would help simplify our design a bit.

We did investigate into some of the advantages and disadvantages of using 802.11 as our form of communication as demonstrated in Table 2 below.

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| Long range | Relatively Slow Data transfer rate |
| Low Cost | Slight signal interference possible |
| Signal not easily blocked | High Power Consumption |
|  | Network Configuration can be  Complex |

Table 2: Pros and Cons of 802.11

The fact that Wi-Fi modules do consume a great deal of power was somewhat of a cause for concern, as one of the major goals for our project was to minimize the power consumption for our robot as much as we possibly could. After researching this technology, it became clear to our group members that proceeding with Wi-Fi technology would most likely not be the way to go because of the power consumption requirement that we have imposed on this project. While Wi-Fi would serve as a great backup idea, we clearly needed to take some more time to gather a little more data before we could finalize our plans.

**3.2.1.2 Bluetooth**

Another great resource for us to investigate was Bluetooth technology. This is yet another wireless communication standard that could actually be quite useful for our project. Similar to 802.11, this standard also relies on the transmission of radio waves in order to send and receive data. The major difference is that where Wi-Fi operates on the basis of Wide Area Networking (WAN), the Bluetooth protocol operates on the concept of piconets, which are much smaller networks, as a Bluetooth module’s range is actually much less than that of Wi-Fi, at about ten meters in all directions from the device. Similar to Wi-Fi and many other technologies, Bluetooth operates in the 2.45 GHz frequency range. While this protocol does share the same frequency spectrum as many other types of devices and protocols, the way in which Bluetooth operates actually makes it quite robust. This protocol operates using what is known as spread-spectrum frequency hopping. This simply means that the Bluetooth transmitter will change frequencies many times per second to ensure that it does not interfere with other devices in the area. The big advantage to this, is that if the device does end up running on a frequency that is already being used by another device, it will not stay on that frequency long enough for it to cause interference, and even if any interference were to take place, its effect would be negligible.

The reason that such a device could be useful for our needs was that we could then use a wireless device to control our robot remotely. Because it would constantly be changing frequencies, there would be less of a need to worry about interference with the other components that we’ll be using, as we’d need to not only control our vehicle wirelessly, but we’d also need to ensure that the data we’d be measuring was being transmitted and received properly as well. Essentially, we would have a Bluetooth module connected to our vehicle, which would communicate with either the Bluetooth module embedded into the operator’s mobile phone, or even their laptop computer. From there, the operator could use either user interface we developed to manage and control the vehicle. From the research we have done, we have constructed Table 3 below, containing some of the advantages and disadvantages of Bluetooth.

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| Spread-spectrum frequency hopping | Very short range |
| Little radio wave interference | Relatively slow data transmission speeds |
| Very low power consumption | Pricing is not very cost-effective |
| Deployable across a wide range of devices |  |
| “Piconets” are easily configurable |  |

Table 3: Pros and Cons of Bluetooth

Though our research into Bluetooth technology offered us yet another great option for how we could have our communications operating, we still felt the need to look into other various means of communications. This in turn provided our group members with a much more broad knowledgebase so that we were able to make a better decision with what we felt would be the best method for communication for our project. The reason that this was such a difficult decision for us was that the way we chose to do communications for our robot would greatly influence the approach that we’d need to take to actually build and test our model. Next, we investigated the applications and utility of infrared communications as a possibility for our robot.

**3.2.1.3 Infrared Communications**

The reason that we looked into infrared as a means of communication for our robot was the fact that infrared is so popular in modern electronics. The infrared portion of the electromagnetic spectrum has countless applications in modern science and engineering, but we limited our investigation and discussion to its applications in the realm of communications.

The concept behind infrared (IR) communication is relatively straightforward. On the transmission side of things, an infrared-based light-emitting diode gives off infrared radiation. This radiation is then focused into a small beam that is modulated via a pulse width modulator in a way that the data being sent gets encoded using either the RC-5 or SICRC IR communication protocol. On the receiving end, the infrared energy is sent into a photodiode, which generates an electric current, which is then sent through the receiving device and decoded such that the data ends up in the right spot.

Unfortunately, the more research we did on this type of technology, the less and less it appeared that we would be able to use this as a viable method of communication for our project. For one thing, we discovered that due to the light-sensitive nature of infrared communications, it would not be a feasible means of communication for outdoor applications. This would be an issue for our search and rescue robot, as a majority of its use and testing occurs outdoors. This would, however, be perfect for something that would be used strictly for night time. Another disadvantage that we found is that devices operating via IR communications work on a “line of sight” basis. This means that in order for devices to communicate using infrared energy, there must be a direct path between the devices. As an example, this works perfectly for the remote control for a television, because there is a direct line of site between the remote control and the television while they are in the same room. If one were to attempt to change the channel on the television from another room, they would find that this clearly would not work, due to the lack of a direct line of sight. For applications like this, this is perfect because it limits interference significantly, as this infrared radiation is unable to penetrate walls or interact with other devices.

For our purposes, however, this would have proved rather ineffective. Because our intention was to have this vehicle controlled remotely from a location where a direct line of sight between the operator and the vehicle is most likely not an option, this option appeared to be rather infeasible. Despite all the advantages that IR could provide, it was very likely that we would not be able to use this for our project, and this proved to be true.

**3.3 Strategic Components**

One of the most difficult stages of the design process was actually choosing which components would be used. This was especially true with a group of senior design students, who all had very little experience in comparison with the components that engineers at large corporations have access to. That being said, we spent a great deal of time doing research on the kinds of components that we needed to complete our project. We found this research to be invaluable in that it taught us such a great deal about component selection for our own project. It also gave us insight into how this process is done in the industry, and we have all benefitted tremendously for having done this research together.

**3.3.1 Mobile Base**

After deciding that we wished to make a robot, we then needed to decide how exactly we would implement this robot. Would we create it all on our own? Would we design a chassis for it? Would we do all of the welding for the robot? How would we pay to make something like this turn into a reality? These were all questions that we had in our initial design meetings. After taking a look at other projects where the groups designed their own robot chassis, it seemed like a great idea, and would be an incredible accomplishment for our team members, but none of us had any experience with robot design. Our group members had neither the technical know-how, nor the tools and experience to make something like this happen. Additionally, in determining an initial budget for our project, this idea seemed rather costly. In addition to the other components we would need to buy for our robot, we would also have to worry about paying for parts and materials for the robot chassis as well. Due to our lack of experience, we would also more than likely need to pay some sort of costs for machining, along with many other related expenses. We agreed that this would not be a viable way of proceeding due to the massive costs that would be incurred from constructing the chassis alone. Also, due to the time constraints with which we were working, we decided that this in and of itself would take much too long for us to be able to complete our project on time. We then looked to other sources of inspiration for our project.

After dawning on the realization that creating our own robot from the ground up was much too risky from a financial and temporal standpoint, we came to the agreement that this simply would not be how we would be proceed. It would simply complicate things, making it much more expensive, and would not leave us enough time to actually focus on getting the major aspects of the project finished. After leaving the idea of building our own robot behind, we decided that we would need to look into other alternatives for our project.

Because of the need for a mobile platform for search and rescue purposes, we then decided to look into using remote-controlled vehicles. There were an incredible amount of remote-controlled vehicles on the market, ranging from boats, to airplanes, to cars, to tanks, among a wide variety of others. We decided that we’d like to have something that was ground-based, which therefore ruled out using a boat or airplane. We then needed to finalize what we were going to be working with. After more research and planning, we decided that it would be best if we just purchased a simple remote-controlled car from a manufacturer such as Mattel. This way, we were able to focus on the more important things such as communications, the sensors, and our user interface.

This train of thought allowed us to pick a method of implementing our vehicle. We decided to purchase a toy remote-controlled car, and removed its internal parts. Our original idea was such that once we finished, our printed circuit board, sensors, camera, and everything else would be able to fit inside the top part of the chassis and be moved by this mobile base of operation. We all came to the agreement that with this, we would be able to concentrate on the important parts of our project, which were on all of the electrical components and software. However, after receiving our boards and assessing all of the parts that we had, we vouched to make a custom platform to hold everything and scrap the idea of hiding everything underneath the chassis.

**3.3.2 Surface Mount vs. Through-Hole**

While investigating components, one of the largest factors was determining which would be the right parts for us. We needed to investigate the advantages and disadvantages of through-hole and surface mount components. The reason for this is that most manufacturers make quite a few of their parts in both styles, so it was important for us to know which would be of more benefit for us when we created our PCB.

After careful consideration and further research, we discovered quite a few things about the differences between surface mount technology and through-hole technology. For example, the advent of surface mount technology in the 1980s has caused the size of electronics to decrease incredibly. The fact that surface mount components are many times smaller than their through-hole counterparts means that the density of components on a printed circuit board can be increased dramatically, allowing for more components on a single board, and therefore vastly increases the overall capabilities of what can be done with a single board. In addition, we discovered that there a variety of other advantages of using surface mount parts over through-hole components. One example was that many surface mount parts actually dissipate slightly less power than through-hole parts. Also, many manufacturers make certain components only in a surface mount package. While there are many manufacturers who do make both surface mount and through-hole packages for their parts, the through-hole parts are starting to become a thing of the past as more and more circuit designers begin to go with surface mount parts. Apart from these few differences, however, we noticed not very many other differences between through-hole and surface mount technology. After much deliberation, we decided that we would do as much prototyping as possible with through-hole components due to the fact that they are much simpler to work with, as we have all had experience working with through-hole parts in various labs. For prototyping, the through-hole parts were perfect because they were easy to place and remove, and soldering and unsoldering

was so much easier as well, which made the prototyping process much easier for us. Once we were ready to begin constructing our finalized prototype for demonstration, we decided to stick with through-hole parts due to their wide availability around us. Both Radio Shack and Sky Craft (a hobby store located in Orlando, FL) only carried through-hole parts, so when it came time to buy capacitors and our crystal, etc, sticking with through-hole technology was the right move for us. We also had plenty of space to mount our parts since the amount of components needed was rather small which further helped our case for using through-hole parts.

**3.3.3 Microcontrollers**

Of course, perhaps the single most important component for any piece of embedded electronic equipment these days is the microcontroller. This piece of circuitry is actually what serves as the brain of our operation, as it is handling all of the data that we will be utilizing. It serves essentially as the command center for our robot. This is where all the programmed information is stored, and this information tells the controller what to do with the data being sent to it by the various sensors. It is the piece that communicates with the transmitters and receivers to get the data back and forth between our base computer and the vehicle. This component takes the information from the sensors, manipulates it, and turns it into something that will make sense for the search and rescue team operating our robot. The microcontroller also processes the commands coming from our base computer to control the vehicle’s movement. Needless to say, the microcontroller we chose to use has a significant amount to do with the success of our robot, and so it was imperative that we made this decision very wisely, as choosing the wrong controller for the job would lead to a catastrophic failure. More specific details about the selection process for our microcontroller will be provided in the microcontroller section of our hardware and software description.

**3.3.4 Camera**

Perhaps the second most important feature for our project was the selection of the camera. Whereas the microcontroller serves as the “brain” for our robot, our camera serves as the eyes and ears for the robot, and without a decent camera, clear vision would have been very difficult to achieve. The camera is what allows the operator of the robot to see the world as the robot sees it from a remote location. The camera is also what allows the operator to see where to guide the vehicle. That being said, it is clear that the camera was perhaps the second most important component for this project, if not just as important as the microcontroller. Because of the significance of the camera for our robot, we knew that great care needed to be taken in its selection because the camera, like the microcontroller, could either make or break this project. We therefore did quite a bit of research on cameras, ranging from small embedded surface-mount cameras, to the very expensive cameras with features including built-in rangefinders. After a great deal of research and healthy debate, we decided to stick with a small, relatively inexpensive IP camera that allowed us to transmit the data over a wireless internet network, instead of needing to purchase an additional transmitter/receiver pair for this as well. Due to the massive importance of this component, we felt that the research that we have done for this component was adequate enough for us to choose something that would provide us with a high level of success for this project.

**3.3.5 Sensors**

Another extremely important aspect of this robot’s design was the selection of the sensors. We spent a great deal of time researching into exactly which types of sensors we wanted to include in our robot. Would we be including an ultrasonic sensor to pick up on sounds in the area? Would we include in infrared camera? Would we be in need of a temperature sensor? We answered all of these questions after investigating various other similar projects. After a bit of discussion and further investigation, we decided that including an ultrasonic sensor was rather superfluous, given that the camera that we chose to use has a built-in microphone, which rendered an ultrasonic sensor unnecessary. After further investigation, we also decided that going with an infrared camera was not the greatest of ideas to follow through on. We decided this based on the fact that we were already dealing with tight spacing constraints in some areas, as well as having budgetary constraints. We therefore decided against purchasing one of these in favor of going with a more robust camera with quite a few more capabilities. We ended up deciding that we would limit the amount of sensors for our robot for a few reasons. First, we did not want to make this task overly complicated, because adding more and more sensors would greatly increase the complexity of the circuits we’d need to build, as well as add to the cost of our printed circuit board. We also simply just did not feel the need to overcrowd our PCB with dozens of unnecessary sensors, as again, this would drive up our costs because these additional sensors would give rise to the need for additional copper traces, which would have given lead to increased PCB costs.

We therefore decided on a few basic components that would help us get the job done. We decided that we would certainly be in need of a temperature sensor. This sensor helps our robot determine the ambient temperature of the surrounding area, which, in turn, also helps the operator to determine whether it would be safe for rescuers to enter the area to help seek out survivors. Much more detail will be given on how this particular component was chosen when we get to the Sensors subsection of our hardware details section.

In addition to the temperature sensor, we also decided to include a pressure sensor for the robot. The rationale behind this was that our vehicle would be able to detect an unsafe rise in air pressure in the area as the result of an explosion. While this would certainly be easy to see/hear with the use of the camera, it would also be helpful to have empirical data taken in the field to see the kind of effect this had on our vehicle. More detail on the actual selection for this item, as well as how it operates and how we have it connected to the microcontroller can be found in the Sensors subsection of the Hardware description.

We also decided that a rangefinder would be an essential piece of equipment to include on our robot. This was actually very useful for a variety of reasons. First, this has helped us determine how far away our vehicle is from potential survivors. Secondly, this also lets us know how far away from certain obstacles our vehicle is so that the operator can make the necessary adjustments when controlling the vehicle such as to ensure that the obstacles are avoided. We actually spent a great deal of time researching this, knowing it would play a major role in our operation and came to the conclusion that an ultrasonic rangefinder would suit our needs perfectly. As with the other sensors, a more detailed discussion of this part’s operation will be given in the Sensors subsection of the hardware description.

**3.3.6 RF Transmitters and Receivers**

In order for our robot to function properly, we needed some way of having the data, video, and commands from the computer to all be transmitted wirelessly. We looked into a great variety of technologies for this. We at last decided on Radio Frequency (RF) technology to help us accomplish our task. One of the major things we knew we would need was a way of transmitting data from the vehicle, as well as having a way of receiving that data as input to our base computer. This is how the operator is able to operate the vehicle, because this is how we are receiving our video feed from the robot. This was also the way we decided to control the vehicle, since we also sent data wireless to the vehicle as well in the form of commands to tell the robot which direction to move.

While the details of how we came to settle on radio frequency technology as the main means of communications will be discussed in our design section, it is important to note that we realized ahead of time that sorting out our communications proved to be a major challenge of getting our robot to work. We decided developing our own wireless communication protocol would be a senior design project in and of itself, and an incredibly difficult one, at that. We therefore opted to look into other existing technologies and decided that RF would be the way that we would like to proceed.

**3.3.7 Power Supplies**

Because one of our major goals for this project was to have our robot consume as little power as possible, it was only logical that we would consider our power supply to be a strategic component. Since this was what provides the necessary power for our vehicle, it was fitting that we took extra care in making this selection. Due to such a wide variation in the available power supplies on the market, we needed to do a great deal of research to find out which would be the best supply for our needs. The most difficult part of choosing our power supply was the fact that we weren’t very sure exactly how much voltage needed to be supplied to the various components as a whole. Thankfully, after studying dozens and dozens of components’ datasheets, we were able to determine how much power would be needed to be supplied to everything, and from there we were then able to do research into what kinds of power supplies to use, as there are an incredible variety of supplies available, and more work needed to be done to figure out which would be the best for our own applications. More information on this will be discussed in great detail in our design section.

**3.3.8 Base Computer**

Another incredibly important aspect of our project was choosing the base computer from which everything was going to be controlled. The reason this was important was that this was where the operator would be able to control the robot and tell it where to go and look. This was a relatively important part of our setup, as this was where the data collected by the robot was transmitted so that the operator had a good idea what was happening, as well as being able to see the various sensor readings as measured by our robot. This in turn would allow the operator to react as necessary in getting the proper resources deployed to the affected area. We decided that the best equipment for the job was to use one of our group members’ laptop computers, as this would ensure portability, as well as compatibility. One of the other factors that came into play in this making this decision was that we needed something that would work with all the parts that we chose to use. This meant that we needed something capable of communicating via a USB-to-serial port so that we could get the data to the computer wirelessly. Because none of us had a serial port on our laptop, we needed to look into purchasing a serial-to-USB adapter, which was both relatively simple to find, and fairly inexpensive. Once we decided on wireless communication, the XBee wireless module setup that we purchased included a USB-to-serial explorer that took care of this problem.

**3.3.9 Graphical User Interface**

While not necessarily a “component” in the physical sense of the word, another incredibly vital part of our project was deciding on the graphical user interface that we used to make the robot simpler to use. The reason that this was an important part of our robot was that this interface was how the operator would see the world as the robot sees it. This was where all of our sensor readings were sent, and was also where the video captured by our vehicle is displayed, which was what allowed us to issue commands and control it from our base computer. This was where we saw the temperature, pressure, and other sensor readings, and that was what allowed the operator to make the necessary decisions. We decided that the user interface would need to be detailed enough such that it would be user-friendly. This would allow just about anyone to operate the vehicle and make search and rescue operations more streamlined so that anyone could help out whenever necessary. Another functionality we wanted this robot to have was to have the ability to integrate captured video into the user interface somehow. We had seen similar projects where this was attempted, but was not accomplished successfully. We therefore took it upon ourselves to make this happen. Accomplishing this made everything a lot more streamlined, and as a result, was much nicer from an aesthetic point of view. Our graphical user interface will be discussed more in-depth in our discussion on interface in the design section of this document.

**3.3.10 Voltage and Current Regulation**

Another important thing for us to investigate was the idea of voltage regulation. How would we ensure that all of our components are getting the proper voltage? How would we ensure that no parts were drawing more current than their maximum rating? What types of technologies could we put in place to keep our parts from exceeding their rated values? The answer to these questions, quite simply, was that we would need to implement things that we have been learning about over the last few years that we have spent in school. We were faced with the issue of current and voltage regulation for our robot. We now had the chance to implement things like voltage dividers and voltage regulators in a real-world application. At our disposal, we had components such as voltage regulators and current limiter circuits. Using tools such as these, and the skills that we have honed over the years, we were able to ensure that all of our parts were operated within the limits of operation specified on their respective datasheets. We also used this knowledge to ensure that they dissipated as little power as possible while operating.

**4.0 Project Hardware and Software Design**

**4.1 Microcontroller (PIC16F886)**

The microcontroller we utilized for our project was one of the most important assets and had to be taken into careful consideration. The microcontroller we used essentially served as the “brain” of DZERV, since it read in all of the sensor data, and sent that data out to the transmitter to be sent to the base computer’s operator. It was set up to constantly transmit data to the computer’s receiver, as well as to receive data from the sensors. It had to be able to correctly tie to and read data from all of the electronic components that we utilized. The block diagram below shows the matter in which we utilized our microcontroller and how we interfaced it with our electronic components.

Figure 2: uC Block Diagram (Created by Group)

**4.1.1 Research Phase**

Initially, we were unsure as to what type or brand of microcontroller we would be utilizing for our project. The possibilities were endless, PIC, AVR, Freescale, Texas Instruments, Atmel, and the list went on. Before immediately choosing the first microcontroller that we came across, we had to contemplate a few things. What kind of programming experience did we hold, and with what language? How much memory did we need? Should we get an 8-bit, 16-bit, or 32-bit controller? How many pins would we need? Did we want a RISC or CISC CPU?

Upon further investigation, our group had reached some conclusions as to what type of microcontroller we would utilize. We had come to the conclusion that we wanted either an 8-bit or 16-bit RISC microcontroller which was easy to work with, had plenty of development tools and online support, and contained internal D/A conversion. We chose to go with a RISC architecture due to the fact that this architecture has a simpler instruction set, making it much easier for the compiler to write programs in. Aside from this, we had come to discover that the CISC architecture was beginning to become obsolete compared to RISC. We ultimately decided to go with an 8-bit microcontroller. We decided upon this due to the fact that such microcontrollers were better suited for a smaller application such as our robot, where we are more concerned about cost, size, and power, as opposed to ultimate precision.

**Texas Instruments MSP430**

Originally, we had decided on the 16-bit MSP430 microcontroller produced by Texas Instruments. We happened to stumble upon this microcontroller through our Senior Design course, which offered us an instruction session with a representative of TI regarding the MSP430. After attending this, we concluded this would be sufficient for our application as the top CPU speed was 25 MHz, supply voltage was 1.8 V to 3.6 V, and it had 51 instructions. Despite the specifications, one of the most attractive features of the MSP43 was price. You could purchase a development board for testing purposes for $4.30, and even had free access to use their Code Composure Studio IDE for writing code for the MSP 430. Although we were not allowed to utilize a development board in our final project, having such a feature would greatly simplify things, since we had the ability to test out components before permanently soldering them.

**PIC Microcontrollers**

Upon further research of the MSP430, we came to the realization that this microcontroller would not be as simple to implement with our sensors as we expected. For instance, we intended on using SPI to communicate with our sensors, as we had prior experience with SPI programming, but the MSP430 utilizes USART, which would be a learning curve for the group. Also, the MSP430 contained 51 instructions, which could eventually prove to be too complex for our application. Due to a group member’s prior experience with PIC microcontrollers, we began looking into this family.

We now had concluded that an 8-bit microcontroller would be better suited for our application. Using an 8-bit microcontroller proved to be easier to deal with and program, since we did not have to deal with large-numbered mathematics. We began to explore the PIC16 and PIC18 models, which were both 8-bit microcontrollers. We would reach our final decision based off of several factors, which included instruction sets, data space, and code space. We would also utilize a PDIP since it would be easier to test with a development board or breadboard, as well as be a lot easier to solder onto our final PCB. Table 4 below displays a few PIC microcontrollers which we took into consideration.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PIC Model** | **CPU Speed** | **RAM Bytes** | **EEPROM Bytes** | **Program Memory** | **Supply Voltage** |
| PIC16F886 | 5 MIPS | 368 | 256 | 14 KB | 2 V – 5.5 V |
| PIC16F877 | 5 MIPS | 368 | 256 | 14 KB | 2 V– 5.5 V |
| PIC16F690 | 5 MIPS | 256 | 256 | 7 KB | 2 V – 5.5 V |
| PIC16F73 | 5 MIPS | 192 | N/A | 7 KB | 2 V – 5.5 V |
| PIC18F2553 | 12 MIPS | 2,048 | 256 | 32 KB | 2 V – 5.5 V |
| PIC18F2680 | 10 MIPS | 3,328 | 1024 | 64 KB | 2 V – 5.5 V |
| PIC18F24J11 | 12 MIPS | 3,800 | N/A | 16 KB | 2 V – 3.6 V |
| PIC18F26J50 | 12 MIPS | 3,800 | N/A | 64 KB | 2 V - 3.6 V |

Table 4: Possible PIC MCUs

All of the PIC microcontrollers listed in the table above contained many of the features that we were looking for in a microcontroller such as a low operating voltage, CPU speed, and program memory. However, we took a closer look at how these PIC processors were programmed and decided to go with the popular Atmel brand of microcontrollers. Our ultimate decision was to choose the Atmel brand of processors to work with and the ATMega328P was the processor that we chose. We felt as if choosing this microcontroller would be sufficient for our application. Despite it having a small RAM and Program Memory size, we felt as if this smaller microcontroller would be simpler to work with and implement than any other.

**4.1.2 Atmel ATMega328P**

Ultimately, we chose to go with the ATMega328P produced by Atmel. We decided that this microcontroller was powerful enough and would be simple to work with and program. One of the major factors in choosing this microcontroller was the fact that it contained 131 single word instructions, most of which would not be needed for our project which in turn, greatly simplified programming since we only had three different sensors which send data to the chip. As well as this, this chip features 1 kilobyte of EEPROM data memory, and 8 different channels of 10-bit A/D conversion, which was perfect for handling the conversion in our sensor data. This chip also features a synchronous serial port which could be configured as a 3-wire Serial Peripheral Interface for communication between devices. This chip also features 28 pins, which was more than sufficient for us to tie all of our necessary components to it. Some of the other key features of this microcontroller include (Info taken from datasheet):

* 32 kHz to 20 MHz
* Power-Saving Sleep Mode
* In-Circuit Debug via two pins
* 23 I/O Pins
* Master Synchronous Serial Port module SPI mode, I2C mode capability
* 10-bit 8 channel Analog-to-Digital Converter
* Operating current: typical 5.2 mA
* Standby current: 1.2 mA at 5 V

**4.1.3 Design Phase**

Now that we have selected our microcontroller, we had to decide and design its configuration. Since we planned on using as small of a PCB as possible, we had to make sure to consolidate room on our chassis. Because of this, we planned on constructing one main PCB to house on our chassis. This main board housed the microcontroller, sensors, and extra parts such as resistors, capacitors, regulators, etc. Figure 3 below displays the schematic, which we implemented for our microcontroller wiring.

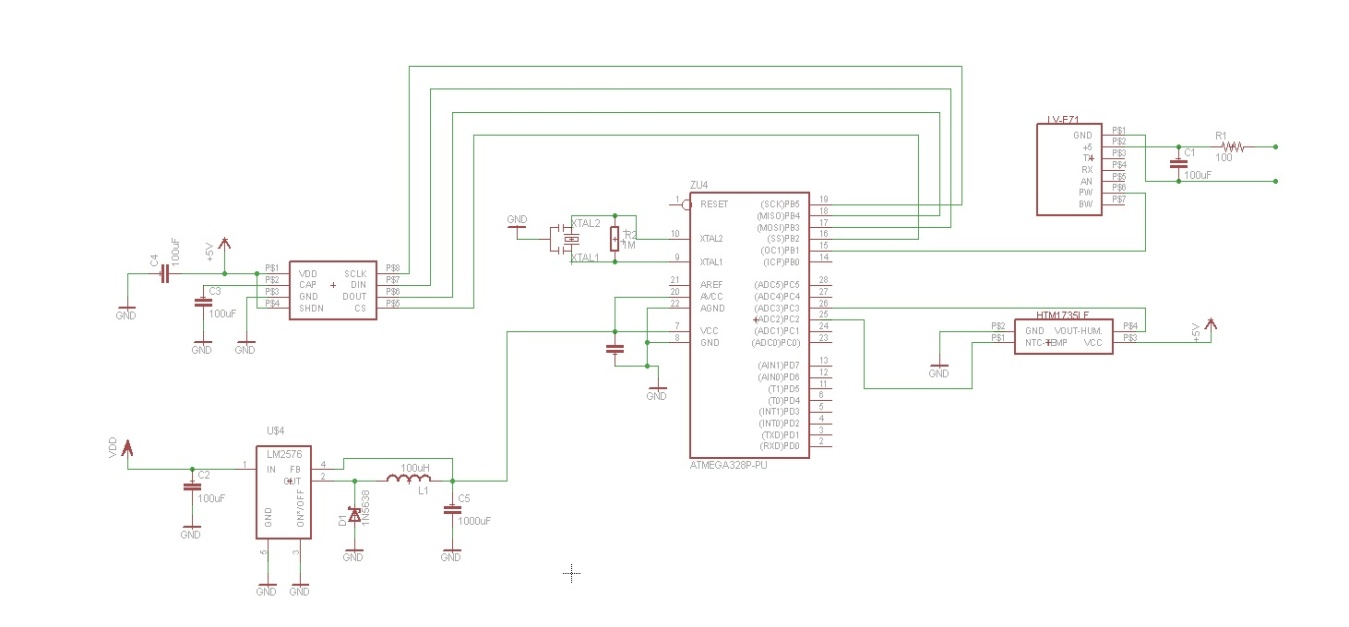


Figure 3: Microcontroller Schematic (Created by Group)

As you can tell from the figure above, regulators were not implemented in our initial schematic. The voltage regulation issue was later decided in the prototyping stages over the summer months. From the schematic, we also note that two of our sensors run into the channel A 10-bit A/D inputs of the microcontroller. These two sensors, the temperature/humidity sensors, and the range finder, output analog values to the microcontroller, which in turn are internally converted to digital values for it to process. The third sensor, the pressure sensor, communicates through serial peripheral interface. Utilizing SPI was not difficult since we only had one device on the bus. Therefore, we did not need to worry about the chip select lines or daisy chaining multiple devices onto the bus. A final feature which one will notice from our schematic, is the fact that we chose to utilize an external crystal oscillator, rather than relying on the microcontroller’s internal oscillator. The crystal oscillator we chose to use was the same frequency as the microcontroller’s internal one, 16 MHz. Going any higher could have overclocked the microcontroller and lead to serious potential problems with our schematic, so to be safe, we stuck with the 16 MHz oscillator. We chose to use an external one for one major reason. Using an external oscillator would prove to give us more precise measurements with exact timing, which would be crucial in a life or death situation. Aside from this, using an external crystal was simple and cost effective to implement.

**4.1.4 Programming**

Upon deciding which microcontroller to use and how to tie the sensors to it, our group began to think of the manner in which we would program out microcontroller to read information from the sensors and get it to the base computer. We decided that the manner in which we send data had to be both uniform and simple. This way, it was very easy for us to decode and convert to the values, and therefore, we did not run into any confusion. Once the microcontroller had read the 10-bit value from the sensor and stored it as an integer, we converted this value to a string and appended it to an array. The format of the array has one letter followed by a period, followed by letters representing a 10-bit integer holding the sensor information. Table 5 below gives a clearer idea of our method of transmitting sensor data.

|  |  |  |
| --- | --- | --- |
| **Sensor** | **Data** | **Output** |
| P (Pressure) | ABCD | P.ABCD |
| T (Temperature) | EFGH | T.EFGH |
| H (Humidity) | IJKL | H.IJKL |
| R (Range) | MNOP | R.MNOP |

Table 5: Microcontroller Sensor Output

Once we calculated these values, we utilized pin C6 on the microcontroller in order to serially send our data to the transmitter, which sent it to the receiver on the base computer where it was then decoded.

In order to program our microcontroller, we had two options to use assembly or the Arduino language. We chose to go with Arduino rather than assembly since it was very user friendly and there were pleny of examples online that we could learn from. After deciding to use the Arduino Uno to code our microcontroller, we had to search for a compiler and code editor. We came across Arduino Visual Studio and decided to use that.

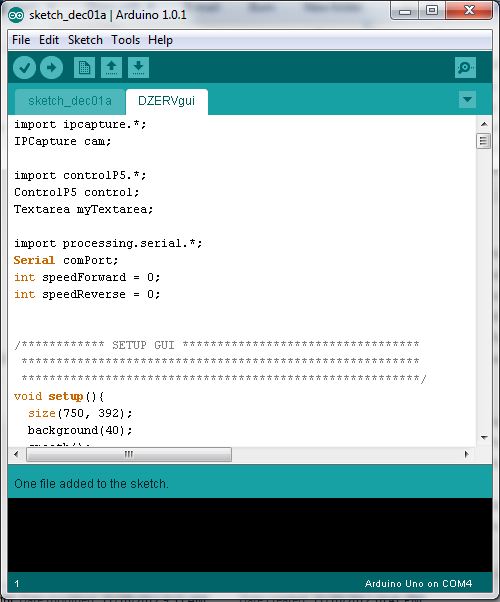


Figure 4: Arduino Visual Studio (Created by Group)

We chose to utilize MPLAB IDE for two major reasons. One, it was produced by Microchip and is specifically designed for PIC microcontrollers, and two, it was free to download and use. MPLAB also includes a C compiler for writing your code in C, which is something we wanted to do. This suite also included pre-made header files which would be utilized in our code. Using an IDE with such features will allow for simpler and more direct coding. MPLAB works like any other C compiler, and is simple to compile and run your code.

Now that we had chosen our microcontroller, language, and IDE, it time to begin writing code. The flow of our code was to be simple. First, we would initialize the pins which we would use, next, we would set high the pin to read, give a delay, read it, and then the data to our function to be converted into a string. The only sensor data which will be calculated in the microcontroller code rather than computing it on the base computer, is the pressure sensor data. This was due to this sensor having seven steps in calculating the final value, rather than using one or two simple linear equations as the other sensors utilized. The flowchart below depicts the manner in which our code will work.

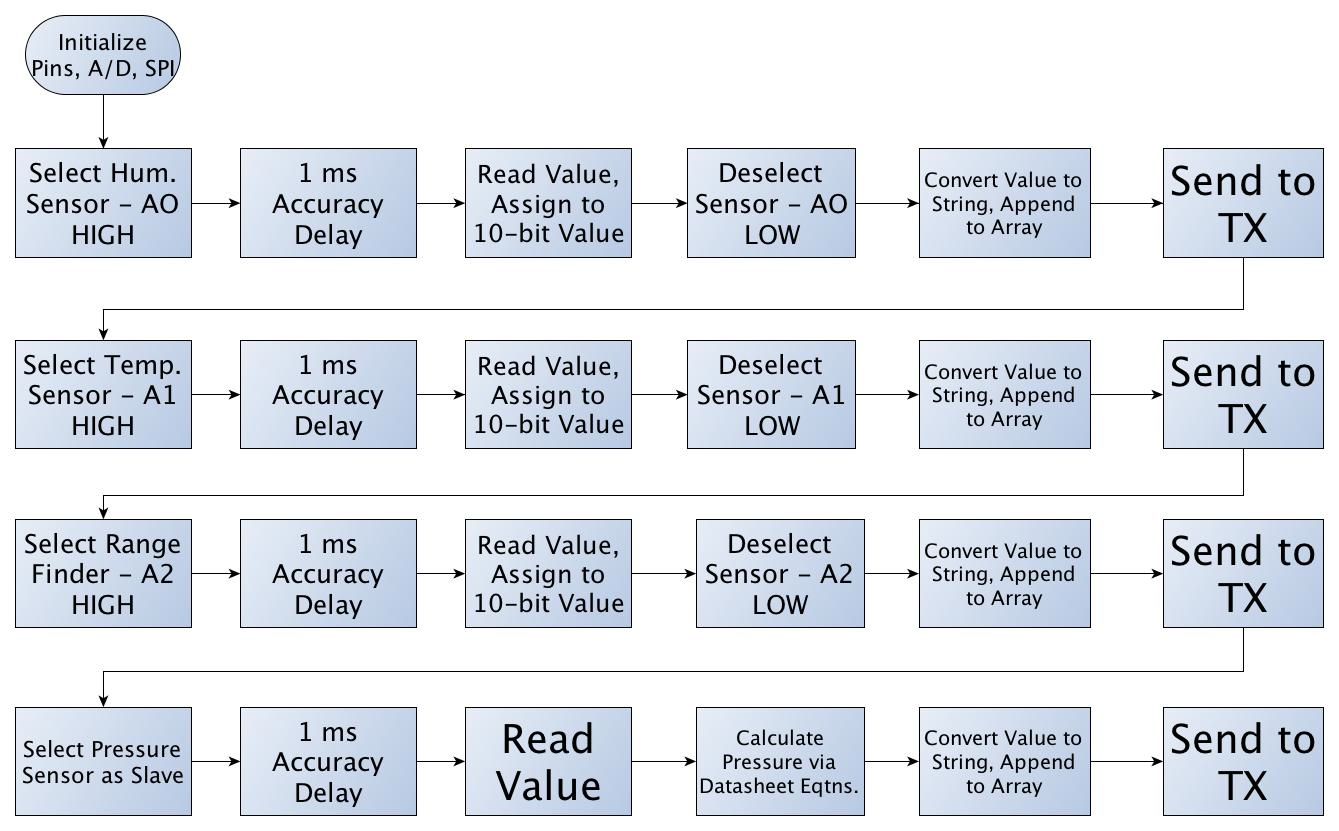


Figure 5: Microcontroller Flowchart (Created by Group)

In order to properly load our code onto our microcontroller, we used the Arduino Uno development board. Since we used bootloaded chips, we did not need an external programmer to directly connect to the microcontroller. The presence of a non-boot loaded chip would have forced us to purchase a programmer. Figure 6 below shows the Arduino Uno model that we used that we used to load code onto our chips. Additionally, we were able to use this board to test things.

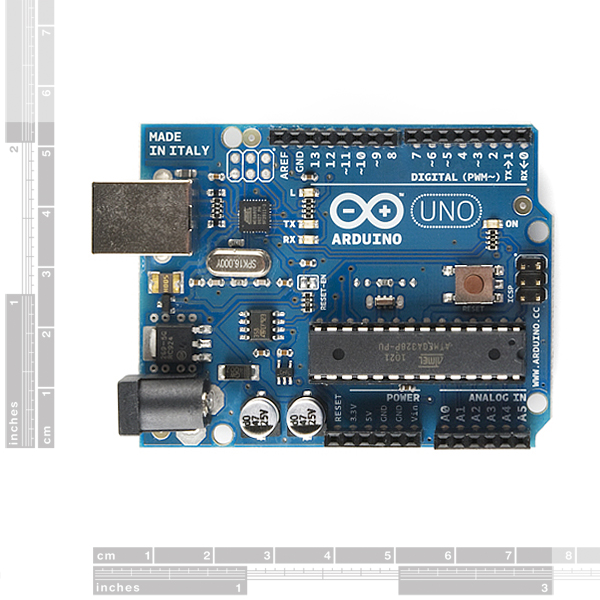


Figure 6: Arduino Uno Used To Program (Reprinted with permission from SparkFun)

**4.1.5 Testing/Debugging**

Testing and debugging were inevasible aspects of writing code for our microcontroller, or for any code in general. Because of this, we had to ensure that we had a thorough testing and debugging plan. We had to prepare for any possible errors or bugs we may have encountered, whether it be fro incorrect sensor readings, erroneous wiring, or part failure. In order to prepare for such risks, we had to create both an efficient and tactical plan to counteract any negative encounters. Because of this, we purchased two copies of our microcontroller. We purchased two through-hole versions of the microcontroller which we implemented on the board by purchasing a 28-pin socket to solder on our final PCB to allow for easy placement and removal of the microcontroller in case anything happened to the chip. Purchasing a through-hole part allowed us to create circuits on a breadboard with the microcontroller and sensors. This was greatly beneficial to us in the final design of our PCB. Once we had created and tested a circuit using the through-hole part, we simply placed the programmed microcontroller onto our final PCB.

In order to efficiently test and debug our microcontroller, we decided to use a development board rather than simply using a breadboard. We decided that using a development board was a lot more beneficial in testing, as it contained more features than a simple breadboard. With a development board, we had the ability to easily interface our transmitter, as well as our sensors. The development board we decided to utilize in our testing phase was the Arduino Uno development board sold by Sparkfun. This development board comes equipped with a +5 V and +3.3 voltage regulator, filtering capacitors, an external 16 MHz crystal oscillator, and a power plug-in jack. Figure 7 below depicts the original programmer we were going to use had we stuck with the PIC processor.

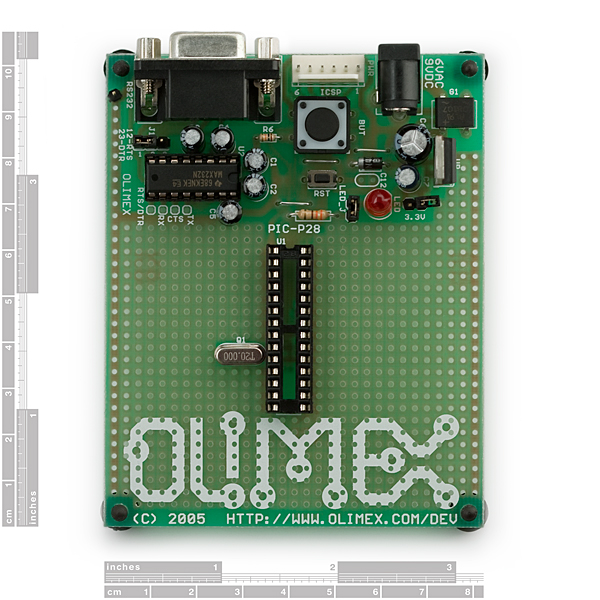


Figure 7: PIC Development Board (Reprinted with Permission of Sparkfun)

**4.2 Sensors/Instruments**

As our design stated, we will be implementing temperature, humidity, and barometric pressure sensors. Aside of these sensors, we will also be mounting an ultrasonic range finder onto our robot in order to avoid obstacle collisions. We will use one chip (HTM1735LF) to gather both temperature and humidity, one chip (MPL115A1) to gather barometric pressure readings, and one chip (LV-EZ1) as our range finder.

**4.2.1 Pressure Sensor (MPLII5A1)**

The pressure sensor we chose for our application is the Miniature SPI Digital Barometer, part number MPL115A1, produced by Freescale semiconductor. The sensor will be purchased from Sparkfun Electronics for $11.95. This sensor was selected based on its characteristics such as output voltage and current, accuracy, and conversion time.

Another sensor which we took into serious consideration was the MS5561 produced by Servoflo. It was extremely similar to the chosen sensor in the sense that it contained an internal A/D converter and operated with a 3-wire serial interface. Ultimately, we chose the MPL115A1 over the Servoflo sensor due to the fact that the Servoflo sensor was almost double the price of the MPL115A1.

Another major reason that this specific sensor was selected was due to the fact that it utilized the SPI bus to send output temperature and humidity readings. This was useful to us because the ATMega328P provided three pins specifically dedicated to SPI, and since we only used one SPI device, it integrated rather well. Also, by utilizing the SPI bus, we were able to free up lines dedicated for analog to digital conversion. The sensor provides a reading of 50 kPa to 115 kPa absolute pressure with a 1 kPa accuracy, which was more than enough for our application. The sensor runs on a 2.375 V to 5.5 V power supply and also includes an integrated ADC. One of the few downsides with this selected sensor is the fact that it is light sensitive. If we have direct light exposure to the port hole, we ran into the possibility of having flawed measurements. After a series of tests, we determined that the measurements we got were not hindered by the light and therefore decided to keep the sensor open on our PCB.

The matter in which we will connect the sensor to our microcontroller is as follows: SLCK to PB5, DIN to PB3, and DOUT to PB4. We also utilized the CHIP SELECT and connected that to pin PB2 on our microcontroller. This was because we only had one slave and one master in our SPI interface, so we didn’t have any need to switch between multiple devices. The SHDN line was only grounded when we wanted the device disabled, and since we decided to constantly refresh data from the sensor, we simply connected it to the power supply connection, VDD. The figure below depicts the manner in which we connected our sensor to our microcontroller.

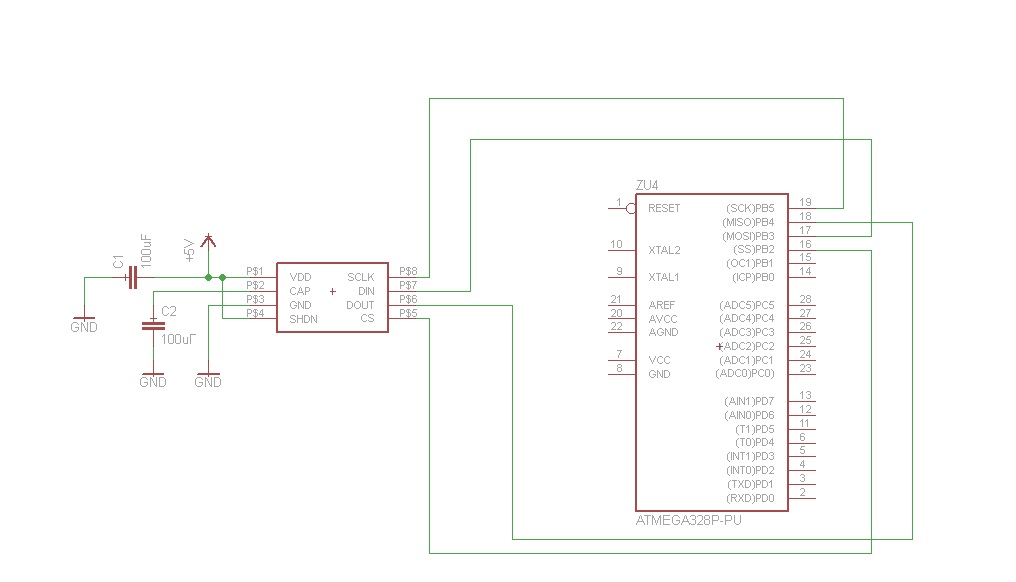


Figure 8: MPL115A1 Connection to MCU (Created by Group)

As we can see from the internal block diagram below, there is an internal A/D converter built into the sensor we chose, thus simplifying calculations and allowed us to avoid the use of bus lines or switches to connect devices to our microcontroller. The 10-bit compensated pressure output, which the sensor calculates, is calculated based on the following equation:

*P = a0 + (b1 + c12\*Tadc) \* Padc + b2 \* Tadc*

Where:

Padc is the 10-bit pressure output of the sensor A/D converter

Tadc is the 10-bit temperature output of the sensor A/D converter

a0 is the pressure offset coefficient

b1 is the pressure sensitivity coefficient

b2 is the temperature coefficient of offset

c12 is the temperature coefficient of sensitivity

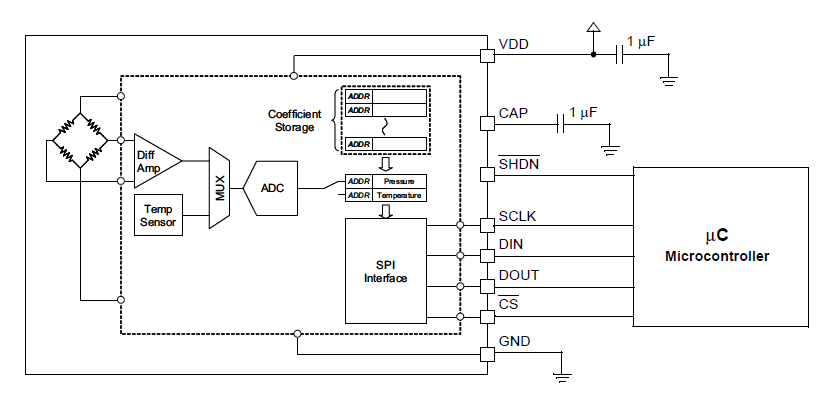


Figure 9: Pressure Sensor Internal Block Diagram (Copyright of Freescale, Inc. 2012, Used by Permission)

All of these coefficients come from the coefficient address map as provided in the datasheet. Once we had this equation solved, we could solve for this pressure in kPa according to the following equation:

*Pressure(kPa) = Pcomp \* + 50*

**4.2.2 Temperature and Humidity Sensor (HTM1735LF)**

The temperature and humidity sensor which we chose for our application was the HTM1735LF sensor produced by Measurement Specialties. We purchased the sensor from Digi-Key, where it is sold for $25.95. Originally, we had chosen the RHT-03 sensor produced by Maxdetect. The RHT-03 was originally chosen due to the fact that it had extremely low power consumption, could run on a power supply as low as 3.3 V, and only used a 1-wire bus for communication between it and our MCU. Despite these characteristics, we ran into some trouble with using this sensor. Since DZERV should be capable of handling harsh and possible dangerous environments, it had to be equipped with sensors to withstand such conditions. The RHT-03 was not suitable for such harsh conditions, and its internal parts or sensitivity could have received interference due to the exposure of chemical materials.

As previously stated, DZERV carries sensors which are capable of being placed under potentially hazardous situations. The chosen sensor had been tested under harsh chemical conditions, therefore making it capable of withstanding acidic, salty, or other extreme chemical conditions. Aside from this, our sensor was not light sensitive, which was a major benefit in the design of DERV. Due to the sensor not being light sensitive, we could have an external location for the sensor, thus assuring accurate results. The temperature operating range is -40 to 100 °C and humidity range is 0 to 100% RH, thus making it more than sufficient for our application.

Direct interface with our Atmega microcontroller was made possible with the module’s humidity linear voltage and direct NTC outputs for temperatures. This sensor measures humidity with respect to voltage based on two linear equations:

*Vout = 25.68RH + 1079*

*RH = 0.03892 Vout – 41.98*

|  |  |
| --- | --- |
| **RH (%)** | **Vout (mV)** |
| 10 | 1325 |
| 20 | 1600 |
| 30 | 1860 |
| 40 | 2110 |
| 50 | 2360 |
| 60 | 2605 |
| 70 | 2860 |
| 80 | 3125 |
| 90 | 3405 |

Table 5: Humidity to Voltage

Figure 10: Voltage vs. Humidity Graph of Temp/Humidity Sensor (Created by Group)

The table above represents some sample data, which we could encounter upon deploying DZERV in various situations. Upon graphing this sample data, we can see that the output is a linear curve, as was expected. Our sensor outputs temperature according to the following equation:

= \*

Where:

RT is the NTC resistance measured in ohms at temperature T in K

RN is the NTC resistance measured in ohms at rated temperature T in K

T, TN is the temperature in K

is the beta value, material specific constant of NTC

e is the base of natural logarithm (2.71828)

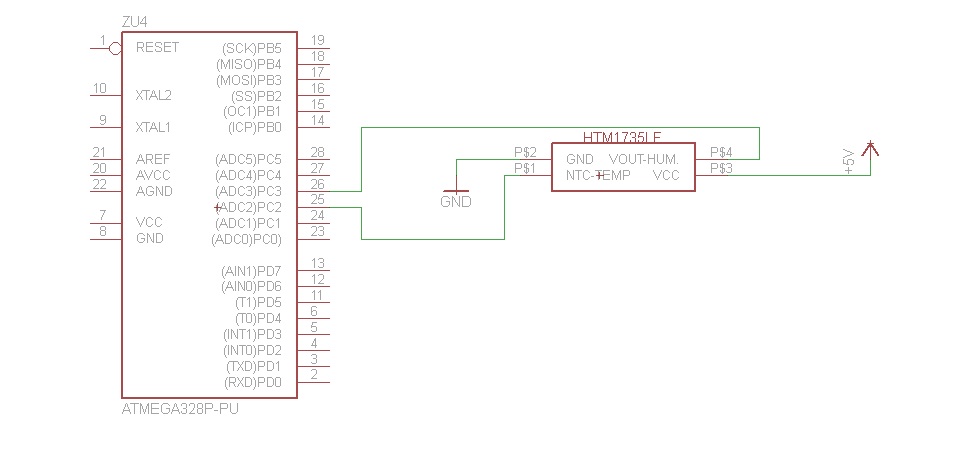


Figure 11: Temp/Humidity Sensor Connection to MCU (Created by Group)

In the figure above, we can see the connection of the sensor to our Atmega microcontroller. The connection is as follows: Vcc is connected to a 5 V power supply, GND is grounded, NTC-Temp is connected to analog input ADC2, and Vout-Humidity is connected to analog input ADC3. The maximum output voltage that the Vout-Humidity pin sends is 3.6 V. The maximum input voltage that our Atmega microcontroller can take in is 5 V, therefore, we did not need to utilize a voltage divider to step down the voltage.

**4.2.3 Ultrasonic Range Finder (LV-EZ1)**

Obstacle avoidance and collision was a substantial aspect to take into consideration with our robot. Not only did we want to avoid any potentially dangerous objects in the robot’s path, but we also wanted to protect our product and make sure that it returned intact. Due to this, we chose to integrate an ultrasonic range finder. We decided to go with an ultrasonic range finder versus an infrared one due to the fact that we are using the robot outside, which causes an infrared rangefinder to be deemed practically useless, since it will not accurately work in the sunlight without going through a lot of extra work. Also, ultrasonic range finders are more accurate and project a wider beam to detect obstacles. The only downside of using an ultrasonic range finder was that they do not accurately detect absorbent materials such as sponges and they are more costly than infrared technology.

For our application, we chose the LV-EZ1 produced by Maxbotix. We purchased the device from Sparkfun Electronics for $25.95. We chose this range finder due its sonar range of 6-inches to 254-inches, which provided us with very short to long-range detection of obstacles. The rangefinders offered by Maxbotix come in five different beam width sizes, EZ0 through EZ4. The EZ0 model has both the widest and longest beam of detection, while the EZ4 has the smallest. As we can see from the figure below, the field of view for our range finder is in an elliptical form. The figure below also depicts how the different rod sizes and beam widths affect the overall ability to see objects in the robot’s peripheral vision and how far ahead it detects objects.

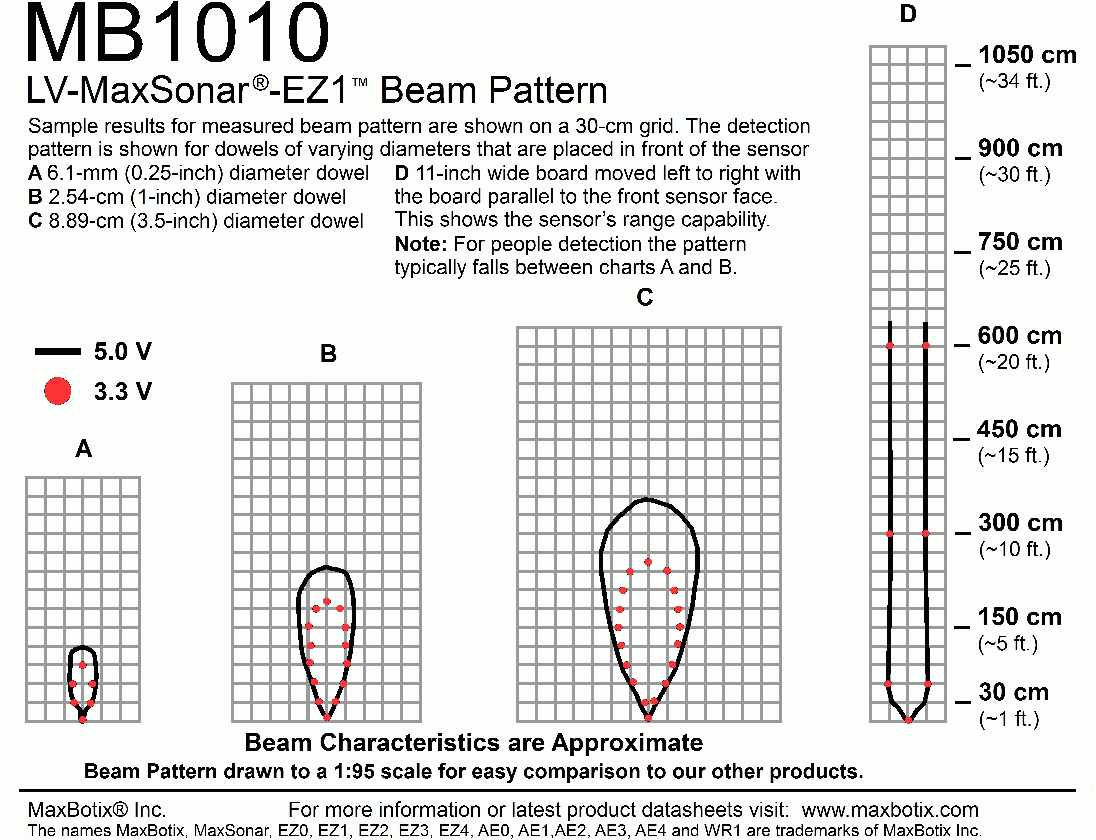


Figure 12: Range Finder Beam Pattern (Reprinted with permission of MaxBotix)

We also had the option of interfacing our output in three separate formats: pulse width, analog voltage, or serial digital output. For our application, we chose to go with the analog voltage output, since it was the simplest to deal with and interface with our microcontroller. In fact, by using the analog voltage output method, we only utilized three out of the seven pins on the device, with only one line running into the microcontroller. Also, readings from this device could occur up to every 50 ms, which was more than sufficient for our application. The only negative aspect of this device was that the project beam is a bit narrow, thus not being able to identify objects in the robot’s peripheral vision.

By supplying the LV-EZ1 range finder with a supply voltage of +5 V and utilizing the analog voltage output, our range finder yields approximately 9.8 mV/in. Since our device provides sonar range from 6-inches to 254-inches, our analog voltage output transmits a value of 0.058 V to 2.48 V. From the datasheet, we knew that the device produced approximately 9.8 mV per inches with the supply voltage as 5 V with a 9-bit factor. However, since our Atmega microcontroller uses 10-bits for A/D conversion, we had to make use of a separate equation to properly measure distance from objects. The following equation is what provides us with the distance:

*(Total inches of capable sensitivity/total a-d value)\*2\*(sensed value)*

From the datasheet, we knew that our max capability of sensing was 254 inches, and having a 10-bit A/D conversion, 210 is equal to 1024. Therefore our final equation for total inches sensed isas follows:

*\* 2 \*(sensed value) = actual inches sensed*

Wiring the LV-EZ1 range finder to our microcontroller was a rather simple process. We only have one line running from the device to the microcontroller, the PW pin will run to digital input pin PB1 on the MCU. Also, in order to provide superior results, our line had to be free of any electrical noise. In order to accomplish this, we placed a resistor in series with a capacitor at the input power pin. The figure below is a schematic of the manner in which we wired our range finder.

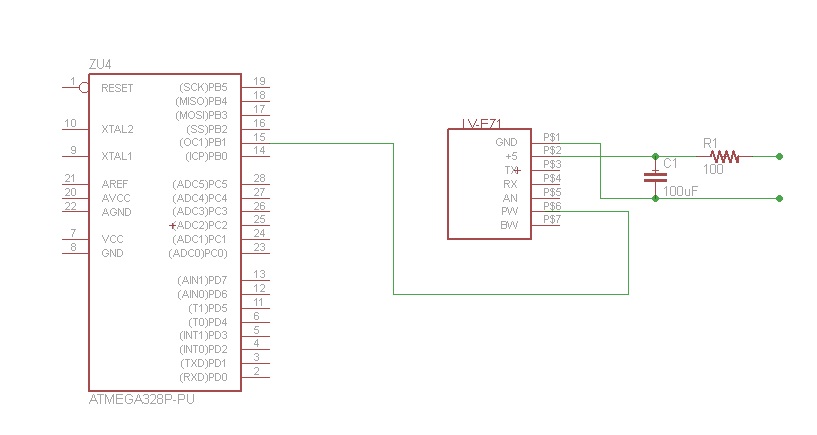


Figure 13: Range Finder Connection to MCU (Created by Group)

**4.2.4 Dark-Detecting Circuit**

Since DZERV is designed to operate under potentially hazardous conditions, the situation may arise where it could encounter poor lighting, whether it be sent on a mission during nightfall, or into a rubbled building in which visibility is poor, therefore, it had to be equipped to handle such situations. Due to this, we designed a dark-detecting circuit which was capable of detecting darkness and turned on an LED only when necessary in order to conserve energy. Originally, we planned on purchasing an LED light bar from Sparkfun Electronics. However, we realized that the light bar provided way too much light which interfered with our camera. To combat this problem, we decided to go with the dark-detecting circuit which drives one LED as opposed to many and it still provided us with the sufficient lighting that we needed. We found many possible schematics for this type of circuit and ultimately decided to implement a very basic one using very few components.

The photosensitive element that we implemented in order to detect darkness was a phototransistor, type LTR-4206E produced by Lite-On Electronics. The phototransistor that we chose is an infrared-sensitive type with a dark lens. The only downside of this was that it did not recognize fluorescent or discharge lamps, but rather only sunlight and incandescent lights. We also implemented one transistor in our circuit in order to turn on our LED at proper time. We decided to use a 10mm White Super Bright LED, part number NTE30071, which we purchased from Radioshack. We decided be power our circuit with with 5 V from our main power supply, which provided us with the miniumum 4.5 V that it needed to operate. The LED that we picked has a forward Voltage of 3.3 V and diode rated current of 25 mA, therefore, we calculated our load resistors to be about 100 Ω. The figure below is the schematic of our circuit which we mounted onto the front of our robot.

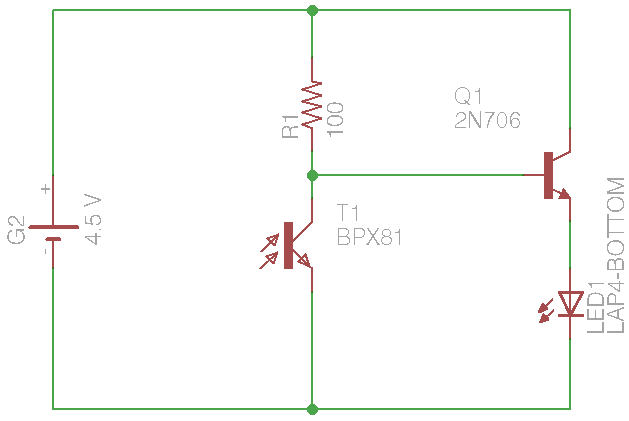


Figure 14: Dark-Detecting LED Circuit (Created by Group)

**4.3 Camera (AirSight Wireless IP Camera)**

In order to produce a superior search and rescue robot, we had to have some means of visual navigation. A robot which cannot be visually guided by the base operator would be of no use for our application. Due to this, we had to find some sort of means of transmitting video back to the operator’s base computer, and we chose to utilize a camera to do so. We decided our camera would have to be affordable, reliable, sturdy, and able to transmit a decent quality video stream.

Originally, we thought that going with a color sensor module with a digital output would be our safest bet, as it was a cheaper alternative to buying a camera. We had found the C3038 color sensor module sold on Electronics123. However, we soon realized that utilizing a color sensor module would prove to be both more costly, and more complicated, as we would have to interface it with the microcontroller and originate a server client program in order to transmit video to the base computer.

After eliminating the idea of utilizing a color sensor module, we went with our only other option, purchasing a ready-to-go camera. Originally, we had decided on purchasing the Planetcam kit produced by x10. X10 produces a kit for $129.99 which includes a wireless PC transceiver, motion sensor, USB video capture adapter, control software, and the camera itself. This camera seemed as if it would be ideal for us, it was weather resistant, sturdy, small, and simple to mount. Also, it already came packaged with a USB adapter in order to transmit the video straight to the machine and software to control the camera. Such items would greatly simplify the idea of controlling our robot with a camera. The only problem we ran into with this camera was that it did not have an ability to pan and tilt, which would only give us a straight line of sight, thus dampening the efficiency of our robot.

Upon erasing the idea of the x10 PlanetCam, we stumbled upon IP cameras. By utilizing wireless internet, we could directly communicate with and control our camera using wireless internet with no extra adapters. According to IEEE standards, 802.11b and 802.11g WAPs and routers support a range of up to 150 feet indoors and 300 feet outdoors, which could have worked for our project. However, we chose to go with the Linksys wireless IP camera (Model WVC54GC) that we found used on Amazon for $40. The camera came packaged with the camera itself and the software in order to be able to stream the video.

The Linksys camera came equipped with an integrated web server built into it. We saw this feature as being extremely beneficial to our application. The pan and tilt of the camera is 270 degrees horizontally, and 120 degrees vertically, however, we decided that the vision provided by the camera was great even without having to pan and tilt it. The camera package also contained software that allowed us to achieve streaming capabilities very easily. The resolution for this camera is only 320x240 pixels, which isn’t amazing, but the quality was good enough for us to determine if obstacles were in our way. The camera also comes equipped with a 5 V barrel jack that did not require a power outlet to operate, which was great for us because DZERV is designed to be completely mobile.



Figure 15: Linksys IP Camera (Reprinted with Permission of X10)

As we can see from the figure above, the camera is rather small, which was perfect for our smaller chassis that we utilized for our robot. One of the major benefits of this camera was the fact that it only required a 5 V power supply instead of the full 12 V power supply, which allowed us to save our battery life. The only problem we faced with utilizing this camera was that it is not designed to withstand harsh outdoor conditions. Due to this, we wanted to create a small clear plastic housing for it, but we simply ran out of time. The housing would not have covered the lens, as to not interfere with video quality, but it would have been there to protect the camera from outdoor conditions such as rain or dust.



Figure 16: Camera Software (Screenshot taken from own computer)

Another major benefit of this camera was that it already came with its own client-server GUI software, which we simply installed on the base computer. A screenshot of the software can be seen in the figure above. As we can see from this screenshot, the software brings full control of the camera, giving us the ability to control everything from the pan and tilt features to the video resolution. By utilizing this camera and software, it made it very simple for someone with either no engineering or real technological knowledge to operate the camera and the robot itself. The only problem with the software for this camera was the fact that we initially ran into problems incorporating the stream with the GUI module which we created for the sensor and motor data. However, after some time, we were able to resolve this issue and only have one screen open that featured all of our sensor and motor data, with most of the screen showcasing our video feed so we always had a nice representation of where our robot was.

**4.3.1 MAX232 Chip**

Since we were reading incoming sensor data through the serial port of our base computer, we had to have some method of converting the digital input into signals suitable for our RS-232 serial port. In order to accomplish this, we looked into building a separate PCB which would connect to the base computer incorporating a MAX232 IC produced by Maxim Electronics. This IC could be purchased from Mouser Electronics for $6.11. This chip would provide RS-232 voltage levels from ±3 V to ±15 V from a +5 V power supply to the chip itself. The table below describes the logic of the MAX232 chip. Whenever it receives a logic 0 or 1, it changes it to the appropriate voltage.

|  |  |
| --- | --- |
| RS232 Line | RS232 Voltage (V) |
| Rx/Tx Logic 0 | +3 to +15 |
| Rx/Tx Logic 1 | -3 to -15 |
| RTS/CTS/DTR/DSR Logic 0 | -3 to -15 |
| RTS/CTS/DTR/DSR Logic 1 | +3 to +15 |

Table 6: RS232 Outputs

In order to properly receive the TTL signals being produced by the microcontroller, we would also have had to incorporate a RXM-916-ES transmitter and the MAX232 chip on the same PCB which would connect to the serial port. We would have had to connect our MAX232 chip into a DB9 232 connector. The pins that would have been utilized for our PCB would be pins 2, 3, and 5. Pin 2 was the RD (Receive Data), pin 3 was the TD (Transmit Data), and pin 5 was for GND.

In order to properly pass data to the base computer, the MAX232 chip had to receive the correct incoming signals. In order to achieve this, it had to be wired to the RXM-916-ES RF receiver. Essentially, the Atmega microcontroller would have to send sensor data to the TXM-916-ES, which would then feed that data to the RXM-916-ES receiver. Upon receiving that digital data, it would be passed to the MAX232 chip to transform into the correct analog voltage to be fed into the base computer. The figure below displays a bock diagram of how these modules would interact.

Figure 17: Sensor Data Flow

Both the MAX232 and RXM-916-ES receiver required only a +5 V power supply, which simplified our PCB design greatly. After purchasing all of the components for this design, we ultimately decided to scrap it due to the fact that we all decided that working with XBee wireless modules would be a lot easier to deal with and since each component was a transceiver in itself, this made our lives a lot easier. We simply ran into lots of issues using the separate antennas for transmitting and receiving. The original RF receiver was a little harder to deal with since it required three connections and two groundings. Pin 1 (ANT) would have ran to the 50-ohm RF input antenna. Pins 2 and 4 (GND) would have been grounded, Pin 5 (Vcc) would have been connected to our +5 V power supply, and pin 12 (DATA) was our digital data output, which would in turn run into our MAX 232. The pins labeled NC on the RXM-916-ES in the figure below did not make an electrical connection, but would have been soldered for physical support only. The MAX232 chip also called for a lot more components. It required wiring the power supply pins, five external capacitors, and a separate connection to the RF receiver. The schematic below displays the manner in which we originally wired our RXM-916-ES RF receiver to our MAX232 chip.

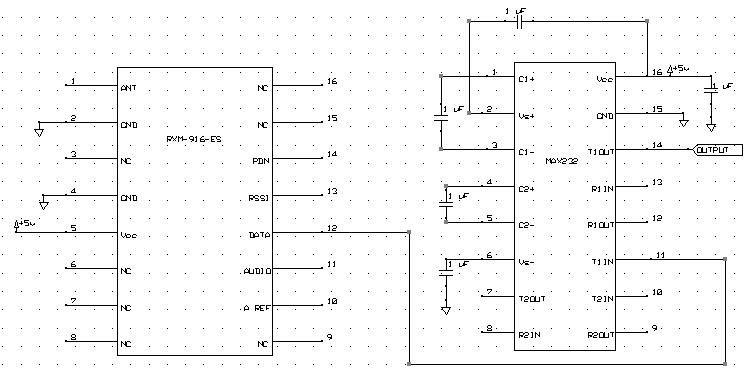


Figure 18: RF RX to MAX232 Wiring (Created by Group)

**4.4 Software/Interface**

Figure 19: Software Flowchart (Created by Group)

The figure above is a flowchart of the way in which our software operates. The user will first go through the proper setup of connecting the receiver box to the base computer and powering on the robot. After this, they will open the camera software in order to obtain visual data. Once this initial setup has been complete, they will open the DZERV.exe icon, which starts the program. Once the serial port has been opened to receive incoming sensor data, the user then has the option of navigating the robot via the keyboard arrows or an onscreen GUI, which also contains arrows. While all of this is occurring, the Atmega microcontroller sends out sensor data to the transmitter, which in turn sends it to the receiver and all of this occurs in the background. Upon receiving the sensor data on the base computer, it is then written to a text file with a time and date stamp. The data in this text file has a refresh rate of about 100 ms in order to ensure up-to-date readings. A sample of what the text file looks like can be seen in the figure below.

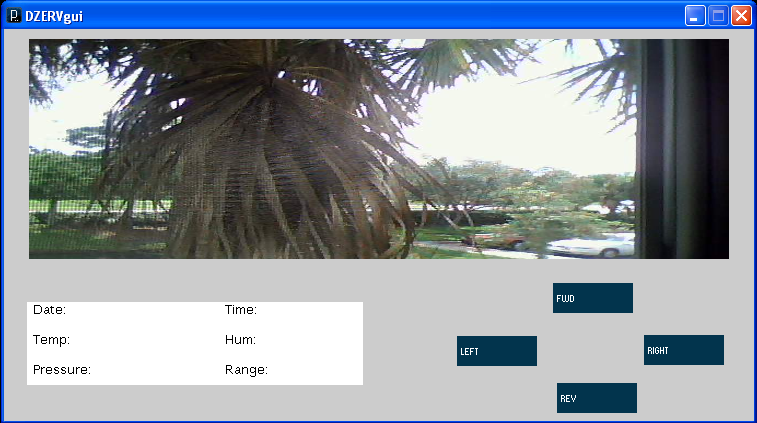


Figure 20: Display GUI (Created by Group)

This portion of the application was coded using Visual Basic since it was a very straightforward language to use. Due to it being straightforward and simple to grasp, the group members with electrical engineering backgrounds could work on the coding as well. The Visual Basic code was written to take in the “X.ABCD” formatted strings from the serial port connected to the receiver. Once our code received this string, it deciphered and arranged the data according to temperature, humidity, and distance. Once we had this information, the code would execute the proper equations to arrive at our sensor values, which were then printed to the GUI described above.

**4.4.1 GUI**

The graphical user interface, which we created, had to be designed in order to allow a non-technical user operating the base computer to utilize the robot. This meant that it had to be simple and easy to use. The user will begin by opening all necessary software as stated in the previous section. After this initial setup, the program will monitor the serial port for new data, and write this data to a text file at a refresh rate of 100 ms. Such a process occurs in a continuous loop, or until the user decides to terminate the program. Our GUI software was coded in an object oriented fashion implementing the Processing programming language.

**openFile Class**

This class is the main component of our GUI program. It serves as a means for the operator on the base computer to open and select the log file which contains all of the sensor data. It contains the following members.

* **private JButton openButton, saveButton:** These buttons allow the user to open a directory to search for the log file, as well as allowing a user to permanently save the log file to their machine for future use or record keeping.
* **private JTextArea logArea:** This serves as a logging area where the program lets the user know what is currently happening with their file. It could have logs such as: “Opening: “ +file.getName() + “.” + newline”, “Saving: “+file.getName() + “.” + newline”, “Couldn’t fine file: “+path”, and “Save command cancelled by operator”.
* **private JFileChooser chooser:** This is a Java API which made it easy to bring up open and save dialogs for the operator on the base computer.

The openFile class contains the following methods:

* **public openFile:** Here we created our log area, as well as created and laid out our buttons using FlowLayout.
* **public void actionPerformed(ActionEvent e):** This method handles the actions of when the user clicks the open and save buttons. If the user clicks “Open a File”, it handles opening the file, as well as handles the saving of a file if the user clicks “Save a File”.
* **private static ImageIcone createIcon(String filePath):** This method returns a Java ImageIcon or null if the file path provided by the user is invalid.
* **private static void createDisplay():** This method creates our GUI and displays it to the user.

Below, we can see a screenshot of our class diagrams which displays the classes utilized as well as the variables and methods located within each of these classes.

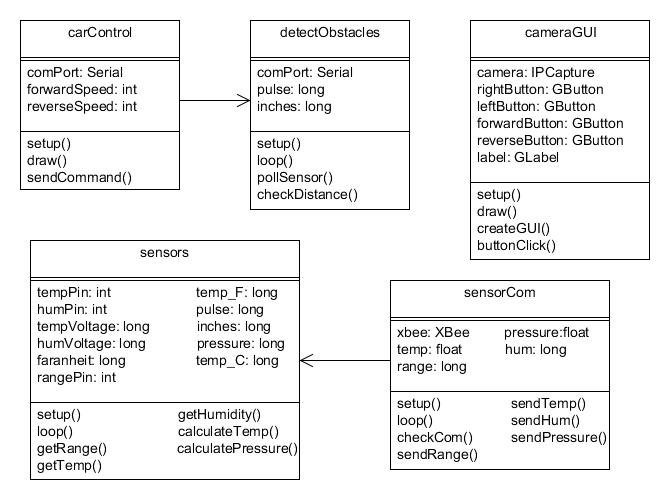


Figure 21: Class Diagrams (Created by Group)

**4.4.2 Robot Control**

As previously mentioned, the base operator has the ability to control the robot by using the arrow keys from the keyboard. It was designed having the user’s discretion in mind as to which control they wanted to implement. In order to use the arrow keys, a user may press any combination of the arrows in order to produce a result as seen in the table below.

|  |  |
| --- | --- |
| **Command** | **Result** |
| ↑ | Move robot straight forward/Increase Speed |
| ↓ | Move the robot straight backward/Decrease Speed |
| → | Move the robot right |
| ← | Move the robot left |
| B or b | Brake/Reduce Speed |
| E or e | Emergency Brake |

Table 7: Wireles Control

In order to control our robot from the base computer, we decided it would be easier and more efficient to utilize a wireless module versus utilizing the computer’s serial port. While utilizing a wireless module was less cost effective, it required less components to tie to the base computer. In order to accomplish this wireless manuevering of our robot, we utilized the XBee wireless modules, a motor controller, and the Processing open source programming language. We bought two XBee modules and received 1 additional module from a friend, one to implement as a transmitter from the base computer, and two to implement as receivers mounted on the robot. One receiver mounted on the robot is connected to the motor controller, which in turn is connected to the motors of our robot. The code we wrote is sent to the transmitter via USB, which is then wirelessly sent to our receiver in order to control our car. The figure below depicts the manner in which our wireless control of the robot works.

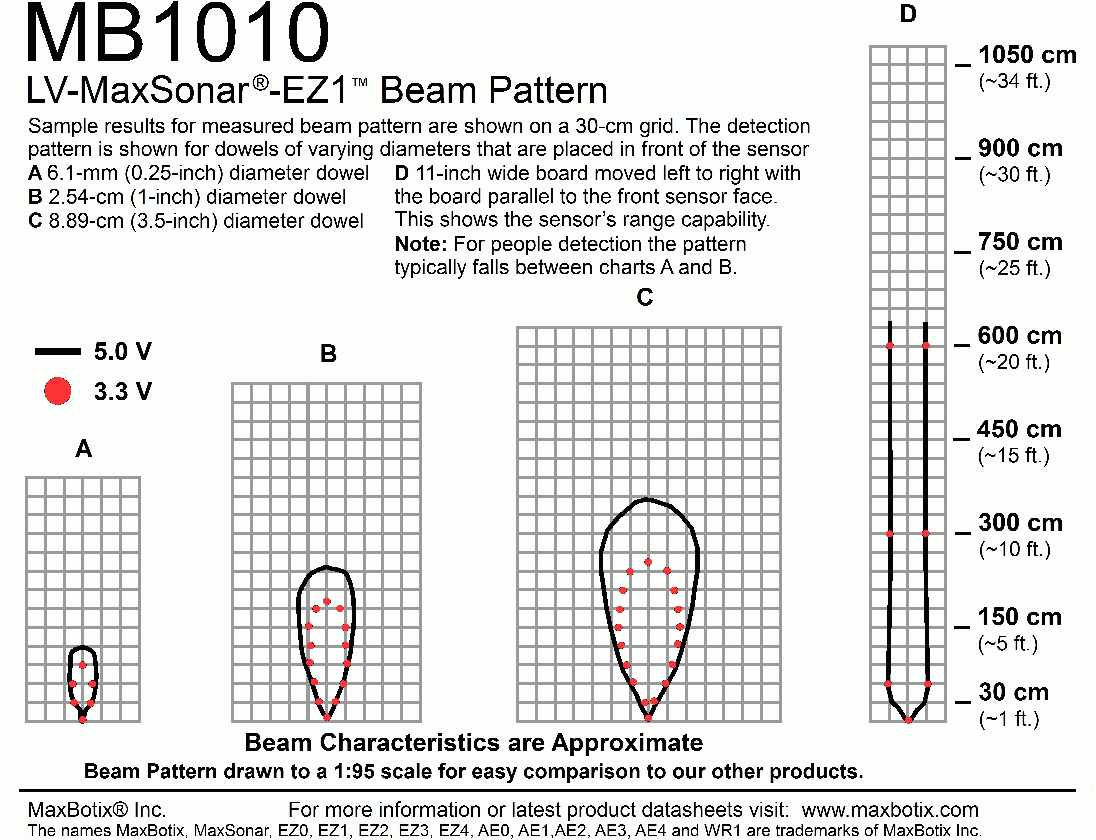


Figure 22: Wireless Control (Created by Group, XBee and Controller Reprinted with Permission of Sparkfun)

For our coding environment and language, we decided to utilize a free open source language called Processing. The structure and syntax of Processing was very similar to the language utilized by Arduino, thus making it rather simple and straightforward to use. Since we were communicating with a communication port to send code to our XBee transmitter, we had to use the serial library provided by Processing. In utilizing this library, our code consisted of if statements which read user inputs, and also write() functions which send data to the communication port. Once the code was completed, it was then converted into a .exe file, which allows the user to simply click the icon to begin maneuvering the robot. The flowchart below depicts the manner in which our code acts.

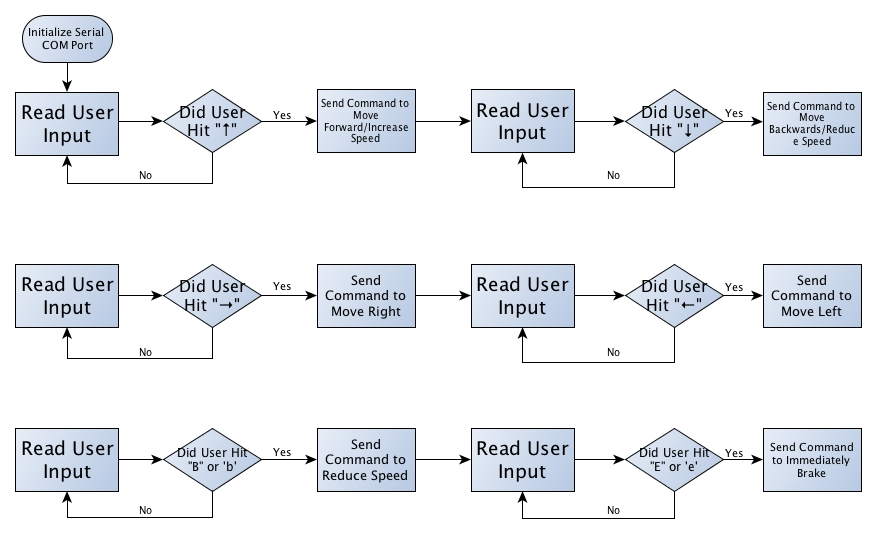


Figure 23: Robot Control Flow Chart (Created by Group)

**4.5 Voltage Regulators**

Because the power supply that we decided to use was actually much higher than the rated voltages for a majority of the parts we used for our robot, it was certainly obvious that we needed to do something to maintain the voltage in our circuits at a manageable level for the components. This could have been achieved in a variety of ways. For instance, we could have used voltage divider networks of resistors to do this. However, due to the amount of components for our robot for which this type of voltage regulation needed, attempting to regulate the voltage in this way would not have been very feasible, as this would have resulted in the need for many more resistors than would likely fit onto our PCB. It therefore became necessary for us to investigate the use of voltage regulators as our means of limiting the voltage flowing into our devices.

Due to the nearly limitless variety of voltage regulators on the market, we needed to conduct further research to see which would be the best for our needs. Would we go with Texas Instruments parts? Would we rather go with components from Linear Technology? would we instead decide to use switching regulators, or would it be best to stick to the linear voltage regulators with which we’ve become familiar from use in the lab? These were all questions that needed to be answered before we could even begin to design a finalized circuit schematic for our PCB.

In order to simplify the process a little, we have constructed a table of the parts which were of the most concern to us in maintaining proper regulation, as these were all very much mission-critical components. Failure to meet these requirements would result in a catastrophic loss for our robot, and it was therefore imperative that we ensured that we stayed within the voltage ranges set forth by these components’ datasheets. If these criteria were not met, our parts would have burned out rather quickly.

|  |  |
| --- | --- |
| Part | Operating Voltage Range |
| Atmega Microcontroller | 1.8 V – 5.5 V |
| Pressure Sensor | 2.375 V – 5.5 V |
| Temperature/Humidity Sensor | 4.75 V – 5 V |
| Ultrasonic Range Finder | 2.5 V – 5 V |
| XBee Transmitter | 2.1 V – 3.6 V |
| XBee Receiver | 2.1 V – 3.6 V |

Table 8: Parts to Keep in Regulation

From this table, it is fairly clear to see that we needed to two separate regulators: one for 5 V and one for 3.3V. With this in mind, we could then begin looking into different Possibilities for voltage regulators. Fortunately, we already had experience with some regulators from Texas Instruments, and we were able to begin our investigation there. However, before we began looking into specific parts, it was necessary for us to investigate whether we would be using switching regulators as opposed to linear voltage regulators.

**4.5.1 Linear Regulators vs. Switching Regulators**

Before we could decide what type of regulators we’d be using for our circuits, we first needed to familiarize ourselves with the differences between the various types of voltage regulators on the market. We could have went with the standard linear regulators with which we are familiar or we could have utilized something like a switching regulator.

Linear voltage regulators work in a fairly simple manner. The difference between the input voltage and the rated output regulated voltage is dissipated across the semiconductor material of the regulator as heat. While this makes for very simple circuitry to work with, and while these types of regulators are much cheaper, it is very important to note that the way in which these regulators actually regulate the voltage is uite inefficient. The equation for power dissipated across one of these types of regulators s the following: P­reg  = (Vin - Vout) \* Iload, where Preg is the power dissipated through the regulator, (Vin - Vout) is the difference between the input voltage and regulated output voltage, and Iload is the current through whatever device may be connected to the output of the regulator. As an example, if a 12-volt power supply were to be connected to a 3.3-volt regulator, which was in turn connected to the power input pin of a microcontroller that sinks 5 mA of current, this would mean that 43.5 mW of power would be dissipated across the voltage regulator. While this is not an issue, since this is a rather low power dissipation and well within the operating limits of most linear regulators, this can become quite a problem when dealing with components that sink much more current, as the power dissipation would grow rather rapidly. Depending on the thermal resistivity of the semiconductor material within the regulator, as well as a few other factors, a very small increase in power could lead to a very rapid rise in temperature across the regulator. This in turn could cause the part to cease its normal functions, which could then lead to major problems in the circuitry to which it is attached.

With this in mind, however, we did note that a vast majority of our components are low-powered enough that these types of issues would not arise, and in the event that they did, we would be able to resolve them fairly easily by adding a simple heat sink to ensure that our parts did not get destroyed. Armed with this knowledge of linear voltage regulators, we were then ready to begin looking at components that could be useful in our design. Before doing so, however, we realized that it would also be necessary to investigate all possibilities for voltage regulation for our device, as this was the only way we would truly be able to know which option would be the best to suit our needs.

In addition to linear voltage regulators, we also investigated into the properties of switching regulators, or switch-mode regulators or DC/DC converters, as they are commonly called. These devices work much in the same manner as their linear counterparts, but in many cases their efficiency is nearly double that of linear voltage regulators. Through our research, we had noticed some regulators with an efficiency as high as 87 percent. This type of efficiency would work wonders for our robot, as these types of regulators are typically used in applications such as robotics. Additionally, these types of regulators are also great in applications where a great deal of heat would cause issues, as these regulators are designed to generate very little thermal energy while in operation. They are commonly used as a means for voltage regulation in applications where battery power is the main source of power for the circuit, and are popular for such applications as a result of their amazing capacity for efficiency.

Switching voltage regulators rely very heavily on the concept of pulse width modulation. There are many various types of switching regulators, such as step-down (Buck) regulators, step-up (boost) regulators, and even inverting regulators, which simply invert the voltage placed on the input pin. While the last two types of regulators have countless uses in modern electronic technology, for the purposes of our own project, we would need to stick with a Buck regulator, which takes an input voltage and regulates it down to a lower voltage. Figure 24 below shows a common setup for one of these types of regulators.

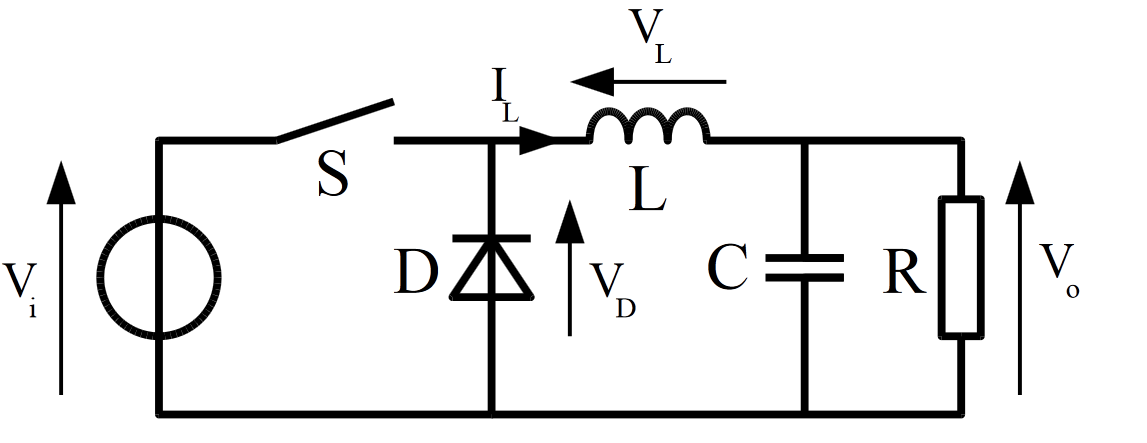


Figure 24: Buck Regulator (Reprinted with Permission of Wikipedia)

The way this works, is that there is a bipolar junction transistor connected to a PWM controller, which causes the transistor to act as a switch. The latter half of the circuit includes an RLC network in which the inductor is then connected directly to the input. When the switch closes, the inductor then dumps current into the capacitor and the load, causing the capacitor to charge up. Once the PWM controller causes the switch to open once again, the current through the inductor cannot change instantaneously. Therefore, the voltage across the inductor must change to compensate for this to hold the current at a constant value. This continuous cycle is what causes the output voltage to remain steady at a fixed value, depending on the values for inductance, input voltage, and various other electrical characteristics for the circuit.

Because all of this is typically included in a single integrated circuit, implementation into our design would be relatively simple to accomplish, and was therefore another option that we had considered. Armed with the knowledge and insight into the operation of linear and switching voltage regulators we have gained, we were then prepared to finally begin our investigation of components for use in our robot. We began with an investigation into various linear voltage regulators.

**4.5.2 Linear Regulator Options**

Based on our experiences taking the Electronics II class at the University of Central Florida, we knew that looking into National Semiconductor’s LM7800 series of voltage regulators was a great place to start, since our group members have had experience creating regulated circuits with these components. With a dropout voltage of 2 volts, this regulator would actually be more than capable to suit our needs, since we decided to use a 12 volt power supply. Fortunately, we discovered that we would be able to purchase this component in a through-hole package. While we would have been able to use this in the event that we were unable to settle on another device, we strongly believed that this would not be the greatest idea. We did decide to keep the 7805 regulator on our list of contingencies though, as this would certainly suit our needs. This particular regulator could be purchased for $1.95 from SparkFun. Figure 25 below demonstrates how we would have had this particular regulator connected to some of our components.



Figure 25: LM7805 Regulator connected to some of our components

In addition to seeking a surface mount component for our project, we also decided that we would look for something more efficient as well, since efficiency was one of the major constraints with which we were working with on this project. The less power dissipated by all our parts, the more efficient our final design would be. With this in mind, we expanded our search to include the LM2936 from National Semiconductor (now a part of Texas Instruments). This regulator offered a number of advantages for us.

One nice specification of this regulator was that it is offered in a surface mount Small Outline Integrated Circuit (SOIC) package, featuring eight pins, of which two are used, four are ground pins, and the remaining two pins are no connect pins. Based on researching this component’s datasheet, it was clear that this part would be more than able to do the job for us, and it would be a great option for us to go with. It has a very low quiescent current value, which was perfect because this enabled the regulator to behave more like an ideal regulator, in which the quiescent current is zero. This regulator would maintain regulation at 5 V for our parts, and even include reverse battery protection in the form of a diode connected between the input and output pins in order to prevent current from flowing back through the part and causing damage to our power supply. This in and of itself was a huge advantage as this would simplify our design a little bit in that we no longer had to account for this reverse current in our design. Additionally, it ensured that the life of our power supply would be preserved as much as possible.

Another major drawback to this voltage regulator was the price. At $2.44 per regulator from Digi-Key, these components proved to be a little expensive, since we needed to purchase a few of them to ensure that all our parts were receiving voltages within their range of operation. Also, as with the 7800-series regulators, one major issue we would have had with this regulator was its lack of efficiency. Because of the fact that this was also a linear voltage regulator, it operates by the same principles as the 7800 series regulators, meaning that even with just a very small amount of voltage regulation, the power dissipation could be incredibly high, which could cause a wide variety of problems for us. These types of issues could have ranged from desoldering the part from the board to the part simply not working at alland thus not being able to prove the proper regulation we needed it to provide, which in turn would severely damage the other components on our PCB. With this in mind, we continued our search for a voltage regulation solution. For a rough idea of how this would look if we decided toimplement it into our circuit, figure 26 shows how this regulator would be connected to some of our components.

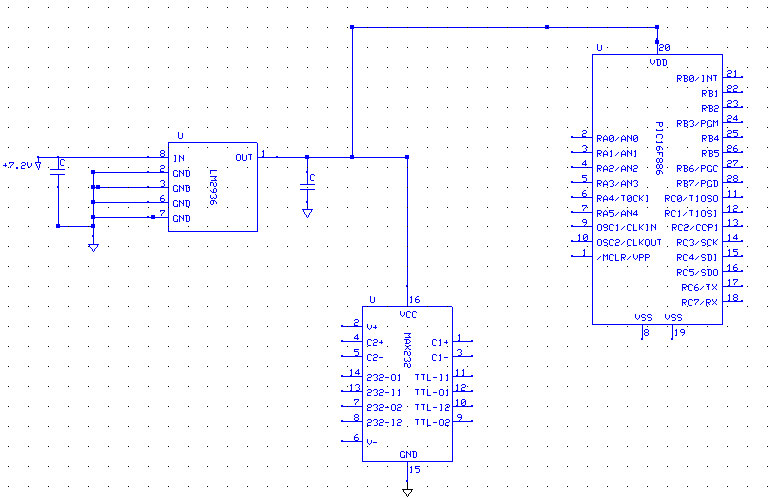


Figure 26: Example of the LM2936 connected to some of our components

As seen in figure 26 above, the regulator is connected to the input voltage source, and its output is connected to the other devices requiring voltage regulation. The capacitors to ground are in place to ensure that the signal is kept clean, and the other five pins are connected directly to ground. If this was an adjustable regulator, other resistors and capacitors would have been implemented in the circuit as well.

Another part that we had considered was the LT1121-5 from Linear Technology. Being a very widely-known manufacturer of IC components such as these, we decided to search Linear Technology’s extensive list of parts to see whether they had something that could suit our needs. We ended up investigating the 1121-5, which was another linear 5 volt regulator that we could have used in our design. One of the reasons that we looked into this particular product was that Marcial had some experience with seeing the adjustable version of this regulator in action from a project that he had been working on with his employer. Because of this experience, we found that this regulator would also have been a good choice for us.

This device has a very low dropout voltage between .13 V and .42 V, depending on the input current. Again, this ensures that we’d be able to maintain the necessary regulation for our parts. Additionally, and perhaps the largest advantage of this, is that this component was available as both a through-hole part and a surface-mount part, which meant that we’d be able to use this regulator for both prototyping, as well as our final design. This regulator also has a very low quiescent current value, which means that this also will act nearly ideal, and will ensure that our parts are well-protected. The only source of concern for us in using this part, really, was cost. This part was available for $3.92 from Digi-Key, which is slightly expensive, but was in fact, less expensive than the LM2936 regulator. Despite these options, we decided that using switching regulators was the right way to go. We were offered so much better efficiency and the extra cost for these parts was well worth the price. The switching regulators dealt with heat much better than the linear regulators which is another reason why we ultimately decided to use switching regulators. Therefore, further investigation of switching regulators was needed

for us to find the right voltage regulator for our project.

**4.5.3 Switching Regulators**

The first switching regulator we investigated was the LTC3632 regulator from Linear Technology. This regulator is a Buck regulator designed to regulate down to 5 V. The part’s datasheet actually gives an example of how this particular part should be connected, which actually proved to be quite useful. This component comes in a convenient 8-pin Mini Small Outline Package (MSOP), which fits our needs for size. Additionally, we would be able to utilize the example circuitry from the datasheet in order to ensure this component regulates to 5 volts, so this eliminated the need for guesswork and additional calculations on our part, which simplified the process for us even more. The largest advantage that this would have given us was the fact that Buck regulators are known for their power-saving capabilities. This would have allowed us to extend the life of our robot’s battery and enable it to run for a longer amount of time. The regulator would be connected to our circuit in a similar manner to what is seen in figure 27 below.

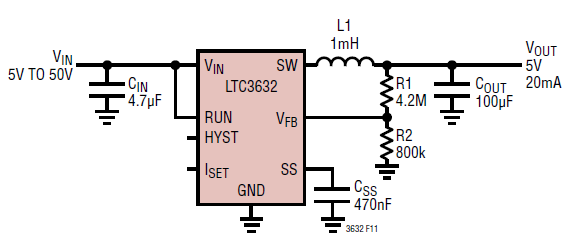


Figure 27: Sample implementation of the LTC3632

(Pending permission from Linear Technology)

Here, the LC filter that was discussed previously can be seen, and these are the components that do a majority of the work for this regulator. This external circuitry is what maintains the voltage at 5 volts.

After further investigation, we finally found the right voltage regulator for our project. We decided to use LM2576 model switching regulators from TI which we were able to sample for free. These were very convenient for us because the LM2576 came in both a 5V and 3.3V part. The setup was rather simple, since the only additional parts we needed for a switching regulator were an inductor and a zener diode. The 5V model is rated at 77% efficiency and the 3.3V model is rated at 75%. In comparison to the linear regulators, this was great. The input voltage range for these parts are anywhere from 7-40V, and since we used a 12V battery, this lied well within the parts range.

**5.0 Power and Communication**

**5.1 Chassis**

Since there were plenty of pre-made remote control cars as well as those we could have put together ourselves available to us, we decided to breakdown the advantages and

disadvantages of each to come up with the best solution for our robots main body. Not only did the design have to be effective, but the cost had to be right as well. The first option we looked into was to build a simple chassis ourselves. One model we found was the Boe Bot robot chassis sold by Parallax.

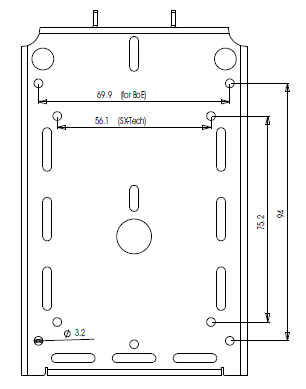


Figure 28: Boe Bot Chassis Dimensions Bottom View (Reprinted with permission from Parallax)

As can be seen from figure 28, there was a lot of room and plenty of premade holes for mounting wheels, motors and other components and at only $24.99, it seemed like it would have acted as a nice base for our chassis. The total length of this chassis is 127 cm and it is 87 cm wide. This would have provided us with plenty of real estate to not only assemble our electronic parts on, but also place the motors with which our robot would move with. The chassis is also squarely punched in order to be compatible with Parallax and Futaba style servomotors. This may have worked to our advantage since no additional drilling would have been needed to mount our motors. We did find other base chassis; however, most of them were way out of our price range. They ranged anywhere from $60 to well over $100. Since this conflicted with our goal of low cost, we had to be careful when it came time to choose our parts, and this included the chassis.

Despite the fact that the design of the above chassis supports servo motors, we decided to look into other types of motors as well. We found that we could either use continuous DC motors, stepper motors or the aforementioned servo motors. We did some research and found advantages and disadvantages to both which are outlined in Table 9.

|  |  |  |
| --- | --- | --- |
| Motor | Pros | Cons |
| Cont.  DC | * Wide selection available, both new and used. * Very easy to control via computer with relays or electronic switches. | * Requires gear reduction to provide torques needed for most robotic applications. * Poor standards in size and mounting arrangements. |
| Stepper | * Does not require gear reduction to power at low speeds. * Low cost when purchased on the surplus market. * Dynamic braking effect achieved by leaving coils of stepper motor energized (motor will not turn, but will lock in place). | * Poor performance under varying loads. Not great for robot locomotion over uneven surfaces. * Consumes high current. * Needs special driving circuit to provide stepping rotation. |
| RC  Servo | * Least expensive non-surplus source for gear motors. * Can be used for precise angular control, or for continuous rotation (the latter requires modification). * Available in several standard sizes, with standard mounting holes. | * Sometimes require mods. For continuous rotation. * Requires special driving circuit. * Though more powerful servos are available, practical weight limit for powering a robot is about 10 pounds. |

Table 9: Various Motor Pros and Cons

**5.2 Motors**

**5.2.1 Continuous DC**

The first type of motor we looked at was the continuous DC motor. We liked the fact that they were easy to find and relatively easy to implement with a computer. Since none of us had any kind of experience with small motors, we looked into some components that would make our DC motor work for us. The first main component needed for most of them was an H-bridge. What this would allow us to do is to get both a forward and reverse response from the motor without having to rewire our circuit. This circuit could be set up with either mechanical switches or electronic switches, but since our background revolved around electronic components, it would have been in our best interest to use electronic based switches.

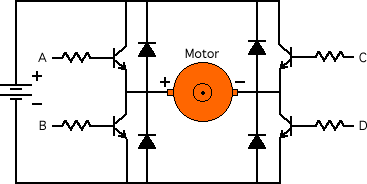


Figure 29: Typical H-bridge using transistors as switches (Reprinted with permission from Micromouse)

In the above circuit, the transistors replace the mechanical switches. In this circuit, the winding in the motor acts as an inductor and when the current through the motor is shut off, the current can’t immediately go to zero, and a voltage is produced that opposes this current, therefore a diode is placed in parallel with each transistor to provide a safe pathway for current to go in order to not blow the transistors. Closing switches A and D would result in forward motion and closing C and B would result in reverse.

Since our design require 2 DC motors (one in the rear of the robot and one in the front), to power each set of wheels, we felt it would be advantageous to find an H-bridge that could handle such a task. We found such a part made by Texas Instruments called the DRV8835 Dual Low Voltage H-bridge IC.

This is a 12 pin device and has a layout as seen in Table 10 below.

|  |  |
| --- | --- |
| Pin | Function |
| 1-VM | Motor Supply Voltage |
| 2-AOUT1 | Bridge A Output 1 |
| 3-AOUT2 | Bridge A Output 2 |
| 4-BOUT1 | Bridge B Output 1 |
| 5-BOUT2 | Bridge B Output 2 |
| 6-GND | Ground |
| 7-BIN2/ BENBL | Bridge B Input 2/Enable Input |
| 8-BIN1/ BPHASE | Bridge B Input 1/Phase Input |
| 9-AIN2/ AENBL | Bridge A Input2/Enable Input |
| 10-AIN1/ APHASE | Bridge A Input 1/Phase Input |
| 11-MODE | Input Select Mode |
| 12-VCC | Device Supply Voltage |

Table 10: DRV8835 Dual Low Voltage H-bridge IC Specification table

This device supports a 2-11 V motor operating voltage and a device supply voltage of 2-7 V. This range sounded perfect for us as it allowed for a lot of flexibility with the power supply that we chose. We found out there were two modes for this device, In/In Mode and Phase/Enable mode. Upon reviewing the data sheet, In/In mode sets the MODE pin to 0 and allows for forward, reverse, brake and coasting modes dependent on the four possibilities for both In pins. The Phase/Enable mode sets the MODE pin highand only allows for forward, reverse and braking features. Either mode of this H-bridge would have worked for us if we chose to go with a DC motor setup. The AOUT and BOUT pins are simply connected to the motor.

Another common part associated with using DC motors is a gearbox. However, since our robot was supposed to be light weight and not have to carry a heavy load, a high torque option was not deemed to be necessary, however, there are high rpm motors out there that we could have used that would suit our needs that come with a good enough gear ratio for low speeds. If we chose to use DC motors in our design, the gears would be taken care of in the initial design of the motor, so we did not need to worry about any extra gear reduction.

We did find a DC motor that would be perfect for our design. The MT-WSM500 was a high torque mini DC gear motor rated from 3-12 V. This model costs $12.95 per unit and is shown in Figure 30. The small size of the motor allows it to be easily concealable and would have taken up very little room on our chassis should we chose to use it.

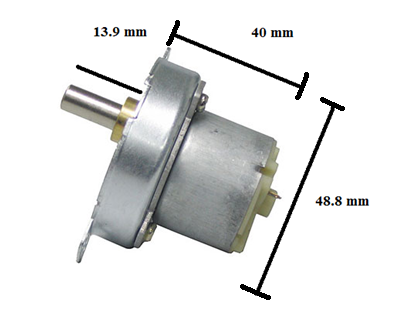


Figure 30: MT-WSM500 DC Motor dimensions (Reprinted with permission from BatterySpace)

If we were to use the above H-bridge, the connections would be easy enough. We would need two for our design to power the rear wheels and all that would be left to do would be to decide which mode (In/In or Phase/Enable) we wanted to use. Here are a few features that made this motor desirable to us.

|  |  |
| --- | --- |
| Characteristic | Specification |
| Gear Reduce Ratio | 1:108 |
| Voltage & Current | 3-12 V DC < 25 mA with no load |
| Torque | 1000 g-cm @ 6V |
| RPM | 5 RPM @ 3V, 25 RPM @12 V |
| Reversibility | Yes |
| Weight Per Motor | 4 oz |

Table 11: Continuous DC motor specifications

Since the terrain our robot was designed to go through may not necessarily be very even, this high torque motor could have been what we needed after all. The fact that each motor weighed only 4 oz was also really great, since we wanted to keep the robot light and mobile. Since the H bridge supports up to an 11 V input, we wouldn’t need to run these motors at their full potential, but we would be able to get pretty close to it if we chose to use the 11 V that the part was rated at.

**5.2.2 Stepper Motor**

The next option we had for a motor was the stepper motor. While the disadvantages seemed to outweigh the advantages of using this type of motor shown in Figure 30, it was still worth seeing what this kind of set up would do for our overall design.

The addition of a stepper motor could have provided some extra precision that simply could not be reached with a DC motor. Since stepper motors have a certain amount of steps that can be implemented per 360 degrees, it would have been very easy to control how many steps were taken by the microcontroller. If the degree per step was known as well as the gear ratio and radius of the tires used, it would have been very easy to make the robot as precise as needed by making the proper calculations. The main problem lied with the terrain our robot was designed for. It would have been really hard to predict the terrain of the robot until it was actually in the disaster area. Since stepper motors use a permanent magnet rotating shaft, the magnets would constantly be shifting around, providing us with the strong possibility of having our motors not work to our standards due to offsetting the magnets. Also, as mentioned before, these precise movements are for more finesse applications and since we already decided we were going to use a video camera to gauge our distance, this extra precision would not be needed. Since the camera would not be affected by the conditions the robot saw, we decided that stepper motors would be unnecessary for this design.

**5.2.3 RC Servo Motors**

This section talks about servo motors and how they would be useful in our design. These motors are common in radio applications, which would be very beneficial to us since we decided to use this type of communication. In the application of RC cars, these motors are often used to control the throttle, brakes and steering, however with modification, the motors could be used to control the remote control cars wheels.

Most servo motors have a range of 0 to 180 degrees, but this did not benefit us in making our potential servo motor into a drive motor. Instead, some modifications would have to be made to allow for a full 360 degrees of rotation. We could have employed the brute force method and opened up the motor, removed the electronics and brought out the power lines and either remove or modify the potentiometer to get it to rotate the full 360 degrees. The only thing we would have had to do then was mount pieces for the wheels to go on and make sure that we had a motor driver circuit to make the motor functional.

The easiest way to modify the motor would have been to break the feedback loop that is internal in the motors circuitry. This could most easily be done by removing the potentiometer (which is basically a variable resistor) from the circuit and replacing it with a voltage divider. This would have been advantageous to us because it would allow us to keep the motor in the neutral position. We wanted this to occur because keeping it in the neutral position would allow for the motor to turn at the same rate when it went either forwards or backwards. This technique would work because the potentiometer would then not be able to limit the servo motors rotations. Thankfully none of these methods needed to be implemented because servo motors now come with a full 360 degree range.

The RC servo motors are controlled by Pulse Code Modulation (PCM). The parameters for this pulse are that there has to be a pulse with a minimum width, a maximum width and another pulse for repetition rate. The neutral pulse, which would allow for the potential of rotation in both clockwise and counterclockwise motions to operate the same would be around 1500 micro seconds. Providing these pulses above and below this rate would have allowed for our forward and reverse motion. In general, the minimum pulse rate was around 1000 micro seconds and the maximum around 2000 micro seconds. However, these values were dependent on the type and brand of the servo motor used.

Servo motors generally consist of a motor, feedback device and a control board. The feedback device is usually a potentiometer. The motor is designed to turn the output shaft and the potentiometer at the same time, however, our design would replace the variable resistor with a voltage divider, hence not allowing for any change in the resistance. The servo would take in a pulse and the control board would decipher this data. Since the motor would now be able to move 360 degrees without any modification, gears will not be limited. The control board would then provide the correct action that the motor would take and this would allow our robot to move either forwards or backwards, depending on the length of our pulse.

Now that we understood how these motors worked, it was time to find different parts and compare their specifications. It delighted us to know that the tedious task of modifying the servo did not have to take place after all, since there were models out there that had 360 degree rotation available at the time of purchase. The first motor we found was sold by SparkFun made by SpringRC. Figure 31 below shows the dimensions of this servo motor.

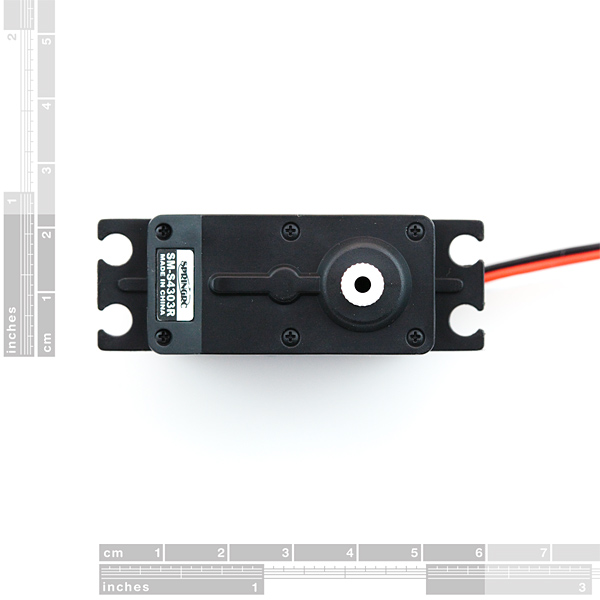


Figure 31: SpringRC 43R Servo motor dimensions (Reprinted with permission from SparkFun)

As mentioned before, this model already had a full 360 degrees of motion making our lives a lot easier since we would not have to make modifications to it. Its operating voltage was anywhere from 4.3-6 V DC. We were not looking for a heavy duty motor with tons of torque, so the maximum voltage allowed could give us a torque of 4.8 kg-cm. Even at 4.3 V, we would have had plenty of torque since it is rated at 3.3 kg-cm at this voltage level. The hook up of this model was relatively easy since like most servo motors, there were only 3 wires involved. In this model, the red wire was set to Vcc, the black wire was set to ground and the white wire would be used to read the incoming signal.

This models intermediate impulse width was 1.5 milliseconds, which matched up with what we expected when we researched servo motors. Any pulse width above this value would result in clockwise rotations of the motor and vice versa. We also appreciated that the motors were relatively light at only 44 g. The price was also very competitive to the DC motors we were looking at since this model cost $13.95 per unit.

The next model we looked at was the SM-S4303R servo made by SpringRC and was sold by Phidgets Inc. This particular model had an operating voltage of 5 V and also produced 3.3 kg-cm of torque at 4.8 V just like the motor found on SparkFun. This motor weighed only 41 g, making it just a few grams lighter than the first model we looked at. One disadvantage of this motor compared to the first one was that we couldn’t gain more rotations out of the motor. It had a set operating voltage whereas the model on SparkFun allowed for a little better range. Despite this minor disadvantage, this motor still had some appeal since it cost only $10.70 per unit.

**5.2.4 Price Comparison**

Now that we had searched for a chassis and researched different parts that could propel our robot, it was time to see if the components provided the right function and what we were looking for, but also if the prices were right. It quickly became apparent that building and mounting everything ourselves would be quite expensive, and with all of the other components that we still had to buy at time, this idea seemed highly unpractical. If we were to use the Boe Bot chassis and either the DC or servo motors, this would have already brought our cost above $50. This price did not account for the four wheels and tires needed or equipment that would assist us in steering the front wheels. Also, the fact that we would have to make additional modifications to the chassis for mounting purposes did not make this method of doing things very desirable.

This left us with the idea of buying a pre assembled remote control car and simply modifying it. This process may not have been too easy since there was a possibility that the remote control car we purchased would not come with schematic diagrams that show how the internal circuitry works, hence making it difficult for us to tap into the controls for the motors. Despite this minor setback, further research showned that this method has been implemented many times before with many cases being greatly successful. Also, since we had a smaller group of only 3 members, we decided it was not in our best interest to build our own anyway since none of us had any prior experience building such things and we wanted to make this project stand out for the technology implemented in it and not for its homemade look. Therefore, we decided to buy a preassembled model and focus on the electronics. Once we received our remote control car during the summer months, we gutted all of the circuitry and made room for everything of our own to avoid the problem of having to tap into the pre-existing circuitry.

We ended up searching online and compared a lot of different models. We wanted something that was rather compact, but also something that had some decent speed. Our plan was to gut the model car of most of its components, leaving the chassis top and motors intact. We then wanted to modify the car in a way such that it looked like we did little to nothing at all to it. This would mean that the components we chose would have to allow for enough room for us to be able to hook up our electronics and then be able to hide them underneath the body of the chassis. Doing this would also test us in our ability to make our design compact. We have seen many cool search and rescue projects in the past, but some looked rather chunky and did not use the space available to them in a way that would give the appearance of a sleek design. It was quite common to see these smaller designs done on a more professional level since the designers in charge have so much more experience than we do. However, with proper attention to detail and careful planning, we felt like we could make this happen. Having a limited space would challenge us to use every inch of what’s available to us to the fullest. After receiving our car, it became apparent that the space the chassis offered would not be big enough to house all of our circuitry, so this idea was scrapped and we built an acrylic platform instead.

We found the perfect candidate for our remote control car at Target. Not only did it look the part of being a rugged search and rescue vehicle, but it also appeared to have enough space underneath the body for us to work with.

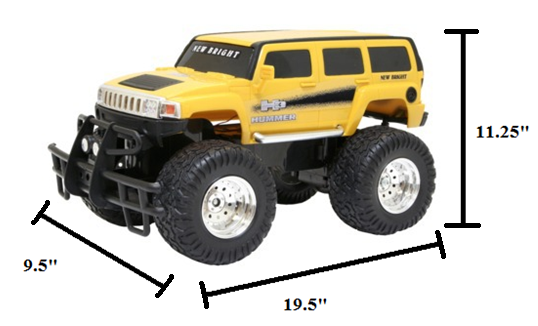


Figure 32: Remote Control Car dimensions (Reprinting pending permission from Target)

Figure 32 shows the remote control car that we decided to use for our project. This model originally relied on RF communication to make it move, but since we gutted the original circuitry, this allowed us to explore other options for communication. This model also came with a 6 V rechargeable battery and charger. The fact that this remote control car was already carefully designed allowed us to include our new 12 V power source on the robot where the originally battery lied without making any further modifications if we chose to. Doing this would allow us to keep going with the idea of keeping the stock look of the vehicle. The original set up of the car allowed for a range of 100 feet with the remote control that it came with, but further research was done to see if this distance could be improved upon with components that we purchased separately. It turned out that the only way to extend the lifetime of our battery was to use a larger battery, and in our case, the 12 V battery gave us 30 minutes of use.

One drawback of this car was that its maximum running time was only 20 minutes if we were to use the 6 V battery it came with. However, we did not plan on running the robot at full speed at all times since the environments it was designed to be used in would simply not allow that. Therefore, this was really not a big issue. We did like the fact that it did have the potential for decent speed ratings. If our robot is in an environment such that there is a clear path and we needed it to make a quick escape, it could ideally reach speeds of up to 7 mph.

Another great feature of this car was its original weight and decent ground clearance. Since it only weighed 4.56 pounds, this allow for us to transport this model very easily from one place to another. Also, the lighter our final robot was, the less power it would need to consume in order to move, so this initial weight was a great starting point. We wanted the weight of our final design to weigh as little as possible, keeping the amount of torque needed to propel it to a minimum. A lot of the other remote control cars were simply too low to the ground and would not thrive in an environment with debris in its way. One concern we did have was the fact that the tires are plastic. They simply did not get the traction necessary to move in some environments, but the advantages that this remote control car had to offer to us simply could not be ignored.

The last specification that sealed the deal on this car for us was the price. It only cost us $29.99. For this price, we were able to get a chassis, motors that would propel the car, and a 6 V rechargeable battery and a charger. Although we did not use this stock battery for anything, it was nice to know that we had a backup in case anything happened. For the price, this simply could not be beat when compared with getting all of the initial components ourselves. At this price, it would also have been relatively cheap for us to buy a whole new unit if necessary in case we damaged anything. This was obviously not ideal, but it would have saved us lots of hassle if we needed to replace it since the cost was so low. The fact that we only needed to go to Target to buy this model instead of having to order one online and not know when we would receive it would also have worked in our favor.

**5.3 Communications**

One major question we had upon building our robot was how we were going to get out robot to move and operate with the best results. We decided to look into both infrared and radio frequency technologies and see what the advantages and disadvantages were for both. Not only did the technology have to be suitable, but it also had to fit our budget. Therefore, it was necessary to take a good look at both kinds of technologies and decide what was best for us.

**5.3.1 Infrared Communication**

In this day and age, the popularity of infrared communication is picking up steam. We decided that it would be worth our time to see if infrared would provide us with the best results for our robots mobility. We knew that this technology was very prevalent in the remote controls of many household items such as DVD players and TVs, but we were curious how infrared would work in the claustrophobic environments which our robot would be in.

It quickly became evident that infrared provided many advantages. The fact that it requires low power to operate greatly intrigued us since we were interested in building a low power robot. The circuitry also happened to cost very little which also piqued our interest. Another interesting thing we found out was that since the whole system would require excellent line of sight, better security was enabled. Any infiltration of our data would require that other parties had to be able to access great line of sight as well, but since we were not concerned who retrieved our data, since the basis of our robot was to inform as many people about certain conditions as possible anyways, we felt that this property did not matter to us.

Infrared provided a much shorter wavelength which would have allowed us to gain readings from our robot at a much higher rate than anything at a lower frequency, such as radio waves. This would have served to make the operation of our robot more accurate, since the response of our robot would occur much quicker.

**5.3.2 RF (Radio Wave) Communication**

After doing research, we decided that this was the ideal type of communication we were going to implement for our project. The biggest problem with the infrared technology was simply line-of-sight. Since our robot was designed to go into disaster areas, the chances that objects would obstruct our sight with the robot would have been extremely high, thus making infrared technology useless to us. Another advantage of using RF communication was that the range was simply greater. With an RF setup, it was possible to have an ideal range of 300 feet from the remote (which in our case is a laptop) to the base unit.



Figure 33: Module showing how we want our robot to receive instructions

Figure 33 shows the very basic operation of how we wanted to control our robot. Basically, the laptop acts as the base for the transmission of the direction we want our robot to go in, which is picked up by a receiver on our chassis. We had decided to forego the use of the remote control that came with our car and decided to use the directional pad on a laptop as the primary means of making our robot go in the desired direction. This allowed us to use the laptop keys to steer while using the screen as the “eyes” of our robot. This system eliminated the original remote control completely and made the number of components of our overall design fewer.

Since RF technology has been used in many applications over the years, the amount of parts available to implement the technology was rather high. Not only was the technology readily available, but it could be implemented in many different ways. This part of the paper looks at a few different RF receivers and transmitters and weighs the advantages and disadvantes of each so that we could pick the parts that best suited our needs.

**5.3.4 Wireless Adaptors**

We first decided to look at different RF transmitters that were currently on the market. The first one we encountered was an RF Link Transmitter rated at 433 MHz found at Sparkfun. The following figure shows how big it is. The part itself was extremely small and for this reason, concerns were raised about how we would integrate such a small part into our project.

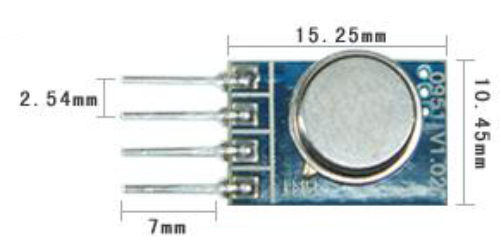


Figure 34: Wenshing Hi Power Transmitting Module (Reprinted with permission from SparkFun)

The following diagram shows the pin layout for the device.

|  |  |
| --- | --- |
| Pin | Signal |
| 1 | GND |
| 2 | Data In |
| 3 | Vcc |
| 4 | ANT |

Table 12: Pin Layout of Wenshing Transmitter

We already knew what the first 3 pins represented but after a little research we figured out that to properly transmit instructions to our robot, an antenna was needed. An in depth look at antennas will be discussed a little later on.

Table 13 below shows some key specifications of this transmitter.

|  |  |
| --- | --- |
| Characteristic | Unit |
| Operating Frequency | 434 MHz |
| Data Rate | 8 kbps |
| Current Consumption | 8 mA |
| Output Power | 32 mW |
| Operating Voltage | 3-12 V |
| Operating Ambient Temp | -20 to +85 degrees C |

Table 13: Highlighted specs of Wenshing transmitter

This transmitter did have a couple of nice features about it. We liked the fact that the operating voltage was so broad. This would enable us to find a power source that fit our budget rather than being limited to a certain voltage. Also, since applications of this transmitter involved it being used in car keys fobs, wireless game pads and wireless toys, this transmitter may have been exactly what we were looking for. We also liked the fact that the output power was low, but we also had to consider that a possible antenna would need power as well.

Te next transmitter we looked at again came from SparkFun. It was an RF Link Transmitter rated at 315 MHz. The pin assignment for this device was just like that of

the 434 MHz component. Just like the previous transmitter, this device required an antenna. Without this antenna, the range we would get from it would be useless in comparison to what we needed for our project. Since using RF technology allowed us to cover great distances, we wanted to optimize the distance at which our robot would be able to operate. If we were able to design or acquire a good antenna, this transmitter would allow us to control our robot from up to 500 feet away. Despite the great range, none of us had experience designing RF antennas or laying one out on a board to achieve optimum performance, so we knew we would not be able to gain a range this good. Despite this fact, we were confident we could at least have successful transmission from at least 200 feet away. The specs for this transmitter were also exactly the same as the previous transmitter we looked at.

The characteristics of this transmitter were relatively the same as the one we looked at before, the only difference being the operating frequency. This brought up the question of which of these frequencies would be best for us to use or if we should use these frequencies at all. It turns out that 315 MHz transmitters were primarily used for remote keyless entry and garage door openers. Since these applications are used quite often, there is a great chance for interference. Since human life may be at stake, we wanted the smallest possible chances for interference. Further research showed that 433 MHz may not have been a good option either. The transmitting distance was great, at around 1000 feet, however, in the United States, 433 MHz is a popular band for amateur radio stations. Since it is impossible to predict where the bases of these amateur radio stations reside, we would still be taking a risk using this technology.

Operating at a less crowded frequency of 418 MHz was also an option, but on top of finding an antenna and receiver combination to support this frequency, we would also have had to develop a whole board with a microcontroller purchased separately to implement it. Since both the 315 and 433 MHz option also provided problems, we looked into other options.

We found that the best option available to us was to use an all-inclusive kit that offered a different frequency. We ended up finding an Xbee Wireless Communication Starter Kit that operated around 2.4 GHz that had been successfully implemented in past projects for only $79.99. Included with this kit were 2 Xbee 1 mW communication modules (transceivers), an XBee Explorer USB, an XBee Explorer Regulated, and a Mini B USB cable. We were also able to get one more XBee transceiver from a friend to complete our three XBee module setup.

We liked this module a lot for many reasons. For one, the modules operated on 3.3 V at 50 mA. This not only allowed us to keep our power output low for this piece of circuitry, but it also required a small voltage level, which had the potential to keep our design compact. Another highlight of this system was that it allowed up to a 250 kbps max data rate. Since the data we were interested in was not overtly complex, thus not taking up nearly that much capacity, this gave us more than enough to work with. The range this kit offered was also very nice. With ideal conditions, it was supposed to provide operation within a 300 ft range. Since we decided that 100 ft of range for our robot would be sufficient, this gave us plenty of extra room to work with, and it allowed us to go deeper into whichever kind of disaster area we send our robot into than we originally hoped for. The key feature that enabled this great range was the built in antenna. With the parts we considered before, we would have had to implement our own antenna, and since none of us had any experience with dealing with all of the factors that generate a successfully working antenna, we felt that this was a great feature to this kit.

The Explorer USB which is a USB serial unit that enabled us to program the pins on the modules also came with the kit and it provided us a way to interface our laptop with the robot via X-CTU software. As mentioned before, the kit also included a mini USB that connects the explorer to our laptop. Figure 35 shows all of the pins on this Device as well as the size. The part itself is very compact and takes up very little real estate. There are multiple pins in which we can input data. We would then be able to connect the data out pin to our receiver and be one step closer to interfacing with our robot. The USB provides a simple connection from the Explorer to our laptop and also provides great portability and easy way to make a connection no matter where our robot is assigned to go.

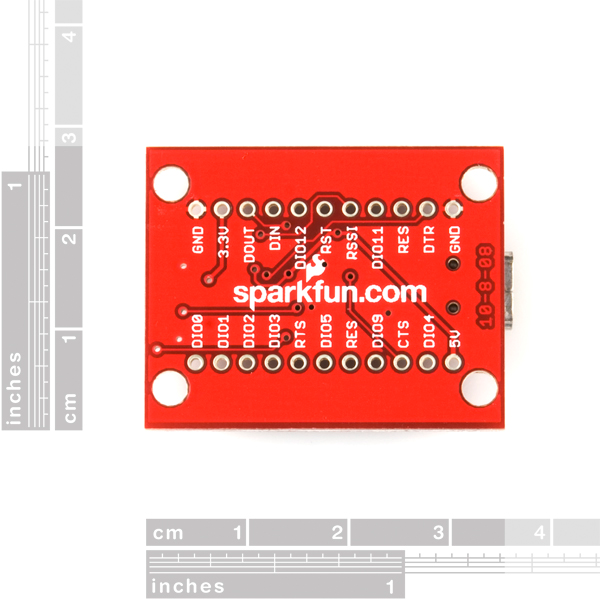


Figure 35: Xbee Explorer USB (Reprinted with permission from SparkFun)

The next part of the kit came in the form of a voltage regulator. It allowed us to input 5 V to the part where it would then regulate it down to 3.3 V, which is the operating voltage that these parts operate at. Any more voltage than that would not guarantee that the part would work and anything less would simply leave us with some fancy pieces of hardware that didn’t do anything. On top of just being a regulator, this component also assisted in signal conditioning. It is als equipped with a few LEDs that indicate to the user that the power is on as well as if the data in and data out pins are functioning the way that they should be. This helped us out a lot with the initial troubleshooting that was done once it was hooked up. To use this part, we simply had to take our Xbee receiver and plug it into the connectors on the regulator which it came with. Figure 36 shows the Explorer Regulated. It too is very tiny. We decided to only use the regulator to troubleshoot during the initial testing phase. We made sure to have a voltage regulator on our PCB that regulated down to 3.3 V to take this parts place.

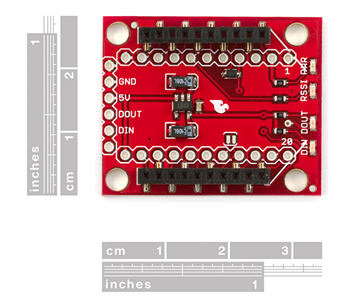


Figure 36: Xbee Explorer Regulated (Reprinted with permission from SparkFun)

The final components that came with the kit were two 1 mW communication modules. One plugged directly into our regulator pictured in Figure 36 during testing and we removed it from the regulator when it came time to put it on the PCB and the other is attached to our motor controller on the robot itself. An additional XBee module was hooked up to our microcontroller and it allowed us to take the sensor data from the microcontroller and transmit it back to our base. For the price, the receiver sensitivity was amazing. It was able to pick up a wide range of signals, the smallest of which would be at -92 dB. This was great because we needed to be able to know that our robot would still pick up signals and operate accordingly even if the environment it was in did not allow for maximum strength from the signal. The best features of these modules in our opinion were the built in antennas. They are whip antennas, but are very small due to the high frequency at which they operate at. This was very convenient for us because we did not have to set aside too much room to accommodate the antennas. Figure 37 shows this module with the built in antenna.

Since this kit has been very popular in recent years, there were plenty of guides and sample code we used to troubleshoot our devices. The LEDs were also a nice touch since we did not have to go far into the configuration of our parts to realize that something is wrong because of them. Any help that we could get was much appreciated since this was the first time we used wireless technology for our own application.

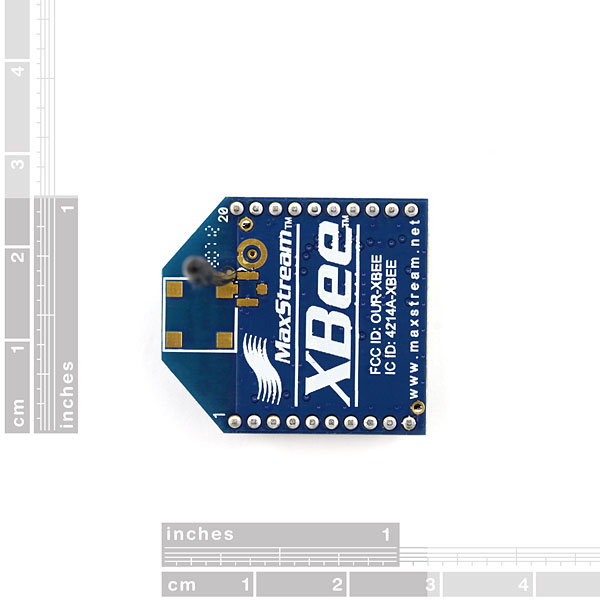


Figure 37: Xbee 1 mW Wire Antenna Transceiver (Reprinted with permission from SparkFun)

**5.4 Power Supply**

**5.4.1 Electronics Power Supply**

It was previously decided that we would use a different battery than the remote control car came with in order to power the vehicle which allowed for a longer battery life. However, one of the biggest obstacles we had to overcome was how we were going to power our robot’s electronics since there were so many options for batteries out there. We knew that some form of battery would be effective, but finding the right one took some research. There were simply variations that we could have used. This section is used to highlight different types of batteries to figure out which one would work best for us.

One thing we considered about the battery was its size. Ideally, we wanted the battery to rest underneath the chassis. However, mounting them to the back or the roof of the chassis would have been an issue if we decided to keep the original top part of the chassis. The next thing we considered was whether we would like it to be rechargeable or not. Depending on how quality the batteries are, this rechargeable feature could help save us money down the line. Not only did the size have to right, but the efficiency of the battery had to be great as well. We would rather spend the money on an efficient battery we’d have to throw away than a rechargeable one that simply doesn’t last. Further research led us to the right battery that worked for us.

There are currently a lot of different types of batteries available out there that can be used for robotics projects. They include, but are not limited to alkaline, lithium (both of which are disposable), lead acid, nickel cadmium (NiCad), and nickel metal hydride (NiMH) among many others. Each battery has its advantages and disadvantages and once each one was looked at carefully and because of this, we were able to find the right one that worked for us.

While looking into these types of batteries, we learned more about the rating of them. While we knew that putting batteries in series would result in more voltage, it was not apparent to us that putting batteries in parallel with each other would produce more current. This was an important realization since some of our components needed more current than others to work to their full potential. It was also important to note that if we were able to wire our batteries in a parallel connection we would have had to avoid self-discharging. This occurs when one battery charges another in parallel, which results in high inefficiency and could make for permanent damage done to the battery. To avoid doing this, we would have had to avoid putting batteries in parallel that did not have equal charges, as one would charge the other, resulting in self discharge. It was also very important to make sure that the batteries were the same type, which would also help avoid this problem. It was also not wise to mix in older batteries with newer batteries, as this would also create problems.

Another new term we were introduced to was the mAh (milli Amp hour). This is the rating of how much power the particular battery could give. If a battery is rated at 1000 mAh, this would mean we could get 1 A of current for a whole hour, 2 A of current for 30 minutes or 4 A for 15 minutes. The amount of current is proportional to the amount of time the battery will last.

**5.4.2 Alkaline**

Now that we had some background information common to all batteries, it was time to start our research by looking into alkaline, or otherwise known as disposable batteries. One great thing about these is that they are very widely available and come in so many shapes and sizes. The typical voltage level for the common alkaline battery is around 1.5 V per cell, however, sizes up to 9 V are available. We decided to take a look at a typical AA battery made by Energizer.

Each unit supplies 1.5 V of power. Since our minimum required voltage on most of our components was 5 V, at least 4 would be needed to provide the proper power as well as some form of regulator to produce the right voltage. As we learned earlier, not only was the voltage important, but the current it was able to produce was very important to us as well.

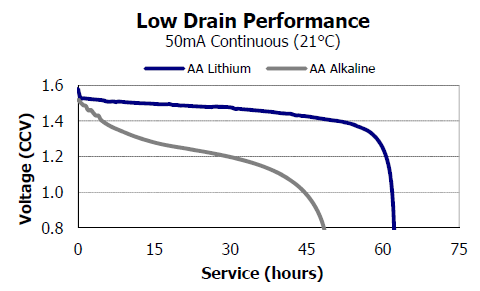


Figure 38: Alkaline and Lithium AA Battery Performance (Pending Permission from Energizer)

Figure 38 shows the performance of both alkaline and lithium disposable batteries made by Energizer. From the graph, it was clear that the lithium battery lasted a lot longer than its counterpart and also stated at its rated voltage of 1.5 V per cell for a lot longer. Therefore, the lithium would have been our choice if we decided to use disposable batteries.

Since power was an important factor in our design, we wanted to keep the current consumption relatively low. This graph shows the performance of a AA battery over time. If we were to use these batteries to get up to 50 mA out of them, we would expect to get around 30 hours of use, since we would not want to run the batteries once they have discharged enough to only be at 1.2 V. However, we needed a lot more current than this to power everything. This graph also brought up the issue that once these batteries did discharge to this level, we would only be at a total level of 4.8 V, which would not sufficiently supply enough power. Therefore, it would have been a better idea to use them for only 4-5 hours, since this would have assured us that our batteries each supplied 1.4 V, which would leave us with a total of 5.6 V available to use. However, again, 50 mA was simply not enough total current for our application. The weight of the batteries was also very nice. Each battery typically weighs only 0.5 ounces. Therefore, this would have added very little added weight to our design.

Another great thing about this type of battery was its great shelf life. We could leave these batteries dormant for years and they would still be just as efficient as if we bought them yesterday. They also worked very well in a wide range of temperatures which would allow us to bring the DZERV to a wide range of different environments. They also provided a slow self discharge rate, which would allow us to power our electronics for a decent amount of time.

One thing we did not like about these batteries was that the cost would be relatively high simply because we could not recharge these batteries. This would not pose a problem in the beginning, but over time, with continuous use, these would have to be replaced. Another issue that came along with this would be the disposal of the batteries. Simply throwing them in the garbage was not an option, since these batteries do contain harmful chemicals that have the potential to cause problems for the environment once they start to break down. Extra care and consideration would be needed to avoid this.

**5.4.3 Lithium**

There was another type of disposable battery out there available to us. It was the lithium battery. They are very similar to the above mentioned alkaline batteries, but provide some significant improvements. These batteries come in a variety of shapes in sizes, such as button cells, AA, AAA and 9 V sizes. However, we were only interested in looking at the commonly used sizes that are found in most consumer electronics today. One advantage this battery held over regular alkaline batteries was that each cell could produce anywhere from 1.5-3.7 V per cell, which is over twice the voltage of the alkaline.

Not only was the voltage provided a little higher, but the information in Figure 38 Showed just how much more efficient they were. If we were to run these batteries to achieve a maximum current of 50 mA, we could run these batteries for a little over 60 hours. However, since we must maintain a voltage level of at least 5 V for most of our electronics, we would have to limit this value a little bit and only run them around 50 hours, since this value would allow us to still have 1.4 V from each battery available to us. However, much more current was required for our project, so this projected battery life was out of the question. When compared to rechargeable batteries, they simply lasted a lot longer. This would be something that we had to consider before we made our final decision.

**5.4.3 Lead Acid**

The next potential candidate for a battery for our electronics was a lead acid battery. We were quick to realize that this would not be a good idea simply for its bulkiness. These are the batteries that are used primarily for motor vehicles. With bulkiness comes added weight, and since our initial robot weighed less than 5 pounds, our battery would clearly weigh more than the vehicle itself, even after the final components were placed on our platform. Even if we were able to use this battery, the added torque needed to get our vehicle to move would drain our motor battery relatively quickly. Another disadvantage

of this battery was its high current output. Since this battery is used to achieve this, the components for our sensors and camera would burnout rather quickly.

**5.4.4 Nickel Cadmium (NiCd)**

Our quest for a potential power source had led us to explore the realm of rechargeable batteries as well. One such battery was the nickel cadmium battery. This type of battery was a great contender for us since it had a lot of advantages and very few disadvantages. These batteries are very difficult to damage since they are able to tolerate deep discharge for long periods of time. Since these batteries are able to be recharged, it was important to find out just how many cycles these batteries could withstand in terms of charge and discharge cycles. It turns out that the NiCd model had the best performance in this area when compared to other rechargeable batteries. The energy density level was also quite high. This simply meant that per unit of volume, more energy was able to be stored, which could help us run our electronics over a greater period of time with smaller batteries. The nickel cadmium batteries also had a smaller self-discharge rate when compared to other rechargeable ones, at only 20% per month. One of the best attributes of this battery was that it had very relatively high current output, which could have been very beneficial in powering a small to moderate sized robot such as ours.

Despite all of the good things to say about the NiCd, there were some disadvantages associated with it. They did tend to cost more than other rechargeable batteries. The metal itself (cadmium) is also highly toxic to most life forms. However, technology has made it so they can be used without worrying about doing harm. One of the biggest problems associated with this battery was its memory effect. These batteries store less and less charge after each recharge. This could have posed a problem to us after a while since we would not be able to rely on the ratings of our batteries. The best scenario for recharging would be to wait until the battery was fully discharged, but this would be very inconvenient for us since the electronic components would cease to work a long time before this would occur. However, if we could get enough uses out of the battery despite its memory effects, this type of battery would still be a good candidate.

**5.4.5 Nickel Metal Hydride (NiMH)**

The last potential battery we looked at was the nickel metal hydride (NiMH) variety. It too was rechargeable and came in many different forms, including the common AA size. These batteries are similar to the NiCd batteries, but instead of using cadmium as the main material used as the anode, hydrogen is used instead. Each cell for one of these batteries would provide 1.2 V.

The NiMH model has made a lot of improvements over the NiCd batteries and provides a lot of advantages for using them. For one, they have a very wide operating temperature anywhere from -30 to +75 degrees C. This would provide a great range of environments for our robot to be in where we could expect solid performance from the battery. These batteries also possess a very long lifetime at around 3000 cycles. Recharging the battery could be as little as 1 hour, but typically it is done over a longer period of time and since we didn’t plan on having to do any emergency recharging, this was not an issue. These batteries also have better results when it comes to memory effects. Unlike the Ni-Cd batteries, the NiMH batteries do not require full discharge every time you wish to recharge the battery. This was a great feature since we would not have to worry about the battery failing since we could recharge it daily. The only condition with this was that we let the battery fully discharge once a month. The last nice feature about these is that they are a lot friendlier to the environment then the Ni-Cd batteries since there is no lead, cadmium or mercury inside of them.

There are some disadvantages to using this battery as well. For one, they have a high discharge rate, which means they will not last as long as the NiCd batteries per charge. Long time storage of these batteries is also discouraged since they do deteriorate over time but reconditioning them (charging and discharging them several times before reuse) will help counteract this problem. These batteries are also susceptible to overcharging, but with a good quality charger, this problem can be taken care of. Also, at only 1.2 V per cell, they provide less voltage than disposable alkaline batteries, thus requiring us to have more batteries for similar voltage levels.

**5.5 Choosing a Battery**

Upon further discussing it between our group members, we decided to go with a rechargeable battery for our electronics. We know we wanted either a NiCd or NiMH battery so we browsed for parts.

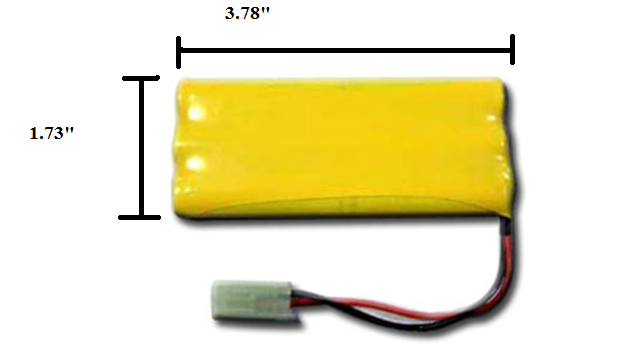


Figure 39: 7.2 V, 700 mAH NiCd battery (Reprinted with with permission from BatterySpace)

Figure 39 shows a potential NiCd battery that we found at BatterySpace. We found a 7.2 V NiCd battery from a website called BatterySpace. This battery had a very high capacity, providing 700 mAh. Since the current needed for most of our electronics was relatively low, we expected that we could get plenty of use out of this battery until it was time to recharge. The weight of this battery is only 4.2 oz, which was great. If we needed to mount this battery on the rear of our chassis, it would not put too much weight on the back of the vehicle. With the right charger, this battery could be up and running at full capacity in only 1.5 hours. This was beneficial because if we were to buy 2 of these batteries, one will have finished charging by the time the other battery runs low. The price of this battery would have allowed for us to do this, since they are only $6.95 each.

Even though this seemed like a great candidate for our battery, we decided to explore NiMH batteries as well, since they offered some advantages over the NiCd variety. From All-Battery.com we found this 7.2 V NiMH battery made by Tenergy shown below in Figure 40.

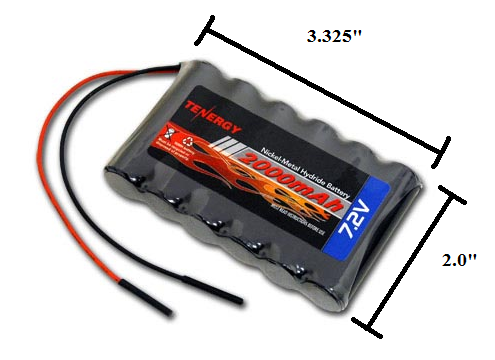


Figure 40: Tenergy 7.2 V 2000 mAh Side-by-Side NiMH Battery Pack (Reprinted with permission from All-Battery)

This particular model has 6 AA batteries at 1.2 V per cell that would provide a total of 7.2 V. The side-by-side configuration seen above in Figure 40 would enable us to get more current out of the battery. The greatest advantage we would have by using this battery was that these cells were not affected by any memory effect. This would have allowed us to get the full use out of our battery without having to worry if the last charge induced unwanted memory effects. This battery would also have given us 2000 mAh, but since we were not looking to use a lot of current, we hoped to get lots of use out of this battery due to the small current levels we needed for out electronics. Like the NiCd battery we considered, it also weighed 4 oz. Charging the battery would not take long either since the average time needed to do so was only 2 hours. The price of this battery was a little more than the NiCd battery, but this was to be expected since the overall performance was better. This model is only $3 more, but would cause us less trouble than the NiCd battery. Despite all of this research, we decided to buy a 2000 mAh 12 V NiMH battery off of Amazon because we were led to believe the camera we used needed a 12 V power source. However, this turned out to be false, but we decided to keep this battery because the voltage regulators we had could take an input voltage of up to 40 V and regulate it down without any problems.

**5.5.1 Battery Charger**

The next involved us finding a sufficient battery that was both efficient and affordable. This led us to a simple charger that could charge a 7.2-12 V NiCd or NiMH batteries. We found such a thing at BatteryJunction.com.

We decided to go with this charger because it did not limit us to just one battery size. If we wanted to upgrade or change the battery that would power our car motors, we would also be able to use this charger. Also, overcharging NiMH batteries could result in problems but this particular model had some good features available to prevent this. This charger only cost $10.05 which made us question the overall quality of the product, however, further research revealed that most of the chargers available cost around the same price.

BatterySpace also recommended a charger for our NiMH model battery that provided us with the range and features that we were looking for and it is shown below in figure 41.

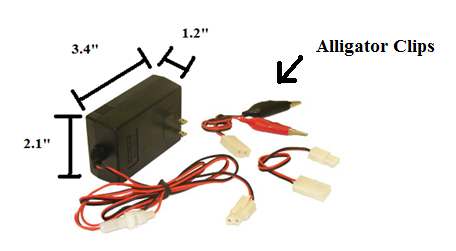


Figure 41: Compact Smart Charger (0.6A) for 7.2-12 V NiCd or NiMH Batteries (Reprinted with permission from BatterySpace)

The above figure shows the dimensions of this charger. It is quite compact and even came with alligator clips that allowed us to charge any battery that fit the chargers specifications. This charger also supported any model of NiCd or NiMH between 7.2 and 12 V. This offered us a lot of range and flexibility for any batteries we may have wanted to use. One great feature of this charger was that the charging process was automatically cut off when the battery was fully charged. This was really nice because the NiMH battery we decided to purchase could be damaged if it was overcharged. This charger also offered protection in case we accidentally attached the battery to the wrong terminals on the charger. This charger also used pulse and negative pulse technology which allowed for protection against overheating. Doing this also increased the battery’s life cycle, and since memory effect can be an issue depending on the battery, this was very beneficial. We purchased this charger and had great success with it.

**5.6 Antenna**

Since we researched micro transmitters and receivers it was now time to do the proper research on antennas that could possibly assist us with the transmission of data that our microcontroller received from the sensors to our laptop. There were many factors to consider while doing so. The only solid thing we knew about the antennas in question was that we wanted them to be surface mountable. This would require that the antenna be really small, and since our transmission frequency would have been quite high at 916 MHz for the antenna shown below in figure 42, this would make the required antenna size rather small. We found out that there were many options available to us that would allow us to achieve a surface mounted design. However, since we had no previous experience with antennas, we decided it was best to research each one to avoid disappointment.

**5.6.1 Ceramic Antenna**

The first antenna we stumbled upon was a 916 MHz ceramic surface mount antenna which is shown below in figure 42.

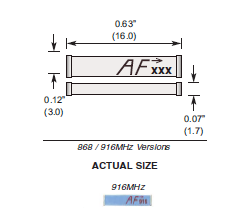


Figure 42: 916 MHz Ceramic Chip Antenna by Antenna Factor (Permission to reprint pending from Digi-Key)

This figure represents a blown up picture of the ceramic antenna. It is one of the world’s smallest high performance chip antennas and works with many applications which include Bluetooth, 802.11, home RF, Zigbee along with others and only costs $1.99. This antenna uses Low Temperature Cofired Ceramic (LTCC) technology, which embeds the antenna in a ceramic substrate. This technology enables the operating temperature to be fairly good. It would operate anywhere from -40 to +85 degrees C. It also has a 10 MHz bandwidth which would give us a decent range. It also had the impedance level required for our micro transmitter, which was 50 ohms. The antenna has 2 pins, one which would attach to the transmitter or receiver and one pin to be soldered down only to add extra support. No electrical connection would be needed. Despite the advantages, there were some concerns with using this model. For one, the antenna itself was really small. It was only 16 mm long and 3 mm wide. Mounting this on our PCB board would have proved to be too daunting. Its small size also raised the question of how fragile it actually was. We needed our possible antenna to be durable enough to withstand some uneven terrain and harsh environments and this may not have been what we were looking for. There were a lot of layout considerations that we would have to follow as well which also made it undesirable. The area underneath the antenna should be free of components, traces and planes in order for this antenna to work properly and even then, it would have required the use of a very small microstrip trace on the PCB.

**5.6.2 Planar Antenna**

The next model we considered was a 916 MHz planar antenna called the Splatch made by Linx Technologies shown below in figure 43.

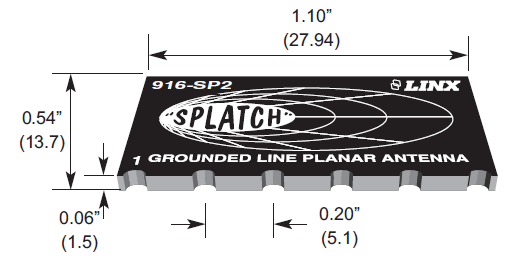


Figure 43: 916 MHz Planar Surface Mountable Antenna (Permission to reprint pending from AntennaFactor)

Pictured above in Figure 43 is what the 916 MHz Splatch antenna looks like. It was Surface mountable, just like we wanted and provided some desirable features. The Splatch was able to achieve good performance by using a grounded-line technique. It also matched the required 50 ohm impedance needed for the original micro transmitters and receivers we were looked at early in the design phase. The bandwidth of this antenna is 30 MHz, which was more than that of the ceramic antenna, which made this one more desirable in case we were unable to tune it to exactly 916 MHz. However, if we were able to mount this antenna in a way to gain optimum performance, we were looking at a VSWR of around 1.9. The VSWR rating of an antenna is a very important number. It stands for Voltage Standing Wave Ratio and is a function of the reflection coefficient which describes the power reflected from the antenna. Ideally, we would want this number to be really small and tuning our antenna to the center frequency of 916 MHz would have provided us with the best VSWR.

Using the equation VSWR = where Γ is the reflection coefficient, and the VSRW is around 1.9, we calculated that a little less than 11.1% of the power of the antenna would be reflected. This was important to us since this power would not be able to be transmitted to our receiver and thus would be no use to us. Research has shown us that a VSRW of less than 2 represents a great antenna match and thus was very desirable.

The cost of this antenna was also great. It would only cost us $2.08. The mounting for this antenna on our PCB would also be simplified. Like the ceramic antenna, no ground plane or traces were allowed underneath the antenna. This would not be a problem because the antenna is rather small, but not too small. It is 1.10” long and 0.54” wide and would not take up a lot of space on our board. It also offered plenty of vias that would go to the ground plane that we implemented on our PCB.

**5.6.3 Whip Antenna**

The last type of antenna we considered was the whip antenna. This kind of antenna consists of a single straight, flexible wire or rod that is usually mounted above the ground plane. These antennas are found in a lot of everyday items such as walkie talkies, cordless phones and are also used for car radios. Since what we tried to do definitely related to these applications, we decided that looking into this antenna would benefit us.

We found a 916 MHz whip antenna but it was not surface mountable. It connected to the receiver or transmitter by an RP-SMA connector. This would have posed a problem to us, but further research showed that there was a part available to us that would allow for an initial RP-SMA mount that could relay to a surface mountable part.

The part that we found that would have been used to mount the RP-SMA based antenna to our PCB board was rather small, which would have been great since all space on our board needs to be optimized. It supported frequencies up to 12.4 GHz and provided us with 50 ohms of impedance. We know how durable whip antennas are, since we all have one on our vehicles and they are able to endure harsh conditions, which was crucial because our vehicle is designed to go into disaster areas.

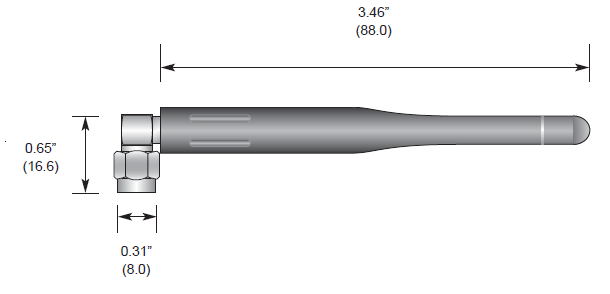


Figure 44: 916 MHz ¼ Wave Whip Antenna (Permission to reprint pending from AntennaFactor)

Figure 44 above shows what the whip antenna we considered would look like. From the figure, it is clear that we could not simply mount the antenna directly to the PCB board without the right connector. With this connector in place, we would have had a couple of different options for the direction in which our antenna could point in since it had a right-angle mount. This flexibility would allow us to find what angle would give our antenna the best gain, hence increasing the range that our robot would be able to travel while still relaying readings to our laptop. If the antenna did need to be perpendicular to our PCB board, a simple modification to the chassis would have allowed it to protrude outside of the body of our remote control car.

Although the center frequency for this antenna was at 916 MHz, there was a bandwidth of 150 MHz. This would have allowed us the ability to test out a few different frequencies and adjust if there was a lot of noise with the particular frequency we were testing. However, we would still have needed to stay near this center frequency for the best VSWR rating. At the center frequency, we had a typical rating of less than 1.7 on the VSWR scale, which made this the best antenna we had looked at up to this point. The length of the antenna was only 3.46” long which made it much easier to see and deal with than the much smaller chip antennas. In case we did need more space between the connector on the PCB and the antenna itself, we could have always got some wire that would also give us 50 ohms of resistance so we didn’t have to worry about how and where the PCB was laid out in order to accommodate the antenna. This antenna would have cost $7.39, which was considerably more money than the others. However, the added benefit of being able to move the antenna where we saw fit was very appealing to us. Also, the broader bandwidth this model offered would have given us even more flexibility that the chip antennas could not.

The biggest problem involving this antenna was the fact that all of these extra components would add noise to our signal, thus making those great specs meaningless; therefore, we decided we would not use the whip antenna for our design. Most of this research was done before we decided to use the XBee modules and once we decided to use them, any additional antennas, receivers and transmitters were not needed.

**5.7 Motor Controller**

A key component to our design was the motor controller. This was the part that enabled us to drive our motors. This section of the paper has us looking at different motor controllers that we felt could do a very sufficient job at guiding our robot through various disaster zones. The function that this component had to perform was crucial to making DZERV mobile so this part had be good quality. The functions that this part provided went beyond just making the motors move. It also adjusted the speed at which our robot moved, selected the direction of our vehicle (forwards or reverse), regulated the torque of the motor as well as had safeguards to prevent our motors becoming overloaded and therefore prevented it from burning out.

The remote control car that we used for the main body of our robot had two motors. One was used to steer the car, which was located in the front, and the other was used to power the rear wheels, which allowed for forward and reverse motions and was located in the back. In order to tap into the pre-existing system and make these motors function wirelessly, we needed a controller that could handle two separate motors. This would take some work and a little finesse but we knew that we were able to find one. We were able to find such a motor controller made by Polulu. It’s called the Qik Dual Serial Motor Controller and it is shown below Figure 45.

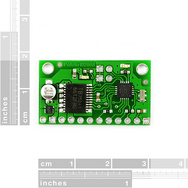


Figure 45: Pololu Low Voltage Dual-Serial Motor Controller

This motor controller set us back $24.95, but the benefits of using this component were well worth the money. It features a two-way serial protocol that features automatic baud rate detection up to 38.4 kbps. It also features two status LEDs that indicate to us that the part is on and whether or not any errors are detected during the initial startup process. We controlled the chip with Xbee wireless products that relayed the desired function to our controller which then activated the motors. This Pololu model has twelve pins and they listed and described below in table 14.

|  |  |
| --- | --- |
| Pin | Function |
| 1 | Ground (logic supply) |
| 2 | Vcc (logic voltage) |
| 3 | Rx (qik serial receive) |
| 4 | Tx (qik serial transmit) |
| 5 | Reset |
| 6 | Error |
| 7 | Motor 1, Positive/Negative Input |
| 8 | Motor 1, Positive/Negative Input |
| 9 | Motor 0, Positive/Negative Input |
| 10 | Motor 0, Positive/Negative Input |
| 11 | Ground (motor supply) |
| 12 | VMot (motor voltage) |

Table 14: Motor Controller Pin Layout

Table 14, located above, shows that this controller had the capability to control two motors. In addition, our XBee Din and Dout pins were connected to the transmit and receive lines on the motor controller. We used the 12 V supply that we ordered from Amazon to act as our motor supply voltage. The only real concern we had with this part was the logic supply voltage. According to the datasheet, it was not recommended that we use the same voltage supply and ground for both the logic voltage and input voltage. However, further reading revealed that this was permissible as long as we used a voltage regulator to differentiate the two. Since the operating range for our Atmega microprocessor is 1.8-5 V and since all of our sensors run at 5 V, the same supply for the sensors was connected to the logic supply pin without any worry of damaging the part. The serial input pin was connected to our Xbee receiver. This way a binary instruction could be converted to a voltage level which then transferred to the motors.

We experienced a lot of problems with this motor controller and its inconsistency forced us to make a decision to chance using it and maybe have it work or simply scrap it altogether. We went down the safer route and used a different motor controller. We used the Ardumoto Motor Driver which uses an H-bridge configuration and PWM signals to make the motors turn. The logic supply operated at 5 V which was very convenient for us because we had an open 5 V port due to us not using the Polulu controller.

Figure 46 below shows the rough schematic that we followed in order to get our robot to move. First we give a command on our keyboard to move in a certain direction. This information is then transmitted from our Xbee module to its receiver located on our robot. This signal is then sent to the motor controller where the data is processed in such a way that it then travels to our motors, allowing for the instruction to be implemented. If all goes well, the correct action by the robot is taken and we can then provide further instructions to guide our robot wherever we would like it to go.

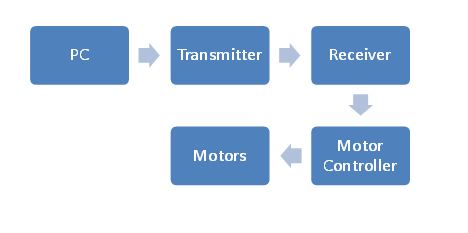


Figure 46: Order of Operation for Wireless Control

**6.0 Project Prototype Construction and Coding**

Of course, the real work for our project actually came from physically constructing and testing our robot. This was the part of the process where we finally stopped designing and started building. This was the most important stage of the process because it was where we found out whether what we had designed actually worked the way we designed it to. It was also the part where all of the learning that we had done throughout our undergraduate careers as engineering students culminated. It was this portion of the Senior Design process that separates the students from the engineers. It was this portion of the process that we’d been waiting for, and it was here that we were able to put our abilities to the test in order to proceed with building something that we had designed, and we ended up with a fully-functional prototype that we demonstrated before our Senior Design committee.

**6.1 Strategic Locations**

Obviously, one of the major things we needed to look into before settling down to begin construction on our project was where we would actually meet up to work on this robot. Clearly, a centralized location was the best option for the three of us, so neither one of us would need to commute too great of a distance to work on the project. Secondly, we needed a place that had the tools that we needed in order to put our project together. This included a soldering iron with multiple attachments, a variable power supply, multimeters, oscilloscopes and various tools to strip wire. While the absence of some of these tools did not necessarily hinder us from working on the robot, they certainly were a welcome addition to our working environment. Therefore we settled in a few places where we could work.

**6.1.1 EECS Senior Design Laboratory**

One of the first places that we considered doing a majority of our construction work was in the Senior Design lab for electrical and computer engineering students located on the fourth floor of the Engineering I building at UCF. The reason for this was that the lab contained many workstations for senior design groups, and all of the equipment we possibly needed for building our robot. This also satisfied our location requirement, since we all spent a great deal of time on campus at UCF as it stood. This in turn provided an extra layer of convenience for our needs. In addition, working on our project on campus in the presence of other engineers gave us the opportunity to ask for the opinions of other electrical engineering students in order to seek advice for troubleshooting issues with our project. This proved to be quite invaluable as we went through the process of building and testing our robot. While we were certain that we would be spending most of our time in the Senior Design Lab, we had also taken into consideration that we wouldn’t spend all of our time there. It became necessary to investigate other possibilities for build locations.

**6.1.2 Group Members’ Homes**

We therefore looked into what we would be able to do as far as working from the homes of our group members. After a few minutes of discussion, we learned that Michael lived

in the Sterling apartments right across the street from campus. This was an ideal place for us to get work done since it was certainly a convenient location for Michael, and was a place that Robert and Marcial were able to reach easily from almost anywhere. The only issue was that because this was an apartment, and other roommates lived there, the space required for the build was sometimes limited. However, after discussing our situation with Michael’s roommates we decided that in the event we weren’t able to work in the senior design campus that this was a convenient location for us to work at. We decided as a group that these two locations were all that we needed in order to build our project.

**6.2 Parts Acquisition**

**6.2.1 Vendor Selection**

Another one of the more important decisions in the overall design process was determining where parts would be purchased. Where would we purchase all of our parts from? Would we buy them in a store, or online? What kind of lead time would there be on the parts that we needed? Would we be able to ensure that we’d be able to have all our parts in our hands in a timely manner so that we could have time to physically build the robot and test it out before our project is due? These were all questions which we now had answers to, since a great deal of research was done in order to determine what would be the best way to go about purchasing and acquiring our supplies. For starters, we had an incredibly wide variety of vendors from which to choose. We purchased and sampled parts directly from manufacturers, such as Texas Instruments for our voltage regulators and various other parts. However, we decided that it was a lot simpler and much less of a hassle to order as many parts as possible from distributors such as SparkFun or Digi-Key.

This allowed us to lump many components into a single order and allowed us to have the components all shipped and/or delivered at or near the same time. Additionally, due to the fact that we consolidated many separate orders from many different vendors into very few orders, it ultimately saved us a great deal of time and money overall during the building process. We further investigated the companies we purchased parts from, and after a great deal of discussion, we decided to look at a few different vendors to purchase all the components that we needed.

Our reasoning behind this was very simple. SparkFun is a very highly-trusted vendor for electrical components when it comes to microprocessors, LEDs, among other things, including modeling kits for amateurs. This alone, combined with many word-of-mouth referrals, and some of our members’ own personal experience with SparkFun all led to the decision for us to purchase some components from them. Digi-Key was another vendor that purchased components from. Because SparkFun and Digi-Key are such well-trusted sources, we were also ensured that our parts would arrive quickly as needed, and this reliable service saved our group from an incredible amount of unnecessary stress during the last few weeks of the construction phase because of it. We ran the possibility of not receiving parts on time or at all if we used unknown vendors, so we stuck with these two vendors whenever possible. Also, based on personal experience, it had been noted that SparkFun’s pricing was actually quite competitive, and therefore saved us a great deal of money in the long run. However, we were not able to purchase all of our components from distributors like SparkFun and Digi-Key. One such component was the camera that we used simply because SparkFun did not carry it so we bought it off of Amazon instead.

**6.3 Printed Circuit Board Vendors**

Another major component of our project was the printed circuit board on which we placed our components. Picking a vendor was a significant decision that our group had to decide on, therefore, we did a great deal of research on the companies that could make our board. None of us had any experience with PCB software before this project, but we decided to use Eagle since members of that company came to UCF and did a demonstration on how to use it.

**6.3.1 ExpressPCB**

One of the first companies that we came across in our research was ExpressPCB. This company offered quite a few features that would be useful for us in designing our board. For instance, their website featured a section containing resources on getting started with PCB design. In addition, ExpressPCB offered their own line of software tools for designing circuit schematics, as well as software for actually laying out the PCB itself. Additionally, these software tools were completely free of charge, and the ExpressPCB software even had a built-in ordering functionality so that users can actually use the software to order their board after laying it out.

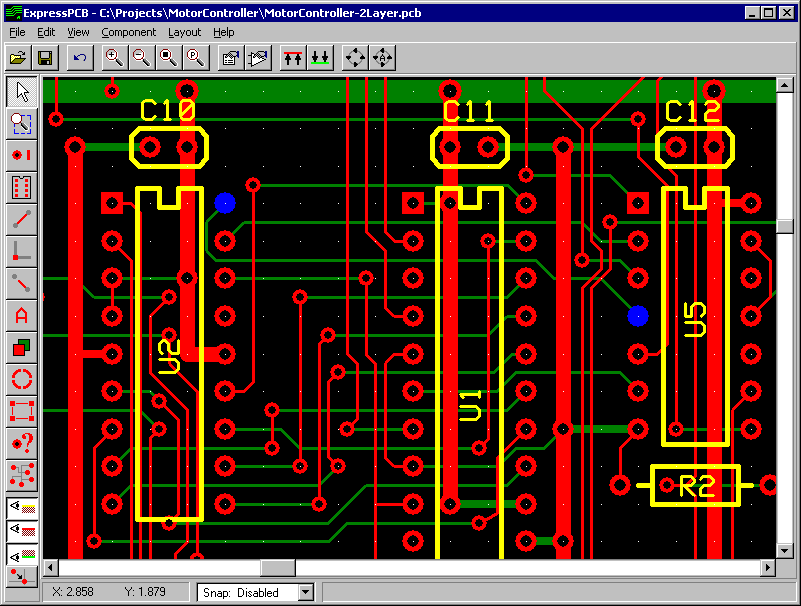


Figure 47: PCB Layout Software (Reprinted with Permission of ExpressPCB)

The figure above demonstrates some of the functionalities of the ExpressPCB software. There are built-in functions that allow the user to choose components and place them on the board. Here, the user can also add traces, and specify how wide they need to be, among other things. This software also offers countless component libraries in order to facilitate the board layout process.

Additionally, ExpressPCB also offered quite a fairly wide range of solutions that could fit almost anyone’s PCB needs. They offered their MiniBoard line, which had two varieties: the MiniBoardPro, and the Standard MiniBoard. They both come standard with two layers and plated through holes. The major difference between the Standard MiniBoard and the MiniBoardPro was that the MiniBoardPro included solder mask layers and a white silkscreen layer, which serve to give the boards a more professional look. There was also a sizeable difference in pricing, as the two-layer MiniBoardPro comes with a set of three identical boards for $75, and the Standard MiniBord comes with a set of three boards for a fixed price of $51. ExpressPCB additionally offered a four-layer version of the MiniBoardPro with layers for a ground plane and power plane. However, we only needed a two layer board to realize our components. Also the additional planes in the four-layer board would cause a fairly significant increase in price. A set of three of these boards would cost $98. The major disadvantage of these MiniBoard models was that the size must be limited to 3.8 by 2.5 inches, which meant that we would need to cram all of our parts into this small space and this was not possible. An alternative to this would have been to go with one of their standard boards, which had much fewer limitations. Their Standard boards are exactly like the Standard MiniBoards, but these are scalable, with a pricing structure based on the physical area of the board. This gave us more room to work with as far as the size of the boards was concerned.

One of the biggest factors we considered when choosing a PCB vendor was the estimated lead time. For most PCB models, they offered to have the boards shipped the following business day, which was actually a considerably fast shipping time.

**6.3.2 Sunstone Circuits**

Another PCB manufacturer that we investigated was Sunstone Circuits. We first stumbled across this manufacturer by accident as we began our search, but this particular manufacturer offered a great deal of support for their customers’ design needs. Similar to ExpressPCB, they offered a design tool called PCB123 that enabled users to lay out their own PCB and then order it using the designs they’ve created. It even included a built-in tool that allowed customers to calculate the total cost of the board before they ordered. The remarkable thing about this software was that it was completely free to use, and included much more functionality than many of the other leading PCB CAD software suites available on the market such as CadSoft’s Eagle software. This would definitely prove useful in laying out our board, as we were looking to reduce costs wherever we possibly could, and having free CAD software would be helpful for doing this. The software was also very robust, and featured a Design Rule Check tool that checked to ensure that the board being laid out met many essential design criteria that are crucial to PCB design.

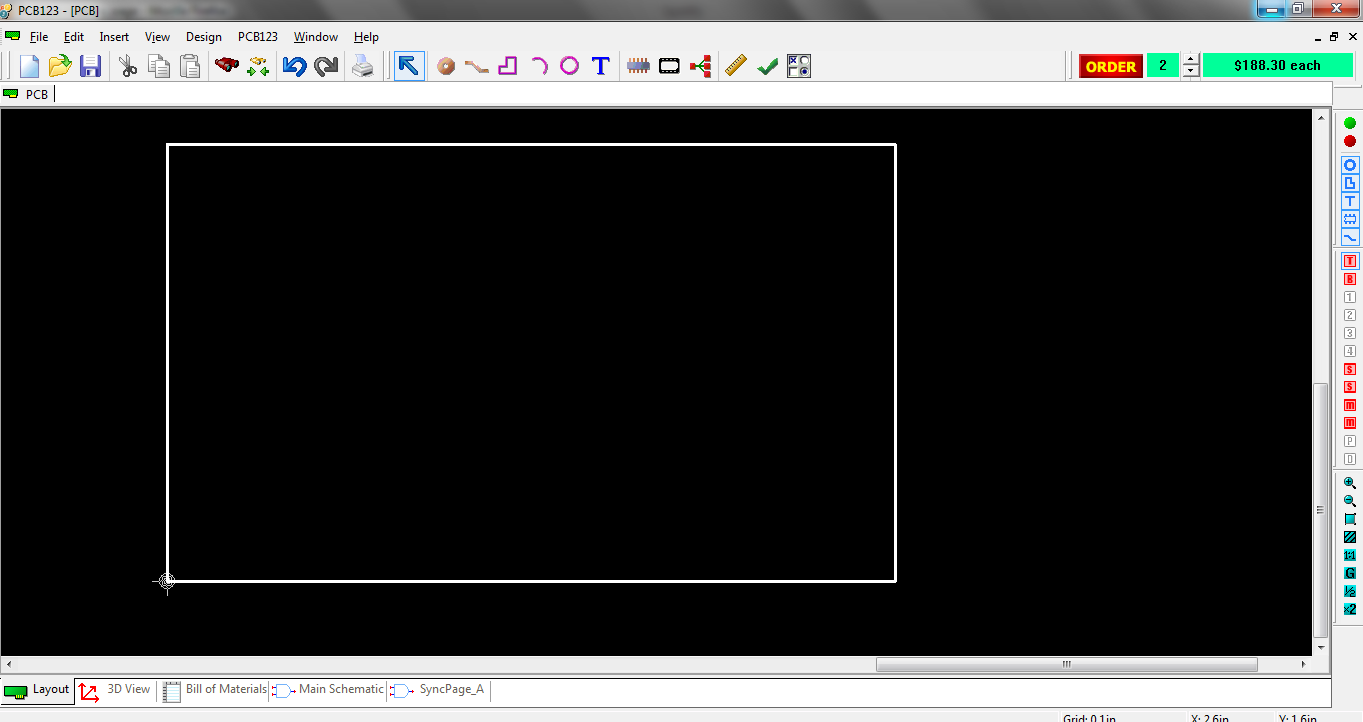


Figure 48: Screen Capture of the PCB123 Software

Above, figure 48 shows a screenshot of the PCB123 environment, showing all of the various design features, including the Design Rule Check tool, the various CAD tools, and even the per-unit cost for such a board. Unfortunately, as can be seen in the screenshot, pricing with Sunstone was a bit on the expensive side. While the board we’d eventually order would be much smaller than the one depicted in the image, we had found through experimentation, that we’d still be spending roughly $180 per board with a two board minimum via this software’s ordering tool. This proved to be far too expensive for our design, and as a result, we collectively decided that we would no longer be taking

Sunstone Circuits into consideration for our PCB design.

**6.3.3 Advanced Circuits**

The final PCB manufacturer we investigated was Advanced Circuits. Based on the experiences of students who had utilized this manufacturer before, we decided that that this could very well be the best option for us. In addition to the word of mouth referrals that we had received for this manufacturer, we had also discovered that there was a discount available for students. Being the thrifty Senior Design group that we were, we decided that this was worth a further investigation, and found that we could actually receive up to $150 off the total cost of our board just by purchasing it from them. This alone was an incredible reason for us to go with Advanced Circuits. While they did not offer their own free PCB CAD software like the other manufacturers we looked into, they did offer assembly service, either via customer-supplied components, or as a turnkey service, meaning that they would supply the parts. While this would be convenient for us, we deemed this to be unnecessary for a few reasons. First, this would likely drive up the costs for us drastically. Additionally, having someone else assemble the printed circuit board for us would almost entirely defeat the purpose of the Senior Design process, since assembling a circuit board is a skill that we as electrical engineering students needed to acquire throughout the course of this process.

**6.3.4 Final Decision**

After a great deal of research and consideration, we had narrowed down our choices to either ExpressPCB or Advanced circuits, as these were the two manufacturers that seemed to provide the most promise to fulfill our needs. In the end, we decided that cost was our most important deciding factor for the PCB manufacturer, as it had been for a vast majority of other things for our project.

While we were aware that lower cost could have meant that the quality of the work done may not have been as high as if we had spend a little more money to go with a more expensive vendor, we realized that the quality of the Advanced Circuits’ boards would suit our application, and we were certain that this would fit our needs. Based on the experiences of other senior design groups, we felt that their work and their pricing were perfect for our robot. More specifically, we decided not to go with the MiniBoardPro model, since this would have give us the look that we had professionally done. Additionally, the pricing for the Advanced Circuits board fit well within our budget.

**7.0 Testing**

This part of our document was set aside to discuss the various testing methods and procedures used to test every individual component of our design. This included checking the durability, quality and overall performance of each piece. Each individual component had to be checked first before it was to be integrated into something else. If a problem was found in this early stage, we were able to troubleshoot it so much easier and either debug any coding issues or trace any hardware issues. A series of tests were done in order to ensure that our parts were in working order. Charts were set up to document the progress of what we should have seen compared with what we did see and if standards were not met, further troubleshooting was done until it had been decided that it was not the wiring or coding that was incorrect, but simply due to the parts limitations.

**7.1 Chassis**

The testing of the chassis proved to be very important. We knew that the initial chassis was stable, since it was manufactured to be that way, but during the building process, we had to modify it. After receiving all of our parts and having started assembling them, it was quickly determined that the space underneath the chassis did not provide enough room to house all of our components. In order to make everything fit, we built an acrylic platform that attached to the robot’s preexisting chassis using nuts and bolts which guaranteed that it would not fall off thus resulting in damaged circuitry. Further modifications included installing a thicker bolt in the middle of the acrylic that we could then screw the camera onto. In order to make sure the robot was not top heavy, we made bolt just high enough so that the camera could see over all of the electronics.

First, we made sure that the robot could support its own weight by picking it up and moving it around. This provided a quick check to see if any components came lose or if there was any violent shifting of components underneath. Once this was complete, we checked to see how steep of an incline our robot could handle. This included driving it up a hill and driving on an incline. Although we were not completely certain of the threshold of incline our robot could handle while it is in a disaster area, at least we could make a better educated guess before we sent our robot into the field since we could see the projected area on our laptop screen from the camera. Next, we checked to see how much weight our robot could support and still function to our standards.

The following data in table 15 shows our testing procedure as well as our expected results.

|  |  |  |
| --- | --- | --- |
| Test | Description | Expected Result |
| Carry and Shake | Pick up vehicle and  shake to see if  components are loose | No parts will move  any considerable  amount |
| Forward Motion  Incline | See what kind of  incline our robot can  overcome | Successful climb  resulting in further  exploration of disaster  area |
| Sideways Incline | See what kind of angle  our robot can handle  while driving on steep  slope | Continue motion  regularly unless robot  topples due to too  much incline |
| Weight Threshold | See what kind of  weight our robot can  support while still  being able to  function | Either the robot will  continue without a  problem or debris will  obstruct our motion |

Table 15: Chassis Testing

**7.2 Speed Test**

There may come a time when our robot is investigating an area where we may need to get out in a hurry. Other times, we may need to go very slow in order to overcome obstacles or dodge them altogether if the space allotted allows us to do so. Our initial remote control car was rated at 7 mph at full speed, however, the added weight to the initial design did not allow us to run at 7 mph. Determining the speed took a little effort on our part. First, we measured a set distance somewhere between 30 and 50 feet and marked it. Then placed our robot at the start and measure the time it took for it to cross the finish line. We then took the average speed in ft/s and converted this number to mph. We were able to achieve a very slow speed of 2 mph on flat terrain. We wanted this test to be done on flat surface, dirt, grass and gravel, since these are all possible environments our robot should have been able to endure, however the robot would not move at all in any other terrain. We concluded that the stock motors that we used were simply not powerful enough to propel our robot with the weight we had on it.

Shown below in table 16 is our testing procedure we wanted use to determine speed in different environments for different situations.

|  |  |  |
| --- | --- | --- |
| Test | Description | Expected Result |
| Speed (Flat Surface) | Max and min speeds  will be determined  on flat surface | Max speed: 5-6 mph  Min speed: 1 mph |
| Speed (Grass) | Max and min speeds  will be determined  in grass | Max speed: 4-5 mph  Min speed: 1 mph |
| Speed (Dirt) | Max and min speeds  will be determined  in dirt | Max speed: 4-6 mph  Min speed: 1 mph |
| Speed (Gravel/Rock) | Max and min speeds  will be determined  in gravel or rock | Max speed: 4-5 mph  Min speed: 1 mph |

Table 16: Speed Testing

**7.3 Battery Life**

It was important that we tested the lifespan of the battery that powered our motors and electronics. To test the battery for the motor we will simply drained the battery three separate times using varying duty cycles for our motors, one being full speed, the next being minimum speed and the last being somewhere in between. The results determined from this test provided the basis for how long our robot would run and for what duty cycle. We knew our battery was no longer supplying the necessary power when our sensors failed to gather and/or deliver results to our GUI.

.

Table 17 shows the test procedure we used to rate our battery life.

|  |  |  |
| --- | --- | --- |
| Test | Description | Expected Results |
| Battery Life: Full  Duty Cycle (Motors) | Constant motion  until battery is  drained | 30-45 minutes |
| Battery Life: Low  Duty Cycle (Motors) | Constant motion  until battery is  drained | 1-1.5 hrs |
| Battery Life:  Moderate Duty Cycle  (Motors) | Constant motion  until battery is  drained | 45 minutes to 1 hr |
| Battery Life  (Electronics) | Constant use until  battery is drained | 3 hrs |

Table 17: Battery Testing

Final testing revealed that we could get 30 minutes out of our 12 V battery when we ran

every component of our project.

**7.4 Battery Charging**

Not only was it important to know how long our battery would last, it was also important to find out exactly how long it took for our battery to recharge. For our 12 V NiMH battery rated at 2000 mAh, the expected charging time should be right around 2.5 hours. The remote control car that we purchased did not give a specified time that it took for the battery to charge but since we didn’t use this battery at all, this information was not important. There is a status LED on our 7.2-12 V charger so we simply let the battery die and started charging it, and we recorded the time it took to fully charge. It took about 3 hours to charge. Table 18 shows the expected results for our battery charger.

|  |  |  |
| --- | --- | --- |
| Test | Description | Expected Results |
| 6V Stock Motor Battery | Measure exact time  it takes to recharge  battery. | Not necessary |
| 12V Electronics  Battery | Measure exact time  it takes to recharge  battery. | 2.5 hours |

Table 18: Battery Charger Testing

**7.5 Video Camera**

The video camera was our sole way of determining the next direction we sent our robot in, so checking to see if it the stream was accurate was crucial. We needed to have a data rate good enough in order to allow for the closest real-time stream as possible. We then tested the quality of our stream by placing our robot in an environment where we did not need the camera to see where the robot was going. Doing this enabled us to see how much lag we experienced and this also allowed us to observe the overall video. The following information in Table 18 outlines the tests that were taken to test certain specs of our camera.

We originally wanted our camera to have the ability to pan and tilt while it was on our robot, but the wireless camera we used did not have this ability. We had to set up the initial angle and leave it like that. Thus the only way to change it was to have direct physical interaction with the robot. To achieve the best angle, we angled the camera parallel to our robots wheelbase so that it would always be looking ahead. Table 19 shows what our expected results were during the testing phase.

|  |  |  |
| --- | --- | --- |
| Test | Description | Expected Results |
| Data Rate | We must figure out  quality of our video  stream to laptop | Stream will be good  enough to make real  time decisions. |
| Visibility | We must make sure  that the environment  we are in can be  properly represented. | Environment will be  recognizable. |
| Camera Angle | We must see if  proper angling of  camera can be  achieved. | Camera angle was fixed |

Table 19: Video Camera Testing

**7.6 PCB Testing**

This was the most crucial part of our testing. If our PCB board was not in working order, all of our electronic components would not work in unison, which would result in lackluster performance in our robot. The testing of our board took place once all the components were soldered in their correct place on the board. First, we used a multimeter to do continuity checks before we powered up the board. Once this was completed, we turned on the switch that powered the board and checked for the expected voltage levels and used our best judgment to see if these readings made sense for the individual component that was being tested. If there happened to be a problem with any connection, we would attempt to salvage it as best as we could. This would be a last ditch effort, however, since individual parts were already soldered to the board and making adjustments would be really difficult to do.

If we did have a problem with our initial PCB, actions had to be taken in order to fix it. Therefore, the utmost care had to be taken before we decided to solder all of the components to our board. Once we determined that our PCB layout was correct we proceeded to the proper testing. The information in Table 20 shows what the expected results were for our PCB tests.

|  |  |  |
| --- | --- | --- |
| Test | Description | Expected Results |
| Check for loose  Components | Make sure that no  component on the  board shifts or falls  off | Parts should remain  soldered to PCB board |
| Test for input and  output voltages and/or  currents and continuity | Voltmeter or ammeter  will be used to make  sure parts are working | Meters should pick up  readings that make  sense based on the  part |

Table 20: PCB Testing

Unfortunately, we ran into a lot of problems with our original PCB. After powering it up, it was apparent that our original motor controller didn’t work, however, this motor controller was extremely buggy from the get go, so we did not blame the PCB design for this issue. However, we already had headers soldered onto our board that only fit this particular controller. Therefore, we had to scrap the idea of having a motor controller on this board. We ended up using the Ardumoto Motor Driver Shield to successfully get our motors to work on a consistent basis, so we built a separate circuit that housed this controller and an XBee module that sent and received instructions for our motors.

For unknown reasons, one of our XBee modules fried after having successfully tested it multiple times. Therefore, as an act of desperation, we rebuilt our whole circuit on a separate solderable perf board since we only had a couple days before our final presentation and we were all extremely grateful that this new board worked just fine, despite having the exact same layout as our original PCB.



Figure 49: Rebuilt Final Main Circuit (Created by Group)

**7.7 Wireless Control Testing**

Our XBEE transmitter and receivers both had to be configured in a proper manner to ensure successful communication with each other. The following tests were performed in order to ensure our wireless modules are properly configured, communicating, and working.

|  |  |  |
| --- | --- | --- |
| **Test** | **Description** | **Expected Result** |
| Base Computer Communication | Send simple code to transmitter connected to base computer via USB | Code should be sent to receiver and seen in the robot |
| Motor Controller Communication | Send forward command to receiver | Wheels should move forward |
| Module Communication | Test the communication between the TX and RX on separate computers | Data is properly sent and received |
| Range | Find the maximum range of communication between TX and RX | 200 feet |
| Climate Control | Run the modules in various temperatures and environments | Both the RX and TX should still properly transmit information |
| Temperature Control | Run the modules in various environments and temperatures | Measured voltage and current should still be as expected |

Table 21: Wireless Control Testing

The XBees proved to be great wireless devices. Using our 3 module set up, we were able to successfully steer and power the motors as well as transmit and receive sensor data. We were able to successfully transmit data from 150 feet without any problems and we were quite happy with that.

**7.8 Sensor Testing**

In order to test our sensors, we utilized an Arduino mega1280 development board, which one of the group members had acquired from a previous project. All of our sensors utilized through-hole technology. The pressure sensor we used was a surface mount part, however, since it was so small, we kept it on a breakout board that we purchased in order to test this sensor. Meanwhile, we utilized a breadboard and the Arduino to test the other sensors. Once we connected the sensors onto our breadboard and with the Arduino, we wrote a series of simple code modules in order to test the sensors. Some of these tests can be seen in table 22 below.

Some of the sensors did not require a breadboard for testing purposes, rather, they could be purchased on a breakout board. Aside from this, our group furthered our soldering and PCB layout knowledge, and created our own simple breakout boards using simple Vector boards rather than complicated PCBs which involved traces and footprints. As previously mentioned, for simple and basic testing purposes, we used an Arduino to test the sensors and make sure that the sensor itself worked properly. We decided to use an Arduino due to the fact that it had fifty four digital I/O pins.

|  |  |  |
| --- | --- | --- |
| **Test** | **Description** | **Expected Result** |
| Temperature Sensor Heat | Hold flame near the sensor | We should see a spike in the temperature output |
| Temperature Sensor Cold | Place sensor in freezer for 5 – 8 seconds | We should see a dramatic decrease in the temperature output |
| Humidity Sensor High | Place sensor in bathroom after taking a warm shower | We should see a high humidity output |
| Humidity Sensor Low | Place the sensor near a running dehumidifier | We should see a low humidity output |
| Range Finder Distance | Hold object 5 inches away from range finder | We should see a distance output of about 5 inches |
| Range Finder Distance | Hold object 10, 15, 20, 25, 30 inches away from range finder | We should see a distance output of about 10, 15, 20, 25, and 30 inches |

Table 22: Sensor Testing

**7.9 Microcontroller Testing**

Since the microcontroller is essentially the brain of our robot, we had to ensure that it was properly working through various tests. In order to test, utilized various tools and equipment. We used the Arduino Uno to load code onto our chip and a breadboard to debug any problems with our microcontroller. When we conducted testing, we loaded code onto our microcontroller and hooked up our sensors to the breadboard and checked to make sure the date we received made sense.

Table 23 depicts some of the tests that were executed on our microcontroller before we permanently had it soldered onto our PCB.

|  |  |  |
| --- | --- | --- |
| **Test** | **Description** | **Expected Result** |
| Power On | Once we made our connections, power it on | Measured voltage and current should not exceed expected calculations |
| Simple Command | Send a command to power on a LED | The LED should light |
| RX and TX Connection | Is our base computer recognizing the transmitter attached to the MCU | We should see the receiver listed under a port on the base computer |
| Climate Control | Run the MCU in various environments and temperatures | Measured voltage and current should still be as expected |
| Temperature Control | Run the MCU with a constant current and voltage for 8-10 minutes | Voltage, current, and temperature should not exceed datasheet specs |

Table 23: Microcontroller Testing

After all of the testing and integration of parts, we finally had a working prototype. The

figure below shows the final product.

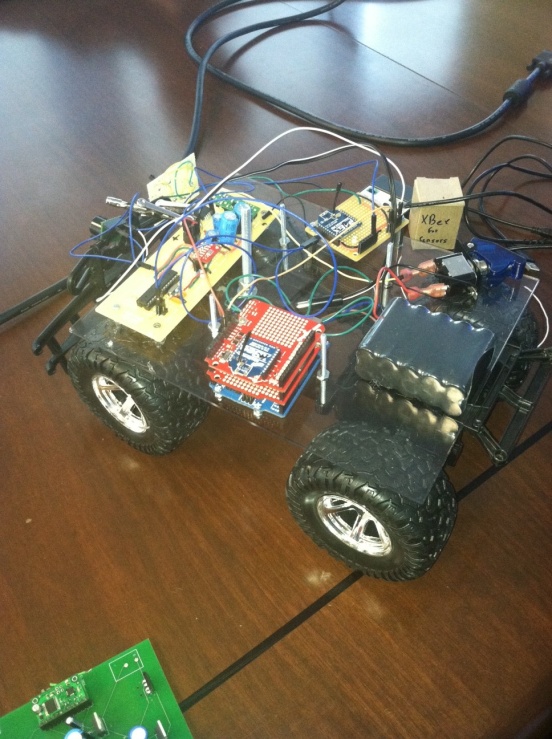


Figure 50: Final Robot Prototype (Created by Group)

**8.0 Bill of Materials and Responsibilities**

**8.1 Bill of Materials**

Due to the fact that our project was sponsored, the cost of the robot was split among the three group members. However, since our project lasted from Spring through Fall as opposed from Spring through Summer, we had more time to find sponsorship, but we were unsuccessful. Receiving sponsorship would only have brought upon a positive impact on our project, but despite this minor issue, we carried on. Logic told us that the more money we had, the better the quality of the components we’d be able to purchase for our robot.

As previously mentioned, one of our main goals was to consolidate companies from which we would order parts from. Luckily, we narrowed down our component ordering to five online stores from which we ordered and two stores which we physically visited to buy components: Sparkfun, Digi-Key, Amazon, Mouser, All-Battery, Skycraft and RadioShack. Table 24 below depicts the parts that we acquired for our project and shows the individual and final cost.

| Item | Unit Cost | Budget |
| --- | --- | --- |
| Microcontroller/Arduino Uno | $29.95 | $78.00 |
| Car | $29.99 | $29.00 |
| Camera | $40.00 | $99.99 |
| Rangefinder | $25.95 | $25.95 |
| Pressure Sensor/Breakout Board | $29.90 | 11.95 |
| Temperature Sensor | $25.95 | $25.95 |
| Motor Controller | 2 x $25.95 | $64.95 |
| Xbee kit | $79.99 | $79.99 |
| Power Supply/Charger | $32.00 | $29.90 |
| Circuit components (resistors, diodes, etc.) | $60.00 | $42.94 |
| PCB Construction | $75.00 | $100.00 |
| Miscellaneous (solder, wire, etc.) | $150.00 | $100.00 |
|  |  |  |
| TOTAL | $630.93 | $688.62 |

Table 24: Project Cost Breakdown

As we can see from table 24 above, our project averaged about $630. From this cost, each group member contributed about $210 to the building of our robot. As previously mentioned, we sought sponsorship throughout the summer months to help alleviate some of the costs, but were ultimately unsuccessful.

In addition to the project cost breakdown table above, the following pie chart in figure 51 depicts the price breakdown for our project. It depicts the percentage distribution from the table above. Every part and component that was integrated into our project is represented as a percentage in order to depict a distribution of the cost.

Figure 51: Cost Breakdown

**8.2 Responsibilities**

Because each group member had his own strengths and weaknesses, we equally divided and distributed portions of the project as best we could to work on things according to our strengths. Overall, our goal was to have all the group members contribute in all aspects of the project, whether it was coding the microcontroller, writing the GUI for the user, wiring a testing development board, or soldering the final PCB. We believe we were successful implementing this goal. Another goal we set out for ourselves was to gain as much knowledge as possible in the fields in which lacked comfortable knowledge in. The following chart in figure 51 depicts an approximation of the amount of work and responsibilities that were divided among the group.

Figure 52: Group Responsibility Chart (Created by Group)

**8.3 Project Milestones**

In order to properly and efficiently complete this project in a timely and organized manner, our group had to devise some sort of milestone chart. Creating such a chart acted as a reference to keep track of all the tasks we accomplished over the course of the past 11 months. We made this approximation as accurate as possible in order to avoid any unforeseen pitfalls or shortcomings. The milestones are listed in table 25 below and it contains all aspects of the project including designing, building, testing, and writing of all the documentation.

| **ID** | **Project Name** | **Days** | **Start** | **End** |
| --- | --- | --- | --- | --- |
| **1.0** | **Senior Design Plan** | 210 | Jan. 9 | Dec. 3 |
|  |  |  |  |  |
| **2.0** | **Writing phase** |  |  |  |
| **2.1** | Initial Document | 8 | Feb. 7 | Feb. 14 |
| **2.2** | Half Final Document | 28 | Feb. 14 | Mar. 12 |
| **2.3** | Final EEL 4914 Doc | 30 | Mar. 12 | Apr. 20 |
| **2.4** | Final Document | 105 | Aug. 20 | Dec. 3 |
| **2.5** | Final Presentation | 105 | Aug. 20 | Dec. 3 |
| **3.0** | **Research Phase** |  |  |  |
| **3.1** | Hardware | 76 | Feb. 7 | Apr. 23 |
| **3.2** | Software | 76 | Feb. 7 | Apr. 23 |
| **4.0** | **Design Phase** |  |  |  |
| **4.1** | Sensors/Electronics | 69 | Feb. 14 | Apr. 23 |
| **4.2** | PCB Layout | 91 | Apr.23 | Jul. 23 |
| **4.3** | Software(Comm.) | 69 | Feb. 14 | Apr. 23 |
| **4.4** | Ordering/Receiving Parts | 91 | Apr. 23 | Jul 23 |
|  |  |  |  |  |
| **5.0** | **Assembly Phase** |  |  |  |
| **5.1** | Electronics Mounting/Soldering | 25 | Aug. 20 | Sep. 14 |
| **5.2** | Electrical Connections | 25 | Aug. 20 | Sep. 14 |
| **5.3** | Server/Software | 50 | Aug. 20 | Oct. 9 |
| **5.4** | GUI | 20 | Sep. 20 | Oct. 10 |
|  |  |  |  |  |
| **6.0** | **Testing Phase** |  |  |  |
| **6.1** | Microcontroller | 52 | Oct. 9 | Nov. 30 |
| **6.2** | Camera | 52 | Oct. 9 | Nov. 30 |
| **6.3** | Sensors | 52 | Oct. 9 | Nov. 30 |
| **6.4** | Navigation | 52 | Oct. 9 | Nov. 30 |
| **6.5** | Software | 52 | Oct. 9 | Nov. 30 |
| **6.6** | Whole Unit | 52 | Oct. 9 | Nov. 30 |

Table 25: Project Milestones

**Appendix A: Bibliography**

“An Introduction to Infrared Technology:Applications in the Home, Classroom, Workplace, and Beyond ...”

<http://trace.wisc.edu/docs/ir_intro/ir_intro.htm>

“IR Vs. RF: Which is the best for my needs?”

<http://www.aclasstechnology.com/downloads/IRvsRF.pdf>

# Which is the best frequency to use for my application: 315MHz, 418MHz, 433.92MHz, or 900MHz?

<http://www.linxtechnologies.com/support/knowledgebase/rf-modules/which-is-the-best-frequency-for-my-application/>

“Understanding Gear Reduction”

<http://www.teamdavinci.com/understanding_gear_reduction.htm>

“XBee Wireless Communication Starter Kit”

<http://www.trossenrobotics.com/p/Xbee-Communication-Starter-Kit.aspx?feed=Froogle>

“Choosing a Motor: DC, Stepper, or Servo – Free How-to Robot Construction Article”

<http://www.robotoid.com/howto/choosing-a-motor-type.html>

“DC Motor: High Torque Mini DC Gear Motor 3-12V, 5-25 rpm for Hobby / Robots”

[http://www.batteryspace.com/dcmotorhightorqueminidcgearmotor3- 12v1300rpmforhobbyrobots.aspx?gclid=CI39kerToa8CFQrCtgodp3JlaA](http://www.batteryspace.com/dcmotorhightorqueminidcgearmotor3-12v1300rpmforhobbyrobots.aspx?gclid=CI39kerToa8CFQrCtgodp3JlaA)

“Stepper Motor Control – AVR Tutorial”

<http://extremeelectronics.co.in/avr-tutorials/stepper-motor-control-avr-tutorial/>

“How to Build a Robot Tutorial”

<http://www.societyofrobots.com/batteries.shtml>

“Alkaline Battery: Advantages and Disadvantages”

<http://www.doityourself.com/stry/alkaline-battery-advantages-and-disadvantages>

“Lithium Battery”

<http://en.wikipedia.org/wiki/Lithium_battery>

“Lead Acid Batteries”

<http://www.mpoweruk.com/leadacid.htm>

“Nickel-Cadmium Battery”

[http://en.wikipedia.org/wiki/Nickel%E2%80%93cadmium\_battery#Comparison\_w](http://en.wikipedia.org/wiki/Nickel%E2%80%93cadmium_battery#Comparison_w_other_batteries)

[\_other\_batteries](http://en.wikipedia.org/wiki/Nickel%E2%80%93cadmium_battery#Comparison_w_other_batteries)

“NiCd Battery Pack: 7.2V 700mAH for RC Cars: Nikko / Radioshack etc”

[www.batteryspace.com/nicdbatterypackone72v700mahforrccarsnikkoradioshacketc.aspx](http://www.batteryspace.com/nicdbatterypackone72v700mahforrccarsnikkoradioshacketc.aspx)

“Nickel Metal Hydride NiMH Batteries”

<http://www.mpoweruk.com/nimh.htm>

ANT-916-CHP-T Datasheet

<http://www.antennafactor.com/resources/data-guides/ant-xxx-chp-x.pdf>

“VSWR”

<http://www.antenna-theory.com/definitions/vswr.php>

Energizer L91 Datasheet

<http://data.energizer.com/PDFs/l91.pdf>

## “RF Link Transmitter - 434MHz”

## <http://www.sparkfun.com/products/10534>

Stevens Institute of Technology Search and Rescue Robot Senior Design Project

<http://tiger.ece.stevens-tech.edu/08-09/grp16/Other%20Documents/EE423_Group16_FallFinalReport1.pdf>

“Robot to the rescue: University of Warwick students gear up for European championship”

<http://www.sciencebusiness.net/news/75691/Robot-to-the-rescue-University-of-Warwick-students-gear-up-for-European-championship>

“Microsoft Kinekt-powered robot to aid earthquake rescue-”

<http://www.bbc.com/news/technology-12559231>

“How WiFi Works”

<http://computer.howstuffworks.com/wireless-network1.htm>

“Bluetooth Basics”

<http://www.baracoda.com/baracoda/technology/why-bluetooth.html>

“How Bluetooth Works”

<http://electronics.howstuffworks.com/bluetooth.htm>

“A beginner’s guide to switching regulators”

<http://www.dimensionengineering.com/info/switching-regulators>

“Switching Regulators”

<http://www.national.com/assets/en/appnotes/f5.pdf>

LM2936 Datasheet

<http://www.ti.com/lit/ds/symlink/lm2936.pdf>

LT1121 Datasheet

<http://cds.linear.com/docs/Datasheet/1121ff.pdf>

MAX-232 Datasheet

<http://datasheets.maxim-ic.com/en/ds/MAX220-MAX249.pdf>

“Control a Car Via PC Using XBee”

<http://sites.google.com/site/electronicprojectsprogramming/control-an-rc-car-via-pc-using-xbee>

“RS-232 Cables, Wiring, and Pinouts”

<http://www.zytrax.com/tech/layer_1/cables/tech_rs232.htm>

“TTL to RS232 Converter”

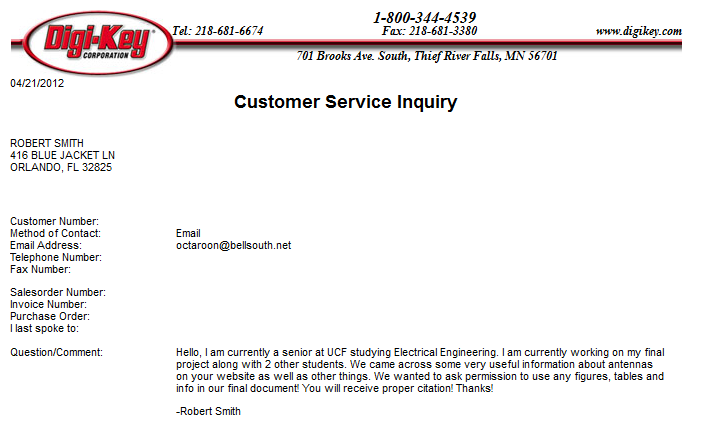
<http://www.coolcircuit.com/gadgets/2008/07/15/ttl-to-rs232-converter/>

PIC16F886 Datasheet

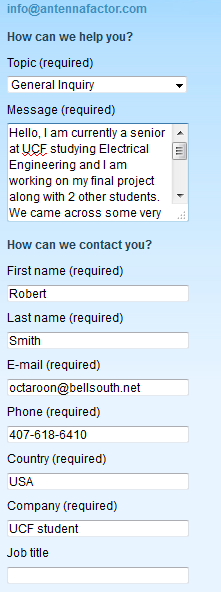
[ww1.microchip.com/downloads/en/DeviceDoc/41291F.pdf](file:///C:\Users\Rob\Dropbox\Senior%20Design\ww1.microchip.com\downloads\en\DeviceDoc\41291F.pdf)

**Appendix B: Permissions**

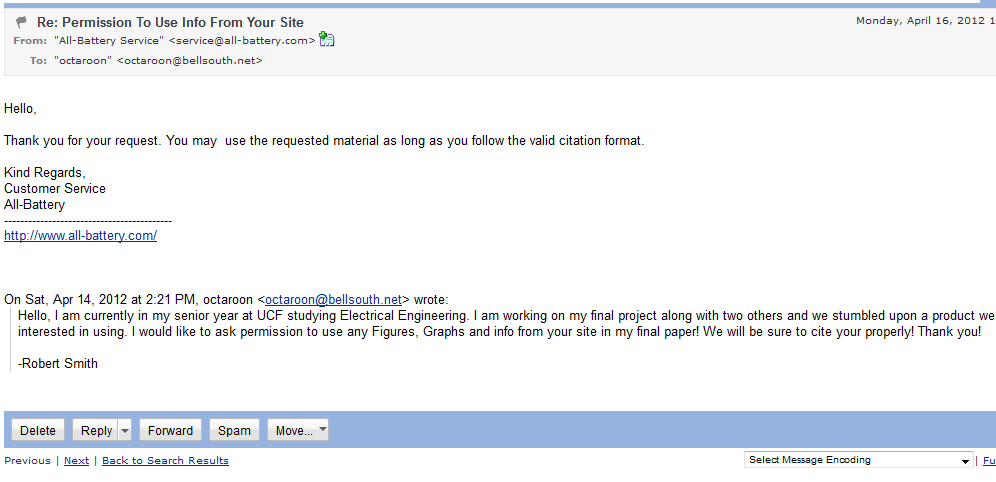
**Digi-Key (PENDING)**



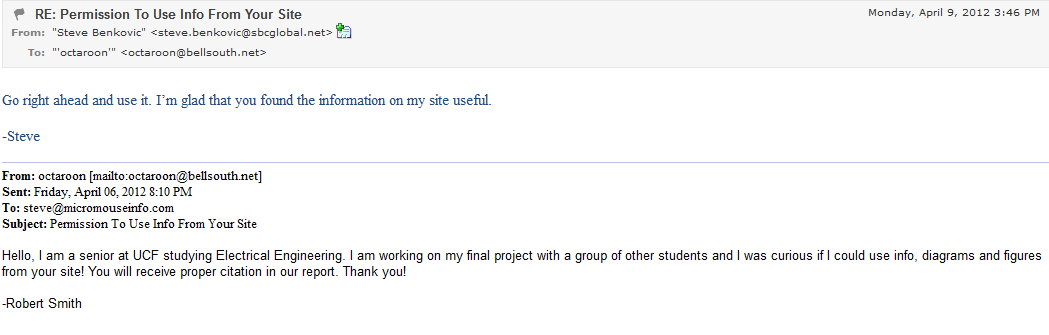
**Antenna Factor (PENDING)**



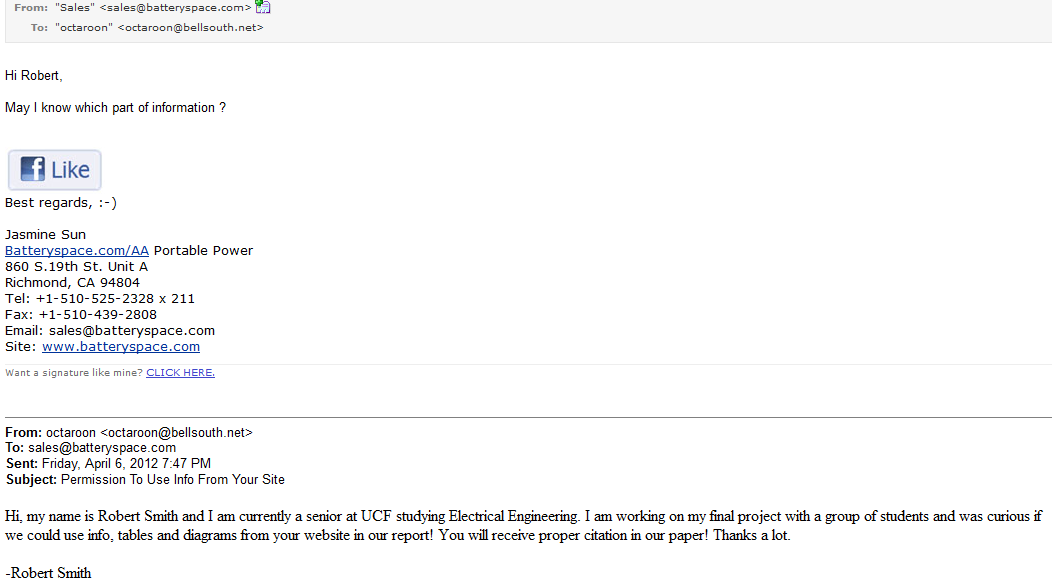
All-Battery Service



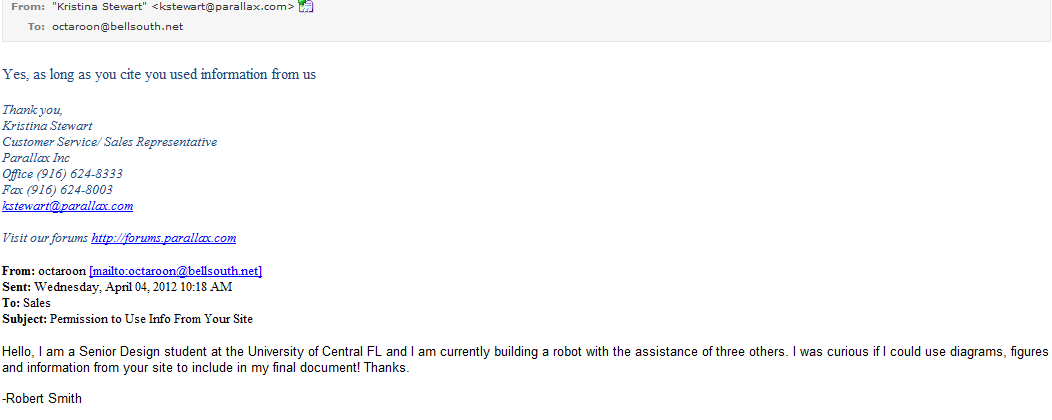
Micro Mouse



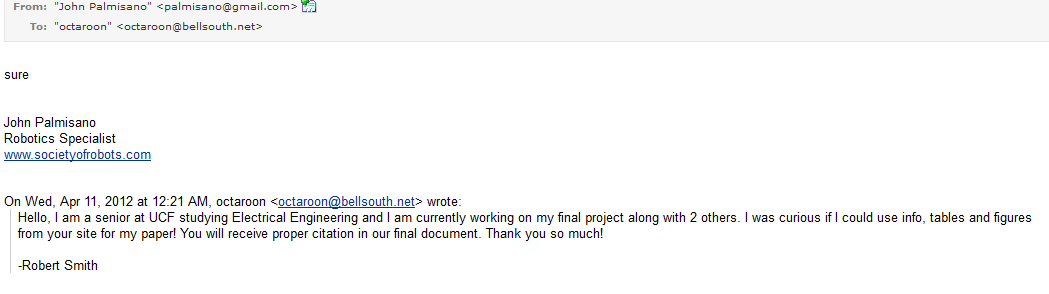
Battery Space (PENDING)



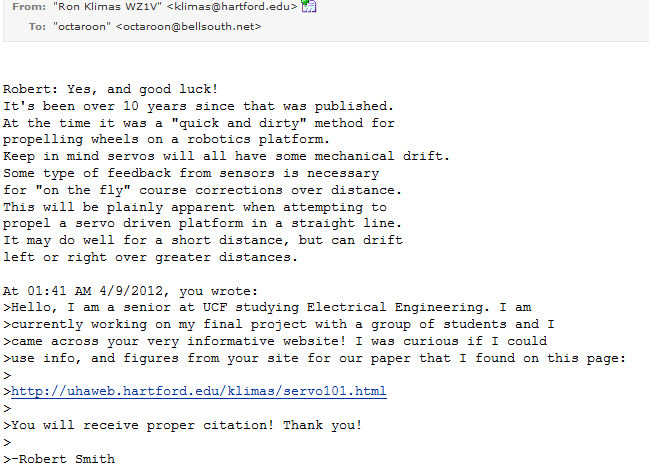
Parallax



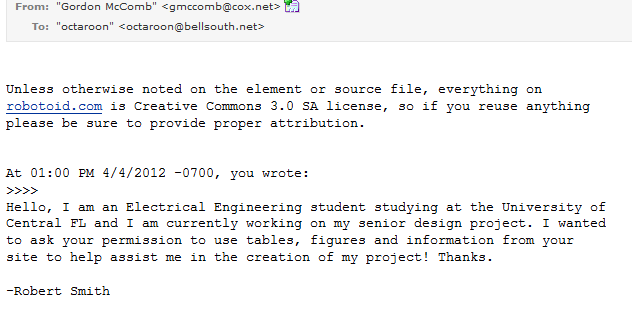
Society of Robots



University of Hartford



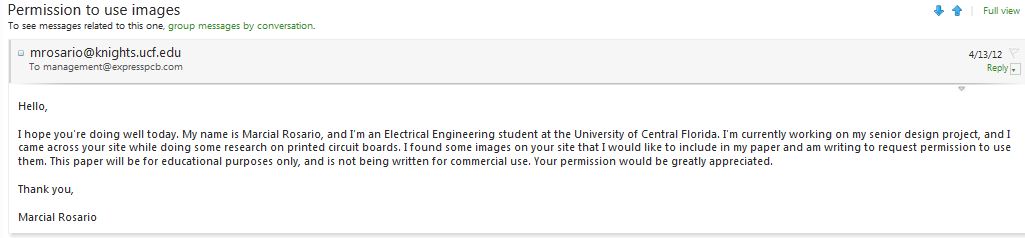
Robotoid



Trossen Robotics

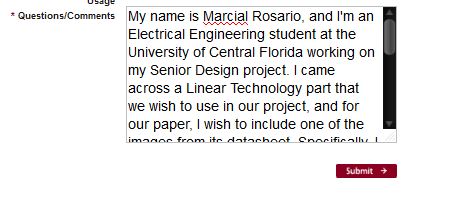


ExpressPCB

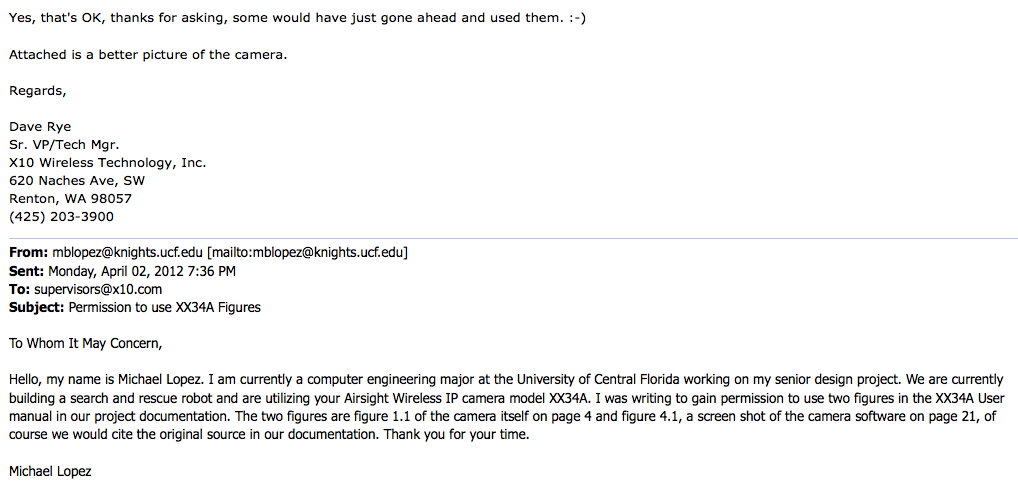




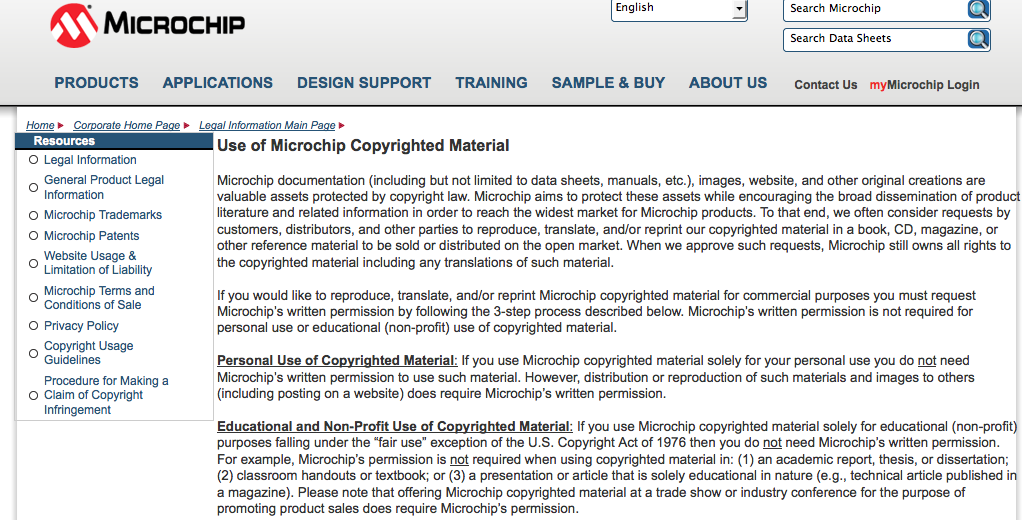
Linear Technology (Pending Permission)



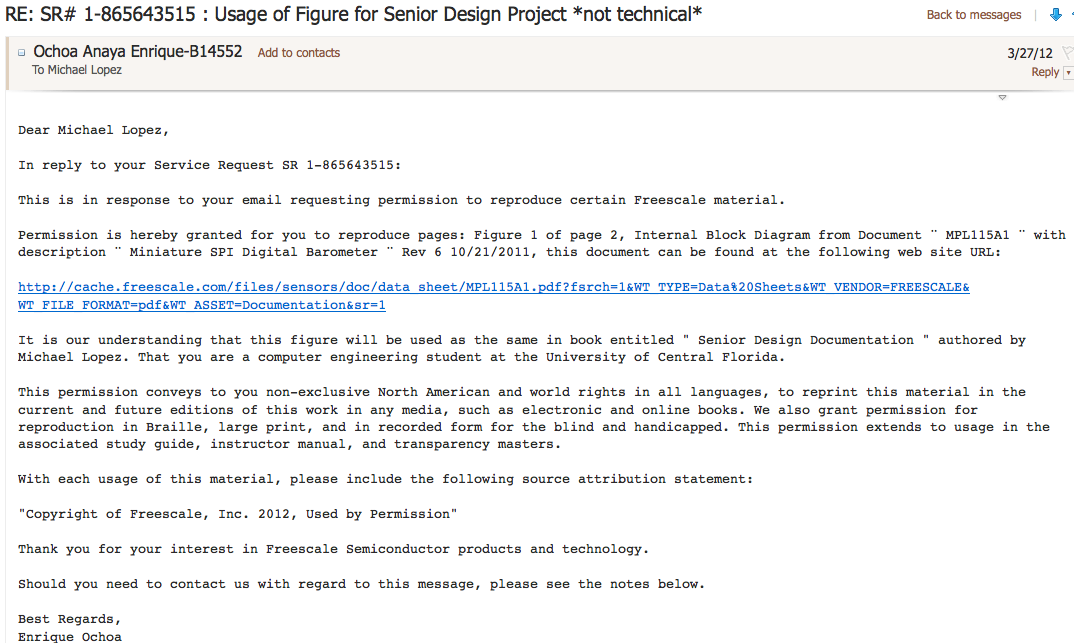
X10



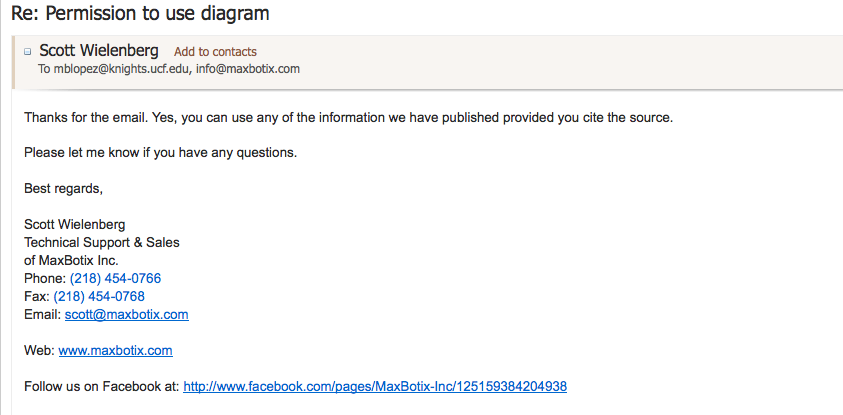
MicroChip



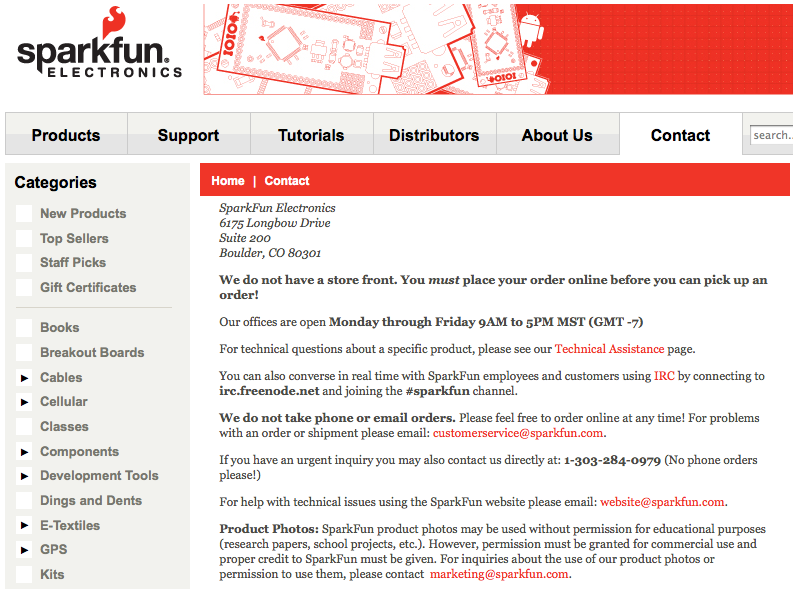
Freescale Semiconductors



Maxbotix



Sparkfun



Additional Pending Sources:

Target

Energizer

**Appendix C: Index of Figures**

Figure 1 – MQ-9 Reaper………...……………………………………………………….11

Figure 2 – uC Block Diagram……………………………………………………………23

Figure 3 – Microcontroller Schematic…………………………………………………...26

Figure 4 – Arduino Visual Studio…………………………………………….……….....27

Figure 5 – Microcontroller Flow Chart…..........................................................................28

Figure 6 – Arduino Uno……………………………………………………………….....29

Figure 7 – PIC Development Board……………………………………………………...30

Figure 8 – MPL115A1 Connection to MCU…………………………………………….31

Figure 9 – Pressure Sensor Internal Block Diagram……………………………………..32

Figure 10 – Voltage vs. Humidity Graph of Temp/Humidity Sensor……………………33

Figure 11 – Temp/Humidity Sensor Connection to MCU……………………………….34

Figure 12 – Range Finder Beam Pattern…………………………………………………35

Figure 13 – Range Finder Connections to MCU………………………………………...36

Figure 14 – Dark-Detecting LED Circuit………………………………………………..37

Figure 15 – Linksys IP Camera………………………………………………………….39

Figure 16 – Camera Software……………………………………………………………39

Figure 17 – Sensor Data Flow…………………………………………………...………41

Figure 18 – RF RX to MAX232 Wiring…………………………………………………41

Figure 19 – Software Flowchart…………………………………………………………42

Figure 20 – Display GUI………………..……………………………………………….42

Figure 21 – Class Diagrams..…………………………………….………………………44

Figure 22 – Wireless Control…………………………………………………………….45

Figure 23 – Robot Control Flow Chart…………………………………………………..46

Figure 24 – Buck Regulator……………………………………………………………...48

Figure 25 – LM7805 Regulator Connected to Components…………………….……….49

Figure 26 – Example of LM2936 Connected to Components……………………...……51

Figure 27 – Sample Implementation of LTC3632……………………………………….52

Figure 28 – Boebot Chassis Dimensions (Bottom View)………………………………..53

Figure 29 – Typical H-bridge Using Transistors as Switches…………………………...55

Figure 30 – MT-WSM500 DC Motor Dimensions………………………………………56

Figure 31 – SpringRC 43R Servo Motor Dimensions…………………………………...59

Figure 32 – Remote Control Car Dimensions……………………………………………61

Figure 33 – Module Showing How Our Robot Will Receive Instructions………………63

Figure 34 – Wenshing Hi Power Transmitting Module……………………....………….64

Figure 35 – Xbee Exploer USB………………………………………………………….66

Figure 36 – Xbee Explorer Regulated…………………………………………………...67

Figure 37 – Xbee 1 mW Wire Antenna Transceiver…………………………………….68

Figure 38 – Alkaline and Lithium AA Battery Performance…………………….………69

Figure 39 – 7.2 V, 700 mAh NiCd Battery………………………………………………73

Figure 40 – Tenergy 7.2 V 2000 mAh Side-by-Side NiMH Battery Pack………………73

Figure 41 – Compact Smart Charger (0.6 A) for NiCd or NiMH Batteries……………..74

Figure 42 – 916 MHz Ceramic Chip Antenna by Antenna Factor………………………75

Figure 43 – 916 MHz Planar Surface Mountable Antenn………………………….……76

Figure 44 – 916 MHz ¼ Wave Whip Antenna……………………………....…………..78

Figure 45 – Polulu Low Voltage Dual-Serial Motor Controller…………………………79

Figure 46 – Order of Operation For Wireless Control……………………………..….....81

Figure 47 – PCB Layout Software……………………………………………………….83

Figure 48 – Screen Capture of PCB123 Software……………………………………….85

Figure 49 – Rebuilt Final Main Circuit…………………………………….…………….91

Figure 50 – Final Robot Prototype………………………....………………....………….94

Figure 51 – Cost Breakdown………….…....……………………………………………96

Figure 52 – Group Responsibility Chart…………………………………………………97

**Appendix D: Index of Tables**

Table 1 – Table of Requirements……………………………………….............................6

Table 2 – Pros and Cons of 802.11………………………………………………………14

Table 3 – Pros and Cons of Bluetooth…………………………………………………...15

Table 4 – Possible PIC MCUs………….………………………………………………..24

Table 5 – Microcontroller Sensor Output………………………………………………..27

Table 6 – RS232 Outputs………………………………………………………………...40

Table 7 – Robot Control…………………………………………………………………44

Table 8 – Parts to Keep in Regulation…………………………………………………...46

Table 9 – Various Motor Pros and Cons…………………………………………………54

Table 10 – DRV8835 Dual Low Voltage H-bridge Spec Table…………………………55

Table 11 – Continuous DC Motor Specifications…………………………………….….57

Table 12 – Pin Layout of Wenshing Transmitter………………………………….……..64

Table 13 – Hightlighted Specs of Wenshing Transmitter………………………….…….64

Table 14 – Motor Controller Pin Layout…………………………………………...........80

Table 15 – Chassis Testing………………………………………………………………87

Table 16 – Speed Testing………………………………………………………………...88

Table 17 – Battery Testing……………………………………………………………….88

Table 18 – Battery Charger Testing……………………………………………………...89

Table 19 – Video Camera Testing………………........………………………………….90

Table 20 – PCB Testing………………………………………………………………….90

Table 21 – Wireless Control Testing…………………………………………………….92

Table 22 – Sensor Testing…………………….………………………………………….93

Table 23 – Microcontroller Testing……………………………………………………...93

Table 24 – Project Cost Breakdown……………………………………………………..94

Table 25 – Project Milestones……………………………………………………………98