ON DESIGNING COLLABORATIVE ROBOTIC SYSTEMS WITH REAL-TIME OPERATING SYSTEMS AND WIRELESS NETWORKS

by

Andrew Jacob Wolin

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Engineering

Boise State University

December 2014

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Andrew Jacob Wolin

Thesis Title: On Designing Collaborative Robotic Systems with Real-Time Operating Systems and Wireless Networks

Date of Final Oral Examination: 21 November 2014

The following individuals read and discussed the thesis submitted by student Andrew Jacob Wolin, and they evaluated his presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Sin Ming Loo, Ph.D.	Chair, Supervisory Committee
John N. Chiasson, Ph.D.	Member, Supervisory Committee
Hao Chen, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Sin Ming Loo, Ph.D., Chair of the Supervisory Committee. The thesis was approved for the Graduate College by John R. Pelton, Ph.D., Dean of the Graduate College.

ACKNOWLEDGEMENTS

I would like to thank my Lord and Savior Jesus Christ. Apart from Him I can do nothing. “Delight yourself in the Lord, and He will give you the desires of your heart.” Psalm 37:4

I would like to thank my thesis advisor, Dr. Sin Ming Loo, for allowing me to pursue this work. His knowledge and experience was an invaluable guide during the research. I would like to thank committee members Dr. John Chiasson and Dr. Hao Chen for their support, time, and guidance. They are true engineers, mathematicians, and mentors. I would like to thank the highly talented members of the Hartman Systems Integration Lab for sharing their experience and insight into the world of embedded systems.

I would like to thank my father, Dale, who is an engineer by birth and has always inspired new and interesting innovations. Thanks to my mother, Lois, who has always gone to great lengths to take care of me. Thanks to my brother, Jason, a computer natural who started programming an HP 48S in his early youth. Thanks to my brother, Joe, who has been an inventor and entrepreneur even before graduating high school. Thanks to Jon and Carol for their love and support.

I would like to thank my wife, Amy, for her bright smile and heart of gold. I would like to thank my children who are constantly innovating, creating, and encouraging me to do the same.

ABSTRACT

Robotic devices currently solve many real-world problems but do so primarily on an individual basis. The ability to deploy large quantities of robots to solve problems has not yet been widely embraced partially due to the complex nature of robotics and levels of research investment.

Existing research in robot collaboration largely exists in the forms of models and simulations. This work seeks to accelerate the next level of research in this area by providing a low cost, collaborative capable robotic system. This platform can be used as a gateway to transport simulations into physical representations. This research not only presents a completed robot but also provides a guide for fellow researchers to use when custom tailored systems are required.

Harnessing the power of robotic collaboration will inspire a new generation of problem solving and eventually produce products that could even save lives. This work demonstrates the principles and technologies needed to design robots in general and for use in collaborative environments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADC	Analog-to-Digital Converter
cm	Centimeter
DC	Direct Current
GPIO	General Purpose Input/Output
IR	Infrared
I/O	Input/Output
I2C	Inter-Integrated Circuit
IGRS	Intelligent Grouping and Resource Sharing
ISR	Interrupt Service Routine
LED	Light Emitting Diode
mm	Millimeter
mV	Millivolt
PWM	Pulse Width Modulation
PCB	Printed Circuit Board
RF	Radio Frequency
RTOS	Real-Time Operating System
SPI	Serial Peripheral Interface
TI	Texas Instruments
USB	Universal Serial Bus
UART	Universal Asynchronous Receiver/Transmitter

CHAPTER 1: INTRODUCTION

1.1 Solving Problems with Robots

One of the most amazing displays of early robotics was the invention of a remote controlled submergible boat. In 1898, Tesla demonstrated a contraption that was far ahead of its time. It was able to receive radio signals, and therefore commands, to move forward, turn, submerge, and toggle lights [1]. It was perhaps so far ahead of its time that the idea of robotic minions was not ready to be accepted by the public or even the military. Attempts at purely mechanical and analog robots were made during the early 1900s. The real fuel to the robotic revolution came with the invention of the solid state transistor in 1947. As advancements in computing were made, robotic creations followed. As programs were written to solve basic math problems the possibilities became readily apparent for computers to join themselves to other electric and mechanical machines.

One of the first robotic inventions ever produced, sold, and made to perform useful work arrived in 1961. The robot was named Unimate and was used in a General Motors assembly line to perform work that was difficult for humans to do. With significant changes in the world since Tesla's idea, society was ready to accept the idea of robots assisting mankind [2].

Japan embraced robotic technologies and has since become the world leader in use and manufacturing of robots both for industry and personal use. The International

Federation of Robotics reports that Japan was the largest producer of robots in the world [3]. Japan has also taken an interest in humanoid robotics and has led several innovations in that field. As an example, Japanese manufacturer Honda has produced the famous humanoid robot ASIMO, an acronym for Advanced Step in Innovative MObility. ASIMO is able to interact with people by recognizing human gestures, sounds, and shapes.

1.2 Solving Problems with Many Robots

Solving a simple problem with a single robot, autonomous or semi-autonomous, is by itself a challenging task. The rewards for meeting the challenge are huge and have led to the betterment and saving of human lives through advanced prosthetics, wheel chairs, and implants. From the depths of the ocean to distant mars, robots have served as competent explorers into the unknown. Taking this technology and enabling it to work as a group, collectively, is the idea behind collaborative problem solving with robotics [4].

1.3 Swarm Robotics vs Collaborative Robotics

Swarm robotics is a specific type of group robotics that has emerged over the last ten years. It has produced very interesting collective behaviors inspired by insect groups. The exact definition of swarm robotics is somewhat undefined. In general, the common goal is to have the robotic group interact with each other, but not communicate with each other using a single intelligent organizer like a computer [5]. Applying this limitation on the discipline of swarm robotics forces the research to solve problems in a highly redundant and scalable manner. The robots interact with each through similar mechanisms found in ant colonies and bee hives. An excellent demonstration of swarm robotics has been compiled at the Universite Libre de Bruxelles. The following series of steps will demonstrate a simple algorithm to explain swarm robotics [6].

1) All robots are tasked to find a light hanging from the ceiling.
2) Initially all robots emit red light around their body.
3) If a robot detects another red robot, it changes direction and continues searching.
4) If a robot detects the light, it changes color to blue.
5) If a robot detects the now blue robot, it latches on and changes color to blue.
6) Once all robots detect the light, all are blue and latched together.

In this example, a problem has not necessarily been solved but the robots have self-organized based on a simple set of rules. All of these activities are done using passive forms of communication as the robots merely sense changes to neighboring robots color. It is a completely different way to solve problems and has led to exciting new possibilities.

Collaborative robotics does not fall into the exact guidelines of swarm robotics since more advanced communication between robots is encouraged and required in order to solve the problems of interest in a time-urgent manner. The most important and time-urgent types of problems involve saving human life. In these types of scenarios, a person could be lost or injured in an area dangerous for rescuers to enter such as a cave system or fire. Other scenarios could be hikers lost in areas too large for humans to traverse quickly. Using groups of robots, on the ground or in the air, to save lives will involve a human response team that then deploys the group and receives instant feedback to the status of the group and the locations of the individual members. Once a member detects signs of life, or even the possibility, search crew efforts can be more efficiently focused and concentrated. Since the members must travel potentially large distances from each other, they cannot use the swarm mechanisms used by swarm robotics to interact. They must communicate in an active way with each other to coordinate the large-scale work and relay critical information such as discovery of the lost or wounded.

Many of the ambitions of swarm robotics agree with collaborative robotics in that many robots, costing as little as possible, should be used to solve a problem. Unlike swarm robotics, however, it is important to this innovation that there are few restrictions. If the most effective solution is to have multiple groups of different types of robots combine to solve the problem, then this is the superior solution until a new solution is discovered.

1.4 Summary of Definitions for this Research

The terminology in this study conforms to existing terminology as best as possible. The terms, "group," "swarm," "collaborative," and "cooperative" are so similar in lexical meaning that some well-placed definitions will aid in understanding the differences. Table 1.1 describes the differences in order from generic to more specific.

Table 1.1 Group Robotics Naming Conventions

Naming Convention	**Description**
Group Robotics	A generic term describing many robots; they may or may not work together.
Cooperative Robotics	A slightly more specific type of group; they cooperate in some way but exactly how they cooperate is undefined.
Swarm Robotics	Specific swarm behavior rules are applied to the group of identical robots and communication is only shared through passive signals, such as color of a neighboring member.
Collaborative Robotics	A group of robots, not necessarily identical, who communicate critical information in order to best understand and solve a problem [7]. Swarm rules can be applied to the collaborative process.

1.5 Thesis Framework

In order to provide a thorough guide to designing robots for collaboration, the chapters that follow are presented in a logical order that follows the design process, from start to finish. Chapter 2 evaluates the previous research that has been done in robotic prototypes. Chapter 2 also looks at the existing studies of robotic collaboration to see if the research is moving beyond models and simulations. In Chapter 3, a design process is presented, which will be referred to throughout the rest of the chapters. Chapters 4, 5, and 6 step through the critical decisions that must be made during component selection, mechanical design, and hardware design. Chapter 6 also goes into detail regarding specific configurations of the microcontroller since this is foundational to a successful project. Chapter 7 contains the final assembly and considers the ramifications of overly complicated designs. In Chapter 8, guidance is given through the presentation of a software framework that takes advantage of a real-time operating system. Considerations are also made regarding collaboration methods. Chapter 9 presents a thorough testing process that will verify the hardware and provide valuable data for reference during software development. Finally, conclusions and future research are made in Chapter 10.

CHAPTER 2: PREVIOUS WORK AND EXISTING TECHNOLOGY

This chapter focuses on existing technology in the field of design and fabrication of collaborative robotics.

2.1 Group Robotics

Many forms of group robotics exist. Their application stretches from the bottom of the ocean to the depths of space. Researchers at California's Institute of Technology Jet Propulsion Laboratory have begun work that will aid the construction of buildings on Earth's moon and even Mars [8]. Construction work such as this requires a collaborative effort at times. Multiple types of robots are likely to be deployed and the technology must exist where a robot can call upon another robot that has the specialized tool set to solve the problem. Their work uses an outdoor-like laboratory setting to perform the experiments. The robots are much larger than the platform proposed in this research and consequently expensive. The low robot counts result in fewer collaborative tests. It would be beneficial for the research to use scaled down versions of the hardware for experimentations on a lab table. Once collaboration experiments have been performed on the scaled model, the results can then be applied and further testing can continue on the large-scale robots.

Collaboration can exist in group robotics at the autonomous and semi-autonomous level. The collaborative robotic construction mentioned earlier will likely be monitored and commanded by humans in a control center and therefore is considered semi-autonomous. Research performed by Tauchi et al. [9] has demonstrated interesting simulations where robots are assigned one of three different types of personalities and then deployed into a fully autonomous mode. In order to collaborate most effectively, the robots discover one another's personalities and then solve a problem by grouping according to the same personality. These simulations show the need for efficient collaboration and have set the stage for additional research. The next logical step for this research is to port the simulation to actual hardware that can support multiple threads and task management.

Similar to the research described in [9], Wang et al. [10] have developed a model where each robotic node has a concept of intention much like a human has intention to solve problems. When multiple robots all share a common intention, resolution must be made on how the robots will work together in order to solve the problem. This involves fairly complex algorithms in order to overcome the inherent contention in collaborative robotics. It uses negotiation prediction and an equilibrium matrix in order to keep appropriate distance between the nodes. This research likely has interesting results but only exists as a model and requires a physical platform to execute.

The Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory (MIT CSAIL) has led several efforts toward collaborative robotics. When using groups of robotics, several different control scenarios are possible. The robots could be asked to work in complete autonomy without any input from the user. In other situations, some input could be given to help direct the group towards a solution. In both cases, a degree of randomness is involved due to the possibility of communication links failing between nodes and unforeseen obstructions that render a worker immobile. A mathematical model surrounding this type of randomness is called the Markov Decision Process (MDP). Dibangoye et al. [11] have utilized this model in collaborative robotics. Functional tasks have been demonstrated such as the ability to recognize objects too large for a single robot to move and asking for assistance from other available worker nodes.

2.2 Collaboration Using ZigBee Protocol

ZigBee is a RF communication protocol that adds mesh networking to the 802.15.4 radio protocol. The technology has grown rapidly since its conception and has been adapted to robotic collaboration in many unique ways. One approach to group robotics has adopted the Intelligent Grouping and Resource Sharing (IGRS) protocol as a foundation for its collaboration mechanism. IGRS was initially designed to support the resource sharing of PCs, TVs, Set Top Boxes, and mobile devices. As robotics begin to act more and more like an appliance in the home and office, this protocol should likewise be inherited by robotics. Zhiguo, Junming, Xu, Zhilang, and Jun have begun research in this area and propose the use of ZigBee wireless communication between robotic nodes which then integrate into an 802.11b/g/n home wireless network [12]. Their study is largely focused on how to provide a scalable grouping strategy based on IGRS and consequently lacks stable robotic test platforms to carry out the experiments.

Swarm robotics has taken an interest in ZigBee technology as well. The data that is communicated in swarm robotics attempts to simulate the same types of data exchanges that would happen in a colony of insects. Zhang and Gao [13] have evaluated ZigBee for use in swarm robotics. They have found that the self-healing properties of the network are critical in order to represent swarm behavior. If an ant is put out of commission in a colony, another ant is readily available to take its place. In the same way, ZigBee is able to re-route data in the event that a node is removed from the mesh network. The network in Figure 2.1 represents the concept behind the mesh implementation. Note that, while a client (aka router) is able to fail, the coordinator must remain active.

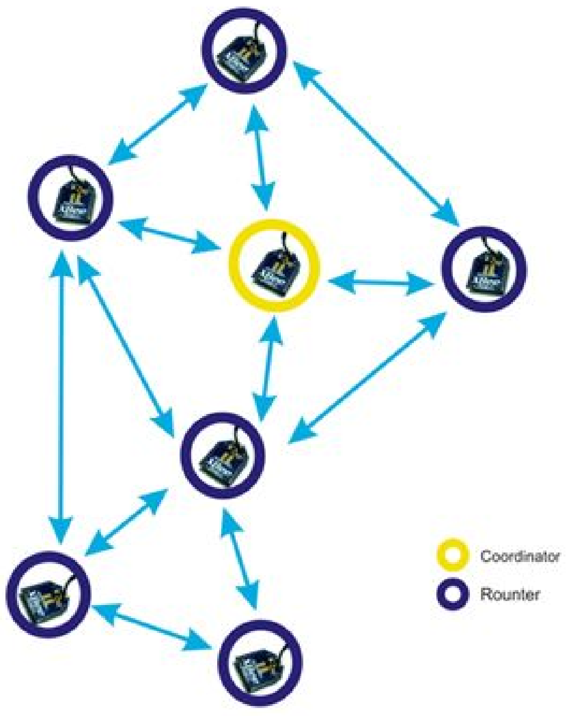


Figure 2.1 Mesh Network

While a mesh network lends itself to swarm robotics, a star network is often sufficient for some types of collaboration. A star network has several clients with a coordinator that is often times connected to a computer. The sharing of data in this style of network is relayed through the coordinator. Consequently the star network lacks the self-healing feature of the mesh network but is simpler to implement and can use a higher processing power central node. Wan et al. [14] have also evaluated and used ZigBee for multi-robot communication but have chosen the star network instead of the mesh network. This allows the use of real-time data manipulations on the higher powered PC that connects to the coordinator. An illustration of a mesh network is seen in Figure 2.2.

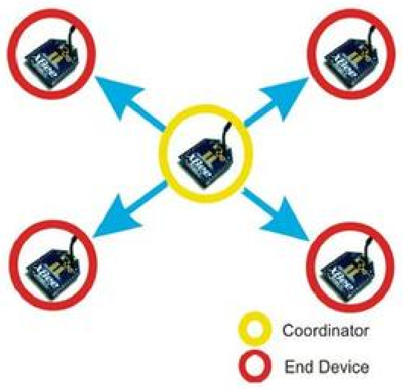


Figure 2.2 Star Network

Many chipsets are available that implement the ZigBee protocol. The research in [14] has selected the MG2455 manufactured by Radio Pulse. Much like the well-known XBee module, the MG2455 is able to carry out mesh and star networking between embedded systems. Two robots carried out experiments to demonstrate the usefulness of ZigBee during collaboration.

2.3 Existing Robotic Platforms

An interesting robotic platform was developed in 2008 by researchers Chen and Liou [15]. This platform rightly embraces the concept of using a real-time operating system (RTOS) in order to abstract the hardware layer and provide functional task management. RTOS platforms such as VxWorks, UC/OS-II and others existed during this time. UC/OS-II, also known as Micrium, was chosen to control the robot. Micrium is a highly capable RTOS, which has even been used on the Mars Curiosity Rover. It is available free of cost to educational institutions but is a closed source project. Because it is closed source, there are fewer resources available to researchers in need of free open source solutions. The robot uses a hexapod leg system to move itself, which lends itself to certain environments but as a consequence it is very slow to move about. This platform is not designed for collaboration but does have desirable features such as a layered software approach that depends on an RTOS.

2.4 Summary

As researchers continue to develop robotic collaboration models and simulations, they will eventually need a physical platform to execute on. Examining the existing research reveals that there is a need for a complete guide to the process of building a robot from idea to conception. Many of the proposed models found in this research can take advantage of a generic platform while others will need to specialize. This work seeks to provide a viable robotic platform that can be used in many different areas of research and provide a guide to the design process.

CHAPTER 3: THE DESIGN PROCESS

The design process involves a series of logical tasks. Some tasks can be done in parallel. Others can only be completed sequentially. This chapter discusses the workflow in general and describes the advantages of using teams to develop products that involve many different disciplines. This design process will be referred to through the rest of the chapters.

3.1 Task Dependency and Workflow

Before engaging in the development of a robotic system, a certain amount of research should be performed. Figure 3.1 shows research as the top most phase in the process. Both mechanical and hardware research should be done and existing technologies leveraged when possible and appropriate. The component selection phase and testing of these components must be completed prior to moving forward with the rest of the design process. The mechanical design depends on the dimensions of motors, battery packs, and PCB sizes. There are likely to be several iterations of component selection, testing, and design. During this phase, the components are bread-boarded (if possible) and tested to ensure compatibility with the design goals. Simultaneously, mechanical prototypes can begin to be designed in Computer-Aided Design software. Performing these tasks diligently allows unforeseen issues to be handled early on where design changes can be easily accommodated. The exact tolerances and friction fittings can be tested on small parts before fabricating large models, which will have a higher cost to produce.

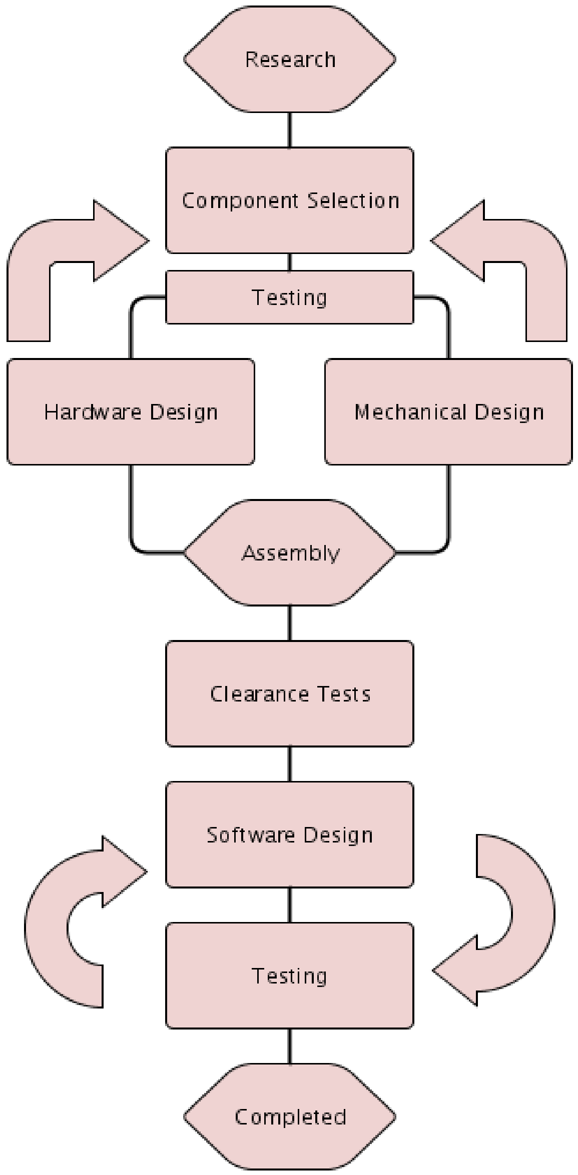


Figure 3.1 Design Workflow

After several phases of component selection, testing, and design, a final assembly will emerge. Testing can then begin to ensure that all aspects of the mechanical design combine together as required without problems such as wheel rub, screw alignment, and ground clearance.

At this point, the software design can commit to design decisions that allow further testing of the system. Several software components will have already been written that tested the components during the initial selection process. These components now need to be integrated into an application programming interface (API). Decisions regarding the threading model and library layers should be made before actual code production. The software design and testing phases will be repeated until all of the hardware components have been verified. Once the robot is built and tested, the targeted robotic research area of interest can begin.

The workflow allows for some phases to be done in parallel while other phases cannot move forward until the previous phase completes. The mechanical and hardware design can be done in parallel after some or all of the components have been decided on. If for example a motor has been decided upon, modeling can begin to mount the motor while bread-board testing can be done to verify circuit and software designs. This assumes that multiple researchers are available. The process is much more linear if a single researcher is tasked with the entire design.

3.2 Summary

When designing a system with many integrating components, it is critical to prepare a plan. The plan helps prioritize tasks and dependency of tasks. This will prevent significant errors that could be difficult to solve later in the design process.

CHAPTER 4: COMPONENT SELECTION

Robotic platforms typically integrate existing components from a variety of manufacturers. Items such as motors and batteries for example are much too costly to have custom fabricated for academic research purposes. For the initial phases of collaborative robotic research, common low cost components are targeted since they are able to achieve sufficient accuracy for useful testing. The mechanical, hardware, and software design rely heavily on the physical dimensions and properties of these components. The required characteristics and capabilities of the robots will help narrow the search criteria during component selection. A certain level of commitment must be made to these components before moving forward with the rest of the design. The physical components for a wheeled platform are the motor, power supply, and wheel/hub assembly. Considerations for each of these components are presented in the following sections.

4.1 Integrated Motor Features

Before examining the different types of motors available, certain features should be considered. Motors are often available with or without encoders, gear reduction, and right angle output hubs. Some motors, such as servo motors, are actually systems that contain encoder, gear, and output hub integrated into a single package.

4.1.1 Encoders

Both stepper and DC motors optionally have integrated encoders that can determine the position of the motor shaft and send a pulse to the microcontroller for use in a control loop [16]. When a stepper motor is given the command to move 10 steps, there is only reasonable confidence that it actually moved 10 steps. An encoder built into the motor will provide feedback to the microcontroller. It reports how many steps it actually moved so that a measure of error can be taken and corrected. In a similar way, power is applied to a DC motor and the encoder counts are fed into the MCU.

4.1.2 Gear Reduction

A DC motor by itself is of limited use without the aid of gear reduction in order to achieve the necessary torque to move a meaningful amount of weight. In contrast, a stepper motor has enough torque to direct drive a small robot. A DC motor with a gear head will usually have much more torque than a stepper motor without a gearhead. For this reason, DC motors in robotics often use a gear reduction system, which could be a gearhead or pulley system.

4.2 Motors

Stepper motors and brushed DC motors are common and available platforms that can be used in small robotic applications. Table 4.1 makes relative generalizations to some of the key features of motors.

Table 4.1 Motor Relative Generalizations

	Permanent Magnet	**AC Induction**	**Stepper**	**Brushless**
Voltage	DC	AC	DC	AC
Top Speed	Medium	High	Slow	Very High
Cost	Low	Medium	Medium	High
Precision	Medium	Low	High	Medium

4.2.1 Brushed DC

The brushed DC motor has existed much longer than a stepper motor and was used as inspiration to invent the stepper motor. The armature rotates continuously as it receives a constant or varying voltage. Control mechanisms are as simple as varying the voltage to the motor. Due to the widespread use of DC motors, all shapes and sizes can be found at reasonable prices. A DC motor is shown in Figure 4.1.

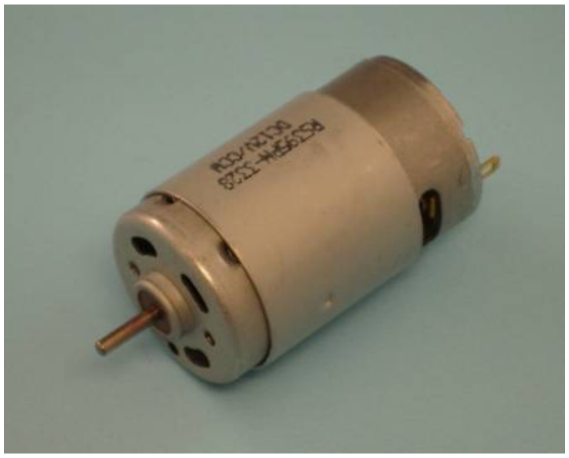


Figure 4.1 Brushed DC Motor

4.2.2 Stepper Motor

The stepper motor became popular in the early 1960s as a replacement for expensive DC motors that required encoders for feedback. Several different types of stepper motors exist with a variety of ways to control them. The general concept is that the axially magnetized rotor will seek to align itself to a stator coil provided the correct energizing sequence of the poles using four different input phases. Changing the input phases repeatedly with logic from a microcontroller allows the stepper motor to rotate a set number of degrees based on the number of poles and coils on the motor [17]. As long as nothing binds or overloads the motor, a stepper can replace a DC motor with feedback. These motors are excellent options for robotic applications but do have design challenges since the signaling must be absolutely precise in order for the motor to perform. Resonance and noise can interfere with the control signals causing additional design challenges. A stepper motor is shown in Figure 4.2.

Figure 4.2 Stepper Motor

4.2.3 Servo Motor

A servo motor is an embedded system that combines a DC motor, gear system, an encoder or potentiometer for feedback, and a motor controller. These components are packaged into a housing that provides a connector for the electrical connections and an output shaft to drive movement.

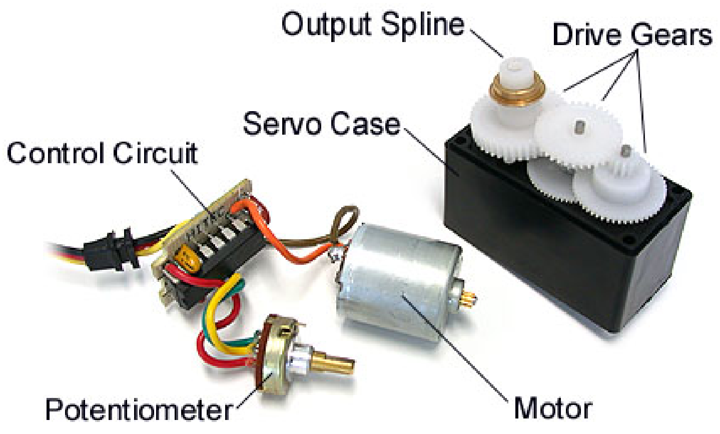


Figure 4.3 Servo Motor Breakdown

There are a variety of different types of servo motors but common to most of them is the control system. A 1-2 ms signal is sent to the servo, which determines which way the servo should rotate. 1 ms will rotate the servo left 90°, 1.5 ms will hold the servo at its center, and 2 ms will rotate the servo right 90°. The frequency of the signals is between 50 and 400 Hz depending on the type of servo. This is the refresh rate of the servo and consequently a higher signal frequency will provide more precise holding of the output shaft.

The simplest servo motor uses a potentiometer to provide feedback to the control circuit in order to determine absolute position. These types of servos only allow for 180° of total rotation.

4.2.4 Continuous Rotation Servo Motor

There are continuous rotation servos available that are very inexpensive ($12-$20) but they completely lack a feedback mechanism and therefore lose the ability to send absolute positioning commands. Trossen Robotics offers highly advanced servo systems that offer continuous rotation with absolute positioning and configurable parameters for the control algorithm. This is an excellent platform for demonstrations of control theory and robotics but the high feature set increases the cost. A low end robotic servo shown in Figure 4.4 starts at $140.

Figure 4.4 Continuous Rotation Servo

4.3 Surplus Motors

4.3.1 Faulhaber 1524E006S123

The simplicity of a DC motor with an encoder and gearhead is an attractive option but the cost for such a motor can be a few hundred dollars if purchased new. An interesting surplus option was found for $23 that provides both the encoder and the gearhead. The Faulhaber 1524E006S123 can be found from surplus dealers. In addition to the gearhead and encoder, the motor comes with a 90° output shaft. Since the encoder and gearhead add considerable length to the motor, the 90° output shaft allows a pair of these motors to oppose each other vertically in a chassis, allowing for a narrow wheelbase. Given the low price point, encoder, gearhead, and 90° output shaft, the Faulhaber was chosen as the power train, which can be seen in Figure 4.6.

4.3.2 Motor Summary

Table 4.2 summarizes the characteristics and cost of the motors researched. While a simple brushed DC motor is by far the least expensive, it requires additional components such as gears and an encoder in order to make it useful for robotics applications. A better approach uses a motor that has these features integrated into the design. High end robotic servos are flexible for many uses and can be configured for continuous rotation but come with additional cost. Ultimately, the surplus Faulhaber 1524E006S123 proved to have the best feature set and value for this particular application.

Table 4.2 Motor Summary

Motor	**Type**	**Nominal Voltage**	**No Load Current (A)**	**Stall Current (A)**	**Cost**
RS395	Brushed DC	12V	0.5	15	$5.25
SM-42BYG011-25-	Bipolar Stepper	12V	0.33	0.6	$14.95
SM-S4203R	Servo	5V	0.25	0.7	$13.95
RX-24F	Continuous Rotation Servo	12V	0.5	2.4	$139.90
1524E006S123	Brushed DC w/ Encoder, Gearhead	6V	0.2	0.5	$23.95*

* Surplus

4.4 Wheel and Hub

It is often the case that a surplus motor will be readily available but will not come with a means to attach the wheel. Robot component retailers such as Pololu, Banebots, Trossen Robotics, SparkFun, and others offer various solutions to this problem. Figure 4.5 shows several different mount and hub options available for various motor and wheel combinations.

a) Clamping Mount

b) Set Screw Mount

c) Stepper Mount

b) Swivel Mount

Figure 4.5 Motor Mounts and Hubs

The output shaft of the Faulhaber measures 7 mm in length, 3 mm in diameter, and has a flat surface for the set screw. Banebots offers a hub and wheel of appropriate size that can mount to this motor. While the hub actually measures 10 mm (3 mm beyond the length of the shaft), the set screw is centered and makes good contact with the shaft. The Faulhaber motor, wheel, and hub are shown in Figure 4.6.

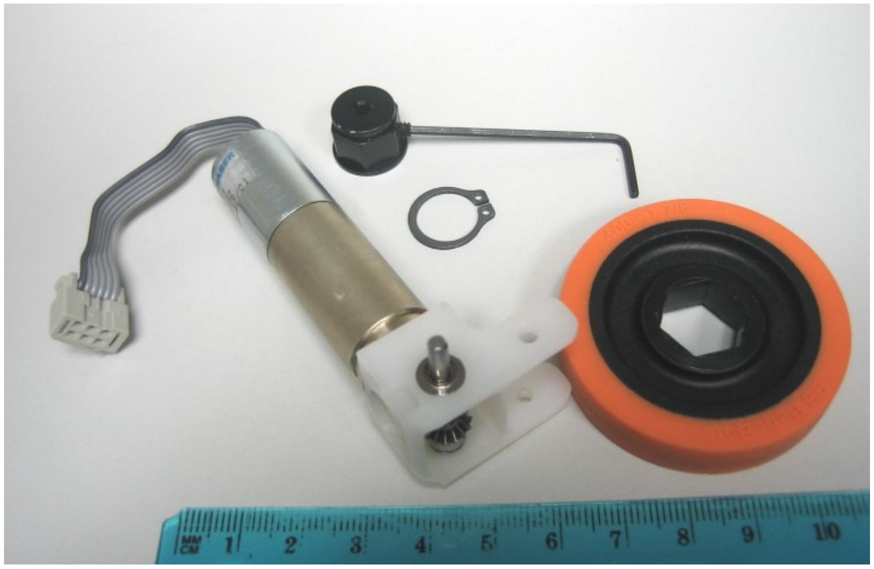


Figure 4.6 Motor, Hub, and Wheel Disassembly

4.5 Microcontroller

The embedded system requires a Microcontroller (MCU) to control the peripheral components. The end goal requirements for the robot will help determine what features are needed from the MCU. The number of peripherals, size of programmable memory, and processor speed should support the design goals. When choosing up front to use a real-time operating system, more programming space and speed should be considered. Efficiency, battery life, and cost are important selection criteria during the design but the goal is not to produce the cheapest, smallest, and most efficient design. The goal is to provide a highly robust, capable, feature-rich platform that can be used as a generic platform for individual and collaborative robotics. The power consumption of the robot can later be scaled back if needed by changing the clock speed, disabling unused features, and optimizing the code base.

Many different MCUs are available from manufacturers such as Microchip, Atmel, Texas Instruments, and STMicroelectronics. STMicroelectronics has an established commitment to supporting open source software and free-to-use tool chains.

This makes them an ideal supplier during research given the absence of licensing costs. The STM32F4 is a series of microcontrollers produced by STMicroelectronics. The particular model chosen from this family is the STM32F4VGT6, which is a 100-pin version containing 100 MB of user programmable flash, 82 lines of IO, and multiple ADC, I2C, SPI, UART, and USART modules. The device is capable of running up to 168 MHz, depending on the input clock configuration [18]. Many other features and abilities are available but are not used for this project. Prototyping with this particular processor is made convenient with an available demonstration board that provides access to all of the pins and has an integrated programmer.

4.6 Power Supply

Once the motor and microcontroller have been selected, a properly sized (physical and electrical) power source can be selected. Power supplies for robotic platforms can range from gas engines to solar panels. The most popular choice is a battery or pack of batteries. Common options include Sealed Lead Acid (Pb), Nickel Cadmium (NiCd), Nickel Metal Hydride (NiMh), Lithium Ion (Li-ion), Lithium Polymer (LiPo), and Lithium Iron Phosphate (LiFePO4). Nominal battery voltages range in value depending on the technology selected and the number of cells used. Table 4.3 makes relative generalizations to some of these common battery types.

Table 4.3 Battery Relative Generalizations

	Sealed Lead Acid (Pb)	**Nickel Cadmium (NiCd0**	**Nickel Metal Hydride (NiMh)**	**Lithium Polymer (LiPo)**	**Lithium Iron Phosphate (LiFePO4)**
Cell Voltage	2V	1.2V	1.2V	3.7V	3.3V
Self Discharge Rate/Month	5%	20%	30%	3%	3%
Size	Large	Medium	Medium	Small	Small
Charge Times	Long	Medium	Short	Medium	Very Short
Cost	Low	Medium	Medium	Medium	Medium-Low
Weight	High	Medium	High	Low	Medium
Energy Density	Low	Medium	Medium	High	High

Before selecting a battery, an estimation needs to be made concerning the target power requirements. Table 6.8 calculates the burst current consumption to be 3.7A and typical consumption to be approximately 1A. Lithium technology batteries have a rating that is used to calculate the rated discharge current. This is known as the “c” rating. Multiply the “c” rating by the amp hour capacity to find the discharge capabilities. Any of the above listed battery technologies could have configurations to support these voltage and current demands. The robot application will help determine what battery technology is most appropriate. For a small form factor robot such as that in this work, size, availability, and cost are the targeted features. LiPo and LiFePO4 batteries will be examined further given their small size and low cost.

The motors in this work have a nominal voltage of 6V, followed by the infrared (IR) modules using 5V, and finally the microcontroller and other ICs use 3.3V. Regulators will source the battery pack and regulate down to these values. The Linear Dropout Regulators (LDO) researched for this purpose typically have a dropout rating of

500mV or less. This suggests that when the pack is fully depleted, the voltage should be 6.5V or greater since a 6V regulator will be used for the motors. A 7.4V 2S LiPo pack has a charged voltage of 8.4V and a fully discharged voltage of 5.5V although 6V is typically used as a precautionary measure since over discharging a pack can damage it. This pack will work but the full energy of the pack will not be utilized since the LDO needs to regulate 6V and the dropout overhead is 500mV. In order to maximize runtime, but reduce maximum speed, the motor supply can be reduced to 5V. This would allow for the pack to drop to 5.5V even though the voltage would be cutoff at 6V for a margin of safety. Regulating the motors to 5V is a reasonable option since maximum speed is the least of concerns during a large part of the development process.

Using 5V for the motors provides another benefit in that LiFePO4 batteries can be used. The majority of the discharge curve for a 2s LiFePO4 pack is 6v and above. This allows for full use of the available battery capacity. LiFePO4 are a greener, safer alternative to LiPo since they use iron and phosphate instead of the cobalt used in LiPo batteries [19]. The concept of dangerous battery fires is nearly eliminated. Never leave any battery on the charger unsupervised since electricity itself is inherently able to start fire. They are not as energy dense as LiPo but considering the power demands of this application, a 700mAh pack can provide 3.5A continuous current and 7A peak, which is more than adequate. Another benefit of LiFePO4 is that they maintain their nominal voltage until the very end of the discharge cycle unlike LiPo, which continuously drops in voltage over the life of the discharge cycle. This benefit is of little concern for this application since regulators are used but for something like a flashlight, the brightness would not slowly diminish over time. One benefit of LiFePO4 that is of interest however is the ability to charge 4 times faster than a LiPo. LiFePO4 also benefit in lower cost up to 50% compared to LiPo as seen in Table 4.4.

Table 4.4 LiPo/LiFePO4 Pack Cost Comparison

	Manufacturer	**Cell Count**	**Amp Hours (mAh)**	**Cost**
LiPo	Turnigy	2	500	$4.39
LiFePO4	Zippy	2	700	$2.89

One major disadvantage of LiFePO4 over LiPo is their weight. A comparable LiFePO4 can weigh 30-40% more than a LiPo. For a small robotics application, this difference of 12 grams is of little concern. For a small, quadcopter, however this would be a significant consideration. LiFePO4 will likely continue to gain in popularity given the larger abundance of materials needed to make the battery and the greener impact on the environment but they will not entirely replace LiPo given the greater weight and lower discharge capabilities. The discharge curves of both packs will be evaluated during testing. Both packs are available in similar enough sizes to fit into the target application, allowing both technologies to be used and tested. Table 4.5 summarizes the electrical characteristics of the two battery technologies considered for this application.

Table 4.5 LiPo / LiFePO4 Electrical Characteristics

	Min Voltage	**Nom Voltage**	**Max Voltage**	**Max Charge**
LiPo	3.0V	3.7V	4.2V	1C
LiFePO4	2.0V	3.3V	3.6V	4C

4.7 Summary

The physical dimensions of the analyzed components are required in order to begin the mechanical design phase. In order to properly choose the microcontroller and supporting integrated circuits, the type of motor must be decided upon. Once the motor and microcontroller have been chosen, a properly sized power supply can be chosen based on size and electrical characteristics. The wheel diameter is critical in a robot since the speed of the robot will depend on its size. The wheel diameter also dictates how much physical space exists underneath the wheel axles. Components such as batteries can utilize this space and streamline the design. Careful consideration and selection of these components will prevent costly disruptions later in the design process due to integration issues that could have been prevented. Modeling these components into a mechanical design will reveal these issues and is the topic of the next chapter.

CHAPTER 5: MECHANICAL DESIGN

Once the mechanical and electrical components have been selected and verified to be available from vendors, a chassis must be constructed. In a robotic system, the mechanical design is a critical component to the success of the overall project. If the embedded system is flawless in every aspect but lacks a solid foundation on which to operate, the final design will have limited abilities. The components collected for the robot are from various vendors without knowledge of how they will integrate together. A customized enclosure is therefore required in order to bind the motors, wheels, battery, and PCB together in a meaningful way. While wood and metal could be fabricated to build a prototype, 3D printing is a much more attractive option. The following sections describe the different types of 3D printing available. A modeling tool is then used to integrate the physical components into a prototype. It is unlikely that the first prototype will be perfect and consequently design changes will be required. These design changes are discussed in order to benefit future designs and to prevent common problems.

5.1 Required Skillset

Before a design can be sent to a 3D printer for processing, it must first be modeled in a Computer Aided Design (CAD) tool such as Solidworks, Sketchup, and Rhino. Some of these tools require license or subscription fees, while others have free versions with limited features. If a mechanical engineer is not able to assist with the project, one of these tool sets must be learned in order to begin design of the model. It can take many years to master the use of a CAD tool but basic usage can be learned in a few days. This will allow for a sufficient prototype to be modeled that can serve as housing for the system of components.

When modeling with the intent of sending the design to a 3D printer, there are several factors to consider. Each printing technology and material will have a different specification for minimum wall thickness. Know the minimum supported thickness before starting a design in order to prevent re-work later on. It is important to design the model with minimal total volume in order to reduce the manufacturing cost. When possible, find non-load bearing faces and create windows or lattice structures in order to reduce the total material. One strategy is to build the model as desired without concern for total material and then find areas that can be removed without reducing the required strength. Finding multiple uses for components is another way to reduce total material. An example is to use the flat side of a battery pack to secure motors in place instead of printing a mounting plate.

5.2 3D Printing Technologies

3D printing was inspired by the laser jet printer and has been in development since the 1980s. The popularity of 3D printing has increased greatly over the last 5 years, making it accessible and affordable to anyone with interest in mechanical design. The options range from desktop 3D printers to companies that will print the design using higher quality machines. For the robotic chassis developed for this work, three different types of 3D printing were evaluated and used.

5.2.1 Form1 - Stereolithography (SLA)

The Form1 3D printer is a desktop printer developed by Formlabs, a company started out of MIT in 2011. It uses a laser to emit ultra violet light through a tank of light reactive resin onto a build platform [20]. Layer by layer the build platform moves up out of the resin as the laser hardens each point on the model. The finished piece hangs from the build platform like a stalactite. Depending on the structure of the model, build supports could be required in order to hold the model in position during manufacturing. Once completed, the model is bathed in isopropyl alcohol, dried, and the build supports removed so that the model is all that remains. This was the primary printing technology used to build the prototypes due to its accessibility and low resin cost.

5.2.2 Maker Bot 2 - Fused Deposition Modeling (FDM)

The Maker Bot 2 can use Poly Lactic Acid (PLA), a bioplastic derived from corn starch [21]. This plastic is housed in a reel and then slowly pushed into a thermally controlled emitter onto the build platform. FDM forms the model from the ground up, making a stalagmite. Build supports are also required if the model has extrusions that lack the necessary support. One model was printed using this technology and is in use by the final design.

5.2.3 Shapeways - Selective Laser Sintering (SLS)

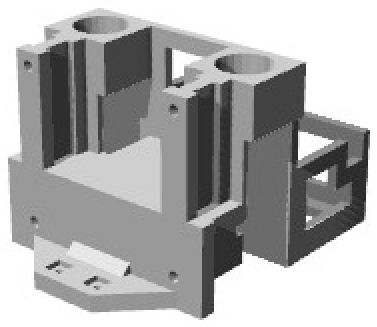
Shapeways is a company headquartered in New York with factories around the world. Many different materials are available for use including wax, plastic, steel, silver, and 14K gold amongst others. The most affordable and versatile material is a powder called PA 2200 designed for sintering. Sintering is the heating process that binds the powder together but does not solidify or melt it. A specialized machine will use a laser to sinter this powder together layer by layer. Any voids that would normally require build supports are instead filled with un-sintered powder [20]. Once completed, the resulting model is pulled from the bin of powder precisely as designed.

5.3 Design for Stability and Minimal Material

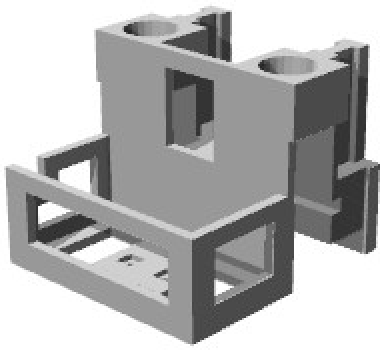
It is highly probable that multiple design drafts will be required in order to arrive at the final prototype. In the sub-sections that follow, the design process will be explained using the lessons learned in each chassis revision. A total of four revisions were produced, each one revealing new insight.

5.3.1 Chassis Revision 1 - Initial Concept

Figure 5.1 shows the first chassis designed for the robot in this work. Observe that the battery housing is connected to the back of the robot and has windows instead of solid walls. This reduces total model material but maintains an adequate structure. Another way to reduce material is to convert solid square columns into I-beam shaped columns. In many cases, this will retain the required strength but reduce the volume by 30% or more. In this design, an I-beam is used to support the PCB and motors, which mount vertically. Despite adhering to some of the basic methods of material reduction, a large amount of material, 70 cm^3, was needed. If only one platform is needed, this could be acceptable. Since robot collaboration requires several systems, a step towards material reduction is desirable. The design process in Figure 3.1 suggests that the mechanical design will require several iterations. Material reduction is one example of these design changes.



a) Front

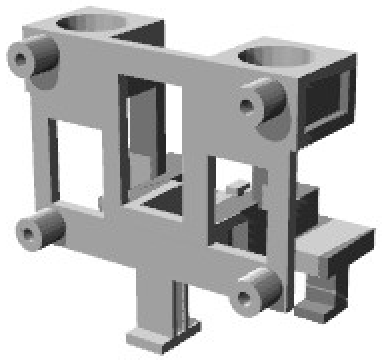


b) Back

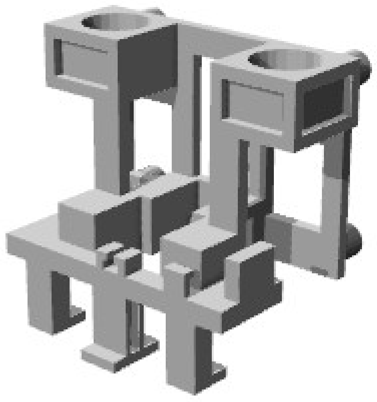
Figure 5.1 Mechanical Prototype - Revision 1

5.3.2 Chassis Revision 2 - Material Reduction

Figure 5.2 shows a massive reduction in total material used for the robot chassis. The total volume was reduced to 30 cm^3 but retained the major features needed to support the components. This was possible by implementing several major design changes, some of which required different components. Looking again at the design process in Figure 3.1, it is common and beneficial to consider components that have different dimensions or properties that can solve a problem. The first method used to reduce material in this design was to excavate all possible areas to a minimum but still retain reasonable strength. The thinnest wall is 3 mm in width, which proved strong enough for most areas except for the long and narrow sections on the sides. The second method used to reduce material was to select a smaller battery that would fit underneath the chassis, below the motor shafts. This lowers the center of gravity on the model and removes the need for the bulky battery case. It also doubles as a securing plate to the motors, preventing vertical movement.



a) Front

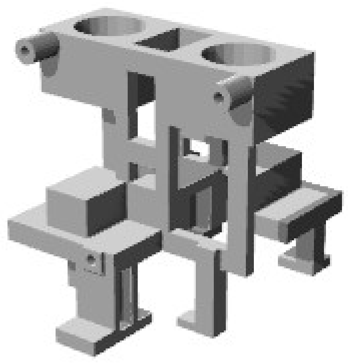


b) Back

Figure 5.2 Mechanical Prototype - Revision 2

5.3.3 Chassis Revision 3 - Motor Rotation

Unforeseen challenges can surface at any stage of the design process. One such challenge occurred in this work between the component selection and mechanical design phases. The Faulhaber DC motors chosen for this design have 90° output shafts but only in one direction, left. This requires that one gearhead be disassembled and flipped 180° so that it outputs on the opposing side, right. This process proved to be risky due to the 0.9 mm hex wrench required to perform the operation. The set screws would strip about 50% of the time, leaving the motor assembly as a left output only motor. A better design for these particular motors is to house them without the need to change the direction of an output shaft. This removes the possibility of damaging a motor and reduces the overall assembly time. Figure 5.3 shows these design changes, which maintained approximately the same total volume and solved the problem of motors requiring disassembly.


a) Front

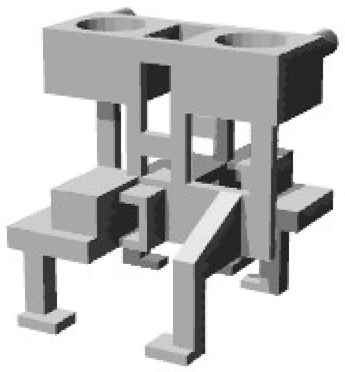
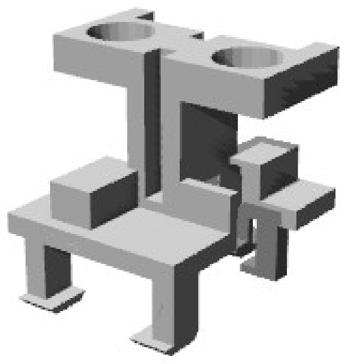

b) Back

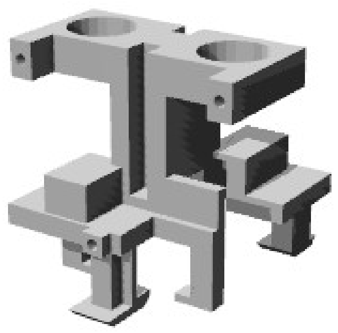
Figure 5.3 Mechanical Prototype - Revision 3

5.3.4 Chassis Revision 4 - Increase Stability

Solving a particular problem can sometimes reveal a new problem. In revision 3, the skid support system and battery case took a step backwards. The arms that hold the battery in place lacked enough supporting material to retain strength and were easily broken. Revision 4 solves this problem by creating a single I-beam down the middle of the model, removing the need for the remotely located support structures. The overall volume increased by 3 cm^3 but was deemed worthwhile given the added strength and simplicity. Figure 5.4 shows the final mechanical design.



a) Front



b) Back

Figure 5.4 Mechanical Prototype - Revision 4

5.3.5 Summary of Build Material and Cost

A summary of the total volume and build cost for the chassis is presented in Table 5.1. For the Form 1 and Maker Bot prices, an 80% markup was added to account for the build supports. Note that while the Form 1 and Maker Bot appear much cheaper, these prices assume the unit has already been purchased as a capital cost and only the build materials are accounted for.

Table 5.1 Material and Cost by Revision

Revision	**Dimensions (cm)**	**Vol (cm^3)**	**F1***	**MB***	**SW***
1	7.1w x 8.93d x 5.8h	70.2491	\$18.97	\$2.80	\$64.67
2	7.1w x 5.23d x 6.698h	30.4016	\$8.20	\$2.19	\$36.78
3	5.85w x 7d x 6.15h	30.5314	\$8.24	\$2.20	\$36.87
4	5.85w x 6.6d x 5.822h	33.1806	\$8.96	\$2.39	\$38.73

* F1 - Form 1, MB - Maker Bot, SW - Shape Ways

5.4 Evaluation of Technologies

All of the 3D printing technologies are effective tools for developing the chassis of a small robot, each with its own set of advantages. The major advantage of using a desktop 3D printer is that the model is finished in about 10 hours. Shapeways takes approximately 2 weeks to produce and ship the model. Regarding build quality, the SLS powder technology used by Shapeways is extremely precise and results in a pristine model, free of build supports. Both SLA and FDM technologies use build supports, which must be removed using a knife and diagonal cutters followed by sanding to account for the remaining burs left behind. The SLA resin material sanded better than the PLA plastic. All materials received the standoff screws well but the tolerances varied between them. Some required a small amount of drilling while others did not. The primary point being that each material must be evaluated for the particular use case. Once a material is chosen, the model can be printed on a different platform but expect slight differences in tolerances. Many prototypes were developed on the Form1 (SLA), which led to the final design for the robot in this work. For the final product, one model was ordered from Shapeways using SLS and another was printed on the Maker Bot 2 using FDM.

5.5 Summary

A 3D modeling program such as Solidworks or Sketchup is required in order to produce the file that the printer will use to make the part. This modeling stage allows for clearance issues to be detected before sending the design file to the printer. It also allows for the visualization of many different configurations. While certain clearance issues can be spotted in the design software, others are only discovered after the model has actually been printed. Depending on the type of 3D printer, it may be difficult or impossible to print a design. If this situation occurs, a redesign is required on the portion that fails to print. In some cases, a redesign will not solve the problem, at which point a different printing technology such as SLS should be considered.

CHAPTER 6: HARDWARE DESIGN AND CONFIGURATION

Similar to the mechanical design, the hardware design benefits from an iterative approach that moves from the evaluation board, to the bread board, and to the final PCB. This allows the individual components and hardware blocks to be verified before final integration. Essential configuration aspects of the microcontroller are covered in this chapter specifically targeting the STM32F4. Knowledge of these configurations are required in order to correctly layout the components. Other microcontrollers will have similar conventions especially concerning crystal selection. The power system, motor control, proximity sensors, and communication modules are also covered. These can be transported to other designs that do not use the STM32F as long as the MCU has the required hardware blocks. Figure 6.1 shows the high level view of the microcontroller and underlying hardware modules.

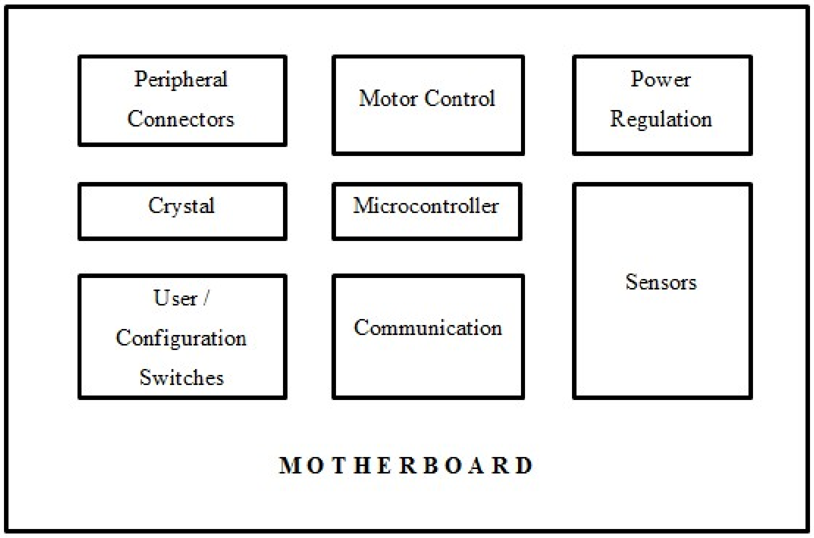


Figure 6.1 Hardware Block Diagram

6.1 Microcontroller Configuration

During the hardware design, it is critical to understand how the microcontroller will boot and what it will use as a clock source. Some MCUs will have multiple boot mode and clock source options. Other more advanced MCUs such as the STM32F4 have many options. These options will determine the layout and supporting components.

6.1.1 Boot Modes

One of the very first operations of the STM32F4 is to jump to a particular memory location. The STM32F4 has a sophisticated bootloader integrated into the chip. This bootloader allows the system to boot from different regions of memory. At startup, boot pins are used to select one out of three boot options:

1) Boot from user Flash
2) Boot from system memory
3) Boot from embedded SRAM

The second option, system memory, contains factory written firmware that initializes the MCU to a state where user firmware can be programed into flash bank 1, flash bank 2, or SRAM. Typically, user code is programmed into flash bank 1 so that upon reset the code can be executed on the target platform. Since flash is used, the code will be retained between power cycles. For faster debugging and prototyping, user code can be quickly programmed into SRAM, where it can also be executed upon reset with the limitation that the code is lost after a power cycle. Table 6.1 shows the boot pin configurations that enable booting to the various regions. Table 6.2 shows the particular section of the MCU memory map where booting takes place [22].

Table 6.1 STM32F4 Boot Modes

BOOT1	BOOT0	Boot Mode	Description
x	0	User Flash Memory	User Flash is selected as the boot space
0	1	System Memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

Table 6.2 STM32F4 Memory Map of Bootable Regions

Memory	Subsection	Address
Block 1 (SRAM)	Reserved	`0x2002 0000 - 0x3FFF FFFF`
	SRAM	`0x2001 C000 - 0x2001 FFFF`
	SRAM	`0x2000 0000 - 0x2001 BFFF`
Block 0 (FLASH)	Reserved	`0x1FFF C008 - 0x1FFF FFFF`
	Option Bytes	`0x1FFF C000 - 0x1FFF C007`
	Reserved	`0x1FFF 7A10 - 0x1FFF 7FFF`
	System memory + OTP	`0x1FFF 0000 - 0x1FFF 7A0F`
	Reserved	`0x1001 0000 - 0x1FFE FFFF`
	CCM data RAM	`0x1000 0000 - 0x1000 FFFF`
	Reserved	`0x0810 0000 - 0x0FFF FFFF`
	Flash	`0x0800 0000 - 0x080F FFFF`
	Reserved	`0x0010 0000 - 0x07FF FFFF`
	Alias to Flash, system memory or SRAM depending on BOOT pins	`0x0000 0000 - 0x000F FFFF`

6.1.2 Clock Source

The STM32F4 can use 3 different sources for its system clock; the High Speed Internal (HSI), High Speed External (HSE), or the Phase Locked Loop (PLL). Note that the PLL requires either the HSI or HSE clock. In a way, this means there are actually 4 clock sources; the HSI by itself, the HSE by itself, the PLL driven by the HSI, or the PLL driven by the HSE. Table 6.3 shows these clock sources and the possible speeds associated with them. The RCC clock configuration register (RCC_CFGR) contains bits to both select the clock source and to check the status of which clock source is actually being used since there are fall back mechanisms in the event one of the clock sources fails [23].

Table 6.3 Clock Options and Associated Speeds

Source	**Min**	**Max**	**Unit**
HSI (standalone)	16	16	MHz
HSE (standalone)	4	$26^{(1)}$	MHz
PLL (using either HSI or HSE)	24	168	MHz

Choosing which clock source to use depends on the target application and environment where the MCU will be used. The main reason external oscillators are used is to gain accuracy, reduce startup time, and resist changes to temperature depending on which type of external clock source is selected [24]. Anything from a simple RC circuit to another IC can be used as an external clock source. A crystal resonator resists changes to temperature better than the internal and IC-based clocks. It also has a faster startup time. Temperature changes can cause 4% of fluctuation. If this accuracy is not a critical factor, the internal oscillator can be used to save time and lower component count by 3 [18].

Changes in temperature are not critical for this particular robotic application. However, the platform being designed is general purpose, which could be deployed to uncontrolled environments in the future. An external clock source is therefore selected. In addition, UART data communication does not use its own clock. It instead relies on the system clock. It is therefore important to have the system clock be as accurate as possible. The targeted radio communication uses UART, so with these factors in mind the HSE crystal resonator was chosen. Since low power is not of critical consideration, the maximum clock speed of 168 MHz is desired. To achieve this speed, the HSE will be used to drive the PLL, which will multiply up the HSE crystal to the targeted speed. If power consumption does become of interest, the HSE or HSI can be used in standalone mode in order to evaluate power consumption at the lower clock speeds. The clock source chosen for this application uses the PLL driven by the HSE.

Several peripheral clocks are derived from the system clock, such as the Advanced High Performance Bus (AHB), the Low Speed Advanced Peripheral Bus (APB1), and the High speed Advanced Peripheral Bus (APB2). These clocks can be adjusted using the various prescalers. A low speed internal clock (LSI) is used for the watch dog timer while an optional low speed external clock (LSE) can be used to run a Real-Time Clock (RTC).

6.1.3 Crystal Selection and Configuration

Parallel resonance crystals contain a load capacitance value in the datasheet and are the types of crystals to use for external clocks (as opposed to series resonance). These crystals require 2 loading capacitors and possibly a resistor to make them function correctly. Choosing these components incorrectly could cause damage to the oscillator or prevent startup all together. The MCU manufacturer has supplied oscillator characteristics in the data sheet and a thorough application note in order to help determine the correct components to be used. The oscillator and microcontroller data sheets should be used in conjunction in order to determine the correct load capacitors and resistor. Table 6.4 contains the required values from the data sheet of a suggested 8 MHz crystal that was verified to be suitable.

Table 6.4 8 MHz crystal datasheet values

Crystal	**ESR**	C_L	C_O
FOXSLF/080-20	80Ω	20pF	7pF

Calculating the oscillator gain margin

Before calculating the components, the crystal specifications must be analyzed in order to determine if it will work with the microcontroller. The gain margin is used to decide this and gives a confidence level as to whether or not the oscillator will start. To calculate the gain margin, the transconductance of the microcontroller inverter (g_m) and the critical transconductance of the crystal (g_{mcrit}) are taken as a ratio. The oscillator will theoretically start if this margin is greater than 1. Changes in temperature and other electromagnetic effects can cause this margin to drop below 1, resulting in oscillation failure. The common rule of thumb is to have this value be greater than 5 [25]. The transconductance of the microcontroller inverter (g_m) is found in the data sheet for the STM32F4 and is represented in Equation 6.1. Equations 6.1-6.9 are derived from the manufacturers application note to calculate the correct components [26].

$$g_m = 5mA/V \text{ (min)} \tag{6.1}$$

The critical transconductance of the crystal (g_{mcrit}) is calculated using Equations 6.2, 6.3, and 6.4, where F is the crystal frequency, C_0 is the shunt capacitance, and C_L is the load capacitance.

$$g_{mcrit} = 4 \times ESR \times (2\pi F)^2 \times (C_0 + C_L)^2$$

6.2

$$g_{mcrit} = 4 \times 30 \times (2\pi 8 \times 10^{6})^{2} \times (7 \times 10^{-12} + 20 \times 10^{-12})^{2}$$

6.3

$$g_{mcrit} = 0.002158 = 0.221mA/V$$

6.4

For correct compensation, the transconductance (g_m) of the crystal should be larger than the critical transconductance calculated in Equation 6.4. Equation 6.5 shows that this is indeed the case.

$$g_m > g_{mcrit}\,, 5mA/V \;>\; 2.158mA/V$$

6.5

Taking the ratio of $g_m \; and \; g_{mcrit}$ provides the gain margin.

$$gain_{margin} = \frac{g_m}{g_{mcrit}} = \frac{5}{0.221} = 22.62$$

6.6

The oscillators gain margin is greater than 5 as recommended by the manufacturer's application note. The calculations provide confidence that the crystal will start oscillating when energized. Note that the manufacture's application note calculated a gain margin of 44 using an unspecified transconductance value. Since that application note was released in 2009, several new families of microcontrollers have been released, so performing the calculations again for this particular MCU proved to be a good practice.

Calculating the external load capacitors

Using an estimate that stray board capacitance (C_s) is 10pF, Equation 6.7 solves for possible solutions to the loading capacitor values.

$$C_L - C_s = \frac{C_{L1} * C_{L2}}{C_{L1} + C_{L2}} = 20pF - 10pF = \frac{C_{L1} * C_{L2}}{C_{L1} + C_{L2}} \tag{6.7}$$

Solving for C_{L1} and C_{L2} reveals many solutions, some of which are negative or not common capacitor values. It is common practice to have the two values match. Table 6.5 shows some of the solutions to Equation 6.7.

Table 6.5 Solutions to Equation 6.7

C_{L1} (pF)	C_{L2} (pF)
5	-10
14	35
15	30
20	20
30	15
35	14

From Table 6.5, 20pF is the best solution for C_{L1} and C_{L2}.

$$C_{L1} = C_{L2} = 20pF \tag{6.8}$$

Calculating the drive level and external resistor

The drive level must be calculated and checked to see if it is greater than or less than the $DL_{crystal}$ value found in the data sheet. The drive level for FOXSLF/080-20 is given as 100uW [27]. The actual drive level that will be presented to the oscillator from the microcontroller must be calculated while the oscillator is running with the capacitors. If the drive level is less than the 100uW rating, then no additional resistor is required. If it is greater, a resistor is required in order to not damage the crystal from excessive vibration. Measuring the current flowing into the crystal is an involved process requiring additional prototyping and test equipment such as a current probe or low impedance scope probe. Fortunately, the STM32F4 data sheet provides current measurements for two different oscillators, one of which is similar in specification to the one used in these calculations (the FOXSLF/080-20). The drive level in Equation 6.9 shows that it is sending nearly 12 times less power than the maximum ratting of 100uW and therefore does not require resistor R_{EXT}.

$$DL = ESR \times I_Q^2 = 30 \times (532 \times 10^{-6})^2 = 8.5uW \tag{6.9}$$

Oscillator PCB configuration

Figure 6.2 shows the recommended PCB layout from the manufacturer with the loading capacitors $C_{L1} and\ C_{L2}$ [26]. As stated above, R_{EXT} is not needed. Short traces and the use of a ground plane are encouraged in order to isolate signals and reduce noise.

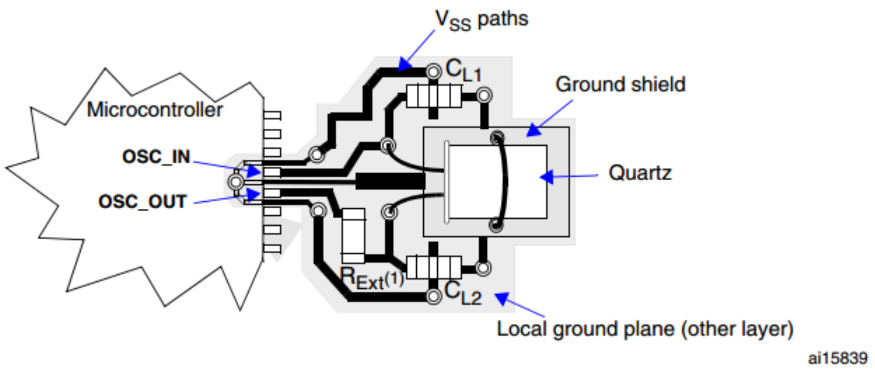


Figure 6.2 Recommended layout for an oscillator circuit

Observations regarding crystal selection

Going back and forth between the gain margin formula and available oscillator parameters revealed that it was very easy to have a gain margin greater than 5 if the crystal used was in the 8 MHz range or less. The STM32F4 microcontroller can use crystal values from 2-26 MHz. Available crystals have a tradeoff between effective series resistance ESR and load capacitance C_L. Lowering the load capacitance will cause the ESR to rise and vice versa. If 25 MHz is used for the oscillator the best gain margin found was 2.3. Some references suggests a margin of 2 to 3 as sufficient margin which suggests that perhaps the rule of 5 is over protective [25]. There is however no reason to use a 25 MHz clock if the PLL will be used as the clock source since any speed between 24 MHz and 168 MHz can be configured for the system clock using even a 4 MHz crystal. Since the FOXSLF/080-20 8 MHz crystal provides a margin of 22, much greater than 5, this was the chosen crystal.

6.1.4 Clock Speed Configuration

Contrary to initial intuition, a higher speed crystal does not result in higher system clock speed. The Phase Locked Loop (PLL) is the mechanism used to multiply a clock input. To avoid jitter, there is a recommended input clock speed of 2 MHz [23]. Since the acceptable ranges for the HSE are between 4 and 26 MHz, it does not matter which crystal is chosen because it will have to be divided down to 2 MHz before entering the voltage controlled oscillator (VCO) of the PLL. Since a 2 MHz input frequency is recommended, a crystal value that is evenly divisible by two is desirable. It was discovered in Section 6.1.3.5 that the higher the crystal clock frequency the lower the gain margin and hence a risk of oscillator startup failure. Considering these factors, a 4

MHz or 8 MHz crystal makes for a good choice, while a 25 MHz clock makes things more difficult and provides no additional benefits unless perhaps it was used without the PLL and exactly 25 MHz was needed for an application.

To set the system clock speed, the prescalers and multipliers must be correctly configured in order to prevent overclocking or underclocking sections of the hardware. Table 6.6 describes the function of each factor and its calculated value.

Table 6.6 PLL Scale Factor Descriptions and Values

Factor Name	**Description**	**Calculated Value**
PLLM	Division factor for the main PLL (PLL) and audio PLL (PLLI2S) input clock	8
PLLN	Main PLL (PLL) multiplication factor for VCO	336
PLLP	Main PLL (PLL) division factor for main system clock	2

The system clock can now be configured to run at the maximum frequency of 168 MHz using the PLLM, PLLN, and PLLP factors in Equations 6.10 – 6.12.

$$PLL\text{ output clock } = \text{ VCO frequency/PLLP } = 336/2 \ = 168MHz \tag{6.10}$$

$$\text{VCO output } = \text{ VCO input } \times \text{ PLLN} = 1\text{MHz} \times 336 = 336\text{MHz} \tag{6.11}$$

$$\text{VCO input } = \text{ PLL input / PLLM} = 8\text{MHz}/8 = 1\text{MHz} \tag{6.12}$$

Note that the reference manual recommends the VCO input speed to be 2 MHz in order to reduce jitter. It is impossible to reach a 168 MHz system clock if 2 MHz is used. A VCO input of 1 MHz was used instead since this is an even division of 2 MHz. These factors are then written to the RCC PLL configuration register (RCC_PLLCFGR). Upon system reset, the HSI oscillator is used as the system clock until the selected clock source becomes ready as indicated by the status bits. A clock source switch is then allowed and if successful the status bits will reflect the new clock source that now uses the new scaling factors. The main PLL configuration parameters cannot be changed once the PLL is enabled [23].

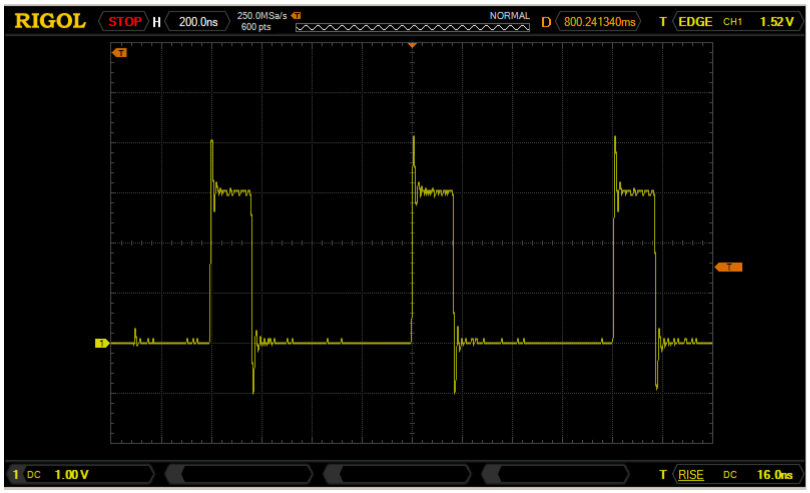
6.2 Programming/Debugging

6.2.1 Serial Wire Debug

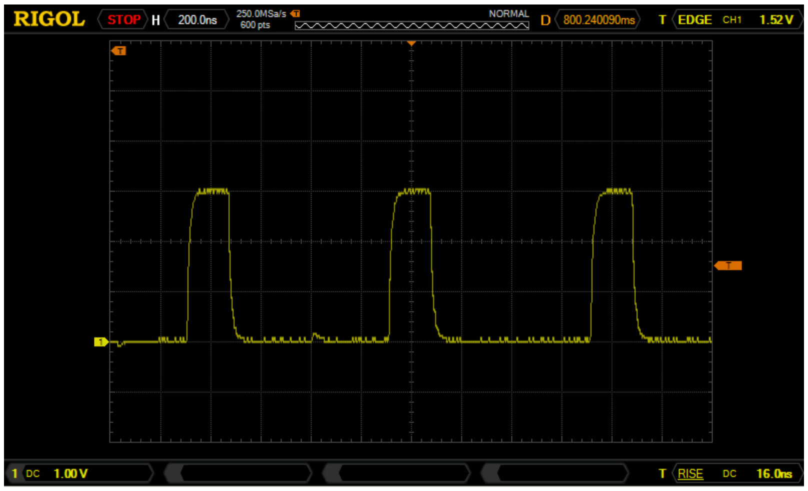
The Serial Wire Debug (SWD) protocol is used in many ARM processors as an alternative to the traditional JTAG protocol for debugging and trace. Note that SWD still uses the JTAG protocol as its foundation but implements a two wire connection instead of six. Consequently SWD does not replace JTAG but instead provides a simpler method to connect to it. While the SWD protocol itself only requires two wires, a ground and reset line are also required in order to hold the processor in reset while being programmed [28].

The STM32F4 Discovery evaluation board provides an onboard debug module known as STLink, which can be used to program the on board microcontroller or an external microcontroller embedded in a project. For ease of use and cable routing, 2 8” jumper wires (16” total length) were used to program the MCU. The STLink utility was however unable to recognize that a chip was connected. The fast solution to the problem was to use single 8” jumper wires but this made the programming board awkwardly close to the target PCB. Probing the SWDCLK and SWDIO lines during a program operation revealed that the clock was significantly degraded. Figure 6.3 shows the clock signal with significant overshoot. Note that this clock actually works and will allow the chip to be programmed but this is due to the additional capacitance of the probe. Once the probe is removed, the chip is no longer recognized by STLink. The actual overshoot and undershoot is so significant, there are likely double clocks happening.

For variable cable lengths ARM recommends introducing signal buffering with an inline 100Ω resistor [29]. Before considering a signal buffer it is common practice to add series resistance of a few hundred ohms in order to alter the RC time constant and improve the rising edge of the signal. Adding 330Ω series resistors with the SWDCLK and SWDIO signals improved the signal quality and allowed the chip to be recognized by STLink. Figure 6.3 shows the original signal followed by the improved signal.



a) overshoot/undershoot



b) improved signal - added series 330Ω resistor

Figure 6.3 SWDCLK signals - 16” jumper wires

6.2.2 USART/UART/RS232

Universal Synchronous-Asynchronous Receiver/Transmitter (USART) and Universal Asynchronous Receiver/Transmitter (UART) are protocols used during serial data transmission, which disassemble bytes of data into bits and then reassembles the bits back into the original byte. RS232 (Recommended Standard 232) is a set of electrical standards used for transmitting data between electronic devices and is commonly used with USART/UART. It was developed in 1962 and is still a common method for data transfer due to its simplicity and stability [30]. USART differs from UART in that a clock is shared between the transmitter and receiver in order to synchronize the data. This allows for a more efficient data transfer since start and stop bits are not needed. UART requires that synchronization data must be continuously sent in order to maintain an artificial clock. Since USART is more efficient over UART, it will be used for data communication. Depending on the chip configuration, the STM32F4 has up to 4 USART and 2 UART hardware modules for transmitting data. Native RS232 ports on laptops are extremely rare today. Future Technology Devices International Ltd. (FTDI) makes a USB cable with an integrated chip that performs the required level shifting and USB protocol encapsulation. This cable and the STM32F4 Discovery board connect to the custom RJ45 cable, which then connects to the PCB for ease of use.

6.2.3 Combining SWD and USART into One Cable

During the debug stage of an embedded system, many programming and debug cycles are done. This is a fairly straight forward process when the board can remain stationary next to the workstation. With a robotic application, the entire system needs to be disconnected, placed in the test environment, tested, and then reconnected for further adjustments to the program. Seven wires are needed to connect both the SWD and RS232 ports to the programmer and USB RS232 cable. Initially, jumper wires can be used to connect each of the appropriate pins on the board but this quickly becomes tiresome and consumes approximately 30 seconds for each iteration. Another option is to have two separate headers, a 4 pin for the SWD and a 3 pin for the RS232. This is better but time is still taken to ensure that the polarity is correct so the total time is reduced to approximately 10 seconds. The fastest cabling solution is a single cable where polarity is solved with the shape of the connector. An RJ45 Ethernet connector has 8 pins and robust housing to ensure a correct and solid connection after many connects and disconnects, making it a suitable candidate for 2 seconds or less connection time [31].

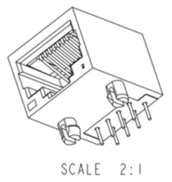


Figure 6.4 RJ45 Ethernet Connector

Table 6.7 shows the pinout from the RJ45 connector to the SWD and RS232 modules.

Table 6.7 RJ45 to SWD/RS232 Pinout

RJ45 Pin	Color Scheme	Module	Description	Destination Pin
1	White/Orange	RS232	RS232 GND	FTDI GND
2	Orange	RS232	RS232 TX	FTDI RXD
3	White/Green	RS232	RS232 RX	FTDI TXD
4	Blue	SWD	GND - SWD	SWD 3
5	White/Blue	SWD	SWDCLK	SWD 2
6	Green	SWD	SWDIO	SWD 5
7	White/Brown	SWD	NRST	SWD 4
8	Brown	UNUSED	UNUSED	

A custom cable is then made that breaks the connections out to the respective programmer and USB cable as seen in Figure 6.5.

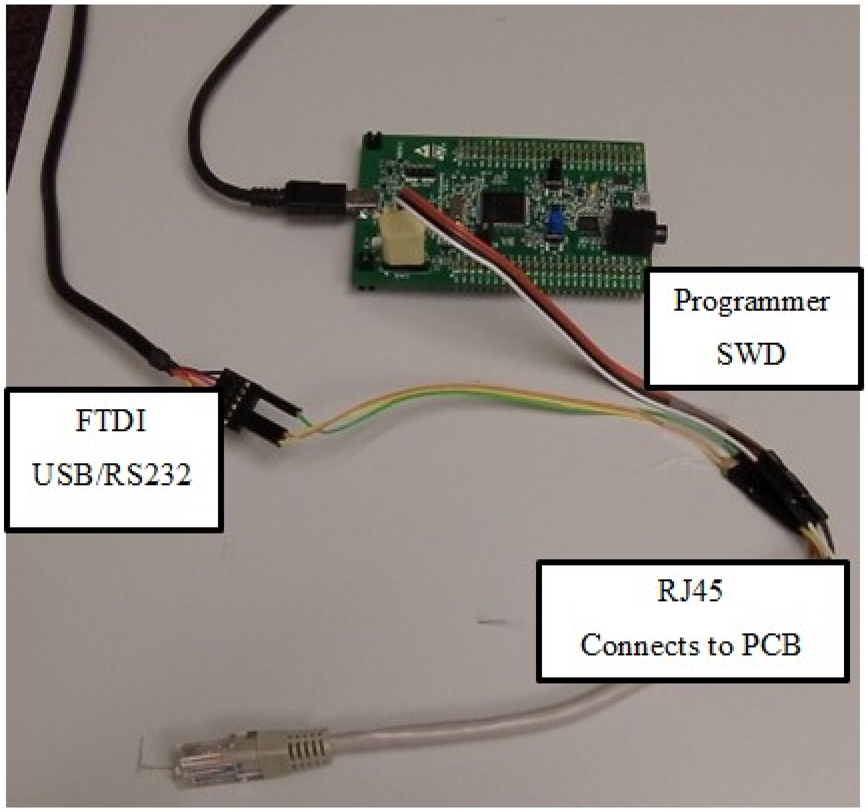


Figure 6.5 RJ45 Programming / Debug Setup

6.3 Power System

Lithium Polymer (LiPo) and Lithium Iron Phosphate (LiFePO4) batteries were targeted for the power source given the high energy density, small form factor, and low cost. An initial estimate shows that at peak power consumption, the entire system could consume approximately 16 Watts of power. Typical power consumption during operation will be significantly less since the motors are not both on at the same time all the time and the LEDs/Radio are also not on all the time. In order to measure the actual power consumption during operation, a current sense IC will be used. Table 6.8 shows the component power breakdown and then estimates the peak power.

Table 6.8 Estimated Component Max Power Breakdown

Component	**Quantity**	**Voltage Used**	**Est. Current**	**Est. PWR**
Faulhaber Motor	2	6V	1000 mA	6 W
STM32F4 MCU	1	3.3V	400 mA	1.32 W
XBEE Radio	1	3.3V	50 mA	165 mW
IR LED	3	5V	400 mA	3000 mW
FSH7773 IR LED	2	3.3V	400 mA	1320 mW
UV LED	1	3.3V	30 mA	99 mW
LED various	7	3.3V	1400 mA	4620 mW
Total			**3.88A**	**16.524W**

Given these peak estimates, the primary supply chosen for the system of components is a 500mAh two cell (7.4v) LiPo battery with a JST connector rated for 20-30c discharge. The pack weighs 32 grams and measures 14.5mm x 31mm x 57.5mm. A similar LiFePO4 pack has also been selected but has 700mAh 6.6v rating. The dimensions are important since the chassis was designed to fit these packs underneath the wheel axels. The chassis allows for additional parallel battery packs to be mounted on the back for longer duration tests.

6.4 Motor Control

6.4.1 H-Bridge Circuit

A mechanism is required to control the direction and speed of the brushed DC motors. In order to provide the ability to rotate the motor shaft forward and backward, an h-bridge is commonly used. The simplest h-bridge circuit involves 4 transistors and 4 diodes arranged so that digital logic can control the direction of current flow. Additional analog circuity is required if additional features are desired, such as brake and speed control. Several integrated circuits are available that integrate one or more h-bridge circuits into a single low cost package. The TI SN754410, TI DRV8833, and Toshiba SN754410 were evaluated for this project. The SN754410 is considered a quadruple half-h driver also known as a dual full h-bridge driver. A dual full h-bridge is capable of driving 2 DC motors forward and backward independently. While the SN754410 worked well on the breadboard, it consumed too much space on the final board since it is a through hole component. The DRV8833 initially appeared to be a valid candidate but proved to have a design that required 4 PWM inputs as compared to others that only require 2. The DRV8833 also uses a package developed by TI known as a "Power Package" or PWP. This package is considerably smaller than other h-bridge ICs but requires that the bottom center of the package be grounded to the PCB in order to maximize heat transfer. This requires that the part be re-flowed, which adds to the assembly time of the unit. Ultimately, the Toshiba SN754410 IC was chosen since it was a surface mount part that did not require reflow and only required 2 power inputs to control the speed.

6.4.2 Speed Control

In a DC motor, the armature is the current carrying mechanism that spins inside the casing in order to produce power when used as a generator or to convert power to mechanical torque via the shaft output. Equation 6.13 shows that the motor speed is proportional to the back EMF of the armature V_E and is inversely proportional to the field flux ϕ, which depends on the field excitation current [32].

$$Motor\ Speed = \frac{V_E}{\phi} \tag{6.13}$$

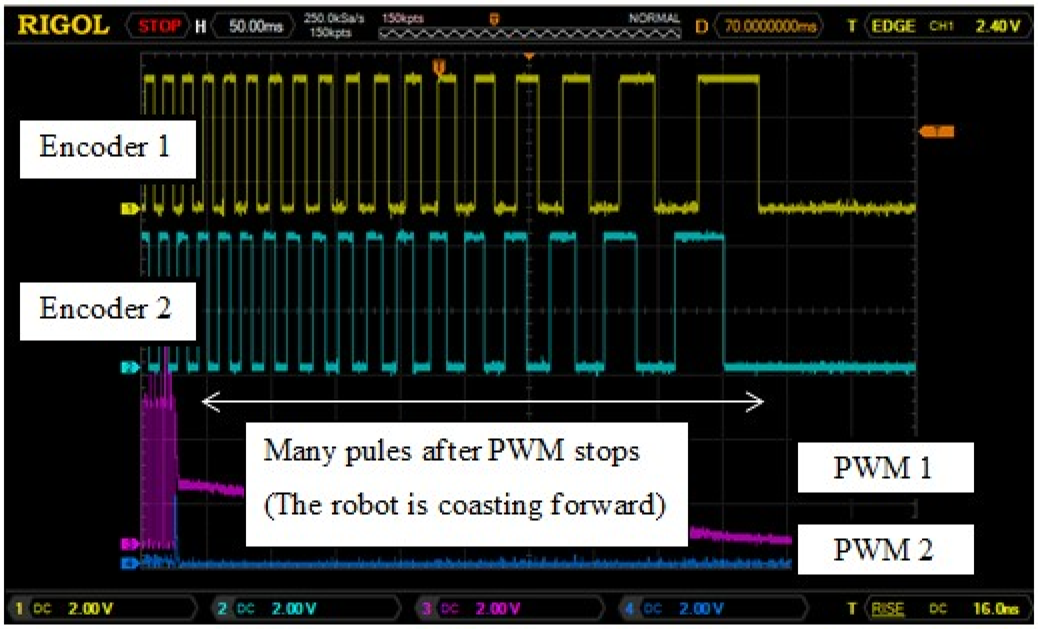
Since the field flux is typically a constant, changing the back EMF is used to vary the speed of the motor. The back EMF is the opposing force of the motor that is overcome by the applied EMF delivered by a power source. Considering the total voltage drop across the motor as the back EMF plus the drop due to the current and internal resistance of the motor, it is clear that varying the armature voltage will consequently vary the motor speed.

$$Armature\ Voltage = V_E + I_A R_A \tag{6.14}$$

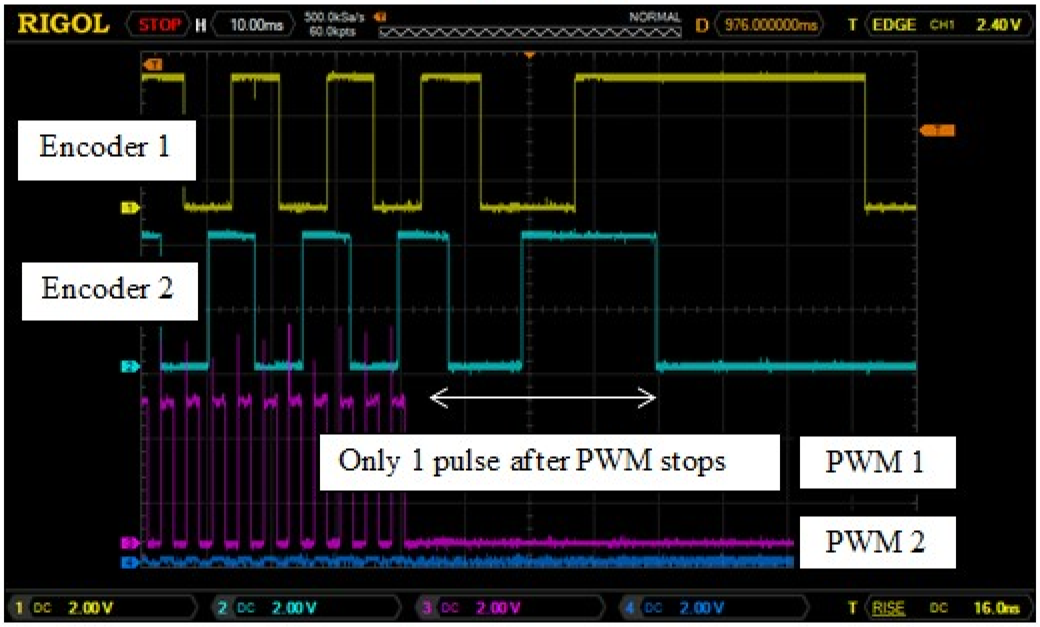
The motor of interest is rated for 6V but it has been determined through experimentation that supplying 5V to the motor still produces enough speed for interesting work in the lab. Consequently, the voltage range targeted for this application is 0-5V. In order to vary this voltage, pulse width modulation (PWM) can be used. PWM produces an average voltage by pulsing the supply voltage, 5V in this case, high for a percentage of time and low for a percentage of time. The total time is the period of the square waveform. The amount of high versus low is the duty cycle, which can be controlled using a microcontroller. The frequency of this square wave is adjustable and often in the range of a few kHz to 10 kHz+ depending on the motor.

6.4.3 Stopping vs Braking

Most h-bridge ICs provide the ability to brake the DC motor as opposed to simply cutting power to it. This is often called “short braking” because shorting the terminals of a motor will dissipate the generated electrical energy into the armature. In order to not exceed the electrical ratings of the motor, a resistor is placed across the armature terminals to dissipate the generated energy [33]. This is done inside the h-bridge IC and is known as dynamic braking. To measure the difference between stopping and braking, the encoder outputs and PWM inputs were observed on an oscilloscope. The signals are not related in time and therefore require that a single sweep be triggered and saved for analysis. Figure 6.6 shows the difference between a stop and brake under a no load condition. Observe that the stop condition shows 16/141 pulses of the encoder after power is cut whereas the brake condition only produces 1/141 pulses. Comparatively, this is an 11% error vs 0.7% error. Clearly dynamic braking or “short braking” is desired when precise movements are required.



a) Motor stop



b) Motor brake

Figure 6.6 H-Bridge Stop vs Brake

6.5 Proximity Sensors

6.5.1 LTE-302/LTR-301 IR Emitter and Detector

The LTE-302 is a 940nm IR emitter built into a square side emitting package. It is spectrally and mechanically matched to work with its counterpart the LTR-301 phototransistor. The side looking package reduces, if not eliminates, the need to baffle the detector from the emitter. The phototransistor is a NPN type Bipolar Junction Transistor (BJT). NPN is not an acronym. It instead represents the configuration of silicon with respect to the Emitter (N-Type semiconductor), Base (P-Type semiconductor), and Collector (N-Type semiconductor). An N-Type semiconductor has a spare negatively charged electron while the P-Type has a spare positively charged electron or “hole.” In a NPN phototransistor, current is allowed to flow once enough voltage is built between the base and emitter via the photoelectric effect. As a result, no current flows when it is dark, and the resistance across the emitter and detector is high. As infrared light is introduced to the sensor, more current flows from collector to emitter, and the resistance between collector and emitter is reduced.

These effects can be used for proximity detection since light reflects off solid objects such as walls. A voltage divider in conjunction with an analog-to-digital converter (ADC) can be used to measure the voltage changes between the emitter and collector. Since the V_CE nominal voltage of the LTE-301 is 5V, a metal oxide semiconductor field effect transistor (MOSFET) is required to switch the circuit on and off using the microcontroller’s 3.3V logic levels. The ability to switch individual emitter detector pairs is required to prevent sensor A’s emitter being detected by sensor B’s detector.

When nothing is in front of the phototransistor, only the ambient infrared will be detected and this base level can be marked in firmware as the obstruction free value. In the same way, an object is placed in front of the detector and its digital value is marked as the obstruction nearby value. Figure 6.7 shows the emitter with a 270Ω current limiting resistor and the detector acting as the second resistor in a voltage divider. IRR enables the circuit and ADCR is sampled by the MCU's ADC port. Note the importance of the 27k resistor in the voltage divider. The MCU's ADC voltage supply is allowed to go up to 5V, allowing voltage measurements up to 5V, but in this design it was configured to use 3.3V. Since 3.3V is the maximum value that the MCU is configured to measure, the voltage divider should be configured to output 0-3.3V.

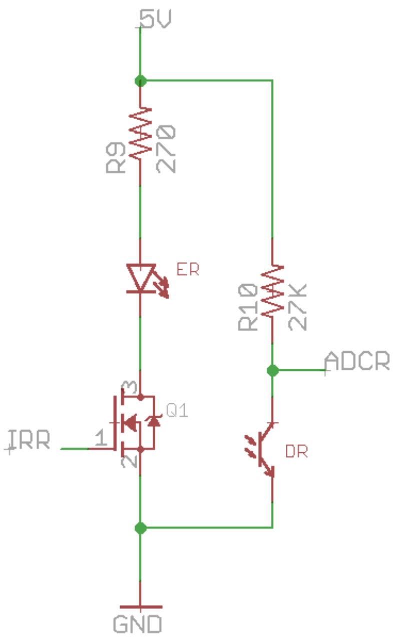


Figure 6.7 LTE-302 / LTR-301 IR Emitter / Detector Circuit

6.5.2 SFH 7773 Integrated IR Emitter and Detector

The SFH 7773 integrated circuit is produced by OSRAM containing two different phototransistors and an 840nm IR emitter. This allows the chip to detect both ambient and infrared light. It has many configuration registers such as LED current, integration time, and trigger modes, which allow it to be used in a variety of ways. It uses the I^2C protocol for communication but has one major weakness for robotics applications in that the address cannot be changed. This requires that each sensor have its own I^2C port on the MCU. The SFH 7773 is designed specifically for smartphone use in order to detect when the phone is next to a face in order to turn off the backlight and save energy. It also adjusts the screen brightness based on the ambient light present. For cell phone use, only one SFH 7773 is needed and therefore additional addressing options were not in the design.

The SFH 7773 is able to detect objects between 0-15 cm away depending on the configuration options [34]. To achieve this maximum distance, LED current has been set to the maximum value of 200mA and the integration time also set to the maximum time of 2.5mS.

There are many advantages to using an IC with features found in the SFH 7773. Separate emitter detector components often require a baffle in order to prevent flooding the detector but the integrated package provides this baffle by default. Without additional circuitry, separate emitter detector pairs can also be impacted by ambient light in such a way that recalibration is often required as the light changes. For example, if the sensor is used outside, the IR emitted by the sun will vary widely depending on cloud movements. A room mixed with tungsten and fluorescent lamps will have different IR intensities depending on where the sensor is in relation to the light sources. Since the target application is robotic, ambient light suppression is of interest. The SFH 7773 has taken this into account and bursts the LED on at 667 kHz for 3ms and then makes the data available after 7ms [35]. When the emitter receives the 667 kHz light that has reflected off of a nearby object, the internal circuit applies a filter to suppress the unwanted DC noise.

Because the target environment for this sensor is a cell phone, the sensor design took advantage of the two different power supplies that would exist in such an environment. The LED inside the SFH 7773 is intended to be driven directly from a single cell 3.7v LiPo battery. The IC itself is to be powered from the regulated 3.3V supply found on the PCB. The datasheet schematic in Figure 6.8 shows separate ground and voltage pins for these supplies [34]. Since this robotic application uses a 7.4v LiPo, additional design factors need to be considered. Because fast response time is desired for navigation, the LED is programmed to sample every 10ms. This creates noise at 100 Hz on the ground plane that powers the LED. In early stages of development for this project, these grounds were tied together. It was discovered that the noise was beyond the filtering capabilities of the IC and its recommended decoupling capacitors. While the chip still functioned, the I^2C data had erratic values that did not meet the data sheet specifications. It is theorized that the chip was constantly being reset by the power disruptions.

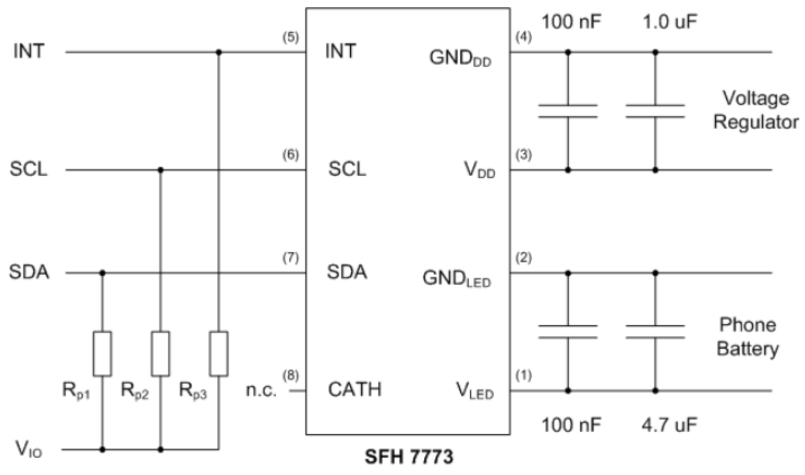


Figure 6.8 SFH 7773 Datasheet Schematic

In order to overcome the 100 Hz noise found on the common ground plane several modifications were made to the design. The primary goal is to emulate the DC power supply that the IC is expecting for the LED power. A dedicated 3.3V regulator was placed near the SFH 7773 to power the LED. Several different regulators were considered but ultimately the MCP1702 was chosen due to its low cost. The low price point comes with performance tradeoffs that must be taken into account. Figure 6.9 shows the load response of the MCP1702 after receiving a 100mA load step [36].

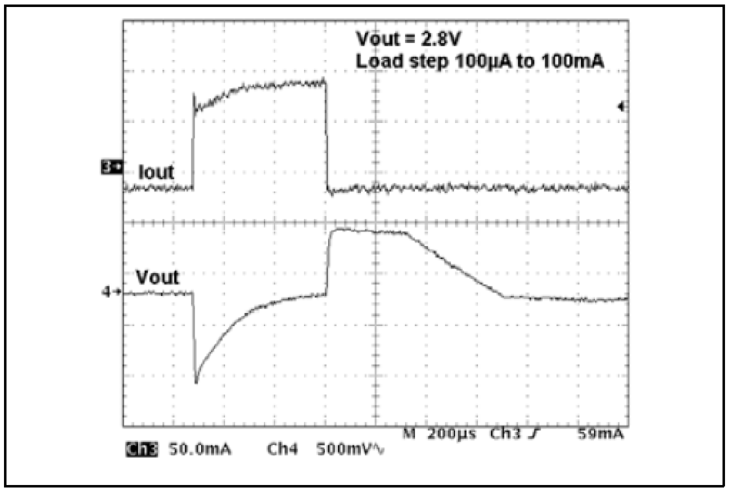
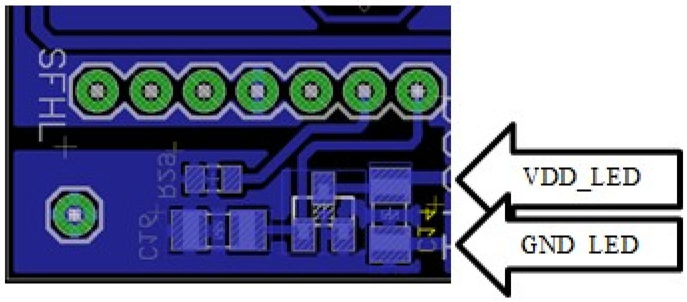
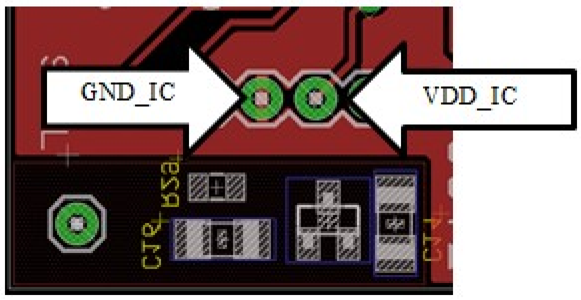


Figure 6.9 MCP1702 - Dynamic Load Response

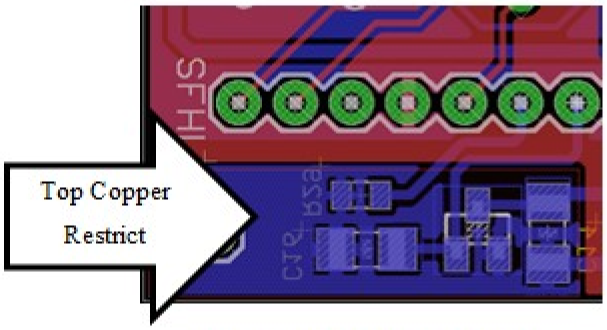
In order to overcome a volt or more of voltage sag and overshoot, 100uF capacitors were placed at the input and output of the regulator. While these capacitors added to the cost of the regulator, the total cost was still nearly a dollar less than the high performance regulators. Despite the dedicated regulator and load capacitors, the issue of the common ground plane still requires attention. In a cell phone application, a dedicated ground trace would route directly from the GND_LED pin of the SFH 7773 to the ground of the battery. This prevents the pulsating LED ground currents from interfering with the logic power supply. In order to provide similar ground isolation, a separate ground pour was laid around the FSH 7773 and its decoupling capacitors. A bridge onto the island was placed for V_LED and a bridge off of the island was placed for GND_LED. These paths have enough physical distance from the logic ground power pins to reduce interference. To prevent capacitive coupling between the relatively large capacitors and the top copper layer, a restricted routing zone was placed above the components. Figure 6.10 shows the Eagle CAD board view of this design.



a) Bottom Layer



b) Top Layer



c) Top and Bottom Layers

Figure 6.10 Ground Isolation for SFH 7773 Power Supply

Since the main board is vertically mounted and forward facing on the robot, daughter cards need to be mounted at 90° angles in order to sense nearby objects or walls to the left and right. The 7-pin header in Figure 6.10 (bottom) holds a right angle female header that receives the FSH 7773 breakout board. Figure 6.11 shows the CAD layout of the daughter card containing the FSH 7773, decoupling caps, and pull-up resistors for I^{2}C lines SDA and SCL. The jumper is used to connect the pull-up resistors if the main board or bread-board test environment does not already have pull-up resistors. The emitter and detector are mounted in such a way that the detector is always in front of the emitter regardless of whether the card is on the left or right side of the robot. Figure 6.12 shows the daughter cards after assembly.

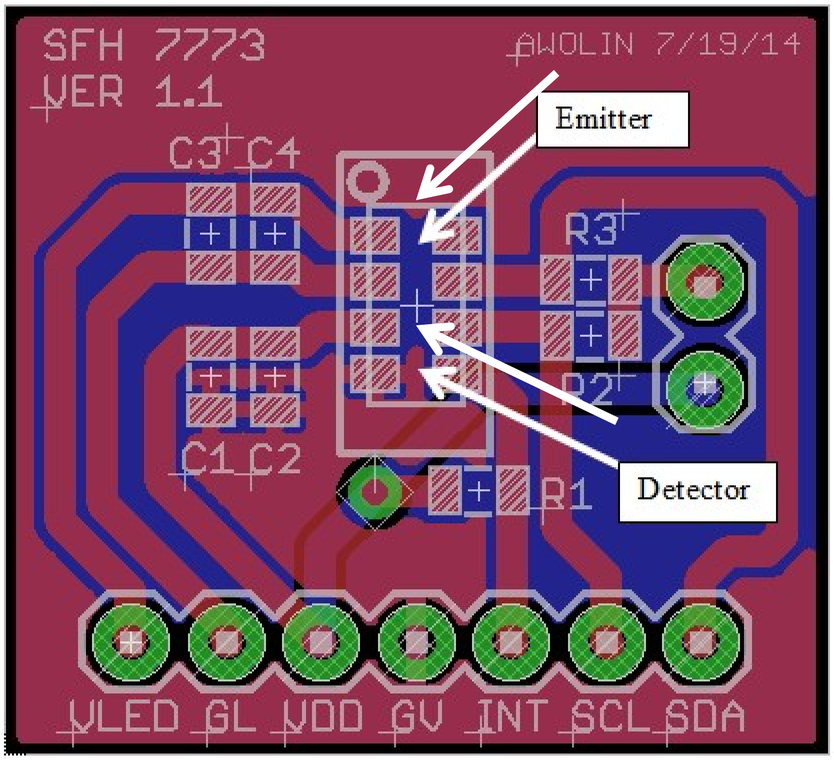


Figure 6.11 FSH 7773 Daughter Card

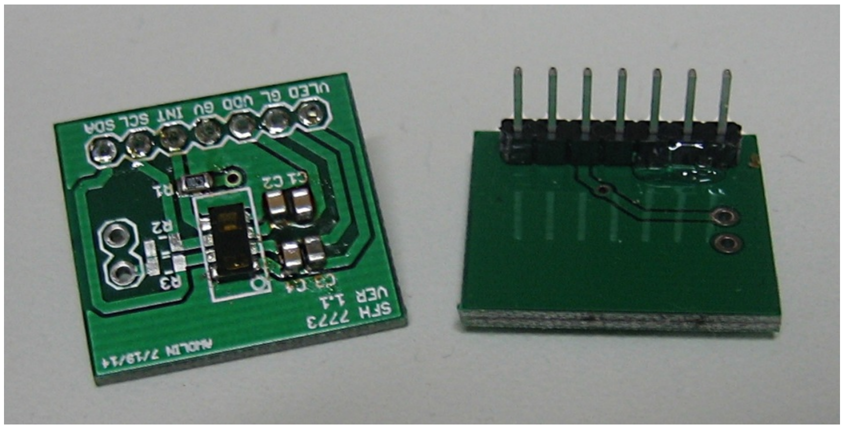


Figure 6.12 FSH7773 Proximity Sensor Daughter Cards

6.6 Communication

Various forms of RF communication exist that can be used for wireless data transport. Bluetooth, 802.11 b/g/n WiFi, and 802.15.4 have the means to communicate data between embedded systems. Extensive research exists in the area of wireless sensor networks using the ZigBee protocol. ZigBee uses the 802.15.4 radio standard and builds a networking layer on top of it. This allows for mesh, star, and other network configurations. The existing research and designs for wireless sensor networks can integrate into this robotic platform as it has an XBee radio module in the design. XBee has added yet another layer to the 802.15.4 standard in that they have taken the ZigBee protocol and have implemented it into a modular block of hardware that has additional microprocessor-like functionality. Figure 6.13 shows how the XBee not only provides a mechanism to use ZigBee, but it also provides GPIO functionality through DIO 0-3. It is often the case that data is being sent but the packets are not decoding correctly on the firmware side. To assist in debugging, red and yellow LEDs have been added to the RX and TX lines as a means of providing visual feedback to the developer during communication transactions.

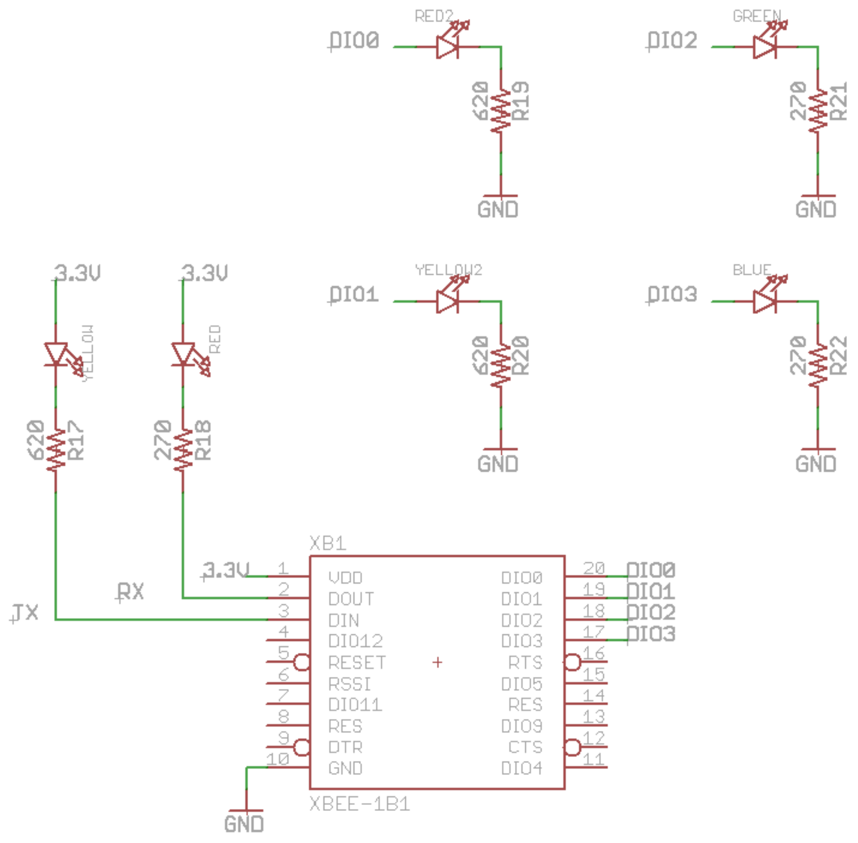


Figure 6.13 XBee Schematic and Debug LEDs

6.7 Assembled Motherboard PCB

The PCB was professionally fabricated and the 100+ parts were manually soldered to the board. The majority of the components use surface mount technology and require extra care when soldering in order to prevent bridges between pins and overheating the part. Figure 6.14 shows the side of the PCB that will face the front of the robot. The PCB mounts vertically on the robot, which allows for forward interacting sensors to directly mount to the PCB. Through hole components require the legs to be bent so that the sensor can face the appropriate direction. Figure 6.15 shows the back side of the PCB, which holds a small percentage of the components. The XBEE radio, RJ45 programming connector, and power management components have been placed on the back side to prevent overly complicated routing on the top layer. De-coupling capacitors have been placed beneath ICs whenever possible in order to provide the most effective reduction of power ripples.

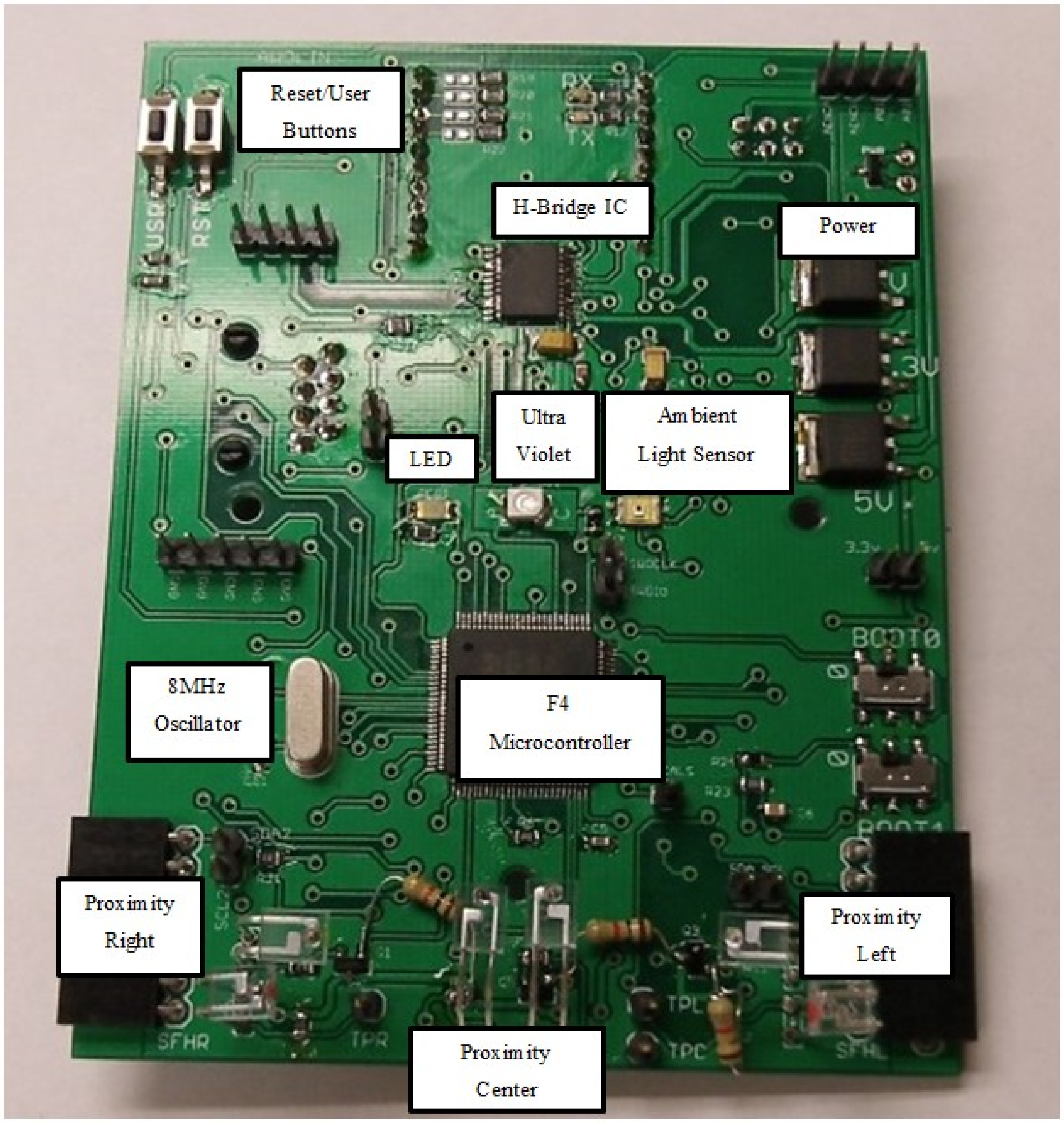


Figure 6.14 Robot Motherboard - Front

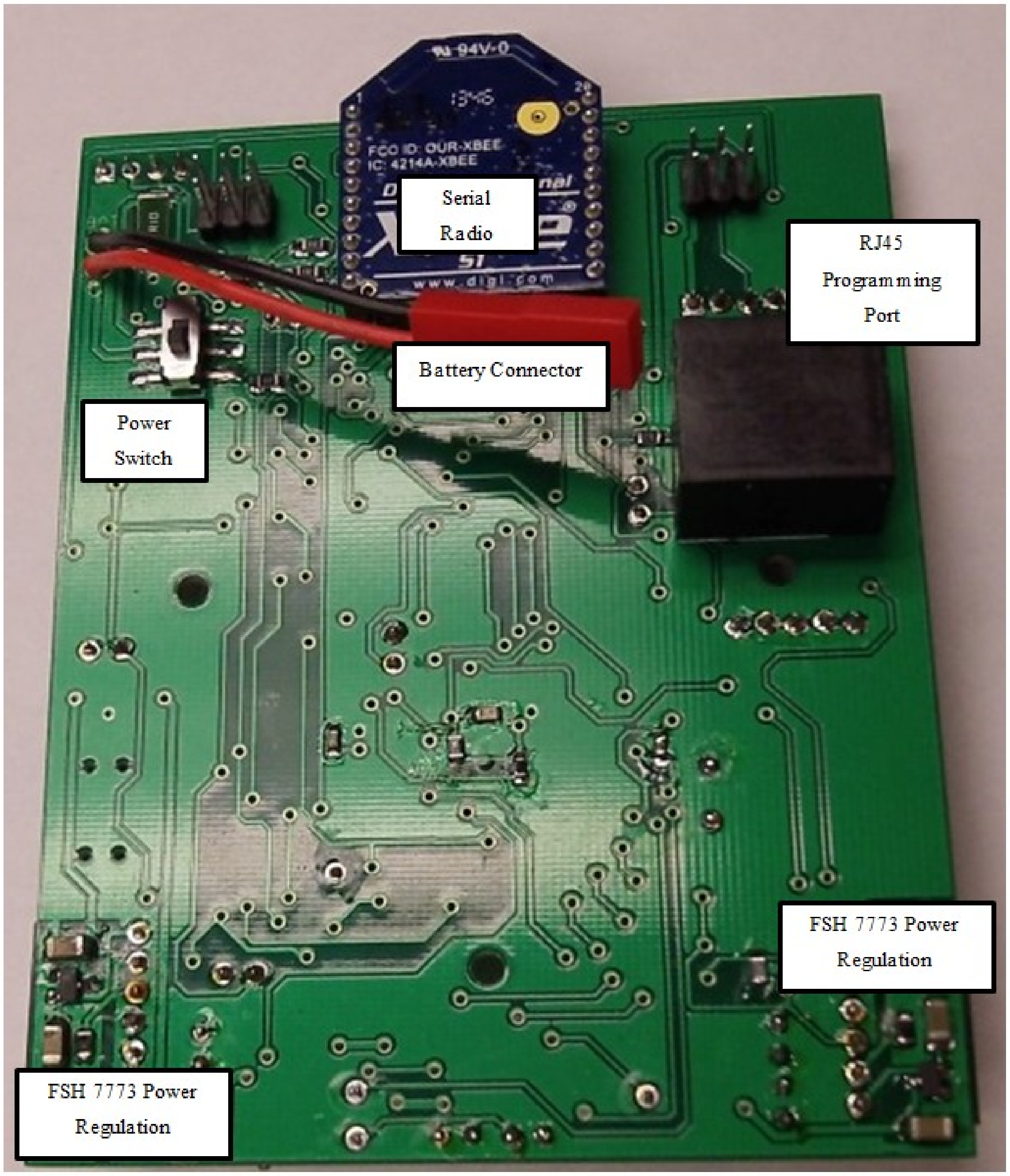


Figure 6.15 Robot Motherboard - Back

6.8 Summary

The hardware design contains many different subsystems that should be tested on a modular basis. Knowledge of the boot modes, crystal selection, and clock speeds are needed during the early stages of the embedded system design. A faster high speed crystal does not equate to a higher total clock speed when using the phase locked loop. Establishing this understanding from the start allows for a much safer design choice regarding the crystal selection. Testing each of the subsystems using an evaluation board and bread board revealed numerous issues. These issues were resolved before fabricating the PCB. Similar to 3D printing, certain issues were discovered after PCB fabrication. This requires wire wrap fixes and trace cutting. In the revised PCB, the routing mistakes were fixed and a RJ45 connector was added to ease the programming process. Issues will arise when integrating and prototyping the embedded system. Solving these issues invariably leads to new creative solutions, which can be leveraged in future designs.

CHAPTER 7: FINAL ASSEMBLY

When designing robots for collaboration, it is important to keep the final assembly in mind. Reducing the total number of parts will reduce the assembly time if numerous robots are to be built. In this particular design, only two Phillips head screws are required. These screws hold the PCB to the chassis while the rest of the components are friction fit into the chassis. The battery acts as a holding mechanism to prevent the motors from sliding out of place. Total assembly time is less than one minute. The PCB, chassis, battery pack, and motors assemble together as seen in Figures 7.1, 7.2, 7.3, 7.4, and 7.5.

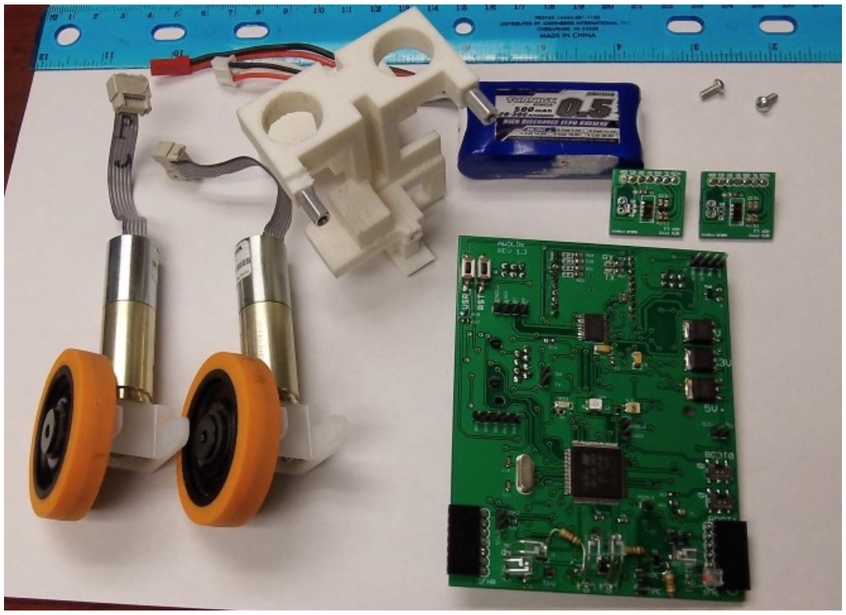


Figure 7.1 Robot Disassembled

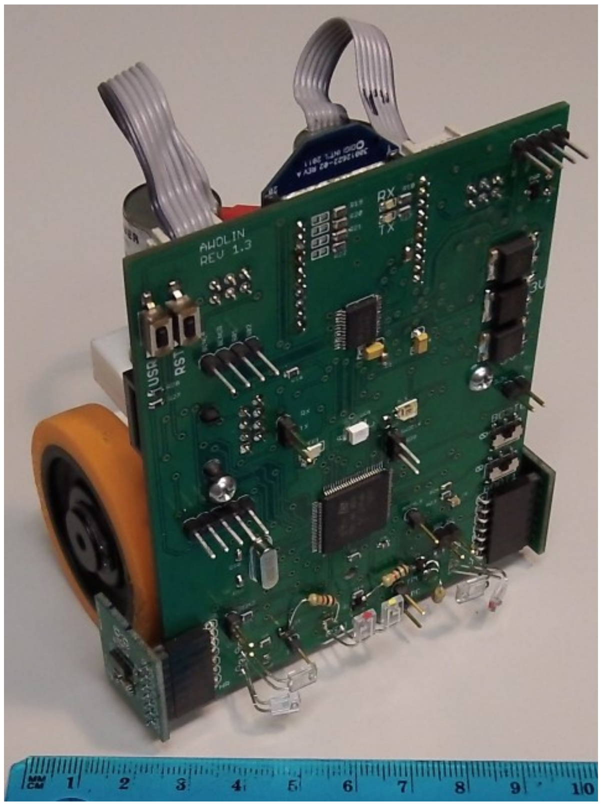


Figure 7.2 Assembled Robot - Front

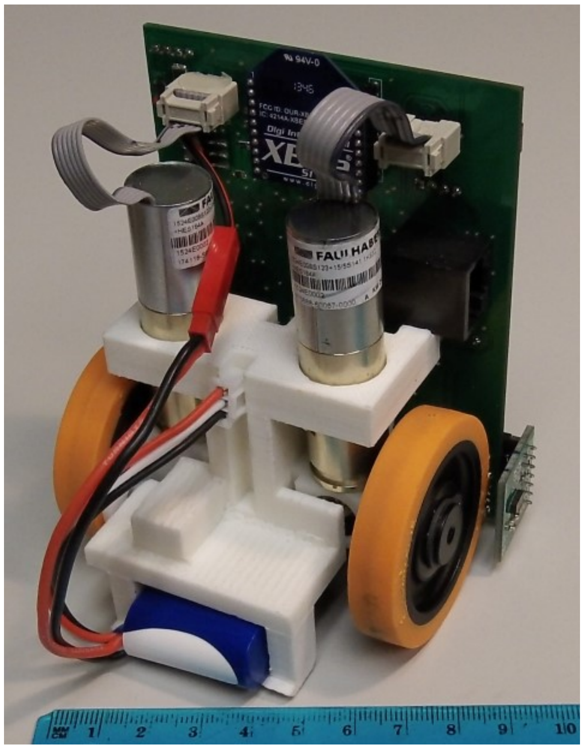


Figure 7.3 Assembled Robot - Back

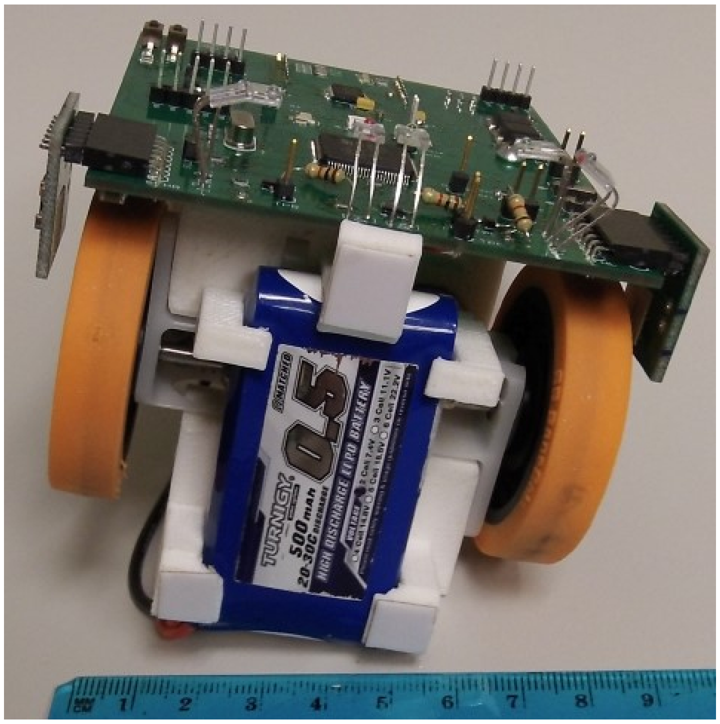


Figure 7.4 Assembled Robot - Bottom

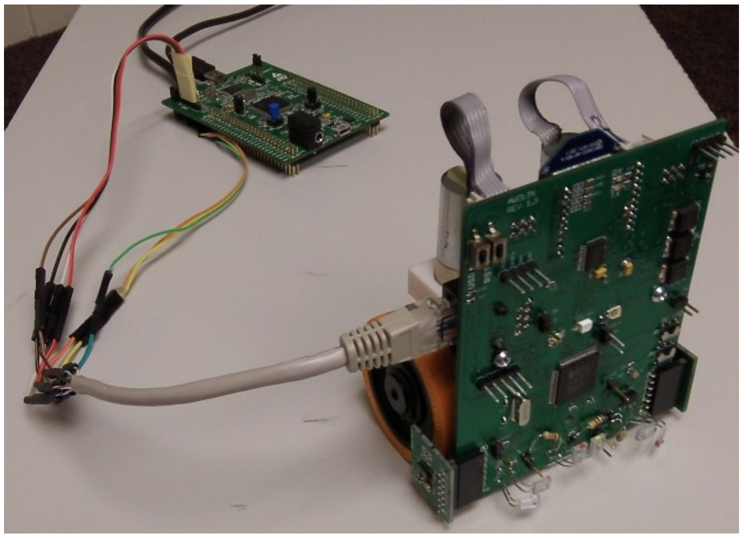


Figure 7.5 Robot with Programmer/Serial Port

7.1 Summary

The final assembly of all the components into the chassis is a significant milestone in a robotic project. This step reveals any clearance issues that were not accounted for in CAD. As mentioned in the hardware design chapter, an RJ45 port was added to the final PCB design. During final assembly, the motor mount clearance was overlooked. For a prototype of one or two robots, a small amount of dremel work can remedy this problem. Details like this are of much more concern if many units had been fabricated, all of which would now require re-work.

CHAPTER 8: SOFTWARE DESIGN

In an embedded system, the terms firmware and software are often interchangeable. When a real-time operating system (RTOS) is running on the microcontroller, it is more appropriate to use the term software since a large amount of the underlying hardware has been abstracted. In this chapter, key operating system concepts are discussed, such as cooperative and pre-emptive scheduling. The layered architecture and threading models are discussed, all of which are good design practices for any embedded system.

8.1 Cooperative Scheduling

The simplest code structure in an embedded system will run without direct use of an operating system. Cooperative scheduling is an operating system concept but is still used indirectly in embedded system code. Without an operating system, a main code loop is constructed that sequentially executes functions. The main loop can be thought of as a process and each of the functions can be thought of as tasks. Coding practices recommend that function calls use as little of the processing unit as necessary so that other blocks of code can execute on time. This is sometimes referred to as the “good citizen” model. Problems with this model arise when functions do not behave as a good citizen and take more time from the processor than what is agreed upon. This can be due to a bug in the code or an unforeseen delay on a sensor reading.

8.2 Preemptive Scheduling

A more advanced embedded system will start to incorporate a solution to the problem of “bad citizens” found in the cooperative scheduling. An interrupt service routine (ISR) is a hardware component of the microcontroller that will execute a block of code provided a trigger. A timer is often used as a trigger so that the code will execute at designated intervals. Other triggers could be the push of a button or an exceeded threshold from a sensor reading. Preemptive scheduling uses ISRs to prevent tasks from running beyond their allowed number of clock cycles. This is called the time quantum and can be configured to meet application-specific needs. Preemptive scheduling algorithms will use the concept of priority in order to optimize performance. This design allows each task to be assigned a priority level ranging from most important to least important. The algorithm can then evaluate the tasks that are in a ready queue and appropriately allow critical applications to run ahead of least critical applications. In a robotic system, it is very important to not allow the robot to damage itself or property. Since obstacle avoidance is of high priority, this should be given precedence over less critical tasks such as collaboration work.

8.3 Real-Time Operating System

An operating system changes into a Real-Time Operating System (RTOS) once it can meet or guarantee timing deadlines. An RTOS is also characterized by the ability to make rapid context switches, which is the ability to switch between tasks. The RTOS is also extremely small in size since it is typically executed on embedded hardware. ChibiOS has been selected as the RTOS to be used in this work given the widespread use in 32-bit ARM microcontrollers such as the STM32F4. It maintains an open source code base, which has allowed for extensive adaptations in many different fields of use. Other open source projects such as LCD library uGFX have assumed the same attitude of cooperation and provided compatible APIs that integrate into ChibiOS. ChibiOS is not the only RTOS that will run on the STM32F4. Other RTOSs can also be used such as FreeRTOS, Micrium, and RTLinux.

8.4 Software Layers

Using a real-time operating system suggests by default that a layered approach will be used in the software design. The RTOS provides the first layer of abstraction to the hardware and also adds the critical scheduling mechanisms needed to support threading and task management. The microcontroller requires an initial configuration in order to start the hardware modules used in the application. After this is successfully executed, the APIs are made available to the developer, including the serial port, I2C, ADC, PWM, and timers (both virtual and physical). Routines are developed based off of these APIs in order to share data, and navigate and map terrain based off of sensor readings. A complete set of routines have been developed for this work to verify the hardware functionality.

These functions have been built into layers, each layer unable to function without the layer beneath as shown in Figure 8.1. This practice allows for high code re-use and portability since application-specific details are removed through abstraction.

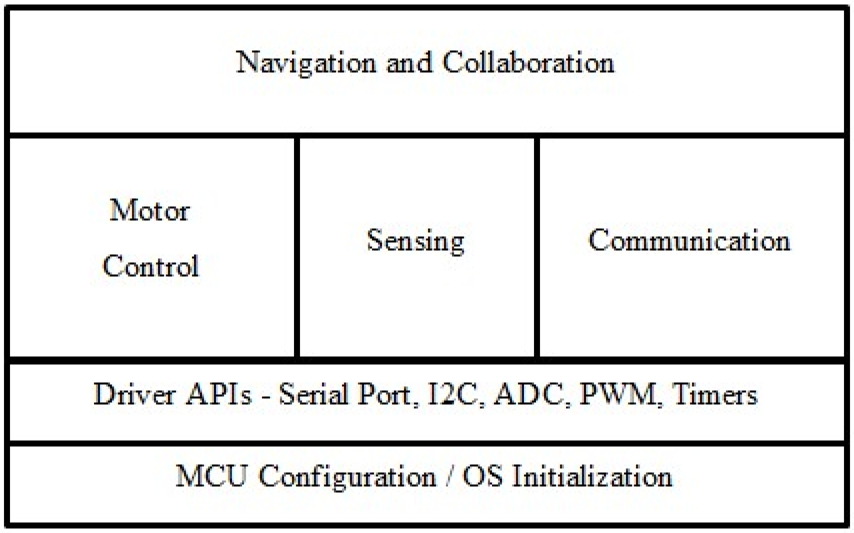

Navigation and Collaboration
Motor Control
Sensing
Communication
Driver APIs - Serial Port, I2C, ADC, PWM, Timers
MCU Configuration / OS Initialization

Figure 8.1 Software Layers

8.5 Threading Model

One of the major benefits to using an RTOS is the ability to create functional blocks of code, which can be organized into separate threads or tasks. The RTOS allows a thread to be created for tasks like sensor sampling. The thread can be given a priority in order to allow more or less clock time to be used for its execution. Each thread is then given a quantity of time to use the clock in a round robin style of rotation. Even though a sensor might take 10ms of time or more to come back with a value, code execution need not halt altogether. The motors can continue to drive forward while another thread maintains the counts of the encoders. Once the sensor does return with data, the thread then finishes execution and responds accordingly. Procedural programming is arguably the simplest form of programming but does not lend itself to highly complex tasks. Multi-threaded systems, in some ways, allow for easy to understand procedural threads to be run in a highly efficient manner. An example of how threads work in a robotic system is shown in Figure 8.2.

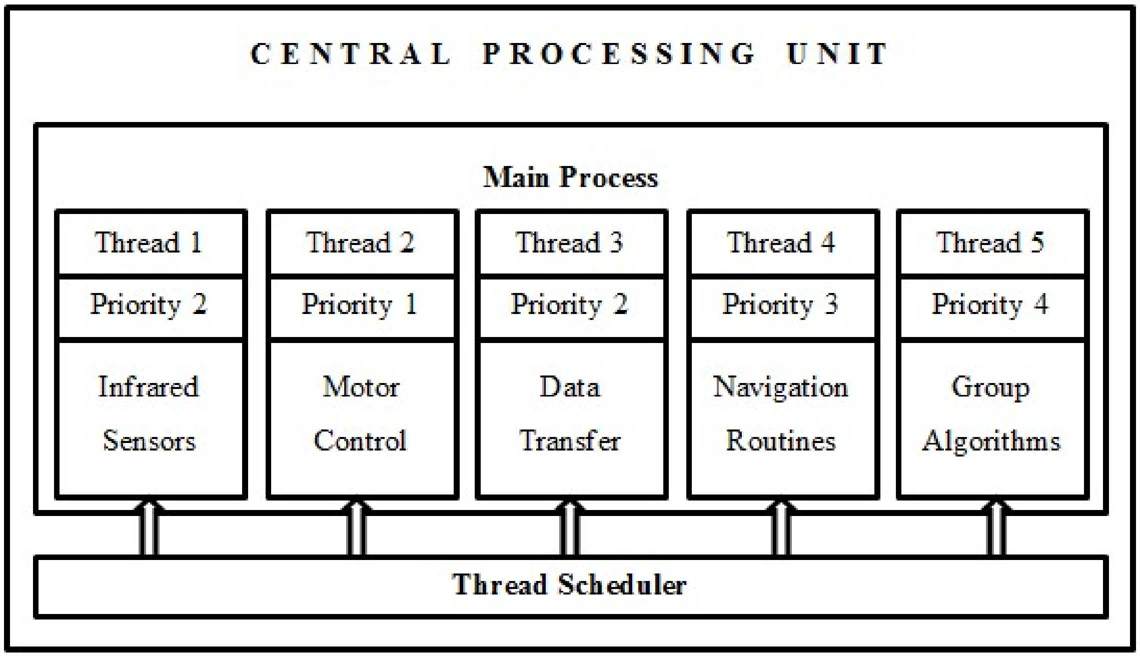


Figure 8.2 Threading Model

8.6 Collaboration in Hardware and Software

8.6.1 Identical Code Base

When mass deploying large numbers of systems, it is important to design the code so that the same software can be used on each robot. This allows for the project to scale with minimal complications. Managing multiple code bases is inefficient and prone to human error.

8.6.2 Unique Identification

In order to collaborate with many systems, and maintain identical software, there still exists a need to uniquely identify each robot. Many of the same concepts behind operating systems and task management begin to apply at the system level. Each robot needs its own unique identifier so that collaboration routines can execute tasks based on the states of each robot. Robots are assigned levels of availability much like tasks are assigned priority in an operating system. If for example a robot was in the process of climbing a narrow ramp and was not able to safely turn around, it would set an appropriate availability level so that the coordinator would not ask for assistance from that robot. In the case of the microprocessor in this work (STM32F4VGT6), the chip itself has a unique 96-bit identifier stored at 0x1FFF7A10. A table of friendly names is easily built in order to associate the identifier with "robot_1". This provides easily recognizable debug outputs for development. If the microcontroller does not have a unique identifier, the wireless radio certainly will. In this work, the radio is detachable, which makes it less desirable for use as a unique identifier. Robot_1 could have robot_2's radio if the radios were put into a box and then randomly re-attached. Tying the uniqueness to the chip prevents this scenario.

8.6.3 Robot States

With each robot uniquely identifiable using the chip id, a list can be generated of available robots (or nodes). Each node is a resource with an associated data structure that represents its state and attributes. Table 8.1 shows how each structure relates to each robot in the form of an array containing doubly linked lists. The collaboration routine is then able to make intelligent decisions to optimize use of each node based on health, availability, location, and power level.

Table 8.1 Robot State Structures

	Health	**Availability**	**Location**	**Power Level**
robot_1	[0-2]	[0-1]	[x,y]	[0-9]
robot_2	[0-2]	[0-1]	[x,y]	[0-9]
robot_n	[0-2]	[0-1]	[x,y]	[0-9]

8.6.4 Atomic Access to Hardware

Each robot must be equipped with a means to communicate with each node. This communication enables collaboration and is done using an 802.15.4 wireless radio in this work. The collaboration routine can run on a computer local to the network, which is acting as a coordinator. The collaboration routine can also be distributed across all of the available nodes. For this work, the use of a coordinator was used. In either case, atomic (or exclusive) access to the data structures is required in order to prevent race conditions. Operating systems use mechanisms such as semaphores and mutexes to properly control the access to critical areas of code. These same concepts can be applied to the larger problem of multiple independent systems sharing data.

8.7 Collaboration Algorithms

In general, an algorithm is a set of rules, simple or complex, which are used by a device to make decisions. The device could be a human, robot, or anything that can take action based on the results of the algorithm. Algorithms in robotics give a form of intelligence and enable autonomous behaviors. In individual robotics, maze-solving algorithms (never intended to be used with robots) can be used to allow a robot to find the exit to a maze.

An algorithm needs to be custom tailored in order to meet the specific design and end goal criteria. In a system of robots, a collaboration algorithm must be developed in order to efficiently manage their locations and activities. The algorithms developed will also depend on the type of network used for the system. If a star network is used, the central computer will run the algorithms and issue orders to the client robots. If a mesh network is used, the algorithms run individually on all of the nodes and each member must resolve conflict using shared data. The robot may only have data from a few neighboring nodes and must make decisions without knowledge of every robot's current state.

An applicable task to collaborative robotics is to create a map of a large area. This could be a square mile of land requiring containing landmines, which need to be disarmed. If a star network was used to solve this problem, the algorithm would be designed to disperse the individual robots into optimal intervals and then send them on task to mark the landmines. Large obstructions in the area will require the robots to navigate around them. This will lead to scenarios where robots must not collide with each other. If a faster route around the obstacle is discovered by a nearby robot, an intelligent decision can be made to have robots re-route and take the new path. A level of uncertainty must be accounted for. The algorithm must decide if it is better to continue around an object of unknown size, or turn around and take a known path around the object. This scenario is depicted in Figure 8.3.

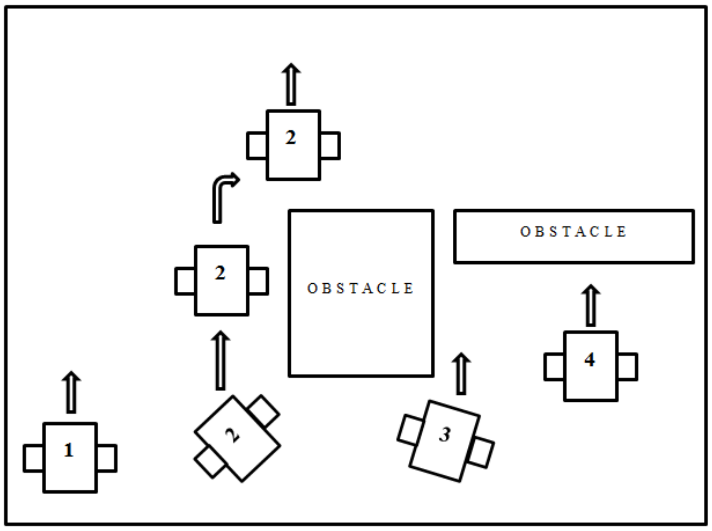


Figure 8.3 Group Obstacle Avoidance

As the number of units increase, the time required to solve a problem should decrease. This is true until the number of units becomes so many that more time is spent avoiding collision with other robots. This is another area where algorithms will be used to evaluate the number of robot detections compared to the number of fixed obstacle collisions. If robots are clustered together, preventing efficient movement, action can be taken to allow the robots to disperse to different regions.

Keeping track of the number of robot-to-robot collusions allows a threshold to be set. Once this threshold is exceeded, algorithms tailored for specific situations can be applied. In the case of congestion, the logic in Figure 8.4 can be applied in order to resolve the conflict.

```
1) If surrounded by neighbors, stop moving.
2) If movement forward is possible, move forward.
3) If movement left is possible, move left.
4) If movement right is possible, move right.
5) If movement backward is possible, move backward.
```

Figure 8.4 Outward Dispersion Algorithm

If the robots were placed in the center of a room in a group all facing one direction, this algorithm would evenly disperse them from the center out. Changing the order or removing certain steps will have significant effects on the group. If line 5 was completely removed, the robots would only advance forward and more time would be required to evenly disperse them. If line 5 is moved up to the third position as shown in Figure 8.5, the movement outward would be more elongated since forward and backward movement has a higher precedence over lateral movements.

1) If surrounded by neighbors, stop moving.
2) If movement forward is possible, move forward.
3) **If movement backward is possible, move backward.**
4) If movement left is possible, move left.
5) If movement right is possible, move right.

Figure 8.5 Linear Dispersion Algorithm

In the collaboration demonstration of this work, the algorithm is very simple. Data sharing allows each robot to stay on task until a critical update occurs. The critical update in this scenario is that the light has been detected and assistance is needed. A visualization of this algorithm can be seen in Figure 8.6.

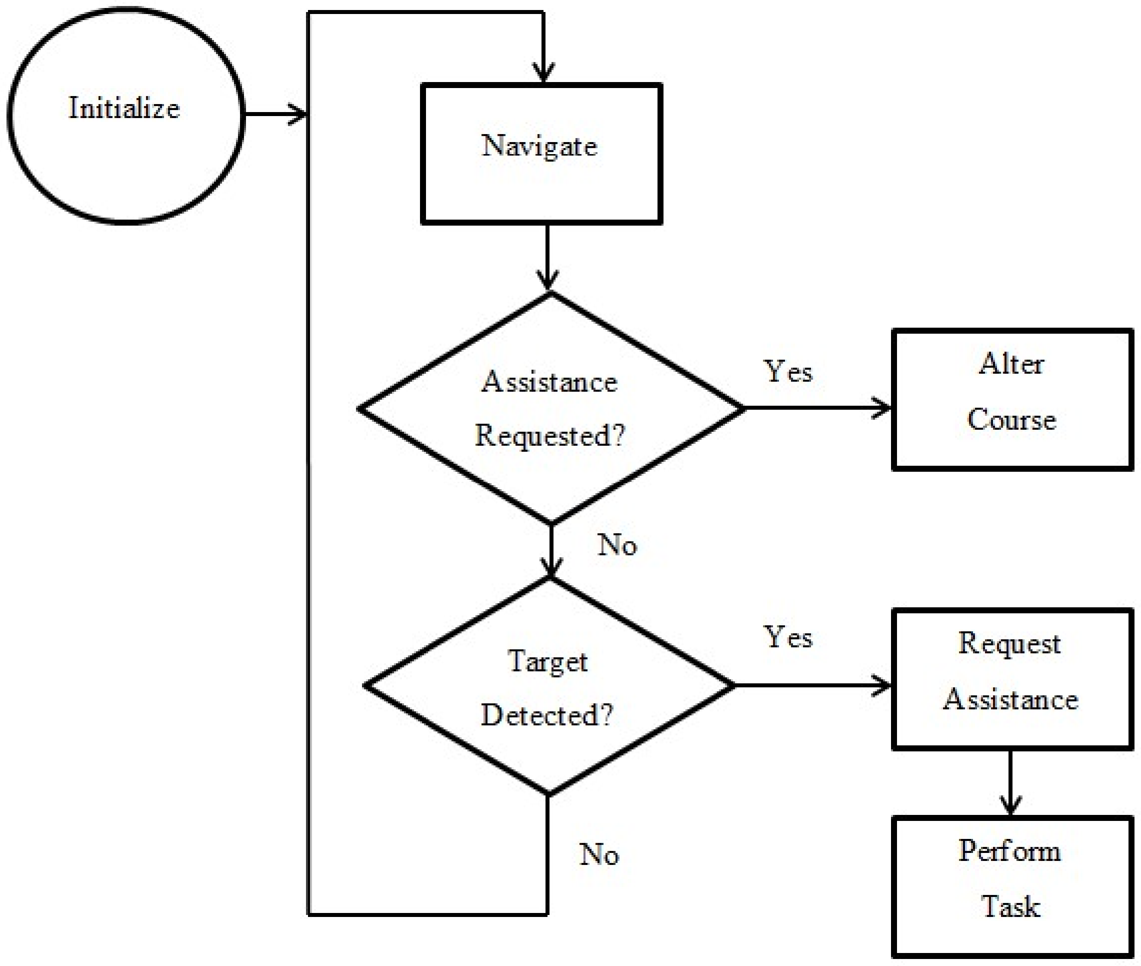


Figure 8.6 Top Level Collaboration Algorithm

Algorithms will often nest within other algorithms. In the scenario in Figure 8.6, two additional algorithms will execute once assistance has been requested. In this work, two robots are searching for the target. When assistance is requested, the algorithm is simple, send the only available robot. If many robots were tasked, the coordinator must search through a list based on current states such as distance, availability, and tool set (if the robots were not identical). Once the designated robot(s) are selected to assist, they must run a modification of the original top-level algorithm, which allows them to reach the appropriate destination.

8.8 Summary

A layered approach is often taken in any embedded system even if an RTOS is not used. It is however valuable to maintain this style of coding when using an RTOS since it provides the same benefits of portability and readability. The use of threads to manage logical blocks of code provides more of a software development environment compatible to algorithm and collaboration research. While threads can simplify some aspects of the logic, it introduces new factors that must be kept in mind such as concurrent access to the same device. It should be noted that while the RTOS does provide benefits of abstraction, the ability to navigate datasheets is still required in order to solve the problems that will invariable arise. In order to allow system collaboration, care must be taken when managing shared resources. Operating system concepts such as mutexes and semaphores prevent race conditions. Collaboration algorithms are required in order to efficiently manage large resource pools.

CHAPTER 9: TEST RESULTS AND HARDWARE VERIFICATION

Testing the individual components and subsystems before the final integration gives a certain degree of confidence that the entire system will function as designed. There are however many new issues and design challenges that can arise after integration. Electrical resonance and noise are common problems. The same set of tests done on a standalone basis should be executed on the final design. In this section, battery pack discharge data is analyzed in order to assess the value of LiFePO4 batteries in robotic applications. In addition, the LTE-302/LTR-301 proximity circuit and FSH 7773 sensors are tested to ensure proper functionality given various configurations. To test the entire system as a whole, a proportional derivative controller is used to autonomously navigate the robot down a corridor. Collaboration is demonstrated through a group problem solving exercise.

9.1 Battery Pack Discharge Data

As mentioned in the component selection chapter, two different battery types can be used to power the system. It is important to note that the two packs are very similar in size and price but the amp hour rating of each pack is different. Figure 9.1 helps to illustrate the discharge differences between LiFePO4 and LiPo. Since the 5V regulator for the motors requires 500mV of overhead, operation should terminate at 5.5V. The microcontroller will monitor the current sense IC for this value and then halt execution. In the case of the LiPo pack, the entire capacity can be used, which lasts 35 minutes given a 700mA load. In the case of the LiFePO4, the pack sinks to 5.5V after 56 minutes due to its higher capacity. The pack can safely be discharged to 4V, so the last 2 minutes of runtime is unused. While both packs can be used, the LiFePO4 has a better value proposition for robotics since it is slightly cheaper, safer, more environment friendly, and can accept 4 times the charge rate over LiPo. The negative aspects of LiFePO4 batteries, weight and discharge rate, have low impact for small robots.

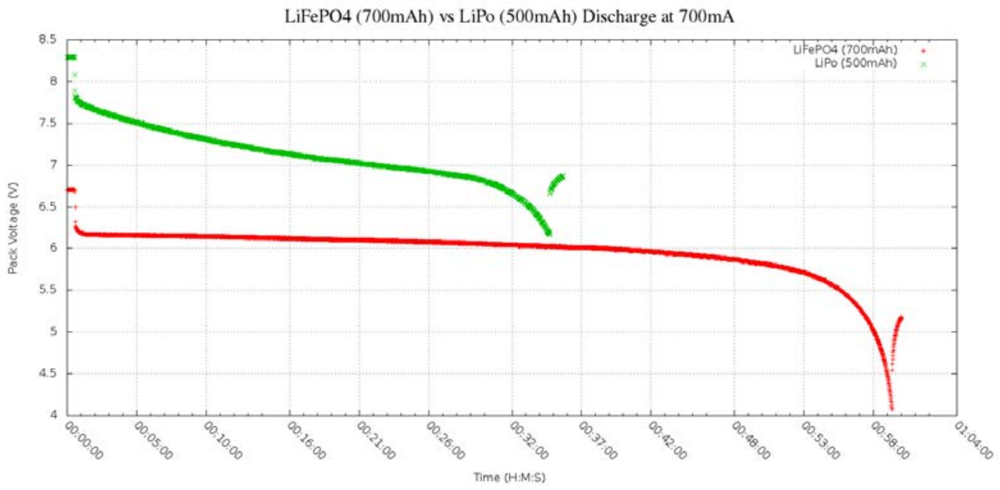


Figure 9.1 LiFePO4 vs LiPo - Discharge Data

9.2 LTE-302/LTR-301 Proximity Circuit Test Results

In order to verify the traditional IR sensing circuit from Figure 6.7, a series of tests were performed on the assembled robot with and without a shroud. The shroud is placed around the phototransistor to prevent ambient light interference. In theory this makes sense, but actual data will be used to make the final decision on whether or not to use a shroud. In addition, the amount of IR light emitted will be evaluated. In theory the more light emitted, the more light reflected and a higher change in resistance can be detected. Turning on more LEDs comes at the cost of more power. Testing will show

how many IR LEDs are needed in order to detect a distance of 5 cm (the distance needed to turn around without scraping the chassis against an obstacle).

9.2.1 LTE-302/LTR-301 Distance Measurements - Without Shroud

Figure 9.2 shows the response when 1 IR LED is on and Figure 9.3 shows the same test but when 3 IR LEDs on. Figure 9.4 makes the comparison clear and illustrates that all 3 available LEDs should be turned on in order to detect the target distance of 5 cm. It is also clear that 5-6 cm is the maximum distance that can be reliably detected.

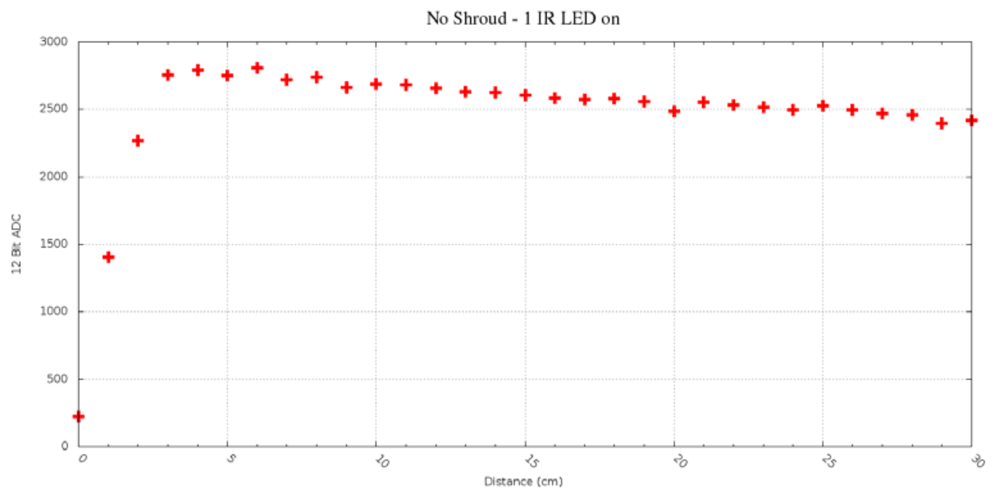


Figure 9.2 No Shroud - 1 IR LED on

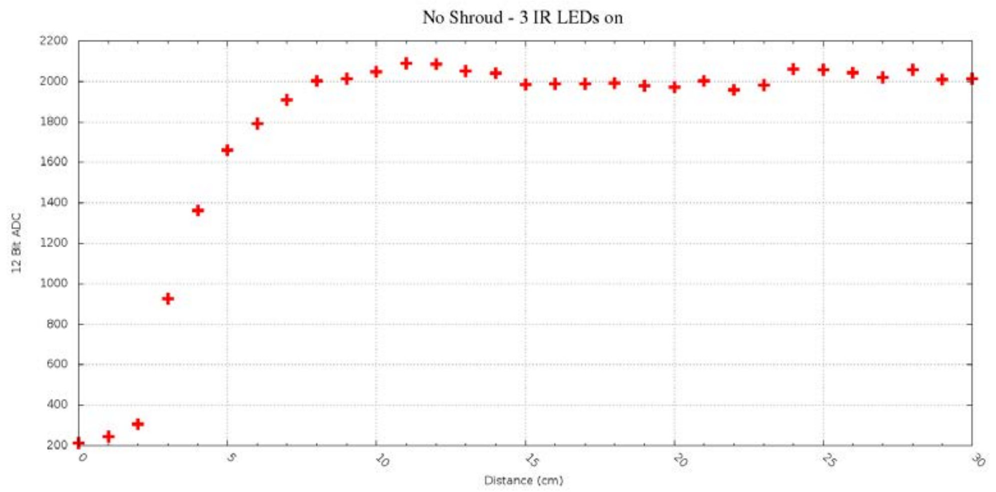


Figure 9.3 No Shroud - 3 IR LEDs on

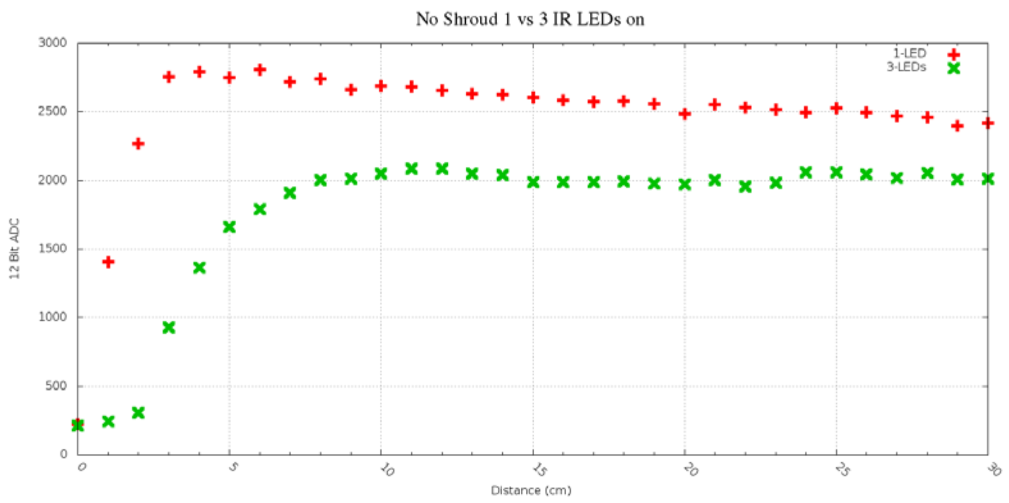


Figure 9.4 No Shroud - 1 vs 3 IR LEDs on

9.2.2 LTE-302/LTR-301 Distance Measurements - With Shroud

Placing a shroud (masking tape) around the phototransistor and performing the same tests as done without the shroud reveals some interesting points. Figures 4.4 and 4.5 show the same tests as done before with 1 LED on and 3 LEDs on. Figure 9.7 summarizes the data and again shows that 3 LEDs are required to reach 5 cm reliably.

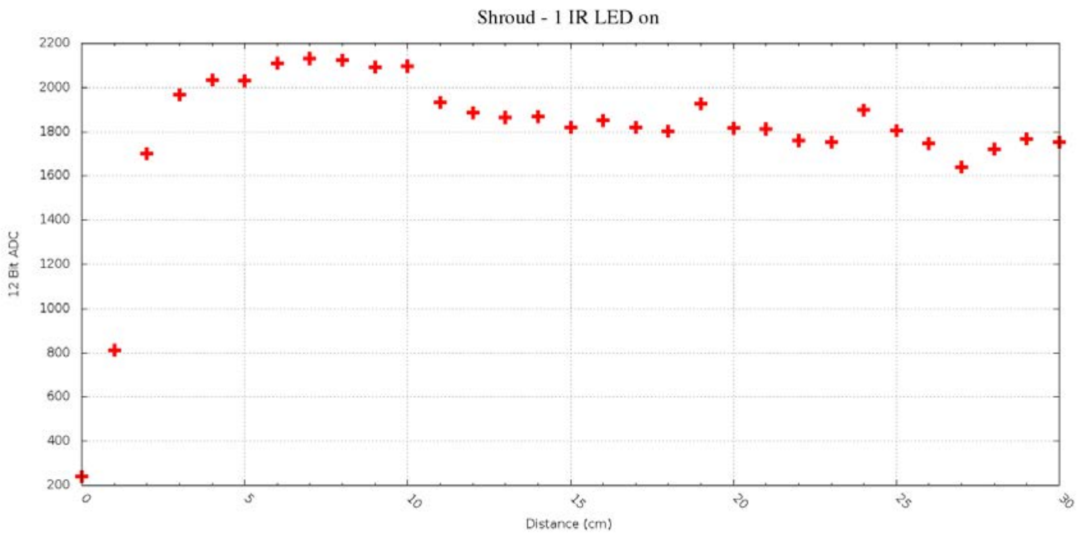


Figure 9.5 Shroud - 1 IR LED on

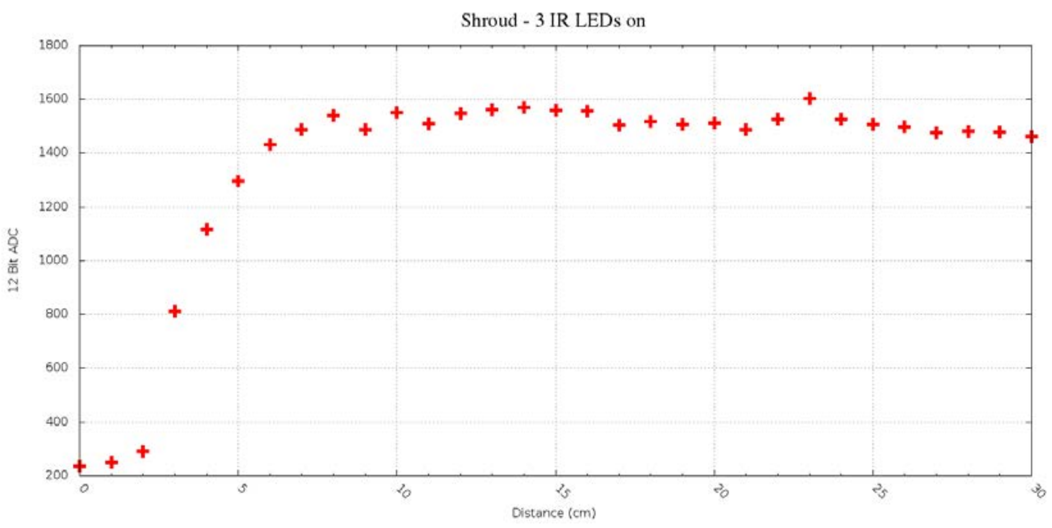


Figure 9.6 Shroud - 3 IR LEDs on

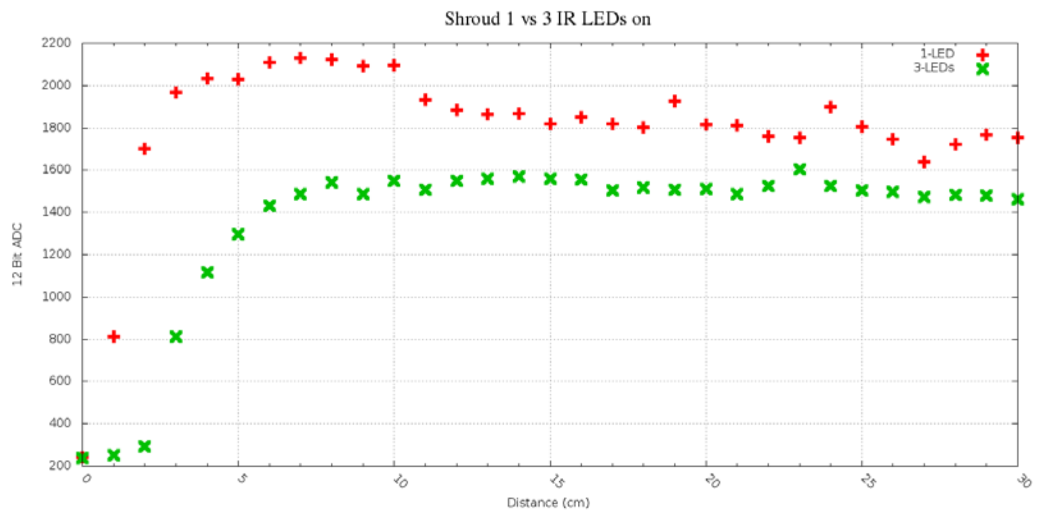


Figure 9.7 Shroud - 1 vs 3 IR LEDs on

9.2.3 LTE-302/LTR-301 Shroud vs No Shroud Comparison

In order to determine if a shroud should be used, the data in Figure 9.8 is considered. The maximum range when a shroud is used is slightly less than the version of the test without a shroud. While ambient light is prevented from interfering from the sensor, it also appears that some of the intended radiation is unable to be received. Note that these emitter and detectors are side emitting and a shroud could be more beneficial on the 5 mm round versions that are available. Operation without the use of a shroud is the recommended configuration provided this data.

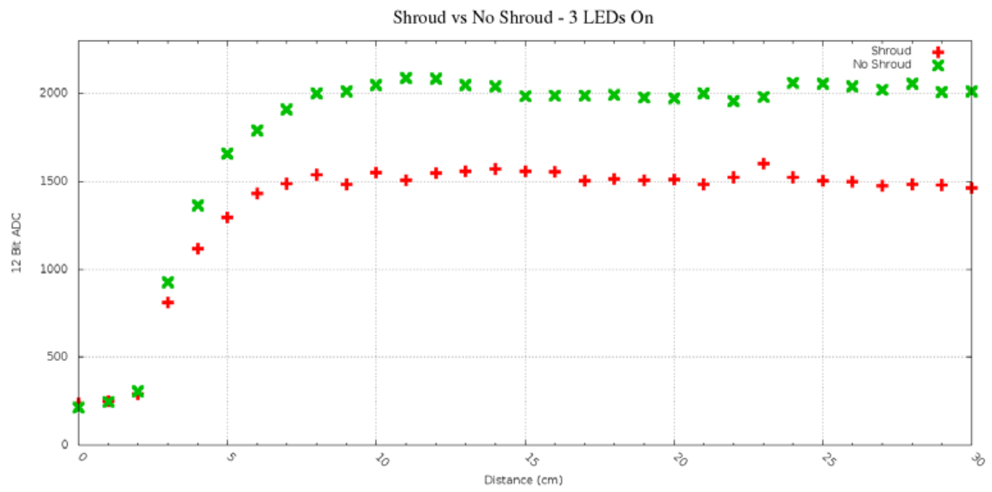


Figure 9.8 Shroud vs No Shroud - 3 LEDs on

9.2.4 LTE-302/LTR-301 Noise Test

Since this is an analog circuit, a certain amount of noise is to be expected. The analog-to-digital converter will track this noise as it produces the digital values. Figure 9.9 shows the amount of standing noise that exists from 5 cm distance over a 1 second period of time. There is approximately 100 units of variance out of the 4096 units available from the 12-bit ADC equating to 2.4% error. This value is important to know so that this variance can be accounted for correctly in firmware.

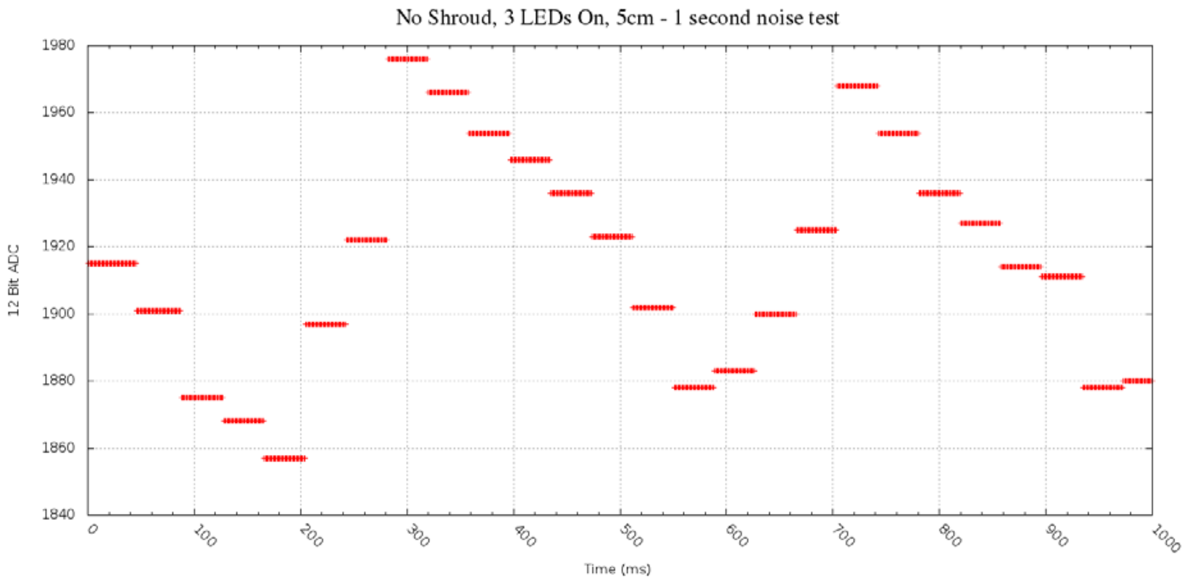


Figure 9.9 No Shroud, 3 LEDs On, 5cm - 1 second noise test

9.2.5 LTE-302/LTR-301 Response Time

Before this circuit can be properly utilized for proximity detection, the response time of the phototransistor needs to be known. In Figure 9.10, it can be seen that it takes approximately 3ms for the phototransistor to transition between its minimum and maximum values. This is in response to a step input, which is the IR LED turning on.

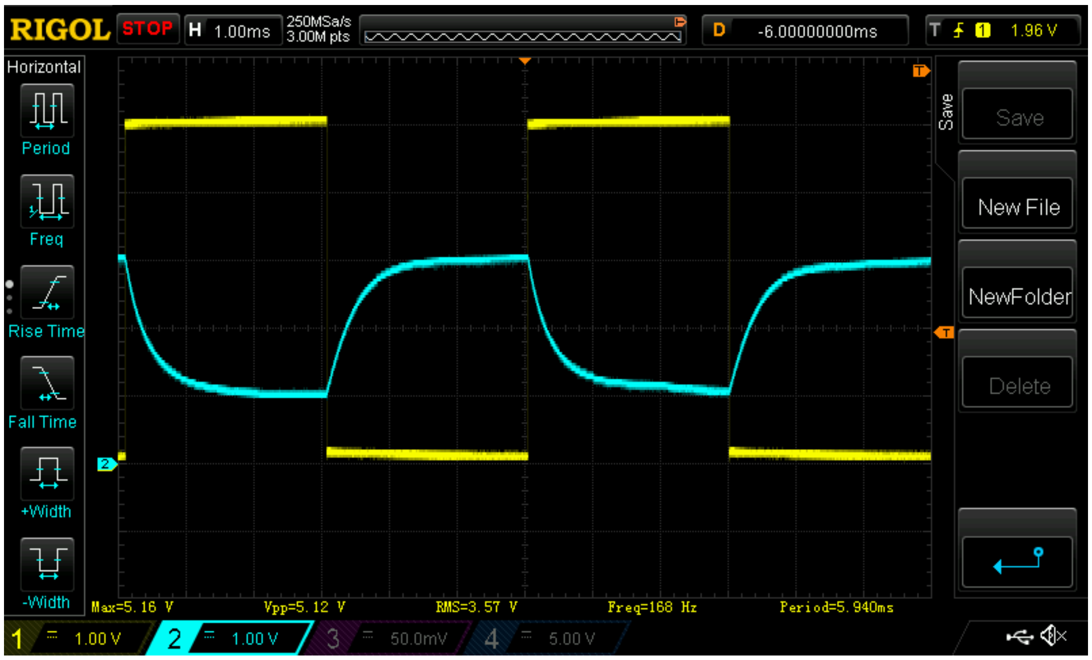


Figure 9.10 IR Phototransistor Response Time

9.3 FSH 7773 Proximity Sensor Test Results

9.3.1 Distance Measurement Data

The same sets of experiments were performed on the FSH 7773 proximity sensor where possible. Since the IC handles the analog-to-digital conversion internally, the data is read from the I2C bus. Figures 9.11 and 9.12 show the left and right sensor daughter card readings as the obstacle moves from 0 to 30 cm away. The range extends to at least 12 cm reliably and resembles an inverse square plot. Figure 9.13 compares the two sensors and verifies that they are indeed identical in performance. This characteristic will prove valuable when applying these measurements to a control algorithm since no calibration will be required.

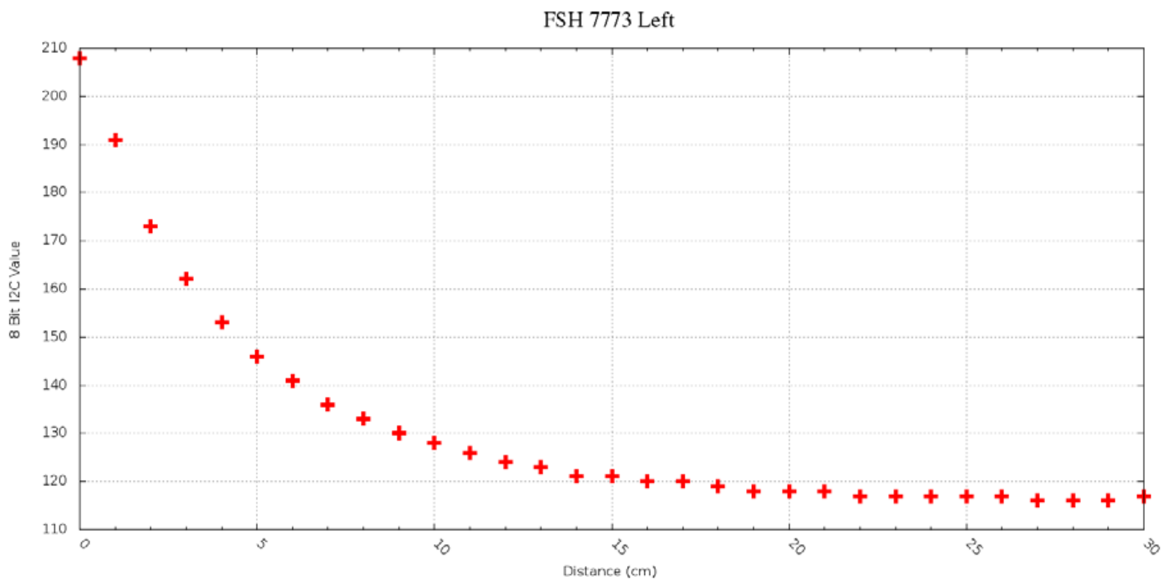


Figure 9.11 FSH 7773 Left

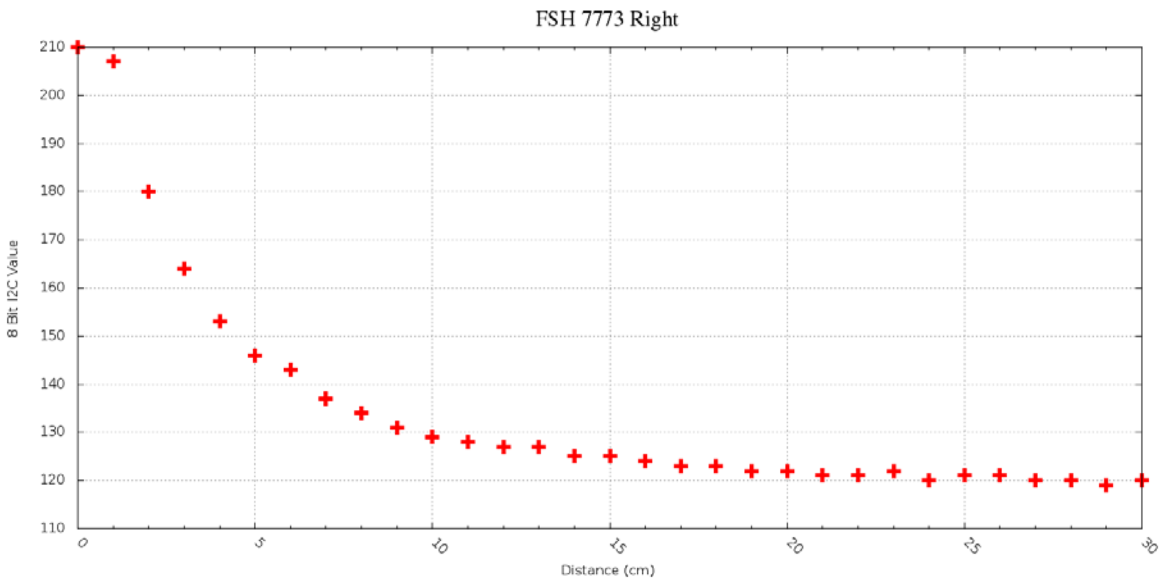


Figure 9.12 FSH 7773 Right

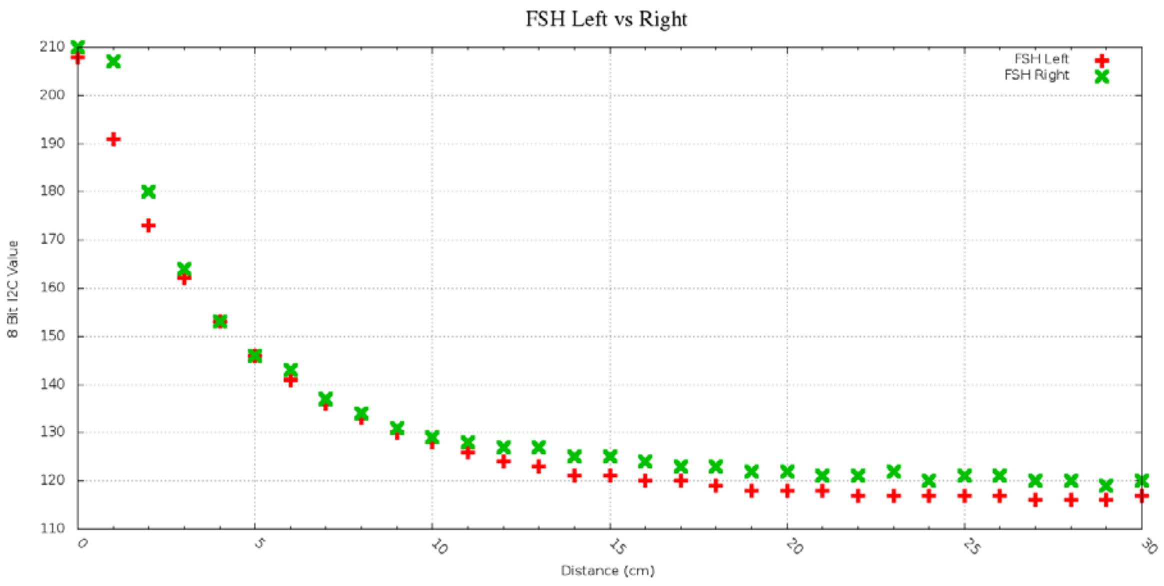


Figure 9.13 FSH Left vs Right

The noise test for the FSH 7773 shows a large amount of improvement over the previously analyzed proximity circuit. While there is only 8 bits of resolution with the FSH 7773, the data is much less noisy. Figure 9.14 shows that there are only 2 units of change out of 256 units of resolution equating to 0.7% error.

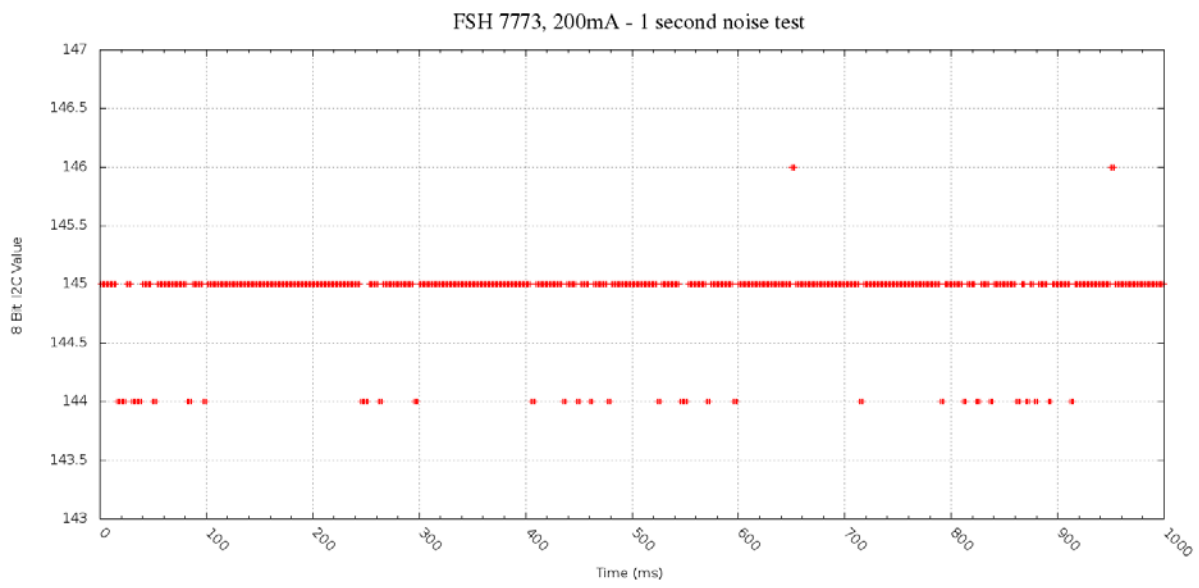


Figure 9.14 FSH 7773, 5cm - 1 second noise test

9.3.2 FSH 7773 Ground Isolation

As mentioned in Section 6.5.2, the FSH 7773 sensors were designed for use in cell phones and consequently required additional design considerations. Figures 9.15 and 9.16 show the shared standing noise when measuring the voltage to the IC's logic and the IC's LED power. As the LED pulses on and off with a 200mA load, the common ground path causes the IC to reset and fail to operate.

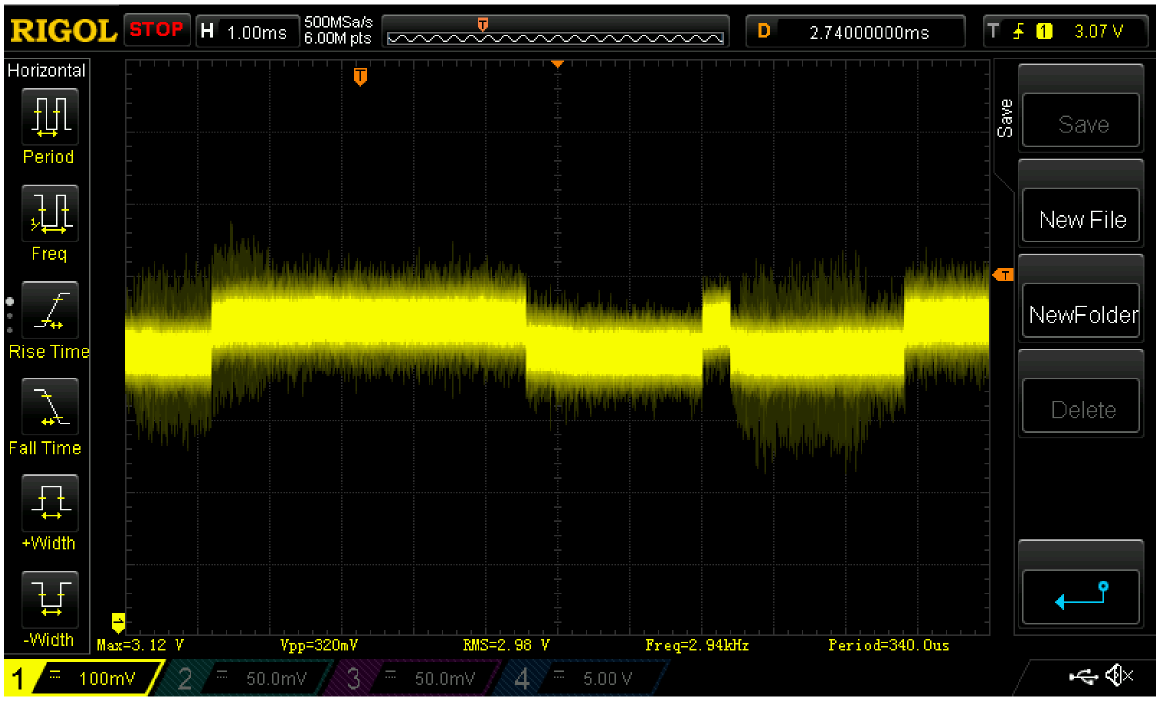

```
RIGOL STOP H 1.00ms 500MSa/s 6.00M pts D 2.74000000ms T 3.07 V
Max=3.12 V   Vpp=320mV   RMS=2.98 V   Freq=2.94kHz   Period=340.0us
1 100mV   2 50.0mV   3 50.0mV   4 5.00 V
```

Figure 9.15 FSH 7773 - VDD using a common ground

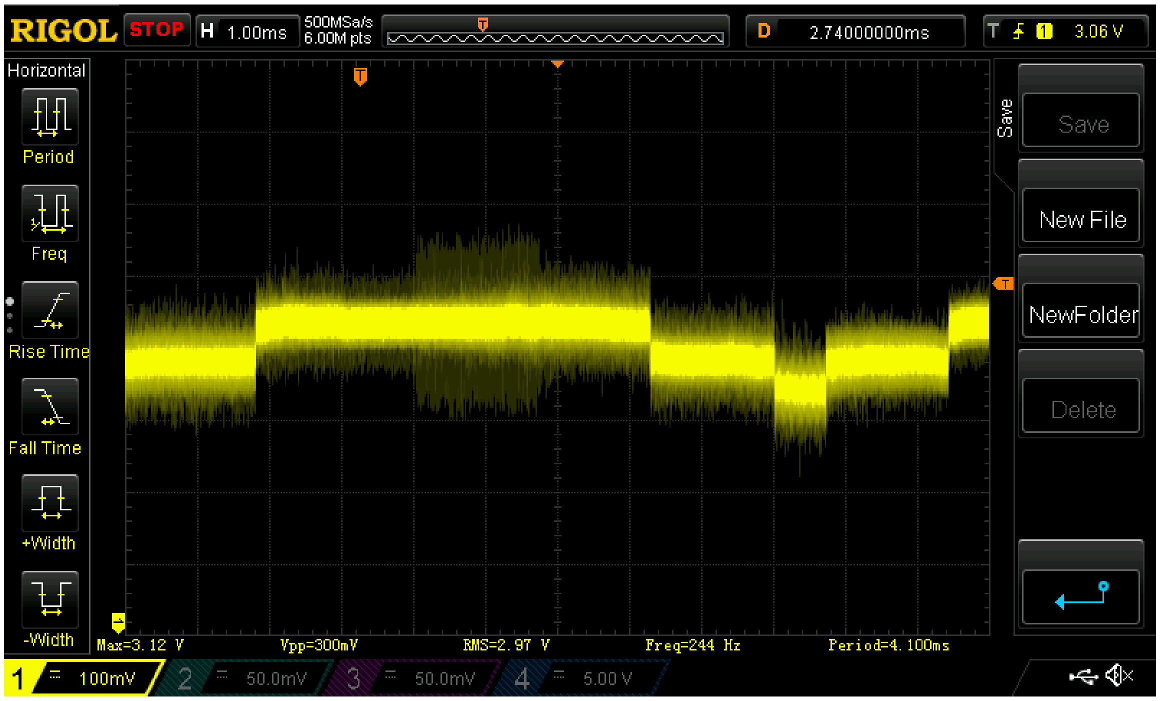
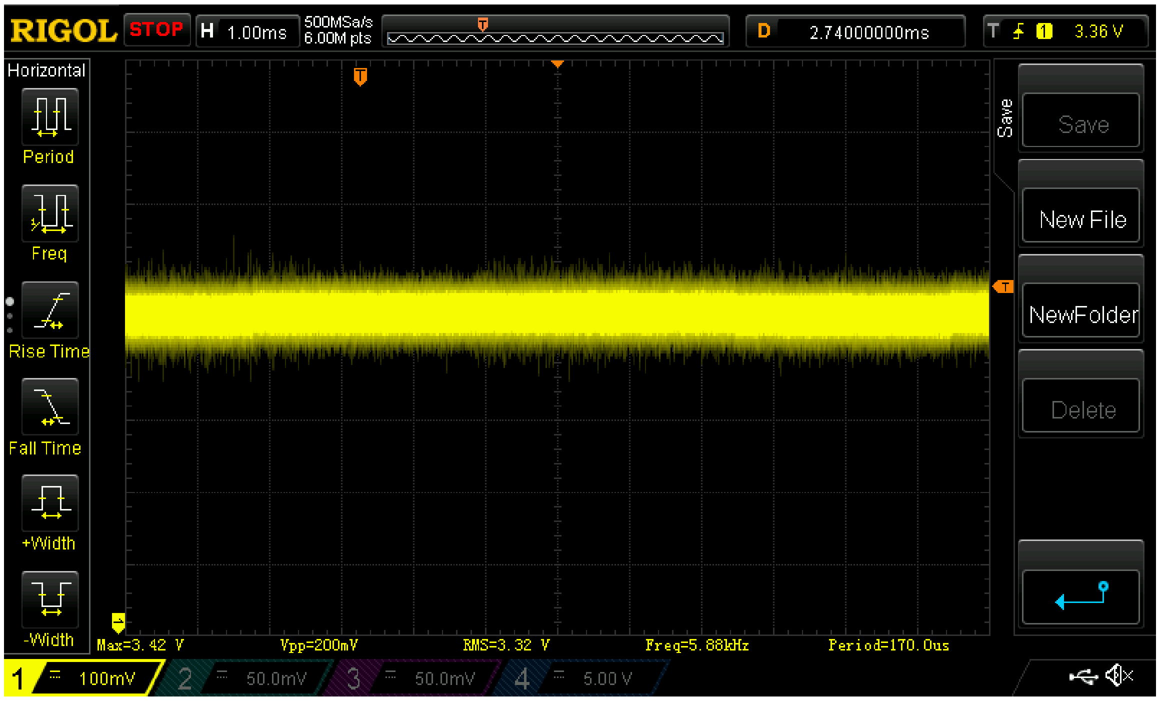

```
RIGOL STOP H 1.00ms 500MSa/s 6.00M pts D 2.74000000ms T 3.06 V
Max=3.12 V   Vpp=300mV   RMS=2.97 V   Freq=244 Hz   Period=4.100ms
1 100mV   2 50.0mV   3 50.0mV   4 5.00 V
```

Figure 9.16 FSH 7773 - V_LED using a common ground

After applying the ground isolation techniques described in Section 6.5.2, the measurements were again taken to verify beneficial changes. Figure 9.17 shows the logic power supply to the IC while using an isolated ground. The abrupt voltage changes found previously have been removed and a stable power supply has been provided to the IC. Figure 9.18 shows the LED power supply as it reacts to the dynamic load response of the regulator. The noise is no longer coupled to the IC's logic power supply as intended by the design.



```
H 1.00ms   500MSa/s 6.00M pts   D 2.74000000ms   T 1 3.36 V
Max=3.42 V   Vpp=200mV   RMS=3.32 V   Freq=5.88kHz   Period=170.0us
1 100mV   2 50.0mV   3 50.0mV   4 5.00 V
```

Figure 9.17 FSH 7773 - VDD using ground isolation

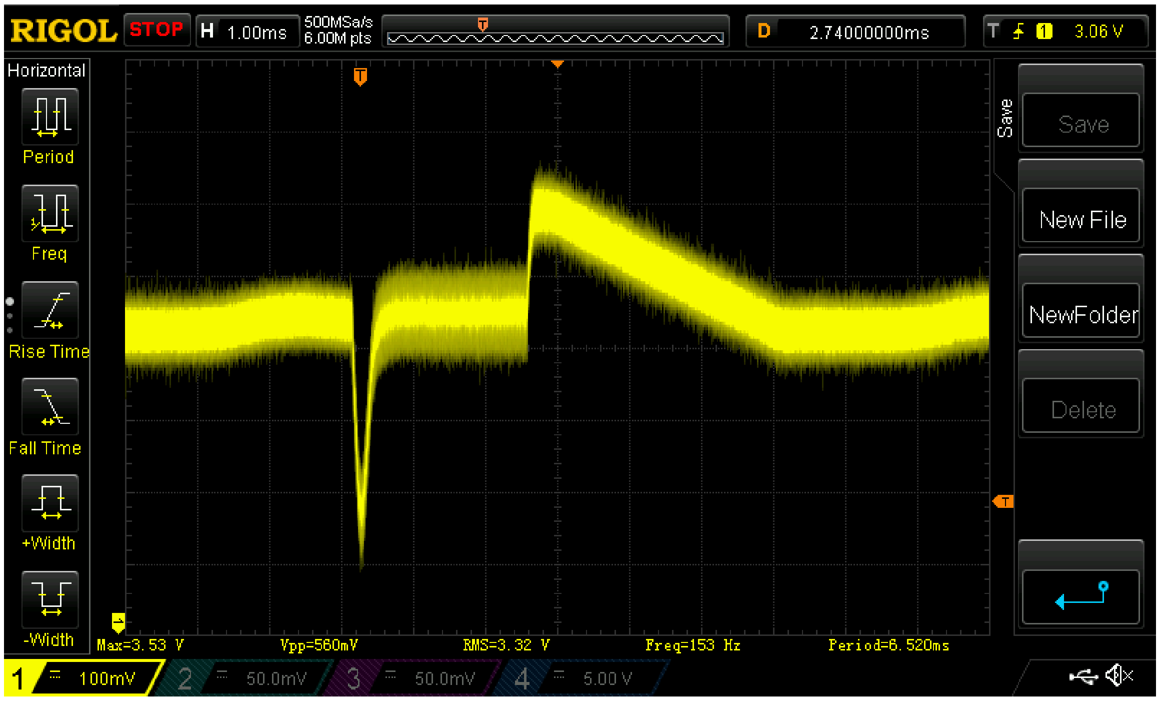


Figure 9.18 FSH 7773 - V_LED using ground isolation

9.4 Proportional Derivative Controller

In order to demonstrate interesting autonomous movement of the robot, a narrow corridor is used as a testing ground. The robot shall traverse the corridor and maintain safe distance from the left and right walls. Once the robot reaches the end of the corridor, it shall rotate 180° and return. If the robot is able to perform this routine several times without becoming unstable, then all parts of the hardware will have been verified including the software API. With this verified, additional collaboration functions can be built.

The proportional derivative controller is a sum of the proportional error and derivative error. The result is the total error from the zero position. The zero position in this case is the center line of the corridor. Figure 9.19 shows the amount of error seen by the robot as it begins navigating 7 cm away from the center line. Each 100 unit of division on the y-axis represents approximately 1 cm of distance. Using this reference, it can be seen that there was approximately 2 cm of overshoot and that stability was reached after 8 seconds.

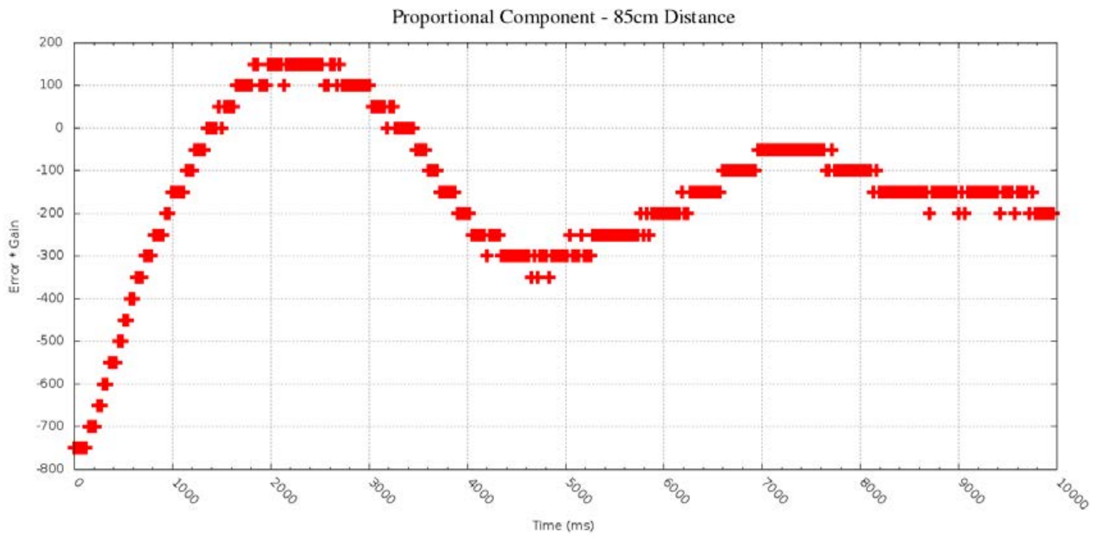


Figure 9.19 PD Error - Proportional Component

The derivative component of a PD controller prevents excessive overshoot when large corrections are required. Without the derivative component, the robot would overcorrect so much that it would actually face one of the walls, get confused, and begin to spin somewhat randomly in an attempt to correct for error. Figure 9.20 shows the derivative component as it accounts for changes in error with respect to time.

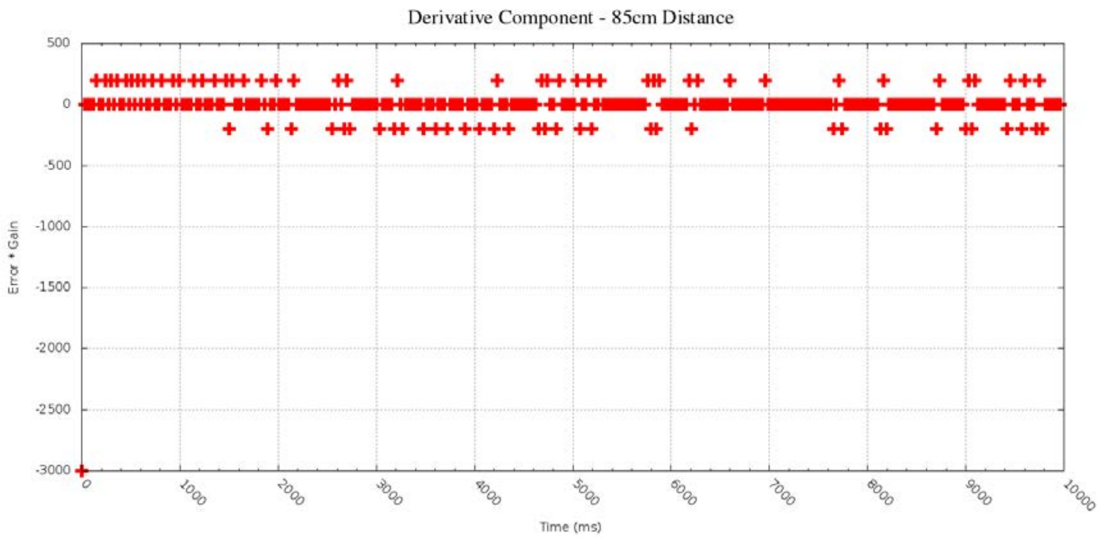


Figure 9.20 PD Error - Derivative Component

Combining both of these components into the total error allows this sum to be used for error correction as the robot navigates the corridor. The pulse width modulation controller uses this error value and proportionally adjusts the speed of each motor (positively or negatively), which results in a stable system. This data is not intended to show success of the controller but instead to show the successful integration of all components.

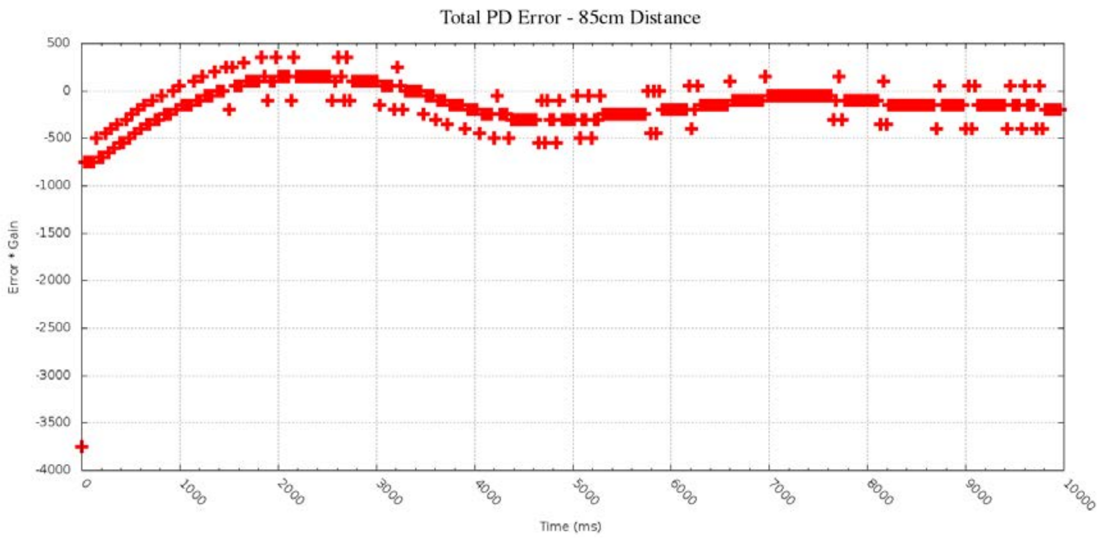


Figure 9.21 PD Error - Total

9.5 Testing Collaboration

With all of the components successfully verified, the notion of collaboration can begin to take shape. The key component to designing robots for collaboration is the ability to build multiple units with minimal time and cost. In this work's example, a minimal number of components were chosen in order to reduce the complexity and build time. A second unit was built with minimal effort and is seen in Figure 9.22.

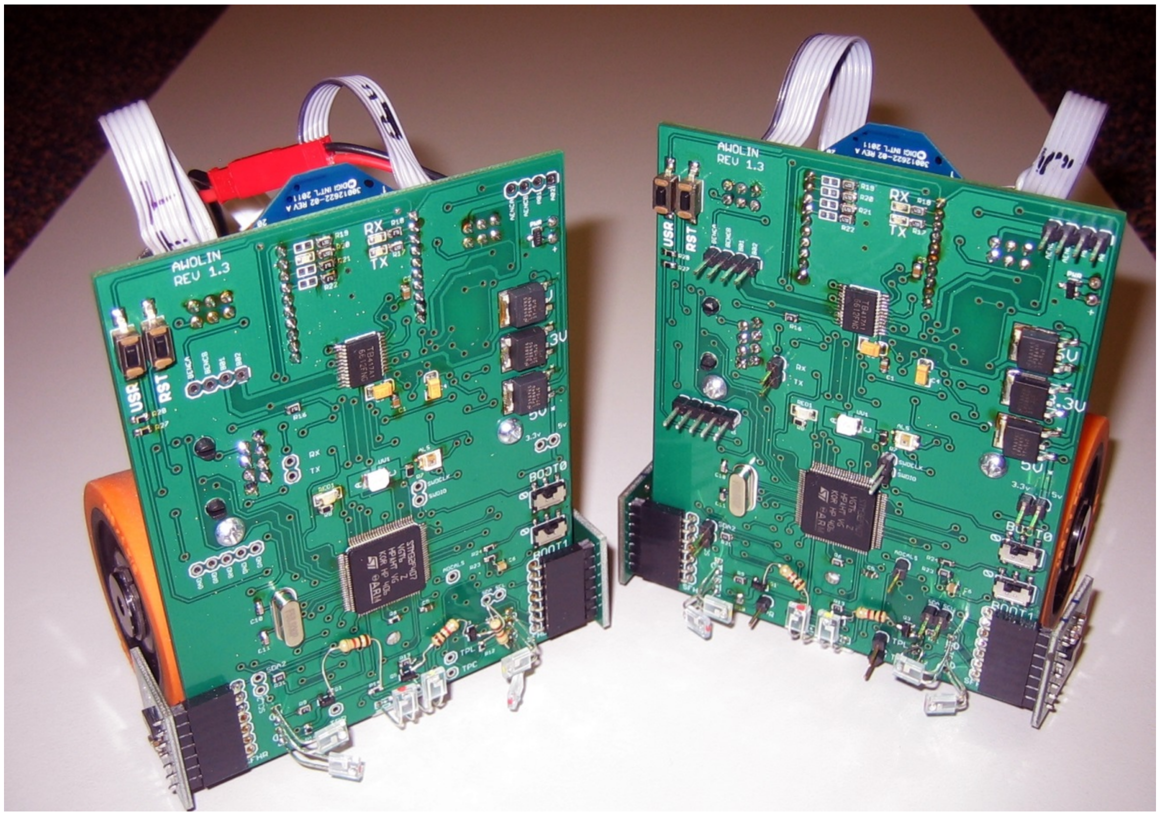


Figure 9.22 Multiple Robots

As mentioned in Section 6.6, an XBEE radio was chosen to communicate with the robots. Robot collaboration can take advantage of both mesh and star networks, which XBEE supports. In order to demonstrate a simple form of collaboration with these systems, a star network was chosen.

With communication now possible between the two robots, a scenario can be constructed that can demonstrate the usefulness of group robotics. A problem that group robotics can solve is the need to locate something of interest in a large area. This could be hazardous waste, a fire, or an injured person in a building. When multiple robots are deployed to find the unit of interest, different actions can be taken once the unit is found. If, for example, a contaminated object was detected, other robots could navigate to that location and assist with cleanup disposal.

For the robots in this work, the following scenario has been constructed. Two robots are inserted into a hallway system at different locations. They are both tasked to find an item of interest, in this case an LED flashlight. Once the light is found by one robot, that robot will send a signal to the other robot indicating that the unit has been found and assistance is required. The other robot will then navigate to that location. This experiment is shown in Figure 9.23.

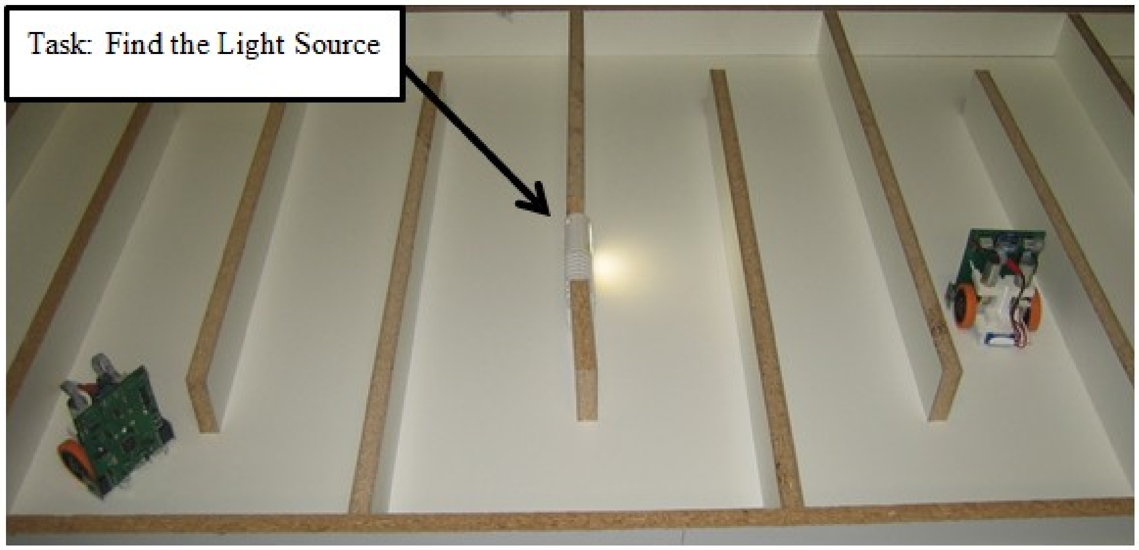


Figure 9.23 Robotic Collaboration Demonstration

9.6 Summary

The hardware verification process described in this chapter provides benefits beyond simple confirmation that the components integrate without conflict. The data collected can now be assembled into a datasheet for the robot, which will become a reference when writing the software. The testing performed for this platform collected the response times and distance capabilities of the proximity sensors. It also revealed that using a shroud around the phototransistor reduced ambient light interference but had a detrimental effect on the maximum range. The ground noise issues discovered earlier in the prototyping process have been solved. In order to ensure a complete and functional design, a routine should be generated that will exercise all components of the system. For this work, a proportional derivative controller was used to navigate the robot back and forth down a corridor without becoming unstable, thus proving a successful design. Successfully building a second robot with minimal effort proves that the design is suitable for collaborative robotics. Multiple systems are easily added to form a group that can be used to demonstrate collaboration techniques and algorithms.

CHAPTER 10: CONCLUSIONS AND FUTURE WORK

Designing and fabricating robots for collaborative behavior is a challenging task that requires persistence in order to overcome the many obstacles that come with embedded and mechanical systems. Breaking the entire task into manageable phases allows the project to reliably grow as each component will have reliance on the others. For the system in this work, all aspects of the initial design goals have been meet and verified on the final assembly. The mechanical design was crucial in order to provide a stable platform for the embedded system, motors, and battery pack to integrate into an autonomous robot. The value of quick turn prototyping systems was made clear as both 3D printing and PCB fabrication services were utilized.

This platform uses the latest software and microprocessor technologies and can provide a suitable stepping stone for researchers of collaborative robotics. The research also serves as a guide to others and seeks to inspire and accelerate the design process. Further revisions of this platform will advance the capabilities even further. The following suggestions are made as avenues where design changes could be made and features could be added.

10.1 Mechanical Changes

The chassis could benefit from an analysis of the skid plates that keep the robot from tilting forward and backwards. The skid pads use Teflon pieces cut to shape as a means of reducing friction while sliding across the labs test table. On a clean lab table, this option is viable and was the primary test location for this research. Other tests are likely to be done on the floor where more complex navigation scenarios can be constructed. The robot by no means was ever designed to go off road, but the smooth floor of a university should certainly be an option. It was discovered that dirt on tiled floors, hardly visible to the human eye, eventually accumulated on the wheels and caused the robot to loose traction. The skid pads would catch somewhat abruptly and at times place too high of a demand on the PD controller, resulting in wall collusions. One solution to this problem could be to remove the pads and simply allow the robot to rock back and forth slightly as it moves forwards and backwards. This would change the angle of the skid plates and reduce the amount of area the plates are in contact with the ground.

10.2 Sensor Changes

The FSH 7773 sensor IC proved to be a highly reliable and robust sensor package that integrates well into robotic systems. The LTE-302 / LTR-301 voltage divider circuit on the other hand proved to disappoint when it came to its range abilities. While it was able to detect enough distance to avoid collusion, there was little room for error. It also had no means to cope with ambient light changes. For these reasons, it is recommended to replace the LTE/LTR components with a forward facing FSH 7773 IC. Since the FSH 7773 does not have configurable addressing, an additional I2C module is required. The microcontroller chosen has a total of three I2C hardware modules and one is currently unused.

10.3 Robot vs Wall Detection

In order to map a room, robots must be able to detect and react differently to the presence of a robot that it would if it detects some other stationary obstacle. An ultraviolet LED has been added to the center of the robot in this work that could be used for this purpose. Next to the ultraviolet LED is an ambient light sensor that was originally intended to detect the presence of ultraviolet light coming from an oncoming robot. The ambient light sensor detects all visible wavelengths of light making it impractical to use as the sensor for robot detection. A better approach is to use an ultraviolet sensor such as the Rohm Semiconductor ML8511. This sensor will ignore the ambient light wavelengths that are not of interest. Since the emitter and sensor are only mounted on the front of the robot, only head on collisions can be detected. A solution to emit UV in a uniform manner around the robot would be an interesting area of research.

REFERENCES

[1] PBS, "TESLA Inside the Lab Remote Control," [Online]. Available: http://www.pbs.org/tesla/ins/lab_remotec.html.

[2] Carnegie Mellon, "Robot Hall of Fame - Unimate," [Online]. Available: http://www.robothalloffame.org/inductees/03inductees/unimate.html.

[3] International Federation of Robotics, "Industrial Robot Statistics," [Online]. Available: http://www.ifr.org/industrial-robots/statistics/.

[4] Campbell, Adam and A. S. Wu, "Multi-agent role allocation: issues, approaches, and multiple perspectives," 2010.

[5] Sahin and Erol, "Swarm Robotics: From Sources of Inspiration to Domains of Application," Middle East Technical University, Ankara, 2005.

[6] Christensen and A. Lyhne, "Morphogenesis: Shaping swarms of intelligent robots," Universite Libre de Bruxelles, July 2007. [Online]. Available: http://videolectures.net/aaai07_christensen_mssir/.

[7] Gokhale and A. A., "Collaborative Learning Enhances Critical Thinking," 1995. [Online]. Available: http://scholar.lib.vt.edu/ejournals/JTE/v7n1/gokhale.jte-v7n1.html?ref=Sawos.Org.

[8] A. Stroupe, T. Huntsberger, A. Okon, H. Aghazarian and M. Robinson, "Behavior-Based Multi-Robot Collaboration for Autonomous Construction Tasks," *Intelligent Robots and Systems*, pp. 1-6, 2005.

[9] M. Tauchi, Y. Sagawa, T. Tanaka and N. Sugie, "Collaboration among a Group of Self-Autonomous Mobile Robots with Diversified Personalities," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendal, Japan, 2004.

[10] X.-j. Wang and C.-y. Shi, "Collaboration planning and conflict resolution among multi autonomous robots," in *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World'*, Munich, 1994.

[11] Dibangoye, Amato, Buffet and Charpillet, "Exploiting Separability in Multiagent Planning with Continuous-State MDPs," in *Proceedings of the Thirteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-14)*, Paris, France, 2014.

[12] S. Zhiguo, W. Junming, L. Xu, W. Zhiliang and T. Jun, "IRGS Protocol Based Mobile Service Robot Positioning and Multi-robot," in *Proceedings of the 30th Chinese Control Conference*, Yantai, China, 2011.

[13] B. Zhang and S. Gao, "The study of ZigBee technology's application in swarm robotics system," in *Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)*, Deng Leng, 2011.

[14] J. Wan, Y. Wang, Q. Qin and Y. Li, "Multi-robots' communication system based on ZigBee network," in *International Conference on Electronic Measurement & Instruments*, Beijing, 2009.

[15] C.-H. Chen and S.-T. Liou, "An Embedded Computing Platform for Robot," in *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, Taichung, 2008.

[16] A.P.Godse. and V.S.Bagad, Mechatronics and Microprocessor, 1st ed., Technical Publications Pune, 2009, pp. 2-58.

[17] TI, "Motor Control - Stepper Motors," [Online]. Available:

http://www.ti.com/lsds/ti/apps/motor/stepper_motors/overview.page.

[18] STMicroelectronics, "STM32F405xx Datasheet," June 2013. [Online]. Available: http://www.st.com/web/en/resource/technical/document/datasheet/DM00037051.pdf.

[19] Hobbico, "LiFe Source Lithium Iron Phosphate Battery Instructions," [Online]. Available: http://manuals.hobbico.com/hca/lifesource-manual-v2.pdf.

[20] THRE3D, "3D Printing Processes," [Online]. Available: https://thre3d.com/how-it-works/3d-printing-process.

[21] 3. Systems, "3D Printing Materials," [Online]. Available: http://www.bitsfrombytes.com/content/3d-printing-materials.

[22] STMicroelectronics, "AN2606 STM32 microcontroller system memory boot mode," June 2011. [Online]. Available: https://my.st.com/public/STe2ecommunities/mcu/Lists/cortex_mx_stm32/Attachments/18225/AN2606.pdf.

[23] STMicroelectronics, "RM0090 STM32F405xx/07xx reference manual," February 2014. [Online]. Available: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/reference_manual/DM00031020.pdf.

[24] Williamson and Tom, "AP-155 Oscillators for Microcontrollers," Intel, 1983.

[25] Phillips, "AN00085 - Start up behavior of the UAA3220TS crystal oscillator," [Online]. Available: http://www.nxp.com/documents/application_note/AN00085.pdf.

[26] STMicroelectronics, "AN2867 Application note Oscillator design guide for ST microcontrollers," [Online]. Available: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/application_note/CD0

0221665.pdf.

[27] Fox, "HC49SLF Crystal Datasheet," [Online]. Available: http://www.foxonline.com/pdfs/hc49slf.pdf.

[28] ARM, "ARM Debug Interface v5 Architecture Specification," 8 February 2006. [Online]. Available: http://www.pjrc.com/arm/pdf/doc/ARM_debug.pdf.

[29] ARM, "RVI and RVT System Design Guidelines," 2010. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.dui0517b/DUI0517B_system_design_reference.pdf.

[30] I. Poole, "EIA RS 232 Standard," [Online]. Available: http://www.radio-electronics.com/info/telecommunications_networks/rs232/eia-rs232-c-d-standards.php.

[31] FCI, "8 POL SHALLOW LATCH CONNECTOR," 31 August 2012. [Online]. Available: http://portal.fciconnect.com/Comergent//fci/drawing/c-bmj-0102.pdf.

[32] Barnes and Malcolm, Practical Variable Speed Drives and Power Electronics, Elsevier Lincare House: IDC Technologies, 2003, pp. 21-23.

[33] Keljik and J. J., Electricity Four: AC/DC Motors, Controls and Maintenance, 9 ed., Ohio Electric Motors, 2009, p. 85.

[34] OSRAM, "SFH 7773 Ambient Light and Proximity Sensor with Integrated IR Emitter," 1 March 2013. [Online]. Available: http://www.osram-os.com/Graphics/XPic0/00082442_0.pdf/SFH%207773,%20Lead%20(Pb)%20Free%20Product%20-%20RoHS%20Compliant.pdf.

[35] OSRAM, "Application Note SFH 7773," 12 December 2011. [Online]. Available: http://www.osram-os.com/Graphics/XPic9/00063497_0.pdf/Proximity%20Sensor%20SFH%207773.pdf.

[36] Microchip, "MCP1702 Datasheet," 2010. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/22008E.pdf.