



Mobile Robotics Platform

An educational robotics system designed for academic use

A Major Qualifying Project
Submitted to the Faculty
of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the
Degree of Bachelor of Science
on

February 25, 2009

by

Keith Chester
keithc@wpi.edu

Matthew P. DeDonato
dedonato@wpi.edu

Jeffrey E. Trost
jetrost@wpi.edu

mobileroboticsplatform@wpi.edu

Approved:

Prof. Fred J. Looft

Prof. Mike R. Ciaraldi

Prof. Eben C. Cobb

Project Number: MXC RBE1/ECC RBE1

Abstract

Students, particularly those that will be studying in upper-tier Unified Robotics courses at Worcester Polytechnic Institute (WPI), are in need of a robotics platform that will provide a foundation facilitating laboratory demonstrations and experiments on theoretical concepts in robotics, as well as act as a basis for robotic projects, allowing a focus on more complicated subjects instead of getting a groundwork system “up and running” first.

Researchers studying on the wide spectrum of topics in robotics also are able to make use of a platform that will

- provide a base to quickly experiment with computational theoretical research, such as artificial intelligence or behavioral and navigational algorithms,
- enable electronic modularity allowing various sensors and critical equipment to interact directly with the robot,
- allow those without the prerequisite knowledge to construct a robotic platform so they may focus their attentions towards their expertise.

The main purpose of this project is to design and prototype a modular robotics base that will fulfill the needs of these intended audiences. This base will be modular, allowing mechanical, electronic, and software modifications while maintaining an ease-of-use approach.

Acknowledgments

We would like to acknowledge and thank Professor Ciaraldi, Professor Cobb, and Professor Looft for advising this project. We would also like to express special thanks to Professor Padir and Professor Miller for their support and advice in overcoming countless problems throughout the course of the project. Special thanks should be given to Professor Beach for taking a personal interest in our success and all the great ideas and feedback he has provided, pro bono.

All pertinent files and documents related to this project are available in digital form from the advising professors.

Contents

1	Introduction	7
1.1	Project Statement	7
1.2	Report Organization	8
1.3	Project Management	9
1.4	Summary	9
2	Background	11
2.1	Introduction	11
2.2	Definitions and Foundational Knowledge	11
2.3	Unified Robotics III and IV Course Outcomes	13
2.4	Robot Bases and Kits Available	13
2.5	Related Projects and Research	16
2.6	Summary	17
3	Problem Statement	19
3.1	Introduction	19
3.2	Problem Statement	19
3.3	Objectives	19
3.4	Requirements	22
3.5	Summary	24
4	Methodology	25
5	System Design	26
5.1	Introduction	26
5.2	Functional Block Diagram	26
5.3	Subsystems	26
5.4	Summary	31
6	Design Details	33
6.1	Introduction	33
6.2	Trade Studies	33
6.3	Construction and Implementation	38
6.4	Summary	38
7	Results	39
7.1	Introduction	39
7.2	Testing and Evaluation	39

7.3	Project Economics/Economic Considerations	40
7.4	Summary	40
8	Summary and Conclusions	41
8.1	Introduction	41
8.2	Completion of Project Objectives	41
8.3	What could be improved upon?	41
8.4	Summary	41
A	Electrical Design Details Reference	47
B	Mechanical and Software Design Details Reference	73
C	Budget Summary	87

List of Figures

5.1	Systems-level functional block diagram of the MRP.	27
5.2	Functional block diagram of the electrical subsystems.	29
5.3	Flowchart of <code>getDI()</code> software routine, querying the digital inputs.	30
5.4	Flowchart of <code>setDO()</code> software routine, setting the digital outputs.	32

List of Tables

2.1	Assumed prerequisite knowledge for Unified Robotics III and IV courses.	18
6.1	Value Analysis of candidate processor systems.	34
6.2	Value Analysis of candidate motor controller solutions.	36
6.3	Value Analysis of candidate optical shaft encoders.	37
8.1	Completion of objectives.	42
8.2	Topics to be improved upon.	44

Authorship

- Introduction KC
 - Project Statement JET
 - Report Organization JET

- Background
 - Introduction KC
 - Def. & Foundational Knowledge JET
 - RBEIII & IV Course Outcomes KC
 - Robot Bases and Kits Available KC
 - Related Projects and Research MPD

- Problem Statement JET
 - Objectives MPD
 - Requirements KC

- Methodology JET

- System Design
 - Functional Block Diagrams KC, MPD, JET
 - Subsystems KC, MPD

- Design Details
 - Trade Studies KC
 - Construction & Implementation MPD

- Results
 - Introduction MPD
 - Testing & Evaluation KC
 - Project Economics JET

- Summary & Conclusions
 - Introduction JET
 - Completion of Project Objectives KC, MPD
 - Summary KC

Chapter 1

Introduction

Worcester Polytechnic Institute (WPI) launched a program earning the degree of bachelor of science for Robotics Engineering in the fall of 2007. A sequence of four courses have been introduced, preceded by an introductory course specifically designed for the Robotics Engineering degree program. The four-course sequence is an introduction to the “foundational theory and practice of robotics engineering from the fields of computer science, electrical engineering and mechanical engineering” [1]. The introductory course and Unified Robotics I and II courses have thus far been conducted using the VEX Robotics Design System, a kit designed by Innovation First, Inc. enabling students to easily build robots [2].

Based upon research to be presented later, it has been determined that there is a need in the market for an affordable yet capable educational robotics platform. In particular, this market niche currently encompasses educational systems that are either

- overpriced,
- bound to proprietary expansions,
- inadequate in their processing capacity,
- convoluted in their operating procedures,
- incompatible with different operating systems,

or a combination of these and other shortcomings. Moreover, it is noted that researchers in fields related to Robotics Engineering also are in need of a cost-effective system.

Determining the characteristics and specifications of a system, particularly one designed for Robotics Engineering students, would logically become clear by itemizing the descriptions of the courses in which the system is to be used. Additional information for topics of future senior design projects and robotic research will be presented later in this document.

1.1 Project Statement

The objective of the project is to design and fabricate a prototype modular robotics base that will be able to meet defined goals in areas such as affordability, capability, open architecture and industry standards. This base will be easily modifiable mechanically, electronically, and within its software. It will be designed to fulfill the needs of students in the Unified Robotics III and IV (RBE300x) courses offered at WPI. It will also be compatible with many readily available components and development tools. The easily-modifiable nature and ease of use will aid researchers

in quickly developing a platform in which to perform their experiments. There will be sufficient documentation for the system and accompanying software, allowing users to easily and quickly become familiar with it and adapt to its use.

Goals and Objectives

In general terms, the goal of the project is to enable students to complete the RBE300x course laboratory exercises. It aids the students in demonstrating the course learning outcomes, facilitating the demonstration of learning goals in the three fields comprising the Robotics Engineering discipline. Allowing students to demonstrate comprehension, analysis, evaluation, application, and finally synthesis of these learning outcomes verifies the project's successful implementation of the stated goals and objectives.

Specifications and Requirements

To determine whether the project meets the intended goals and objectives, certain requirements must be met. In particular, since the end result will be a prototype of the designed system, specifications of the system must be met, demonstrating the achievement of the desired goals. These specifications are separated into disciplines within three categories; electrical and computer engineering, mechanical engineering, and computer science.

It is best to use a scenario providing a functional description of a system's typical operating condition, otherwise referred to as a "use case," often used in software engineering [3]. A use case for the system demonstrating some characteristic specifications is as follows.

The modular robotics platform is equipped with a four-wheel powered drive train. The system approaches a curb with a height of 10 centimeters. The system is able to negotiate the obstacle and continue on its course past the barrier.

This use case presents a situation that the system deals with. In use cases, there are specifications that are implied by the scenario. These are typically required in order to complete the scenario. Other specifications, those that are not explicitly required by the scenario, are not requirements of the system. For example, in the previous scenario, there would be a certain size requirement that the platform's wheels must meet in order to adequately traverse the curb. This minimum size for the wheel is a requirement. However, strictly abiding to this use case, it is obvious that the color of the wheel is not a requirement for the system.

Referring to the material presented in section 2.3, the required specifications for the system begin to evolve. Using the proposed laboratory exercises and expected performance as a group of concurrent use cases, a compilation of specifications and requirements can be extrapolated.

There are certain requirements that the modular robotics platform must meet in order to satisfy the use cases presented. By satisfying these requirements implied by the use cases, it is ensured that the goals and objectives of the project are also satisfied simultaneously. A complete listing of requirements and specifications is presented in section 3.4.

1.2 Report Organization

The organization of the report is as follows. Chapters 2 through 4 are concerned with the planning and preparations for designing a Mobile Robotics Platform (MRP). In Chapter 2, background information is presented on the problem being solved, including mention of the stakeholders in

the project. Previous senior design projects and relevant external study is discussed, and market research is provided to demonstrate the problem statement. Chapter 3.2 explores in more depth the problem statement of the project, and the initial objectives to be considered as the project progresses. Also presented in Chapter 3.2 are the quantifiable requirements which will be used in determining the success at the completion of the project. Chapter 4 expresses the methodology of how the final prototype was designed to fulfill the requirements presented in Section 3.4.

Chapters 5 through 7, describe the design of the MRP system through evaluation and testing. The final system design is detailed in Chapter 6, with elaboration of the subsystems comprising the framework of the prototype. Chapter 7 illustrates the results of the system design, including the construction, implementation, and testing of the produced prototype. The economics of the project are presented in Section 7.3, detailing the project's total budget and an analysis of the total cost of the completed system for fiscal considerations in using the results of the project in Robotics Engineering (or other) courses.

Finally, conclusions are drawn and a summary of the project is given in Chapter 8, including possible improvements and future work in education-oriented modular robotics platforms.

1.3 Project Management

The Mobile Robotics Platform project team consisted of three students, each of whom worked on very diverse and overlapping facets of the project. With the vast majority of the work being done on computers and in digital form, there is always the concern of large amounts of data being lost with if a computer crashes or a storage drive fails. To ameliorate these concerns, there were checks in place that ensured redundant copies of critical files were made to prevent against complete loss of work (and time) with a single software or hardware malfunction. In addition to having redundant copies on local hard drives, a network storage space was provided by the Electrical and Computer Engineering Department on their secure (and equally fail-safe) server, `farad.ece.wpi.edu`. This also provided a common central location to allow better productivity and easier locating of digital resources.

Another valuable resources utilized in the development of the project was Microsoft Office OneNote. This software allowed easy collaboration and organization of data during the gathering of information for background research. It also allowed simultaneous editing of documents with dynamic updates. By using OneNote, the project team benefitted from a more organized approach to research.

The additional time taken to familiarize the team with using Microsoft Office OneNote and instilling the practice of backing-up documents and files was insignificant compared to the time saved on multiple occasions where lost data was recovered or document creation came off with time-saving synergy.

1.4 Summary

This paper is a presentation of the Mobile Robotics Platform and the efforts of the project team. The project was founded as a response to poor selection in the current market for a college level educational platform. The requirements for the platform shall be determined from analyzing the course descriptions for the Unified Robotics courses RBE2001 and RBE2002. While determining the requirements, use cases shall be used to determine requirements from probable scenarios of student use.

The report shall be organized to cover the engineering process behind the platforms design to its construction and testing. First, in Chapters 2 through 3, requirements shall be derived. Chapters 3 and 4 shall cover the construction of the MRP and its methods behind its design. Chapters 5 through 7 shall cover the design of each element of the MRP, its testing, and the end result of the teams efforts.

The project was worked on by three students, whom worked on overlapping parts of the overall project. The MRP was designed, built, tested, and maintained in WPI labs. All information was stored on the Electrical and Computer Engineering Departments network server. Microsoft OneNote was used for team organization and compilation of research information.

Chapter 2

Background

2.1 Introduction

The target audience for this project is defined as university-level students studying Robotics Engineering, specifically students participating in the RBE300x courses at WPI. Researchers in related fields are considered to be the extended audience. Constructing a framework of requirements is essential in keeping the project faithful to the problem statement. Since the result of the project is intended to be used in Robotics Engineering courses, logic dictates that a thorough review of these courses is needed in order to become aware of the experience and skill set exhibited by the target audience.

Researching what is publicly available for purchase will confirm that there is no appropriate robotic base that fits these needs.

2.2 Definitions and Foundational Knowledge

In the context of this project, a robot is considered an autonomous vehicle with means of sensing its environment. It is not to be confused with industrial robotic arms.

A researcher that would require and use this robotic platform is assumed to be college educated or better and have a basic understanding of programming methods and languages, static systems and basic mechanics, as well as the fundamentals of electronic circuits and computer architecture.

A Robotics Engineering student that is considered the target audience for the project is assumed to be a student with the following experience. Example WPI course numbers are included in parenthesis [1].

- Mechanical Engineering
 - knowledge of static systems (ES 2501)
- Electrical Engineering
 - ability to design basic analog circuits combined with an understanding of circuit diagrams (ECE 2011)
 - familiarity with designing digital circuitry, as well as Boolean logic (ECE 2022)
 - capacity to read a sensor datasheet and understand the principles of its operation
 - experience designing and configuring an embedded system (ECE 2801)

- Computer Science
 - programming experience, particularly in a systems programming language such as C (CS 1101, CS 1102, CS 2102, CS 2301)
- Robotics Engineering
 - course experience in Unified Robotics, including Introduction to Robotics and Unified Robotics I and II (RBE 1001, RBE 2001, RBE 2002)

Taken from the WPI course catalog and experience with the courses, the Unified Robotics curricula consists of the following.

Introduction to Robotics serves as a general multidisciplinary introduction to robotics. Students are taught how to design basic analog circuitry. Basic analog circuitry covered consists of

- resistors,
- capacitors,
- operational amplifiers, and
- voltage dividers.

Students are also exposed to mechanic forces and moments in static systems. Students gain experience in programming with C in an embedded system through both class lectures and laboratory and project work.

Unified Robotics I, the first of a four course series, focuses on the mechanical and electrical aspects of robotic systems. Mechanical engineering concepts covered in this course include calculating force, moments, and inertia, as well as an introduction to friction in static systems. Students learn the principles of various electrical systems including power systems and signal analysis; voltage regulation and power supplies, selecting application-appropriate batteries, design and implementation of H-bridges, and the design and analysis of Pulse Width Modulated (PWM) signals. In regards to computer science, different types of variables and memory storage techniques are proposed for writing software in the C programming language.

Unified Robotics II expanded upon the fields taught in the prior course and introduced students to the basics of control engineering. This aspect of the course consisted of types of control loops with examples of their application, and application of feedback loops. The course's electrical portion had students study the physical and electronic principles behind the functioning of sensors, along with a brief overview of continuous and discrete signal analysis. Computer science topics expanded in the class included pointers and arrays in embedded C, memory management with proper use, as well as an introduction to real-time and structured programming.

Unified Robotics III and IV will not be offered until the spring semester of the 2008-09 academic year. The expected curricula from these courses are speculated from their course descriptions.

Unified Robotics III's computer science portion of the course will be more advanced. Topics will include complex response and feedback processes implemented in software in an embedded system, concepts of real time operating systems, and interrupt signaling and re-entrant coding. Topics in mechanical and electrical engineering will cover actuator design and interface methods. Such scheme will cover magnetic, pneumatic, piezoelectric, and linear actuators.

In **Unified Robotics IV**, students will study computer science concepts such as navigation, position estimation, vision processing, and communications and network protocol. Students will also be exposed to wireless networks and radio communication. Operation and design of robots in hazardous environments shall also be covered.

2.3 Unified Robotics III and IV Course Outcomes

An interesting challenge is presented to the engineers of this project. Traditionally, it would be appropriate to analyze the content of a course after the curriculum has been formulated and decided upon. However, as a result of the infancy of the Robotics Engineering program at the time of this document's production, this luxury cannot be afforded. The only documentation to base an analysis of the Unified Robotics III and IV courses upon is one brief descriptive paragraph apiece, offered by the WPI Course Catalog.

Assumed Prerequisite Knowledge

Based upon the assumption that students prepared to take the Unified Robotics III and IV courses have taken the preceding Unified Robotics courses, familiarity with certain concepts and techniques are presumed. Table 2.1 provides a sampling of the multidisciplinary prerequisites the RBE300x courses build upon. Given this foundational base of knowledge, certain liberties can be afforded that take advantage of these concepts. It also defines a range of applications that would be expected of students in the courses' laboratory exercises.

Inferring Course Objectives

Based upon these course descriptions and preliminary material provided by Professor Fred J. Looft, a brainstorming meeting was held to determine some general course objectives for these two courses. Beginning with an outline of the course descriptions, general topics in robotics theory began to materialize. Following this, it became clear that in order to determine what was needed in an appropriate base platform to be used for these courses, several laboratory exercises would have to be proposed.

Furthermore, learning objectives for the courses should be compared to the labs to ensure that the laboratory exercises successfully fulfill the expected learning outcomes for the courses. These learning outcomes, although still preliminary, are the guides that will dictate the design of the project. It is best to focus the formulation of laboratory exercises around the Unified Robotics IV course since this course is the culmination of the Unified Robotics sequence. The brainstorming activity resulted in an outline of proposed course material to be used in the RBE300x curriculum.

2.4 Robot Bases and Kits Available

Several robot kits and bases are available for purchase. These kits provide intriguing ideas for consideration in new robot designs but also exhibit limitations for the application of being used in a college level robotics course.

The cost of each kit is determined by either the price listed by the manufacturer or the most commonly found retail price. The rating of "Expandable" is determined by the ability of the kit or base to be either reformed into different robots or by the kit's or base's ability to be built upon.

All kits and bases listed in the table are programmable. The score of "High Level Programming" is determined by several factors:

- Is there any programming support?
- Do the software development tools allow complete and unrestricted programming of the robot?

- Can the kit or base be programmed in high-level languages, such as object oriented languages?

Kits that did not receive a check mark for programmability usually did not have object oriented language support, though a few also lacked proper software development tools. Those that did not support object oriented languages tended to support C.

For the “Outdoor Operation” category considered whether a unit can operate outdoors in an urban environment. The ability to handle slightly rocky terrain and a curb were the consideration when assigning a check mark. Units that did not receive a mark were usually too small to be considered for use in an urban environment.

While there were ratable similarities between each kit, each was unique enough to warrant individual explanations. A short summary of each kit within the above table follows.

The **VEX Robotics Design System**, produced by Innovation First Inc., has been used by WPI for the first three robotics courses within the Robotics Engineering program—Introduction to Robotics and Unified Robotics I and II. The processor in the VEX system can be programmed in either EasyC, a graphical C based programming environment produced by Innovation First, Inc., or in C using a standard development environment with the proper proprietary compiler and software library. This library, WPILIB, was created at WPI. It has been used by the robotics classes at WPI thus far [4]. VEX’s mechanical kit allows for many possible configurations of your robot and for students to construct many different mechanical systems [5].

The **iRobot Create** is based upon their popular Roomba robotic vacuum cleaner. The Roomba had a dedicated hobbyist following to program and “hack” the Roomba by adding capabilities and turning it into a robotics base. iRobot decided to release the Create, a vacuum-free platform that allows one to easily add electronics and completely reprogram the base. The Create is only reprogrammable and accepts other electronics with the “command module” attachment [6]. The Create, with the command module, has open ports on the top of the robot for easy wiring of electronics. It is programmable in both C and C++ and by Microsoft Robotics Studio [7]. The base is not easy to mechanically modify, however. While additional parts can be attached to the top of the base, the user is limited to how much they can change the base.

Evolution Robotics’ ER1 is a robot base with no controller. It is a metal laptop stand with attached wheels, motors and motor controllers, and webcam. A laptop is attached to the base to act as the robot’s controller. The ER1 can be programmed with the use of proprietary software from Evolution Robotics. The only I/O or sensor input that the ER1 has is a webcam. Evolution Robotics’ software takes care of the complicated aspects of vision processing, allowing users to merely program the robot’s behavior and responses [8].

The **Robotech RDS-X01 Robodesigner** is a robotics kit geared towards middle school and early high school students [9]. There is a very limited range of sensors and motors. This kit can be programmed with the proprietary graphical user interface that is included or in C. The website for the RDS claims the intended audience is high school and university students. The plastic parts and limited sensor options restricts the usefulness and expandability of the kit.

LEGO’s popular **Mindstorms** kit (the NXT) has been used extensively in a large number of educational applications. The kit is used in the FIRST LEGO League competition, in which high school students across the nation compete with LEGO robots [10]. The kit consists of a single box that has all necessary I/O and motor ports on it, and connectors for programming. The controller for LEGO Mindstorms can support up to three motors and an additional four sensors, and has Bluetooth and USB ports. LEGO’s large library of compatible bricks for building allows for heavy modification of the robot. Many LEGO accessories allow outdoor robots to be built with Mindstorms [11]. While the Mindstorm can be programmed with LEGO’s graphical language, there are

numerous development tools that allow the Mindstorms to also be programmed with C, C++, and Java [10].

Surveyor makes the **SRV-1 Blackfin** robot. The robot is very small, only 5 inches by 4 inches by 4.5 inches. Despite its dimensions, the SRV-1 has many powerful onboard electronics. Its processor, attached to its camera, is the 500 MHz Blackfin from Analog Devices [12]. This camera–processor combination can actually be separated into two different components. The camera is 1.3 megapixels made by Omnivision and can capture directly to SDRAM at four different resolutions. The processor can handle basic image processing, including histograms and blob detection. The robot has a WiFi module and a dual motor H-bridge that can source up to 1 ampere of current. It also has a laser rangefinder onboard [13]. The Surveyor SRV-1 was designed to aid the development and research of SWARM robotics [12]. It cannot be extended upon mechanically and is too small to deal with even basic environmental obstacles.

Acroname's Garcia robot is modifiable and made upon order. The simplest version—a chassis with a serial port, battery and basic electronics, and infrared remote control—sells for \$1500. Garcia robots through customization can reach prices up to \$6000 with the following options [14].

- Hokuyo Laser Scanner
- USB instead of a serial port
- Stargate 400MHz processor with optional WiFi support
- Camera boom
- Text-to-speech module

The Garcia robot base can be programmed in Java, C, and C++. The product page for the robot claims that the primary focus of the Garcia robot is for enthusiasts, researchers, and students to explore robot behavior. They recommend an advanced knowledge of programming and communications in order to work with the Garcia 20 Acroname 2008. The Garcia is not mechanically or electrically modifiable. It does not allow additional components to be added onto the base, nor does it allow modifying its hardware.

The **Seekur**, by **MobileRobots**, allows for high-level programming and is capable of outdoor operation. Its drive system is holonomic, meaning that it can drive in any direction and rotate independently, with a built-in suspension to absorb inconsistencies in outdoor terrain. MobileRobots' outdoor guidance system (mOGS) includes GPS, accelerometers, laser range finders, and mOGS software. There are several additional options available.

- Multiple degrees of freedom arms
- Vision systems
- Sonar units
- Gyroscopes
- Additional embedded PCs
- Wireless Ethernet
- Thermal sensing

Seekur is programmed in C or C++ created with or without ARIA, one of their many proprietary development and robot operations programs designed to work with their available sensors and robots [15].

Segway has four models for their Robotic Mobility Platform (RMP) line. The smallest and least expensive of the four, the **RMP 50**, is labeled as ideal for teaching laboratories. The circular base has two large powered wheels and a single smaller caster wheel to balance the robot. RMPs come without any sensors but do come with a programmable controller. The controller can be programmed in C or C++. The base is designed for indoor use, but can travel long distances (approximately five to six miles) and can handle simple urban obstacles. The base is priced at \$7000. The base supports construction and additions on top of it, but does not allow the base's drive train to be modified at all [16].

2.5 Related Projects and Research

Projects

Previous Senior Design Projects at WPI have focused on robots designed only for a specific purpose with commercially available parts. Examples of applications in prior Senior Design Projects focused on are as follows.

- Autonomous robot security guard
- Inspection of gas pipes [17]
- Robotic test bed for exploring capabilities to be incorporated in future Mars rovers
- Walk on six legs [18]
- Autonomously measure temperature, distance, and acceleration to participate in the Trinity College Home Robot Fire Fighting Contest [19]

All past robotics projects and research at WPI has been towards a specific (as opposed to general) application. There has not been a project with the intent of developing a modular robotics base. This can be attributed to the relative infancy of the Robotics Engineering degree at WPI. The similarities that exist between the aforementioned projects and this project will provide valuable information on how to proceed.

Research

Carnegie Mellon University and The Instituto Tecnológico y de Estudios Superiores de Monterrey have either developed a robot base for use in a course or studied what was previously available on the market.

Carnegie Mellon University produced Trikebot for a summer high school robotics course. The robot used common and easily replaceable components for its base. The course was targeted at high school students entering their senior year with no previous robotics experience and would teach the students to program robot behavior and algorithms. The Trikebot design was limited because each student was allowed to take their robot home, forcing a lower overall budget. The cost of each robot was kept below \$1200 [20].

The Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM) teaches a senior level robot design course wherein the students make proposals for robots that solve specific problems.

Students then design and construct these robots. A test-board robot was created to aid in the development of students' robots. The platform was designed so that it would be affordable for medium and small educational and research centers. This resulted in the capabilities and expandability of the platform being limited [21].

2.6 Summary

As can be seen by the presented background information, there is much evidence owing to the need for an appropriate platform for the Unified Robotics courses, one that cannot be satisfied by the currently available platforms on the market. This evidence is provided through the descriptions of the Unified Robotics courses in which the platform will be used, as well as past work related to this project's goal. This past work includes both outside research done by independent universities, as well as internal work conducted as the topics of past Major Qualifying Projects here at WPI. Hence, the basis for the project is established and a formal statement of the project's purpose shall follow.

Table 2.1: Assumed prerequisite knowledge for Unified Robotics III and IV courses.

Prerequisite Course Number	Course Title	Concepts Gained
ES 3011	Control Engineering	Mathematical representation of control components and systems. Laplace transforms, transfer functions, block and signal flow diagrams. Transient response analysis.
CS 2223	Algorithms	Data structures, data abstraction techniques, algorithm design and analysis, worst case and average case. Greedy algorithms, divide-and-conquer, dynamic programming, heuristics, and probabilistic algorithms. Introduction to sorting, graph theory, and string processing.
ECE 2801	Embedded Computer Systems	Fundamentals of computer architecture and organization. How hardware, software, and the passage of time must be managed in an embedded system design. Logic flow, real-time programming, maintainability and software maintenance cycles.
MA 2051	Ordinary Differential Equations	Techniques for solving ordinary differential equations. Modeling using first-order differential equations, solution methods for linear higher-order equations, qualitative behavior of nonlinear first-order equations. Oscillatory phenomena including spring-mass systems and RLC-circuits and Laplace transform.
RBE 2002	Unified Robotics II	Interaction with the environment via sensors. Concepts of stress and strain as related to sensing of force. Principles of operation and interface methods for electronic transducers of strain, light, proximity and angle. Feedback mechanisms for mechanical systems via electronic circuits and software mechanisms. Modular design and implementation of decision algorithms and finite state machines.

Chapter 3

Problem Statement

3.1 Introduction

The purpose of creating a problem statement is to be able to have a clearly defined goal to which the project can be directed. When the requirements of the project are clearly defined, an objective approach to measuring the success of the project is more easily attained.

3.2 Problem Statement

The purpose of this project is to design and prototype a modular robotics platform for educational use in the Unified Robotics course curriculum at WPI.

3.3 Objectives

To expand upon the original problem statement, it should be clarified that the design considerations focus primarily on modularity, expandability, universality of use, and low cost. When describing the final prototype, these four descriptors should be easily demonstrated. The four concepts were prominent throughout the design process and are worth further defining.

Modularity

In regards to the delivered system being modular, the intent is that the overall prototype is comprised of smaller subsystems, each of which can be thought of as a module. In addition, these modules can be easily applied to the final product and should readily enhance functionality, since modularity of the system was an objective for the final design of the system.

Consider the drive train of the final design; a “modular” mobile robotics platform would have a standard drive train system that is self-contained in that the drive train can be used in the overall system even if the frame or chassis of the MRP is altered into a new configuration. This implies that these modules are not dependent on the structure or design of the rest of the system, and are able to perform their intended task if the rest of the system changes configuration.

Expandability

It is recognized that the student team involved in the inception and birth of the MRP will not be able to design and prototype for all intended applications for the MRP. Also, to prevent the MRP

from becoming outdated in a year or two because of a simple change in curriculum, the MRP should be expandable and should be able to be used in a large range of applications as a result of this expandability.

For instance, consider the laboratory exercises required of the Unified Robotics courses. A sample laboratory scenario might have students use the standard base MRP (without significant additional features) to complete certain tasks. At the conclusion of the course, the curriculum might warrant a change in some of the laboratory exercises to incorporate different aspects of the course learning objectives. An expandable and universal platform should be able to accommodate these future curriculum changes. This is another objective of the system as a whole.

Low Cost

As was identified in the Background, many robotics systems have a tendency to be outside the fiscal budget of most educational institutions. Also, since this project's principal stakeholder is an educational institution, it is logical to solve the problem of learning tools being prohibitively expensive with the solution to the overall objective as well. Therefore, the MRP should be designed with the mentality of keeping the end result "low cost."

Since the term "low cost" is relative, the administrators of the Robotics Engineering curriculum were consulted. Based upon the cost of the systems currently in use for the Unified Robotics I and II courses, it was determined that a range between \$1000 and \$2000 was an appropriate definition. Thus, a further objective of the project is to complete the design of the MRP with a final prototype cost in this range.

Field-of-Study Specific Objectives

The field of Robotics Engineering is an amalgamation of multiple engineering disciplines generally aimed at providing solutions for application-specific problems. The three disciplines constituting the bulk of the Robotics Engineering field are Mechanical Engineering, Computer Science, and Electrical and Computer Engineering. Hence, the objectives presented in this section are separated into these three subsections.

Mechanical Engineering

Objectives with regard to Mechanical Engineering are few; however they are universal in their application. Considerations for providing adequate support in the implementation of actuator design are expected. Different mechanical interfacing methods should be afforded forethought. Some of these include magnetic, pneumatic, piezoelectric, and linear actuators [1]. Looking into the future, operation in hazardous environments is a growing application for robotics, and design of platforms for these conditions is a possible course topic.

Computer Science

As more complex systems are miniaturized and implemented into embedded systems, robots will gain proportionally more complex tasks. These tasks can range from simple navigation and position estimation to processor-intensive vision processing and wireless communication using security network protocols. Scaling down to embedded systems and applications, real-time operating systems are becoming expected of robotic systems, which go hand-in-hand with interrupt signaling and re-entrant coding. Finally, applying control theory to robotic platforms dictates the implementation of complex response and feedback processes in embedded systems.

From the perspective of the end-user, it would be considered presumptuous to choose a programming language that is inappropriate for robotic applications. Because of the subjectivity of this issue, an ideal system would not be bound to a single programming language. Similarly, as embedded systems become increasingly capable of running operating systems as opposed to merely programs loaded into memory, more avenues for expansion begin to be explored. So too, open source and non-traditional operating systems are quickly becoming more common, especially with the academic and research community. All of these things suggest a need for a system capable of being developed independent of a host operating system.

Electrical and Computer Engineering

When considering a robotic system, there are many aspects encompassed in the field of Electrical and Computer Engineering that must be recognized. Thus, a distinction must be made between analog and digital concentrations, as they relate to the field of Electrical and Computer Engineering. Specifically, "electrical engineering" corresponds to the analog aspects of the field, i.e. circuit analysis focusing on current and voltage relationships. The complementary term "computer engineering" refers to the world of digital logic circuits using logic gates, i.e. NOT, AND, OR, NAND, XOR, as well as designing logic networks via truth tables, optimizing for speed, etc. Because of the scope of these two fields, it is important to make a delineation when specifying objectives in Electrical and Computer Engineering.

Electrical Engineering (Analog) As the field of robotics evolves, many things may change from how they are presently. However, one thing that will remain constant into the near future is the need for electrical power in robotic applications. Students learning to analyze and demonstrate concepts of analog circuitry and apply them to robotic applications, a few basic "lab bench" needs become apparent.

The tools every electrical engineering student becomes intimately familiar with are the equipment ever-present on the laboratory bench tops. These tools include the multimeter, power supply, prototyping breadboard and oscilloscope, to name a few. Therefore, the MRP shall provide students with tools to create and analyze analog circuits, as well as collect and react to feedback from these circuits when they are designing for applications in robotics.

Computer Engineering (Digital) As computers become smaller and cheaper, many things that were once accomplished via analog circuitry have made the transition into the digital domain. The field of Computer Engineering provides an elegant solution to the previous statement. With speed increases in processing ability, tasks that traditionally called for analog circuitry, such as sound recording or signal analysis, are not able to be reproduced digitally with fidelity rivaling the analog counterpart.

Since many advanced concepts and theory of Robotics Engineering require more processing power than is available in an embedded system, an alternative again appears in the form of digital circuits. With the proliferation of smaller and inexpensive microcontrollers, tasks requiring large desktop computing systems can be broken up into smaller tasks and computed in parallel. Also, signal acquisition and analysis can be processed from the raw data taken from sensors into intelligible information that is at a level the main processor can easily and quickly work with. Additionally, many tasks usually handled by an embedded processor or microcontroller can be accomplished using a digital logic circuit. This frees the processor from menial or time-intensive tasks such as keeping count of wheel encoders or processing data through algorithms. As Field

Programmable Gate Arrays (FPGAs) become more common and cheaper, logic circuits can be implemented and modified with merely re-flashing the device. These logic circuits can be highly complex, even reaching the complexity of a 64-bit RISC architecture microprocessor with 165,000 or more logic gates.

3.4 Requirements

Computer Science

The requirements discerned from the course descriptions of the Unified Robotics courses specify characteristics of the processor board to be chosen. The computer science subject matter of the courses suggest a preference for high-level object oriented languages. Languages commonly used at WPI are Java and C++, though Python also seemed a viable choice for object oriented programming. Furthermore student's prior experience in C seemed to also denote compatibility with this language. The controller to be used in this project had to be able to compile and execute code written in these languages. The object oriented languages immediately led away from micro controllers, as these typically only execute C, Assembly, or other non object-oriented languages.

Several Professors and students expressed the desire to be able to network to their robot controllers. This would theoretically allow programming to be done without a direct connection. It would also allow for the robot to be remotely controlled. The Unified Robotics 3002 course description does state the course will discuss methods of communication—network capabilities is a requirement that can be derived from this.

Regardless of network capabilities, a direct connection to the robot would still be required. This is for when the robot is in use in an environment with no network connections possible. This was an easy requirement to fulfill—nearly all the boards considered defaulted to a serial connection.

Users of the processor board needed to be able to have direct access to read and write to a number of inputs and outputs on the board. This ranged from digital and analog inputs to writing to digital outputs. This meant direct hardware control on a software level. There could be no software permissions blocking a user from gaining access to different aspects of the hardware.

Electrical and Computer Engineering

Based on our research and experience we were able to we came up with some basic requirements for the electrical system of our robot. The robot was to have at least the following:

- 10 Digital I/O
- 5 Analog inputs
- 5 Serial I/O
- 1 Stepper Motor Port
- 5 Servo Ports
- 4 Driven Motors
- 4 Wheel Encoders
- 12V, 5V and 3.3V Power

- Co-Processors

The 10 digital I/O's, the 5 analog inputs, and the 5 servo ports were all based off of the vex controllers. Based on market research, the systems currently available show the need for at least 10 digital, 5 analog and 5 servo ports, from reading over the course descriptions and based on experience this would be sufficient for the labs (If necessary additional ports can be added through the serial ports).

When we looked at the different types of sensors that were being considered for the courses, we found that many used various types of serial to communicate. Most serial methods could be recreated in software through the digital I/O ports, but RS-232 and TTL serial communications were found to be the most commonly used therefore we require 5 serial ports of either TTL or RS-232, both of which can easily be converted using level shifters.

The drive system requires 4 wheel drive, therefore 4 motors are necessary. Each motor will be attached to an encoder in order for the user to determine speed and direction of each wheel. A stepper motor port was required because the code required to run a stepper driver takes up valuable processor time. The co-processors are microcontrollers that will offload some of the tedious tasks from the main processor. The stepper driver and the quadrature encoders require a repetitive task that will be done on the microprocessors.

12V power is required because it was the highest voltage required by any of the researched components. This 12V will then be converted down to 5V and 3.3V in order to power most common sensors.

Mechanical Engineering

Based upon our prior knowledge and research of similar systems the following requirements were set:

- The final dimensions of the robot allows for it to fit inside a cube with an edge length of 0.5 meters to allow for multiple robots (at least 3) to interact on a 12 foot by 8 foot table and also to be easily transportable by a single individual.
- A maximum weight of 7.5 kg allows for an average human adult to transport the robot over long distances (across WPI's campus) without undue fatigue.
- The chassis design is flexible so that it can be reconfigured in 30 minutes using only hand tools and without making any permanent modifications to the individual parts.
- The motors will be sufficiently powerful in order to propel the robot up a 45 degree incline at a speed of 0.6 meters per second, a theoretical power requirement of 11 Watts.
- All moving parts of the drive mechanism with exception to the output shaft and attached wheels shall be fully enclosed to prevent injury.
- The drive mechanism is able to operate for 500 hours with no need for adjustment, repair, or part replacement.
- The drive mechanism is mounted so that an individual familiar with the design and in possession of the proper tools is able to remove one unit and replace it with another in less than 5 minutes.
- The total time to construct the frame from finished parts should take no longer than 30 minutes.

3.5 Summary

In summary, the purpose of the project is stated to be to design and prototype a modular robotics platform for educational use in the Unified Robotics course curriculum at WPI. In order to qualify whether or not the project has been successfully completed, certain general objectives were established. Also, these objectives will give the project a direction to aim towards and a goal to achieve. Furthermore, concrete and specific requirements of the project are listed to give quantifiable measures to which to gauge the success of the project.

Chapter 4

Methodology

This Chapter describes the methodology used to ensure the final design realizes the completion of the requirements presented in Section 3.4. When creating the Mobile Robotics Platform we used the following methods to guide us to our final goal:

1. We researched the requirements for the 3000 level robotics classes and focused on key concepts that would be taught in the class.
2. A preliminary concept was created, along with requirements and a list of possible components.
3. Each product that was considered for use on the Mobile Robotics Platform was extensively researched and compared with other similar products.
4. System diagrams were created to plan out all of the electronics, each component was looked over to determine what was needed for an interface. Mechanical sketches were made and critiqued to determine the necessary specifications for the design. Program flow charts were created for the software planning, and programming language was decided on.
5. A final design was created and analyzed in Solidworks for all the mechanical components. Each electrical circuit was designed and tested on bread boards, and then put onto a circuit board using PCB123. Software functions were prototyped in pseudo code.
6. Finally, all the mechanical parts were machined and assembled, the hardware was ordered, along with the circuit board. All the code was written and tested on the assembled parts.

When designing this project we focused on the objectives of the 3000 level robotics classes. Each product that was considered for use on the Mobile Robotics Platform was extensively researched and compared with other similar products. Function and price were compared, and the best product was ordered. The software and electrical hardware was then designed to be compatible with the chosen hardware.

Chapter 5

System Design

5.1 Introduction

The final system design is detailed in this Chapter, with elaboration of the subsystems comprising the framework of the prototype. The design of the MRP is broken up into the modular systems that provide the structure for its operation. Each subsystem is described in detail, including the specifics behind the design of these subsystems.

5.2 Functional Block Diagram

A functional block diagram of the MRP system provides a useful understanding of the high-level system concepts that will be implemented in the design. An example of a preliminary functional block diagram for the MRP can be seen in Figure 5.1. A complete breakdown of the functional block diagram presents details as to the input, output, and function of each system and subsystem of the diagram.

5.3 Subsystems

Electrical Subsystems

The electrical systems consists of five subsystems; the single board computer, the interface board, the co-processing board, the motor drivers, and the power system. The single board computer is the main component of the electrical system. It is what holds the main processor that controls the robot. All of the other subsystems of the electrical system, aside from the power system, interface with the single board computer. The single board computer must have a console port to allow for programming directly from a PC, an Ethernet port for remote programming, analog inputs and digital I/O ports for communication with sensors and control of circuits, and finally serial ports for communication with other peripheral devices. Through the serial ports the single board computer can then communicate to two motor controllers. Each of these motor controllers will then convert the signal to drive up to two motors each.

The co-processing board will be used to offload some of the tedious tasks from the main processor. These will include six AVR chips that will each handle one task. Four of the chips will each get assigned one encoder to continuously keep track of the count. These encoders will remember the value they used until the main processor asks for it. Another AVR will control the

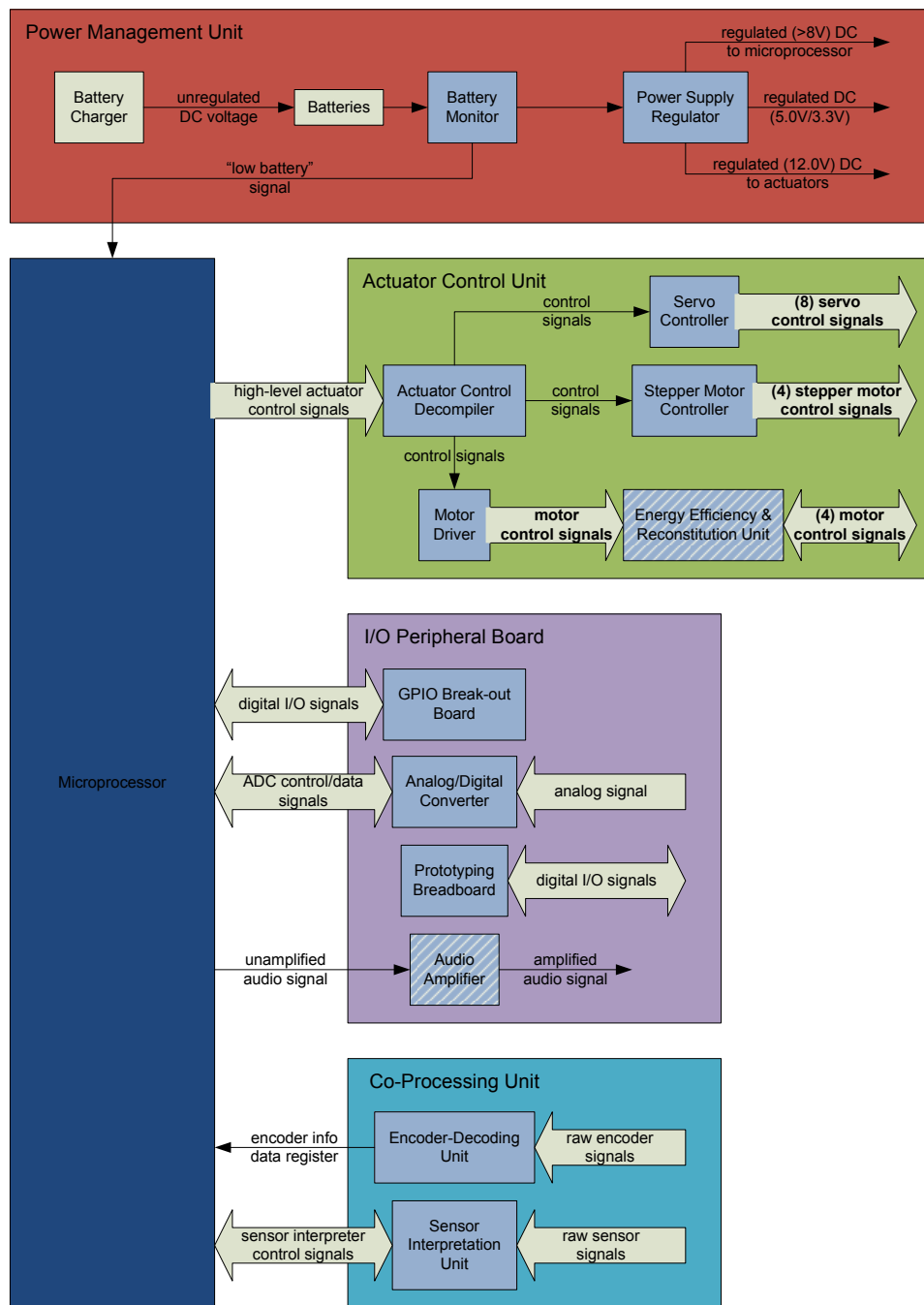


Figure 5.1: Systems-level functional block diagram of the MRP.

stepper driver. This AVR will do nothing until the main processor tells it to move the stepper motor a certain number of steps, then the chip will generate the necessary control signal. Each chip will communicate to another AVR through TTL level serial. This AVR will act as a multiplexor, directing the incoming serial signal from main processor to the correct chip.

The interface board takes the open ports and makes them easily accessible on the outside of the robot with standard connectors and terminal blocks. Another function of the interface board is to protect the single board computer from receiving a signal that could potentially harm it. The interface board connects to the digital I/O ports and limits the current to what the single board computer can handle, and then connects each to a terminal strip. Some digital I/O ports are used for status LED's, a reset button, and jumpers for user interaction. The analog inputs are limited to the specified voltage then connected to terminals. Each RS-232 serial port will be wired up to a standard DB9 connector, while TTL ports will use two terminals for TX and RX pins. The servo controller on the interface board will receive a serial signal and convert it into a PWM signal to control servo motors.

The Power system of the robot consists of a battery fed through a fuse block that distributes 12V to each system of the robot. Those systems which require 5V or 3.3V will go through a buck converter to lower the voltage. 5V and 3.3V terminals will be accessible on the interface board to power external sensors. A functional block diagram of these electrical subsystems can be seen in Figure 5.2.

Processing Subsystems

getDI () Software Routine

`getDI ()` is the first of two fundamental programs to deal with digital inputs and outputs. The program accepts an integer value of a pin, which is expected to be between one and fifteen inclusive. The function will eventually return the digital value on the requested pin. The pin numbers are selected as if they were sequentially ordered as on the MRP's breakout board. This is not the case, and the pin actually chooses the appropriate bit to return from the correct register.

First the program determines if any action is required to prepare the digital input pins to be read. For the first five pins, attached physically to the TS-7800's DIO header and to register `0xE8000004`, the program must first set the appropriate bit on its register to a logic high. For pins six through fifteen, belonging to the GPIO C header, the program must first set the appropriate bit in GPIO C's direction register, `0xE800001C` before reading its data register. After these sets are completed, the appropriate bit is selected via `getBit ()`.

`getBit ()` accepts a pin number and, through a simple switch statement, will return which bit location the desired information is in.

A "bitmask" is created, which is a value of 0's in every location in a register except the desired bit.

The program once again checks to see which set of pins is being read—this determines which register is finally read. An integer value—`readvalue`—is assigned either the DIO header's data register or GPIO C's data register. This value is then bitwise ANDed with the bitmask in order to set all bits but the desired bit to 0. This value is bit shifted to the right `getBit ()`'s amount of bits and returned—it is the lone value that was set on the desired bit, and thus the pin.

A flowchart of the `getDI ()` software routine can be seen in Figure 5.3.

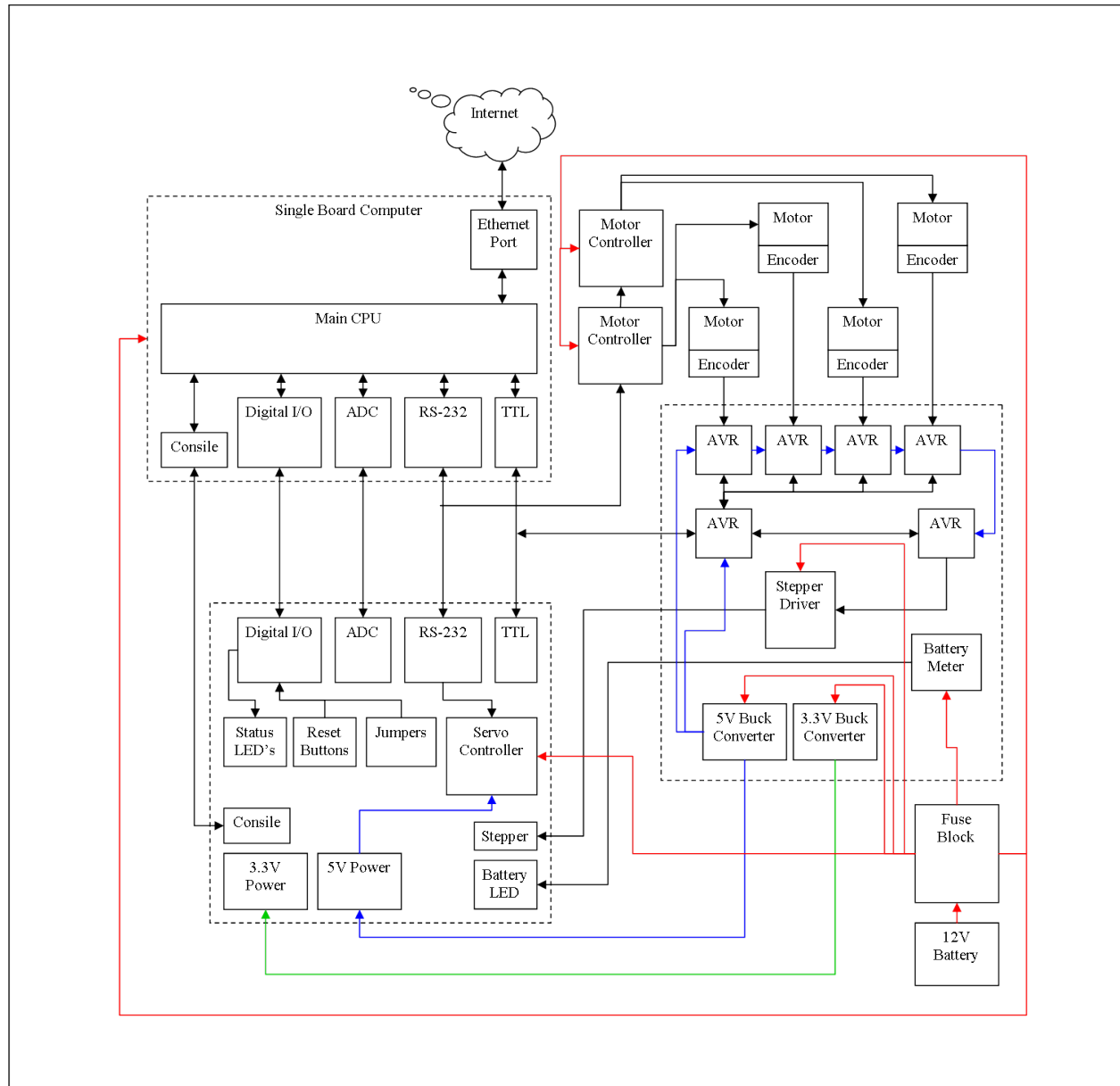
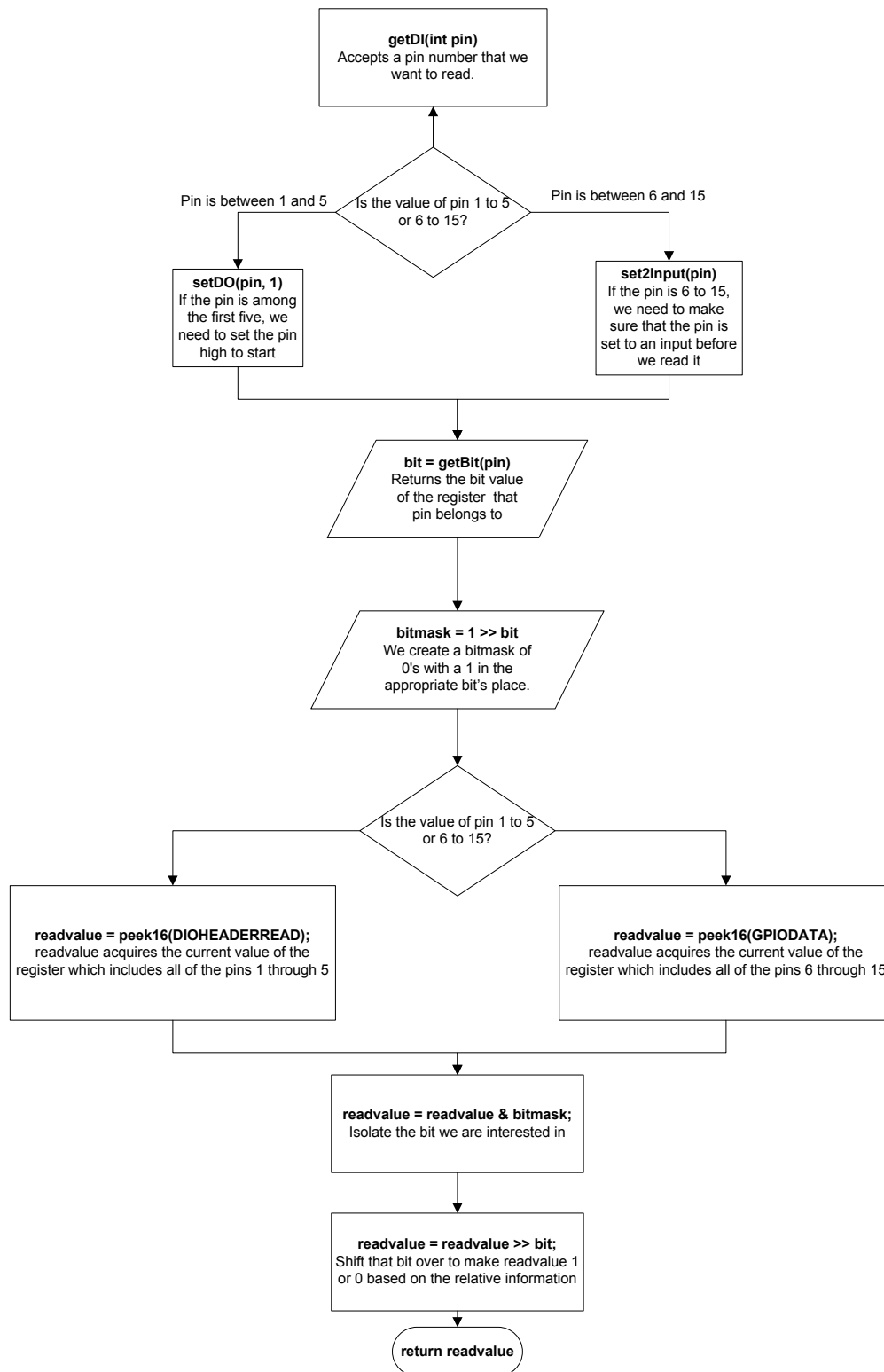


Figure 5.2: Functional block diagram of the electrical subsystems.

Figure 5.3: Flowchart of `getDI()` software routine, querying the digital inputs.

setDO () Software Routine

`setDO ()` is very similar to `getDI ()`—they operate nearly identically until `getDI ()` would return a value. The primary difference beforehand is in the first if statement, `setDO ()` prepares both the DIO header and GPIO C pins to be outputs instead of inputs.

Once the program reads the appropriate registers for the pin, a new bitmask is created, different from `getDI ()`. This bitmask is a value of 0's in all locations except the desired bit, which is set to the value desired. This way the value may be bitwise ORed with the register's read value to set all values stored to logic high except the desired bit. This prevents other pins from having their values changed each time a digital output is set.

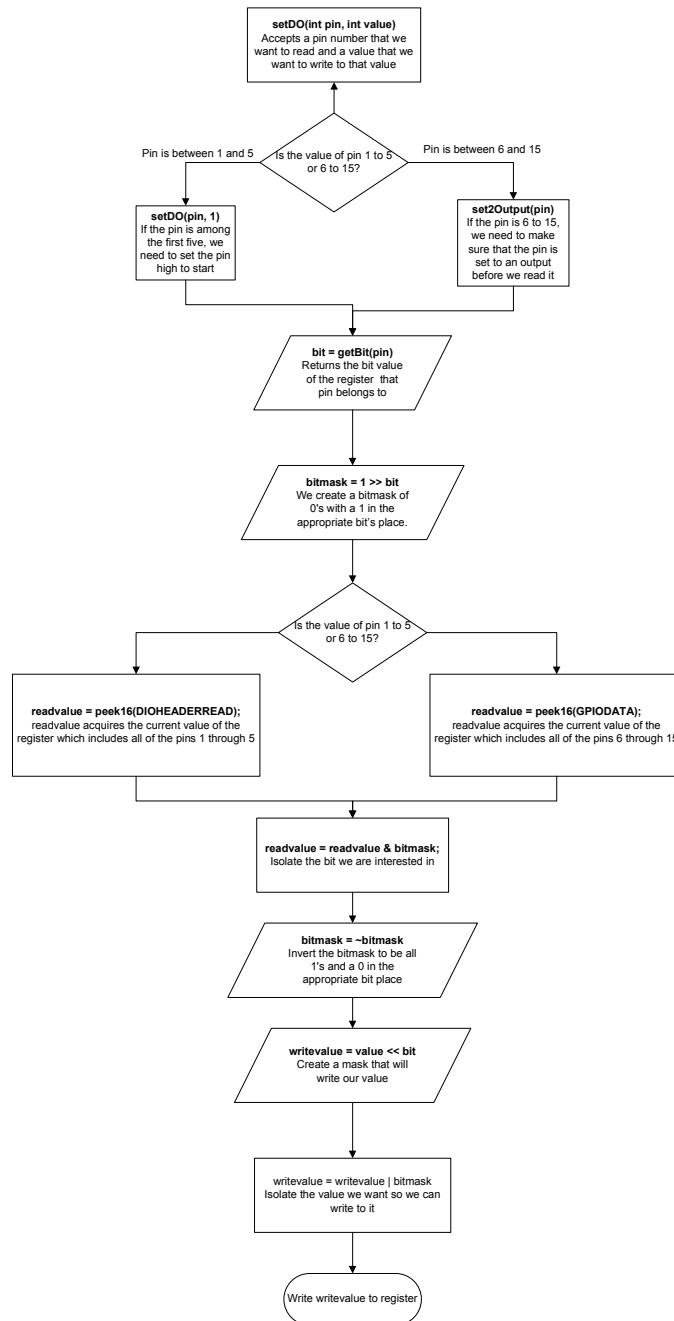
A flowchart of the `setDO ()` software routine can be seen in Figure 5.4.

5.4 Summary

This chapter described the final design of the electrical and computer systems. The electrical system contained several subsystems, controlling features such as:

- Power management
- Central Processing
- Actuator Control
- Input and Output Control (Digital and Analog)
- Co-Processing Subsystem Unit

The software was divided into a series of sub routines that controlled the interaction between software abstraction to hardware control.

Figure 5.4: Flowchart of `setDO()` software routine, setting the digital outputs.

Chapter 6

Design Details

6.1 Introduction

This section will go over the research into each component group, as well as why each component was eventually selected. It will also look over the design details of each section of the MRP individually and the design decisions that were made.

6.2 Trade Studies

Processor

When researching single board computers for comparison, much of the data came from PC104 Embedded Consortium. This community put together a database of PC104 boards made by different manufacturers. This was very useful in picking out boards that suited our needs. We also referenced an on-campus project titled “Dig-It”. Much of the hardware that has been selected for Dig-It fulfills many of this project’s requirements due to both projects being robotic in nature.

Some of the more important features we were focusing on were processing power, data storage and input/output. Wireless capability and support was also considered an important facet. More key considerations were the development tools available for the boards as well as operating systems supported. This data was placed into table form for easy comparison.

The media, in which the onboard memory is stored, as well as if it is expandable, played a key role in the selection of a board. Almost all of the boards have an interface to connect an external hard drive. Since our robot will be mobile in numerous environments there is the potential for some abrupt mechanical shock from collision. This could damage a hard drive. Therefore a solid state memory flash drive is the best solution. Also, a removable flash card will allow for the creation and duplication of an image of the final software, OS, development tools, and project-created library for easy production of the platform. This will speed up the creation of a labs worth of robotic platforms.

The last common feature looked at was the input and output features of the board. Almost all of the boards had USB ports, serial ports, video ports and Ethernet ports. This was so universal that it is pointless to seriously compare boards this way.

A value analysis of the available boards was performed, as can be seen in Table 6.1.

Based on these specifications the Technologic Systems TS-7800, the CompactRio, and the VersaLogic Puma top all other boards. These all have support for compact flash card support, 500

Table 6.1: Value Analysis of candidate processor systems.

General Specifications	Total	TS-7800	TS-7200	CPC304	Jrexplus	Hurricane	Cool RoadRunner	Helios	Puma	SBC	CompactRio
Ease of use	5	4	4	3	3	2	2	4	4	3	5
Cost	10	10	9	5	0	0	0	0	6	5	1
Company Reliability	15	10	10	6	7	10	10	7	11	7	15
Part Reliability	15	12	10	7	10	9	9	8	11	10	13
Documentation/Knowledge Base	15	12	10	8	5	11	11	7	8	10	12
Product Specifications											
RAM	10	6	3	8	10	10	10	6	8	9	8
Processor Speed	10	10	5	10	10	10	10	7	10	8	8
Memory	10	10	10	7	8	5	9	7	6	7	3
Expandability	15	15	15	10	9	12	12	12	12	9	15
FPGA?	10	10	10	0	0	0	0	0	0	0	10
Outputs	10	10	10	10	10	10	10	10	10	10	10
Total	125	109	96	74	72	79	83	68	86	78	100

MHz processors, and have all of the necessary I/O support. Only one of these three boards, the Technologic Systems TS-7800, supported every requirement for the project, while also having

- Wireless network support
- On board Field Programmable Gate Array (FPGA)
- Low power consumption
- Expandable hardware
- ADC and DAC converters

For this reason the TS-7800 is the foremost recommended processor for this project.

Motor Controller

Finding a number of different brand names of motor controllers was particularly easy—merely searching robot hobby sites pointed me towards a number of manufacturers that produced motor controllers of all kinds of sizes, power rating, and quality. Almost immediately, however, I found worrying words of wisdom that proved true at times. One manufacturer’s website boasted how their tests for rating continuous amperage of their motor controllers were done over a long period. They made claims to their competitor’s ratings being inflated by only reading the max continuous amperage for mere minutes or less! Obviously this would be a significant problem for our base design. This became an almost common theme amongst the more “combat robot” oriented motor controller distributors/manufacturers. In the end it became an issue of trust. The manufacturers that list the time that their motor controllers can maintain continuous amperage are to naturally be trusted more than those that do not.

Common features throughout the motor controllers were regenerative breaking. Some had channel inversion; two even had signal-lost fail safes to cut the motors. A few I noted as being used widely in robotic competitions, proving their quality.

Input methods varied. The most common was a standard RC control signal. A handful had support for serial input while one or two supported just a varying analog voltage.

Dimension Engineering’s speed controllers fulfilled every requirement we looked for, with interesting features such as accepting a large number of inputs: analog PWM, RC signal, serial input, and packetized serial input. Each one controls two motors (with the exception of their SyRen series). They claim their continuous amperage rating is for prolonged use, and all their controllers have built in regenerative breaking.

While pricier per motor, IFI’s Victor 884 Speed Controller is also highly recommended. While it controls only one motor per Victor, is pricier, and also requires a separate breaker to protect it, the Victor is well proven through its extensive use in the FIRST Robotics Competition. The robots in these competitions are heavier than our expected maximum payload weight, so we know that Victors can handle motors with sufficient amp draw.

BaneBot’s controllers are miniature—nearly the size of a quarter. They claim decent amp draw for several minutes nonstop use and are listed for a low price. Heat dissipation due to its compact design may become problematic. BaneBot’s products are designed for and used in robot combat competitions, proving that they can run a robot’s motors at least for several continuous minutes.

Dimension’s Engineering’s Sabertooth line is the recommended choice because of the following features:

- Decent price for dual channel speed controllers

Table 6.2: Value Analysis of candidate motor controller solutions.

		BaneBots Controller								
General Specifications	Total	Sabertooth	Victor 884	BaneBots Controller	SyRen	Scorpion	Sidewinder	Simple-H	OSMC	AX2550
Ease of use	15	15	10	10	15	15	15	12	15	12
Cost	15	10	5	10	10	10	0	10	0	0
Company Reliability	10	8	5	8	8	8	8	8	8	8
Part Reliability	10	8	10	7	8	8	10	7	8	10
Documentation/Knowledge Base	15	12	15	12	12	12	15	8	15	10
Product Specifications										
Interface	15	15	5	5	15	5	15	12	5	10
Number of motors	5	5	0	0	0	5	5	5	5	5
Continuous Amperage	15	15	15	10	10	12	15	12	15	15
Additional abilities	5	5	0	0	5	3	5	0	0	5
Total	105	93	65	62	83	78	88	74	71	75

- Voltages and continuous amperage is within the required range
- Regenerative braking
- Flexibility of accepted input signals

Wheel Encoders

In the search for the most appropriate rotary shaft encoder we came across several industrial options. These options were very high precision, rather large, and expensive considering the need for four in total. We narrowed our search to include optical shaft encoders that used quadrature, were designed for the enthusiast rather than the industrialist, had sufficient accuracy, and were relatively inexpensive. We found the two manufactures used in our comparison; US Digital and Bane Bots. US Digital is “in the business of solving motion control problems by employing the absolute best minds with the most innovative manufacturing facility in the industry” and BaneBots is a small retailer of robotic electronics

The Common features throughout the offerings of both companies were the use of quadrature as well as similar construction methods and the standard 5V power voltage. Only the products from US Digital allowed the use of many sizes of shafts for each model, the offerings from BaneBots are designed to fit the motors and gear trains that they sell and have a single size for each model.

Input methods of both companies are identical, four pins. The channels A and B are binary outputs. Depending on the direction A will transition before or after B.

Table 6.3: Value Analysis of candidate optical shaft encoders.

General Specifications	Total	E4P	E7P	36mm Encoder
Ease of use	10	7	7	7
Cost	15	13	9	9
Company Reliability	10	8	8	5
Part Reliability	10	7	7	5
Documentation/Knowledge Base	10	8	8	4
Product Specifications				
Interface	10	8	8	8
Accuracy	10	8	10	4
Total	75	59	57	42

Based upon a value analysis of the choices for optical shaft encoders, the first choice is the E4P from US Digital, in particular because of its small size and low cost. The E4P has options of accuracies from 100 Counts Per Revolution (CPR) to 360 CPR.

6.3 Construction and Implementation

Much of the design details corresponding to the electrical systems in the MRP are highly technical. For the sake of portability, the information included in this section has been compiled separately and is included in Appendix ??.

6.4 Summary

In this section we went over the details of the selection process for selecting components. After gathering research on all of the potential components that we could use, trade studies were done on each one. The cost, compatibility, and documentation database were weighted along with some individual characteristics of the components. This data was then used to select the best fit for our project. The products selected were the Technologic Systems TS-7800 single board computer, Dimensions Engineerings Sabertooth motor controller, and US Digital's E4P wheel encoder. This section also covers the design details of the MRP. Going over in detail the design of each circuit schematic designed into the project, and all of the code written for the libraries.

Chapter 7

Results

7.1 Introduction

This section will present and analyze the outcomes of the Modular Robotics Platform. By using these results and comparing them to our original objectives we can determine the overall success of the project and individual systems. An understanding of the performance of the MRP can be derived from these results.

7.2 Testing and Evaluation

Computer Science

In order to test the library developed for regular users and the individual hardware components, several test programs were developed. These programs implemented the functions individually to provide terminal based control of the hardware on the MRP.

Digital Inputs and Outputs

The program `DIOTest.c` is designed to test all the digital inputs and outputs on the MRP. The program accepts a terminal-based command and then either returns the value of the input, or sets the output to the desired value. To test the input capabilities of each pin, a 5 volt source was applied to each pin individually. A change in the value returned by `DIOTest.c` was observed. To test the output capabilities of each pin, an LED in series with a small resistor was hooked up to each pin. Calling `DIOTest.c` set the LED either on or off properly.

Analog-to-Digital Conversion

A relatively simple program, `ADCTest.c` would print the analog value read at the requested channel. In order to demonstrate a working analog-to-digital conversion (ADC) program, both a digital power supply and potentiometer were used to modify the voltage. This change was witnessed through terminal control of the robot.

Motor and Servo Testing

In order to test both motors and servos, `motorTest.c` was written. This program allows the user to, through the terminal, implement the library's motor and servo controller. Servos and motors

were attached to the appropriate terminals and the appropriate physical reaction to the program's control was witnessed.

7.3 Project Economics/Economic Considerations

One application of the results from this project is the use of the MRP in the Unified Robotics courses at WPI. In particular, it is intended for use in course laboratory exercises for the demonstration of robotics theory. This dictates that the final product be reproducible for a sizeable group of students at a reasonable cost. Stated earlier, the target cost for the complete MRP system is between \$1000 and \$2000 per unit.

7.4 Summary

This section goes over the testing done on the MRP. All parts of the project were extensively tested throughout the design and construction phases. Since all of the debugging was done beforehand, the final assembly of the prototype was met with minimal errors. This section also talks about how the final project fell into the budget range of \$1000–\$2000.

Chapter 8

Summary and Conclusions

8.1 Introduction

With the completion of the project design process, a detailed documentation of the comprising processes has been presented. In defining the problem statement, project objectives were presented. These are evaluated and discussed, as well as topics of further development, including items that can be improved upon with future work.

8.2 Completion of Project Objectives

To determine the completion of project objectives, the initial project objectives were divided into two categories: Computer Science objectives, and Electrical & Computer Engineering objectives. These objectives are again separated into two columns, the first reflecting the specific objective being evaluated, and the second stating the state of the objective at the conclusion of the project. This evaluation can be seen in Table 8.1.

8.3 What could be improved upon?

As is common with most project, throughout the course of the project, the students discovered certain things that, in retrospect, could have been improved upon if the project were done again. Similarly, issues were found that would be helpful to students were they to take the results of this project further as the topic of an additional project. Again, as in the previous section, these topics were divided into two categories: Computer Science and Electrical & Computer Engineering. These topics are separated into two columns, the first reflecting the specific topic to be discussed, and the second describing ways in which the topic could be improved upon. This evaluation can be seen in Table 8.2.

8.4 Summary

This project started with the lofty goal of designing a suitable platform for upper level Robotics Engineering courses. These objectives were expanded to include budget restraints, a target audience, theoretical course requirements, and an expanding list of desired capabilities.

Over the course of the project, these objectives eventually created requirements, described earlier in this document. These requirements guided the construction of the MRP prototype. Once the

software library to allow simple hardware control of the robot was created, the MRP's possibilities became apparent.

The original goal for the platform—to be used in a series of upper level Robotics classes—

Table 8.1: Completion of objectives.

Objective	Objective Status
Computer Science	
Programmable by user	The MRP is programmable in any language that can be compiled in Debian Linux, including, but not limited to, C/C++, Java, Python and Scheme.
Wired Connection	The MRP can connect to a student's computer through its serial port. A student can completely control and program the robot through this.
Network abilities	The MRP can connect to WPI's campus network through an ethernet connection. Wireless connection proved successful in areas with unprotected wireless networks. WPI's network is protected by a WPA Enterprise encryption and has not yet allowed the MRP to connect.
Network Programming	A student may wirelessly control, program, and completely modify the on-board programming of an MRP that is connected to a network.
Library	A C library that, upon compile time, can allow students to easily control the MRP's hardware was completed and tested.
Field Control	A student can start, stop, and reset code on a robot without a computer.
Operating System	Debian Linux, compiled for the ARM9 processor, successfully runs and controls the MRP.
Expandability	Due to the vast array of programming languages available to control the MRP, the constant updates of the Linux for ARM operating system, and the vast amount of untapped power within the processor, the MRP has excellent potential in expanding to meet the needs of complicated projects in research and education.
Electrical & Computer Engineering	
Digital I/O	The MRP has a total of 10 easily accessible digital I/O ports with built in current protection.
Analog inputs	The MRP has 5 Analog to Digital Converter ports that can accept a range of 0–3.3V, and are protected from overvoltage.
Serial I/O	The MRP has 4 serial ports for use, 2 RS-232 ports accessible through DB9 connectors and 2 TTL ports assessable through terminal blocks.
Servo Ports	A servo Controller is connected to the MRP allowing for control of up to 8 servos.
12V, 5V and 3.3V Power	Internally the MRP runs on a 12V system with both 5V and 3.3V accessible through terminal blocks on the interface board.
Co-Processors	The Co-Processors of the MRP are wired up but still await programming.

seemed to only touch upon a fraction of the possible applications for the MRP. While we among the project agree that, had we started the project knowing what we did now, the outcome would be significantly different, we also agree that the outcome of the project was a success.

By the time that this document was written, a platform (or no platform, as it is a one-axis arm) has been chosen for the first rendition of Unified Robotics 3001. No platform has been chosen for Unified Robotics 3002 as of yet. Some would consider this a failure for a platform designed to fill this “market space,” but we would argue otherwise. The expected educational outcomes for the Unified Robotics course has since changed over the course of the project, changing drastically what was required in a platform. The MRP also still has an excellent future as a project prototype, hastening development as it was meant to do. For an initial attempt at developing such a platform, the resulting platform is a success.

Table 8.2: Topics to be improved upon.

What could be improved upon?	How could it be improved?
Computer Science	
Wireless Connectivity	Despite our best efforts, the MRP never successfully connected to WPI's wireless network due to difficulties with its encryption. Linux has since made significant improvement with wireless compatibility and shows promise that future ARM distributions will perform better.
Boot-up Time	The initial boot-up sequence of the MRP could be drastically improved. Currently it can reach a time up to two and a half minutes depending on the previous method of shutting down the MRP and network connection settings. Modifications to the operating systems and networking scripts could be made to improve the boot-up time.
Additional Distributions	Since the project's end, additional distributions of Linux have either been released or update for the ARM processor. An exploration of compatibility with other distributions could improve the performance of the robot.
Libraries for other languages	Only a C library was created to control the hardware on the MRP. Libraries in other languages would also greatly enhance using the MRP for more advanced projects.
Electrical & Computer Engineering	
AVRs	The pins on the AVR chip could have been wired to a header terminal to allow for each chip to be completely re-programmed by students.
ADC	The MRP could have used more than 5 ADC ports, as well as an ADC with a faster refresh time.
FPGA	When the TS-7800 was purchased we were under the impression that the TS-7800 would have a completely user programmable FPGA, when we received the board we determined that the FPGA was non user programmable.
Remove ribbon Cables	The ribbon cables connecting the circuit board to the TS-7800 clutter the platform. A stackable arrangement of these boards would be ideal.

Bibliography

- [1] Worcester Polytechnic Institute, "Robotics engineering - courses".
- [2] Innovation First Inc, "Classroom lab kit".
- [3] Gentleware AG, "Model to business - uml glossary", 2007.
- [4] Brad Miller, "Wpilib", 2008.
- [5] Innovation First Inc, "Vex robotics design system", 2008.
- [6] iRobot, "irobot create", 2008.
- [7] Microsoft Corporation, "irobot create and roomba with microsoft robotics studio", 2008.
- [8] Evolution Robotics, "Evolution robotics er1 robot kit", 2008.
- [9] Carl's Electronics, "Robodesigner educational robot platform", 2008.
- [10] Wikipedia contributors, "Lego mindstorms nxt", 2008.
- [11] LEGO, "Lego mindstorm's nxt overview", 2008.
- [12] Surveyor, "Surveyor srv-1 blackfin robot", 2008.
- [13] Surveyor, "Surveyor srv-1 blackfin camera", 2008.
- [14] Acroname, "Customize a garcia", 2008.
- [15] MobileRobots Inc., "Seekur", 2008.
- [16] Inc Segway, "Robotic mobility platform (rmp)", 2008.
- [17] Michael David Medeiros and Aaron CHristopher Bergeron, "Development of a gas-pipe climbing robot", Tech. Rep. 05B014M, Worcester Polytechnic Institute, 2005.
- [18] Eric Carter Tripodi and Christopher Phillip Bitzas, "Design of a six-legged walking robot", Tech. Rep., Worcester Polytechnic Institute, 2001.
- [19] Christopher D. Korzeniowski, Francisco T. DeMolina Cobo, and Kevin M. Bobrowski, "Search and rescue robot", Tech. Rep. 07D070M, Worcester Polytechnic Institute, 2007.
- [20] Illah R. Nourbakhsh, Kevin Crowley, Ajinkya Bhawe, Emily Hamner, Thomas Hsiu, Andres Perez-Bergquist, Steve Richards, and Katie Wilkinson, "The robotic autonomy mobile robotics course: Robot design, curriculum design and educational assessment", *Autonomous Robots*, vol. 18, no. 1, pp. 103, Jan-1 2005.

-
- [21] J. M. Mirats Tur and C. F. Pfeiffer, "Mobile robot design in education", *IEEE robotics and automation magazine*, vol. 13, no. 1, pp. 69, Mar-1 2006.

Appendix A

Electrical Design Details Reference

Much of the design details corresponding to the electrical systems in the MRP are highly technical. For the sake of portability, these Electrical Design Details for the Mobile Robotics Platform are included as a standalone entity in the pages that follow.

MOBILE ROBOTICS PLATFORM: ELECTRICAL DESIGN DETAILS

Compiled in majority by Matthew DeDonato.

Originally included in the *Mobile Robotics Platform* team report as an appendix.

1 TS-7800

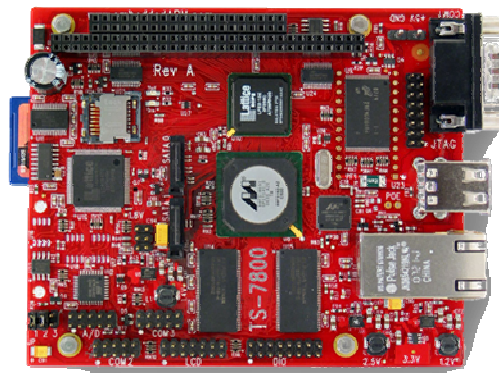


Figure 1: TS-7800

For the main processing board we choose to go with the TS-7800 single board computer created by Technologic Systems. The TS-7800 has connectors and headers that allow access to all of the features of the board. We will be using the DIO header, the 40-PIN GPIO header, the A/D header, the COM 1 connector, and COM 3 header.

1.1 LCD HEADER

The LCD header is numbered as follows (Pin 1 is next to the dot on the silkscreen):

2	4	6	8	10	12	14
.1	3	5	7	9	11	13

Figure 2 - LCD Pin Numbers

The pin assignments are as follows:

Table 1 - LCD Pin Assignments

Pin	Feature	Note
1	5V	
2	GND	
3		

4		
5		
6		
7	Digital I/O	2.2K Resistor Pull Up
8	Digital I/O	2.2K Resistor Pull Up
9	Digital I/O	2.2K Resistor Pull Up
10	Digital I/O	2.2K Resistor Pull Up
11	Digital I/O	2.2K Resistor Pull Up
12	TTL_TxEn	UART #7
13	TTL_Tx	UART #7
14	TTL_Rx	UART #7

1.2 DIO HEADER

The DIO header is numbered as follows (Pin 1 is next to the dot on the silkscreen):

2	4	6	8	10	12	14	16
.1	3	5	7	9	11	13	15

Figure 3 - DIO Pin Numbers

The pin assignments are as follows:

Table 2 - DIO Pin Assignments

Pin	Feature	Note
1	Digital I/O	2.2K Resistor Pull Up
2	GND	
3	Digital I/O	2.2K Resistor Pull Up
4	Input Only	
5	Digital I/O	2.2K Resistor Pull Up
6	SPI_FRAME	For Temp Sensor
7	Digital I/O	2.2K Resistor Pull Up
8	Digital I/O	20k-150k Resistance Pull Up
9	Digital I/O	2.2K Resistor Pull Up
10	SPI_MISO	For Temp Sensor
11	TTL_TxEn	UART #6
12	SPI_MOSI	For Temp Sensor
13	TTL_Tx	UART #6
14	SPI_CLK	For Temp Sensor
15	TTL_Rx	UART #6
16	3.3V	

1.3 40-PIN GPIO HEADER

The GPIO consists of two headers. The first is a 64 pin header labeled “A” and “B”, and the second is a 40 pin header labeled “C” and “D”.

D	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 4 - 40-PIN GPIO Pin Numbers

A	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
B	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Figure 5 - 64-PIN GPIO Pin Numbers

For our purposes we have decided to use only the 40-PIN GPIO header because the header contains more than enough I/O pins, plus serial ports and power pins. The 64-PIN GPIO only contained I/O and power ports. The pin assignments for the 40 pin connector are as follows:

Table 3 - 40-PIN GPIO Pin Assignments

Pin	Feature	Note	Pin	Feature	Note
C0	GND		D0	GND	
C1	Digital I/O		D1	Digital I/O	
C2	Digital I/O		D2	Digital I/O	
C3	Digital I/O		D3	Digital I/O	
C4	Digital I/O		D4	Digital I/O	
C5	Digital I/O		D5	Digital I/O	
C6	Digital I/O		D6	Digital I/O	
C7	Digital I/O		D7	Digital I/O	
C8	Digital I/O		D8	3.3V	
C9	Digital I/O		D9	Digital I/O	
C10	Digital I/O		D10	Digital I/O	
C11	Digital I/O		D11	Digital I/O	
C12	Digital I/O		D12	Digital I/O	
C13	TTL_TxEn	UART #9	D13	Digital I/O	
C14	TTL_Tx	UART #9	D14	Digital I/O	
C15	TTL_Rx	UART #9	D15	Digital I/O	
C16	TTL_TxEn	UART #8	D16	5V	
C17	TTL_Tx	UART #8	D17	Digital I/O	
C18	TTL_Rx	UART #8	D18	GND	
C19	GND		D19	GND	

1.4 A/D HEADER

The analog to digital header provides connections for the analog to digital converter. The A/D header is numbered as follows (Pin 1 is next to the dot on the silkscreen):

2	4	6	8	10
.1	3	5	7	9

Figure 6: A/D Pin Numbers

The pin assignments are as follows:

Table 4 - A/D Pin Assignments

Pin	Feature	Note
1	ADC_0	0 to 3.3V tolerant
2	GND	
3	ADC_1	0 to 3.3V tolerant
4	GND	
5	ADC_3	0 to 3.3V tolerant
6	GND	
7	ADC_2	0 to 3.3V tolerant
8	GND	
9	ADC_7	0 to 3.3V tolerant
10	GND	

1.5 COM 1 CONNECTOR

COM 1 is wired into a male DB9 connector. The pins in this connector are numbered as follows:

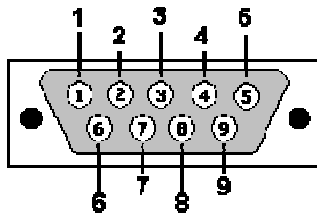


Figure 7 - DB9 Connector Pin Numbers

The pin assignments are as follows:

Table 5 - COM 1 Pin Assignments

Pin	Feature	Note
1	RS-232_Rx	UART #1
2	RS-232_Rx	CONSOLE
3	RS-232_Tx	CONSOLE
4	RS-232_Tx	UART #1
5	GND	
6		
7	RS-232_Tx	UART #0
8	RS-232_Rx	UART #0
9		

1.6 COM 3 HEADER

The COM 3 header is used to connect to two more serial ports, it is numbered a bit differently from the other headers. The pins in this connector are numbered as follows (Pin 1 is next to the dot on the silkscreen):

6	7	8	9	10
.1	2	3	4	5

Figure 8 - COM 3 Pin Numbers

The pin assignments are as follows:

Table 6 - COM 3 Pin Assignments

Pin	Feature	Note
1		
2	RS-232_Rx	UART #4
3	RS-232_Tx	UART #4
4		
5	GND	
6		
7	RS-232_Tx	UART #5
8	RS-232_Rx	UART #5
9		

1.7 ANALOG TO DIGITAL CONVERTER (ADC)

The analog to digital converter built into the TS-7800 is powered by an ATmega48 AVR microcontroller. This microcontroller has 5 ADC channels that it can sample at 10-bit resolution every 0.5 seconds. The value is then communicated back to the main processor through I2C serial protocol.

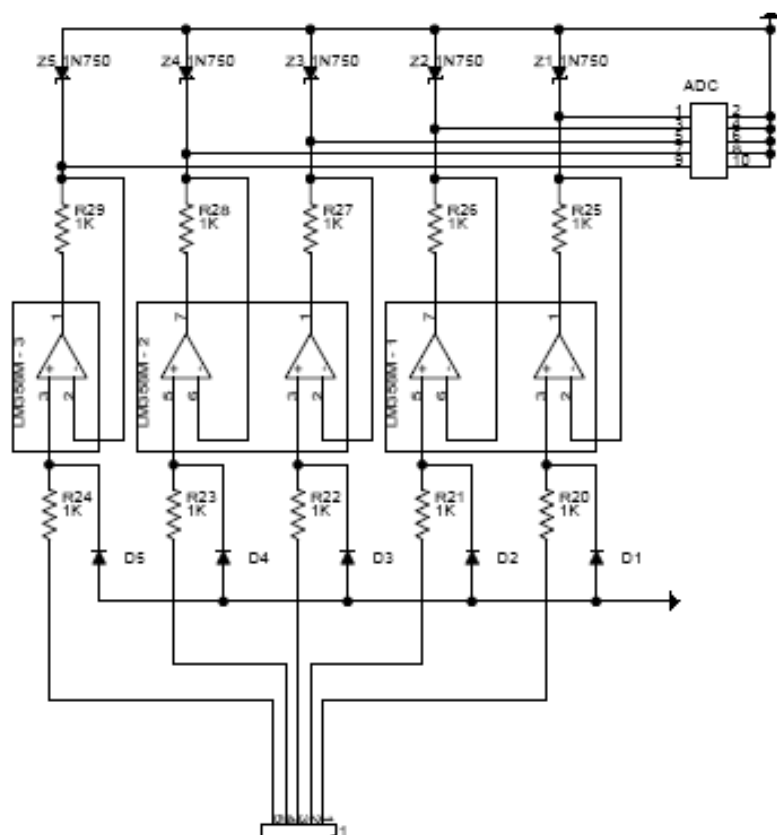


Figure 9 - ADC Schematic

The A/D header on the TS-7800 plugs into the ADC socket on the interface board. Pins 2, 4, 6, 8, and 10 are all tied to ground. Pins 1, 3, 5, 7, and 9 are each run through a voltage limiting circuit, to protect the onboard ADC, and then connected to a terminal block on the interface board.

Table 7 - A/D Pin Use

A/D PIN (TS-7800)	USE (Interface Board)	NOTE
1	ADC 1	0 to 3.3V tolerant
2	GND	
3	ADC 2	0 to 3.3V tolerant
4	GND	
5	ADC 3	0 to 3.3V tolerant
6	GND	
7	ADC 4	0 to 3.3V tolerant
8	GND	
9	ADC 5	0 to 3.3V tolerant
10	GND	

1.7.1 VOLTAGE LIMITING CIRCUIT

Since the onboard ADC chip only reads voltages from 0 to 3.3V, it cannot handle voltages that are much higher than 3.3V without being destroyed. The problem is that we have a 5V power rail; this creates the potential for 5V to be put through the ADC. To protect against this potentially catastrophic scenario, we designed a voltage limiting circuit that would output any voltage inputted into the circuit from 0 – 3.3V, but once the input voltage passes 3.3V the output of the output voltage will still always remain at 3.3V.

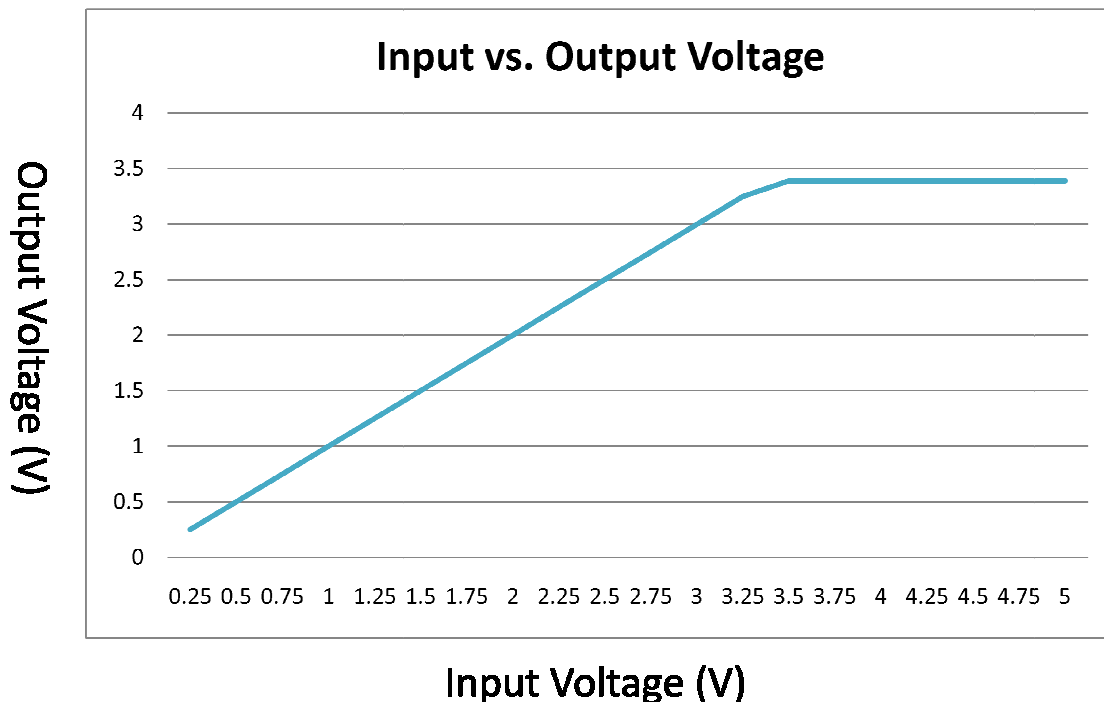


Figure 10 - Input vs. Output Voltage Graph

The circuit in Figure 11 is what was designed to limit the voltage. When the input voltage comes in it is sent through a 1K ohm resistor to protect the op-amp from over current. The voltage is then sent into a general op-amp. In our case we used chip LM358M because it met our specifications and was readily available. Because of the feedback loop in the circuit the op-amp will continually increase the voltage at the output point 1, until the voltage at point 2 is equal to the input voltage at point 3. The voltage out of the op amp which is equal to the input will travel through the resistor, to protect the ADC from over current, and then out to the output of the circuit. Unless of course if the voltage passing out exceeds the threshold of the zener diode. In this case the zener diode will close and direct all of the current to ground. The zener diode we picked was 1N750. It is rated for 4.7V, but because zener diodes work on a curve the op-amp will continuously force open the zener diode at 3.3V exactly, creating the desired effect the output curve in Figure 10. This circuit also protects against negative voltage by means of the general diode D5. If a negative voltage is placed on the input the diode will allow the current to flow from ground instead of through the output.

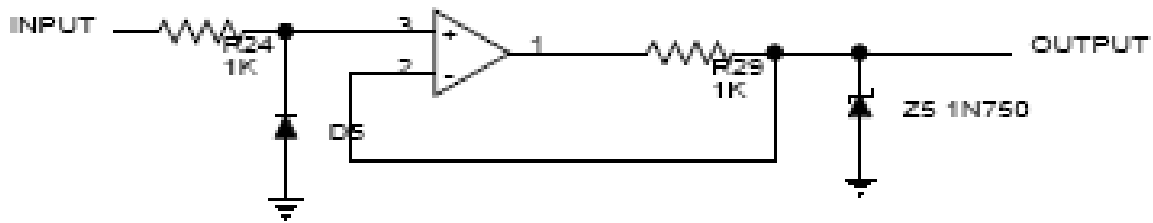


Figure 11 - Voltage Limiting Circuit

2 CONNECTOR PLATE

The connector Plate is where most of the interaction with the robotics platform will occur. It is designed to be mounted on the top or back of the robot depending on the user's preference. The plate is the interface point for all of the robots electrical systems. The power switch on the bottom left directly controls the power to the distribution block. The switch can handle up to 20 amps and illuminates when activated. To the right of the power switch a breadboard is mounted to the connector plate. This breadboard provides an incredibly useful and modular environment for interfacing sensors and creating circuits that will interact with the robot. These circuits and sensors can communicate with the robot through the interface board, which is flush mounted on the connector plate.

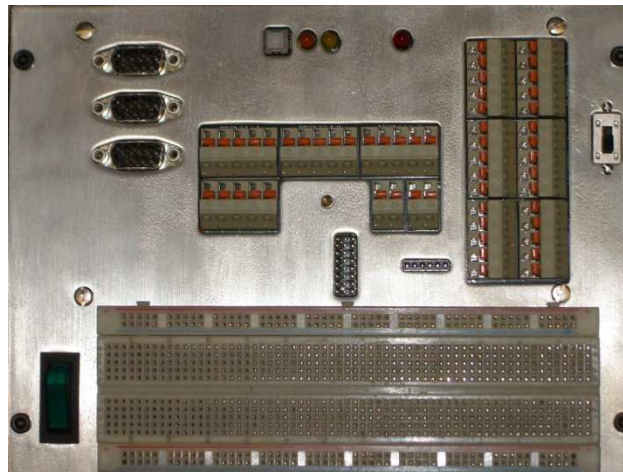


Figure 12 - Connector Plate

2.1 USER ACCESSIBILITY BOARDS

The TS-7800 board was able to satisfy all of our design requirements. In order to make the different input and output ports user friendly and easily accessible a two part circuit board was designed as an interface to the TS-7800. These boards not only made the features of the board accessible, they also off-loaded some of the tedious computing to a series of Atmel AVR microchips.

2.1.1 INTERFACE BOARD

The interface board is the first board in a two part design that creates easy to use modular electronics for the robot. It takes the ports on the 7800 and organizes them into easily accessible terminals for connecting components. The interface board holds the serial ports, the status LED's, the reset button, 15 digital input output ports, 5 analog to digital converter ports, 8 servo motor ports, 1 stepper motor port, 10 five volt terminals, 10 three volt terminals, and 10 ground terminals. A series of five ribbon cables connect the interface board to the TS-7800. The ribbon cables attach to various headers on the TS-7800 and plug into the corresponding headers on the interface board. The circuitry on the interface board sorts the scattered ports of the TS-7800 into user friendly terminals on the interface board, as well as provides protection circuits to prevent the processing board from getting accidentally destroyed.

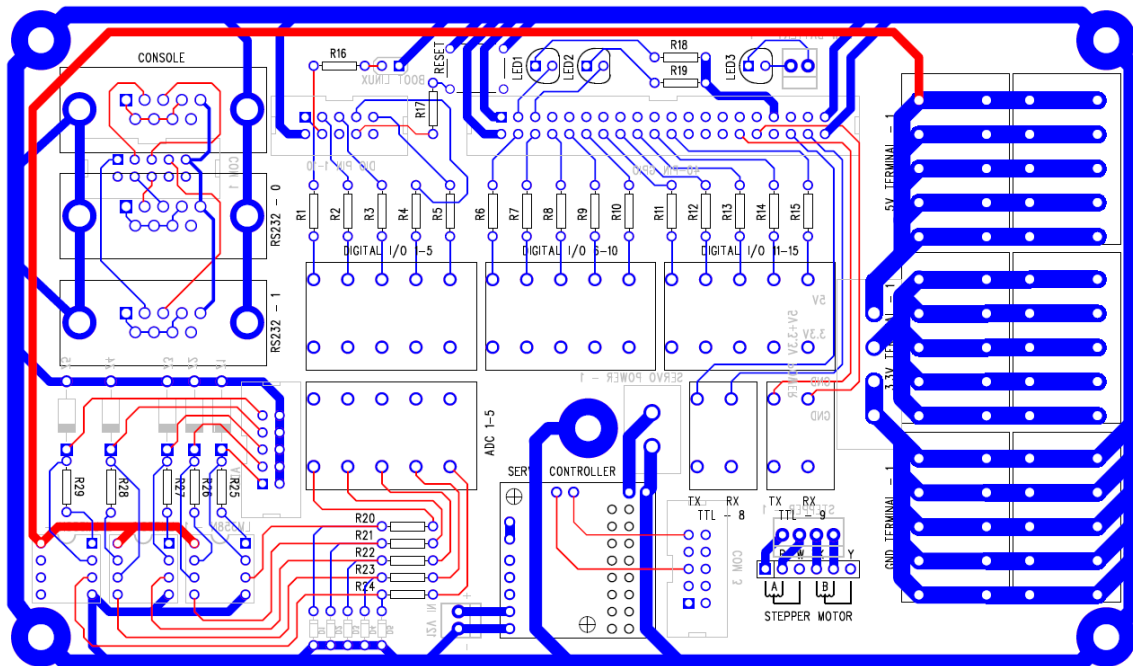


Figure 13 - Interface Board Schematic

2.1.2 DIGITAL I/O (DIO)

The ribbon cable coming off of the DIO header of the TS-7800 board is split into two cables. The first, of the two connects to pins 1-10 of the DIO header and plugs into the socket labeled "DIO PIN 1-10" on the interface board. The other cable connects to the co-processing & power regulation board and will be talked about later. When the cable comes into the interface board they are distributed as seen in Figure 14.

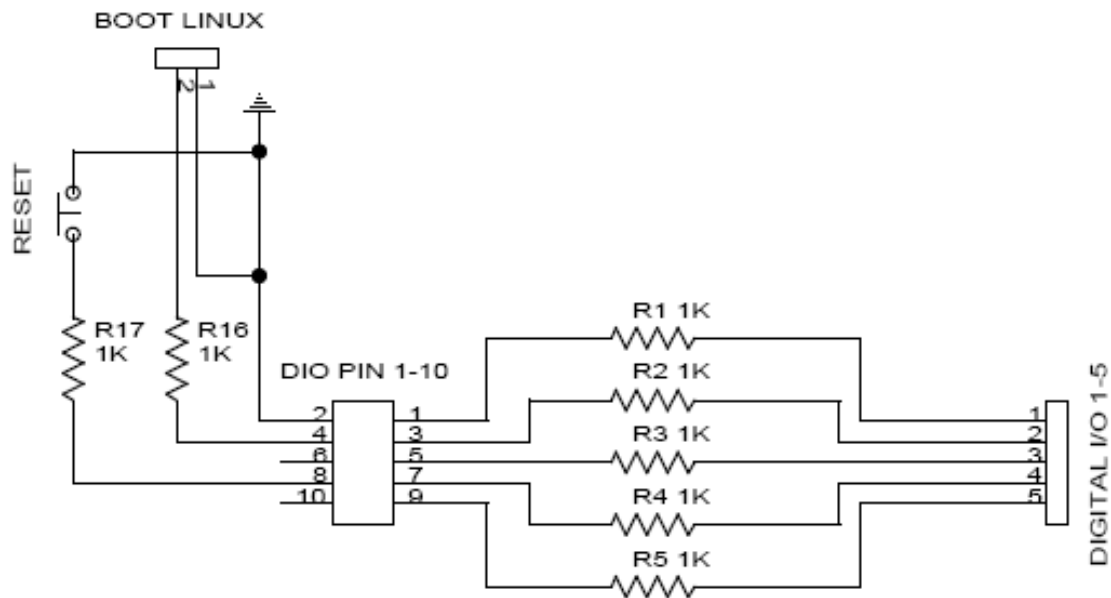


Figure 14 - DIO PIN 1-10 Schematic

All of the digital input and output pins are routed through a 1K ohm resistor to protect the TS-7800 from over current. The TS-7800 can only sink 8mA of current into the board. So to prevent the accidental discharge of more than 8mA into the board, we use a resistor to limit the current. We assume that the maximum voltage that can be supplied to the input is 5V, since only 5V and 3.3V terminals are accessible on the connector plate. We also decide to build in an error margin of 3mA leaving us with 5mA maximum current. Knowing these two values we can calculate the size of the resistor using:

$$V = IR$$

We find that a 1K ohm resistance is required to limit the current to 5mA. 1K ohm resistors are used on every non serial, digital or analog input that connects to the board.

Table 8 - DIO Pin Use

DIO PIN (TS-7800)	USE (Interface Board)	NOTE
1	DIO 1	1K ohm Resistor
2	GND	
3	DIO 2	1K ohm Resistor
4	BOOT TO LINUX	1K ohm Resistor
5	DIO 3	1K ohm Resistor
6		
7	DIO 4	1K ohm Resistor
8	RESET	1K ohm Resistor
9	DIO 5	1K ohm Resistor
10		

Table 9 - 40-PIN GPIO Pin Use

GPIO PIN (TS-7800)	USE (Interface Board)	NOTE	GPIO PIN (TS-7800)	USE (Interface Board)	NOTE
C0	GND		D0	GND	
C1	DIO 6	1K ohm Resistor	D1	LED 1	1K ohm Resistor
C2	DIO 7	1K ohm Resistor	D2	LED 2	1K ohm Resistor
C3	DIO 8	1K ohm Resistor	D3		
C4	DIO 9	1K ohm Resistor	D4		
C5	DIO 10	1K ohm Resistor	D5		
C6	DIO 11	1K ohm Resistor	D6		
C7	DIO 12	1K ohm Resistor	D7		
C8	DIO 13	1K ohm Resistor	D8		
C9	DIO 14	1K ohm Resistor	D9		
C10	DIO 15	1K ohm Resistor	D10		
C11			D11		
C12			D12		
C13			D13		
C14	TTL - 9_Tx		D14		
C15	TTL - 9_Rx		D15		
C16			D16	5V	
C17	TTL - 8_Tx		D17		
C18	TTL - 8_Rx		D18		
C19	GND		D19	GND	

2.2 SERIAL PORTS (RS-232)

There are 3 DB9 serial port connectors on the interface board. The top connector is the console port. This is the port used to directly connect the TS-7800 to another computer for programming and feedback. The next two are RS-232 serial ports. They allow the user to connect any serial RS-232 device for use with the robot.

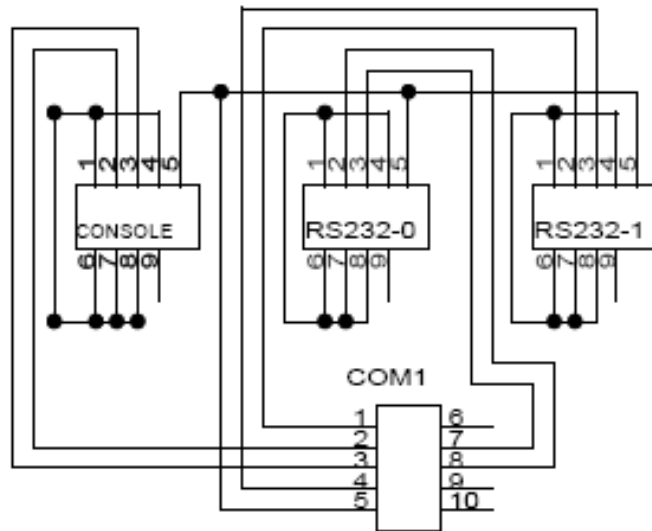


Figure 16 - RS-232 Ports Schematic

These ports are wired to the COM 1 socket on the back of the interface board, and that connects to the DB9 connector on COM 1 of the TS-7800. Pins 2 and 3 of COM 1 connect to the console port, pins 7 and 8 connect to the RS-232-0 port, and pins 1 and 4 connect to the RS-232-1 port. The standard wiring of a RS-232 male DB9 connector is as follows:

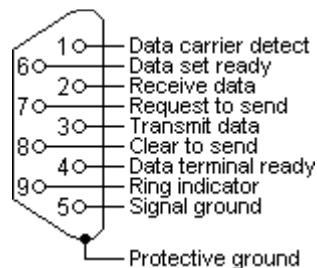


Figure 17 - RS232 DB9 Pin-Out

Pin 2 is the receive data pin (Rx) and pin 3 is the transmit data pin (Tx). Pin 2 on the COM 1 header is the Rx pin for the console, and pin 3 is the Tx pin. Therefore pin 2 and 3 on COM 1 goes to the corresponding pins 2 and 3 on the DB9 connector for the console. The Rx pin for UART #0 on COM 1 is pin 8, so it is wired to pin 2 on the DB9 connector for RS-232-0 and the Tx pin for UART #0 on COM 1 is pin 7, so it is wired to pin 3 on the DB9 connector for RS-232-0. The same concept applies for RS-232-1, pin 1 on COM 1 is the Rx pin so it connects to pin 2 on the DB9 connector for RS-232-1, and pin 4 on COM 1 is the Tx pin so it connects to pin 3 on the DB9 connector for RS-232-1. Pin 5 on all of the DB9 connectors is signal ground. These pins all connect to each other then tie to the signal ground on pin 5 of the COM 1 header. This ground is kept isolated from the board ground once again to prevent noise on the serial lines.

Table 10 - COM 1 Pin Use

COM 1 PIN (TS-7800)	DB9 Connector (Interface Board)	DB9 Connector Pin	Serial Signal (RS-232)
1	RS-232-1	2	Rx
2	CONSOLE	2	Rx
3	CONSOLE	3	Tx
4	RS-232-1	3	Tx
5	ALL	5	GND
6			
7	RS-232-0	3	Tx
8	RS-232-0	2	Rx
9			
10			

By using standard connectors for the RS-232 we create the opportunity for any number of standard serial devices to be interfaced with the robot. Because of the wide range of devices that can be connected to these ports we need to take into account the potential for a device that might require the use of the other 6 pins on a standard RS-232 DB9 connector. These pins are used when flow control is being used or the detection of a signal is required before the data can be sent. Since the TS-7800 only has the capability to communicate using the Tx and Rx ports for each serial connection, we can use a technique called “spoofing” to fool any device trying to use these pins. This is done by connecting pins 1, 4, 6, 7, and 8, on the DB9 connectors, together. What this does is creates a feedback loop that sends any signal coming from the serial device back to the device, where it is interpreted as a signal from the computer.

3 POLOLU SERVO CONTROLLER

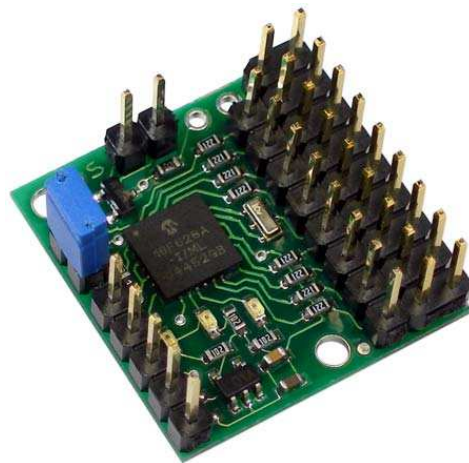


Figure 18 - Pololu Micro Serial Servo Controller

The Pololu servo controller is used to relieve the main processor of the tedious task of generating a PWM signal to drive the servos. This controller receives packets of information, through serial

communication, specifying the servo number and speed. This information is then converted into a PWM signal that drives the servo motor.

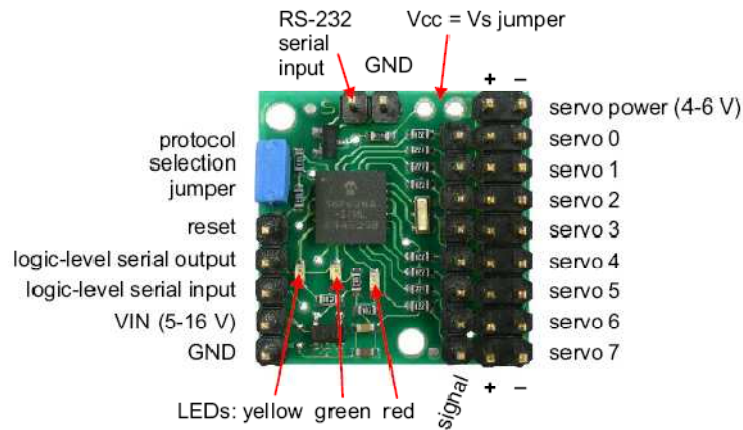


Figure 19 - Servo Controller Layout and Pin-Out

The interface board is designed so that the Pololu controller can be mounted directly to the board. A 12V power is connected to the interface board to power the servo controller. Because individual servos can draw at around 0.5A under normal use and have a stall torque of 1A, a separate 6 amp 5V power supply is connected to power the servo power rails.

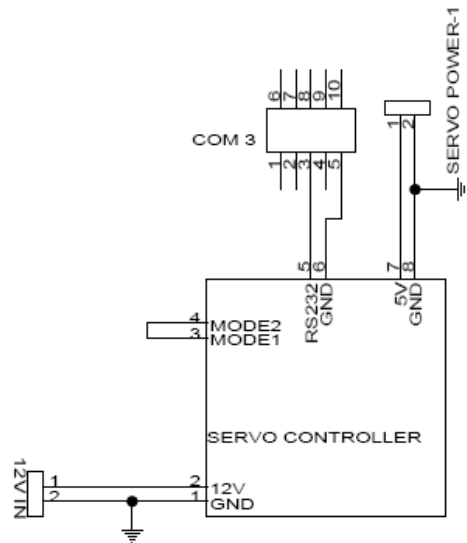


Figure 20 - Servo Controller Schematic

The ribbon cable from COM 3 on the TS-7800 connects to the COM 3 socket on the interface board. Pin 3, which is the transmit pin for the RS-232 UART #4, is connected to the RS-232 serial input of the servo controller. Pin 5 which is serial ground is connected to the controller as well but is kept isolated from to board ground to prevent noise on the serial line.

4 CO-PROCESSING & POWER REGULATION BOARD

The co-processing & power regulation board or CPRB is the second half of the user accessibility board design. The CPRB holds 6 reprogrammable ATmega168 AVR microcontrollers, a stepper motor driver, 3 power regulating circuits and a battery monitoring circuit.

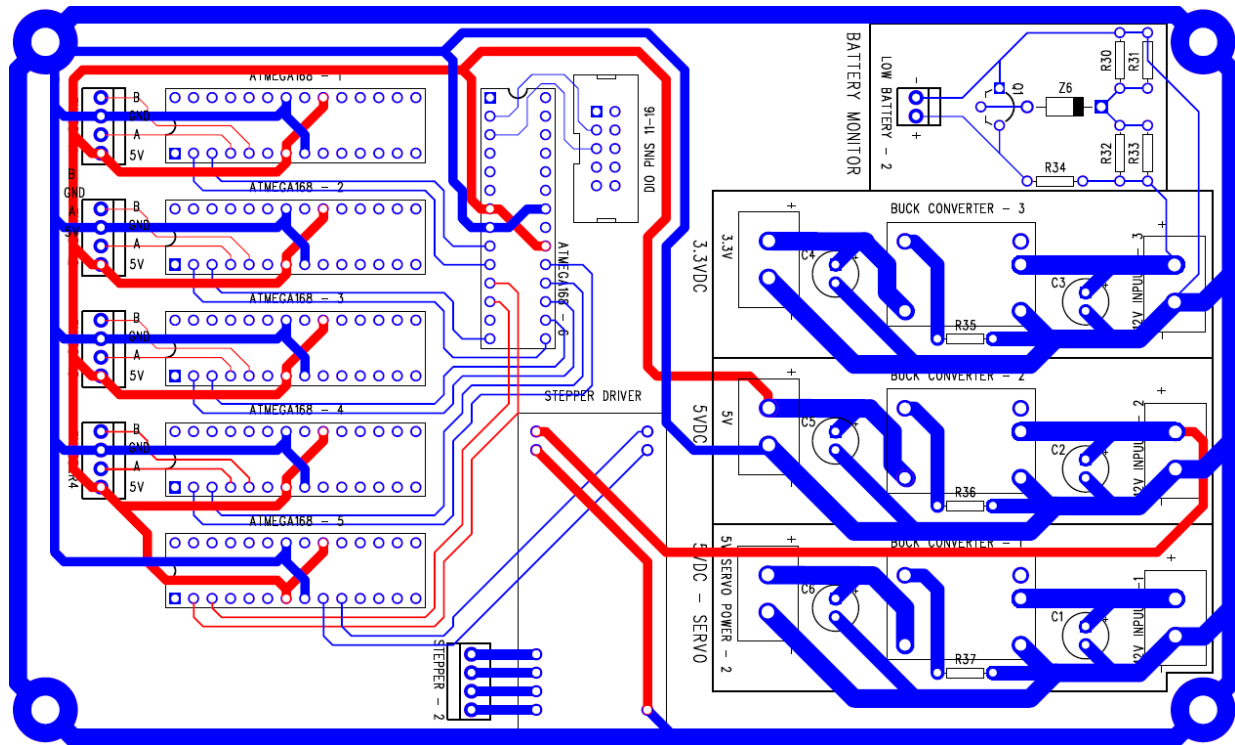


Figure 21 - Co-processing & Power Regulation Board Schematic

4.1 ATMEGA168 AVR MICROCONTROLLERS

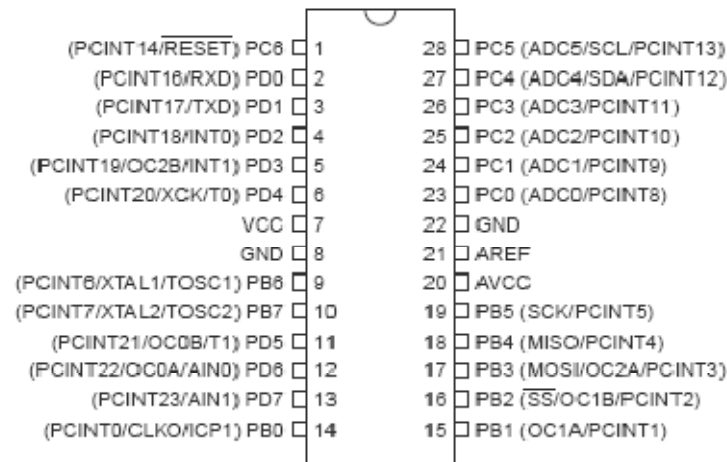


Figure 22 - ATmega168 Pin-Out

The ATmega168 chips were chosen as co-processing chips because they are fast, cheap and easy to work with. The AVR chips are widely used; therefore there is a lot of software already written for them. They are also programmed in C, simplifying programming even more.

4.2 BUCK CONVERTERS

The robot is supplied with 12V from the battery, but most sensors and servos are run on a voltage somewhere in the range of 0-5V, the most common being 3.3V and 5V. In order to reduce the 12V from the battery to a more common voltage type we used buck converters. Buck converters work with an inductor and two switches that control the inductor. It alternates between connecting the inductor to source voltage to store energy, and discharging the inductor into the load. After considering the current that could be drawn by sensors, as well as servos, we concluded that 3 buck converters would be necessary, each with a rating of 6 amps. 2 buck converters are set to 5V, one to power the servos, and the other to supply power to the 5V power terminals. The third buck converter is set to 3.3V to supply power to the 3.3V power terminals. Based on these requirements we choose the PTH12000W variable buck converter.



Figure 23 - PTH12000W Buck Converter

The buck converters are wired up as shown in Figure 24. Rset was set to 2K ohms for the 3.3V regulator and 270 ohms for the two 5V regulators. The inhibit pin was left open because it was not

needed for our application, 12V was supplied to V_{IN} and V_{OUT} was connected to its corresponding place on the interface board. The Capacitors are used to smooth the voltage output from the buck converter.

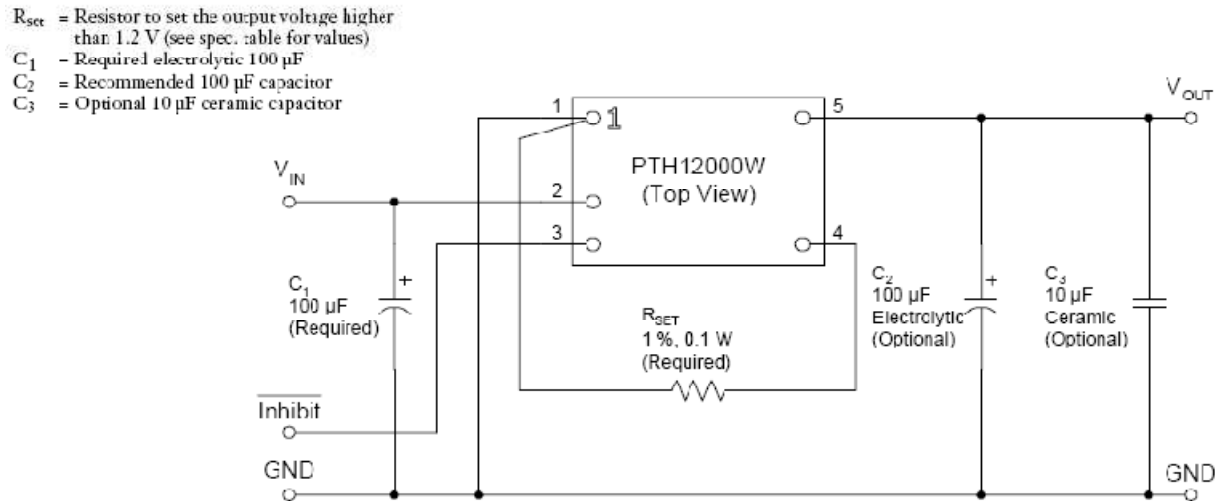


Figure 24 - Power Regulation Circuit

4.3 BATTERY MONITORING CIRCUIT

This circuit alerts the user when their battery is getting low by turning on an LED located on the interface board. The need for a battery monitoring circuit became evident when we were researching batteries, we found that the type that best suited our needs was a nickel-metal hydride (NiMH) battery, and with further research discovered that NiMH batteries can be damaged if they are drained below a certain voltage. So by designing a circuit to alert the user when the battery was getting low we reduce can reduce the chances of the user damaging the battery.

NiMH batteries are rated to a nominal voltage that they can steadily output for the majority of their discharge cycle. When the battery is fully charged it will output a voltage higher than its rated voltage output. The output voltage will slowly drop as the battery is discharged, it will follow a curve known as a discharge curve. By studying this curve the remaining battery capacity can be determined based on the voltage output.

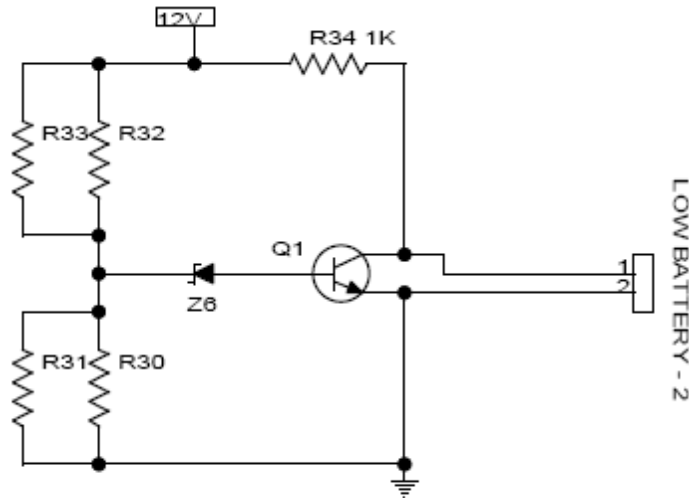


Figure 25 - Battery Monitor Schematic

Figure 25 shows the circuit we designed to illuminate an LED at a set voltage. The concept of this circuit is that while the input voltage is above the set trigger voltage the zener diode breaks down applying a high to the transistor which shorts across the LED, diverting the current to ground. As the input voltage lowers beyond the trigger voltage, there is not enough voltage supplied to the zener diode for it to break down therefore the transistor receives a low, disconnecting the short and sending current through the LED which is located on the interface board and connected to pin 1 and 2 through a cable.

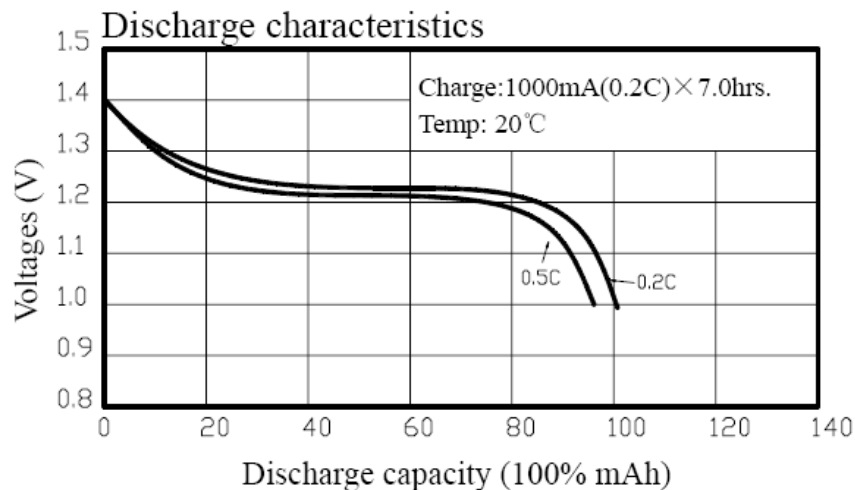


Figure 26 - Discharge Curve for MH-C5000 Cells

In order to determine where to set the trigger voltage we looked at the discharge curve for our selected battery (Figure 26). Since this graph is for each individual cell in the battery pack, we need to multiply the voltage and discharge rate by 10, because our battery pack contains 10 cells.

Assuming a standard discharge of 2Ah and deciding to drain the battery by approximately 90% we can see that the approximate output voltage of the battery at that point would be 11V.

Now that we know the trigger voltage is going to be set to 11V, we can calculate the resistor values need to use in order for the circuit to react to the desired trigger voltage. The zener diode we are using is 1N4735; it has a nominal breakdown voltage of 6.2V. The calculation for the resistors can now be done by a simple voltage divider calculation.

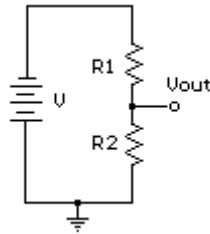


Figure 27 - Voltage Divider

Where $V = 11V$, $V_{out} = 6.2V$, and $R1 + R2 = 10K$.

We come up with $R1 = 4.4K$ ohms and $R2 = 5.6K$ ohms.

This is the ideal situation, but since the zener diode is not triggered exactly at 6.2V we used these numbers as a starting point and through trial and error determined a value of 3.4K ohm for R1 and 6K ohm for R2. We achieved this ratio using two 6.8K ohm resistors in parallel to make 3.4K ohms, and two 12K ohm resistors in parallel to make a 6K ohm value. With this resistance ratio the LED illuminates just as the voltage drops below 11V.

4.4 MOTOR CONTROLLERS

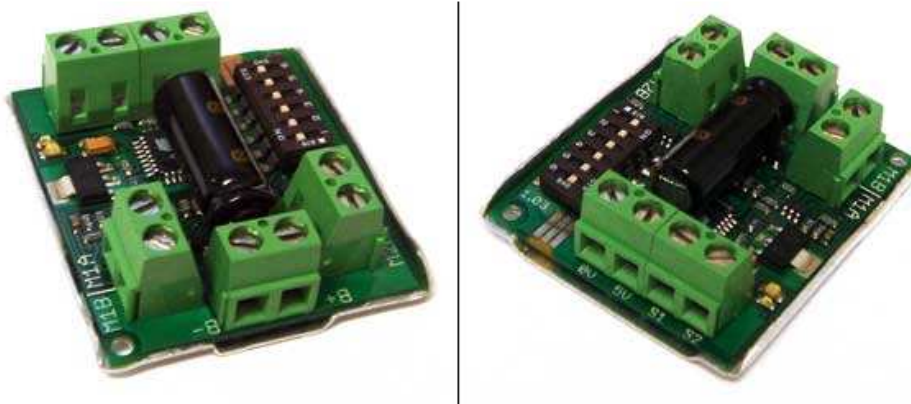


Figure 28 - Sabertooth Dual 5A Motor Driver

Two Sabertooth dual 5A motor drivers were used to control the 4 12V motors that drive the robot. The motors used on the robot had a stall torque of 3A so the 5A version of the Sabertooth satisfied our needs by supporting up to 5A per motor.

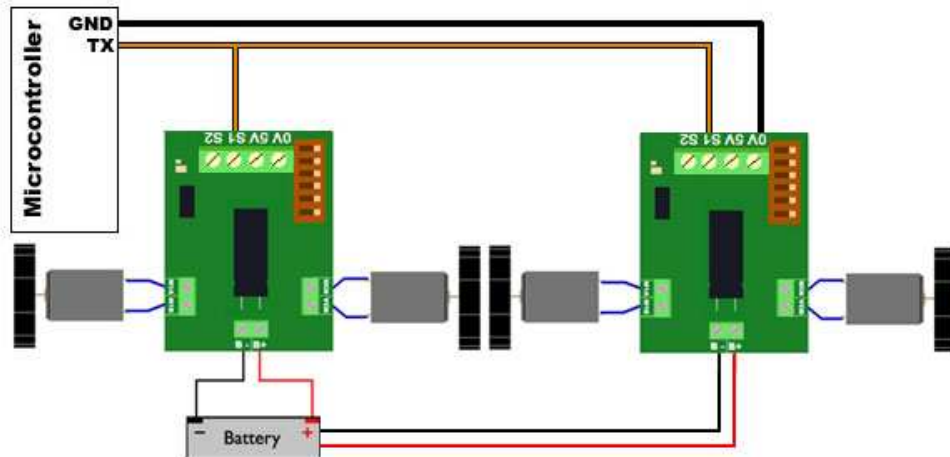


Figure 29 - Sabertooth Wiring Diagram

A positive 12V is connected to the B+ terminal and B- was connected to ground on each motor controller. The front two motors were wired to the M1 and M2 terminals of one Sabertooth, and the rear motors were wired to the M1 and M2 terminals of the other Sabertooth. By connecting the front motors to one and the rear motors to another, each motor controller had control of a right and left wheel. The 0V terminals on each Sabertooth controller are connected to ground on pin 2 of the TS-7800 LCD header, and the S1 terminals are tied to the TTL_TX_7 port on pin 13 of the LCD header.

Table 11 - Sabertooth Pin-Out

Terminal	Connected
B+	12V
B-	GND
M1A	Motor 1
M1B	Motor 1
M2A	Motor 2
M2B	Motor 2
0V	Pin 2 TS-7800 LCD Header
5V	
S1	Pin 13 TS-7800 LCD Header
S2	

Since the Sabertooth controllers both connect to the same serial port they receive commands, from the TS-7800 UART #7, in packetized serial form. Each controller is given an address that it can determine if the information it receives was intended for itself or the other motor controller. One controller is set to address 128 and the other is set to 129 by setting the dip switches as shown in Figure 30.

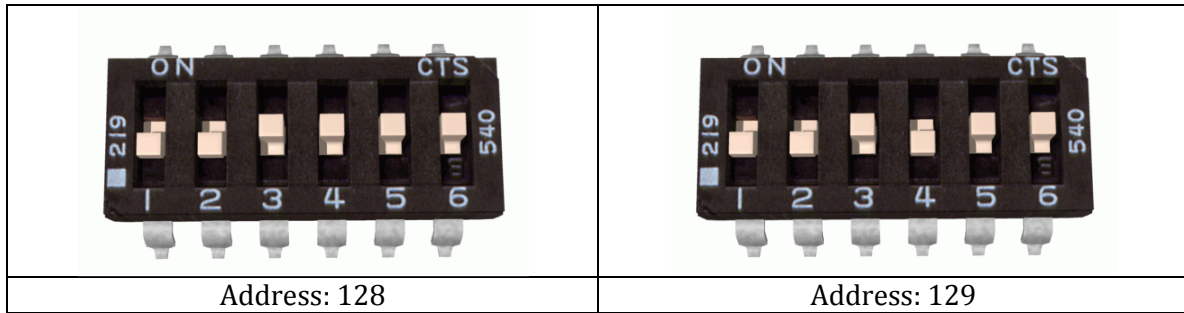


Figure 30: Sabertooth Dipswitch Settings

4.5 FUSE BLOCK

After the power from the battery is routed through the main power switch on the connector plate, it is sent to a fuse block. The fuse block is used to protect the electronic components and motors from stalling or shorting and drawing too much current. Trouble shooting shorts and other power problems on the robot is also made easier since each system of the robot has its own fuse.



Figure 31 - 10 Circuit Fuse Block

The fuse block used on the robot is shown in Figure 31, it accepts 10 ATC blade style fuses. It has a distribution block, for the positive side of the battery, of 10 fused terminals each rated for 30A, and a 14 terminal distribution block for the negative side. Table 12 shows the fuses used for each device.

Table 12 - Fuse Ratings

Device	Fuse Rating(A)
TS-7800	2
Sabertooth 1	7.5
Sabertooth 2	7.5
Servo Controller	1
Buck Converter 3.3V	5
Buck Converter 5V	5
Buck Converter 5V - Servo	5

5 SENSORS

To keep the cost of the platform down we decided not to integrate sensors into the base design of the robot. There is one exception to this and that is the wheel encoders, they were designed into the drive train. These are necessary to make the robot drive straight. Minor differences in the resistance of the gear boxes cause the wheels to spin at slightly different speeds, the encoders monitor the rotation of the output shaft and adjust the motor controllers to compensate. Another reason is the signal generated by the quadrature encoders would interrupt the processor 360 times for every revolution of the wheel, so we assigned the task of interpreting this signal to one of the AVR's.



Figure 32 - EP4 Optical Encoder

The robot contains 4 US Digital E4P OEM Miniature Optical Kit Encoder, one on the output shafts of each gearbox. We used this encoder because of its small size and cheap price. The encoders connect to headers on Co-processing & Power Regulation Board that supply it with 5V power from the buck converters, and connect the output signals on pins 2 and 4 to an AVR chip for processing.

Table 13 - EP4 Pin-Out

Pin	Description
1	+5VDC Power
2	A Channel
3	Ground
4	B Channel

6 BATTERY

When selecting a battery you must analyze the energy requirements for your system, and use it to define specifications for your battery. The battery voltage is required to be greater than or equal to the maximum voltage required by the robot. Looking at Table 14 we can see that the voltages required by the motors and electronics of the robot were no higher than 12V, therefore the selected battery had to have a voltage no lower than 12V, so in order to avoid using a transformer to lower the voltage the battery should be 12V. The next criterion in selecting the battery is its capacity. Capacity is given in amp hours; this means that the battery can produce the specified amperage for

a period of one hour. Again looking at Table 14 we see that the total current drawn by the robot, with 4 motors and 8 servos, running continuously under normal operating conditions is 5.63A. The ideal running time of the robot in this condition would be 50 minutes, or the time of one class period. Therefore a 5Ah battery will satisfy our current and time requirements.

Table 14 - Component Voltage and Current

Component	Voltage	Current - Normal (A)	Current -Max (A)
Motor 1	12V	0.7	3.5
Motor 2	12V	0.7	3.5
Motor 3	12V	0.7	3.5
Motor 4	12V	0.7	3.5
TS-7800	12V	0.8	0.8
Stepper Driver	12V	0.75	0.85
Servo Driver	12V	0.5	0.5
Motor Controller 1	12V	0.1	0.1
Motor Controller 2	12V	0.1	0.1
Servo 1	5V	0.16	0.9
Servo 2	5V	0.16	0.9
Servo 3	5V	0.16	0.9
Servo 4	5V	0.16	0.9
Servo 5	5V	0.16	0.9
Servo 6	5V	0.16	0.9
Servo 7	5V	0.16	0.9
Servo 8	5V	0.16	0.9
Total		5.63	20.05

The final decision that needs to be made in order to select the right battery for our robot was the type of battery. There are many different types of batteries; however, since we require a rechargeable battery we can narrow down the search to lead acid, lithium ion, nickel cadmium, and nickel metal hydride. Lead acid batteries can be ruled out because they are too big for our application. Since this platform will be used in a classroom environment we decide to avoid lithium ion batteries, due to their potential to explode if used incorrectly. And lastly we cross off nickel cadmium because they tend to have a high memory effect. This means that over time they tend to hold less and less charge. This leaves us with nickel metal hydride batteries. These batteries are relatively low cost, have good current output, long life and a high energy capacity. The only downside to nickel metal hydride batteries, is that they have higher charging times and have a self-discharge rate, meaning that if left for a period of time, they tend to lose their charge.



Figure 33 - 12V 5000mAh NiMH Battery Pack

Based on the criteria we selected a 12V 5000mAh NiMH Battery Pack from batteryspace.com. The battery pack contained 10 NiMH C size cells wired up side by side, with a 12V normal voltage output and a 14.5V max. The standard discharge rate is 5A but the battery is capable of a 10A continuous discharge. For this reason a 10A fuse was wired into the battery packs output terminal, to prevent the over-draw of current from the battery. The battery can accept a standard charge of 1.8A and has a maximum charge of 5A. Two terminals are wired into the battery pack, one for discharge and one for charging. The charging terminal has a built in temperature sensor to protect against overcharging, it disconnects the charger if the temperature of the battery rises over 70 degrees Celsius. Because the battery pack has two terminals the battery can be charged while the robot is still on. The only catch is that the charger will only output 1.8A. To protect against overdrawing the charger a 2A fuse was wired in to the charge terminal.

Appendix B

Mechanical and Software Design Details Reference

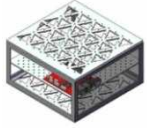

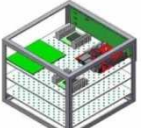


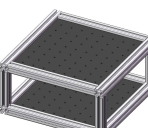
Much of the design details corresponding to the mechanical and software systems in the MRP are highly technical. For the sake of portability, these mechanical and software Technical Design Details for the Mobile Robotics Platform are included as a standalone entity in the pages that follow.

1.1 MECHANICAL SYSTEMS

1.1.1 *FRAME*

Being a modular robot we wanted to design the frame separately from the drive mechanism. The primary function of the frame is to provide mounting options for the electronics, the drive mechanism and for any peripheral devices such as arms or sensor arrays. In addition to mounting the frame also provides grounding for many of the electronics. The features of the frame light weight (under 3.5 pounds), easy to assemble by one person and low in cost all while being able to survive a fall from a 3ft tall table without permanent deformation. The initial designs are detailed in Table 1 - Frame Designs.

Table 1 - Frame Designs

	Square 1	Square 2	Square 3	Hexagonal	Octagonal	Extruded Al
Description	Short Square frame with horizontal mounting plates	Short Square frame with removable vertical mounting plates	Cube design with easily removable horizontal mounting plates	Hexagonal shaped design with both vertical and horizontal mounting plates	Octagonal shaped design with both vertical and horizontal mounting plates	A short Square frame with horizontal mounting plates
Picture						
Criteria						
Mass (Lbs)	3.15	3.3	3.88	2.74	2.8	3.1
Center of Gravity	(5, 5, 2.65in.)	(5, 5, 2.64in.)	(5, 5, 3.5in.)	(5, 5.1, 3.41in.)	(5, 5.09, 3.5in.)	(5, 5, 2.5in.)
Dimensions	10x10x5in. (lxwxh)	10x10x5in. (lxwxh)	10x10x8in. (lxwxh)	10x8in. (minor dia. x h)	10x8in. (minor dia. x h)	10x10x5in. (lxwxh)
Material Costs	\$80	\$90	\$90	\$120	\$150	\$100
Estimated Build Time and Cost	10 hrs. \$300	12 hrs. \$360	12 hrs \$360	18 hrs \$540	22 hrs \$660	4 hrs. \$120
Material Wasted	~20%	~20%	~20%	~50%	~50%	~5%
Ability to be Mass Manufactured	long cycle times (2hrs +), needs to be welded with precision	long cycle times (2hrs +), needs to be welded with precision	long cycle times (2hrs +), needs to be welded with precision	long cycle times (2hrs +), close tolerances	high cycle time (2+ hours), 50+ fasteners, close tolerances	Simple machining and finishing (cut to length, and tap 16 holes)
Expandability	Universal mounting bracket on top, standardized holes	Easy access removable plates	Universal mounting bracket on top, standardized holes	Universal mounting bracket on top, standardized holes	Universal mounting bracket on top, standardized holes	Item frame allows for multiple configurations
Standard Drive Train	Yes	Yes	Yes	No (adapters needed)	No (adapters needed)	Yes

The final design was a combination of the Extruded Aluminum, and Square 3. The construction material, item, similar to 8020, allows for a low build time and cost of material as well as making for

a very expandable design with channels for mounting located along the face of each piece of the frame. By combining the extruded design with a taller design close to a cube the internal volume is maximized relative to the use of material. However this raises concerns with the center of gravity that will have to be dealt with by mounting the heavy components as low as possible.

1.1.2 DRIVE MECHANISM

One of the basic functions of the MRP is the ability to move around under its own power. The first limitation we placed on the possibilities was the choice of using a DC electric motor because it would be the most familiar method of propulsion known to the potential students using the MRP. After making that selection there are several methods and configurations left to compare and determine the best solution. In Table 2 - Drive Mechanisms, each configuration is ranked one to six for the given criteria.

Table 2 - Drive Mechanisms

Criteria	One Steerable/ Driven Wheel*	Treads	Two Wheel Drive*	Four Wheel Drive	Six Wheel Drive	4 Wheel Drive Swerve
Example/ Description	A child's tricycle	Tank	Segway	2 wheels on each side driven together	3 wheels on each side driven together	Each wheel is independently driven and turned
Weight	1	4	1	3	4	6
Maneuverability	6	5	4	2	2	1
Ease of Use (complexity)	3	5	1	2	4	6
Handling of Rough Terrain	6	1	5	3	1	4
Cost	3	5	1	2	4	6
Manufacturability	4	5	1	2	3	6
Maintenance	1	6	1	1	1	5
Total	24	31	14	15	19	34
Average	3.43	4.43	2.00	2.14	2.71	4.86

* Note: Additional points of contact with driving surface may be used to increase stability.

Explanation of Criteria:

- Weight: the estimated mass of the drive system
- Maneuverability: the ability to make the robot go to a location and to have the desired orientation when it gets there
- Ease of use: the difficulty in controlling the robot in order to make a desired motion
- Cost: the estimated cost of materials and manufacturing time

- **Manufacturability:** a gauge of how easy it would be to mass produce MRP's with this particular drive mechanism
- **Maintenance:** how much work the end user has to do to keep the drive mechanism working

After comparing each possibility the two best designs are the two and four wheel drives. The main performance difference between the two designs is that with four wheel drive as long as wheels are contacting the ground the robot will be able to move towards the goal. With two wheel drive if one drive wheel loses traction the ability to move is greatly impaired. For this reason the four wheel drive solution was chosen as the drive configuration of the prototype.

1.1.3 SOFTWARE LIBRARY

The programming library is a key element towards making the MRP easily accessible. A user with a basic handle on programming languages should still be able to write a program that controls the MRP's hardware. Functions were made to control the basic hardware of robotics platform in terms that a basic user could understand. **Error! Reference source not found.** provides a brief overview of what the library consists of.

Table 3: Library Functions

Hardware Controlled	Description	Function Name
Digital Inputs and Outputs	Wrote high or low to digital outputs and returned high or low from digital inputs	setDO() and getDI()
Analog to Digital Converter	Contacted the TS-7800's Analog to Digital Converter and gets the value of a channel	getADC()
Servo Controller	Commands the servo controller to send a standard servo signal to one of 8 servos	setServo()
Motor Controllers	Contacts motor controllers and orders a specific motor to go a certain speed	setMotor()

1.1.3.1 Digital Inputs and Outputs

Digital inputs and outputs were controlled by specific registers. These registers contained 32 bit values, though they could also be read as 16 and 8 bit values. Reading these registers as 16 or 8 bit values means certain bits are ignored that may contain important information. Digital Input and Output pins hail from two areas of the board – 5 from the TS-7800's DIO Header, and 10 from a row

of its PC104 header. Each group of pins had its own set of registers and was handled different depending on the function required.

1.1.3.1.1 PeekPoke.c (need to talk to Ciaraldi to properly explain this)

Both digital inputs and outputs require the ability to read and write to memory registers. All functions that require these capabilities call functions within *peekpoke.c*. Written by one of this project's advisors, Professor Michael Ciaraldi, for another project, *peekpoke.c* will read and write to registers in hardware memory and was vital to this project's success.

NOTE: I need to meet with professor Ciaraldi, as this is based on his code. I believe I know how this works, but I want to make sure I 100% understand this before I write more to this section.

1.1.3.1.2 Digital Inputs

By calling the function `getDO()`, the user receives either a 0 or a 1 to represent the status of that digital input. The program controlling this function call is `DOhandler.c`, which controlled how to return the value based on the several considerations that had to take place.

The pins chosen to be brought out to the breakout board did not progress regularly – the first 5 even numbered pins on the DIO header were chosen, then an almost progressive set of odd numbered pins on the DIO header were selected. The reason for skipping pins is due to the pins being dedicated through hardware to other uses. To make the pin numbering easier for users, an easier 1 through 15 pin numbering was assigned. Since the average user is unaware of the unusual pin numbers in hardware, a simple program and function was created to return what bit in a register represents the selected pin. This function, `getBit()`, is in `getBit.c`:

```
int getBit(int pin){
    switch(pin){
        case 1: return 0;
        case 2: return 2;
        case 3: return 4;
        case 4: return 6;
        case 5: return 8;
        case 6: return 1;
        case 7: return 2;
        case 8: return 3;
        case 9: return 4;
        case 10: return 5;
        case 11: return 6;
        case 12: return 7;
        case 13: return 8;
        case 14: return 9;
        case 15: return 10;
    }
}
```


getBit.c consists of a switch statement that merely returns the corresponding bit of the pin requested. It is shared between DOhandler.c and DIhandler.c

If the pin being selected was in the first group of 5 pins, the register that needed to be written to is 0xE8000004. By reading this register, the program merely needs to isolate the required bit and return it. In order to do this, the following code from getDI() is used:

```
int getDI(int pin){
    int bit;
    unsigned int bitmask;
    unsigned int readvalue;

    if(pin < 1 || pin > 15) return 0; //error check

    //We must make sure that input pins are set high each time we
    //read them for DIO HEADER pins only.
    // This is because they are open-drain.
    if(pin <= 5) setDO(pin, 1);

    //Likewise, for GPIO pins we must set the pin as an input.
    else if(pin >= 6) set2Input(pin);

    bit = getBit(pin);
    bitmask = 1 << bit;
    //Set appropriate bit in bitmask - shift over bit bits

    if(pin <= 5) readvalue = peek16(DIOHEADERREAD);
    else if(pin >= 6) readvalue = peek16(GPIODATA);

    readvalue = readvalue & bitmask;
    //Isolate the bit we want
    readvalue = readvalue >> bit;
    //shift to right to force return value to 0 or 1

    return readvalue;
}
```

Once the register is read, its value then goes through a number of bitwise operations. First the function creates a value bitmask, which is a 1 bit shifted bit bits high, where bit is the location of the bit we are interested in. The register value is bitwise anded with bitmask, setting everything to 0 except potentially the value in the desired bit location. The register value is now bitshifted bit bits to get the first bit. What was once the register value read is now the desired bit's real value – a 0 or a 1.

The remaining 10 digital IO pins belong to the PC104 header and belong to a different register. Before reading the register, however, we must first check to see if the pin's corresponding bit is set as an input before reading. This invokes a function inside of DIhandler.c, set2Input():

```

void set2Input(int pin){
    int bit;
    unsigned int bitmask;
    unsigned int writevalue;
    int temp, read;
    int mask;

    bit = getBit(pin);

    read = peek16(GPIODIRECTION);

    // Create a mask with all 1, except the desired bit = 0;
    mask = 1;
    mask = mask << bit;
    mask = ~mask;

    // Force desired bit to 0; leave the rest unchanged.
    writevalue = read & mask;

    // Set the desired bit = 0
    temp = 0;
    temp = temp << bit;
    writevalue = temp | writevalue;

    poke16(GPIODIRECTION, writevalue);
}

```

To set the requested pin as an input, the corresponding bit on 0xE800002C for the pin must be set to 0. Once the direction is set as an input, the program then reads the register 0xE800001C. The code for set2Input() is similar to setDO(), which is explained in the following subsection.

1.1.3.1.3 Digital Outputs

Digital outputs are handled similarly to digital inputs, and like digital inputs, the DIO header pins and PC104 header pins have to be handled differently for them to work. DIO header pins required the least overhead, merely requiring flipping an individual bit on their register. The register for writing out to the DIO header is 0xE8000008, different from the input register. The register is read and preserved to prevent other pin's values from inadvertently being changed when attempting to change the value of a single pin.

```

void setDO(int pin, int value){
    int bit;
    unsigned int bitmask;
    unsigned int writevalue;
    int temp, read;
    int mask;

    if(pin < 1 || pin > 15) return;
    if(value < 0 || value > 1) return;

    bit = getBit(pin); // Which bit in the register goes with this pin

    if(pin >= 6) set2Output(pin); // Set direction to output

    if(pin <= 5) read = peek16(DIOHEADERREAD);
    else read = peek16(GPIODATA);

    // Create a mask with all 1, except the desired bit = 0;
    mask = 1;
    mask = mask << bit;
    mask = ~mask;

    // Force desired bit to 0; leave the rest unchanged.
    writevalue = read & mask;

    // Set the desired bit = value
    temp = value;
    temp = temp << bit;
    writevalue = temp | writevalue;

    //Now write to the appropriate place.

    if(pin <= 5) poke16(DIOHEADERWRITE, writevalue);
    else poke16(GPIODATA, writevalue);

}

```

Much like digital input handling in `getDI()`, we create a bitmask. In this case, however, *mask*, our bitmask, is inverted to be all 1's save the position of our desired bit. Then *mask* is anded with the register value, forcing the bit we want to 0. The function then create an additional mask then, *temp*, which is a value of all 0's save a single 1 in the desired bit location. *Temp* is bitwise ored with the modified register value to set the desired bit to the value we wanted.

The digital output pins 6 through 15 originate from the PC104 header pins. The register for these pins is the same as the one used for the inputs. The direction register, 0xE800002C, however, must have the corresponding bit for that pin be set to 1 for the pin to be successfully written to. This is

done via `set2Output()`, who's code is identical to `set2Input()`, save the change in registers read and written to.

1.1.3.2 Servos and Motors

While controlled by different hardware systems, both motors and servo motors are controlled in very similar ways. *setMotor.c* controls both servos and regular motors through different functions. Their combination was warranted by their similar nature – both require a proper serial connection to be opened such that a command can be sent.

1.1.3.2.1 Serial

TBD

1.1.3.2.2 Servos

In order to set a servo's position or speed, *setServo()* is called. The parameters that are passed to *setServo* are the servo number (one through eight), the position or speed (the ranging depending upon the duplex), and half or full duplex (0 is half, while 1 is full).

```
write(fd, &startBit, 1);  
write(fd, &servo, 1);  
write(fd, &pos, 1);
```

Writing to the servos requires first sending a “start byte”. The servo controller will not accept commands until you first send it 0xFF (255). After this start byte is sent, *setServo()* will send the servo being called and its position. If a servo is being called to work in full duplex mode, *setServo()* will add 8 to its address to notify the servo controller.

1.1.3.2.3 Motors

Motor controllers also use a serial connection in order to convey the desired control over motors. Sabertooth motor controllers, however, can be daisy-chained, allowing a single serial port to control up to eight separate controllers, or sixteen separate motors. For the purposes of the library, it is assumed that the user has no more than two motor controllers, though more can easily be accounted for.

```
write(fd, &baud, 1);  
write(fd, &address, 1);  
write(fd, &command, 1);  
write(fd, &speed, 1);  
write(fd, &checksum, 1);
```

The motor controller accepts commands in a certain format. Each command is sent via a *write()* command. Commands must be sent in a certain order for the motor controller to understand the request. The commands sent are, in order:

1. A baud-check. This is a known value issued by Dimension Engineering, the motor controller's manufacturer. The defined bit, 0xAA (170 decimal). This notifies the motor controller what baud rate the MRP will communicate to it with.
2. The address is the motor controller being controlled. There can be up to eight motor controllers on a single serial line, so eight different possible addresses from 127 to 135. The address is set physically on the motor controllers via a DIP switch.
3. The motor controller has 14 possible commands, 0 to 13, for controlling its two motors. These commands change how one addresses the motor controller and controls the MRP's motors. For simplicity's sake the library always uses commands 6 or 7 (depends on which motor you are addressing). These commands drive a single motor with 7 bit accuracy (0 to 127). Additional commands can provide better control and feedback for the MRP, but make controlling the MRP for a new user more difficult. Future renditions of the library should include functions that take advantage of these commands.
4. The speed of the motor is the next value sent. The range of speeds is 0 to 127, with 64 being a rest speed, 0 being full reverse, and 127 being full forward.
5. A final checksum value is submitted last. This is a value generated by the previous pieces of for the motor controller to confirm a successful transfer of information. The checksum is the value of the address, the command, and the speed added together. This value is then bitwise anded with 0x7f (127) to generate the checksum. If the checksum fails or is wrong, the motor controller will not respond and ignore the MRP until a correct checksum is submitted.

1.1.3.3 Internet Connection

1.1.3.3.1 Wired Internet

1.1.3.3.1.1 Getting Connected

Since Debian Linux was used, network configuration was automatic. The board's MAC address was properly reported to WPI's Network Operations Center and, once approved to be on the network, connected without a problem. Merely plugging the Ethernet into the board quickly connects the MRP to available networks.

NOTE: I seem to not have much to say in this area. Professor Ciaraldi – what can I add here?

1.1.3.3.2 Wireless Internet

NOTE: As of now, even after the best efforts of Professor Ciaraldi, myself, and the Free Software Association of WPI (new group on campus), we still can't get the Wi-Fi to work. I have one last idea I still need to test. If this doesn't work, should I just write about my attempts albeit failure?

1.1.3.3.3 Remote Connections

Once an internet or network connection is established, the ability to remotely connect to the MRP becomes a possibility. Two protocols become available, each for suited for distinct goals.

Secure Shell, or SSH, is a connectivity protocol that allows users to remotely connect to a system. Once connected, their user commands are treated as if entered on the host system. This connect type is equivalent to being directly connected to the MRP via a serial cable. The MRP can be completely controlled by a user remotely connected through SSH.

Linux users who wish to connect to the MRP using SSH usually need only use the command

```
ssh username@XXX.XXX.X.XX
```

and then, when prompted, enter their password. Windows and Mac users can use the freely available program Putty in order to connect to the MRP via SSH.

Secure File Transfer Protocol, or SFTP, was another protocol commonly used in the project. SFTP is typically used for downloading and uploading files to and from the MRP. This is useful for backing up important files or uploading new code to the robot. Future uses could entail downloading and receiving sensor feedback from the MRP.

Many programs exist in order to connect to the MRP through SFTP. Linux users again merely need to use their command line interface. Using the command

```
sftp username@XXX.XXX.X.XX
```

will prompt for a password and then connect the user to the MRP. Windows, Linux, and Macs have numerous freely available programs that will support SFTP. Many of these programs boast graphical user interfaces and are very accessible to users of all levels.

1.1.3.4 Serial Connection

The serial connection is defaulted on the operating system to always output to COM1. Several programs exist in order to establish a connection to the MRP through serial. Linux users may use the terminal based Minicom. Minicom treats the user's Linux terminal as a terminal based on the host machine. A graphical version of Minicom is available, called CuteCom. CuteCom features the same capabilities, but for long term use, Minicom had noticeably fewer issues during this project's use. Windows users have HyperTerminal, a program that comes with most versions of Windows by default. Windows and Mac users may also use Putty in order to connect to the MRP through serial.

To connect to the MRP through serial, specific settings are required to be set in each program. These settings are:

- 115200 bits per second baud rate
- 8 data bits with no parity (commonly called 8N1)
- No flow control
- 1 stop bit

1.1.3.5 Operating System

1.1.3.5.1 Operating System Upgrade

During the project, Technologic Systems issued an upgrade to the recommended operating system. They issued a newer version of Debian Linux, recompiled to work with the Journaling File System (JFS) instead of the Network File System (NFS). JFS works better in embedded and portable applications as it deals with sudden power loss better. This makes it more ideal for the MRP.

1.1.3.5.2 Internet Switch

The MRP's expected environment of operation is twofold – it is not unexpected to find it placed in a laboratory complete with wireless network. Likewise, it is expected that the MRP will be used at some point out of reach of wireless networks. When turning on the MRP, the time to reach a state in which code can be executed can be drastically different depending on the variables.

Table 4: Boot-up Times

Description	Time to Boot
The MRP is in a lab with the network connected and it attempts to connect	1 minute, 45 seconds
The MRP is in an area with no network and it attempts to connect	1 minute
The MRP skips all network steps	30 seconds

The table demonstrates a very noticeable difference in boot time between all the possibilities. An “internet switch” was included on the MRP. Early in the boot sequence, the script *internetOnOff* is executed. This script checks the state of the internet switch. If on, it forcibly creates soft links (creates them over ones that may already be there) to start up scripts throughout */etc/rcS.d/* and */etc/rc3.d/*. These are the folders that contain, in order, boot up scripts for the MRP's OS. If the switch is off, indicating that no network connection will be needed, the script will forcibly (do it even if they are not there) destroy these soft links. This will prevent the operating system from even attempting a network connection, drastically improving the boot up time of the MRP.

1.1.3.5.3 “Boot-to-Code/Kill” Switch

The MRP is designed to be used both in and away from an equipped laboratory. To allow users to activate their code while away from the library, as well as end the code when necessary, two features were added that are closely dependent. The first to be activated is a script, *resetButton*, first checks to see if a jumper is set to high or low. If it is set high, *resetButton* executes code in a known file in a known folder – in this case, the file *runMe* in the folder */CODE*. When the code is ran, a file in */var/run* labeled *studentCode.pid* is created. This file is filled with the process identification (or PID) of the program being ran. By reading *studentCode.pid*, one can find the PID of the student code. This allows a program to terminate the process, suspend it, monitor it, or directly communicate with it.

resetButton also creates a daemon process, or process that runs in the background forever. The process merely checks the MRP's reset button routinely. If it detects a press, it will execute the command:

kill -9 'cat /var/run/studentCode.pid'

The command *kill -9* will terminate any process running with the following PID. The command *'cat /var/run/studentCode.pid'* will return to that section of the command the contents of *studentCode.pid*, in which *resetButton* has placed the PID of the user's code. The result is the termination of any code that was ran through *resetButton*.

If the jumper was not set to boot the MRP to code, or the code was terminated at some point by the reset button, there is still a way to execute code without access to a computer. When the reset button is pressed, *resetButton* first checks to see if there is an existing process to be killed. If no process exists, then it will execute */CODE/runMe*, recreating the PID file.

A proper and safe termination of code, however, does not stop at merely ending the process. All outputs – be they digital, servos, or motors – will continue to execute the last known command even when the process is terminated. A function was created, *resetAll()*, which would safely reset each of these systems. All digital inputs would be set to inputs. The servo controller would receive a “reset” signal that halts all servos. Motors are set to stop to prevent the robot from colliding with anything.

Appendix C

Budget Summary

An approximate budget summary of the entire course of the project is summarized in the following pages. Figures in red indicate going over budget.

Part Description	Price Ea.	Q	Cost	#Used	Platform Cost
Electronics					
Processor					
500MHz ARM9 SBC with 128MB DDR-RAM and 512MB Flash	\$269.00	1	\$269.00	1	\$269.00
Battery Backed Real Time Clock	\$10.00	1	\$10.00	1	\$10.00
On-Board Temperature Sensor	\$3.00	1	\$3.00	1	\$3.00
RS-232 or RS-485/422 Full or Half Duplex on COM 2	\$14.00	1	\$14.00	1	\$14.00
64 and 40 Pin (16-bit) Stack-Thru Connectors	\$25.00	1	\$25.00	1	\$25.00
8-30VDC on-board switching-mode power regulator	\$28.00	1	\$28.00	1	\$28.00
USB 802.11g wireless network interface	\$35.00	1	\$35.00	1	\$35.00
18VDC wall mounted regulated power supply	\$22.00	1	\$22.00	1	\$22.00
Double-headed null modem cable with DB25F & DB9F at each end	\$9.00	1	\$9.00	1	\$9.00
10 Pin header to DB9M ribbon cable	\$4.00	1	\$4.00	1	\$4.00
Video Board for TS-7000 series	\$99.00	1	\$99.00	0	\$0.00
subtotal:	\$518.00				\$419.00
Computer Expansions/Peripherals					
Sandisk 8GB Ultra SD Card	\$89.98	1	\$89.98	1	\$89.98
ATtiny45 4kB Flash, 0.256kB, Atmel Microcontrollers – RISC	\$2.13	5	\$10.65	0	\$0.00
ATmega168 16kB Flash, 0.5kB, Atmel Microcontrollers – RISC	\$3.25	#	\$32.50	6	\$19.50
subtotal:	\$133.13				\$109.48
Motor/Servo Controllers					
Sabertooth dual 5A motor driver	\$59.99	2	\$119.98	2	\$119.98
EasyDriver v3 Stepper Motor Driver	\$14.95	1	\$14.95	1	\$14.95
#208 Pololu micro serial servo controller (partial kit)	\$17.95	2	\$35.90	1	\$17.95
Resistors	\$0.05	#	\$1.85	3	\$0.15
subtotal:	\$172.68				\$153.03
Encoders					
CONN HEADER 4POS 1.25MM VERT TIN	\$0.71	4	\$2.84	4	\$2.84
CONN HOUSING 4POS 1.25MM	\$0.37	8	\$2.96	8	\$2.96
CONN TERM SOCKET CRIMP 28-32AWG	\$0.12	#	\$4.80	40	\$4.80
Quadrature Encoder	\$22.95	4	\$91.80	4	\$91.80
subtotal:	\$102.40				\$102.40
Hardware					
12V 5000mAh NiMH Battery Pack	\$54.95	1	\$54.95	1	\$54.95
subtotal:	\$54.95				\$54.95
Breakout Connector Board					
Bill of materials (BOM)	\$0.00	0	\$0.00	0	\$0.00
terminal block 5mm 2pos pcb	\$0.90	1	\$0.90	0	\$0.00
terminal block 5mm 2pos pcb	\$0.83	1	\$0.83	0	\$0.00
terminal block 5mm 2pos pcb	\$0.90	1	\$0.90	0	\$0.00
terminal block 5.08mm 2pos pcb	\$0.96	1	\$0.96	3	\$2.88
terminal block 5mm 2pos pcb	\$0.90	1	\$0.90	0	\$0.00
terminal block 5mm 2pos pcb	\$0.74	1	\$0.74	0	\$0.00
terminal block 5mm 2pos pcb	\$0.76	1	\$0.76	0	\$0.00
term block 5mm vert/hor 2pos pcb	\$0.34	1	\$0.34	0	\$0.00
pin header 02 pos tin 5mm	\$0.36	1	\$0.36	0	\$0.00
term block hdr 3.81mm 2pos pcb	\$0.62	1	\$0.62	0	\$0.00
term block plug 3.81mm 2pos pcb	\$1.11	1	\$1.11	0	\$0.00
term block hdr	\$1.85	1	\$1.85	0	\$0.00
term block plug 5.08mm 2pos pcb	\$2.47	1	\$2.47	0	\$0.00
conn db 9 female gold metal shell	\$4.65	1	\$4.65	0	\$0.00
conn db9 female au shell lo pro	\$5.08	1	\$5.08	1	\$5.08
conn d-sub plug vert 9pos	\$7.47	4	\$29.88	3	\$22.41
conn d-sub plug str 9pos pcb au	\$7.05	1	\$7.05	0	\$0.00
conn d-sub plug str 9pos pcb au	\$6.27	1	\$6.27	0	\$0.00
conn d-sub rcpt vert 9pos	\$9.75	1	\$9.75	0	\$0.00
conn header lopro str 10pos 30au	\$0.94	6	\$5.64	5	\$4.70
conn header lopro str 40pos 30au	\$3.39	2	\$6.78	1	\$3.39
conn header lopro str 16pos 30au	\$1.62	1	\$1.62	0	\$0.00
conn header lopro str 14pos 30au	\$1.47	1	\$1.47	0	\$0.00
switch rotary dip bcd top adj	\$2.88	1	\$2.88	0	\$0.00
switch dip top slide 4pos	\$1.78	1	\$1.78	2	\$3.56
switch dip top slide ext 3pos	\$1.16	1	\$1.16	0	\$0.00
switch bcd rotary dip thruhole	\$3.01	1	\$3.01	0	\$0.00
cap black window version 94 ser	\$0.76	1	\$0.76	0	\$0.00
switch push spst mom 100ma 14vdc	\$1.92	2	\$3.84	1	\$1.92
lens for t1 3/4 LED green dome	\$0.49	1	\$0.49	0	\$0.00

Part Description	Price Ea.	Q	Cost	#Used	Platform Cost
LED super brite grn t1-3/4 vert	\$0.90	1	\$0.90	0	\$0.00
TERMINAL BLOCK 5.08MM 5POS PCB	\$2.15	#	\$21.50	10	\$21.50
IC SOCKET 8PIN LOW PROFILE .300	\$0.86	3	\$2.58	3	\$2.58
IC SOCKET 28PIN LOW PROFILE .300	\$2.80	6	\$16.80	6	\$16.80
CONN HEADER VERT 4POS .100 TIN	\$0.31	6	\$1.86	6	\$1.86
POLOLU SERVO CONTROLLER	\$17.95	1	\$17.95	1	\$17.95
CONN HEADER FEMALE 6POS .1" TIN	\$0.55	1	\$0.55	1	\$0.55
LED	\$0.05	3	\$0.15	3	\$0.15
CONN HEADER 2 POS TIN PCB	\$0.62	7	\$4.34	7	\$4.34
CONN HEADER 4 POS TIN PCB	\$0.93	1	\$0.93	1	\$0.93
Transistor	\$0.05	1	\$0.05	1	\$0.05
Zener Diodes	\$0.05	6	\$0.30	6	\$0.30
75V Diode	\$0.05	5	\$0.25	5	\$0.25
Printed Circuit Board Manufacturing	\$137.75	1	\$137.75	1	\$137.75
subtotal:			\$310.76		\$248.95
Auxiliary Control Board					
Bill of materials (BOM)	\$0.00	0	\$0.00	0	\$0.00
Buck Converter	\$11.50	3	\$34.50	3	\$34.50
subtotal:			\$34.50		\$34.50
Miscellaneous Shop Parts					
Regular 9V Battery	\$2.00	1	\$2.00	0	\$0.00
9V battery clip	\$0.25	1	\$0.25	0	\$0.00
1uF ceramic capacitor	\$0.25	#	\$3.00	0	\$0.00
10uF electrolytic capacitor 25V-63V	\$0.70	3	\$2.10	0	\$0.00
male header connector, 2x3 pins	\$0.25	2	\$0.50	0	\$0.00
header connector, female, 2x20, long pin	\$0.75	1	\$0.75	0	\$0.00
header connector, female, 2x32, long pin	\$1.00	1	\$1.00	0	\$0.00
male header connector, 2x4 pins	\$0.30	1	\$0.30	0	\$0.00
header connector, 2x6 male pins	\$0.35	1	\$0.35	0	\$0.00
RC connector, 2x20 female	\$0.50	2	\$1.00	0	\$0.00
RC connector, 2x32 female	\$1.00	2	\$2.00	0	\$0.00
7-segment LED display	\$1.00	4	\$4.00	0	\$0.00
small protoboard, Jameco #105099	\$4.75	2	\$9.50	0	\$0.00
large protoboard, Jameco #105152	\$9.00	1	\$9.00	0	\$0.00
medium protoboard, Jameco #28177	\$5.50	2	\$11.00	0	\$0.00
Cables(Est)	\$10.00	1	\$10.00	0	\$0.00
subtotal:			\$56.75		\$0.00
Electronics subtotal:			\$1,383.17		\$1,122.31
Mechanical					
Wheels					
2-pack Axial 2.2 Rockster Beadlocks (Chrome)	\$24.95	2	\$49.90	0	\$0.00
2-pack Dirt Paw Tire – Fits 2.2" truck front or rear	\$14.95	1	\$14.95	0	\$0.00
2-pack Speed Hawg Tire – Fits 2.2" truck front or rear	\$18.95	1	\$18.95	0	\$0.00
Vintage racing tire 26mm D Compound (2pcs)	\$12.95	1	\$12.95	0	\$0.00
2-pack TE27 Wheel 26mm CHROME (6mm offset)	\$8.95	1	\$8.95	0	\$0.00
Rubber wheel, cushion tread, 5"x1-1/4", plain bearing, 210lb capacity	\$4.91	2	\$9.82	0	\$0.00
neoprene rubber wheel, cushion tread, 3-1/2"x1-1/4", plain bearing, 200lb capacity	\$11.47	2	\$22.94	0	\$0.00
rubber wheel, cushion tread, 4"x1-1/4", plain bearing, 200lb capacity	\$3.23	2	\$6.46	0	\$0.00
Super-soft rubber-tread wheel, 5"x1", 5/16" axle, plain bore, 120lb capacity	\$4.88	2	\$9.76	0	\$0.00
Super-soft rubber-tread wheel, 4"x1", 5/16" axle, plain bore, 110lb capacity	\$4.38	6	\$26.28	4	\$17.52
Super-soft rubber-tread wheel, 3"x1", 5/16" axle, plain bore, 100lb capacity	\$3.60	6	\$21.60	0	\$0.00
subtotal:			\$202.56		\$17.52
Motors					
BH31 Geared Motor 31:1	\$23.99	4	\$95.96	4	\$95.96
subtotal:			\$95.96		\$95.96
Metal					
Aluminum Plate 12"x12"x 0.125"	\$26.06	6	\$156.36	0	\$0.00
Aluminum 6061 2-1/2" Square, 1' Length	\$59.88	1	\$59.88	1	\$59.88
Aluminum 6061 3/8" Thick, 4" Width, 1' Length	\$16.65	1	\$16.65	1	\$16.65
Aluminum 6061 .0625" Thick, 12" X 12"	\$13.54	2	\$27.08	2.5	\$33.85
Aluminum 6061 7/8" Diameter, 1' Length	\$14.60	1	\$14.60	0.5	\$7.30
Aluminum 6061 .125" Thick, 12" X 12"	\$29.44	1	\$29.44	0.5	\$14.72
Item - Per Inch	\$0.27	#	\$58.32	104	\$28.08
Aluminum Angle 1.25 x 1.25 Mc88805K52	\$0.24	#	\$3.84	0	\$0.00
Item Profile 5 20x20	\$29.29	2	\$58.58	1	\$29.29
subtotal:			\$424.75		\$189.77

Part Description	Price Ea.		Q't Cost	#Used	Platform Cost
Hardware					
Bronze Thrust Bearing for 1/4" Shaft	\$1.13	#	\$27.12	19	\$21.47
Slotted Spring Pin 1/8" Diameter, 5/8" L	\$4.92	1	\$4.92	0.07	\$0.34
Slotted Spring Pin 1/8" Diameter, 7/8" L	\$6.42	1	\$6.42	0.05	\$0.32
Steel Flat Head Screw #6-32 1/2" Black Oxide	\$10.84	1	\$10.84	0.48	\$5.20
Socket Head Cap Screw 8-32	\$7.16	1	\$7.16	0.08	\$0.57
Threaded Hex Standoff 1/4"	\$0.66	4	\$2.64	4	\$2.64
Alloy Steel Cup Point Socket Set Screw #8-32 1/4" L	\$8.88	1	\$8.88	0.14	\$1.24
Metric 18-8 SS Socket Head Cap Screw M3	\$7.80	1	\$7.80	0.1	\$0.78
Button Head Socket Cap Screw 8-32 0.5" L	\$10.10	1	\$10.10	0.16	\$1.62
Steel Drive Shaft 1/4" OD, 5" L	\$3.68	6	\$22.08	6	\$22.08
Steel Drive Shaft 1/4" OD, 3" L	\$3.07	6	\$18.42	6	\$18.42
Sleeve Bearing for 1/4" Shaft	\$0.32	#	\$7.68	18	\$5.76
Unthreaded Round Spacer 1/2" OD, 5/16" L	\$1.85	7	\$12.95	7	\$12.95
Unthreaded Round Spacer 1/2" OD, 15/16" L	\$3.49	6	\$20.94	6	\$20.94
Low-Clearance C-Style Retaining Ring	\$9.14	1	\$9.14	0.11	\$1.01
Aluminum Round Spacer 1/2" OD, 1/2" Length, 1/4" ID	\$1.31	#	\$13.10	8	\$10.48
Standard Fasteners	\$0.82	#	\$13.12	0	\$0.00
M4 x 8mm Screws - Mc91292A108	\$0.07	#	\$2.24	0	\$0.00
Captive Screw 8-32, 0.125" - Mc92060A120	\$3.57	#	\$99.96	0	\$0.00
Captive Screw 8-32, 0.0625" - Mc92060A110	\$3.39	8	\$27.12	0	\$0.00
TNut	\$1.00	#	\$32.00	0	\$0.00
Item Std. Fastening Set 5	\$0.82	#	\$26.24	12	\$9.84
Item Multiblock PA Set 5	\$1.88	#	\$18.80	12	\$22.56
	subtotal:		\$174.44		\$158.23
Gears/Chains/Linkages					
30 Tooth Miter Gear Pair (Set of Two)	\$3.34	6	\$20.04	6	\$20.04
	subtotal:		\$20.04		\$20.04
	Mechanical subtotal:		\$917.75		\$481.52
Grand Total			\$2,536.15		\$1,603.83
Budget			\$1,500.00		\$1,500.00
Remaining			\$1,036.15		\$103.83
per-Student cost:					\$534.61