A TELECONTROLLED ROBOTIC PLATFORM

Department of Electrical and Computer Engineering

By

Mohammad Roushown Alam

A Thesis

Submitted to the Faculty of Graduate Studies
In partial fulfillment of the requirements
For the degree of

MASTER OF SCIENCE

Faculty Supervisor:
Dr. Robert D. McLeod

Department OF Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba, Canada

© Copyright Mohammad Roushown Alam, October 2006

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

A TELECONTROLLED ROBOTIC PLATFORM

by

Mohammad Roushown Alam

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

of

Master of Science

Mohammad Roushown Alam © 2006

Permission has been granted to the Library of the University of Manitoba to lend or sell copies of this thesis/practicum, to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film, and to University Microfilms Inc. to publish an abstract of this thesis/practicum.

This reproduction or copy of this thesis has been made available by authority of the copyright owner solely for the purpose of private study and research, and may only be reproduced and copied as permitted by copyright laws or with express written authorization from the copyright owner.

Abstract

The present work is part of the development of a semi-autonomously controlled robot. For our purposes an instance or example of a semi-autonomous robot is one that would be able to locate, identify and exterminate weeds in a lawn. The present system is controlled remotely with the help of a vision system, aided by an obstacle avoidance subsystem. The semi-autonomous functionality would become active once the robot was coarsely positioned. In this manner an operator would maneuver the robot assisted by sensor feedback and once in position an apparatus would take over the semi-autonomous function or identifying and extracting weeds. The main focus of the thesis research here is the platform itself and the coarse telecontrol positioning of the robot aided by sensor feedback. This includes the design an implementation of an extendable architecture borrowing and integrating concepts from embedded systems, networking, network programming, and power electronics. The robot consists of sensors, motor drive circuits and different machine elements. The platform is an example of a new generation of personal service robot as identified by S. Thrun [1] among others who have noted that autonomy is one of the major challenges facing the robotics community. The long term goal of the research is to contribute to machine learning operating in conditions of uncertainty with the platform developed providing a research vehicle for future research.

The actual semi-autonomous task will make use of a decision making controller or similar process for locating weeds in its field of view once maneuvered into position under remote control of an operator. More specifically in the present instance an operator can remotely control the system using a steering system with feedback provided by video and relatively simple sensors. Ideally, a number of accelerometers would be mounted to the chassis to sense the vibrations caused by the environment. These would then be transmitted to the user along with the video image. In this manner the user can feel the vibrations at the control console reflecting those encountered on the real chassis. Challenges at present include the synchronization of the sensory data along with the video. The variable delay associated with the video also presents a major challenge for

any remotely controlled device and even more challenging here as all control is over an Internet Protocol (TCP/IP).

In summary this thesis examined various architectures and developed an extendable platform or testbed suitable for addressing robotic autonomy. The core technologies include capitalizing on Internet protocols, Ethernet onboard the mobile, embedded linux processors or reconfigurable hardware that when combined provide for a modular testbed. The wireless technology is 802.11 although could be extended to any that support TCP/IP. The work is oriented towards open source and standards making it of greater utility to those who want to extend or use the base platform. Perhaps the most significant aspect of the work is that the basic platform developed here, can be used for a variety of applications requiring the inclusion of telecontrol and semi-autonomous functionality without having to reinvent the core building blocks.

Acknowledgement

The work presented in this thesis would not have been possible without the help and support of a number of people. First and foremost I must thank my supervisor, Dr. McLeod for his advice and support. I would also like to thank the ECE Tech. Shop (including Allan McKay, Ken Biegun, Mount-First Ng, Guy Jonatschick, Gordon Toole) for all the technical assistance, and the Department of Electrical and Computer Engineering for providing the funding for this project. In addition, I would like to thank Mr. Jeff Anderson of the Internet Innovation Centre for providing some additional funding through a Manitoba Hydro Research Grant.

And finally, thank you to my wife, Karabi, and our children Ismail, Juwairiyyah and Yousuf for all the love, support, and encouragement a man could ask for. You are the reason, the motivation, and the inspiration. You gave me the time and support needed to research, write this report and complete my degree. Without them, I never would have made it.

Table of Contents

List of Figures

List of Tables

Acronyms and Abbreviations

Acronym	Meaning
AP	Access Point
BPSK	Binary Phase Shift Keying
BSS	Basic Service Set
CCK	Complementary Code Keying
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CTS	Clear to Send
DCF	Distribution Coordination Function
DHCP	Dynamic Host Configuration Protocol
DIO	Digital Input Output
DS	Distribution System
DSSS	Direct Sequence Spread Spectrum
ESS	Extended Service Set
ETSI	European Telecommunications Standards Institute
FCC	Federal Communications Commission (USA)
FHSS	Frequency Hopping Spread Spectrum
IBSS	Independent Basic Service Set
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPSec	Internet Protocol Security
ISA	Integrated Services Architecture
ISM	Industry, Scientific, and Medical
JPEG	Joint Photographic Experts Group
ISO	International Organization for Standardization

LLC	Logical Link Control
MAC	Media Access Control
MIB	Management Information Base
MKK	Radio Equipment Inspection and Certification Institute (Japan)
NIC	Network Interface Card
NOS	Network Operating System
PCF	Point Coordination Function
PCI	Peripheral Component Interconnect
PRNG	Pseudo Random Number Generator
PWM	Pulse Width Modulation
QPSK	Quadrature Phase Shift Keying
RC4	Ron's Code or Rivest's Cipher
RTS	Request to Send
SNMP	Simple Network Management Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
WECA	Wireless Ethernet Compatibility Alliance
WEP	Wired Equivalent Privacy
WLAN	wireless local area network
WLANA	Wireless LAN Alliance

Chapter 1 : Project Overview

CONTENTS

1.1 Motivation

Recently Design News [Design News 11.06.06, Volume 61, Number 16] featured an article about BEAR, a (**Battlefield Extraction-Assist Robot)**. This type of robot was deemed to fill a niche between small robots and large remotely controlled vehicles. The Bear robot's most impressive attribute is its ability to balance somewhat like a Seqway using gyros to maintain its center of gravity over the wheels. Bear consists of a number of microcontrollers networked together each providing control and sensory feedback. The article goes on to mention that the main goal of these types of robots is full autonomy. A large number of applications of these robots are also mentioned on the Vecna web site homepage www.vecna.com (the developers of Bear) and include patient care and robotic assistants for the elderly. Many other similar projects are being developed also filling the niche between small robots and large industrial robots. Another example is a robot project by **Spark Robotics Technology Inc.**[1], a small Vancouver start-up. They specialize in robotic solutions that provide: Collision Avoidance, Tele-Robotic Control, Operator-Assist Functionality, and Semi-Autonomous Operation. Their approach is similar to ours except that the system they are developing does not appear to be as open or standard as the platform we are developing.

[1] www.sparkrobotics.com/index.html

Other more traditional research projects include telecontol of manipulators with emphasis on issues such as teleoperations and virtual reality [2]. Other projects include augmenting the robot with local embedded control functionality [3] to perform a number of onboard tasks such as to move along a predefined trajectory, avoiding obstacles via fuzzy control, and identifying objects that it may have to maneuver around.

Projects such as these, industrial interest, and research papers illustrate a considerable interest in robots of reasonable size that are required to perform tasks of some physical difficulty or work. The research developed here recognizes some of the issues associated with reasonable sized robots and is an attempt to build an extendable platform using a flexible architecture capitalizing on best of class, open, and off the shelf proven technologies. Once the basic platform is developed it can be used for a variety of applications requiring the inclusion of telecontrol and semi-autonomous functionality without having to reinvent the core building blocks.

1.2 Introduction

The present work considers development of a telecontrolled robot platform. The platform developed has several core technologies useful as a starting point for many semi-autonomous applications. More specifically these core technologies for moderate sized robots have been identified and include:

1. A rugged chassis and frame driven by relatively powerful electric motors, capable of being fitted with various apparatuses.
2. A wireless communication interface allowing command and control from a remote host. The communication utilizes TCP/IP as opposed to more traditional RC (Radio Control).
3. Embedded processor(s) or hardware modules connected on board the robot via Ethernet
4. Custom motor controllers and speed controllers
5. Sensors such as Video, sonar, and GPS. (GPS not available at this time)

To put the platform in perspective, an instance of a semi-autonomous application that can be developed on the platform would be a robot that would be able to locate, identify and exterminate weeds in a lawn. This application was selected early in the project process as one that captures design requirements that similar complexity semi-autonomous applications would have. In addition, as a motivation this application is also expected to be useful to those who would like to destroy weeds on their lawns without the use of herbicides. As mentioned in the abstract, this would be an example of the new generation of personal service robot as identified by S. Thrun [1] who has noted that autonomy is one of the major challenges facing the robotics community and represented the type of design challenge that we wanted to undertake. The specific example mentioned above is less of an issue as compared with overall notion of building autonomous robots. The main objective here is to develop a platform or testbed where techniques and algorithms to deal with uncertainty can be implemented, invariant of how routine the application may appear. Even the most pedestrian or routine of applications present challenges when one attempts to actually implement automatic solutions. The long term goal is to contribute to machine learning operating in conditions of uncertainty and to provide a flexible platform and architecture for future research.

1.3 Project Objectives

The present system is to be controlled remotely with the help of a vision system, aided by an obstacle avoidance subsystem. The robot consists of sensors, embedded controllers, motor drivers, and different machine elements (chassis, motors, wheels, etc).

The actual semi-autonomous task will make use of a decision making controller or similar for locating weeds in its field of view once maneuvered into position under remote control of an operator. A preliminary image processing study has been undertaken in our laboratory by Mr. Marek Laskowski who evaluated a Centroidal Voronoi Tesselation method for weed identification [http://www.iic.umanitoba.ca/docs/robot-final.pdf]. A technique such as this would be a candidate for the weed recognition algorithm deployed here. My specific task here is to develop the platform and its remote control over wireless using TCP/IP. As such, the current robot combines aspects of

remote control with a semi-autonomous function with a long term goal of full autonomy in the future. In the present instance an operator can remotely control the system using a steering system with feedback provided by video. Ideally, a number of accelerometers mounted to the chassis will sense the vibrations caused by the environment. These will then be transmitted to the user along with the video image. In this manner the user can feel the vibrations at the control console reflecting those encountered on the real chassis. The current platform only supports video at this time and limited sonar input to serve to alert the operator if they are encroaching to close to an object that they otherwise could not see. The auxiliary sensors are in fact core to the platform and its remote operation. The reason for this is due to excessive delay introduced by the video transmission and processing. Under these conditions a limited amount of onboard sensor processing and decision making may avert an accident or collision that the operator would not be able to know about or respond to as a direct consequence of the uncertainty in the delay and video processing. Challenges at present include the synchronization of the sensory data along with the video. As mentioned, the variable delay associated with the video presents a major challenge for any remotely controlled device and even more challenging here as all control is over an Internet Protocol (TCP/IP). The main thrust of the thesis is to develop the remote control functionality using TCP/IP over wireless access augmented by onboard processing of limited sensory input. In a moderate alert scenario the onboard sensors would report to the operator. In a more severe situation the onboard processing of sensor inputs would assume control and take the necessary action to avert a collision. These notions are not new to this project and can be seen in a number of commercial and industrial settings. For our purposes however they provide an opportunity for algorithm exploration that we would not otherwise have.

This project is constrained in design to something that is manageable yet captures the essence of the problems involved with all efforts in building autonomous or semi-autonomous systems. In addition to the platform described above a mechanical three-axis robotic platform equipped with a drill head is being designed and implemented to root out weeds. Even if we are unable to remove the weeds in an autonomous fashion under full autonomy with the platform developed here an operator will still be able to select the target weed using a joystick with the help of a vision system and execute the desired

operation through the push of a button. It is uncertain at this time if this type of processing can be used in training the robot in the future but is included here as a goal while building the robot platform itself. The notion of being able to control or supervise the robot remotely may become an essential mechanism in developing semi-autonomous functionality for many personal service robots. This again is not new to our application as it is a common theme in the neural network area. However the platform being developed will facilitate supervised learning as a by-product of the remote control architecture developed and presented here [4].

1.4 Project Feasibility and Methodology in Brief

1.4.1 Feasibility of Implementation

The remote control part of the robot will be implemented over an 802.11 wireless network which is the most widely available and deployed wireless LAN technology [5]. Initially this project is not expected to be an autonomous robot and as such requires some means of telecontrol. As mentioned earlier, it would be controlled remotely by an operator. Thus, it is necessary for the operator to see the environment before driving the robot in any particular direction. A vision system is necessary to facilitate that function. This can be implemented using the MPEG2 video stream. Over 802.11 the vision system would transmit the MPEG2 video stream to the operator using the TCP/IP protocol. In the present manifestation, the camera is fixed and is directed towards the front. Even though the operator can see the front of the robot, the image is delayed due to the encoding, transport and decoding processing which is on the order of seconds. A delay of this type in this environment is non negligible. This in effect means that the operator sees only what happened in the past (i.e. a few seconds ago depending on the encoding, transport, and decoding delay). This leads to a possibility of potential collisions in a dynamic environment. A variety of sensor technologies are of potential use in this situation. A number of proximity sensors could be employed in the design of a collision avoidance subsystem. Proximity sensors were experimented with using the new Altera DE2 FPGA board but could not be fully integrated as there is a problem with voltage conversion from 5V to 3.3V signaling. This problem is currently being addressed and will be fully functional in the near future. Ultra sonic sensors are used in measuring the

distance of the closest obstacle within collision range. But these are somewhat limited as they are not good for determining distances closer than a foot or so. On the other hand Laser Range Finders are very good in determining object with greater precision. Due the limited funding available to this project, ultra sonic sensors are being used for the time being. In spite of these limitations we believe that the basic framework developed here will prove useful in building additional apparatuses or attachments in future work.

In order to provide some degree of feeling of the environment to the operator, the robot could be equipped with accelerometers for sensing vibration felt by the chassis or sensors for determining the environment temperature, humidity, toxins, etc. These sensory data could be captured using a data acquisition card and sent to the operator along with the video. A force feedback joystick can reflect the sensory data to the operator's hand. Based on our preliminary experience with maneuvering the robot in the field it is also important to synchronize the video with the sensory data. A solution to this synchronization problem is beyond the present scope but a potential solution will be provided in the future work section of the thesis.

1.4.2 Methodology in Brief

For the weed eater application the vision system has two parts. One is to provide front view of the environment and the other is to provide an aerial view to detect weeds from the lawn through the use of its image recognition subsystems. A colleague (Marek Laskowski) is currently evaluating a number of image recognition strategies such as Centroidal Voronoi Tessellations or CVTs [6] that may be integrated into the present research if time permits.

A number of sonar sensors are incorporated for a more robust obstacle avoidance subsystem. These will provide information in terms of obstacles that may not be available from the vision system alone. In general if the system approaches an obstacle or an obstacle approaches the system a decision controller will be able to assess the situation and alert the operator or determine a suitable course of action to avoid collision. This may include overriding the operator's command by reducing speed, possibly changing direction or even stopping the movement of the system completely. These types of

problems will involve the development of suitable heuristics analogous to those used in similar environments. A number of bumper pressure sensors are deployed as part of the fuzzy controller to reduce the impact of a possible collision by bringing the system to a complete stop position.

The field of robotics is changing very rapidly and has evolved considerably over the past ten years. Many of the changes have been driven by the reduction in cost of sensors, actuators and processors. This has allowed research to be conducted in a more practical manner than previous. In addition, the progress made in telecommunications is having a direct bearing on mobile robotics. There are basically only a few broad categories of robots ranging from industrial to the more recent personal service robots. This thesis primarily focuses on firstly developing the telecontrol of a platform that will allow us to address open questions in the area of personal service robots and in secondly addresses issues associated with autonomy and human robot interaction.

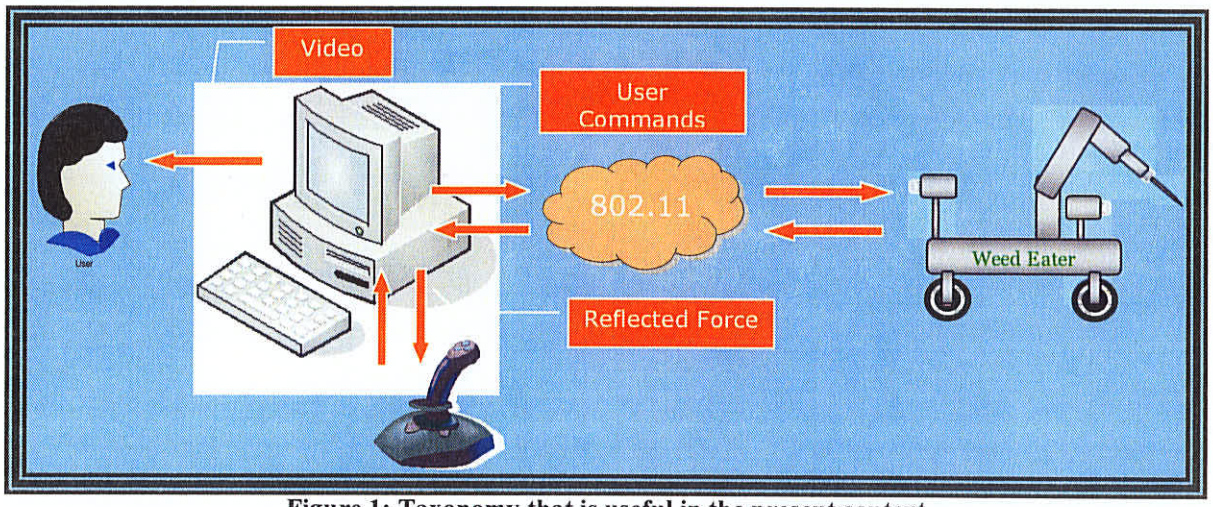

Figure 1: Taxonomy that is useful in the present context

Figure 1 outlines a partial taxonomy that is useful in the present context. Traversing the taxonomy we quickly descend to the major factors influencing the robots of interest here. From left to right we classify the major factors as the human-machine interface, the wireless communication technology and protocols, and lastly the remote platform tasked with some semi-autonomous function. A similar taxonomy or classification can be created that roughly illustrates views of intelligence or reasoning that we hope to be able to exploit. As such, in terms of intelligence we plan on emulating high level functionality where possible with at least a plausible biological model as a

reference. This basic architecture provides a platform such that the robot can be remotely controlled, operate in an autonomous or semi-autonomous fashion, extendable in functionality and/or provide a means of off-loading computation to a facility, cluster or grid for processing if required.

More specifically and with the robot and our intelligence taxonomy in mind, one of the early goals of the current project was to design the telecontrol module of a robot that could be remotely maneuvered over an 802.11 wireless network, positioned and then tasked to perform an operation in a semi autonomous fashion. Figure 2 more precisely illustrates a more detailed architecture of the actual mechanisms and functional units.

For an objective of the present platform a specific task would be to recognize an object (weed) in its field of view and move a 3 axis positional system to the object. The latter task requires the robot to reason and act in a manner similar to a "human" or at least be able to reason and decide upon an action to be undertaken mimicking human behavior. This thesis is focused on the design of the remote telecontrol and the human machine interface that would allow semi-autonomous tasks to be implemented such as the one mentioned above without reinvention of the core technologies. The design of human machine interface is also one of the most challenging aspects and opens questions facing modern personal service robots [1].

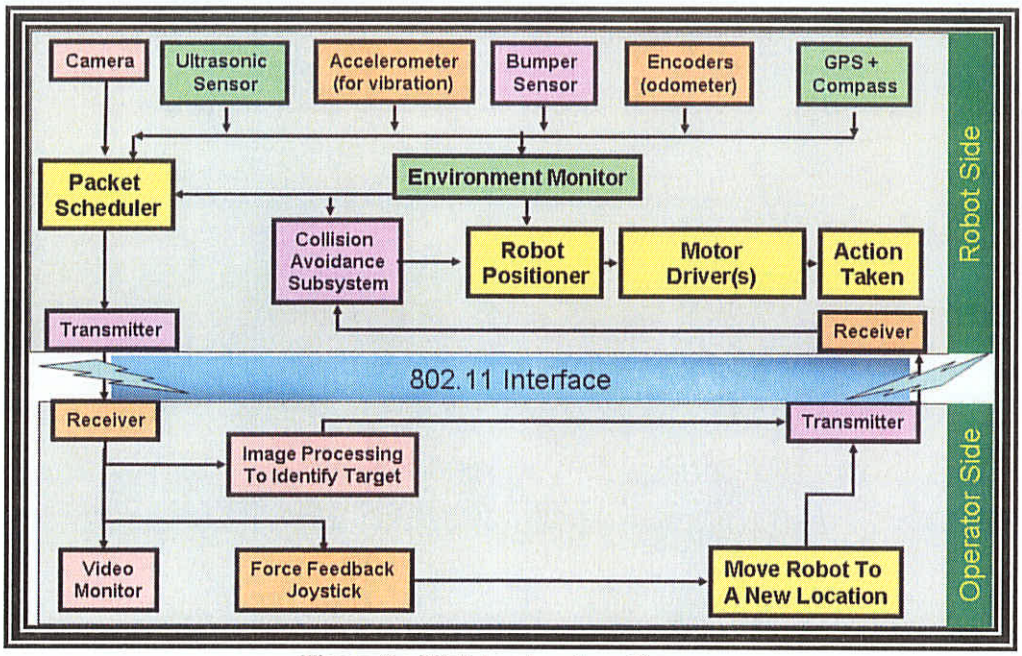

Figure 2 : High level system diagram

This thesis discusses the design of the robot including the mechanical considerations, the motor control circuits, the man machine interface, and subjective discoveries made along the way. The thesis concludes with recommendations that will be useful for others wanting to remotely control a robot utilizing protocols over IP. The other overall constraint was one of cost with the project constrained to an absolute minimal of budget. This constraint is one that although not the most desirable is one that requires considerable problem solving strategies requiring thought, patience, and perseverance.

One of the requirements of the project was the development of a robot that would be relatively robust capable of carrying a reasonable payload. The reason for this was to make the platform general enough that other apparatuses could be attached for a variety of projects. Although these are fuzzy notions it was clear from the outset that the types of machines envisioned would be capable of real work. A caveat of real work is that the robot would be of sufficient size to do something manual, thus in effect having inertia. Under this constraint a heavy duty wheel chair frame and 24 volt DC motors were procured. This environment is one that requires controlling considerable currents as the payload of the robot is well over 100 pounds with the basic chassis weighing the same.

This requirement necessitates design considerations that would not be as prevalent in small scale robots.

The remainder of the thesis is organized as follows. The background study of technologies used in building the mobile platform is described in Chapter 2. The available hardware components and software tools of the project are discussed in Chapter 3. The system design and development of the robot itself is described in Chapter 4. Some preliminary tests were conducted in basic remote control of the robot and are discussed in Chapter 5. A summary and future work follows in Chapter 6.

Chapter 2 : Background Study

CONTENTS

2.1 802.11

802.11 is a data transmission system which is also known as a wireless LAN (WLAN) technology [5]. 802.11 provides location-independent network access between computing devices by using radio waves rather than a cable infrastructure. The OSI model of 802.11 is given in Figure 3. Its design is focused mainly on providing high throughput, highly reliable data delivery, and continuous network connection. 802.11 covers the Medium Access Control (MAC) sub layer, MAC management protocols and services. It also contains three physical layers namely Infrared (IR), Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS). At this time the physical layer is primarily some variant of spread spectrum or Orthogonal Frequency Division Multiplexing (OFDM). The point for us here is less in the choice of physical layer and more the fact that 802.11 most closely resembles traditional Ethernet and as such, applications developed on TCP/IP are easily deployed over 802.11.

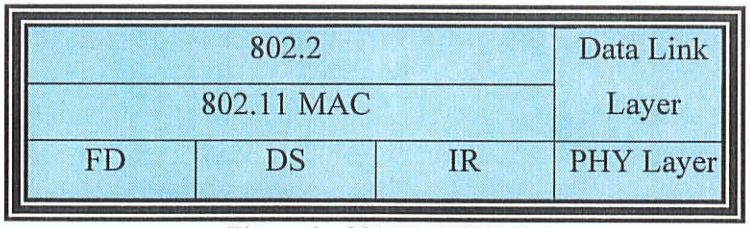

Figure 3 : 801.11 OSI Model

As mentioned, there are a number of 802.11 variations. The physical layer specification for 802.11, 802.11b, 802.11a and 802.11g is shown in table below.

Physical Layer Specification			
802.11 Variation	PHY Max data Rate	Frequency	Modulation
802.11	2Mb/s	2.4GHz	IR, FHSS, DSSS
802.11b	11Mb/s	2.4GHz	DSSS
802.11a	54Mb/s	2.4GHz	OFDM
802.11g	54Mb/s	5GHz	OFDM

Table 1 : Physical Layer Specification

802.11 Wireless LAN (WLAN) is based on a cellular architecture where it may contain one or more cells. These cells are called Basic Service Sets (BSS). Each of the BSS is controlled by a base station called Access Point (AP). In a multi cell WLAN APs are connected through backbone called Distribution system (DS) (i.e. Ethernet LAN or WLAN). The combined Wireless LAN including the cells, the DS and the corresponding APs is called Extended Service Set (ESS) (see Figure 4).

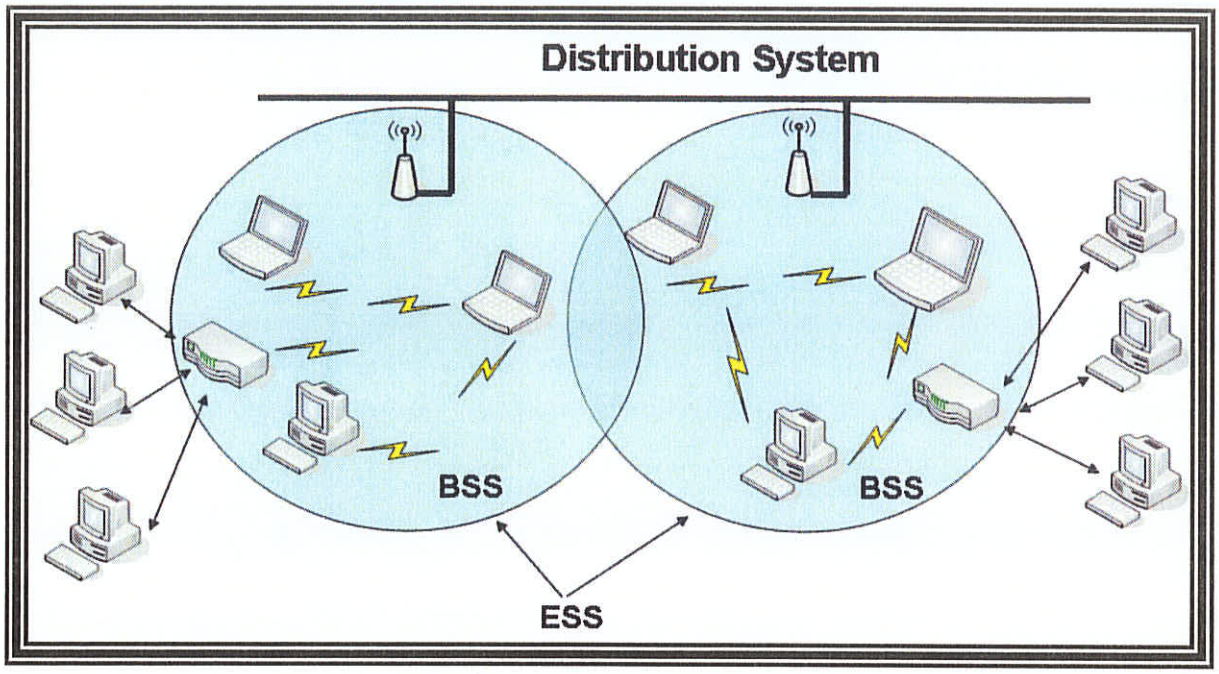

Figure 4 : A typical 802.11 ESS

The choice of 802.11 as a core technology for the robot platform is more evident when considering Table 1 and Figure 4. In Table 1 it is clear that 802.11 will continue to remain the dominant WLAN technology with newer physical layers being developed to continually improve its performance while Figure 4 illustrates that 802.11 is suitable for most instances of personal service robots where range will likely be restricted to a campus size area or less. In the event greater distances are required than offered by 802.11 it is possible with limited changes in the system to migrate the platform to technologies such as 802.16 for example (Wi-Max).

2.2 MPEG2

MPEG 2 is a lossy video compression technology. Raw digital video source is amenable to efficient encoding. MPEG2 capitalizes on the fact that a tremendous amount of redundancy exists in digital video not only within a given frame but across frames. A transmitter converts raw video and encodes it into an MPEG2 stream. This stream is transmitted to a receiver that decodes the stream creating a close approximation to the original image. These processes of encoding, packet encapsulation, decapsulation, and

decoding consume non negligible compute and transmission time and are issues that need to be addressed.

Although this project is not concerned with video encoding itself it is apparent that some form of video encoding is required. As such, the MPEG2 codecs were selected due to their availability and maturity. As newer and better codecs become available these will be utilized. In any event, independent of which ever codecs are used there will always be some delay, made particularly worse as remote control of platforms such as ours is capable of being controlled across the Internet.

This delay is one of the reasons that make controlling a robot remotely a challenging task. Although the control can be reasonably responsive, the video delay adds a degree of uncertainty that makes control potentially unstable. As a consequence it was decided to add auxiliary sensors that would aid the operator in situations where the feedback from a delayed video could be problematic.

2.3 Pulse Width Modulation

In this project the motor speed is controlled using Pulse Width Modulation (PWM). In general PWM is way of encoding an analog signal into digital form. PWM is also used in efficient voltage regulators. By switching the voltage to the load with the appropriate duty cycle, the output will approximate a voltage at the desired level. PWM uses a square wave whose duty cycle is modulated resulting in the variation of the average value of the waveform [7]. If we consider a square waveform $f(t)$ with a low value y_{min}, a high value y_{max} and a duty cycle D (see Figure 5), the average value of the waveform is given by:

$$\bar{y} = \frac{1}{T}\int_0^T f(t)\,dt$$

As $f(t)$ is a square wave, its value is y_{max} for $0 < t < D \cdot T$ and y_{min} for $D \cdot T < t < T$. The above expression then becomes:

$$\begin{aligned} \bar{y} &= \frac{1}{T}\left(\int_0^{DT} y_{max}\,dt + \int_{DT}^{T} y_{min}\,dt\right) \\ &= \frac{D \cdot T \cdot y_{max} + T(1-D)y_{min}}{T} \\ &= D \cdot y_{max} + (1-D)\,y_{min} \end{aligned}$$

This latter expression can be simplified in many cases where $y_{min} = 0$ as $\bar{y} = D \cdot y_{max}$. From this, it is obvious that the average value of the signal ($\bar{y}$) is directly dependent on the duty cycle D.

The duty cycle is the ratio of the on-time to the period; and the modulating frequency is the inverse of the period. Figure 6 shows different PWM signals with 20%, 50% and 90% duty cycle respectively. These three PWM outputs encode three different analog signal values, at 20%, 50%, and 90% of the full strength.

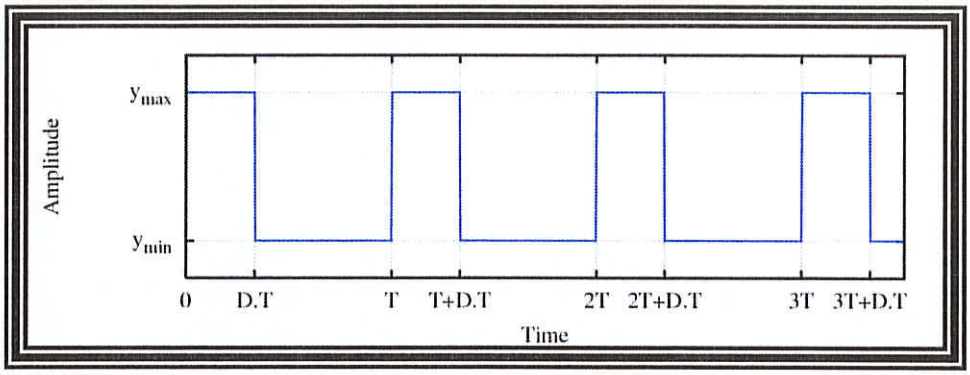

Figure 5 : A square wave, showing the definitions of y_{min}, y_{max} and D
Source: http://upload.wikimedia.org/wikipedia/commons/9/9e/Duty_cycle_general.png[2]

In order to control speed of a 24V DC motor, one may vary the input voltage to the motor between 0V and 24V. Using a PWM signal speed control can be achieved in following manner. Using the example from Figure 6, if the duty cycle is 20%, the resulting analog signal would be 4.8V, similarly 12V for 50% duty cycle and 21.6V for 90% duty cycle. Although these pulse trains only have an associated average voltage it is easier to control a motor using PWM than using an analog voltage directly.

[2] All the materials found at http://en.wikipedia.org/wiki/Main_Page are available for all form of reproduction. Permission granted under GNU public license agreement found at http://en.wikipedia.org/wiki/Wikipedia:Copyrights

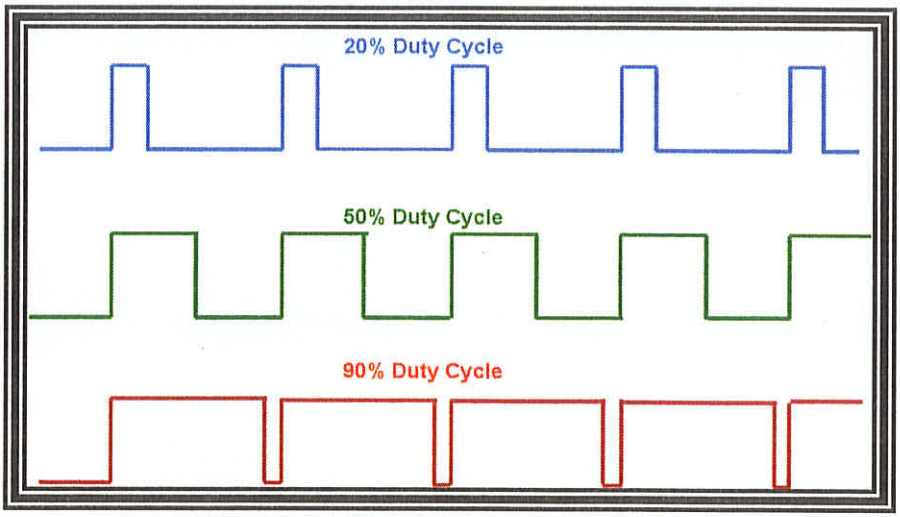

Figure 6 : Examples of PWM duty cycle

2.3.1 Generation of the Pulse Width Modulation (PWM) Signal

There are many ways to generate PWM signals. This can be done in analog or by digital methods. A digital method provides flexibility through the use of microcontrollers and appropriate software and hardware.

2.3.1.1 Analog Method of PWM Generation

The easiest way to create a PWM signal is to use a triangular wave generator; a reference DC voltage level and a comparator connected in the following configuration (see Figure 7). The higher the DC level is, the wider the PWM pulses are. The DC level is denoted the 'demand signal'. The DC signal can range between the minimum and maximum voltages of the triangle wave.

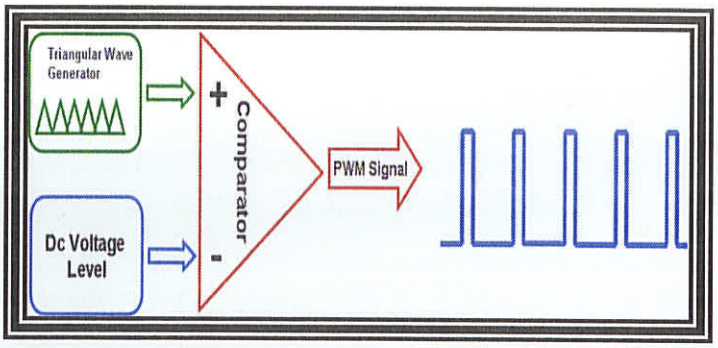

Figure 7 : Generation of PWM signal using comparator

2.3.1.2 Digital Method of PWM Generation

As mentioned earlier, a digitally generated PWM signal could be produced using hardware or software. Regardless of the method, the procedure is more or less the same for each of the methods (see Figure 8). The digital method requires two high resolution timers which contain counter register(s). This register of the timer is normally loaded with a number which normally represents the on/off time. Using a clock, this timer register is decremented. Once the register value goes to "0" then it produces a pulse. Using this pulse one can control the duty cycle of the desired PWM signal.

The first step is to obtain two timers denoted On-Timer and Off-Timer. The process is as follows. Stop the timers if they are running, determine the frequency of the PWM signal desired and obtain a clock signal to meet the requirement. Then load the On-timer with a number that would represent the duty cycle. Start the On-Timer. Once the On-Timer underflow is detected, disable it, set the output to "0" (i.e. 0V for level low) at the same time load the Off-Timer register with Off-Time value and start the Off-Timer. Once the Off-Timer underflow is detected, disable it, set the output to "1" (i.e. 5V for a level high) at the same time load the On-Timer register with On-Time value and start the On-Timer. This cycle continues until it is stopped at will. As noted above the control of the PWM signal is through the use of register values that set timers corresponding to the desired frequency as well as the duty cycle. A disadvantage of the method described above is that if it is done entirely in software in an embedded microcontroller it can be a time consuming process that limits what the microcontroller may also do.

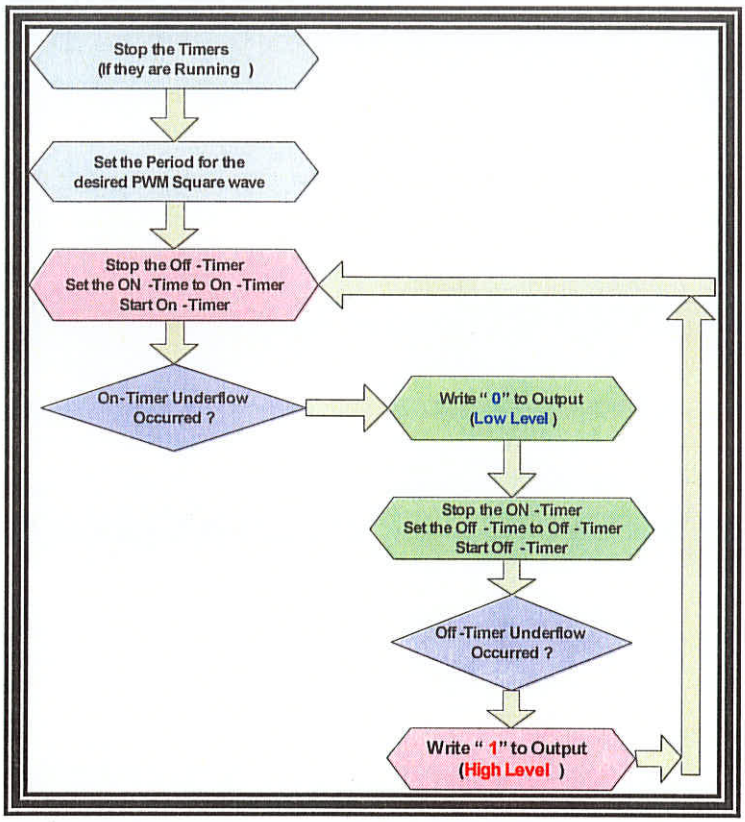

Figure 8 : Steps for producing PWM signals

2.4 H-Bridge Motor Control

The H-bridge allows a motor to be controlled in terms of direction as well as duty cycle. The H-bridge configuration consists of four switching devices that allow you to change the direction of current flow through a load. In our case the H-bridge allows the motor to be driven either forward or backward. This is done by selecting which pair of switches is on and which pair is off. Originally the H-bridge used in this project was designed and fabricated according to our requirements using a Printed Circuit Board (PCB) milling machine. The primary requirement was for the H-bridge to handle approximately 30 Amps driving the 24 Volt motors under steady state conditions. It was found that this approach was difficult in terms of making additional controllers and too costly and time consuming to manufacture by hand. A second attempt was to custom manufacture a board using in-house photolithography but it was also difficult to fabricate the PCB with the desired resolution in terms of trace widths. Another problem was the

availability of the laminated copper board. In our case 4 oz. copper was required to handle the motor drive currents. The PCBs produced by this method were too unreliable and unable to handle the heavy currents demanded by the platform. As such, an open source motor controller [14] was customized to meet the requirements. This controller and components are discussed below. The open source controller PCB provides a base design that can be customized to meet specific current requirements and overload protection.

S1	S2	S3	S4	Function
On	Off	Off	On	Forward
Off	On	On	Off	Reverse
On	On	Off	Off	Brake
Off	Off	On	On	Brake
On	Off	On	Off	Short Circuit
Off	On	Off	On	Short Circuit

Table 2 : H-Bridge Functionalities

Figure 9 shows the basic circuit configuration of an H-Bridge with four switches namely S1, S2, S3 and S4. Here the load is a DC motor which is connected as shown in Figure 9. When all the four switches are open, the circuit is configured as being in OFF mode, when the pair S1 and S4 switches are closed, the circuit is configured in the Forward mode and similarly when the pair S2 and S3 switches are closed, the circuit is configured for Reverse mode operation. Table 2 shows a number of functionalities of an H-Bridge circuit in a table format.

If under any circumstances, S1 and S3 or S2 and S4 are closed at the same time, they will constitute a short circuit from power to ground. This short circuit is sufficient to destroy the circuit board or its components. The switching devices can be implemented using transistors (i.e. MOSFETs) or relays. MOSFETs were selected for this project. There are a variety of MOSFET H-Bridge drivers available in the market. Most of them provide some protection against this short-circuit configuration.

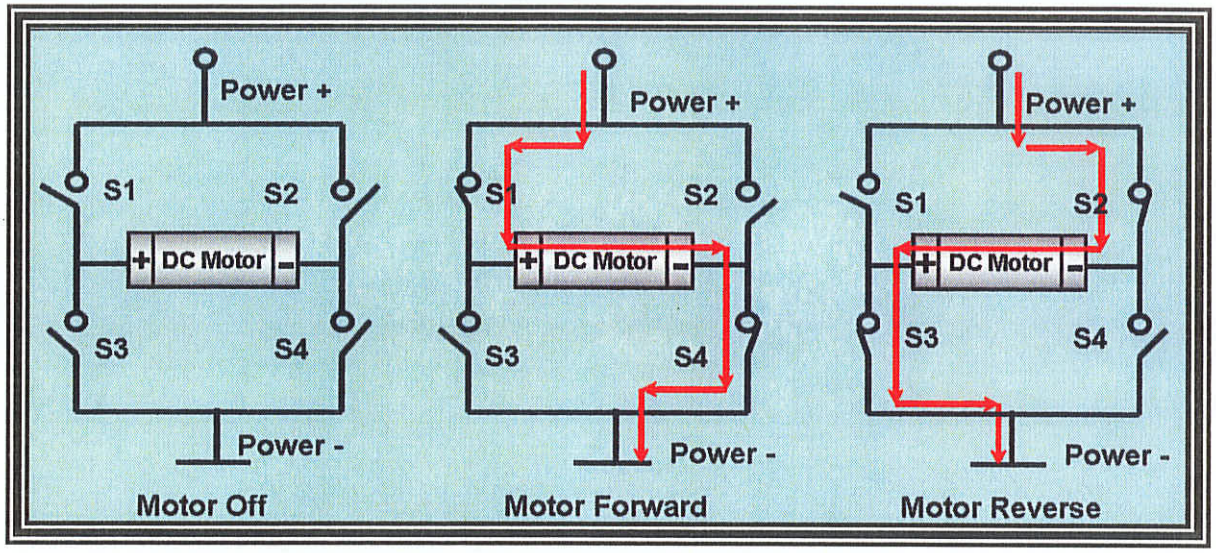

Figure 9 : The basic structure of an H-Bridge circuit

2.4.1 The protection against gate-source voltage spike

DC motors are inductive loads. During switching they produce voltage spikes, sometimes exceeding the maximum voltage limit allowed by the MOSFET devices. Without a good protection from these spikes, the driver circuits may not last because the gates would breakdown. Transient Voltage Suppressor (TVS) diodes can be used to suppress these voltage spikes. For example, in a case where the maximum allowable drain-source voltage V_{DSS} is 40V. Any voltage above 40V needs to be clipped. The 1.5KE33CA TVS is made by On-semiconductor [15] and can be used for this purpose. It will normally clip any voltage above 33V for 1 millisecond up to 1500W (see Figure 10). These are connected between each of the motor terminals and the ground for the desired protection.

2.4.2 The protection against high gate current

As MOSFETs gates have substantive input capacitance, while the gate is the charging, depending on the size of the gate capacitance, it could draw a substantial amount of current which could destroy the switching device. According to the device specification the IRF1404 N channel MOSFET has an input capacitance of approximately 5669pf (pico-farads). By connecting a multiple of these devices in

parallel, the combined gate capacitance would be much higher and could draw a very large amount of gate current. As stated in the datasheet [Add], the HIP4081A (an H-Bridge Driver) can supply up to 2.5Amps of current to its output to drive the MOSFETs. Supplying this amount of current to the gate could potentially cause them to overheat and possibly self-destruct. The solution to this problem is to limit the current to the gates to a desired level. This is done through the use of small resistors between the driver outputs and the gates. Resisters of 150 ohms were selected for this design (see Figure 10).

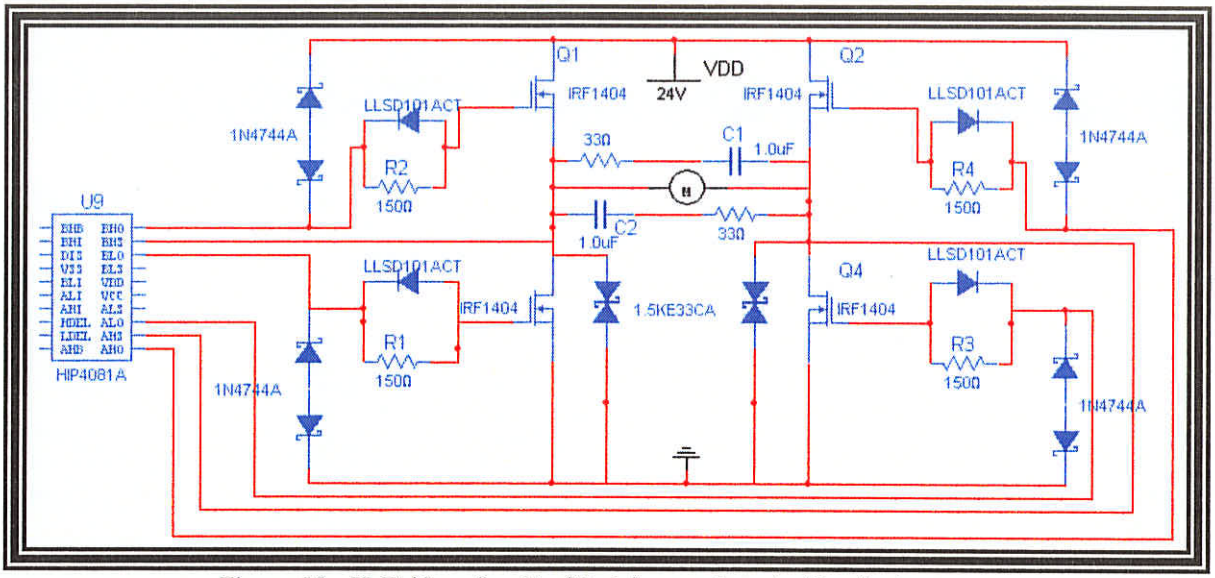

Figure 10 : H-Bridge circuit with driver and protection features

2.4.3 The protection against shoot-through

Through the introduction of the gate resistor for gate current limitation, we have created a possible scenario where the H-Bridge could potentially go into the short circuit mode. The main cause of this shoot-through phenomenon is the fact that it takes some time for capacitors to charge and discharge. In the case of MOSFETs, they tend to charge faster than discharge. For example, during switching (i.e. changing direction) the upper side may still be discharging while the lower side turns on. As a result, both of the upper and lower MOSFETs turn on at the same time. This condition lasts only a few microseconds. But as a result a huge amount of current passes through the switching devices and could destroy them instantaneously. This problem of shoot-through is

avoided by connecting Schottky diodes in parallel to the gate resistors facing the opposite side of the gate. In this manner the gate capacitor charges as usual, but during discharge, the diode produces a bypass path from the gate causing the gate capacitor to discharge faster (see Figure 10).

2.4.4 The protection against gate to source voltage fluctuation

Even though the gate driver provides the correct voltage to turn the MOSFETs on and off while it is in the H-Bridge the gate to source voltage can exceed the specification. This is due to the fact that the DC motor (inductive load) is connected directly to the source or drain. The specification requires that this voltage not exceed 20 V. Zener diodes connected between the gate and source terminals of the MOSFETs can limit the gate to source voltage such that the specification is not violated. These diodes are also shown in Figure 10.

2.5 Differential Steering

Differential steering is a drive system where the steering is done by independently controlling the velocity of the wheels. In this project the robot has two drive wheels and two castor wheels. The drive wheels are used to steer the robot where the castor wheels are there to support and balance the robot. In order to steer the robot in a straight line, it is necessary to turn both of the drive wheels in the same direction at the same speed. To turn left, both of the drive wheels must turn to forward direction keeping the speed of the left wheel lower than that of the right wheel. To turn right, the opposite of turning left is done. To pivot, both of the driving wheels are turned at the same speed but in the opposite direction. A detailed discussion of differential steering can be found at [10].

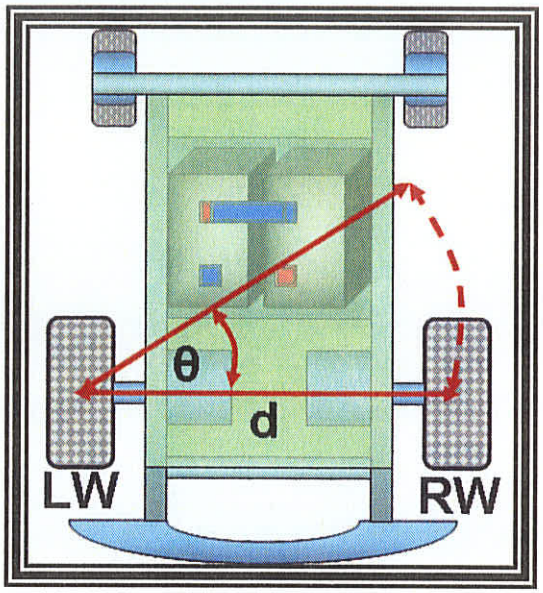

Figure 11 : Differential steering using two drive wheels[3]

Determining the turning angle of the robot in terms of each of the drive wheel's velocity is done as follows. Consider the distance between the two drive wheels *d,* the turning angle θ, velocity of the left and the right wheels are *vl* and *vr* respectively. From the Figure 9 : The basic structure of an H-Bridge circuit the rate of change of the turning angle could be described by:

$$\frac{d\theta}{dt} = \frac{(vr - vl)}{d} \qquad [1]$$

By integrating equation [1] we get

$$\int_0^t \frac{d\theta}{dt} dt = \int_0^t \frac{(vr - vl)}{d} dt$$

$$\theta(t) = \frac{(vr - vl)t}{d} + \theta o \qquad [2]$$

Equation [2] represents the turning angle where θ_0 is the initial turning angle. These equations clearly illustrate how the angle varies as functions of the wheel velocities. However, these types of equations are idealized allowing for considerable error due to

[3] The details about differential steering could be found at
http://en.wikipedia.org/wiki/Differential_wheeled_robot
The original picture is modified. Permission granted under GNU public license agreement found at
http://en.wikipedia.org/wiki/Wikipedia:Copyrights

slippage as well as other factors. One of these factors is a consequence of the platform also having castor wheels. Caster wheels are found on many types of mobiles. They are convenient as they allow for rotations or 360 degrees. They can also cause some difficulty in terms of trying to control the platform over less than ideal terrains. The caster wheels also tend to redirect the robot when they encounter small obstructions or changes in surface topology.

2.6 Summary of Chapter 2

Chapter 2 overviewed some of the technologies used in constructing the robot platform. These included an overview of 802.11 which is the technology selected as the wireless technology used to remotely control the robot over. A brief description of the MPEG2 video compression standard and delay problems associated with controlling the robot when video is used as the main means of feedback was discussed. This consideration leads to a requirement of adding sensors to assist the remote operator in situations when the operator may not be able to determine encroaching obstacles. The most extensive section of Chapter 2 was devoted to the design of the motor controller and selection of components that would provide sufficient current while protecting the electronics in situations where there would otherwise be excessive voltages of currents.

Chapter 3 : Hardware and Software Resources used in the Project

CONTENTS

3.1 Hardware

For this project the main piece of hardware available was an old motorized wheelchair. The wheelchair was inoperable due to its non functioning controller. Thus the only useable salvage was approximately 100 pounds of metal chassis with two 24V DC motors. Two powerful bidirectional DC motor controllers were acquired and assembled for driving the DC motors. These DC motor controllers are very primitive. Although they were modified to accommodate the currents required and equipped with protection circuitry they merely provide an interface to the motors themselves. They did not have any intelligence built into them; therefore another controller was needed to make the motor controllers more useful. For this purpose, an ARM single board computer (SBC) was used. The ARM SBC effectively provided an API to the motor controller. To provide visual feedback one Logitech QuickCam camera was acquired for visual feedback while driving the robot, for the actual controller interface a force feedback Logitech MOMO

steering set was purchased. The integration of these components are discussed in more detail subsequently.

3.1.1 Acer Laptop

An Acer laptop (see Figure 12) was provided as a software development platform for the robot platform. The laptop simplified the wireless interface for the remote control and simplified the video capture. The Acer has an Intel® Celeron® processor with 512MB of memory and 60GB of Hard Drive space. Its USB interface was used for interfacing the Logitech QuickCam®. It also has an Ethernet port, which was used for establishing connection between the Laptop and the ARM SBC. One of the design objectives of the platform was to use Ethernet wherever possible making the platform even more versatile in the future. In addition, using the Acer accelerated the implementation of the user interface also shown in Figure 12. Although a laptop is currently being used, the final version of the platform will replace the laptop with a Linux box to reduce the overhead associated with Windows environment.

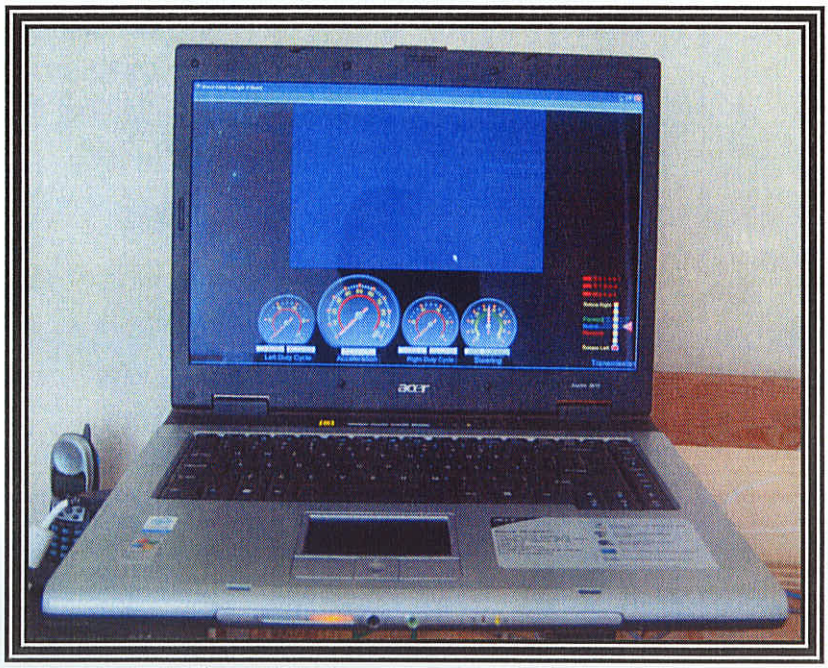

Figure 12 : Acer Laptop GUI displayed

3.1.2 ARM Single Board Computer

As one of the design goals was to develop embedded sub-components on the platform an ARM SBC was selected as it supported a wide variety of interfaces. The TS-7250 is a full featured Single Board Computer (see Figure 13), and comes with the following standard features:

- 200 MHz ARM9 CPU with MMU
- 32MB RAM
- 32MB on-board Flash (Boots to Linux)
- 10/100 Megabit Ethernet port
- 2 USB compliant Full Speed host (OHCI) ports -- (12 Mb/s max)
- 2 COM ports (up to 230 KBaud)
- Watchdog Timer
- PC/104 expansion bus
- SPI bus interface
- 20 Digital I/O lines
- Alphanumeric LCD and matrix keypad interface
- Single +5VDC supply @ 400 mA – (under 2 watts)
- Small size -- 3.8 x 4.5 inches (9.7 x 11.5 cm)

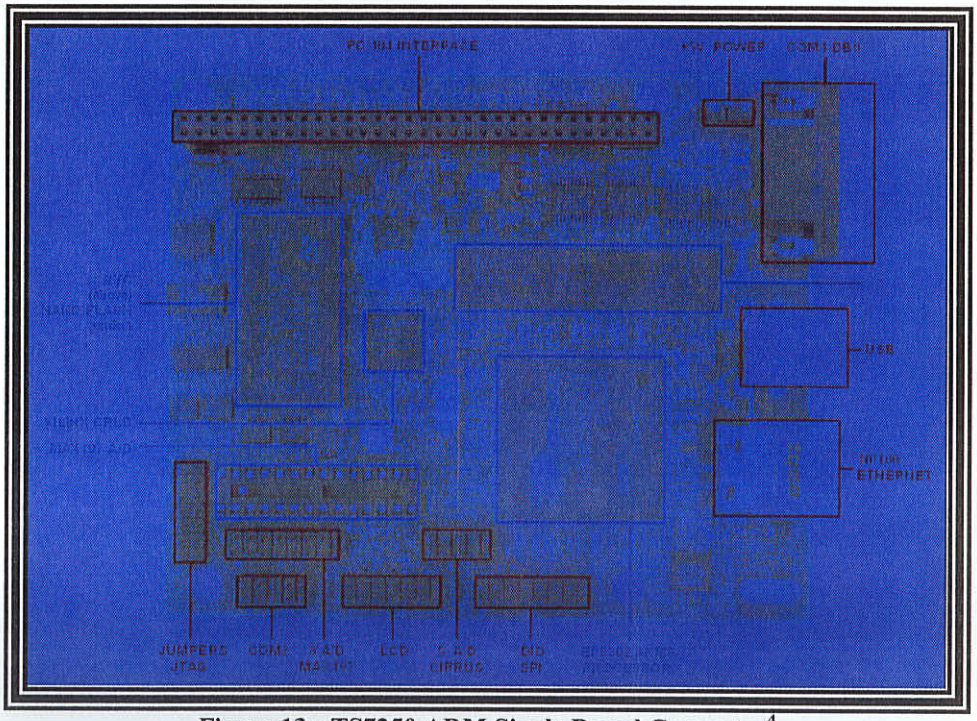

Figure 13 : TS7250 ARM Single Board Computer[4]

[4] Taken from ts-7250-manual-rev2.2.pdf
http://www.embeddedarm.com/Manuals/ts-7250-manual-rev2.2.pdf

The Ethernet port was used to communicate with the Robot Server (see Chapter 4). Basically the Robot Server resides on the laptop on the robot and provides a gateway for commands to be relayed from the remote operator to the motor controller. The ARM processor EP9302 [16] has two 16-bit timers and one 32-bit timer. In this project the two PWM signals were generated with the help of these three timers. These PWMs and other control signals were sent to the motor driver circuit via the DIO (SPI) port (see Figure 13). Additional details will be discussed in the following sections.

The ARM SBC board accepts 5V DC power without any built-in protection features. The ARM SBC TS-7250 supplier also offers an optional metal casing with a built-in power supply, which operates between 8V DC and 30V DC. The ARM SBC board and the metal casing were purchased for this project.

3.1.3 Motor Driver

Due to the weight of the chassis and the type of the motors used, it was determined that the motor drivers would have to provide 30A of steady current at 24V DC. As such, for a first attempt we designed a motor driver that was capable of providing 30A of steady current at 24V DC. But several problems arose when trying to manufacture it. Due to the high amount of current flowing through the circuit, the PCB required a 4oz. copper laminated board. It was not possible for us to find 4oz. laminated copper board, however we were able to find 2oz. copper plated board. Using a bench top milling machine a prototype of the circuit was constructed. This approach proved too costly in terms of time, effort, and money to produce multiple controllers. So a decision was made to find a motor driver that was available off the shelf. This limited the design alternatives to the components and protection circuits. This approach, greatly improved the reliability of the motor controller while still requiring the design and implementation that would meet our specifications.

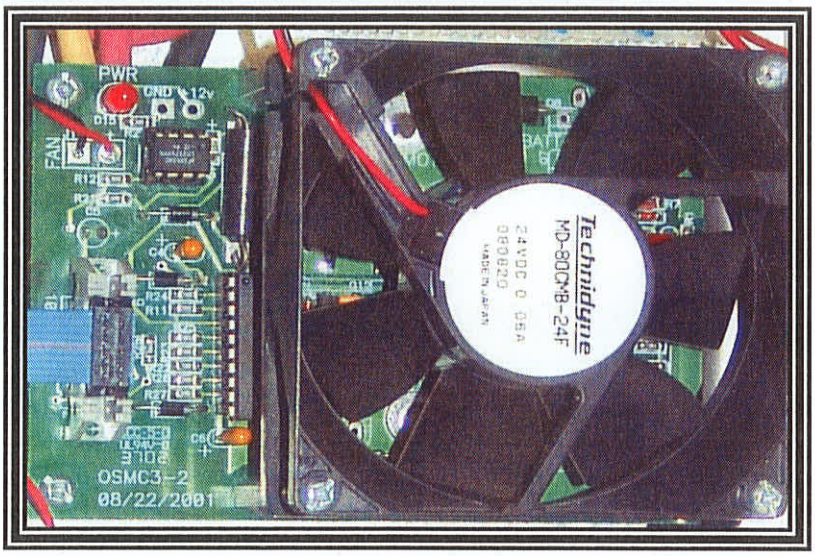

Figure 14 : OSMC Motor Driver with cooling fan

After searching for a number of different motor drivers, we found one that was close to what we need but not exactly what we have been looking for. It was decided to customize it. After contacting the manufacturer, they agreed to sell us the Printed Circuit Board (PCB) without the parts. This was an attractive option for us; given the fact that we had made a number of unsuccessful attempts to manufacture a reliable motor driver. The controller is an open source motor controller (OSMC) (see Figure 14). The PCB and the corresponding customized parts were purchased separately and assembled together to meet the requirement. Each motor driver circuit contained more than 150 through hole and surface mount components.

3.1.4 Logitech QuickCam camera

For the vision system a Logitech QuickCam (see Figure 15) camera was selected for video capture. The camera is manufactured by Logitech Inc. The QuickCam is equipped with a CMOS image sensor with a still image resolution of 640x480, and can operate on both Windows and Linux platforms. Although not ideal, the camera was considered sufficient to provide the front view from the robot. In order to get a better view, it will be necessary to use a better camera with a wider field of view.

Figure 15 : Logitech QuickCam

3.1.5 Logitech MOMO steering wheel

The Logitech MOMO steering wheel (see Figure 16) is a control device which works similar to a joystick by providing control of three different axis namely x, y and z. In this case x-axis controls are derived from the steering movement from the left to the right providing counts -1000 to +1000 respectively. The y axis controls are derived from a throttle and controls the wheel velocities through the use of counts from 0 to +1000. It also has a number of buttons which could be used to trigger some other functionality to control the robot's arm for example. Another feature of this device is to provide force feedback through the use of its actuator. Using the feedback received from an accelerometer it is possible to generate the vibration felt by the chassis.

Figure 16 : Logitech MOMO Steering Wheel

3.2 *Software*

In this project, software was written in various languages. The cockpit GUI is a user interface to control the robot from a remote location. The GUI was written in C# with the help of Microsoft DirectX [12]. The development platform used to develop and compile the C# code was Microsoft's Visual Studio 2005. The dials and gauges were part of the demo software components provided by Dundas software [11]. The Robot Server was written in C# to run on the Windows XP environment, the motor driver software to produce PWM was written in C to run on Linux. The interface between the Robot Server and the ARM SBC is written in C++. All of the C and C++ code were edited on a PC and compiled using cross-compilation tools provided by Technologic System Inc [9]. The Linux environment was simulated on a PC using cygwin [13].

The use of a number of development environments is both time saving as well as challenging. An additional difficulty faced was associated with the limited documentation that was available for the ARM SBC.

3.3 *Summary of Chapter 3*

This chapter described the available resources for this project such as the Hardware components and Software tools. The major starting point was an old chassis of a motorized wheel chair. It was equipped with two powerful DC motors without functioning motor drivers. The PCBs for the motor driver circuits were purchased and customized them to meet the requirements of our relatively large platform. To interface with the motor drivers, an ARM SBC was purchased. A Logitech steering system was used as a control device. A Logitech QuickCam camera was purchased to be used as a video capture device. Microsoft Visual Studio 2005 was used to edit all the code written. Microsoft's compiler was used to compile the remote control user interface and the Robot Server software, which was written in C# language. They were written for the Windows XP operating system. Remaining code was written in order to run on a Linux platform. They were written in C and C++ language. Cigwin's cross compilation tool was used to compile this part of the code.

Chapter 4 : The System Design

CONTENTS

Figure 1 illustrated the basic taxonomy of the system and the Figure 2 the system level diagram including different software and hardware system components in detail. The overall control of the system has two main sections, the Cockpit and the Mobile Unit residing on the remote control PC and Robot respectively.

4.1 The Cockpit

The Cockpit is the remote user interface for the robot or mobile platform. It consists of a group of hardware and control software modules. The required hardware is as follows:

- A desk top computer,
- A monitor,
- A force-feed-back joystick or a force-feed-back steering wheel system,
- A key board and a mouse.
- Wireless interface (802.11)

The software used to capture the user inputs and send them to the mobile unit through the 802.11 wireless network is called the Mission Control module (Figure 17). The receiver is named the Receptionist and receives video and sensory data feedback from the mobile unit and sends them to the appropriate hardware devices so that the operator can realize the robot's actual environment.

4.1.1 The Cockpit Software Architecture

The Cockpit software (see Figure 17) has two main sections; The Mission Control and the Receptionist. The Mission Control module captures the user input such as the robot's intended speed, direction, breaking information, emergency stop commands, and the operational directives for the robotic from a joystick or a similar device. At present the user input is derived from either a mouse or a Logitech steering system. These directives are then sent to the mobile unit as appropriate commands for execution. The Receptionist handles the feedback received from the mobile unit. It then displays the video, processes sensory data such as sonar, GPS and/or compass data and displays this information on the monitor. If available it will also processes the data from accelerometers and sends it to the force-feed-back device so that the user can feel the vibrations felt by the robot chassis.

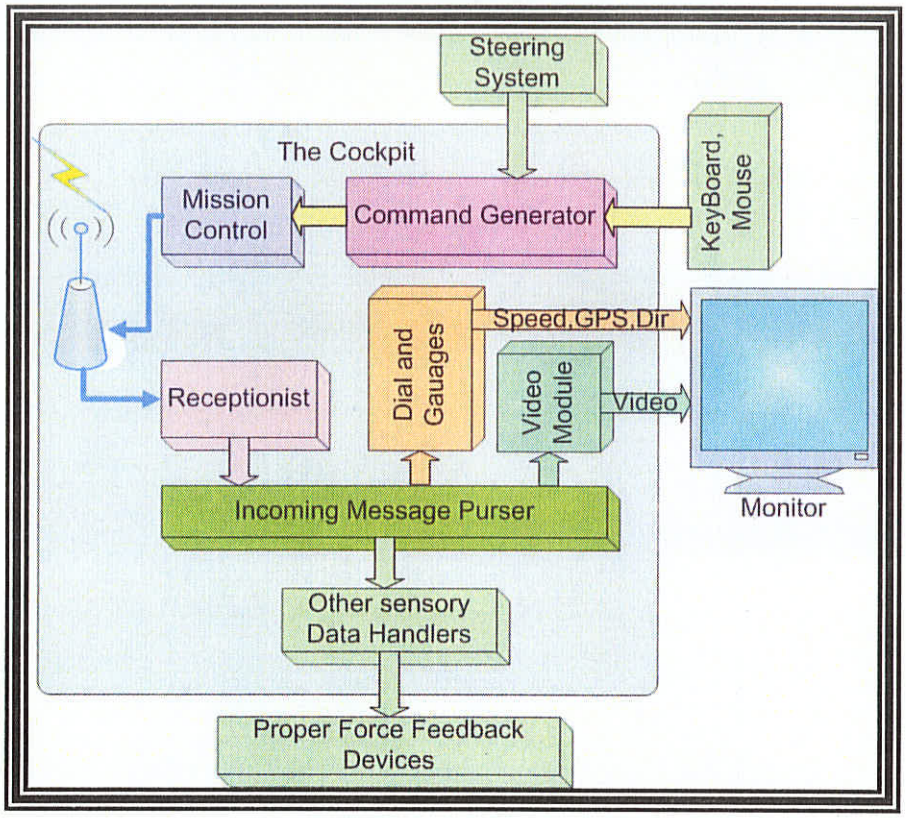

Figure 17: The Cockpit Architecture

4.1.1.1 The Mission Control

This section describes the message structure, the detail command set and the communication protocol of the Mission Control module. As mentioned , Mission Control captures the user input from the Logitech steering system (see Figure 19) and sends directives to the mobile unit as messages which can encapsulate up to three different commands (see Figure 20). The Mission Control samples the user input ten times a second which was determined empirically to be a sufficient rate for controlling the mobile. The tenth of a second is well below other delays inherent in the system. The input sampling is done by creating a software interrupt after every one hundred milliseconds. This timer interrupt is not a very accurate one, but it is sufficient to serve our purposes. The structure of the command execution is shown in Figure 18.

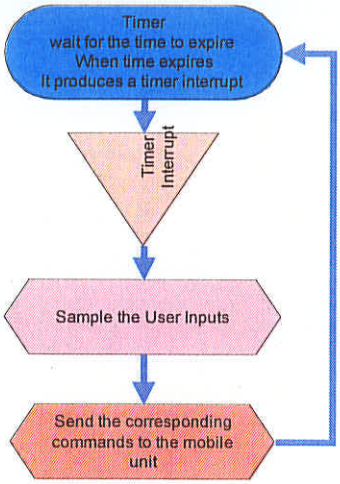

Figure 18 : The command execution diagram using a software interrupt

As shown in Figure 19, the Logitech steering system consists of two main parts. The top part contains a steering wheel, a stick shifter, one left paddle, one right paddle and six other push buttons starting from button3 to button8. The bottom part contains an accelerator pedal and a brake pedal. The summary of the functionalities of the different elements of the steering system are shown in Table 3.

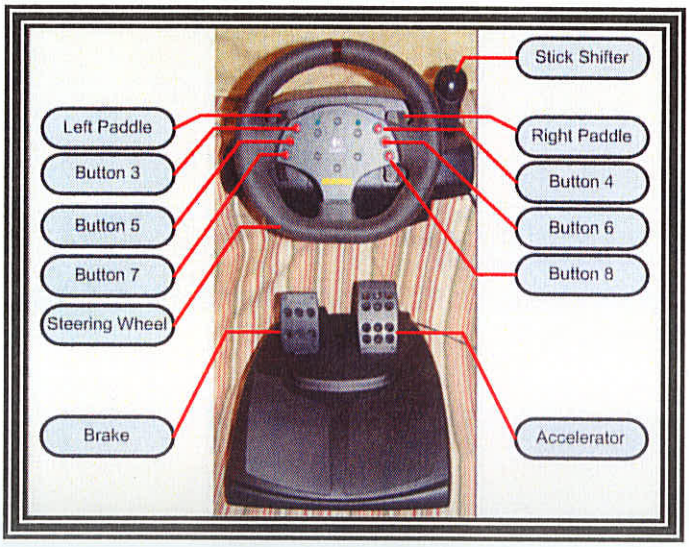

Figure 19 : Logitech Steering system

The control components to drive the Robot:

The following components and their functionality are described below. Table 3 provides a listing of the control functions and the degree to which they have been implemented in the current prototype. All the following functions were first implemented using the mouse as an initial input device. As such the platform can be controlled with the mouse alone or using an input device such as the Logitech steering system. An advantage of the command protocol developed is that it also makes it possible to script a routine for the mobile to follow without any physical input device at all. Although not of central importance here the scripting may be useful as the platform evolves to being more autonomous.

The Emergency Stop:

The Emergency Stop function is provided through Button3 of the steering system (see Figure 19). It applies a break function to both of the DC motors connected to the drive wheels causing the robot to stop immediately. The emergency stop control is crucial to the platform as it has considerable inertia and would be capable of inflicting serious injury or damage if it were to collide with an object while operating at speed. This is also an example of a command that the platform should also be able to generate locally in the event a collision can be inferred. The top level command format is detailed in the following.

The Steering Wheel:

The robot has two independent drive wheels. In the current architecture the Mission Control part of the software reads the steering position of the steering system. The steering wheel (see Figure 19) position is a number ranging from the left most position -1000 to the right most position +1000. This number provides the intended direction with respect to the robot's current direction. As explained earlier the steering is done through controlling the difference in the speed of each of the two wheels. Using the following equation one can calculate the required speed of the individual wheel.

$$\theta(t) = \frac{(vr - vl)t}{d} + \theta o$$

Where $\theta(t)$ represents the robot's angular position with respect to time, vr and vl represents the velocity of the right and the left wheel respectively, d is the distance between the two drive wheels, t is the elapsed time since the robot started to move and finally the θ_0 is the initial angular position of the robot. The angle presented here is measured in radians. A C# dial gauge component from the Dandus Software is used to display the current steering position within the Cockpit frame work.

Driving the Robot	
Name of the Control	Function (Fully Functional)
Steering Wheel	To steer the robot to the left or to the right
Stick Shifter	Sets the position of the transmission
Accelerator Pedal	Controls the speed of the robot
Brake Pedal	Controls the brake
Button 3	Emergency Stop
Operating the Three-Axis Arm (Not Implemented)	
Left Pedal	Moves the arm to the left
Right Pedal	Moves the arm to the right
Button 5	Moves the arm forward
Button 6	Moves the arm reverse
Button 7	Moves the Drill Head upward
Button 8	Moves the Drill Head downward
Button 4	Start/Stop the drill

Table 3 : Functionalities of the steering system

The Stick Shifter:

The DC motors are connected directly to the drive wheels. Therefore the transmission functionalities are provided through software. Rotate-Right, Drive-Forward, Neutral, Park, Drive-Reverse and Rotate-Left are the positions of the transmission control (implemented in software). The steering system's (see Figure 19) stick shifter (in short denoted "shifter") is used to change the transmission position. It works similar to a mouse. Pushing the shifter forward provides the forward click and pushing it reverse provides the reverse click. With the help of local variables, one can easily implement some simple events using these "mouse" clicks and determine the current position of the shifter. Using slider gauges the current transmission status is displayed on the screen. The next paragraph explains the transmission functionalities in detail.

a. ***Rotate-Right:*** The feature allows the operator to rotate the robot 360 degree clockwise at a "0" degree radius. To achieve this goal both of the drive wheels must turn at the same speed but the left wheel turns to forward direction while the right wheel turns to the reverse direction. During this operation, the steering function is disabled. However the acceleration function is operational to allow the operator to control the speed in which the robot would be rotated.

b. ***Rotate-Left:*** The feature allows the operator to rotate the robot 360 degree counter clockwise at a "0" degree radius. To achieve this goal both of the drive wheels must turn at the same speed but the left wheel turns to the reverse direction while the right wheel turns to the forward direction. Like the previous case, during this operation, the steering function is disabled. However the acceleration function is operational to allow the operator to control the speed in which the robot would be rotated.

c. ***Drive-Forward:*** In this position the robot drives ahead by rotating both of the drive wheels to the forward direction. The actual duty cycle or the speeds of the individual wheels are determined by the steering position.

d. ***Drive-Reverse:*** In this position the robot drives reverse by rotating both of the drive wheels to the backward direction. The actual duty cycle or the speeds of the individual wheels are determined by the steering position.

e. ***Neutral:*** In this position both of the DC motors are placed in coast configuration. In this situation the robot can move freely with the help of external force like gravity or manual push.

f. ***Park:*** In this position both of the DC motors are placed in Brake configuration. However as there are no mechanical brakes used in this project, Park and the Neutral position exhibit the same.

The Accelerator Pedal:

The acceleration is controlled through the use of the Accelerator Pedal (see Figure 19). If the pedal is not pressed, it provides a value of "0". While it is pressed the position of this control is provided as a number between 0 and +1000. Using the steering position, stick shifter position and this number from the accelerator pedal, one can determine the duty cycle of each of the left and the right DC motors and send the appropriate commands to the mobile unit. It is to be noted that the accelerator pedal and the brake pedal are used to control the same axis. This implies that the value of the accelerator position could be altered using the brake pedal. The detail of this matter is discussed in the paragraph named The Brake Pedal.

The Brake Pedal:

The braking feature is available through the use of the steering system's Brake Pedal (see Figure 19). Its position value ranges from -1000 to 0 and works in conjunction with the Accelerator Pedal. If not pressed it provides a value of "0". As mentioned earlier, the Accelerator Pedal and the Brake Pedal controls the same axis. The final value of this axis is the sum of the Break Pedal position value and the Accelerator Pedal position value. For example, when the accelerator pedal is pressed half way, it shows its

position value equals to "500", and for the brake pedal it is "-500". If both of the pedals pressed simultaneously, it should show the combined value "0". The negative value of this axis should be interpreted as a break command. The magnitude of the negative position value is the breaking duty cycle.

Components to drive the Three-Axis ARM system:

The left pedal, the right pedal and all other buttons behaves like an On/Off switch. The three axis arm system is driven by stepper motors with a fixed speed. Therefore in order to drive them, one has to make the corresponding function call with the appropriate fixed (direction) parameters like drive forward or drive reverse. The present arm has only analog DC motors and as such can be moved into position visually under on/off control. The control of the 3 axis apparatus is not part of the basic platform and only included here for completeness. The provisioning of the control however illustrates how an auxiliary apparatus may be controlled using a device such as the MOMO.

The Left Pedal:

The *Left Pedal* of the steering system is used to move the drill head to the left on the robotic arm platform until it reaches the limit switch. The limit switch makes sure that the drill head does not move outside its assigned travel zone. In order to move the drill head continuously to the left, the operator must lift and hold the pedal until it reaches the desired location.

The Right Pedal:

The *Right Pedal* of the steering system is used to move the drill head to the right on the robotic arm platform until it reaches the limit switch. The limit switch makes sure that the drill head does not move outside its assigned travel zone. In order to move the drill head continuously to the right, the operator must lift and hold the pedal until it reaches the desired location.

The Button5:

The *Button5* of the steering system is used to move the drill head to the forward direction on the robotic arm platform until it reaches the limit switch. The limit switch makes sure that the drill head does not move outside its assigned travel zone. In order to move the drill head continuously to the forward direction, the operator must press and hold the button until it reaches the desired location.

The Button6:

The *Button6* of the steering system is used to move the drill head to the reverse direction on the robotic arm platform until it reaches the limit switch. The limit switch makes sure that the drill head does not move outside its assigned travel zone. In order to move the drill head continuously to the reverse direction, the operator must press and hold the button until it reaches the desired location.

The Button7:

The *Button7* of the steering system is used to move the drill head to the upward direction on the robotic arm platform until it reaches the limit switch. The limit switch makes sure that the drill head does not move outside its assigned travel zone. In order to move the drill head continuously to the upward direction, the operator must press and hold the button until it reaches the desired location.

The button8

The *Button8* of the steering system is used to move the drill head to the downward direction on the robotic arm platform until it reaches the limit switch. The limit switch makes sure that the drill head does not move outside its assigned travel zone. In order to move the drill head continuously to the downward direction, the operator must press and hold the button until it reaches the desired location.

The Button4:

The *Button4* is used to start and stop the Drill Head. If the Drill Head is idle, the pressing this button will start it and pressing it again will stop. This is done by turning a relay On/Off.

4.1.2 The Message Structure

As mentioned, Mission Control sends commands to the mobile unit to perform desired operation. To ensure that the message received by the mobile unit is a valid one, the mission control sends a *CodeWord* with every message it sends. It could be a fixed string or a dynamic assigned one. The only rule is that both the sender and the receiver of the message must know it ahead of time. In the current implementation it is a fixed string. But in the future work it should be implemented as a dynamic one. The message also contains the valid sender and a valid receiver. The Application Layer Message format is given below:

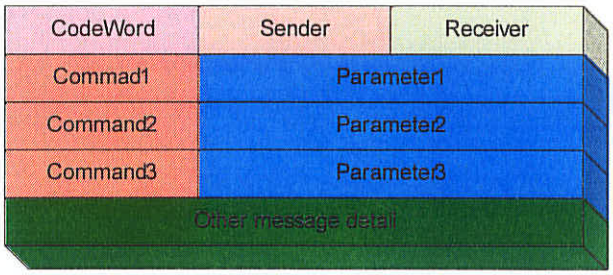

Figure 20: Mission Control Command Format

The Mission Control implements a total of fifteen commands. Each of the commands has a name and the associated parameter. The detail description of the commands and their formats are given below:

1. **`Login`:**

   Description: This command allows the operator to log on to the Robot server. The User_id provided to the mobile unit to make sure that the only authorized users can log on to the system.

   Command Format:

2. **Logout:**

Description: This command allows the operator to logout from the Robot server. The User_id provided to the mobile unit to make sure that the only the current user who was logged in can logout of the system

Command Format:

3. **EmergencyStop:**

Description: This command immobilizes the robot immediately. The User_id provided to the mobile unit to make sure that the only authorized users can apply the **EmergencyStop** command.

Command Format:

4. **Reset:**

Description: This command stops all the robot functions by applying the **EmergencyStop** command and then resets the entire mobile unit. The User_id provided to the mobile unit to make sure that the only authorized users can apply the **Reset** command.

Command Format:

5. **Hello:**

Description: This command sends a simple hello message to the mobile unit. If the mobile unit is functioning properly, then it sends a "*Hello there, I am fine*" message back to the Mission Control.

Command Format:

6. **SetPWMFrequency:**

Description: This command sets the PWM frequency for the motor driver circuit.

Command Format:

7. **GetPWMFrequency:**

Description: This command retrieves the current PWM frequency from the system.

Command Format:

8. **LeftMotorForward:**

Description: This command drives the left motor forward at the desired duty cycle.

Command Format:

9. **LeftMotorReverse:**

Description: This command stops all the robot functions by applying the **EmergencyStop** command and then resets the entire mobile unit.

Command Format:

10. LeftMotorCoast:

Description: This command coasts the left motor

Command Format:

11. LeftMotorBrake:

Description: This command applies brake to the left motor with the desired duty cycle.

Command Format:

12. RightMotorForward:

Description: This command drives the right motor forward at the desired duty cycle.

Command Format:

13. RightMotorReverse:

Description: This command drives the right motor to the reverse direction at the desired duty cycle.

Command Format:

14. RightMotorCoast:

Description: This command coasts the right motor

Command Format:

15. **RightMotorBrake:**

Description: This command applies brake to the right motor with the desired duty cycle.

Command Format:

4.1.2.1 The Receptionist

The Receptionist is part of the cockpit user interface which deals with the feedback data received from the mobile unit. The Receptionist receives command acknowledgements, video, sensor data such as sonar, GPS data, and compass data and displays it on the monitor. Once the Receptionist has received data from the mobile's sensors such as from the accelerometer and from bumper sensors, the Receptionist sends them to the force feedback device connected to the desktop computer (the actual controls). The Receptionist also implements a number of dials displaying the direction, speed, duty cycle of each motor etc.

4.2 The Mobile Unit

The mobile unit has a flexible architecture (see Figure 21) based on Ethernet LAN. In this architecture all of the main controllers must have Ethernet connectivity and are connected to an Ethernet switch in a LAN configuration. It is possible to connect other controllers to the main controller provided the fact that they can communicate with each other. One host PC called the Robot Server is connected to an Ethernet switch. It is also connected to an 802.11 wireless network (Access Point) through its wireless network access card.

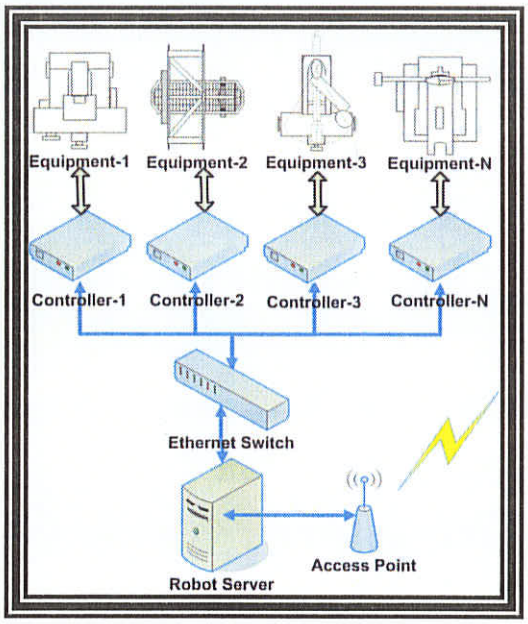

Figure 21: Mobile Unit Architecture

The Robot Server acts like a bridge between the remote control platform and the mobile unit. The Robot Server receives high level commands from the control PC over the 802.11 wireless link and forwards them to the appropriate destination, such as the ARM SBC that is used to control the motors. The Robot Server also captures all the sensory data, video etc. and sends them to the control PC as feedback. The main benefit of this architecture is the ease of expandability. At any time if one decides to add more controllers that have an Ethernet interface; they can be incorporated into the system easily.

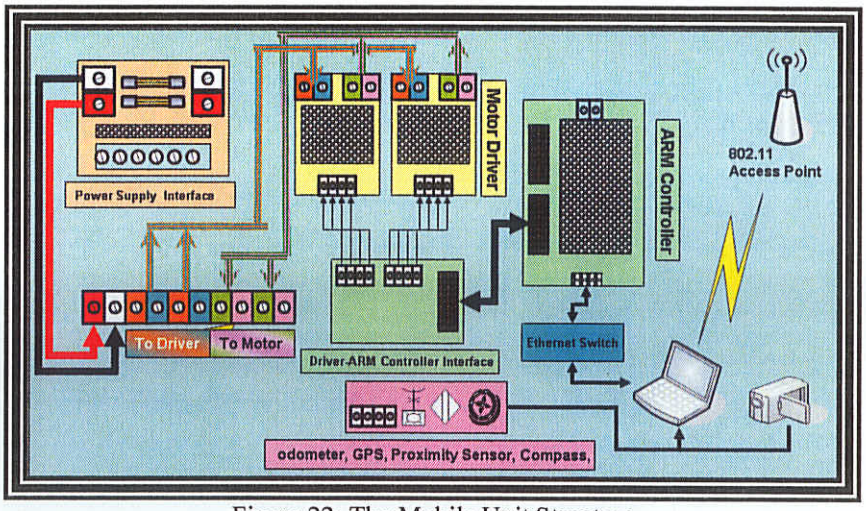

Figure 22: The Mobile Unit Structure

The current mobile unit (see Figure 22) consists of a chassis with powerful two motor driver circuits, an ARM Single Board Computer, a driver-ARM controller Interface, an Ethernet switch, a Webcam and a laptop computer. The platform also contains a power supply interface circuit and two deep cycle marine batteries. The odometer, GPS, compass and the accelerometers are yet to be implemented but can be easily integrated within the basic modular architecture. Beside the hardware components, mobile unit has the following main software components. They are:

- The Message Receiver
- The Message Sender
- The Environment Monitor
- The Collision Avoidance Subsystem
- The Robot Positioner
- The Robotic Arm System Driver

All of the above software components are part of the Robot Server. The ARM (SBC) serves as an interface between the Robot Server and the two motor driver circuits. This is necessary because of the fact that the motor driver circuits are primitive ones. The motor drivers require the direction and PWM signals as inputs and do not have an Ethernet interface. This problem is solved through the use of the ARM SBC. The ARM SBC receives the top level commands from the Robot Server and converts those commands to low level commands that the motor driver understands. The details of the different software components of the mobile unit are described next.

4.2.1 The Software Design

4.2.1.1 The Robot Server

The Robot Server is one of the most important parts of the mobile unit. It works as a bridge between the remote control PC and each of the individual electrical components of the mobile unit. The architecture of the robot server is shown in Figure 23. After booting up the system, the Robot Server determines which components are available on the mobile unit and initializes them as necessary. Then it waits for the commands from the remote control unit. The Robot Server's Message Receiver module

captures the messages sent to the Robot Server and its Message Sender module sends feedback to the remote operator.

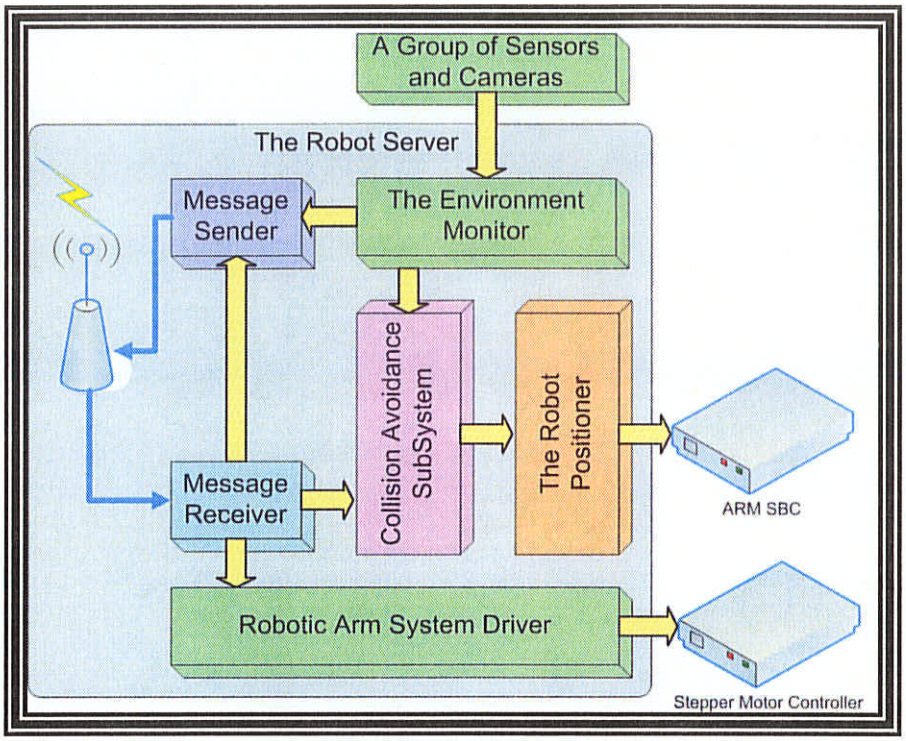

Figure 23: Architecture of the Robot Server

The collision avoidance subsystem assists the Robot Server in avoiding any potential collisions with the objects located in the environment. The Environment Monitor provides input to the collision avoidance subsystem in making its decision. In this manner the Environment Monitor may detect or infer a potential collision and send a message to the Message Sender. At this time depending on the message severity a message may be sent to the remote operator or directly to the Collision Avoidance Subsystem circumventing the remote operator and any delays that would have arisen from relying on direct operator intervention.

4.2.1.1.1 The Message Receiver

The Message Receiver captures messages sent by the remote control unit. Before sending commands, the remote operator must log on to the Robot Server with a valid user id and password. Once the authentication is complete, the Robot Server is ready to take

commands. After receiving messages from the operator, the message receiver verifies them through the CodeWord, sender, receiver etc. and forwards them to the proper destination. For example, if a message contains commands to drive the robots wheels, then it will forward the commands to the Collision Avoidance Subsystem, where they are forwarded to the Robot Positioner module. If the message contains commands to drive the robot's arm system, then it will forward the commands to the Robotic Arm System Driver.

4.2.1.1.2 The Message Sender

The Message Sender sends messages generated by the Message Receiver back to the operator. Even though it does not support any type of Quality of Service (QoS), depending on the condition of the network status, the Message Sender can adapt the video frame rate while transmitting the video. The Message Sender also sends the sensor data to the operator periodically. This feature is not fully implemented in the current design but provided here for the sake of completeness.

4.2.1.1.3 The Environment Monitor

The Environment Monitor receives all the environmental information from various devices and forwards them to the Message Sender. The Message Sender in turn forwards that information to the remote operator. For the architecture designed here the Environment Monitor also determines the distance of the closest obstacle in the direction of travel and sends them to the collision avoidance subsystem. The Environment Monitor classifies the distance of the closest objects into three zones. The distance is measured in seconds to reach the obstacle at the current speed. This is because of the fact that traveling at different speed requires different stopping distances. The summary of this information is given in the following table. These are implicit linguistic variables that can be used with many non linear control strategies. In the case here for illustration they are inputs to a simple rule based if-then decision process. In our case they either alert the operator or more aggressively assume direct control of the mobile. As they are based on an estimate of time to collision they are extracted from estimates of distance and speed.

Here again more refined membership would be required but are sufficiently illustrative for demonstrating the platform and architecture. The type of inference here is best described as a Fuzzy rule based inference such as that implemented in Mathematica [17].

Distance from the robot in second(s)	Zone
More than 3 seconds	Too far
Between 2-3 seconds	Medium distance
Less than 2 seconds	Too Close

Table 4: Zone Determination Table

4.2.1.1.4 The Collision Avoidance Subsystem

The robot platform is not an autonomous one and is designed to be driven remotely with the help of a vision system over 802.11 wireless networks. The vision system consumes a nonzero amount of time to process and transport video over the wireless network. From experiments done it is known that, depending on the network conditions, this delay could exceed several seconds. This is an unreasonable delay and one in which the author is attempting to rectify. Reasons for this excessive delay are uncertain but assumed due to the use of Microsoft's Media Player, its encoder and decoder and inherent buffering. As a result the operator's current view represents the view of the environment that happened some seconds ago. As a result it is very difficult for the operator to operate the robot without worrying about potential collisions with the environment. This obviously could result in a huge degradation in the robot's performance. An early attempt was made to reduce the delay to an absolute minimum using an analog camera over an RF link which unfortunately interfered with the 802.11 to the degree that communication between the remote operator and robot was not possible. At this time we are experimenting with alternative software solutions to the delay problem, the results of which will be included in the final version of the thesis. In any event, although the video delay necessitated the use of an auxiliary sensor system, having some onboard processing of sensor information and operator assistance was one of the original objectives of the thesis project.

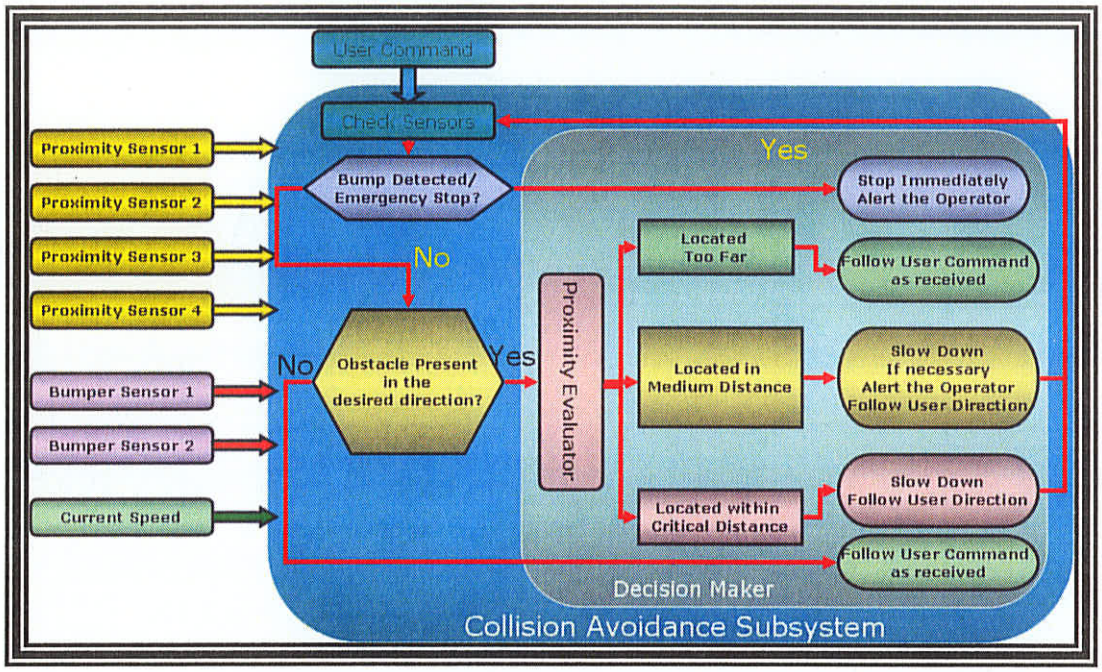

Figure 24: The Collision Avoidance Subsystem

A Collision Avoidance subsystem is required to fill this performance gap. The architecture of the Collision Avoidance subsystem is shown in Figure 24. It uses four proximity sensors on the four sides of the robot to determine the closest obstacle. There are a number of different proximity sensors are available in the market. Laser range finders are the most accurate ones to measure distances but very expensive (at least for the moment). Another option is to use the ultra sonic sensors, however, if the object is located within a foot or so they will fail detect it. To fill this information gap a number of bumper sensors are necessary. The bumper sensors are triggered when they come in contact with an object. For this project two sets of these sensors are required where there are three sensors in each set. Three sensors are placed in front of the robot's front bumper and the other three are place on the rear bumper. The subsystem also needs to know the current speed and the direction of travel of the robot. This is ideally done through the use of encoders. It may also be necessary to reduce speed while traveling on a bumpy surface. A number of accelerometer can assist in serving this purpose. In lieu of encoders the speed can be inferred by the duty cycle supplied to the motor controllers.

The decision maker receives the user commands in terms of speed and direction of the travel from the message receiver. It also receives all the necessary sensor inputs.

As shown in Figure 23, first it determines if any of the bumper sensors is triggered. If so, it stops the robot immediately by sending the emergency stop command to the robot positioner. If none of the bumper sensors are triggered, it then attempts to determine the location of the closest obstacle. There are only three types of location category considered,

a. The object is too far or no object is detected with in the range.
b. An object is located but at a medium distance.
c. The object is too close.

This information is provided by the Environment Monitor. For case (a) the decision maker simply forwards the user command to the Robot Positioner for execution. For case (b) the decision maker determines the speed to be reduced keeping the direction of travel the same and sends the user command with the modified speed parameter to the robot positioner for execution. The decision maker also notifies the operator of this action. In the case where the object is too close to the robot, then the decision maker reduces the speed to a safe one such that the robot can approach the obstacle very slowly and make a soft contact with it until the operator chooses to change the direction. The operator also gets notified that such action has taken place.

4.2.1.1.5 The Robot Positioner

The Robot Positioner receives commands from the obstacle avoidance subsystem and forwards those commands to the ARM single board computer for execution. It also maintains communications to the ARM SBC through the Ethernet LAN.

4.2.1.1.6 The Robotic Arm Driver

The Robotic Arm driver sends the commands related to the robot's three axis arm system to the stepper motor controller. The current stepper controller does not have the Ethernet interface instead it takes input from the parallel port of a PC. Therefore it is necessary for this software component to translate the top level command to the

corresponding low level stepper motor driver command and send the using the PC's parallel port. Ideally as we acquire additional single board computers, these will be used as intermediate command interpreters.

4.2.1.2 The PWM Generator

The PWM generator was originally called the DC Motor Controller in short denoted the DCMC. The PWM Generator receives the top level commands from the Robot Positioner (see Figure 22) and produces the PWM signal with the corresponding duty cycle. Then it sends the produced PWM signal along with the direction parameter to the corresponding motor driver. The PWM generation method used in this module is explained in detail in chapter 2.3.1.2.

4.2.2 The Hardware Design and Device Selection

As mentioned earlier, the wheel chair chassis used in this project weighted about 100 pound including the two marine batteries. The structure was capable of supporting about 250 pounds. With the total weight (the vehicle and the payload) approximately 350 pound, it employed two powerful DC motors to drive the system. Each motor was capable of drawing about 30A (Amperes) of current at 24V (Volts). Thus it was necessary to use motor drivers that could supply that amount of current to the motors.

4.2.2.1 Selection of the switching devices

The customized OSMC motor driver uses eight IRF1404 N-Channel Metal-Oxide Semiconductor Field-Effect Transistors (MOSFET) as opposed to four. The H-Bridge circuit consists of four switching segments or devices. In order to increase the current carrying capacity, this customized OSMC circuit employs two MOSFETs per switching segment or leg. As each of the MOSFET's could potentially carry approximately 70A of current at 150° Celsius [18], the combined current carrying capacity of this circuit would be a maximum of 140A.

H-Bridge circuit provides the capability of controlling analog devices with the help of digital control signal. However, the control signal must not constitute a short circuit scenario. Also the switching devices could be destroyed by voltage spikes coming from the connected inductive loads. Therefore in order to build a stable H-Bridge circuit, one must necessary steps to protect the circuit. The OSMC incorporated many of these features. The customization of those features is given below:

4.2.2.2 The protection against gate-source voltage spike

In this project, the MOSFETs are used as switching devices. They are turned on or turned off providing the corresponding gate voltages. If the gate currents are not controlled properly, it could damage the MOSFETs within less than a second or so. A 150Ω resistor was placed in series to the gate to control the gate current (see Figure 10).

4.2.2.3 The protection against shoot-through

To guard against a potential shoot-through, the original OSMC circuit contained a Scotty diode LLSD101ACT. This was sufficient to create a bypass path while the corresponding MOSFET is discharging (see Figure 10).

4.2.2.4 The protection against gate-source voltage spike

The transient Voltage Suppressor (TVS) diode can suppress voltage spikes. For the IRF1404 MOSFET, the maximum allowable voltage V_{GS} is 40V. Any voltage above 40V needs to be clipped. The 1.5KE33CA is made by On-semiconductor [15]. This diode will normally clip any voltage above 33V for 1 millisecond up to 1500W (see Figure 10). These are connected between each of the motor terminals and the ground for the desired protection.

4.3 Summary of Chapter 4

Chapter 4 summarized the design of the platform. This includes the remote control, mobile unit, command structure, and provided more detail on some of the more critical component selections.

Chapter 5 : Results and Discussions

CONTENTS

5.1 The Motor Driver Circuit

The motor driver circuit was assembled successfully and it met all the operational requirements (see Figure 25). The circuit was tested under a variety of scenarios with 20Amp fuse and functioned as expected. The circuit designed and developed was able to drive the robot motors under no-load and load conditions. The motor controller also worked as expected when a payload of 200 lbs was added to the chasis. In theory it should support up to 80Amps of steady current and about 160Amps of spike for up to a minute. This is about four times the current required to operate the mobile on a flat surface.

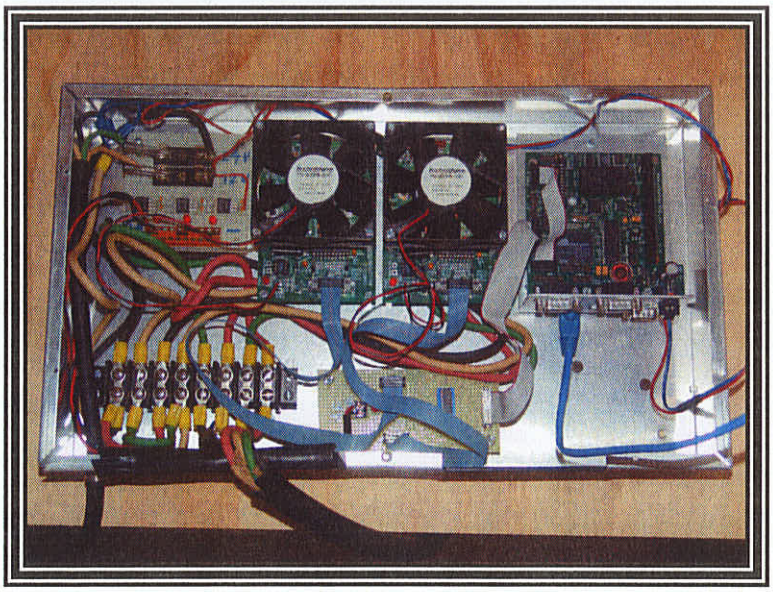

Figure 25 : Robot controller circuit boards in their metal case

5.2 Basic robot operation

The initial goal was to build a robot where an operator could control the robot through an 802.11 wireless communication network. This objective was successfully completed (see Figure 26). A user interface was designed as shown in Figure 27. A vision system assists the operator in the navigation operation. As there is a delay in processing and video and in its transmission over the Internet, this could result in potential collision between the robot and its operating environment. As such, a collision avoidance subsystem was designed to solve the collision problem or at least mitigate it. To get better feel of the environment, a number of sensors were included in the design which would provide information like vibration felt by the chassis, the speed of the robot, location, and the direction. However due the lack of funding for the project, these feedback features were not fully implemented but were provisioned on the platform illustrating the basic platform functionality and its extendibility.

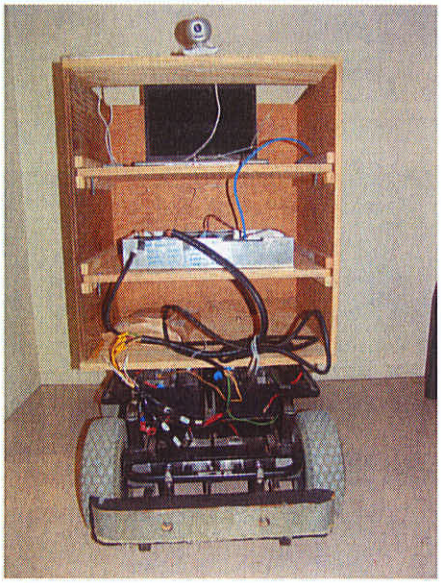

Figure 26 : The completed Robot

5.2.1 Basic Robot Maneuvering Operation

The robot was taken to the nearest parking lot for demonstration and testing. A mouse was used as a control device to maneuver the robot. Even though we were able to control the robot with the mouse, it was quite difficult for the operator to operate it smoothly in terms of estimating the exact turning angle, determining the proper stopping distance etc. This was due to the fact that the turning angle was calculated on a linear scale without considering the speed of each of the wheel and was displayed on the screen. This gave the operator the impression that the robot was turning at a certain angle, but in reality it was different. The video delay also hampered the operator's sense of where the robot was heading. Shaft encoders could be used to better determine the speed of the robot. Thus the determination of the turning angle and the speed would be somewhat closer to the real one. A steering system would give the operator better control of driving the robot similar to driving a car. It would still be somewhat difficult to steer as opposed to a car due to the fact that the Robot had two drive wheels and two casters.

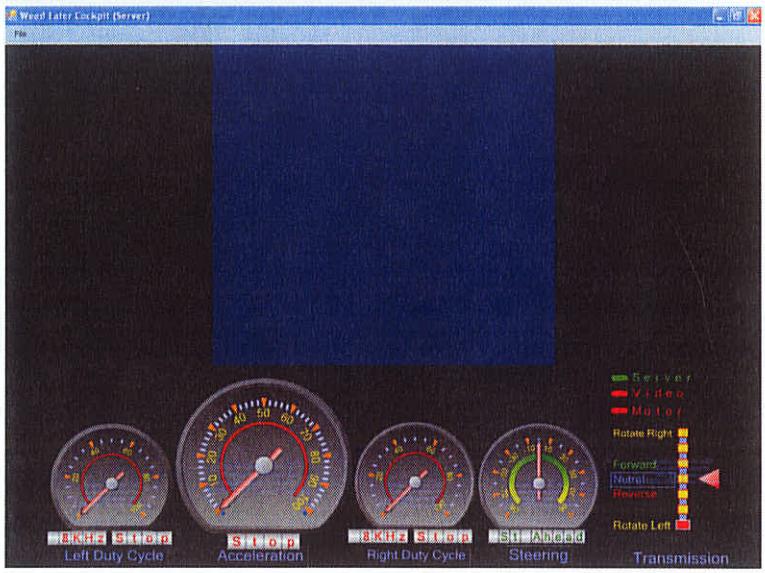

Figure 27: the Cockpit Control Panel

5.2.2 The vision system

Initially the vision system was implemented using the Microsoft Media Encoder 9. It displayed the image captured by the onboard camera, however there was about 10-12 seconds delay between the actual event and the one displayed on the screen. Efforts were made to reduce the delay by reducing the frame buffering to the minimum allowed by the tool which is one frame. However it only reduced the delay down to between 6-7 seconds.

The next attempt was made using the video conferencing application named VIC [19]. VIC is developed by the Network Research Group at the Lawrence Berkeley National Laboratory in collaboration with the University of California, Berkeley. It reduced the delay down to between 2-3 seconds but with a very small frame size (approximately 100x100 pixels). It was enough to operate the robot with extreme care to avoid collision with the objects in the operating environment. Even though this delay seems to be higher than expected, it is possible to reduce the delay by implementing the video encoding in hardware or using faster machine.

Another attempt will be made to improve the video transport delay using Motion JPEG. No matter how small the delay could be, in designing the systems to be controlled over the Internet, one must consider the safety factors required to operate it safely in a

dynamic environment. This could be done through the use of obstacle avoidance subsystem such as the one developed here.

5.2.3 The Robot Server

A Robot Server was designed to maintain necessary communications between the remote control operator and the robot. It successfully completed all the necessary functions to operate the robot remotely.

Initially the robot was mounted on a stand such that the wheels could rotate freely while the testing of the motor functions in progress. The tested functionalities were

a. **Drive Left Motor Forward**
b. **Drive Right Motor Forward**
c. **Drive Left Motor Reverse**
d. **Drive Right Motor Reverse**
e. **Rotate Left**
f. **Rotate Right**
g. **Brake**
h. **Coast**
i. **Emergency Brake**

All of the functionalities mentioned above worked as expected. After that the Robot was operated within my apartment where the operator was able to maneuver the Robot easily on the smooth surface. Then the Robot was moved to a nearby parking lot for field testing. In terms of receiving and sending commands over the 802.11 wireless network, the system performed as expected, however the maneuverability was not exactly as we were expecting. The castor wheels were sometimes redirecting the Robot to an unintended direction. Also this was due to the fact that there was no feedback mechanism to control the motor's speed accurately causing the motor to turn at different speed with the same duty cycle on surfaces with different friction parameters.

5.3 Summary Chapter 5

This chapter described the results from the Robot Platform implementation. The platform architecture was a robust one and the ease of integration of different components in the system proved the point.

On a smooth surface the Robot performed as expected. However on the rough surfaces the robot had difficulties following the direction properly because it was being redirected by the castor wheels. This behavior was anticipated ahead of time and necessary steps should be taken to correct the problem in the future work.

Chapter 6 : The Conclusions and Future Work

6.1 Conclusions

For this project, a robot was designed and built with the capability of being controlled remotely over an 802.11 wireless computer network. Even though the design of the system was complete, the whole design was not realized primarily due to the lack of sufficient funds needed to procure hardware. However, the basic platform and architecture could be used to continue the future work. As designing, developing and testing a suitable architecture for a mobile platform was the main objective of the thesis the overall project is deemed a success. The basic architecture as depicted in Figure 21 is robust, flexible, extendable, built upon core technologies albeit not normally associated with the robotic projects.

The completed portions of the design include the complete control of the mobility related components such as the control of the speed, direction, breaking, and the emergency stop. More specifically this included the remote telecontrol utilizing IP over 802.11, the motor controllers and their interface, received from the remote user interface, and the robot server responsible for marshalling commands from the remote control client. In order to have precise controls of the speed and direction, it is necessary to include some form of shaft encoders with the implementation.

The vision system was implemented but did not work as well as expected. There were difficulties in transporting video with only a small amount of delay. The Microsoft Media Encoder was just not suitable for transporting video over the Internet to be used in a real time robot control operation. As a result an experimental video conferencing application VIC [19] was used as a vision system, but it also had a delay of somewhere between two and three seconds. The work is still in progress to implement the vision system using Motion JPEG. In addition, the camera used for the vision system was not

sufficient enough to provide a wide enough view of the environment. Better cameras equipped with wide angle lenses should be used for this purpose.

The obstacle avoidance subsystem was designed and the implementation is still in progress. Given the robustness and the simplicity of the design of a slow moving robot, it is expected to work smoothly, at least in theory as it is a very simple control policy. The results associated with the collision avoidance subsystem will be included as soon as it is completed.

Two powerful DC motor drivers were assembled using PCB designs from an Open Source Motor Controller. The motor controller designed and assembled is a robust design and exceeded all of our expectations. An ARM SBC was used to bridge the communication gap between the Robot Server and the Motor Driver Circuits. The motor drivers required PWM signals as inputs. Using three internal timers from the ARM processor two 8 KHz variable PWM signals were generated. These PWM signals took a significant amount of the CPU power to generate. As a result it was rather difficult to maintain fast communications with the Robot Server. The lesson learnt is that if possible, the PWM signals should be generated using dedicated hardware such as FPGA or similar components.

In summary given the limited means to implement the design, the project was a success and set the foundation for more challenging future work. The architecture is extendable and the robotic platform is suitable as a starting point for additional projects.

6.2 Future Work

As mentioned earlier, the design of this robot platform is a very comprehensive one, but limited funding resulted in a partial implementation. The main functional components were completed and demonstrated. However, there is much more room for additional improvements to the project. Some of these could include:

- The use of a better video processing and transport methods (i.e. use of hardware) for the vision system.
- Improve the obstacle avoidance subsystem by using more accurate instrumentation to measure speed and distance.

- Add additional cameras equipped with wide angle lenses to cover more viewing areas around the mobile unit, and/or provide for controlling the camera of a turret.
- Implement the weed eater apparatus which is currently only partially implemented. At this time a two axis table has been assembled but a redesign is required using stepper motors and suitable encoders.
- Implement a Three Axis Robotic Arm system or similar apparatus such as the one discussed in the thesis as would be remotely controlled using the MOMO.
- Improve the Environment Monitor by incorporating GPS, Compass and Accelerometer for feedback purposes.
- Implement several "supervised" autonomous functions to gain more experience in the algorithms that will be required for future machine learning within the context of personal service robots of reasonable size.

References

[1] S. Thrun. Towards a framework for human-robot interaction. Human Computer Interaction, 19(1&2):9-24, 2004.

[2] J. Kofman, X. Wu, T. J. Luu, and S. Verma, "Teleoperation of a Robot Manipulator Using a Vision-Based Human–Robot Interface", IEEE Trans. on Industrial Electronics, Vol. 52, No. 5, Oct. 2005

[3] S. X. Yang, Hao Li, M. Q.-H. Meng, and P. X. Liu, "An Embedded Fuzzy Controller for a Behavior-Based Mobile Robot With Guaranteed Performance", IEEE Trans. on Fuzzy Systems, Vol. 12, No. 4, August 2004

[4] B. Sofman, E. Lin, J.A. Bagnell, N. Vandapel, and A. Stentz. "Improving Robot Navigation Through Self-Supervised Online Learning", http://www.roboticsproceedings.org/rss02/p04.pdf

[5] IEEE 802.11 Standards Achieve, http://standards.ieee.org/getieee802/802.11.html

[6] Q. Du_, M. Gunzburger†, L. Ju‡, and X. Wang, "Centroidal Voronoi Tessellation Algorithms for Image Processing", J. of Mathematical Imaging and Vision, 24, pp.177-194, 2006

[7] http://en.wikipedia.org/wiki/Pulse-width_modulation (As of Oct 21, 2006)

[8] Barr, Michael. "Pulse Width Modulation" Embedded Systems Programming, September 2001, pp.103-104

[9] http://www.embeddedarm.com/Manuals/ts-7250-manual-rev2.2.pdf

[10] http://en.wikipedia.org/wiki/Differential_wheeled_robot

[11] http://www.dundas.com/Products/Gauge/index.aspx

[12] http://msdn.microsoft.com/directx/

[13] http://www.cygwin.com/

[14] http://groups.yahoo.com/group/osmc (Viewing of these documents requires membership)

[15] http://www.onsemi.com/pub/Collateral/1.5KE6.8CA-D.PDF (As of Oct 21, 2006)

[16] http://www.embeddedarm.com/downloads/Components/EP9301_User_Guide.pdf , page 425 (As of Oct 21, 2006)

[17] http://documents.wolfram.com/applications/fuzzylogic/Manual/8.html

[18] http://www.irf.com/product-info/datasheets/data/irf1404.pdf

[19] http://www-nrg.ee.lbl.gov/vic/

Appendix 1

At this time there are a number of aspects of the thesis that others interested in similar project may be interested in utilizing. These include the motor controllers and the software associated with both the mobile platform as well as the remote control human interface.

All of the code and complete specification of the motor controller will be hosted on the Internet Innovation Centre's web site under the related projects tab. The web site is at www.iic.umanitoba.ca.

Appendix 2

This appendix outlines a brief specification of a more ideal robotic platform maintaining many of the notions of the original but improved through reflection of work completed.

Flexible Robot Platform (FRP)

A Flexible Robotic Platform (FRP) should allow integration of devices that are commonly used in the robotic applications. These include DC motors, Stepper Motors, Servo Motors and various kinds of sensors. In order for this integration to be successful and uniform, the devices to be connected to the FRP must comply with certain guidelines. The next paragraphs will explain the guidelines in detail.

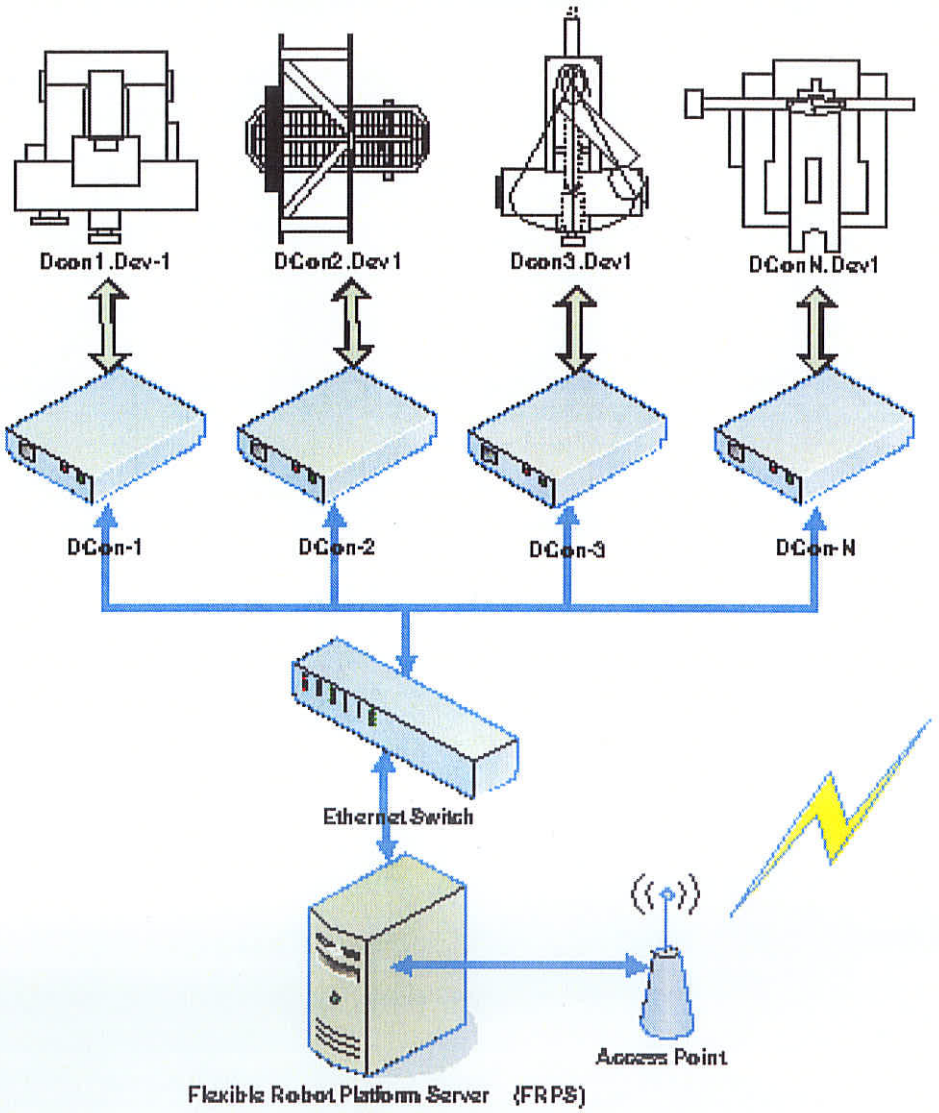

Figure A28: Flexible Robot Platform (FRP)

Flexible Robot Platform Server (FRPS)

FRPS is ideally connected to one or more Device Controllers (DCon) through a LAN onboard the mobile robot. The FPPS would also be connected to the Remote Control Station (RCS) through a wireless communication Link (i.e. 802.11). During the startup period the FRP server must go through a startup sequence in order to establish connections with the available DCons and the RCS. Similarly the DCons must establish connections to the device(s) connected to each one of them. These startup sequences are described in the following paragraphs.

FRPS-DCon connection:

Device Controller (DCon) serves as bridge between the devices connected to it and the FRPS. During start up, the DCon must initialize all the devices connected to it correctly. In order to establish connection to the available device controllers, both the FRP Server and the DCons must follow the following protocol.

1. The DCon opens a known TCP port (i.e. port 40000) for listening to the *DCon_Query* connand from the FRPS and waits.
2. The FRPS opens a known TCP port (i.e. port 41000) for listening to the response from the DCons.
3. The FRPS the sends ten *DCon_Query* commands in 500ms interval to make sure that all of the connected DCons are powered up and can receive the *DCon_Query* commands.
4. As soon as a DCon receives a *DCon_Query* command, it replies to the FRPS with the following information
    a. Device ID
    b. Device Name
    c. Device Type
    d. Available commands
5. FRPS receives these responses and creates a DCon database (i.e. XML file) containing the detail description of the devices that are connected to different DCons.

During the entire duration of the robot operation, the FRPS-DCon connection must remain intact. Under any circumstances if the connection is broken, the DCon must disable the device and wait for the *DCon_Query* command from the FRPS. Similarly the FRPS must notify the Remote Control Station (RCS) about the situation and send the *DCon_Query* command to the DCons. The DCon may reply to the *DCon_Query* command only under two cases:

1. During the startup, and
2. After establishing the new connection between the FRPS and the DCon.

RCS-FRPS Connection:

Once the FRPS is finished creating the DCon database, it is ready to establish a connection to the RCS using the following protocol:

1. First the FRPS opens up another TCP port (i.e. 42000) to listen to the *FRPS_Query* commands sent by the Remote Control Station (RCS).
2. The FRPS opens a known TCP port (i.e. port 43000) for listening to the response from the FRPS.
3. The RCS the sends up to ten *DCon_Query* commands in 500ms interval to make sure that all of the FRPS are powered up and can receive the *FRPS_Query* commands.
4. As soon as an FRPS receive a *FRPS_Query* command, it replies to the RCS with the following information
   a. Device ID(s)
   b. Device Name
   c. Device Type
   d. Available commands associated to each of the devices
5. The RCS receives these responses and creates a database (i.e. XML file) containing the detail description of the devices that are connected to different FRPS.

During the entire duration of the robot operation, the RCS-FRPS connection must remain intact. Under any circumstances if the connection is broken, the FRPS must disable the device and wait for the *FRPS_Query command* from the RCS. Similarly the RCS must send the *FRPS_Query* command to the FRPS. The FRPS may reply to the FRPS_*Query* command only under two cases:

3. During the startup
4. After establishing the new connection between the RCS and the FRPS.

The above protocol will make the platform more robust and easier to manage. Following this suggested protocol will facilitate adding apparatuses in a more flexible manner as less no code modifications would be required on the FRPS side.