A Framework for Telecontrolled Service Robots

by

Monirul Khan

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfilment of the requirements of the degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

A Framework for Telecontrolled Service Robots

BY

Monirul Khan

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

MASTER OF SCIENCE

Monirul Khan

ABSTRACT

The present work is a part of the ongoing development of a semi-autonomous remotely controlled IP centric service robot framework. An instance of the framework is a robot for weed extermination in an outdoor environment. Technologies built upon in this thesis include an embedded platform, reconfigurable hardware, a software platform and 802.11 WLAN technology for communication. The hardware includes a DC motor controller subsystem, sensory subsystem and stepper motor controller subsystem, whereas the software handles the communication between the processor and the remote PC, transforms two-dimensional information of the joystick to speed and direction commands using a vector-based control scheme and provides the video feedback stream. The robot is comprised of a heavy duty chassis with four wheels. Two DC motors provide the drive for the back wheels operating from two 12 volts car batteries. Two open source motor control modules (OSMC) implement the high power H-bridge control system for each of the motors. This operator assisted robot includes some degree of machine intelligence in order to deal with uncertainly in an outdoor environment. A collision avoidance subsystem provides local intelligence designed to avoid obstacles that the operator may not be able to respond to quickly enough. This aspect of the framework was based on a fuzzy logic controller and utilizes sonar to estimate distances to obstacles. The semi-autonomous operation also includes suitable APIs to implement weed removal service tasks with the help of a stepper motor controller subsystem.

ACKNOWLEDGMENTS

I would like to express my deep and sincere gratitude to my supervisor, Professor Bob McLeod for his advice and support. His wide knowledge and intelligent way of thinking is of great value of me. His encouraging and personal guidance have provided good basis for the thesis. Without his help, this work would not be possible.

I would also like to thank my collegues: David Sanders, Venkateswara Reddy and Marek Laskowski for helping me complete my thesis.

Lastly, I would like to thank my father Mozibul Islam Khan and mother Sultana Begum for their support over the past years. I dedicate my thesis to my father and mother.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The development of Internet based teleoperated systems have garnered considerable attention in past few years due to new challenging applications. These applications include space exploration, underwater exploration, remote surgery, traffic control and health care to mention a few. Our interests here are the development of a framework integrating aspects of the Internet with the field of robotics. For our work this implies the use of IP based wireless for at least portions of the framework.

In order to realize a complete wireless system, 802.11 was chosen as the communication technology to connect the robot to a remote host or the Internet itself. Other researchers have also used radio modems and cellular technology to control teleoperated robots wirelessly. However, IEEE 802.11b/g WLAN technology [2] has become very popular because of its cost, bandwidth and flexibility. As such we decided to use this WLAN technology for our primary means to communicate with the robot. By being IP centric, extending the platform to cellular or other technologies such as 802.16 (Wi MAX) becomes considerably easier as these also support TCP/IP on the

data side.

Recent work in mechanical weed control [3] has also motivated us to extend our robot into an agricultural mobile robot capable of operating in an outdoor environment. Since chemical herbicides have an adverse impact on the environment, there is actually a real demand to use chemicals in the most effective (minimal) way possible. In general, farmers spray herbicides over the field regardless of where the weeds are specifically located. An additional motivation for our work here is to reduce the use of chemicals by replacing spraying with intelligent weed detection and mechanical weed control with a drilling machine. The robot can also be adopted for the dedicated spraying of herbicides into a particular location depending upon intelligent detection of weeds using a vision system. Even further reduction in chemical usage could be reduced by spot touching the leaf of an identified weed.

The framework developed here is an example of a new generation of personal service robot that is required to have some degree of autonomy to undertake a service task in an uncertain environment. In order to build such a system that has to reason under conditions of uncertainty, it is desirable to initially have remote control that can be operator controlled or assisted with eventual evolution to a more fully autonomous robot. The work presented here develops an operator based telecontrolled service robot suitable for incorporating semi-autonomous functionality.

1.2 Related Work

First generation of web *telerobotics* started in 1994 with K. Goldberg's Mercury project [4]. It consisted of an telecontrolled industrial robot arm which allowed the user to move a camera or to direct a short burst of air to view a desired location. Ken Taylor developed a six-axis telerobot at the University of Western Australia with a

fixed camera the same year [5]. In 1995, Goldberg extended his Mercury project idea into a telegarden project which allowed the remote users to dig and water plants [6]. These first generation robots mostly dealt with a mechanical arm under the direct control of users with no mobility and no built-in intelligence on the controlled device. With the rapid development of Internet, more *telerobotic* research results for laboratory experiments over the Internet were carried out [7], [8]. These second generation robotics project introduces limited autonomous behaviors. For example, The Khep-On-The-Web *telerobotic* project was developed with the long term goal of enabling the sharing of expensive or unique equipment for educational purposes [8]. As a demonstration a mobile robot was accessible by web user allowing for remote control of its movement in a wooden maze. The museum Tour Guide Robot [9] and Xavier [10] are examples of robots exploring distant environments with some degree of autonomous mobility. This brings up the concept of personal service robot as identified by S. Thrun where autonomous behavior is a major determining factor of the robot's performance as service robots share the same physical space as human beings [11]. This kind of robot provides assistance to make it easier for us to perform certain service tasks. For instance, Fung et al. [12] proposed a hospital service robot which can help to transport medicine and medical instruments. In addition to that, M. Y. Shieh et al. [13] proposed a vision-based shopping assistant robot which can serve people in a mall by providing services such as guiding, communication and accompanying. There is virtually no end to considering tasks that are well suited to service robots, operator assisted or more fully autonomous.

To achieve the eventual goal of a autonomous mobile robot, effective obstacle avoidance is an essential key to a control strategy which is able to sense the environment and provide control information to the robot. In this manner the robot can autonomously (locally) make decisions from the lower level sensory data perhaps

overriding the control of the remote operator. A number of methodologies have been proposed based on the sensor data for achieving obstacle avoidance. S. K. Pradhan et al. [14] employed a fuzzy logic controller to facilitate the collision-free navigation task for a mobile robot based on the sonar data. Other research work based on fuzzy control for sonar based obstacle avoidance can be found in [15], [16], [13].

In addition to the autonomous behavior, communication is also a major determining factor of the robot's performance. The use of a wireless channel in teleoperated mobile robots is not new. Most of the existing robots have a radio modem providing a connection between the desktop client and robot server. Since wireless LAN (WLAN) technology has become widely deployed, many researchers have been using WLAN as a communication technique to operate mobile robots [17], [18], [19]. This basically implies the use of the TCP/IP protocol over wireless for telecontrol which has always been one of our design objectives.

1.3 Thesis Overview

The present work is a part of ongoing development of a semi-autonomous service robot frameworks with this instance being for weed extermination in an outdoor environment. The specific task is less important than the framework itself, which if properly designed will allow the framework to be extended to many applications. Our design centers on the Altera based Cyclone II field programmable gate array (FPGA) device containing memory, logic elements and a NIOS II embedded soft core processor, as well as use of the Altera developments tools. This is a system-on-a-programmable-chip (SoPC) system concept that greatly improves the design flexibility and gives greater flexibility for application development. Any SoPC-based robot system will typically include two modules: hardware and software. Altera's FPGA implements

the hardware modules, which includes a DC motor controller algorithm and a pulse width modulation (PWM) signal for the robot control subsystem. The SoPC is also configured as an Ethernet peripheral. Software modules are implemented in uC/OS II RTOS on the Nios II processor. The software handles the communication between the processor and the remote PC, and transforms two-dimensional information of the joystick to speed and direction commands using a vector-based control scheme. By using this hardware/software strategy we were able to develop a extendable robot control system. The robot itself consists of a chassis with four wheels and related electronics. Two DC motors provides the drive for the back wheels operating from two 12 V car batteries. Two open source motor control modules (OSMC) implement the high power H-bridge control system for each of the permanent magnet DC motors.

With the rapid development of the Internet, many intelligent devices are being developed that are IP centric. As such, we decide to make use of wireless local area network (WLAN) technology and Internet Protocols (IPs) as our main communication technology for *telerobotic* control and wired Ethernet for on board communication. The robot is intended to serve as an operator assisted agricultural robot. Within this context a primary goal of the WLAN is to serve as the medium to send control information and transfer images. Once transferred, image processing tasks are used to classify weeds and differentiate them from the grass or crops. In order to avoid obstacles, it was decided to augment the robot with ultrasonic sonar and infrared sensors to detect objects that may be on the robot's path. In addition, a Global Positioning System (GPS) is used to get positioning information for the robot's location. In this manner, a user can remotely control the robot using a joystick with visual feedback in near real time to navigate the robot around the field in search of weeds. A collision avoidance subsystem provides the first level of semi-autonomous behaviour and utilizes a sonar based system to avoid obstacles that the operator may not be able

to respond to quickly enough. In general with any semi-autonomous functionality some degree of machine intelligence is required in order to deal with uncertainties. In the present case in addition to obstacle avoidance semi-autonomous functionality will take place once we successfully managed to position the robot in a place where weeds are present. This also requires building and controlling of a mechanical XYZ table with a drilling machine or weed elimination apparatus to successfully perform the semi-autonomous behavior of weed elimination.

We have also implemented a wireless access point (WAP) on the robot which is being used for relaying data between the wired and wireless devices. The WAP is mounted on the mobile robot which can act as a server for use in extending wireless coverage to multiple wireless mobile clients.

The general area of research, namely "operated assisted service robots" is arguably one of the most significant robot development areas. Challenges include both the robot itself in addition to those introduced by the telecommunication and protocols required to remotely operate the robot. Even fully autonomous robots undertaking any serious service task will require a telecommunication infrastructure.

1.4 Design Formulation

This section presents some of the design principles used in our approach and proposes an object oriented framework for our service robot.

1.4.1 Use case

We based our initial design on the following use case that are essential for building a service robot framework. These are presented in Figure. 1.1 and interactions between them are described as follows:

- The user wants to provide the robot with maneuver information.
- The user requires visual and sensory information from the robot.
- The user provides commands for the robot to perform in a operator controlled manner, with user control potentially overridden by a semi-autonomous subsystem running on the robot.
- The semi-autonomous operation includes the collision avoidance subsystem and weed removal operations.

1.4.2 Class Model

In our service robot application, it is essential to not only provide a robust hardware architecture but also to provide an object oriented software architecture for modularity and system extendibility. Based on the use case formulation, an object oriented model for the service robot framework needs to be defined with associated data types and procedures. This class model provides a common platform that can be instantiated to support a variety of services in an outdoor environment, in addition to the weed eating application. The top level class diagram is presented in Figure. 1.2. There are of course other design approaches that need to be carefully considered to integrate all the application components as described below:

- First, since the service robot framework is designed to provide services such as weed eating; a coherent control architecture needs to be defined for both the operator assisted and semi-autonomous operation in order to perform effectively.
- Second, there are many to many relationships between the hardware components and software components to consider. For example, the collision avoidance subsystem running on the embedded processor receives information from

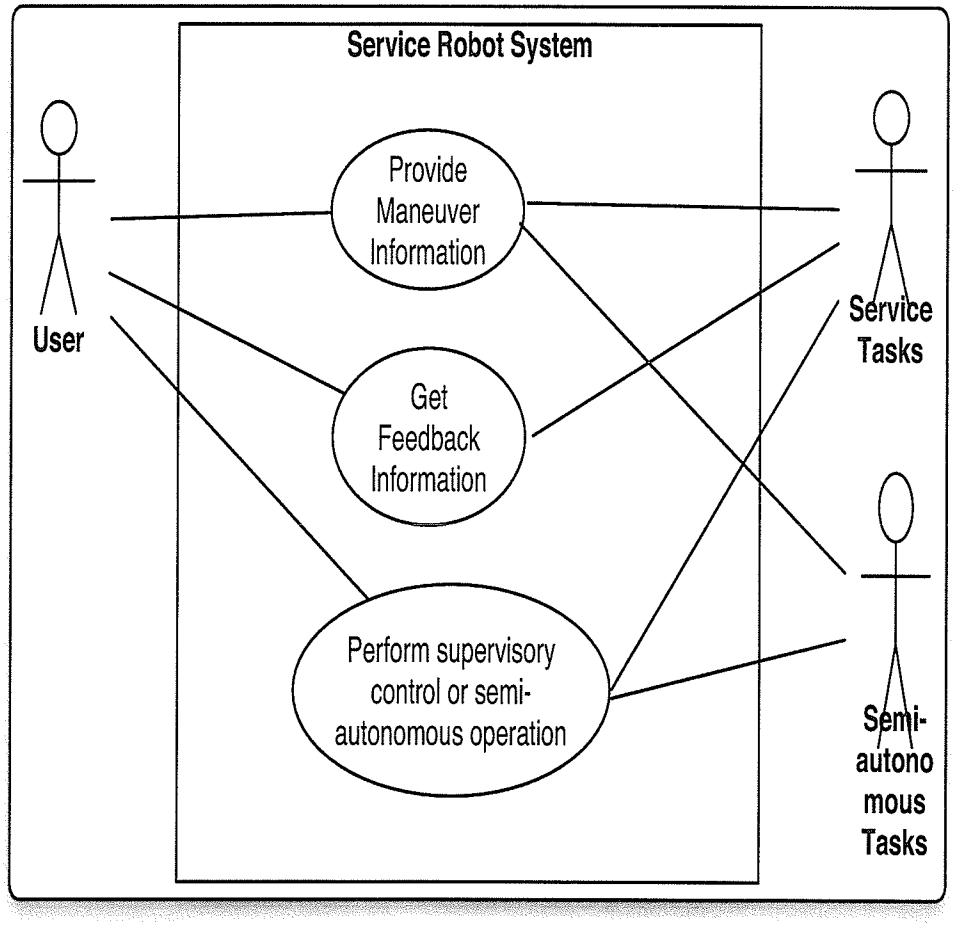

Figure 1.1 Use case for design formulation

the sensor modules, and can under certain conditions override the supervisory control information and directly communicate with the control hardware module that is responsible for maneuvering the service robot.

- Third, the functional performance of the system needs to remain satisfactory at all times to avoid any accidents which may otherwise be caused by the service robot. For example, even if we are running another service application on the communication channel, we need to make sure the image updating rate on the operator side display is always sufficient to provide near real time feedback.

- Fourth, if any new services need to be added to the framework, our system design should allow integrating this new module into the service robot class entity so that it can be instantiated easily. This flexibility would enable the modularity and reusability of our framework for use with other service robot applications.

1.5 Thesis Statement

The work of this thesis is to develop a telecontrolled service robot framework which consists of the following objectives:

- to allow the mobile robot to be remotely controlled through the Internet to navigate in an unfamiliar real world environment

- to provide an object oriented approach for accomplishing a service task, for example, a weed removal operation, that can also be extended for other similar service tasks in an outdoor environment.

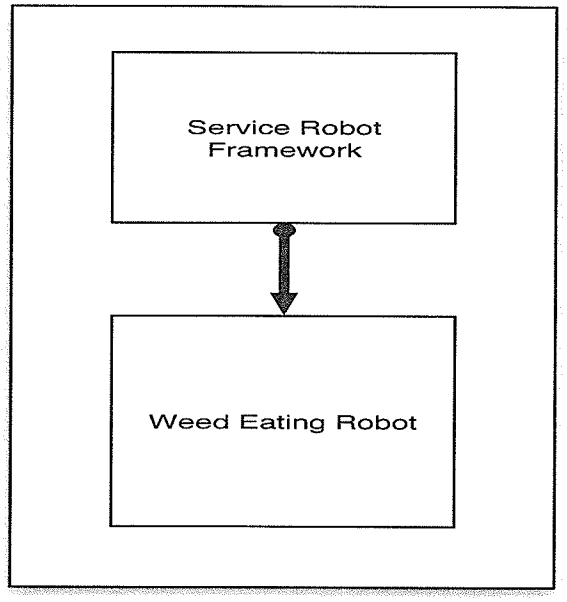

Figure 1.2 Class diagram for service robot framework

- to develop a robust hardware design in order to provide all necessary control logic signals to maneuver the robot and to perform the weed destruction or spraying operation.

- to develop a sensory feedback subsystem; this will collect image, distance and location information through a webcam, ultrasonic sensors, and GPS sensor respectively and transmit this back to the remote operator.

- to obtain a satisfactory image updating rate on the operator side display necessary to provide a real time visual feedback to the operator to avoid collisions.

- to support a small degree of autonomous operation to avoid obstacles in an uncertain environment along with the problems of arbitrary network delay and restricted bandwidth. A second semi-autonomous operation incorporates correct positioning of the XYZ table to perform spraying or weed destruction operation.

- to develop a obstacle avoidance subsystem using a fuzzy rules based system to infer control decisions for early object detection based on the lower level sonar and infrared sensor inputs.
- to develop supervisory control software which will provide a protocol for acquiring data from the joystick interface and for sending and receiving data using the TCP/IP protocol.
- to develop a Graphical User Interface (GUI) for the remote operator.
- to mount a wireless access point (WAP) on the mobile robot for use in extending wireless coverage area.
- to integrate all hardware, software, power electronics and mechanical design in developing the service robot framework.

1.6 Thesis Scope

The scope of this thesis to provide a robust hardware design and object oriented software approach for a telecontrolled robot framework oriented to service tasks using 802.11 WLAN technology. The service robot framework requires semi-autonomous operations in order to perform weed removal operations and to avoid obstacles in an outdoor environment. A fuzzy logic based reactive control strategy is implemented to avoid obstacles by providing an early detection of objects based on the lower level sensor data. However, actual autonomous maneuvering of the service robot around the detected object is outside the scope of this study and will be left to further work.

1.7 Design Components

A major portion of the design focuses on creating a *telerobotic* system that can be controlled by a remote operator with the help of real time monitoring system and a collision avoidance sub-system. Basic teleoperation system consists of at least two computers communicating with each other. One computer is used at the operator end to control the robot, and the remote computer (robot) is used to perform the control actions and provide any sensory feedback. We used the basic teleoperation system architecture and wireless WLAN-802.11 as our primary communication technology. As can be seen in Figure. 1.3. Our teleoperation system consists of a robot chassis, desktop machine, camera, drilling machine (or sprayer) and robot controller. In more detail it consists of several components described below that make up the complete teleoperation system.

1. Rugged chassis with two wheels, two DC motors, car batteries, platform for holding embedded processors, camera and sensors and electronic components.
2. Wireless communication interface (WLAN-802.11 in this case, as opposed to traditional Radio Controlled (RC), or proprietary technology)
3. Embedded Processors (CPUs and FPGAs)
4. Sensors such as Camera, GPS, Sonar and IR.
5. DC Motor Controllers
6. Stepper Motors and Controllers
7. XYZ table (partially completed)
8. Drilling Machine (partially completed)

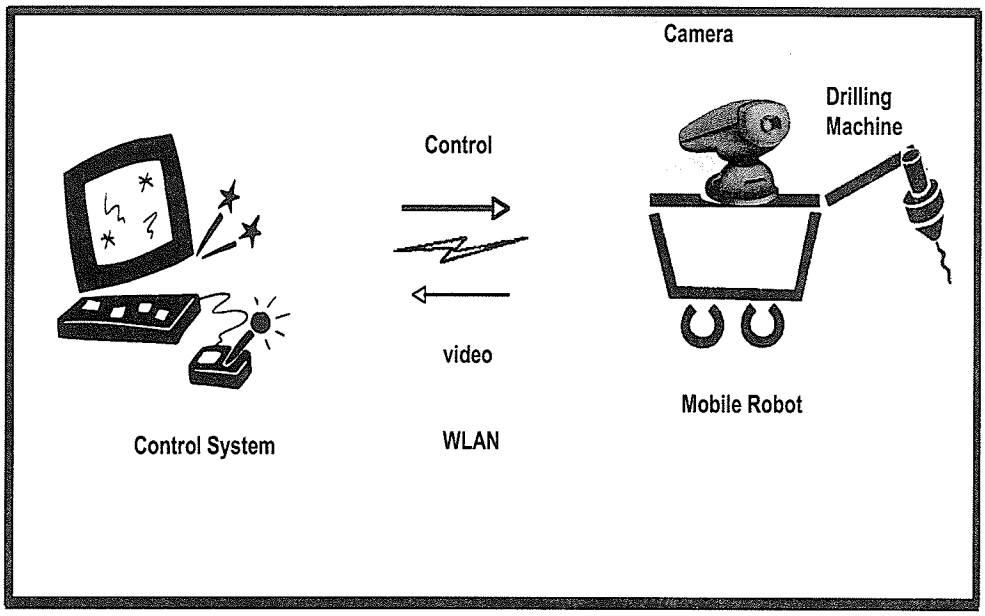

Figure 1.3 Our proposed teleoperation system

9. Interface for robot command input

10. Client program for sensory feedback rendering (including visual feedback)

1.8 Thesis Goal

The main goal of our research is to build a service robot framework using an embedded FPGA-based robot controller that can perform some degree of semi-autonomous service tasks that can be precisely controlled remotely with the help of visual and sensory feedback over a WiFi network and/or the Internet. In this specific case, the semi-autonomous operations is in providing collision avoidance and automatically detecting weeds and their subsequent removal or destruction. My specific responsibilities were in building/designing the framework including the collision avoidance subsystem. Automatically detecting weeds is a subcomponent outside the specific

scope of my thesis but position information obtained from such a process would be utilized by the framework to drive the stepper motors of the XYZ table.

The goals and strategies are set out below to establish a complete framework for the agriculture teleoperated mobile service robot.

- Motor control -The Pulse Width Modulation (PWM) core functionality has been designed and implemented using a hardware description language (HDL). It was essential to develop the PWM hardware on an FPGA, because the output signal needed to be updated very frequently. This hardware design is then connected to register mapped peripherals which can be accessed by the application running on NIOS II processor. Two PWM signals and other motor control logic signals are taken out of the Altera Development and Education (DE2) [20] board's programmable I/O pins and connected to the two OSMC [21] motor control drivers. The OSMC motor controllers accepts the PWM signals and other controls signals at the input of the H-bridge driver controller circuits, thus generating appropriate voltage and current to drive the two DC motors. This is a significant design improvement over a previous robot built in this lab when the PWM design was software based which resulted excessive CPU resources being required [22].

- Real-time image acquisition - Initially a CMOS image sensor was connected to the expansion interface of the Altera DE2 board which can be accessed through the GPIO pins of the cyclone II FPGA. The output of the camera follows the Bayer color filter array pattern in which each pixel is represented by single color value of Red (R), Green (G) or Blue (B). The raw data format needs to go through an image processing stage in order to construct an image which represents each pixel with RGB values. Although useful, the CMOS image sensor

option appeared to be unable to meet our real time requirement. Alternatively and at present we are using a USB webcam which can be connected to a Soekris embedded single board computer (SBC) to provide faster image transfer to the remote PC. The CMOS image sensor can however be reused for image capture of the field of interest. The difficulty at present is that once captured in SRAM on the DE2 board the high resolution image is too large for the relatively primitive stack on the DE2 board to transmit in reasonable time. This would require writing an intellectual property (IP) block interfacing the SRAM to the Avalon bus, a minor thesis in itself.

- Image processing to identify a target - An image processing study is going on in our IIC lab by Mr. Marek Laskowski who is evaluating a number of approaches such as neural network and centroid-based approaches to identify and locate weeds (dandelions) in a lawn. His project work can be retrieved from the site: www.iic.umanitoba.ca/docs/robot-final.pdf. This component of the project is not within the main scope of my research although aspects of it may be incorporated. Specifically once extracted normalized coordinates will be used to drive stepper motors on the XYZ table.

- Collision avoidance subsystem -To ensure a safe operational telecontrolled robotic system, it is important that the mobile robot be able to navigate without colliding with obstacles. A number of sonar and infrared sensors are incorporated to implement a collision avoidance subsystem. The subsystem will provide a control strategy where it will able to sense the immediate environment and react with the aid of some form of reasoning with the sensory data. For example, if the collision avoidance controller detects any object ahead, it will choose the best direction for the robot to avoid a collision . The controller may direct

the robot into a different direction, or slow the speed of the robot or make an emergency stop. The collision avoidance controller will override supervisory control information. These aspects will rely on specific rules of inference, likely implemented as a fuzzy or neural type of inference engine as further discussed in Section 2.2. One of the primary reasons a collision avoidance subsystem is necessary is a consequence of the unreliable nature of the communication channel and its unpredictable delay. Using best effort protocols and wireless 802.11 communication can result in unpredictable delay and possible performance degradation. As such, a collision avoidance subsystem is crucial to any type of telecontrol service robot.

- Construction of the XYZ table - A mechanical three degree of freedom table is being constructed that is equipped with a drill or chemical sprayer apparatus. Three stepper motors will be used to provide correct positioning of the table. My aspect of this work is demonstrating the remote control of the stepper motors.

- Semi-autonomous functionality- Collision avoidance mechanisms and weed destruction operations without human intervention provide the semi-autonomous *telerobotic* functionality. The mobile robot can be operated under supervisory remote control and is able to maneuver safely with the help of vision system and collision avoidance subsystem. Fuzzy or inference reasoning is implemented based on the data from the sensory inputs to provide the robot with overriding operator control information to avoid obstacles. For example, if the robot is in close proximity to an obstacle ahead of it, the fuzzy logic controller will instruct the service robot to execute emergency stop. Similarly, once we maneuver the robot to its desired position, the robot captures an image for processing

to detect weeds from the grass. If it finds any the FPGA controller positions the three dimension mechanical axis precisely and provides necessary signals to perform weed killing operation. These are difficult tasks but our objective is to demonstrate the utility of the framework when performing such service tasks.

- Global Positioning Information - A GPS sensor is mounted on the robot to determine its position (longitude, latitude, altitude) and time (Universal Time Coordinates) in an outdoor environment by the satellites of the global positioning system. GPS was deemed an important hardware module for inclusion in the framework as it is primarily intended for outdoor operation.

- Other feedback sensors - Hall, strain gauges, gyro and 3D compass sensors are optional to the platform but could be integrated on the system to provide additional feedback of the outdoor environment. Being IP centric and modular our conjecture is that their implementation will be similar to sensors such as sonar and as such relatively easier to incorporate.

- PC hosted supervisor control software- A socket based client software program was written which is able to acquire joystick information and send it to the robot server program.

- Graphical user interface (GUI) development - A simple graphical user interface was developed to provide information on visual feedback, sensory feedback and control information.

- Server robot software - A socket based server software program was written on the robot side, which listens for a client PC to connect and once it is connected it executes tasks according to the messages received from the client. The server software is also responsible for sending the video stream and sensory feedback

to the client PC application program. The basic client-server architecture will be discussed in section 1.9.

- Network transmission - The control and feedback system has been developed from the client-server paradigm using the TCP/IP protocol stack. Since the Altera FPGA does not have a support for the wireless interface, we decide to use a Soekris 4826 SBC to provide wireless Internet access to the Altera board.

- Power electronics design -Our mobile robot is comprised of a heavy duty chassis with four wheels where the two real wheels are used to maneuver the robot. Two DC motors and two small car size 12 volts batteries (24 volts in total) are used for each drive unit. Due to the heavy weight of the chassis and motor type, we needed high power motor driver circuits to drive the motors . The specifications that we were to look for the motor driver circuit were :

  - The circuit should be able to accept single logic-level PWM signal for controlling a wheel rotation and to accept single logic-level control signal for controlling a wheel direction.
  - The circuit would require dual H-bridge drivers for each drive unit.
  - Each of the H-bridge drivers must capable of driving 30 Amps of current at 24 volts.
  - The circuit should be able to operate with a high frequency PWM signal for example, 30 KHz.

We found an open source motor controller (OSMC) which satisfies our functional requirements. The 24 volt car batteries supply the voltage for the 12 volts fan and the motor controller on each drive unit. The FPGA controller provides

all the logic functionalities to the motor controller and the motor controller provides the direction of the current to the motors to drive each unit.

The system is also comprised of three stepper motors from powermax and bipolar motor drivers which provides dual H-bridge support. A 30 volt power supply with 2.5 Amps current is required for the stepper motors.

- System Integration - This was a project integrating hardware, software, power electronics and mechanical development. As far as practical each of the areas was developed in parallel to others in order to meet the functional requirements of each individual module. In this manner some degree of experience with concurrent design concepts was also obtained facilitating decisions as to various hardware/software tradeoffs.

1.9 The Proposed Framework

1.9.1 Client Architecture

The proposed framework for the client architecture of our *telerobotic* system can be seen in Figure. 1.4. In a client PC an operator is able to interact with the system with the help of either joystick or by directly controlling the buttons of the GUI. The joystick interface provides more flexible control of the mobile robot but it is not obvious in terms of optimal implementation. All the information to and from the client PC is sent/received in the form of TCP/IP packets over a wireless 802.11 network. The communication takes place with the help of a socket based programming. This provides bidirectional communication between the client PC and the robot server. The buffered image is displayed on the monitor on the operator side to provide visual feedback. A buffered image from the high resolution CMOS camera can also be sent

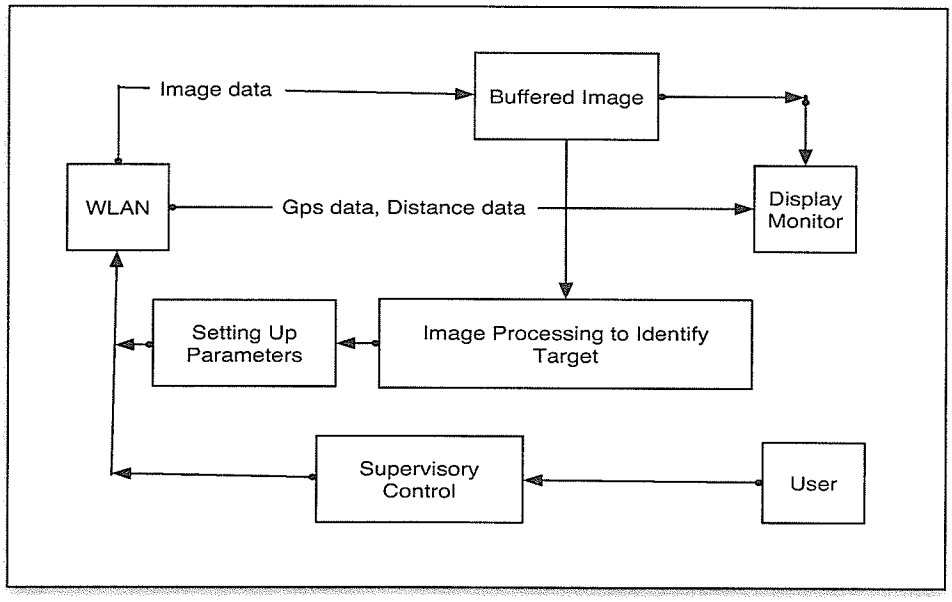

Figure 1.4 Our proposed client PC framework

to an image processing module if necessary or requested by the user.[1] In this case the supervisory control will then be handed over to the autonomous operations. An image processing module will perform a weed detection algorithm and if it identifies weeds, it will send relevant parameters to control the XYZ table and to perform destruction operation. The algorithm for weed recognition was not part of my research but rather suitable for the development of the platform incorporating those types of algorithm. All the sensory feedback data from the sensors such as GPS, sonar, infrared in forms of distance data and location data are displayed in the monitor. The status of the robot such as autonomous mode, collision avoidance mode, image processing mode or weed killing mode will also be displayed. The GUI also keeps track of and displays the speed and direction of the robot.

[1] A number of difficulties were encountered in attempting to transmit the high resolution images as discussed previously.

1.9.2 Server Architecture

The proposed framework for the server architecture of our *telerobotic* system can be seen in Figure. 1.5. The server socket program running on an embedded processor is able to receive commands (speed, direction) for the robot drive module over the WLAN. The server robot also captures the video and sends it to the user for visual feedback. It also obtains data from the sensor units and applies those data to construct a collision avoidance subsystem. If the collision avoidance subsystem detects an object, it may alter the direction and speed of the robot to avoid any collision. The robot server also sends the sensory feedback to the user for display.

1.10 Design Methodology

Figure. 1.6 shows the flowchart for the system design steps. Our design steps have 6 main parts as described below.

- Robot Controller -The robot controller is running on the Altera DE2 board which is running a NIOS II soft core processor. The PWM hardware circuit and control truth table are implemented using the Verilog hardware description language with the circuit running on the Altera FPGA. Software running on the processor receives and extracts commands from the remote-user and writes the duty cycle and period into the hardware peripheral which provides inputs to the hardware circuit.

- Collision Avoidance - The collision avoidance subsystem is also running on the Altera DE2 board. Sonar and infrared sensor data acquisition circuits are written in Verilog and are accessible to the software using NIOS II IDE. At present fuzzy controller based software is being implemented to make decisions

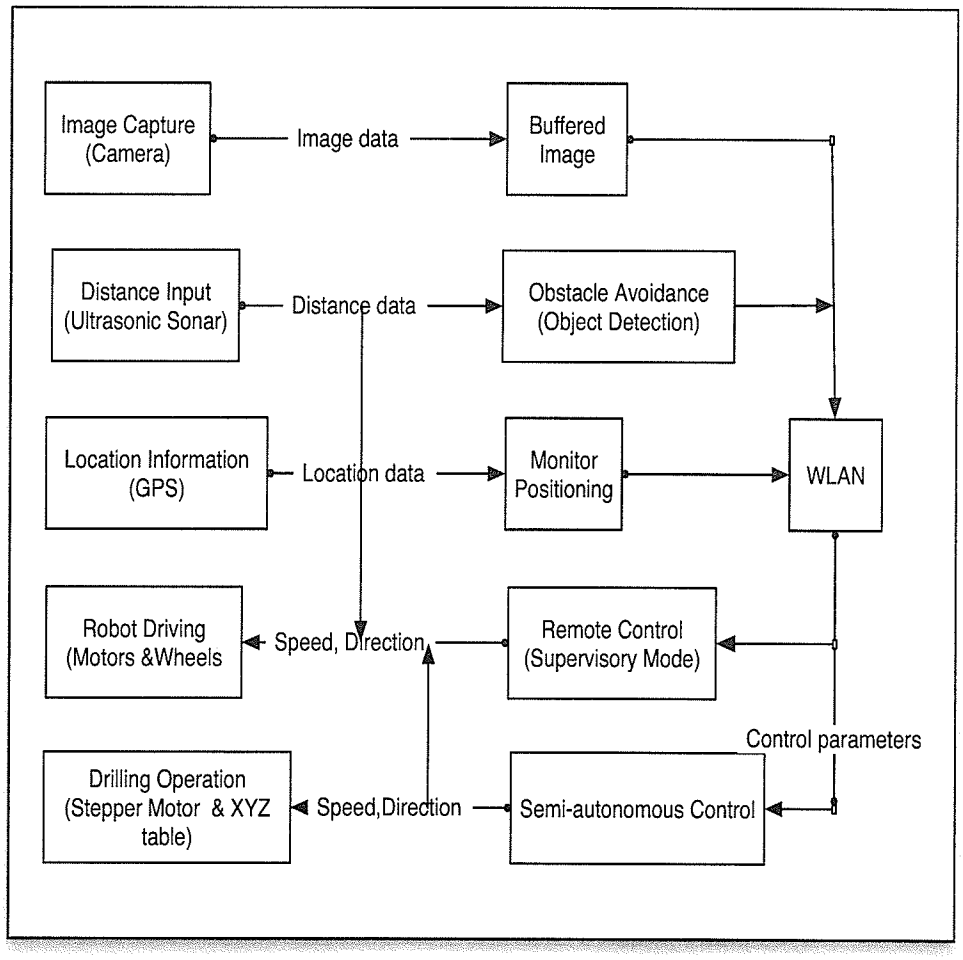

Figure 1.5 Our proposed server robot framework

based on the sensor inputs in order to avoid collision. Alternative reasoning strategies could also be implemented in a similar manner. This forms a part of the semi autonomous operation of the service robot.

- XYZ table Controller -A XYZ table controller is also running on the Altera DE2 board. The hardware circuit and control truth table are implemented as a Verilog hardware circuit and it is also running on altera FPGA. We will be implementing a positioning algorithm to drive three stepper motors in such a way that they are synchronized and can perform weed destruction operation.

- Supervisory Controller - The supervisory controller software module is running on a client PC which is responsible for acquiring joystick data. It also runs necessary algorithms to send remote user commands to the mobile robot using TCP/IP over the WLAN communication channel.

- Mechanical Design - The main robot comprises a chassis with two DC motors and four wheels. PWM driver comprises DC motor drivers, stepper motor driver and other IC devices. Two small car size 12 volt batteries are used to provide power supply to all electronic modules with the exception of the stepper motor themselves.

- Soekris Board Design -Two Soekris embedded CPU boards are used to provide multiple functionalites to the mobile robot. A Soekris board net4826 is used as an access point (AP) to provide wireless interface to the robot. We have also installed a GPS module and driver to read and send data over the internet. Another Sokeris board net5501 is used to capture and buffer images and send stream video over the network. Here again we are capitalizing on reusing network concepts on the robot itself, a methodology that we believes will be dominant for large scale robots.

1.11 Design Feasibility

Our *telerobotic* system is capable of sending video stream data, sensory feedback data and receiving navigation control data over WLAN technology. The system consists of a robot with semi-autonomous functionality incorporating a camera, sensors and 80211b/g network adaptor. The PC client sends navigation control commands to the robot and, at the same time, receives a video data stream and sensory feed data that provides information of the surrounding environment. In this way, a remote user will be able to see the front view of the robot as a visual aid and will be able to control the robot at the same time. However, it is difficult for the remote user to make correct decisions solely based on the visual feedback on the display monitor particularly, if the received data stream suffers unexpected network delay or obstacles are beyond the camera's field of view. A delay of few seconds may be incurred due to image capturing process, the RGB conversion process, encoding process, transmission delay and the decoding process at the end points. This delay is unacceptable for *telerobotic* application as it may cause a collision even under supervisory control. Sensors may also provide erroneous readings or become dysfunctional, which makes it harder for the operator to make correct decisions relying on the available readings. A collision avoidance subsystem is designed based on a fuzzy logic model to assist a remote user in addition to the vision feedback system to maneuver the robot safely to perform its operation. This also provides the telerobot with a small degree of local intelligence to handle network bandwidth and transmission delay issues. This idea of a low level collision avoidance subsystem is *biologically motivated.* In many cases low level sensory responses intervene on behalf of the neocortex which may not otherwise be able to respond quickly enough.

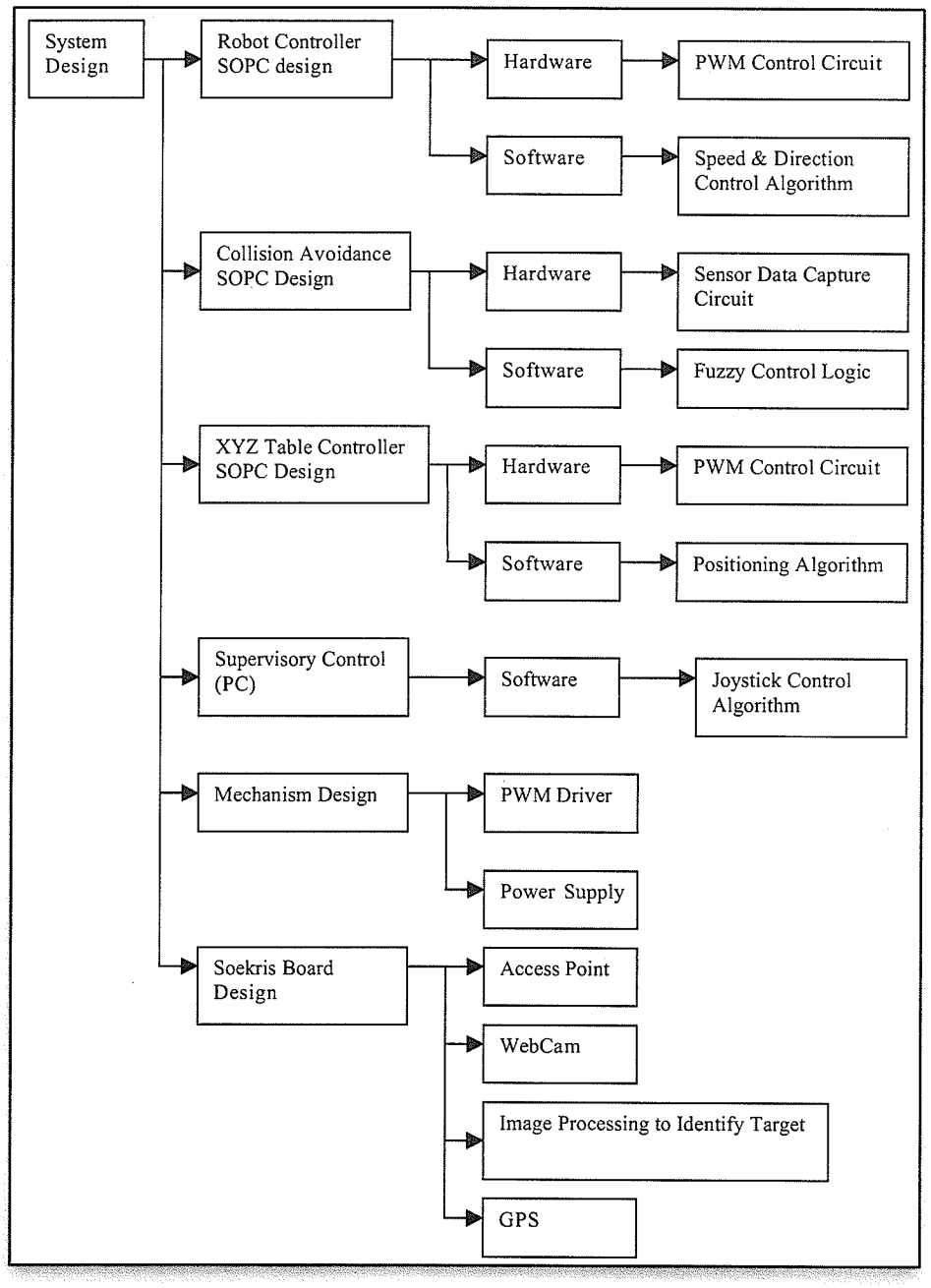

Figure 1.6 Design steps flowchart

1.12 Report Structure

The purpose of this thesis document is to describe the design of a telecontrolled mobile service robot framework and implementation of the proposed framework using an 802.11 WLAN technology and provide an instance of semi-autonomous operation for an agriculturally motivated application. The next chapter will provide a brief introduction to *telerobotic* system, 802.11 WLAN technology and background study for the development of the semi-autonomous mobile *telerobotic* platform. This includes the topics covering fuzzy logic control system, DC motor control, stepper motor control and sensor data acquisition (Ultrasonic, GPS, and Infrared). The hardware and software considerations of the project will be discussed in Chapter 3. The hardware and software implementation will be presented in Chapter 4. The results obtained are presented in Chapter 5 along with a discussion of those results. Finally, conclusions are drawn in Chapter 6 along with suggestion for future work that could extend the work completed in this study.

Chapter 2

Background and Preparation

2.1 Introduction

The literature that was reviewed in relation to this study covers a number of fields. Telerobotic systems in general are discussed as well as a fuzzy logic control subsystem which can be integrated into a collision avoidance subsystem to provide semi-autonomous operations to the service robot framework. This is followed by a discussion of 802.11 WLAN technology, DC motor driver operations and stepper motor driver operations. Finally, feedback systems are discussed, including vision, sonar, GPS and infrared sensors.

2.2 Telerobotics

A *telerobotic* system by definition is a robotic device which is operated by a human from a remote location. The main purpose of the *telerobotic* system is to perform the job which is otherwise difficult, dangerous, inaccessible for human beings to perform or is too complex for an autonomous robot to undertake. A *telerobotic* system can be

equipped with manipulators, cameras, sensors, actuators and any other electronic or mechanical devices which will be used to provide information about the environment and possibly perform some physical task if required. No matter what the application is, whether it is an airborne or underwater vehicles or surgical robot, the human operator is always an integral part of the *telerobotics* system. *Telerobotic* systems are usually designed by the system engineers in such a way that the robot is able to perform its tasks in near real time to avoid any accidents. A high degree of local intelligence is required by the telerobot to provide any real time services and to ease the operation of the remote users. If the end-to-end transmission delay between the robot and remote user is large, then it is harder for the user to teleoperate the robot in a real time. For example, if the delay is excessive by the time the user sees the obstacle, the robot might have already struck the obstacle. A collision avoidance system and environmental monitoring system is an essential part augmenting human control to maneuver the mobile robot safely, and thus provides a robust *telerobotic* system. In our weed eating application, the approach would be to maneuver the robot to a desired location with the assistance of remote human operator, vision system, collision avoidance subsystem and environmental monitoring subsystem. A local intelligence inference operation will then take over the supervisory control to perform a function autonomously at the target location. Another approach could be to provide an algorithm to perform path planning for the robot, so that it can maneuver itself to the target location without any human intervention and perform its task totally autonomously. As mentioned earlier, the first approach is more reasonable in terms of feasibility for our agricultural operation in an open field environment. The localized autonomous or semi-autonomous operation will then control the three mechanical axis table synchronized to the drill or sprayer device to perform its operations. Thus, the human operator will be completely free of control of each of the table axis positioning

which is time consuming and difficult to synchronize manually and can concentrate on monitoring visual feedback and status information of the mobile robot. As such, the instance of the framework demonstrated will introduce semi-autonomous function of the aspect that is either the most difficult or tedious to do manually. As such our telecontrolled weed eating robot fits nicely in the personal service robots taxonomy of S. Thrun [11].

2.3 WLAN Technology

As the platform incorporates telecontrol over wireless, the section provides a brief background of 802.11. 802.11 is a wireless LAN technology developed by the IEEE LAN/MAN standard committee (IEEE 802) [2]. The 802.11 standard specifies the air interface between wireless stations and an access point or between multiple stations. A station (STA) is an intelligent device within the wireless network and can be a portable laptop, mobile handheld device or stationary computer. The 802.11 architecture includes the use of one or multiple cells. Each cell is called Basic Service Set (BSS) and consists of multiple stations and a base station (access point). Access points provide basic communication between stations within a BSS and to other access points through a Distribution System (DS). A Distribution System is the backbone of the 802.11 wireless LAN, and is typically wired network 802.3. A Detailed 802.11 family protocol specification is shown in Table 2.1.

The original version of the standard IEEE 802.11 physical (PHY) layer defines global 2.4GHz band of operation based on either direct sequence spread spectrum (DSSS) or frequency hoping (FH) with a maximum data rate of 2 megabits per second (Mb/s). The currently most popular techniques among the 802.11 family are 802.11b and 802.11g. 802.11b uses the same frequency band but with higher

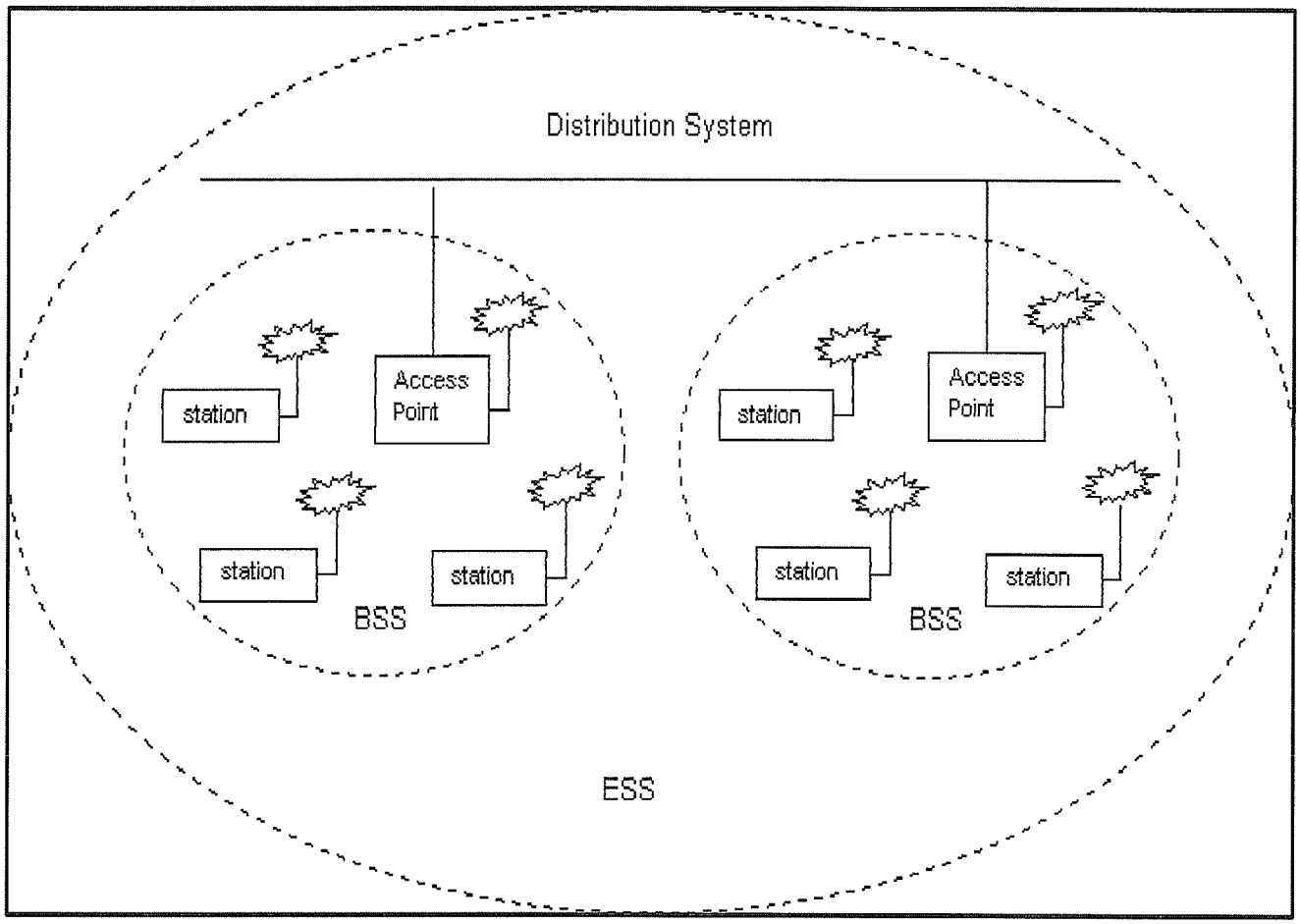

Figure 2.1 802.11 architecture.

data rate of 11 Mb/s while 802.11g operates at a maximum data rate of 54 Mbps. These standards continuously evolve, but being IP centric as our framework is, we are somewhat immune to these low level technology variations.

Table 2.1 802.11 Protocol Specification

Protocol	Frequency	Max Data Rate	Range(indoor)	Range (outdoor)	Modulation Technique
802.11b	2.4GHz	11Mbs	35m	110m	DSSS
802.11g	2.4GHz	54Mbps	35m	110m	OFDM

2.4 Fuzzy Logic Control Systems

As the platform under development includes a fuzzy controlled collision avoidance subsystem, this section provide a background study of fuzzy control. A fuzzy logic based system has the advantage that it allows the designers to incorporate a human expert's knowledge into a reactive control strategy using linguistic variables. These systems appear to be useful for handling problems which are complex or available sources of information is inadequate and unclear where it would be difficult to make crisp control decisions [23]. The main idea here is to incorporate fuzzy logic control with fuzzy rules and fuzzy reasoning to provide a collision avoidance subsystem which will be able to sense the immediate environment using sensory data and react to it accordingly. For example, if the robot is in close proximity to an object ahead of it, it will determine the speed of robot to avoid colliding with the obstacle. A fuzzy logic controller consists of a rule-base (See Table 2.2) that fuzzifies distance range measurements into "fuzzy" variables and defuzzifies the results of reasoning into control commands as illustrated in Figure. 2.2.

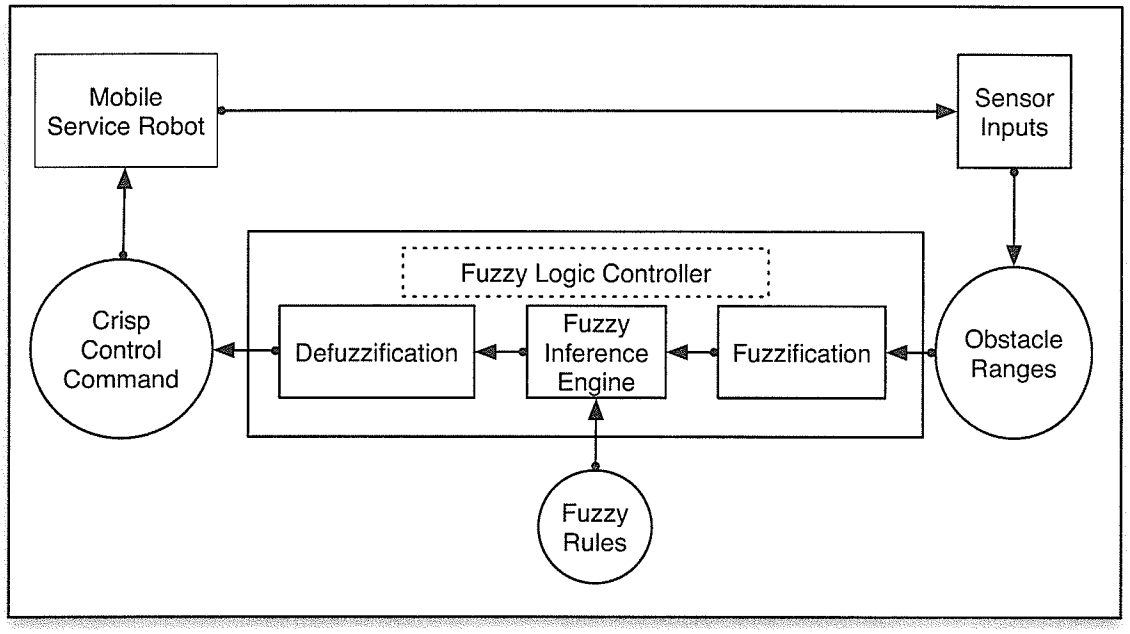

Figure 2.2 Fuzzy logic controller architecture

2.4.1 Fuzzification

Fuzzification is the means of transforming measurements into a *degree of membership*. For example, a *distance* to an object, which would be a linguistic variable in a mobile robot where a measurement could corresponds to 100 inches. By considering this measurement value, the distance variable can take on linguistic values such as "very far", "far", "close" or "very close". Fuzzification accepts measurements from the sensory data and decides to what degree it is "very far", "far", "near" or "very near". This substance of degrees is determined by human experts and is usually expressed as membership functions. The human expert might consider 150 inches to be considered as "near" rather than "far". Where "1" represents the full membership, the measurement might be

- "near" to a degree of 0.6 because it is above a close distance, but it may get

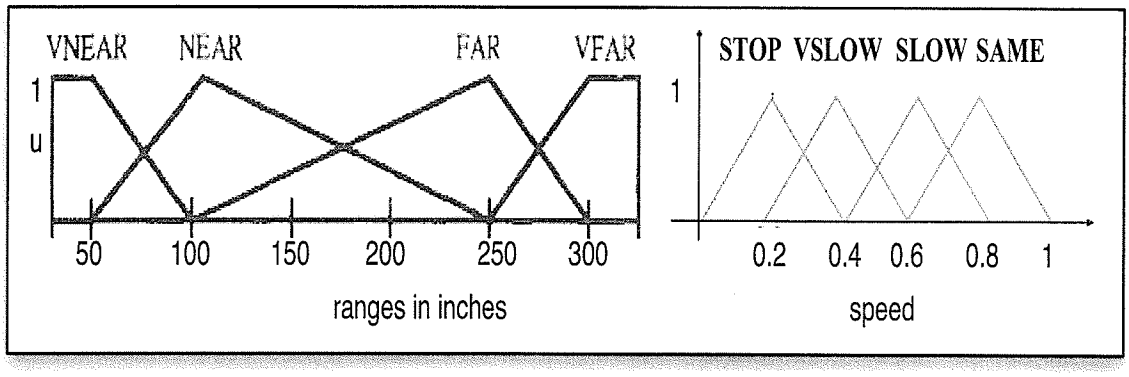

Figure 2.3 Membership functions

closer

- "far" to a degree of 0.4 because it may be considered somewhere between near and far
- "very near" to a degree of 0 because this distance can not be considered as very close.

Membership functions and their corresponding linguistic values are shown in Figure. 2.3.

2.4.2 Fuzzy Inference Engine

The fuzzy inference engine is used to combine the fuzzy IF-THEN in the fuzzy rule-based and map fuzzy input (distance) information into output (speed) membership functions. The fuzzy rule-base consists of a set of IF-THEN rules. Typical IF-THEN rules in our collision avoidance system are

- IF distance from both sensors are VNEAR THEN speed is STOP.

Table 2.2 Fuzzy Rules Used in the System

Range	Very Near	Near	Far	Very Far
Very Near	Stop	Stop	Stop	Stop
Near	Stop	Very Slow	Very Slow	Very Slow
Far	Stop	Very Slow	Slow	Slow
Very Far	Stop	Very Slow	Slow	Same

- IF distance from both sensors are NEAR THEN speed is VSLOW.
- IF distance from both sensors are FAR THEN speed is SLOW.
- If distance from both sensors are VFAR THEN speed is SAME

2.4.3 Defuzzification

After the rules have been evaluated, their combined result need to be calculated. The most common form of *defuzzification* is the centroid of area method. In this method the value of the linguistic variable and its degree are considered to represent an area. The moments areas are summed using (2.1) and produces gradually changing control output.

$$y = \frac{\sum_{i=1}^{n} \mu_A(y_i)\, y_i}{\sum_{i=1}^{n} \mu_A(y_i)} \tag{2.1}$$

2.5 Motor Control

2.5.1 Pulse Width Modulation Technique

Pulse Width Modulation (PWM) is an on-off rectangular wave and a standard technique to drive DC motors. PWM consists of series of pulses with a fixed frequency (cycles/sec or Hz) and with a variable on-to-off ratio. This duty cycle determines the percentage of time the signal is high. For example a signal which has a 50 percent duty cycle is high for half of the period and low for another half. Thus, it produces a signal with ON (high) and OFF (low) voltage within a period.

A PWM voltage and its software code are illustrated in Figure. 2.4. The effective voltage delivered to the motor coil is the peak voltage times the ratio of time the voltage is on within a period. The greater the ratio of an on time versus off time, the higher the supply voltage appears on average. As a higher average voltage is delivered to motor coils, the motor runs faster. This PWM can be produce from an embedded processor or microcontroller and later can be sent to an H-bridge circuit to amplify the current in order to deliver the desired power to the motor. Using a PWM technique has several advantages over analog voltage control. PWM does not require an digital to analog signal as it can be directly generated by the processor and thus less susceptible to interference than an analog signal. PWM current also produces enough torque with short pulses of full supply voltage to drive the motor slowly where it is impossible for analog current to deliver that amount of torque with reasonable control. A potential disadvantage is that it tends to produce more noise than a continuous analog voltage.

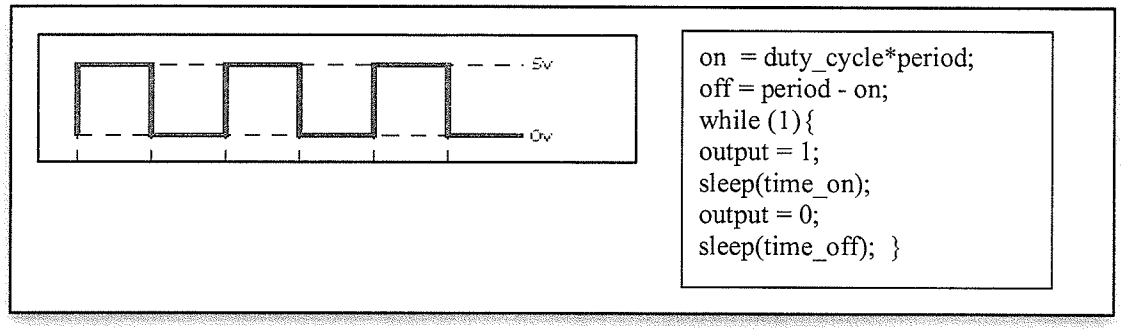

Figure 2.4 Pulse Width Modulation signal

2.5.2 H-bridge

An H-Bridge is designed to amplify voltages to run the motor at various speed in different directions. An H-Bridge has four switching elements at the both sides of the H bridge circuit. In order for the H-bridge to be functional two switches need to be turned on, either the upper left and lower right, or upper right and lower left. Current flows either in the clockwise or in anti-clockwise direction as illustrated in Figure. 2.5. When the upper left side (BHI) and the lower right side (ALI) are on, current flows from BHI to ALI, the motor starts to turn in a clockwise. When the upper right side (AHI) and lower left side (BLI), current flows from AHI to BLI, the motor starts to turn in an anti-clockwise direction. Any two switches on from the high sides or low sides cause the motors to brake and decelerate. Any two switches on from left or high sides cause short circuit. When all the switches are off, the motor freewheels. A truth table is illustrated in Table 2.3.

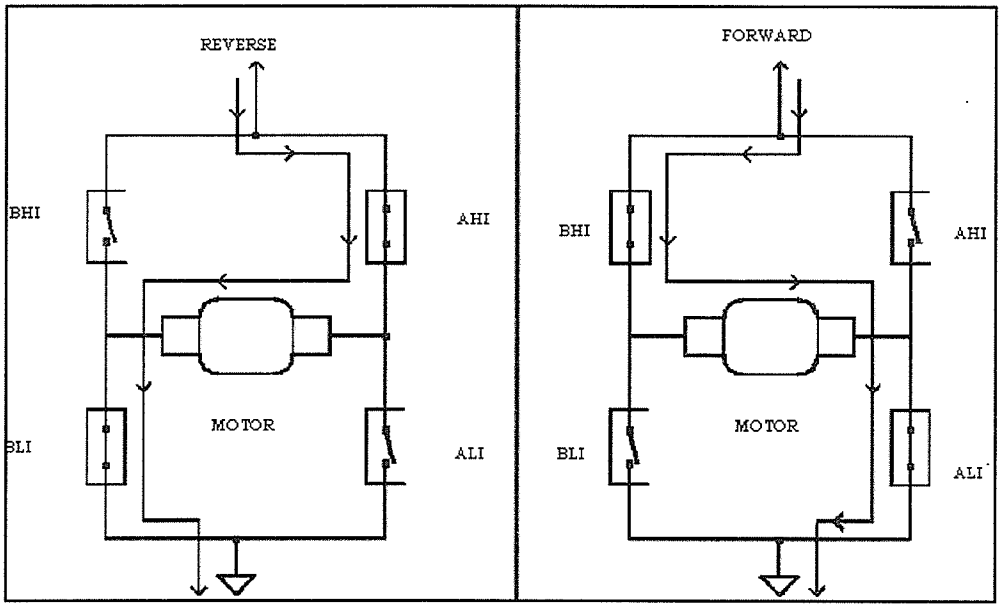

Figure 2.5 H-Bridge

2.5.3 H-bridge Circuit

For service robots of the type built here, it is required to provide 24 V and 30 Amps of current in order to drive the motors coupled to the back wheels of the robot chassis. We decided to make use of two Open Source Motor Controllers (OSMCs) to produce the high power that is required to drive the motors. OSMC consists of an HIP4081A driver which takes four logic inputs from the microcontroller. The OSMC has has all the circuitry to control the direction and speed with four FET (Field Effect Transistor) drivers and is capable of boosting the applied input voltage. For the high current application it required for the OSMC to have a low power dissipation which occurs from a voltage drop across the circuit elements. The OSMC uses N channel MOSFETs (Metal Oxide Semiconductor Field Effect Transistors) as voltage controlled switches which are much more efficient and can provide necessary current to the motor or any other output devices while having very low resistance.

Table 2.3 Truth Table for H-bridge

BHI	AHI	BLI	ALI	Direction
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
off	on	off	on	short
off	off	off	off	freewheel

The low on resistance means the less power is wasted in the MOSFETs. N channel MOSFETS are used as high and low side drivers for both current sources and sinks in the HIP4081 driver circuit. MOSFETs consist of gate, source and drain connections as shown in Figure. 2.6. In an N-channel MOSFET, if there is no voltage on the gate, it is usually off. As Vgs (voltage between gate and source) exceeds a threshold voltage Vt, current starts flowing from drain to source. It is desirable to maximize the gate voltage in order to minimize the resistance value. Thus, N channel MOSFETs are ideal switching elements for the H-bridge.

The HIP4081A driver chip provides the logic interface to all N channel MOSFETs as can be seen from the block diagram in Figure. 2.7. The driver also ensures that the two upper MOSFET's gate drive voltage is above the voltage at the MOSFET's source. The driver chip provides direction by passing the current clockwise or counter-clockwise according to the logic provided by the external FPGA controller. This means that when one of the upper side drivers is on, the opposite side lower driver is also on. For instance a logic high (+5v) at the ALI and BHI pins would turn the motor clockwise. Speed control is achieved by proving PWM to the lower side drivers

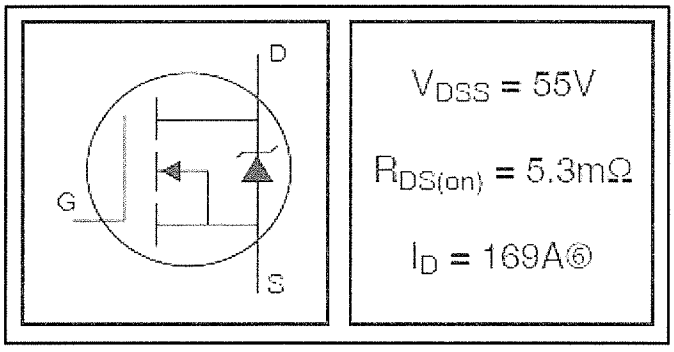

Figure 2.6 1RF1405 MOSFET

with the opposite upper side driver being on. The more current passing through the load, faster the motor moves in a given direction. The amount and direction of current to the drive motors controls the speed and the direction of the robot. The OSMC H-bridge driver provides protection features against high gate current, gate drive voltage spikes, shoot-through and voltage spikes from the inductive loads.

Protection against Gate Drive Voltage Spikes

The gate of the MOSFET acts like a capacitor. It charges and discharges with the switching the MOSFET on and off. This may cause voltage spikes at the MOSFET gate. Eight 1N74744A zener diodes are added to the circuit to minimize these transient voltages for protecting the gates as shown in Figure. 2.6.

Protection against High Gate Current from HIP4081A

Each of the leg in the OSMC is consists of four MOSFETs. The gate capacitance increase with the number of MOSFETs increases. This results in driving more currents from the HIP4081A to each of the MOSFETs gates. In order to protect the high gate current passing into each FET and minimize the total current drawn out

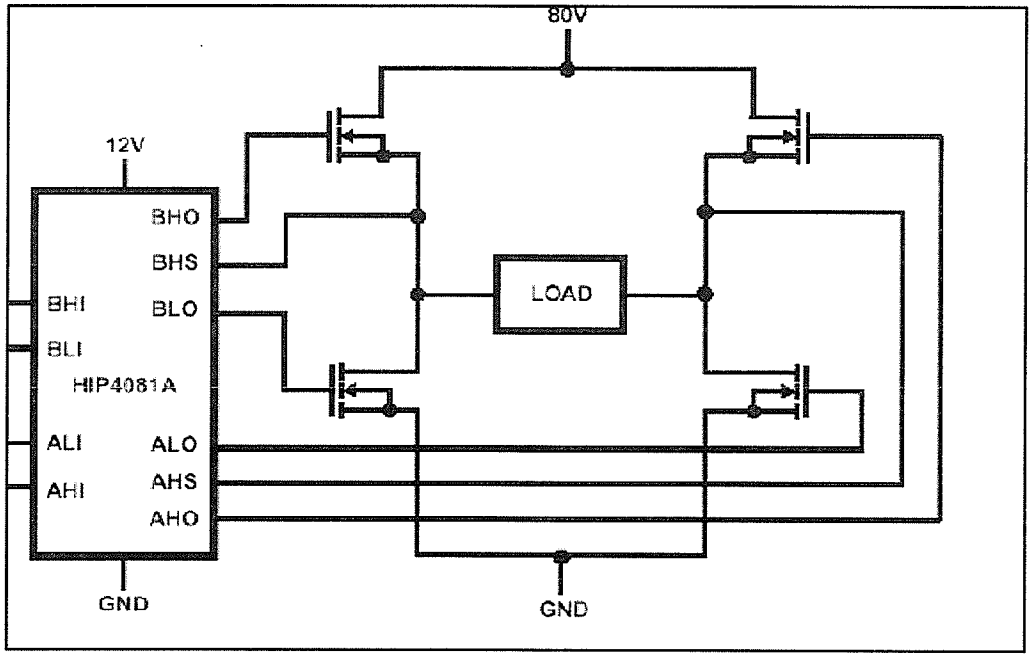

Figure 2.7 Block diagram of H-bridge and HIP4081A [24]

of the 4081A, small resisters (150 ohms) are added in between them [24].

Protection against Shoot-through

MOSFETs turn on much more faster than they turn off. Shoot-through occurs when the same side of the MOSFETs is on. This can only occur when the high side of the MOSFET is switched off and low side of the MOSFET is switched on at the same side. In order to speed up the turn off time, sixteen LLSD101ACT fast schottky diodes parallel to the small gate resisters (150 Ω).

Protection from Inductive Loads

Voltage spikes can be generated from the inductive loads such as motors . This may cause exceeding the current supply from drain to source in a FET resulting in damage. To prevent this, the OSMC adds three 15KE51CA diodes (D1,D6,D7) Transient Voltage Suppressors also called super Zener diodes. In addition to that an RC snubber network (33 Ω and 0.1 μF) across the motor leads is added as can be seen in Figure. 2.8 to protect against high frequency spikes coming from the motor.

2.6 Stepper Motor

A stepper motor converts electrical pulses into discrete mechanical movements. Figure. 2.9a shows 8-wire bipolar motor in series connection. This type of hybrid motor consists of a magnet rotation shift, called the rotor, coil windings and electromagnet on the surface of the motor, called the stator. Energizing the two coil windings in phase A and phase B as shown in Figure. 2.9b produces an electromagnetic field with a north and south pole. The magnetic rotor aligns itself with the newly created electromagnetic field in the stator. Thus, by proper sequencing of energizing the coil

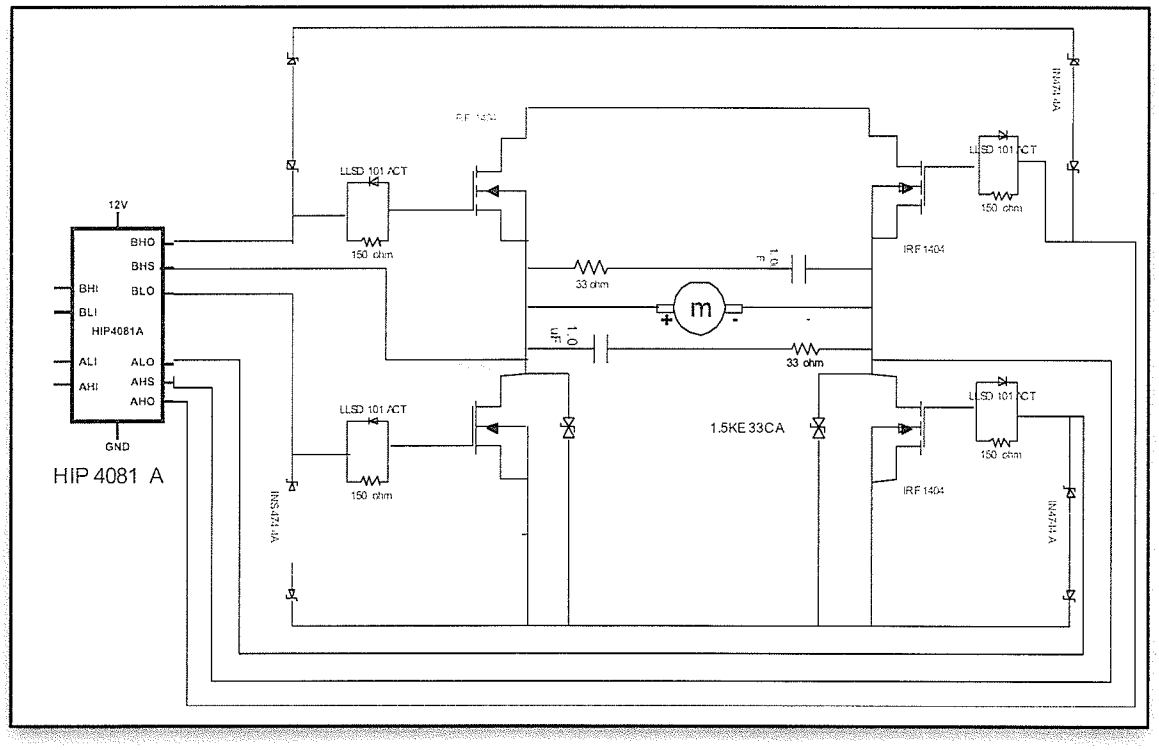

Figure 2.8 Diagram of motor controller driver circuit

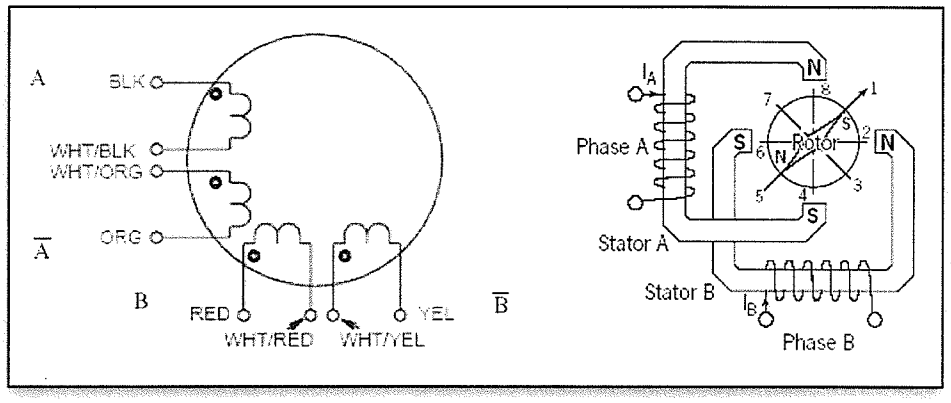

Figure 2.9 (a) 8-wire configuration. (b) Bipolar wound stepper motors [25]

windings and proper polarities of current would cause the rotor to move in clockwise or counter clockwise rotation.

It can be seen clearly that the length of the rotation is determined by the number of input pulses applied. For example a motor with a resolution of 1.8 degrees would move its rotors 1.8 degrees per step, thus requires 200 pulses to complete a 360 degree complete revolution. The speed of the magnetic rotor can also be controlled by the frequency of the input pulses. In our case, a Pulse Width Modulation (PWM) technique is used to generate input pulses as discussed in section 2.3.1. Electromagnetic polarity is reversed by reversing the current IA and IB in the windings. The standard way to operate this stepper motor with a controller is with full step pattern shown in Table 2.4 and half step pattern shown in Table 2.5.

In full step drive, two phases are energized at any given time. With two phases on, the rotor aligns itself between the north and south poles. By considering the situation where the rotor is at step 7 in Figure. 2.9b, where $A\bar{B}$ is energized. So, the proper sequence to energize the stator would be $A\bar{B} \rightarrow \bar{A}\bar{B} \rightarrow \bar{A}B \rightarrow \bar{A}\bar{B}$. This would drive the rotor from position 7→5→3→1.

Table 2.4 Full Step [1]

Step	A	$\bar{A}$	B	$\bar{B}$
1	+	-	-	+
2	-	+	-	+
3	-	+	+	-
4	+	-	+	-
5	+	-	-	+

Table 2.5 Half Step [1]

Step	A	$\bar{A}$	B	$\bar{B}$
1	+	-	0	0
2	+	-	+	-
3	0	0	+	-
4	-	+	+	-
5	-	+	0	0
6	-	+	-	+
7	0	0	-	+
8	+	-	-	+

In half step drive, both the 1 phase and 2 phase on in a proper sequence. At every second step both the phases are on and during the other steps single phase on each stator. This would drive the rotor from position 1→2→3→4→5→6→7→8. After eight steps one revolution completes and the sequence repeats.

2.7 Infrared Sensor

The IR distance measuring sensor (model GP2D02) can be used in front and rear of the robot to detect any obstacles [26]. This sensor cost is very inexpensive ($16.50 US dollar) and capable of detecting distance from 10 cm to 80 cm. It requires two lines from the controller; one pin for input and one pin for output as a digital 8-bit data stream. Figure. 2.10 shows the functionality of the IR sensor, which consists of an IR LED and a position sensitive IR detector. The IR LED emits infrared light and this beam of light is reflected back to IR detector by any objects in the way. The 8 bit integer value produced at the output represents the position where the infrared hits the object. Some of the reflection may scattered back to Position Sensitive IR detector. Any distance less than 10 cm or greater than 80 cm light will not reflected back to the detector because of the geometric location of the lenses.

2.8 Sonar sensor

The Devantech SRF04 sonar range finder sensor is operated by generating a pulse on its trigger input signal. This causes the range finder to transmit an ultrasonic (40 KHz) pulse outside the range of human hearing. The pulse travels through the air at the speed of sound (1.125 feet per millisecond) and hits any object on its path and reflects back to the ranger finder. The ranger finder pauses for a short interval of

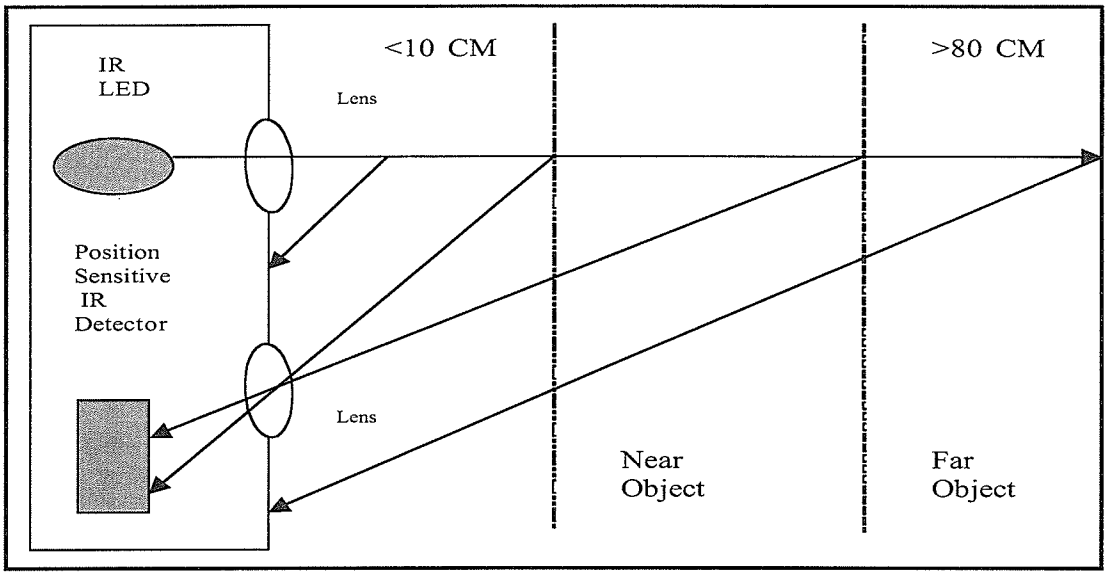

Figure 2.10 Operation of sharp infrared distance detector

time so that it does not get any signal from the transmitted pulse and then awaits the reflected pulse in the form of an echo signal. The distance to the nearest object can be calculated by measuring the time between the transmission of the pulse and the echo return which can be referred to as elapsed time. It takes elapsed time to travel twice the distance of where the object is detected at the speed of sound in air. The sonar range finder sensor provides distance measurements from about 3 cm to 3 meters. The distance to the object can be measure from :

$$Distance = \frac{ElapsedTime}{2 * SpeedTime} \tag{2.2}$$

There are a couple of basic timing requirements for the echo pulse generated by the range finder and trigger pulse applied to the range finder. The trigger pulse has to remain high at least for 10 microseconds in order for the range finder to generate a sonic pulse. This pulse is generated on the falling edge of the trigger signal as can be seen in Figure. 2.11. The receiver at the range finder waits for about 100

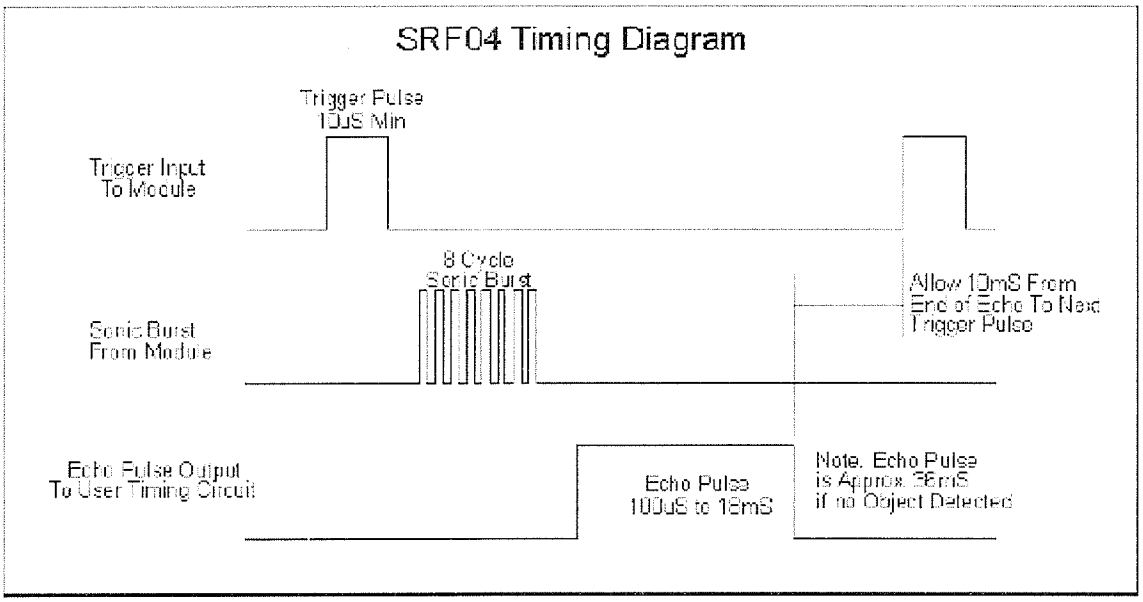

Figure 2.11 Devantech SRF04 signals [27]

milliseconds to avoid any noise coming from the trigger input signal and then it is active to receive the echo signal. The echo pulse output line is set to high once the receiver is active and remains active until the range finder detects any object or time out occurs. The elapsed time can be calculated by measuring the time between the falling edge of the trigger input and the falling edge of echo signal. The other requirements is that microcontroller has to wait of at least of 10 milliseconds in between each measurements.

The sonic range finder requires an input (trigger pulse) pin and output (echo pulse) to be connected to a controller and two other pins are connected to power and ground as well. The connections are shown in Figure. 2.12.

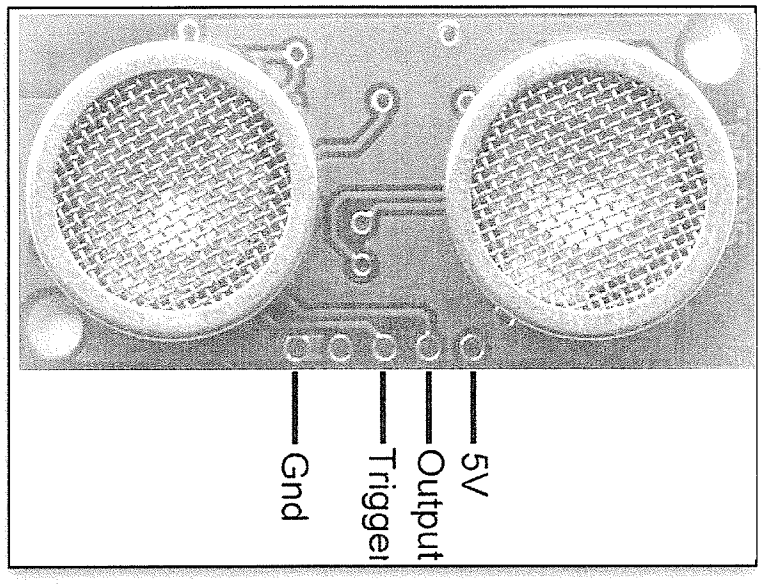

Figure 2.12 Devantech SRF04 Connections

2.9 Summary of Chapter 2

Chapter 2 presented some of the background reading and preparation gathered for understanding the service framework. The *telerobotic* system which is the backbone of the framework was discussed in this chapter. An overview of a fuzzy logic controller, which can be integrated into a collision avoidance subsystem to include semi-autonomous operations for the framework, was also presented. This semi-autonomous features leads to adding a number of sonar and infrared sensors to provide sensory feedback to assist remote operator to maneuver the mobile robot safely. DC motor control and stepper motor control were also discussed in this chapter.

Chapter 3

Design Consideration

3.1 Basic Architecture

The service robot framework includes two main components: software and hardware. The robot control system is implemented using Altera DE2 development board which includes a Nios II processor and Cyclone II FPGA as illustrated in Figure. 3.1. The framework provides hardware and software partitioning of the motor control subsystems, collision avoidance subsystem, XYZ positioning subsystem and drilling operation subsystem. Two other Soekris single board computers (SBCs) are used to provide the vision subsystem, communication subsystem, and location information system. The framework also includes a supervisory client application running on the WLAN to provide necessary control information to the server robot. All the necessary design considerations and resources used to design and implement the complete framework are described in this chapter. The ultimate goal of the project is to make use of the hardware/software codesign in order to provide a robust hardware and object oriented software implementation for the service robot framework.

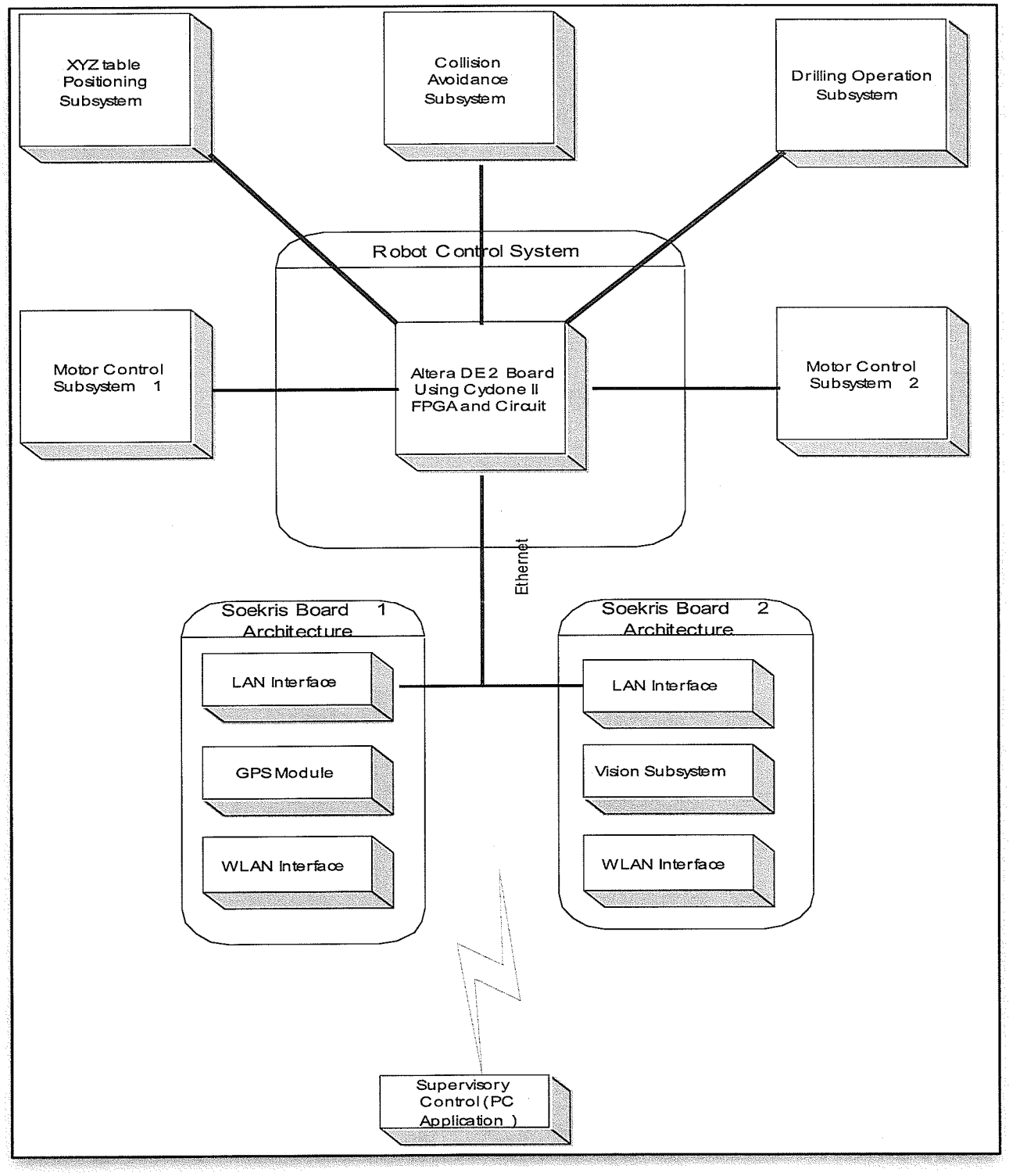

Figure 3.1 Service robot framework architecture

3.2 Hardware Design Consideration

Our service robot framework is comprised of several hardware components working together as a whole. The integration of these components is discussed in more detail subsequently.

3.2.1 Robot Control System

The robot control system comprises of an embedded controller and motor driver to provide all the necessary voltages and currents to the DC motors which are connected to the wheels. This allows the remote operator to turn the wheels in different directions with different speeds. The embedded controller is the brain for the IP centric robot. The embedded controller allows the designer to implement both hardware and software and provides APIs for software applications to interact with hardware modules. The embedded controller provides flexibility to add external hardware peripherals, is equiped with a real time operating system for executing application layer tasks in real time and includes a networking protocol stack. The embedded controller also provides capabilities for general purpose input/output (GPIO) to be connected with motor drivers, sensors and a camera module. The following sections describes all the technologies and hardware we used to build the robot control system comprising the embedded controller and motor driver and all the Altera technologies used to build the control system.

HDL, FPGA and Embedded Controller

Hardware description languages (HDLs) allow hardware designers to model and implement digital circuit designs using a high level structured programming language. An HDL is analogous to a software programming languages (for example, C++) in a

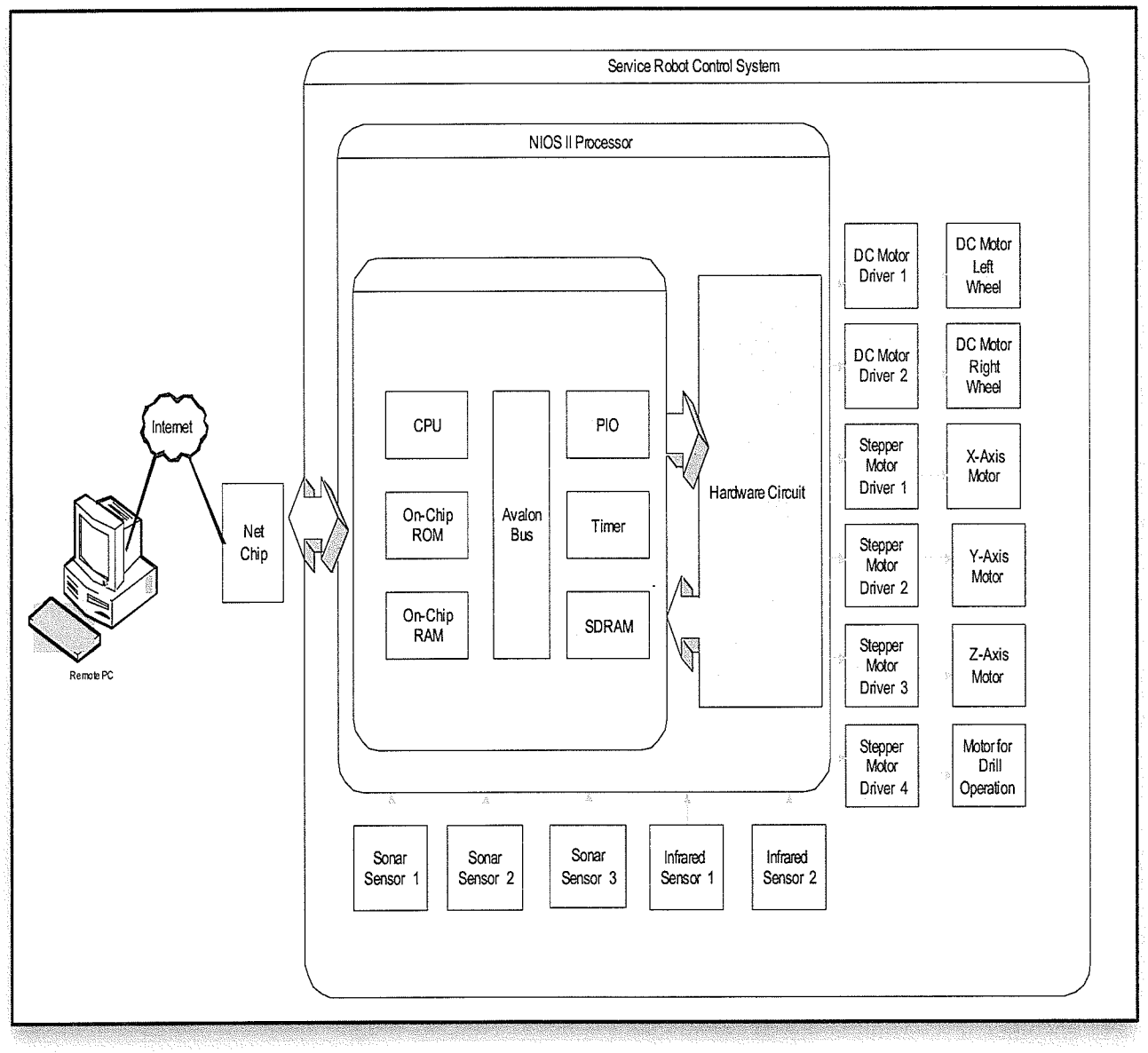

Figure 3.2 Block diagram of the robot control system

sense that both can specify the functionality of the design. HDLs also allow the programmer to describe the timing constraints of the design and allow multiple processes to run concurrently, and independently of each other, where the time and concurrency are the major attributes of a hardware design. This frees the designer to concentrate more on the high level system while the HDL-code listing can be passed on to the synthesis. Synthesis is an automated process of transforming the HDL code into a physically-realizable gate netlist that describes the basic components and connections to be implemented in hardware [28]. Verilog HDL is used in this project to design and implement the hardware circuits to provide the necessary timing constraint and logic signals and for handling concurrent processes for all the motor drivers and physical sensors in the design as illustrated in Figure. 3.2.

HDLs allow the designers to express both the behavioral and structural aspects of each stage in the design and have a method for creating switch level wires (structural) and registers (behavioral). Logic primitives such as gates and buffers are connected using wires and registers and can be combined into larger interconnected modules that can be instantiated, and thus allows for the reuse of existing modules. Altera's Megafunction library also provides pre-tested design blocks or intellectual property (IP) for commonly used circuit functions targeted for programmable devices allowing for a faster development time and greater design integrity [20].

Field programmable gate arrays (FPGAs) comprise the next step of the design process. An FPGA is an integrated circuit that contains an array of reconfigurable logic blocks with a flexible interconnect structure between the logic elements. An HDL design is synthesized to a configuration of logic blocks, and then programmed to the FPGA. In this way, the FPGA can be used to build early prototypes of the final product. FPGAs are portable and consume very little power, thus making them suitable for service robot applications where the robot will be running on batteries

for long period of time.

The project uses Verilog along with the Altera Cyclone II FPGA on an Altera DE2 (see Figure. 3.3) development board which is widely used and easy to develop hardware for. Altera's system-on-a-programmable-chip (SoPC) builder allows the designer to incorporate intellectual property (IP) cores, memories and interfaces to the off-chip devices easily which shortens the design cycle time. This system-on-a-programmable-chip system concept greatly improves the design flexibility and gives greater flexibility for application development. By taking the advantage of a fluid hardware-software strategy, time critical blocks such as the DC motor controller including generating pulse width modulation (PWM) signals can be implemented using Verilog HDL while the remaining application logic responsible for performing control maneuver operations and making decisions for the obstacle avoidance subsystem, can run on the NIOS II soft core processor. The application logic is written using the NIOS II IDE which also provides support for the TCP/IP socket programming implementation, which provides the API for the Altera DE2 board to communicate with the rest of the world using its Ethernet interface. The above mentioned PWM signals and other control signals are sent to the motor driver circuit via the digital I/O (DIO) port. Additional details regarding motor driver hardware circuit will be discussed in the following sections.

The development board has several rich features. These includes the following hardware features; the NIOS II softcore processor, advanced Cyclone II FPGA, advanced I/O features (USB and Ethernet interface) and flexible memories. Design features include user friendly Quartus II software, SOPC builder's easy to add component ability (Figure. 3.4) and a Hardware Abstraction Layer's (HAL's) facility that allows application software programs to communicate with the underlying hardware. These combined makes the Altera DE2 board an ideal platform for implementing a

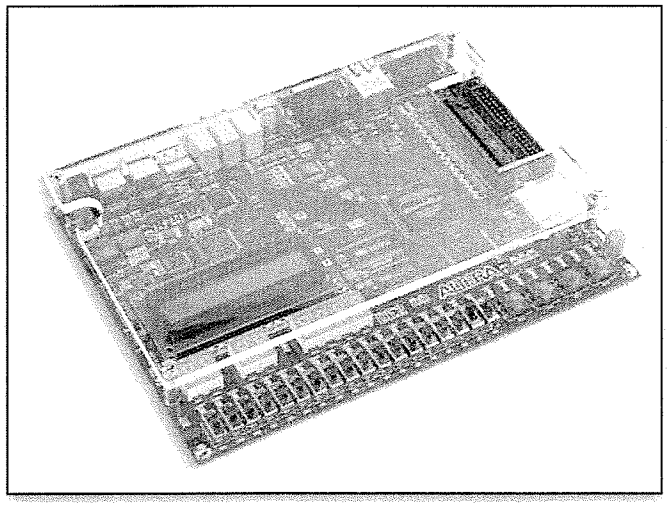

Figure 3.3 Altera DE2

service robot framework.

Motor Driver

As described in Section 1.7, due to the heavy weight (close to 100 pounds) of the chassis and motors, we needed high power motor driver circuits to drive the DC motors. According to the specification described in Section 1.7, we decided to buy off the shelves motor drivers called Open Source Motor Control (OSMC) from Robot Power (see Figure. 3.5) [21]. Each of the motor drivers has 10 pin headers which provides logic interface to the embedded controller and can accept a single logic-level PWM signal with as high as 30 KHz carrier frequency. The OSMC also provides a full H-bridge driver chip which has circuit protection and provides necessary signal and voltages to H-bridge. This motor driver greatly increases the reliability of the control system and satisfies all the requirements including providing necessary voltage and current to drive the DC motors.

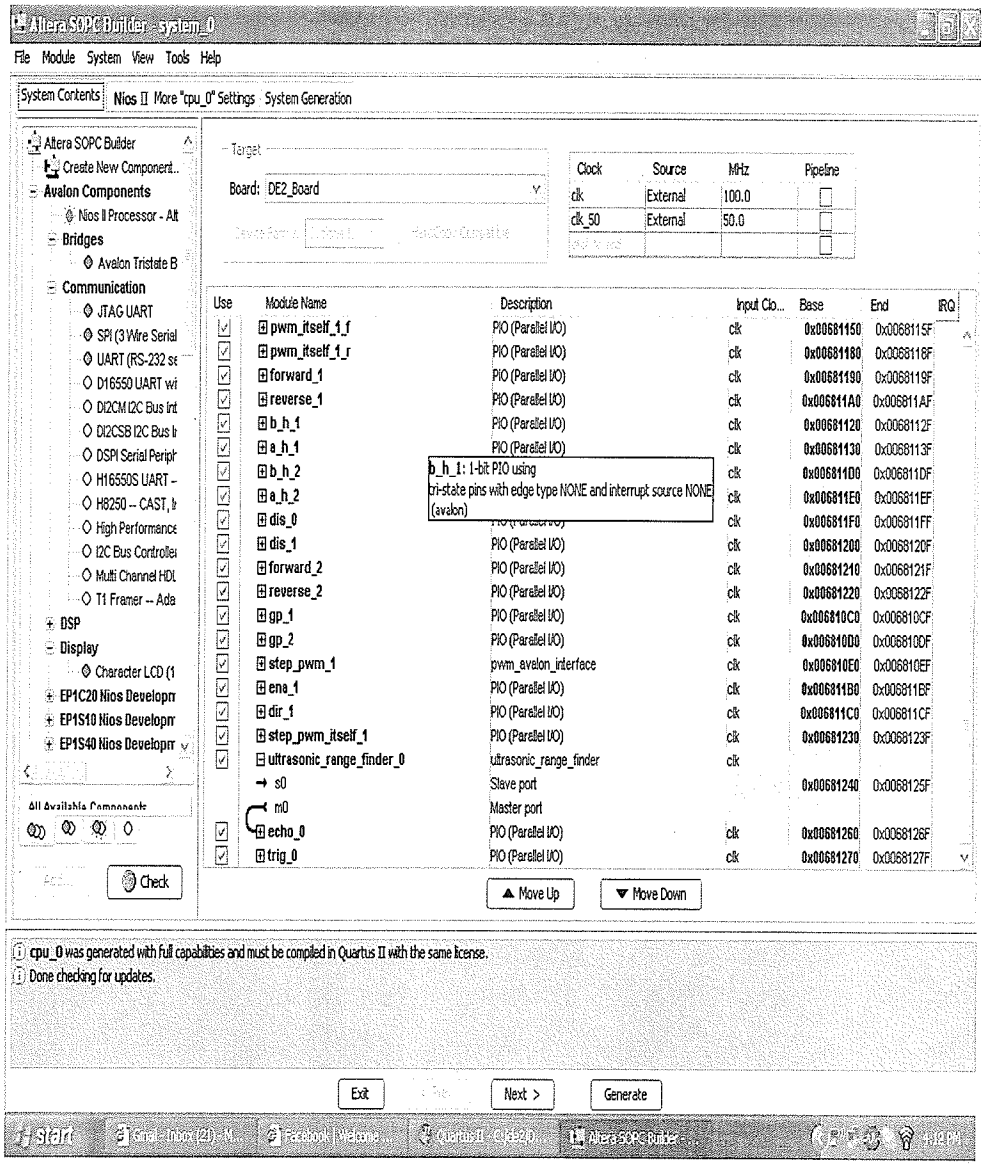

Figure 3.4 Nios II implementation in SOPC builder

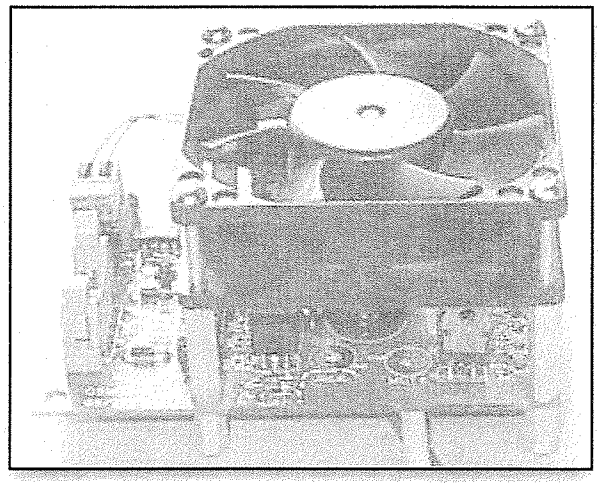

Figure 3.5 OSMC motor driver

Stepper Motor Driver

The system also comprises three stepper motors from powermax (see Figure. 3.6) which operate at maximum of 32 volts with 2.8 Amps of continuous current. We decided to use bipolar motor drivers which provides dual H-bridge support, and operated at a minimum of 30 volts power supply at 2.5 Amps of current. The bipolar motor drivers operate the bipolar motors in full, half, quarter, and eighth-step modes. Each of the controllers requires three logic pins including the PWM signal from the FPGA controller and provides power supply and control logic to the hybrid 8-lead bipolar motors. These stepper motors are used to drive three stepper motors for the purpose of controlling the XYZ table positioning subsystem.

3.2.2 Visual Feedback Subsystem

We were required to use a camera to provide visual feedback subsystem for the service robot framework. There are various types of camera available on the market. We had

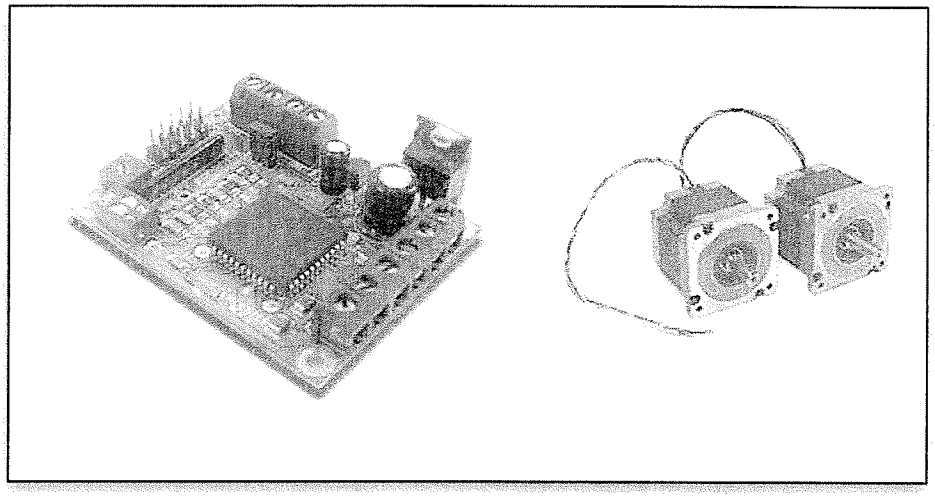

Figure 3.6 Stepper motor driver and stepper motors

several options to choose from:

- USB cameras are widely used in the networking applications. These cameras are easy to integrate into the single board computers with USB interfaces. The main reason for easy integration is the availability of the open source driver for the camera and frame grabbing applications. However, these cameras have low resolution which may not be suitable for image processing application like the one built in here.

- CMOS (complementary metal-oxide-semiconductor) cameras are used for monitoring and security camera applications. These cameras have high resolution and are good for high pixel updating rate.

We decided to use both types of cameras. USB Logitiech camera (see Figure. 3.7) is connected to a Soekris single board computer running a Linux operating system. This can be used to provide a visual feedback to the remote user. Another type

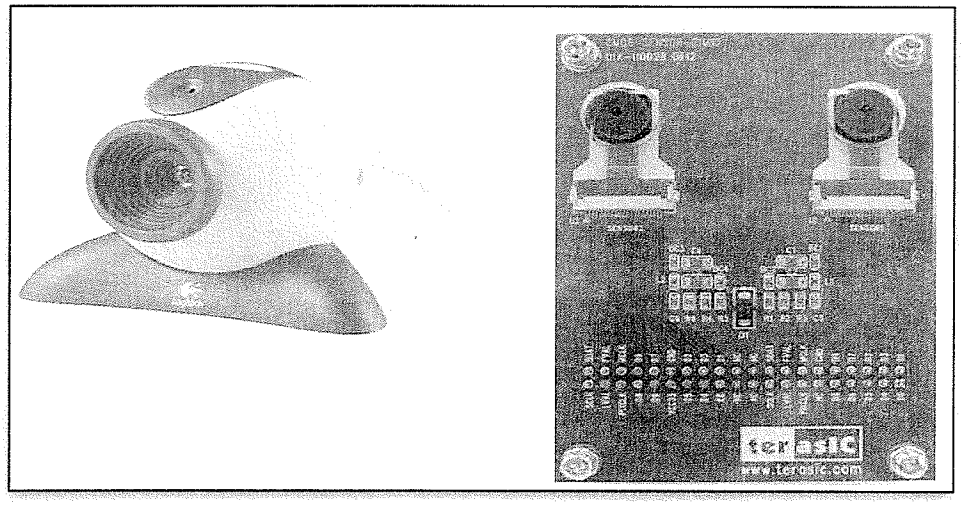

Figure 3.7 USB camera and CMOS camera

that we used is the CMOS camera. We have purchased a Terasic Tehnologies, Inc 1.3 megapixel digital camera module to be connected with Altera DE2 board [29]. This module has a 1.3-megapixel CMOS image sensor chip and a focusing lens. This module was easy to develop and controlled using the FPGA and improve upon the USB camera resolution greatly.

3.2.3 Sensory Feedback Subsystem

Two Devantech ultrasonic range finder sensors (See Figure. 2.12) are used to for measuring distances to the nearest object with resolution between 3cm and 3m. Two other infrared sensors were used in the project which is described in Section 2.7.

3.2.4 User Control subsystem

The Logitech joystick (See Figure. 3.8) is an input device which provides two axis (X and Y) control information of movement of the handheld stick attached to it. In this

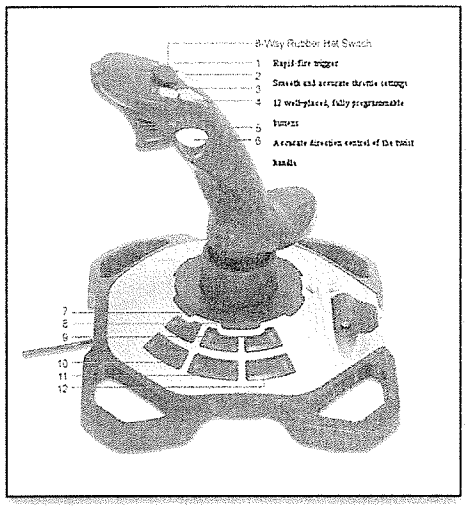

Figure 3.8 Logitech joystick

case the X-axis controls are derived from the steering of the stick from the left to the right providing counts from 0 to 65535 respectively. The Y-axis counts are derived by steering the stick from rear to front and ranges from 0 to 65535. These range values are then scaled down to the range of 0 to 100. The position of the joystick is then interpreted as desired speed and direction. It also has a number of buttons which can be used to perform some other functionality such as emergency stop. A force feedback joystick could also be used and left is as a future work which will add extra features to the feedback subsystem of our framework presented here.

3.2.5 Single Board Computers and Sensor Modules

We decided to use x86-class based single board computers (SBCs) to incorporate vision subsystem, wireless interface and GPS module into the service robot framework as illustrated in Figure. 3.1.

Soekris Board 1 Architecture

The Soekris board 1 architecture illustrated in Figure. 3.1 consists of a Soekris net4826 embedded single board computer, x86-based 233 MHz AMD Geode processor, 64MB SDRAM, 64MB flash storage, 2 miniPCI ports and serial port connectivity [30]. The system is enclosed in a NEMA enclosure for outdoor use. As an x86-based architecture, the Soekris board supports Linux, BSD, and a variety of other operating systems. We have chosen Wistron Neweb CM9 802.11 a/b/g wireless cards [31] to use in one of the miniPCI slots (See Figure. 3.9). This card has 100 mW of transmit power which works without an external antenna and is built on the Atheros chipset, which has widely available drivers for Linux. This card is used as an access point (AP) and is responsible for relaying data between PC, embedded controller and Soekris single board computers.

Soekris Board 2 Architecture

The Soekris board 2 architecture illustrated in Figure. 3.1 is consists of a Soekris net5501 embedded single board computer, x86-based 433 MHz AMD Geode processor, 128MB SDRAM, 2.5" Hard Drive, USB and serial port connectivity [32]. In order to provide vision system a Logitech QuickCam was selected for capturing real time video image. The QuickCam is equipped with a CMOS image sensor with a still image resolution of 640x480, and can operate on Linux based platforms. The Logitech webcam is connected with the Sokeris net5501 using a USB interface (See Figure. 3.9). This USB based camera was considered to provide sufficient frame rate over wireless connection and an inexpensive option for this design.

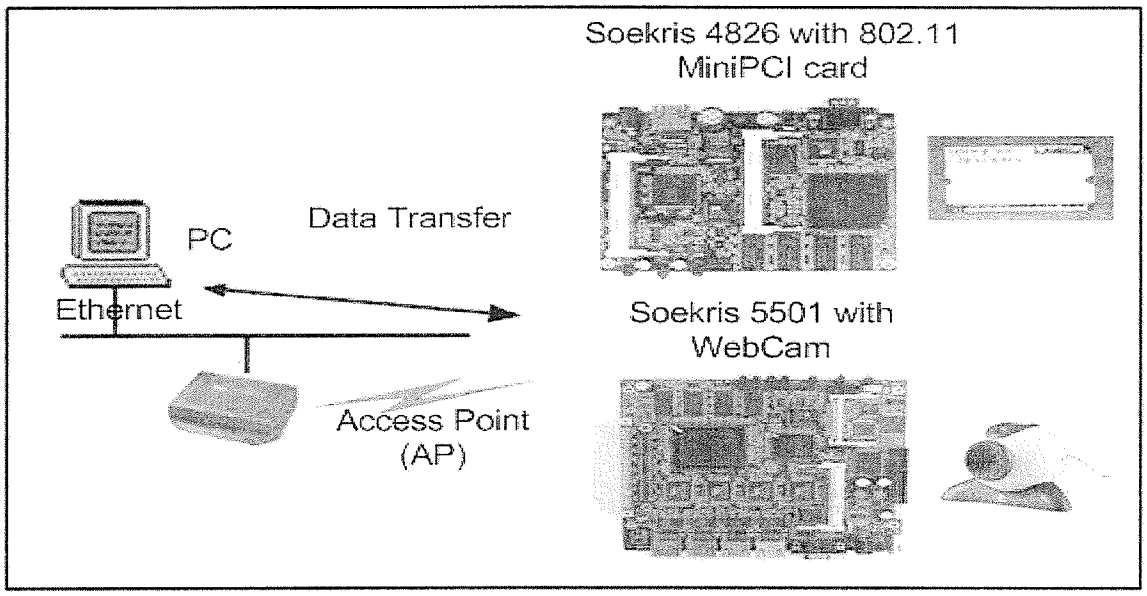

Figure 3.9 Soekris board design with sensor modules and wireless interface

3.3 Software Design Consideration

The software architecture for the service robot framework employs a client-server paradigm as illustrated in Figure. 3.11.

3.3.1 PC Hosted Supervisor Control Software

A Socket based client software was written in Microsoft C # which receives two-axis commands from the joystick continuously, which in turn calls the command interpreter and converts a command into appropriate duty cycles and directions for each wheel and sends it to the robot server. The supervisor PC also has functionalities to receive sensory feedback and image data. The received image data is then buffered and displayed on the monitor providing visual feedback to the operator. Implementation of client software module will be discussed in Section 4.2.3.

3.3.2 Graphical User Interface (GUI) Development

The user interface is designed with the intention of making it simple for any user to interact with the mobile robot. A simple graphical user interface is implemented to provide information on visual feedback, sensory feedback, control and status information about the mobile robot. The GUI is divided into four panels. The top panel allows users to connect to a mobile server using its IP address and socket port. The left panel is defined as a control panel and it consists of four direction buttons. The user can control the mobile robot by clicking the direction button on the control panel. A joystick provides more flexible controls of the robot, thus the control panel also shows two dimensional axis positions of the joystick control and corresponding speed and direction calculated from our software program. The right panel shows the visual feedback in a continuous jpeg image with 640x480 pixels resolution. The bottom panel which is defined as a feedback panel shows the feedback information from all the infrared and sonar sensors. It also shows the geographical position and speed of the robot. At the time the video panel is not quite functional but the video can be displayed on the operator console in a web browser.

3.3.3 Server Robot Software

A Socket based server software program was written for the robot side, which listens for a client PC connection and once it is connected it will execute tasks according to the message received from the client. The server software is responsible for sending the video stream and sensory feedback to the client PC application program. The additional but major functionalities provided by the server robot program are semi-autonomous operations and obstacle avoidance algorithms as illustrated in Figure. 3.10. Implementation of the robot server module will be discussed in Section

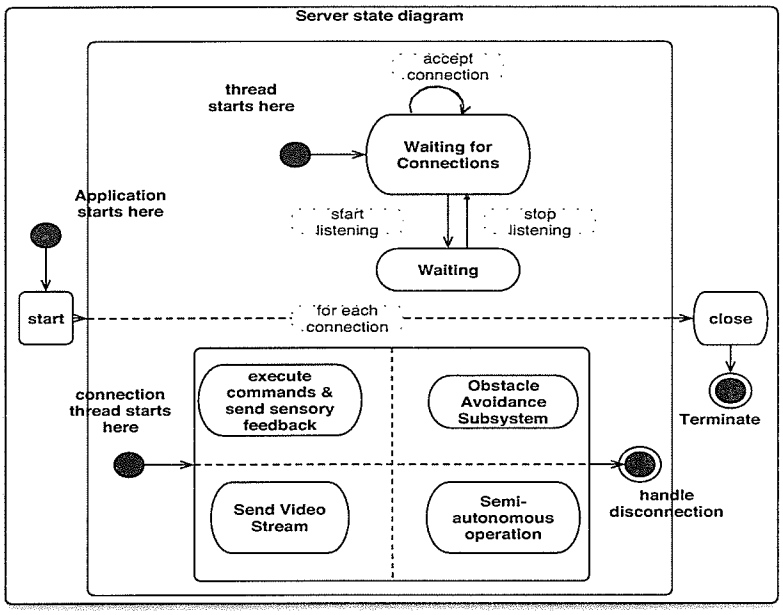

Figure 3.10 Server side state diagram

4.2.2.

3.4 Summary of Chapter 3

This chapter described the basic architecture of the framework, hardware and software design considerations associated with construction of the complete service robot framework. Initially the project started with building components on the top of chassis with four wheels. The robot chassis was equipped with two powerful DC motors but there were no motor drivers for it. We decided to get off the shelf DC motor drivers which satisfied the functional requirements necessary for driving those motors safely and efficiently. An FPGA controller was required to provide all the necessary logic signals to the DC motor drivers, stepper motor drivers and ultrasonic sensors used in the project. A joystick was used in the project to provide all the necessary

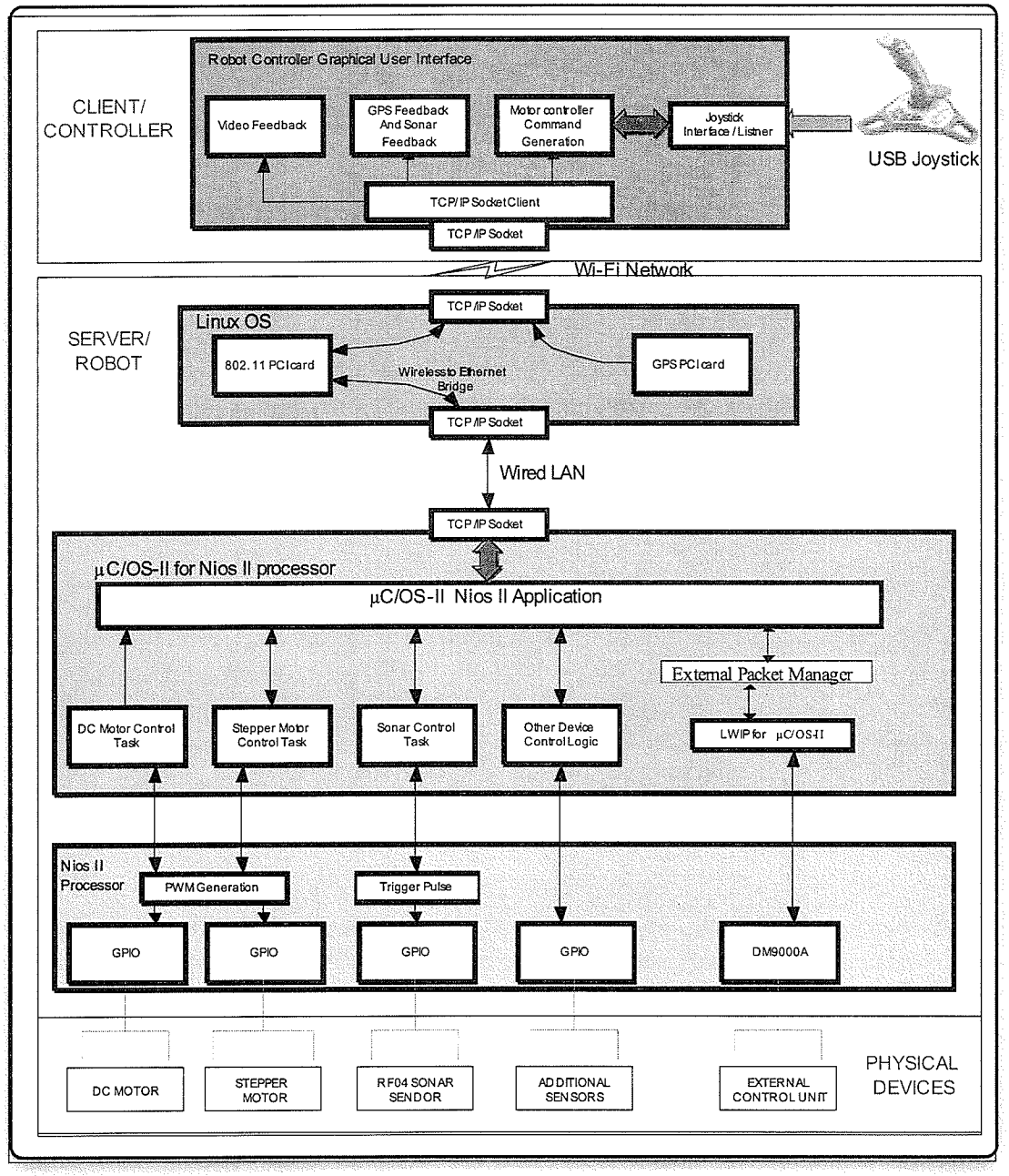

Figure 3.11 Server robot software architecture

control information by the user sitting at the client PC. A socket based client program was written to acquire the joystick information and send it over the network. A graphical user interface was added to facilitate user input and display visual image and the robot's status. A server program is also running on the mobile robot side to execute the task assigned by the client program. Two Soekris embedded single board computers were added in the project later to incorporate WLAN and webcam interface to facilitate the video stream over the 802.11 WLAN.

Chapter 4

Design Implementation

Parts of this chapter were done in collaboration with Venkateswara Reddy. We both developed versions of what was required but the final version used usually was the one that had greater functionality. Specifically instances of what was eventually done by the individual collaborators are itemized as footnotes in this chapter.

4.1 Hardware implementation

Initially the hardware available for this project was the robot chassis with two DC brushless motors, two small car size 12 volts batteries (24 volts in total) for each drive unit and four wheels. Due to the heavy weight of the chassis and motor type, we decided to use two powerful DC motor controllers. These motor controllers provide all the necessary current supply and equipped with built-in protection circuit. But they didn't have any local intelligence associate with them; therefore we needed to provide FPGA-based embedded controller to provide all the necessary logic signals including the Pulse Width Modulation (PWM) to the motor controllers. Thus, this embedded controller provides APIs to the motor controllers, stepper motor controllers

and provides necessary trigger signal to each sonar based sensors. The embedded controller also receives the echo signal from each of the sonars and calculates the distance between the robot and any object ahead in its way. All the hardware design associated with the FPGA controller and power electronics implementation for the service robot are discussed here.

4.1.1 Motor control subsystem

Pulse Width Modulation

Pulse Width Modulation (PWM) IP core has been implemented in the Altera cyclone II field-programmable gate array (FPGA) using the Verilog hardware description language (HDL).[1] The core is designed to operate at any frequency up to 50 MHz. A PWM circuit has been described in details in Section 2.5. The basic idea is to produce a pulse train with some on-to-off duty cycle ratio ranging from 0 to 100 %. Two registers are used to store values and load the Period and Duty Cycle. The register UP Counter is incremented at every rising edge of a clock cycle. The Period register defines the number of clock cycles during one cycle of the PWM output [33]. When the counter value reaches the value in the period register that means it completes one cycle, it is then reset to 0 to start a new cycle. PWM output is low throughout the cycle unless a value is loaded in the Duty Cycle register; the PWM output is kept high while the counter value is less than the value in the Duty Cycle register. The main advantage of implementing the PWM in hardware is that the output pin is continuously updated that means it goes high at the exact clock cycle depending on the value in the Duty Cycle register. Software is used to select the value of the period and duty cycle using the register map.

[1]With the specific task of the motor controller I was responsible for implementing PWM logic blocks in the FPGA.

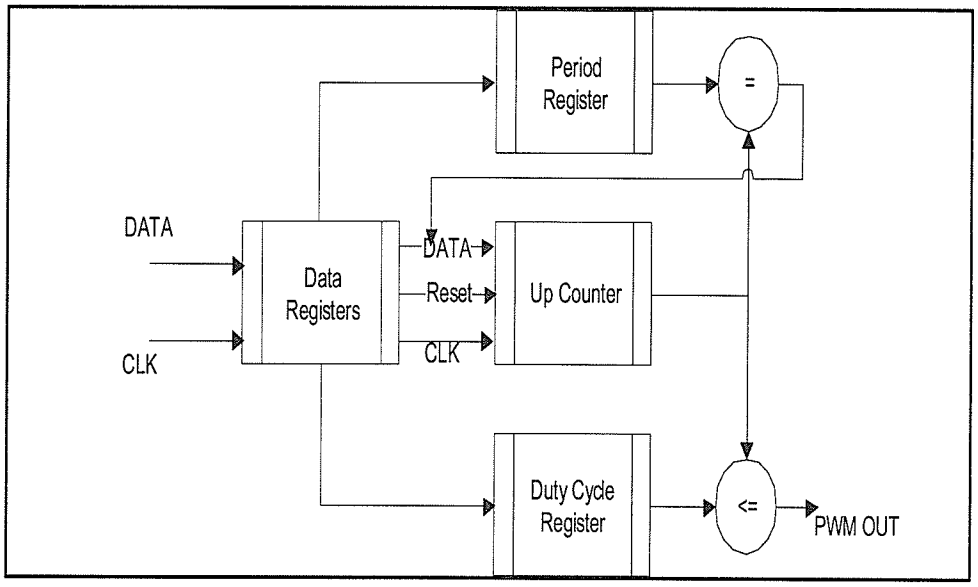

Figure 4.1 PWM Logic Diagram

The functional description of the PWM core was synthesized using Verilog and simulated using ModelSim editor. The simulated results are shown in the waveforms in Figure. 4.2 with period =10 and duty cycle =7. The PWM output is taken out on one of the output pins of the GPIO of the Altera DE2 board and can be observed on an oscilloscope. We need to produce two PWM output with same period but with variable duty cycles.

Other control logic

Other control logic for the motor driver are implemented in a hardware according to the truth table provided in a Table 2.3.[2]

[2]Venkateswara Reddy was responsible for implementing this portion of design in the FPGA.

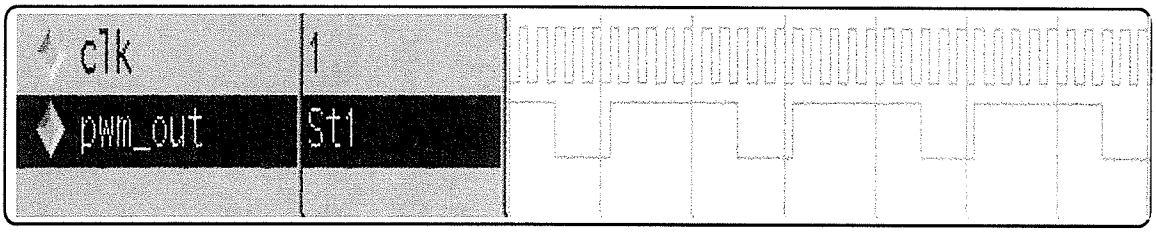

Figure 4.2 Simulation result

4.1.2 Sensory feedback subsystem

The service robot presented here is equipped with two sonar sensors and two infrared sensors which are connected to the FPGA controller. The sensory feedback subsystem is responsible for acquiring distance data from the sensors modules and sends data to the fuzzy controller for further processing. In order to do so, the FPGA needs to be able to communicate with the sonar modules. All the necessary logic blocks for generating trigger pulse, receiving echo signal and estimating elapsed time between the trigger pulse and echo signal are implemented using the Verilog hardware description language.[3] The connections are shown in Figure. 2.12 and timing diagram is illustrated in Figure. 2.11.

The controller initially generates a trigger pulse on its output pin and the pin is connected to trigger pin of the Devantech sonar sensor. The trigger input is valid or ON for 20 microseconds to allow the sonar module to issue a sonic pulse. The pulse travels through the air; hits an object if there is any on its way and is reflected back to the sonar range module. The range finder enables the receiver 100 microseconds after the trigger and raises the sensor's echo output signal. The echo signal is valid for 50 milliseconds unless it receives any reflection from sonar waves before that. If there is a time out no object is detected and the echo signal will be low. If there is

[3]This portion of the work was done in collaboration with V. Reddy.

any object detected within the period of 100 microseconds to 18 milliseconds after the trigger was issued, there is an object ahead of it and the embedded controller counts the elapsed time by keeping track of the clock cycles. The distance is then calculated in centimeters using (2.2). The FPGA controller waits for 10 milliseconds after the object is detected or time out occurs and then issues a trigger signal again. The flow chart for obtaining elapsed time from the sonar module is illustrated in Figure. 4.3.

4.1.3 Image Capturing Module

The TRDB-DC2 camera is connected to the expansion interface of the DE2 board which can be read/write through digital input/output (DIO) pins. This module has a 1.3-megapixel CMOS image sensor chip and the main features are:

- array format: 1280 X 1024 (1.3 megapixels).
- frame rate: 60 frames per second at frame size 640 X 480.

In our image capturing system, we used the TRDB-DC2 camera module with Verilog source code for image capture.[4] The image capture code can display the image on a VGA display. The image capturing is illustrated in Figure. 4.4 and operates as follows:

- Configure the image sensor using the I^2C bus.
- Read the raw data output of the camera which uses a *Bayer* color pattern [34].
- Use the algorithm of *"Nearest Neighbor Interpolation"* [35] to convert the Bayer color space to get the complete image which represents each pixel with a triplet of RGB values.

[4]This aspect of the work was done in collaboration with V. Reddy.

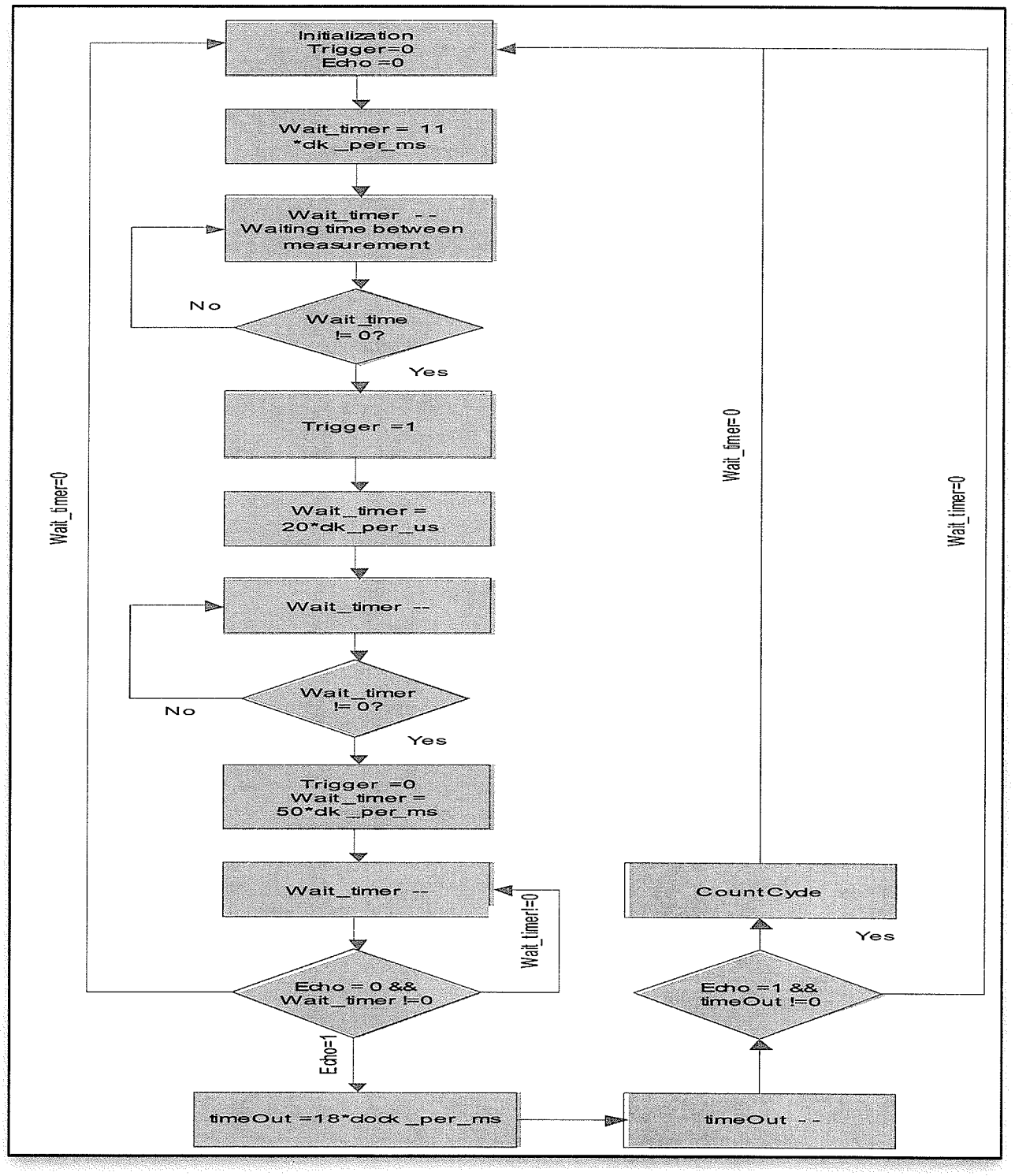

Figure 4.3 Sonar ranger flowchart

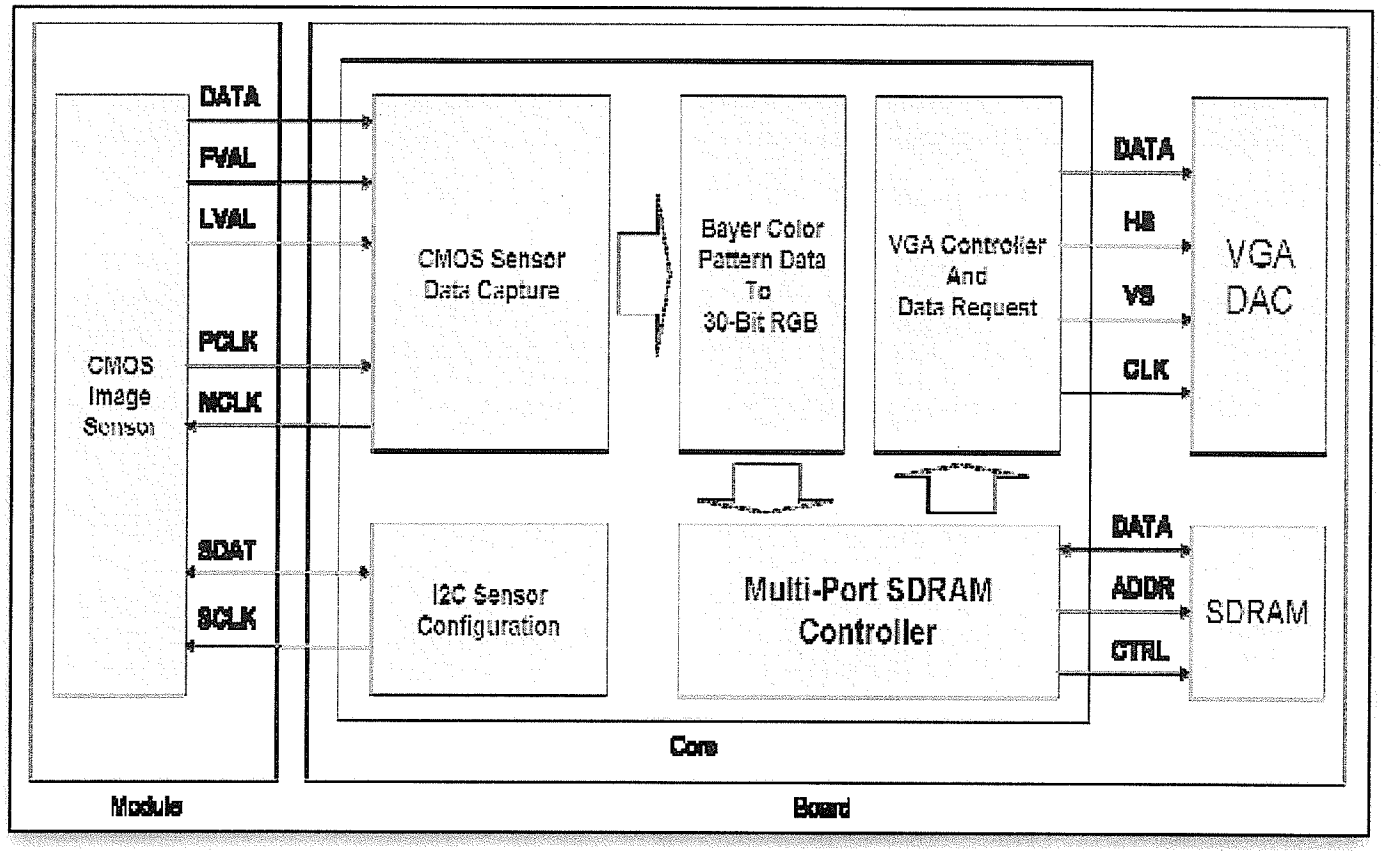

Figure 4.4 Image capturing hardware circuit [36]

- Save the image data to SDRAM.
- The VGA display code reads the data from the SDRAM.

Fast and efficient image processing can be done without implementing the complex algorithms in an FPGA or without the use of limited memory and slower embedded processor provided by the Altera DE2 board. The data output captured by the FPGA controller can be transferred to a HPC (high performance computing) resource for image processing. The use of a HPC resource has been demonstrated for image

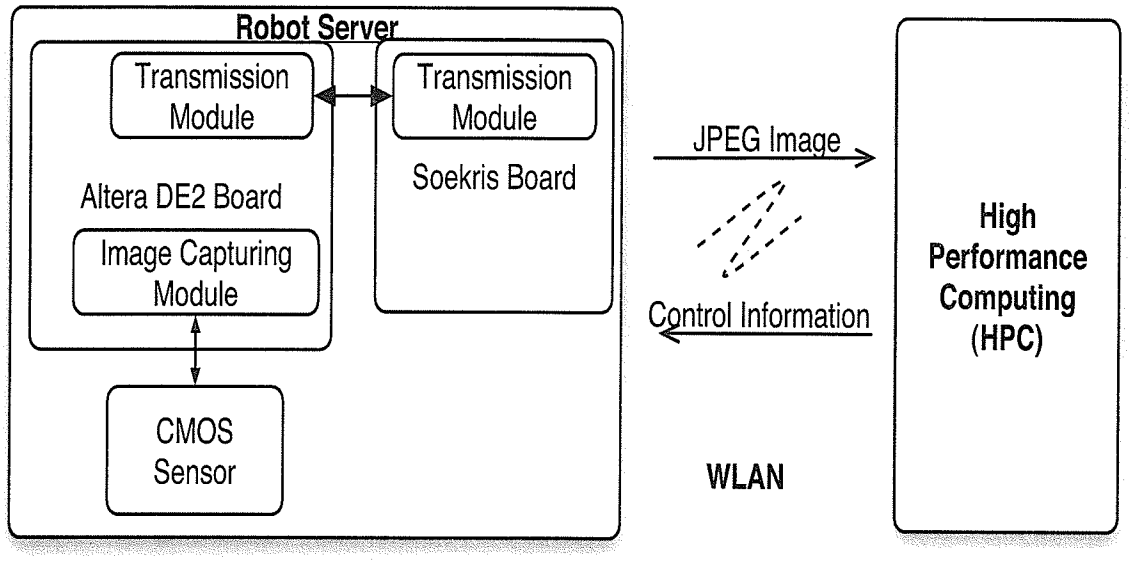

Figure 4.5 Image processing architecture

processing and for making decision on weed location.[5] Based on this information the robot can locate and perform weed removal operations within short period of time and without any intervention from the remote user. This later aspect is essentially scripted control of the stepper motor which was my responsibility. The proposed image processing architecture is illustrated in Figure. 4.5.

4.1.4 Power electronics

As mentioned, the robot chassis used in this project weights around 100 pound including two car batteries. Including some extra weight of around 25 pounds it can adds up to about 125 pounds in total. The two powerful DC motors used in the project can draw current of about 30 Amps at 24 volts. Thus, we decided to use off the self open source motor controllers (OSMCs) which are capable of supplying

[5]Work of M. Laskowski.

high current to the motors. The OSMC is robust H-bridge amplifier which is capable of operating over a wide input voltage range and at high currents as described earlier in Section 2.5.3. An H-bridge driver has four switched legs, and the OSMC uses four MOSFET devices in parallel for each leg as illustrated in Figure. 2.8 [24]. Various protection components, including TVS diodes and Zener diodes are used to protect the components from possible motor transients [24]. The HIP4081A driver chip provides the logic interface to all N channel MOSFETs as described in Section 2.5.3. The 12 volts power supply for the two fans, the HIP4081 driver, and other components are provided by one of the two car batteries. The functionalities of the OSMC are extended by providing a FPGA controller that controls the OSMC and communicates with the external world through an Ethernet port.[6] The circuit board is shown in Figure. 4.6.

4.2 Software Implementation

4.2.1 The Application Layer Command Protocol

An application layer command structure was constructed in such a way it is very simple and consists of a message exchange protocol.[7] The robot client sends a message to the embedded controller every time there is input from the user. The robot client either sets the direction and speed of the mobile robot or enquires as to the status of the sensors attached to the robot. So, there are two types of message requests. One type of request message is called COMMAND with three parameters in order to control the robot. These parameters are *direction*, *dutycycle1* and *dutycycle2* as illustrated in Figure. 4.7, Figure. 4.8. These COMMAND messages are sent from

[6]This portion of the work was done in collaboration with V. Reddy.

[7]The protocol was designed jointly with V. Reddy.

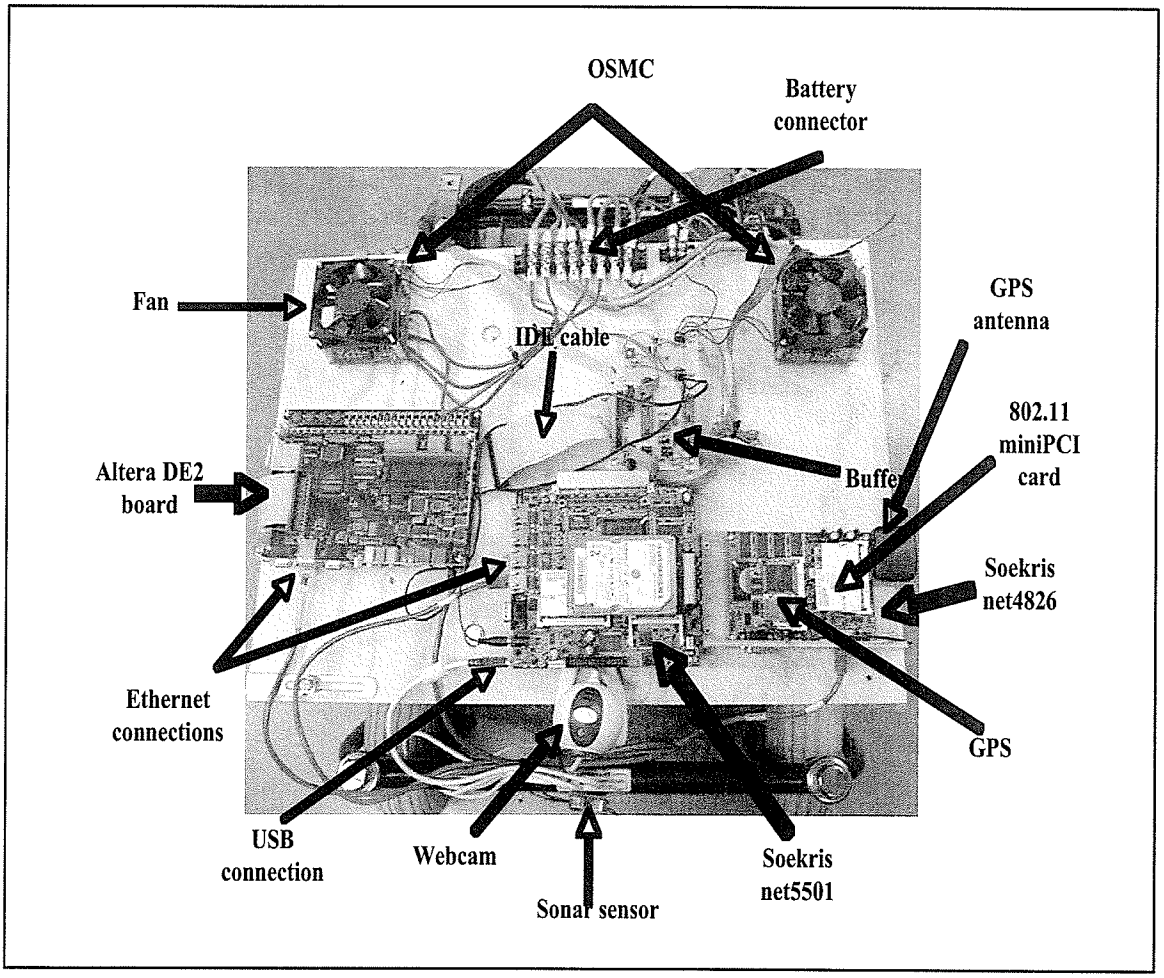

Figure 4.6 Robot controller circuit boards

the client to server to indicate that the server has to execute the task according to the message instantly. The COMMAND messages specifies the direction of the robot and duty cycles for the both motors. Another type of message is called STATUS (see Figure. 4.9). These messages are sent from the client to the server to get the sensor readings from the server. This type of message requires a response (Figure. 4.10) back from the server controller. The STATUS message consists of a single parameter.

The COMMAND message consists of a value which is one string character followed by optional parameters. Each of the COMMANDs is represented by a string character for simplicity and the parameters are digits from 0-100 to represent the duty cycles for both wheels. Table 4.1 lists all the command with respective parameters. The STATUS message consists of a value which is only one string character "X" or "Y". X means client is requesting the status of sensor 1 and Y means client is requesting the status of sensor 2. When the server receives the message "STATUS", it sends a RESPONSE message to the client in return. The RESPONSE message consists of single sonar *reading* corresponding to the sensor number and is displayed in a GUI on the client side.

For example, if the robot client has to send a command to move the mobile robot into to the right direction with different speeds for the two wheels, the message will consists of a single character value 'R' with left wheel and right wheel speed as shown in Figure. 4.7. Dutycycles take value from 0 to 100, the first parameter represents the speed of left wheel and second parameter represents the speed of right wheel.

4.2.2 Robot Server Module

Our proposed server robot framework was illustrated in Figure. 1.5. The server robot program runs on the embedded processor (Altera DE2) which acts as a *server*. The Soekris net4826 is used to provide bridge between wired and wireless network

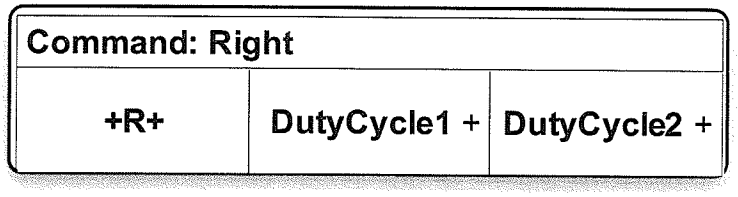

Figure 4.7 COMMAND message shows command Right with two parameters

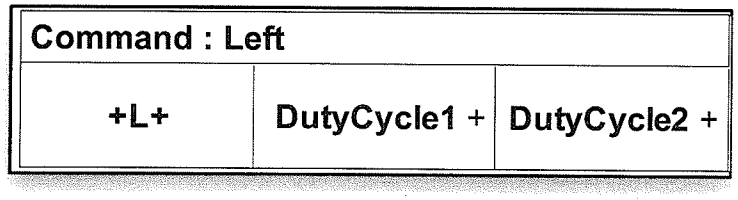

Figure 4.8 COMMAND message shows command Left with two parameters

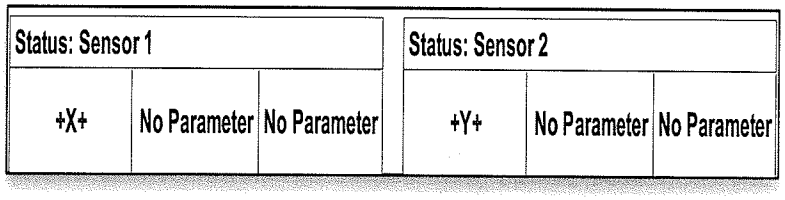

Figure 4.9 STATUS message shows request Status with no parameters

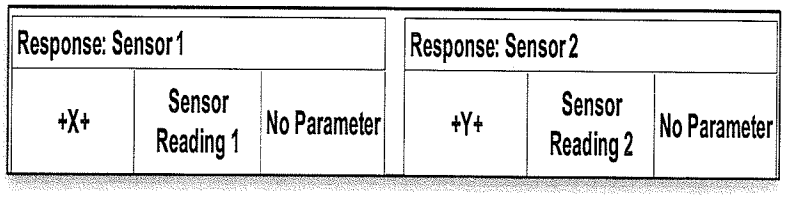

Figure 4.10 RESPONSE message shows Response with one parameters

Table 4.1 Command Values

Command	Value	Parameters	Response	Comment
NULL	N	None	No	Keep alive
RIGHT	R	Two	No	Going Right
LEFT	L	Two	No	Going Left
FORWARD	F	Two	No	Going Forward
BACKWARD	D	Two	No	Going Reverse
CLOCKWISE	Z	Two	No	Clockwise Spinning
ANTICLOCKWISE	Q	Two	No	Anticlockwise Spinning
STOP	P	None	No	Smooth Stop
EMERGENCY STOP	S	None	No	Hard Stop
Infrared SENSOR STATUS	U	None	Yes	Read Infrared Sensor 1 Status
Infrared SENSOR STATUS	V	None	Yes	Read Infrared Sensor 2 Status
Sonar SENSOR STATUS	X	None	Yes	Read Sonar Sensor 1 Status
Sonar SENSOR STATUS	Y	None	Yes	Read Sonar Sensor 2 Status

and Soekris net5501 is used to capture video images and send visual feedback to PC/desktop client on the WLAN. The robot server application program is responsible to carry out tasks defined by the following software modules:

- Robot Driver[8] - When a command is received from the supervisory control module (see Figure. 1.5) to maneuver the robot with different direction and speed, the robot will perform the command immediately. If the command requires it the robot will respond with a sensory status message. We have facilitated the handshake with socket programming which provides the API between the

[8]V. Reddy was responsible for this portion of the work.

application and TCP layer. In this way the server and client application software can connect to each other using a point-to-point communication channel provided by TCP with binding to a socket at each end of the connection. The server and client programs are able to read and write to the socket streams bound to the connection. The structure of this software is very straightforward and is illustration in Figure. 4.11. The first step is to create an initialization phase in which the sockets and associated data structures are created. In the second step the server waits for a connection to be established by the client program. The third step is when the server waits in an infinite loop to receive commands from the client, and then executes tasks according to the received message string. The pseudo-code shown below illustrates the structure of the server code with Berkeley sockets.

```
Phase 1
step 1:= Create a TCP stream socket, using the socket() function call:
listenSocket = socket(AF_INET, SOCK_STREAM, 0);
if (listenSocket < 0) {
    cerr << "cannot create listen socket";exit(1);}
Phase 2
step 2:= Bind the socket to the port that client connects to:
serverAddress.sin_family = AF_INET;
serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
serverAddress.sin_port = htons(listenPort);
if (bind(listenSocket,(struct sockaddr *) &serverAddress,sizeof(serverAddress))<0) {
    cerr << "cannot bind socket";exit(1);}
step 3:= Wait for clients:
listen(listenSocket, 5);
step 4:=Establish a connection with client
clientAddressLength = sizeof(clientAddress);
connectSocket = accept(listenSocket,
                       (struct sockaddr *) &clientAddress,
                       &clientAddressLength);
if (connectSocket < 0) {
  cerr << "cannot accept connection ";exit(1);}
```

```
step 5:= Close the listening socket
close (listenSocket);
Phase 3
step 6:= Loop checking for received commands from Client
while ()
{   if read(..)
    {// received a command
    // first parse the command value
    str = strtok(message, "+");
    cmd = strtol (str);
    switch (cmd)
    {   case:MOTORCOMMANDS  // set motor speed and direction
        direction =strtok(message,"+");
        speed = atoi(strtok(message,"+"));
        case:STATUS
        send ();            // send sensor status
        case:NULL           // do nothing
    }}
    else // nothing received
}
```

Within the phase 3, the standard C library function strtok () is used to parse the received command message (See Figure. 4.7). There are three different kind of commands, such as *motorcommands* (See Figure. 4.7) , *status* and *null.* Strtol () converts the numeric digits into the command value *cmd.* A switch structure is then used to select the control code corresponding to the received command value. If the robot server receives the *motorcommands*, it extracts the directions code and duty cycle values from the rest of the message structure where each of the parameter is separated by a "+" sign. The server flow chart is illustrated in Figure. 4.12. If nothing has been read from the socket connection, the robot server continues to perform its local operations, for example, it can still be moving while listening for the message from the client and it will continue

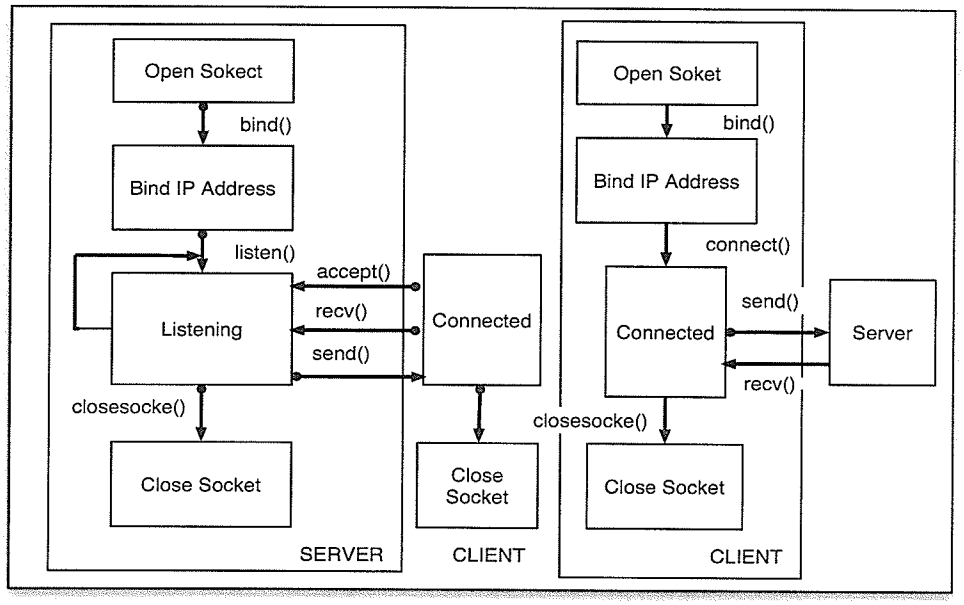

Figure 4.11 Socket based general client server architecture

monitoring all the sensors status for obstacle detection.

- Visual Feedback Subsystem[9] -

  The USB based Logitech webcam is attached to Soekris net5501 single board computer. The Soekris platform is running a stable version of Debian Linux [37]. We decided to use the VideoLAN project for the video streaming from the USB webcam [38]. The VideoLAN solution includes VLS (VideoLAN Server), which can stream MPEG-1, MPEG-2 and MPEG-4 files and a VLC (VideoLAN Client), and can be used as a client to receive, decode and display MPEG streams under different operating systems [39]. The VLS driver was installed under Debian Linux on the Soekris net5501 platform in order to capture the image data and stream it from the webcam to the VLC running on the Desktop PC where it is displayed.

[9]Venkateswara Reddy was responsible for implementing this portion of design work.

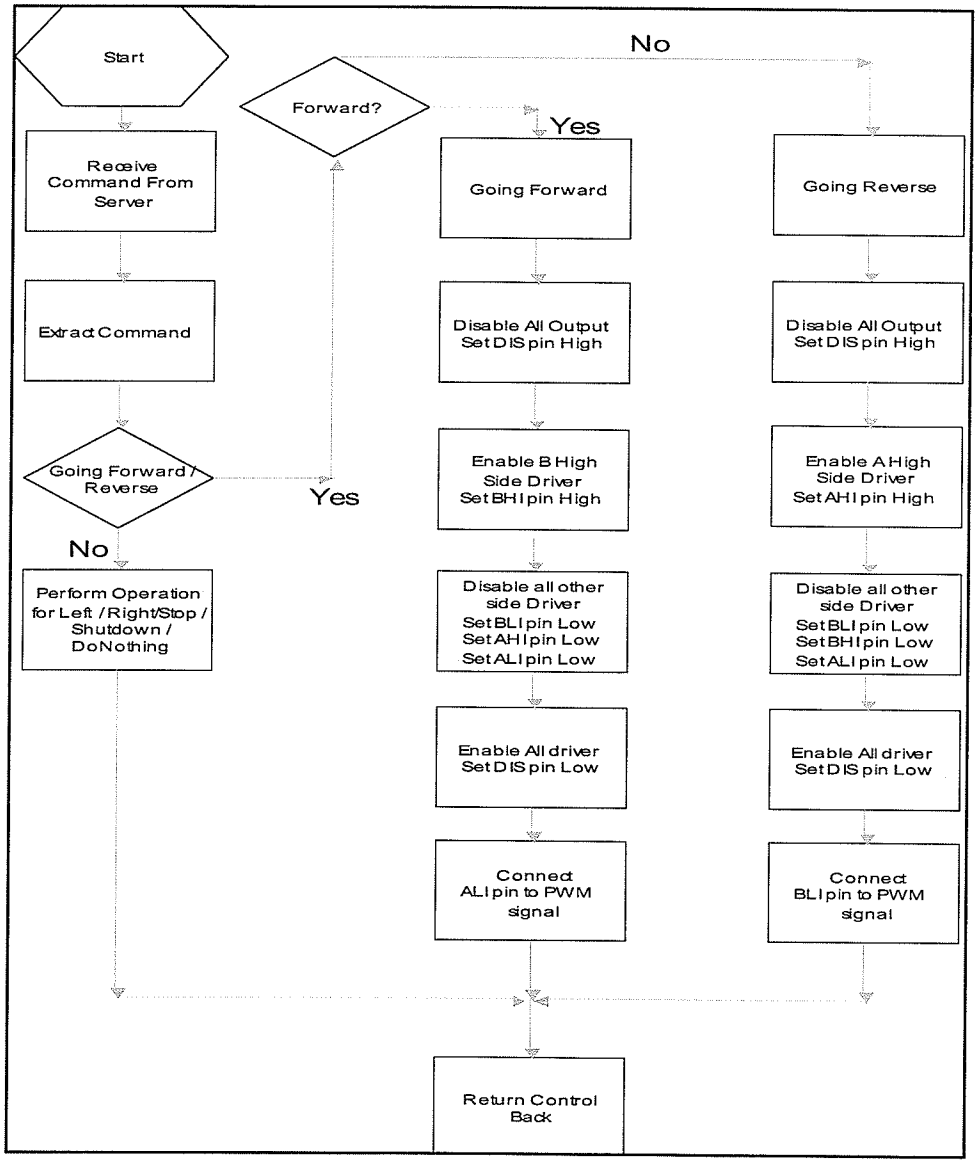

Figure 4.12 Server flow chart

- Obstacle Avoidance Subsystem[10] -

  A number of sonar sensors are incorporated for a robust obstacle avoidance subsystem. This provides additional information on the surrounding environment that may not be available from the vision subsystem alone. Due to the network delay and low transmission rate over the WLAN, a vision subsystem may not be able to provide the real time feedback which makes it harder for the teleoperator to make any decision as he only be able to see the video stream of few milliseconds or seconds ago. By the time the operator sees an object, the mobile service robot might have already struck the object ahead of it. In order to avoid such an accident, a local intelligence algorithm is introduced which is biologically motivated. The sensor inputs derived from the FPGA controller can be fed into the fuzzy logic controller as illustrated in Figure. 2.2. The fuzzy logic controller then constructs degree of membership based on data from two sonar sensors as illustration in Figure. 2.3. This membership functions determines four types of obstacle location categories:

  - The object is very near; the distance is around 0 to 100 inches.
  - The object is near; the distance is 50 to 110 inches.
  - The object is far; the distance is 100 to 250 inches.
  - The object is very far; the distance is 250 inches or more.

  The fuzzy inference engine then combines these two input membership functions and transforms them into one output membership function using a fuzzy rule base. The fuzzy rules are described in Table 2.2. This output membership function is directly related to the control of the both motor's speeds. Four

[10] Although of much of the project was collaborative this aspect of the collision avoidance using fuzzy inference was my own.

types of speed categories are determined by the output membership function. They are:

- Emergency Stop the motors; multiply the duty cycle of both motors by 0
- Motors will be reduced in speed to Very Slow; multiply the duty cycle of both motors by 0.2 - 0.6
- Motors will be reduced in speed to Slow; multiply the duty cycle of both motors by 0.4 - 0.8
- The speed will remain the Same; multiply the duty cycle of both motors by 0.6 - 1

After the output membership functions are evaluated, their combine results are calculated using the Centroid of Area method as described in Section 2.4.3. The final crisp output is then multiplied by the duty cycles for the both motors, thus overriding the supervisory control and performing a local intelligence of its own as illustrated in Figure. 4.13. This low level sensory responses are the part of the semi-autonomous activities of the service robot framework and are quick enough to avoid any obstacle. The whole idea can be compared to a human's neural responses to touch or feel and react quickly to avoid a hot surface. A control signal is also displayed to the operator so that they are aware of the action taken by the robot.

- WLAN Driver Installation [11] -

The Soekris board net4826 is used as an access point (AP) to provide the wireless interface to the robot. Pyramid Linux [40], which is Ubuntu based, was installed in the net4826 embedded platform. The installed file from Pyramid Linux image

[11] This aspect of the work was done in collaboration with V. Reddy.

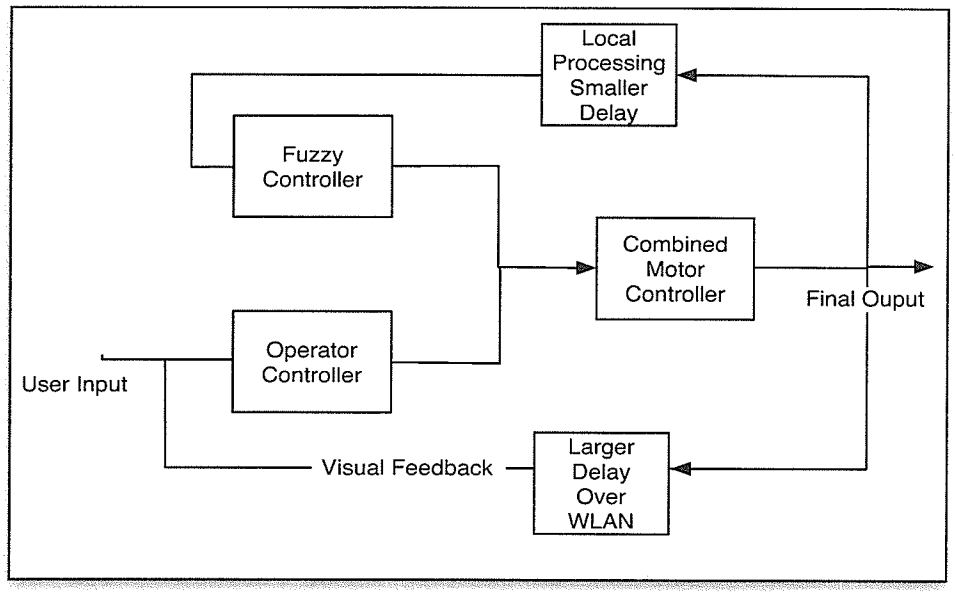

Figure 4.13 Collision avoidance subsystem

occupies only 48 megabytes and can easily fit into the 128 megabytes of soldered flash storage. The MadWiFi driver was installed and configured, which acts as a driver for the Atheros wireless NIC used in this project. An Ethernet Bridge was build between the wireless and the wired NICs as illustrated in Figure. 3.11.

4.2.3 Client Module

Our proposed client framework is described in Figure. 1.4.[12] In a client PC running at the spoke of the WLAN, an operator is able to send supervisory control information with the help of either a joystick or by directly using the buttons of the GUI. The client code makes use of Winsock which is a Berkeley sockets API for Windows. All the associated libraries and codes are also available with most Windows programming

[12]I was responsible for the design and implementation of the client design and joystick control algorithm.

environments, including Microsoft Visual C #.

Joystick Control Algorithm

A joystick was the primary choice as an input device for controlling the speed and direction of the two drive wheels of the mobile robot. The joystick position can be represented as $\bar{X}$ and $\bar{Y}$ as illustrated in Figure. 4.14. Both of the vectors together represent how far and towards what direction the joystick has moved from the origin. Joystick position can be represented as a vector [41]. The direction of the vector is decided based on which axis it resides on. The magnitude of the vector is purely based on the joystick offset, which is the location from the origin (X=0, Y=0). By moving the joystick handle to the positive Y axis means to drive the wheels in the forward direction and moving to the negative Y axis means to drive the wheels in the backward direction. The speed of the both wheels can be determined by the how far is the handle from the origin toward negative or positive Y axis. Duty cycles for the left wheel and right wheel is defined as $\mathbf{d_l}$ and $\mathbf{d_r}$; speed for the left wheel and right wheel is defined as $\mathbf{s_l}$ and $\mathbf{s_r}$. For forward and reverse direction $\mathbf{d_l}$, $\mathbf{d_r}$,$\mathbf{s_l}$ and $\mathbf{s_r}$ can be calculated using (4.1),(4.2),(4.3),(4.4).

$$\mathbf{d_l} = sin\theta * \bar{Y} + cos\theta * \bar{X} \tag{4.1}$$

$$\mathbf{d_r} = sin\theta * \bar{Y} + cos\theta * \bar{X} \tag{4.2}$$

$$\mathbf{s_l} = \mathbf{c_l} * \mathbf{d_l} \tag{4.3}$$

$$\mathbf{s_r} = \mathbf{c_r} * \mathbf{d_r} \tag{4.4}$$

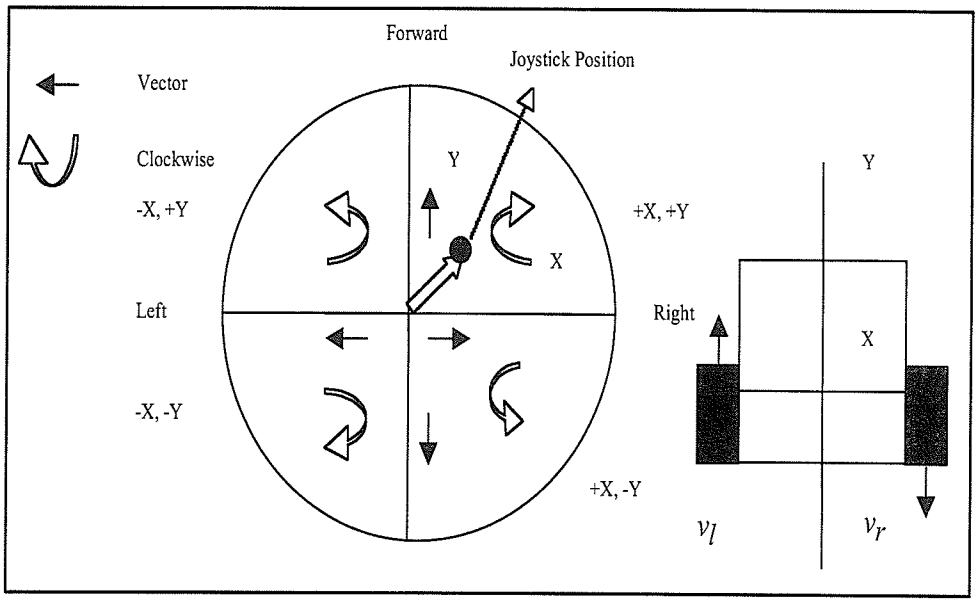

Figure 4.14 Joystick control

Where as $\mathbf{c_l}$ and $\mathbf{c_r}$ are the constants. It can be seen from the above equations that velocities remains same for the both wheels for forward and reverse directions. When the joystick moves along the X-axis, the mobile wheels start to turn to the left or to the right. If it is turning to left direction, the right motor will have a relatively higher speed than the left motor and vice versa for when it is turning to right direction. In order to have smooth turning, a log base 10 algorithm is implied to slow down the respective wheel. For turning left, $\mathbf{s_r}$ will have the same value as (4.4) and $\mathbf{s_l}$ can be calculated from (4.5). For turning right, $\mathbf{s_l}$ will have the same value as (4.3) and $\mathbf{s_r}$can be calculated from (4.6).

$$\mathbf{s_l} = \frac{s_r}{log_{10}(s_r)} \tag{4.5}$$

$$s_r = \frac{s_l}{log_{10}(s_l)} \tag{4.6}$$

It is also possible to have a small or large duty cycle on both wheels to move clockwise or counter clock wise. The variation in spinning depends on the velocities of the wheels running in different directions. For a clockwise spinning, the left wheel will have a forward direction and the right wheel will have a reverse direction and vice versa for counter clockwise spinning. In this way both of the wheels have the same magnitude but run in different directions.

The client module waits for the joystick event and updates the message structure that needs to be sent through its socket connection. This interruption has been implemented in using software as illustrated in Figure. 4.15.

GUI Implementation

The GUI is the user interface to control the robot from remote location. The GUI was written in C #.[13] Microsoft's Visual Studio 2005 was the development platform to develop the C # code. Figure. 4.16 shows the class diagram of the client side of the system. The GUI class is the main class and it starts the connectToServer thread. The GUI uses the eventListener class to detect any joystick event, text or button events within the class and the connectionToServer repeatedly transfers data packets to the server.

4.3 Summary of Chapter 4

Chapter 4 summarized the implementation of the service robot framework. Hardware implementation includes motor control subsystem and sensory feedback subsystem. Software implementation includes collision avoidance subsystem, robot server module, client PC module, and GUI and joystick control.

[13]This portion of the work was done with collaboration with V. Reddy

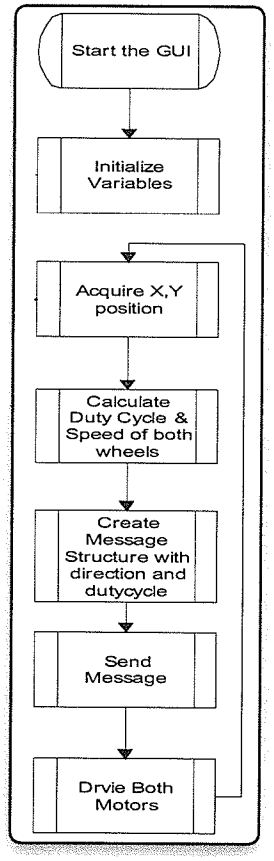

Figure 4.15 Client flow chart

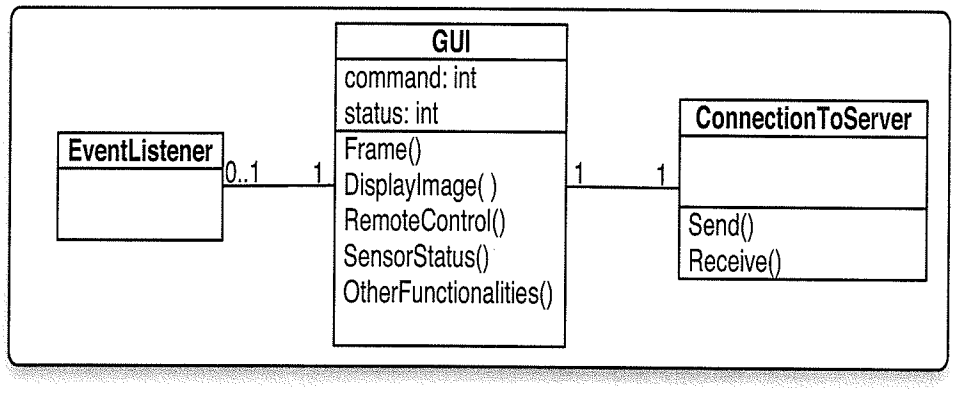

Figure 4.16 GUI class

Chapter 5

Results and Discussions

This chapter presents the results of the implementation described in Chapter 4. The performance of the communication technology, control and visual feedback system are essential when comes to evaluation of the telecontrolled platform. Other performance parameters such as how robust the system is in terms of hardware and software design, and extendibility and reusability of the framework and other design features that the framework provides are discussed in context.

5.1 WLAN Performance

The control of the service robot has been successfully carried out in a wireless networked environment. Our IP centric solution for teleoperation application allows us to monitor and control the maneuverability of the service robot in near real time. However, Internet specific issues like restricted bandwidth and transmission delays are the major challenges of Internet-based robotic teleoperation. The constraint was always in our mind while designing the service framework. An IEEE standard 802.11g wireless network adapter was used as described in Section 3.2.2 to provide the com-

munication between desktop client with the robot server. This includes a transceiver implementing a carrier radio frequency of 2.4 GHz. The maximum physical layer throughput of the system is 54 Mbits/sec unidirectional and the communication work range is 100 meters as described in Table 2.1.

The control information is composed of TCP packets that flow from the client to the server carrying navigation commands for the robot driving subsystem (See Figure. 1.5). When the user requests a command through a joystick device, the desktop client sends the messages to the server. When the server receives the command it processes the command and the robot is instructed to execute that command instantly. Delay time introduced by remote operator and the WLAN technology is a major performance factor for this type of application. In this case larger than expected delay may produce unexpected results at the server end. The measured average round trip time for these short command messages to be sent from the client, to a mobile robot and then echoed back to the client, is 10 ms. It takes 5 ms for the robot server to execute any given command sent by the client. This time delay lies within a reasonable range but may degrade if we had to perform the test where the client and server would be residing on different Local Area Networks (LANs).

5.2 Basic Robot Operation

The initial goal was to build a robot where an operator has full and continuous control over the robot's maneuverability. A user interface was designed where an operator can provide the direction and speed of the robot using a joystick over the 802.11 WLAN. The joystick interface provides precise control of the robot in forward and reverse direction. However, better control of the robot in terms of moving the robot to the left and right direction may have been achieved using a steering system with

accelerator pedal attach to it. However, we decided to use joystick which is easy to use, inexpensive to buy and provides a sufficient number of buttons and triggers to be used to provide all the basic robot operations (See Table 4.1). In addition to that, an application layer command protocol have been designed (see Figure. 4.7) which also allows the designer to include scripting software routine instead of any hardware input devices to maneuver the robot in an automatic fashion. We were successfully able to remotely control the robot using a joystick with visual feedback to rotate the robot's wheels with their desired speed and direction. The type of robot chassis we had to work with was also not amenable to control with a conventional steering mechanism.

5.3 Server Robot

The robot server was designed to provide all the necessary communication between the client PC and the robot itself. The robot server implementation was successfully completed to provide all the necessary functions to operate the robot over the 802.11 WLAN technology. Initially the robot chassis body was lifted up on a stand so that the two controlling wheels at the rear of the chassis can rotate freely. These functions enable basic maneuverability of the robot and functions for reading from sensors.

The tested functionalities are summarized and described in Table 5.1. All the functionalities work as expected. The experimentation has been done in a lab environment where a user uses joystick interface to send commands to robot server program using 802.11 WLAN technology and the robot successfully manages to perform all the operations described in Table 5.1.

Table 5.1 Basic Functions in Robot Server

Functions	Description
Drive Forward	moves motors in a forward direction
Drive Reverse	moves motors in a forward direction
Rotate Left	moves motors in a left direction
Rotate Right	moves motors in a right direction
Stop	stops but does not lock the motor
Emergency Stop	stops and brakes the motor
Drive clockwise spinning	moves right motor reverse and left motor forward
Drive anticlockwise spinning	moves left motor reverse and right motor forward
readSonarValue1	Read from sonar sensor 1
readSonarValue2	Read from sonar sensor 2
readIrValue1	Read from infrared sensor 1
readIrValue2	Read from infrared sensor 2

5.4 The Motor Driver Circuit

The motor driver circuit was assembled successfully, and it met the entire functional requirements as described in Section 1.7. The circuit was designed and implemented in such a way that it receives all the necessary voltages from the car batteries and logic inputs from the FPGA controller. The circuit was tested in an environment where it can support up to 75 Amps of current without getting the MOSFETs too hot. But in theory it can support up to 75 Amps X 4 = 300 Amps for a shorter period of time. The available current rating is more than enough for driving our robot chassis with few hundred extra pounds or driving in an outdoor environment

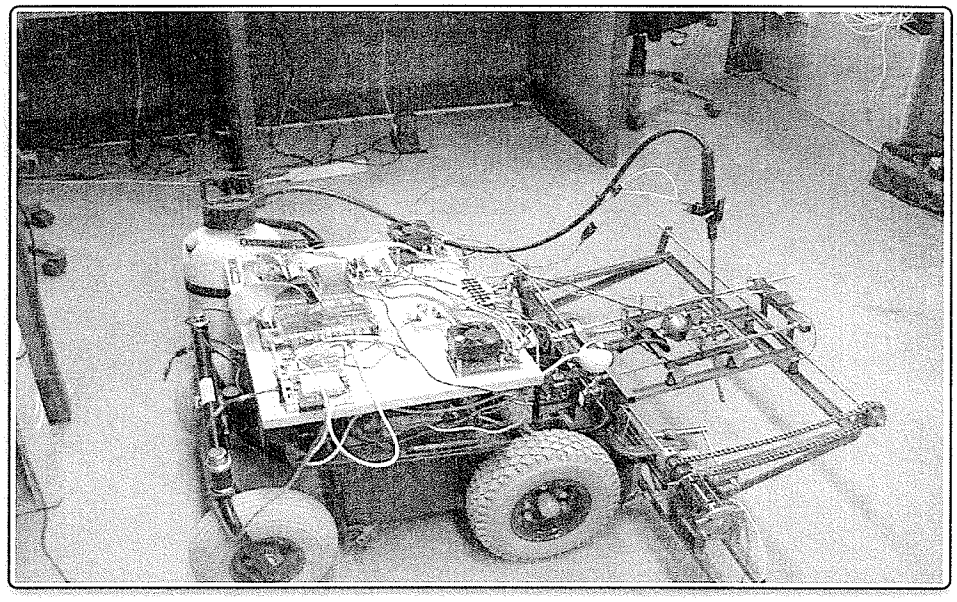

Figure 5.1 The completed robot

(e.g. lawn). The completed robot is shown in Figure. 5.1.

5.5 Collision Avoidance Subsystem

A collision avoidance subsystem was designed based on a fuzzy logic model and utilizes a sensory feedback subsystem to assist the remote user in addition to the supervisory control and visual feedback subsystem. The robot is augmented with ultrasonic sensors and infrared sensors which comprises the sensory feedback subsystem to detect objects that may be on the path. The main idea here is to provide a low level data processing on the mobile robot server. This way the mobile robot can still be able to avoid obstacle under worst case situations such as transmission delay in video feedback or navigation commands. The collision avoidance subsystem is an important design feature for inclusion in the framework as it is important that the personal service

robot built here would have some degree of local intelligence. The sensory feedback subsystem have been designed and implemented successfully to provide distance data in centimeters to the collision avoidance subsystem. However, the fuzzy logic model described in Section 4.2.2 and Section 2.4 has not been implemented completely at the present period of time. However, the design and partial implementation of the fuzzy logic controller for the collision avoidance subsystem is complete.

5.6 Robust Hardware Design

PWM and other control logic signals are implemented in hardware because the output pins connected to the motor controller must be updated in a near continuous manner. For the PWM case, the signal goes high from the starting of each period for proper time and then goes low for the remainder of the period. This proper time duty cycle time is corresponds to the value received by the FPGA controller and is user defined. The PWM logic block holds the register for this new duty cycle value and changes the on-to-off ratio of the signal instantly using a counter. It is also easier to implement the counter in hardware which counts the system clock cycles as opposed to an alternative software approach. All the other control logic pins connected to the motor controller were tested using an Oscilloscope and the signals were updated instantaneously by changing the joystick value at the client desktop.

5.7 Remote Access Flexibility

Our robot consists of two Soekris single board computers where each of them equipped with a WLAN access points. These access points provides bridge between wired and wireless networks as illustrated in Figure. 3.11. This means that mobile robot

equipped with WLAN interface and Linux operating system installed on it can be accessed via SSH (Secure Shell) or telnet from any desktop computers or laptops that are connected to the same access points. These access points are limited to provide a small range of IP address using DHCP (Dynamic Host Configuration Protocol), which requires a login ID and password to securely log-in to the robot's onboard Linux operating system. This allows the user to direct access to each of the Soekris boards, thus provides the flexibility to change configuration of the Camera or the GPS module or to restart the frame grabbing application in case of a video transmission failure due to a hardware problem or network delay.

5.8 Video Teleoperation

Web cam server has been setup in Soekris net5501 embedded single board computer that support our camera's connectivity requirements, i.e. USB port. The Soekris board is running a Debian Linux operating system, and Apache Web server software, Web camera drivers, and frame grabber utilities were installed properly. The frame grabber took a snapshot of the robot's view and the Web server encoded the saved image as part of the HTML display. The image was a JPEG picture. This way the operator would be able to use any of the commercially available Web browsers to download the video streams. The Webcam frame grabber ran continuously. The HTML based Web page continuously re-reads the file and displays the video at a rate of about 7 frames per second. This is slightly lower than expected and we anticipate this to improve once the video stream is directly imported into our GUI using VideoLAN project as described in Section 4.2.2.

5.9 CMOS camera design

The CMOS sensor (camera) used in this project has enabled us to develop a high resolution and high speed video camera which is programmable through a two-wire serial interface that offers digital output and consumes very little power. The high speed data provided by the sensor output can be a difficult design constraint for processing and storing and for this reason high speed data acquisition was required. The FPGA provided by the Altera's DE2 board was ideally suited for this application. The FPGA provides necessary control signal and quickly performs repetitive tasks (for example, Neighborhood Replication algorithm as described in Section 4.1.3) on each pixel using a hardware pipeline. This allows images to be processed from the sensor with resolution (640 X 480) at 60 images per second. Figure. 5.2 shows the image display on the monitor.

5.10 Graphical User Interface

A simple graphical user interface is shown in Figure. 5.3 to provide information on visual feedback, sensory feedback, control and status information about the mobile robot. The GUI is divided into four panels. The top panel allows users to *connect* to the mobile robot server using its IP address and socket port. The robot server must be in the *listening* state in order for the connect process to complete successfully. Once the application has established connection the application detects any joystick, text or button events and sends the navigation commands to the server. The middle panel shows the control panel and it consists of four directions buttons. The bottom right panel also allows the remote user to send text base message by click on send button. This way the GUI allows user to send commands in three different ways. If a command is sent to server requires a response back, the server sends a status of

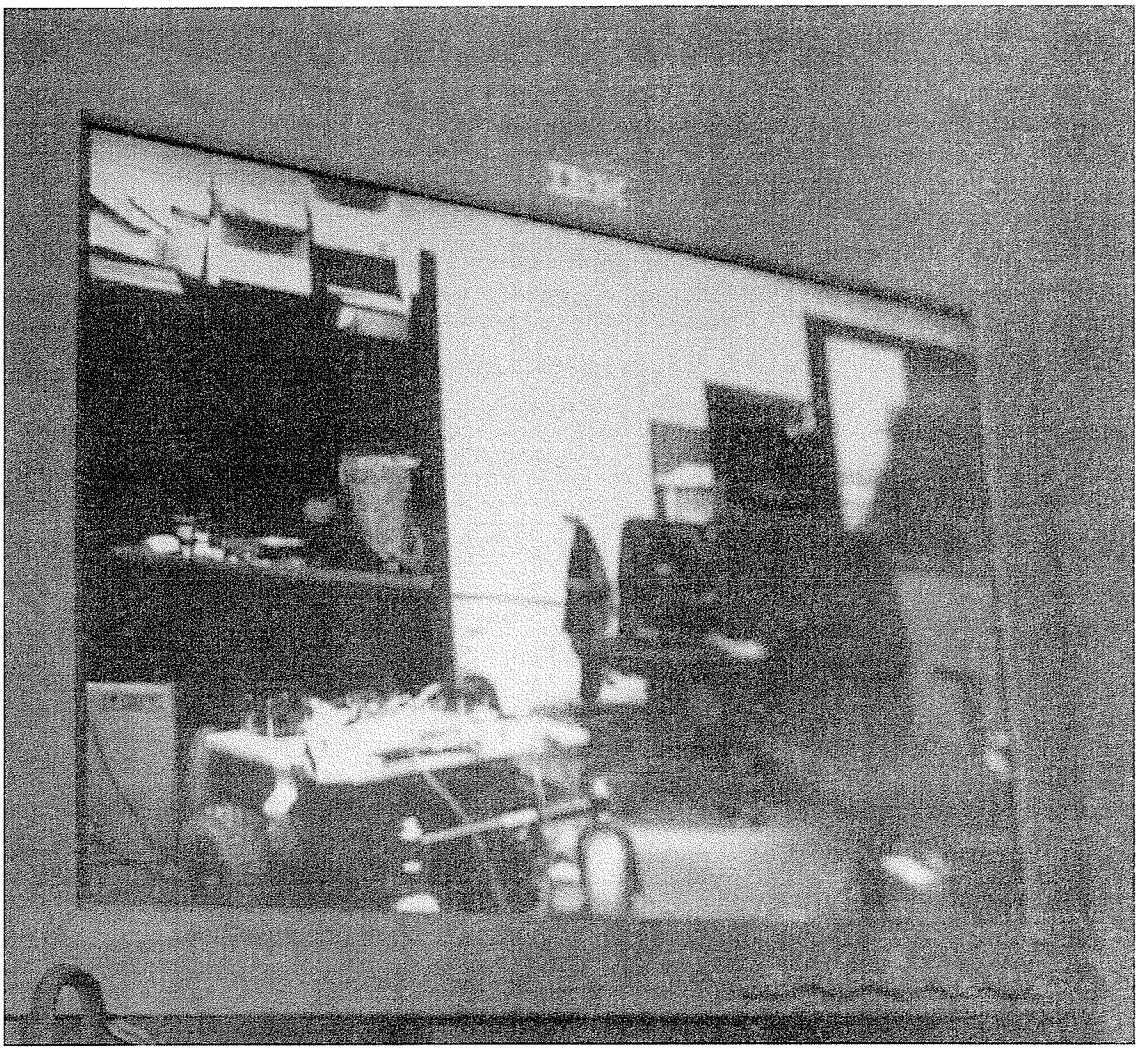

Figure 5.2 VGA display

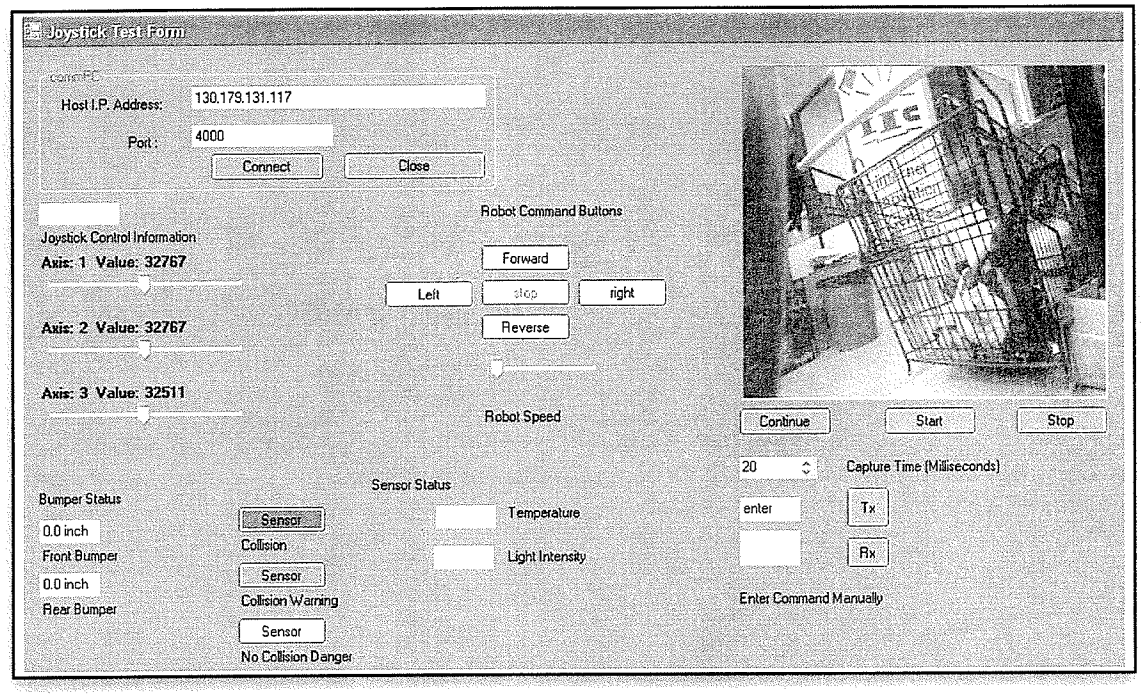

Figure 5.3 GUI

respective sensor in return. The right panel shows the visual feedback in a continuous jpeg image with 640x480 pixels resolution. At this time there is some difficulty in generating the videos in the GUI and as a backup the video is streamed to a browser which is also running on the client.

5.11 Summary of Chapter 5

This chapter described the results from the personal service robot implementation. The framework presented here is extendable and reusable and satisfied the design constraints required for robotic platform to provide service tasks. This includes: basic robotic maneuver operations, efficient motor control and a joystick control algorithm,

flexible client-server architecture, high speed image acquisition system, sufficient image and command transfer rate, sensory feedback subsystem, user friendly graphical user interface, remote access flexibility, robust hardware and power electronics design and standard WLAN communication techniques.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

For this thesis, a service robot framework was designed and implemented with the conveniences of being controlled remotely over 802.11 WLAN. The overall framework is a semi-autonomous system. The main challenge was to build a working system that integrated many different components. This framework was built with the objective of providing service tasks for a weed eating application. The framework can be reused and extended to many environments with a remote operator engaged in various tasks. The framework provides services by incorporating a collision avoidance subsystem, a vision subsystem, a motor control subsystem and is well tailored to provide semi-autonomous operations such as the ones discussed here. This framework also provides flexible client-server architecture for navigation command transfer and video stream transfer in near real time.

By using a hardware/software strategy we were able to develop a more extendable robot control system with improved processing facilities to provide a more robust solution than previous robots developed here [22].

6.2 Future Work

The design of a service robot framework involves wide-range of technology from different disciplines. Thus, there are various design features that can be integrated to the framework for additional improvements to the project. Some of these may include:

- The use of a hardware video processing module for the vision subsystem.
- Extending the framework to IEEE 802.16 WiMAX for providing a long range communication channel.
- Improve the sensory feedback subsystem by incorporating Hall, strain gauges, gyro and 3D compass for feedback purposes.
- Study various adaptive learning algorithms for better control of the user input.

6.3 Limitation of the Framework

As with any physically realizable project the framework developed here could benefit from more complex and sophisticated hardware. Specifically, commercially available IP cores and a more powerful FPGA would benefit the framework. In addition more sophisticated sensors such as laser range finders would also have been useful. However, the main design challenges and the framework designed and implemented here represent a very reasonable approach to tackling technologies and algorithms that concern the development of personal service robots.

In our case we are highly constrained by what is available and have made near optimal use of these resources in developing a framework that is conceptually optimal.

Bibliography

[1] (2007) Stepper motor manual. [Online]. Available: http://www.pacsci.com/support/documents/stepper/pacscistepperinstallbulletin.pdf

[2] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1997.

[3] B. Astrand and A. J. Baerveldt, "An agricultural mobile robot with vision-based perception for mechanical weed control," in *Autonomous Robots*, vol. 13, Netherlands, 2002, pp. 21–35.

[4] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley, "Desktop tele-operation via the World Wide Web," in *Proc. IEEE International Conference on Robotics and Automation*, 1995, pp. 654–659.

[5] The Telelabs project website. [Online]. Available: http://telerobot.mech.uwa.edu.au/

[6] The Telegarden project website. [Online]. Available: http://goldberg.berkeley.edu/garden/Ars/

[7] K. Brady and T. Tarn, "Internet-based remote teleoperation," in *Proc. IEEE International Conference on Robotics and Automation*, 1998, pp. 65–70.

[8] P. Saucy and F. Mondata, "Khepontheweb: Open access to a mobile robot on the Internet," IEEE Robot. Automat. Mag., 2000.

[9] W. Burgard, A. B. Cremers, D. Fox, D. Hhnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The interactive museum tour-guide robot," in *Proc. of the Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, 1998, outstanding paper award.

[10] The Xavier project website. [Online]. Available: http://www.cs.cmu.edu/People/Xavier/

[11] S. Thrun, "Towards a framework for human-robot interaction," *Human-Computer Interaction*, vol. 19, no. 1,2, pp. 9–24, 2004.

[12] W. K. Fung, Y. Y. Leung, M. K. Chow, Y. H. Liu, Y. Xu, W. Chan, T. W. Law, S. K. Tso, and C. Y. Wang, "Development of a hospital service robot for transporting task," in *Proc. IEEE International Conference on Rohotics, Intelligent Systems and Signal Processing*, vol. 1, Changsh, China, Oct. 2003, pp. 628–633.

[13] M. Y. Shieh, Y. H. Chan, Z. X. Lin, and J. Li, "Design and implementation of a vision-based shopping assistant robot," in *Proc. IEEE International Conference on System, Man, and Cybernetics*, Taipei, Taiwan, Oct. 2006, pp. 4493–4498.

[14] S. K. Pradhan, D. R. Parhi, and A. K. Panda, "Navigation of multiple mobile robots using rule-based-neuro-fuzzy technique," *International Journal OF Computational Intelligence.*, vol. 3, no. 2, pp. 142–152, 2006.

[15] S. Thongchai and K. Kawamura, "Application of fuzzy control to a sonar-based obstacle avoidance mobile robot," in *Proc. IEEE International Conference on Control Applications*, Alaska, USA, Sept. 2000, pp. 425–430.

[16] R. Malhotra and A. Sarkar, "Development of a fuzzy logic based mobile robot for dynamic obstacle avoidance and goal acquisition in an unstructured environment," in *Proc. IEEE International Conference on Advanced Intelligent Mechatronics*, California, USA, July 2005, pp. 1198–1203.

[17] W. Wei, Y. Pan, and K. Furuta, "Internet-based tele-control system for wheeled mobile robot," in *Proc. IEEE International Conference on Mechatronics and Automation*, vol. 3, Niagra Falls, Canada, July 2005, pp. 1151–1156.

[18] H. Hu, L. Yu, P. Tsui, and Q. Zhou, "Internet-based robotic system for teleoperation," in *International Journal of Assembly and Automation*, vol. 21, no. 2, 2001, pp. 143–151.

[19] G. Bright, J. Potgieter, S. Tlale, and O. Deigel, "Internet control of a domestic robots using Wireless LAN," in *Australian Conference on Robotics andAutomation (ACRA'02)*, Auckland, New Zealand, Nov. 2002, pp. 212–215.

[20] Altera. (2007, Mar.) Altera's development and education board. [Online]. Available: http://www.altera.com/education/univ/materials/boards/unv-de2-board.html

[21] (2007) The OSMC project website. [Online]. Available: http://www.robotpower.com/osmc_info

[22] M. R. Alam, "A telecontrolled robotic platform," Master's thesis, University of Manitoba, Manitoba, 2006.

[23] A. A. S. Al-Jumaily, S. H. M. Amin, and M. Khalil, "A fuzzy multi-behaviour reactive obstacle avoidance navigation fora climbing mobile robot," in *Proc. IEEE International Conference on Intelligent Engineering Systems*, Budapest, Hungary, Sept. 1997, pp. 147–152.

[24] (2007) The OSMC motor driver manual. [Online]. Available: http://www.robotpower.com/downloads/OSMC_project_documentation_V4_25.pdf

[25] (2007) Stepper motor basics. [Online]. Available: http://www.solarbotics.net/library/pdflib/pdf/motorbas.pdf

[26] The infrared sensor datasheet. [Online]. Available: http://www.acroname.com/robotics/parts/R146-GP2D120.html

[27] The SRF04 sonar range finder datasheet. [Online]. Available: http://www.acroname.com/robotics/parts/R93-SRF04p.pdf

[28] Wikipedia. (2007, Oct.) Hardware description language. [Online]. Available: http://en.wikipedia.org/wiki/Hardware_description_language

[29] (2007) TRDB DC2 1.3 Mega Pixel digital camera module for DE2. [Online]. Available: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=39&No=50

[30] (2004) The Soekris 4826 website. [Online]. Available: http://www.soekris.com/net4826.htm

[31] "CM-9 802.11 data sheet," Metrix Communication, Seattle, USA.

[32] (2007) The Soekris 5501 website. [Online]. Available: http://www.soekris.com/net5501.htm

[33] (2007) The developing components for SOPC builder website. [Online]. Available: http://www.altera.com/literature/hb/qts/qts_qii54007.pdf

[34] (2007) The bayer color format. [Online]. Available: http://www.answers.com/demosaicing

[35] Wikipedia. (2007, Oct.) Nearest neighbor interpolation. [Online]. Available: http://en.wikipedia.org/wiki/Nearest_neighbor_interpolation

[36] (2007) TRDB DC2 user manual. [Online]. Available: http://www.terasic.com.tw/attachment/archive/50/TRDB_DC2_UserGuide_061017.pdf

[37] (2007) Website for debian linux. [Online]. Available: http://www.debian.org/

[38] (2007) Website for video LAN project. [Online]. Available: http://www.videolan.org/

[39] (2007) How to video LAN project. [Online]. Available: http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/VideoLAN-HOWTO.html#AEN46

[40] (2007) Website for Pyramid Linux project. [Online]. Available: http://pyramid.metrix.net/trac/

[41] J. P. Hong, H. M. Shim, S. B. Jung, E. H. Lee, and S. H. Hong, "A steering algorithm of the MCU based controller for two-wheeldrive vehicles," in *Proc. IEEE International Symposium on Industrial Electronics (ISIE'01)*, vol. 3, Pusan, South Korea, June 2001, pp. 1887–1890.

Index