DESIGN AND IMPLEMENTATION OF A GSM-GPRS CONTROLLED ROBOT

by

FREDDY DESTIN MAKAYA ONDENGUE

Submitted in partial fulfilment of the requirements for the degree

MAGISTER TECHNOLOGIAE: TELECOMMUNICATIONS

in the

Department of Electronics

FACULTY OF ENGINEERING

TSHWANE UNIVERSITY OF TECHNOLOGY

Supervisor: Mr. D. Chatelain

Co-Supervisor. Prof. B. J. Van Wyk

External-Supervisor: Mr. H. Goodhead

June 2005

SPONSORS

MTN
Hello the Future

F'SATIE
French South African Technical Institute in Electronics

Tshwane University of Technology
We empower people

DECLARATION AND COPYRIGHT

I hereby declare that the dissertation submitted for the M-Tech: Telecommunication, at Tshwane University of Technology, is my own original work and has not previously been submitted to any other institution. All authors quoted are indicated and acknowledged by means of a comprehensive list of references.

F. D. Makaya Ondengue

To my dear mother

Veronique Ngoumba…

ABSTRACT

The design and construction of a “GSM Technology Demonstrator” robot to be used as an educational and advertising tool by one local GSM network provider namely MTN, is discussed. The robot will be used to introduce the public to new services that are not often used, with the purpose of boosting the demand and popularity of such services. The “GSM Technology Demonstrator” robot is remotely controlled by a mobile phone using GPRS and is able to receive and reply to SMS and MMS messages.

On one hand, the robot will work toward the Public Understanding of Science, Engineering and Technology (PUSET). Public awareness and understanding of the impact of science and technology in people’s everyday lives are very important both for people and industries involved in the development of new technologies. On the other hand, it will promote the services and boost the consumption of those services not used by the general public.

The “Demonstrator” is an autonomous and remote controllable robot, in which most of the latest telecommunication technologies such as GPRS (General Packet Radio Service), MMS (Multimedia Messaging Services), voice recognition and phone web browsing, is implemented. The robot will be used in a science centre to demonstrate or show to the public the importance of GSM services offered by the operator and how to use them.

The design and construction of this telecommunication robot required a lot of expertise in many different fields such as Mechanics, Electronics, Telecommunication, and Software development. This document covers every aspect of the project, from the conceptual ideas to the detailed design of each main part of the system.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and appreciation to: My three supervisors, Mr Damien Chatelain (F'SATIE), Mr Hilton Goodhead (MTN) and Prof. Ben Van Wyk (F'SATIE), for their positive attitude and guidance toward the successful completion of this project.

My gratitude also goes to Prof M. A. Van Wyk, Mr Pierre Abeille (Director of F'SATIE) and to all the staff of F'SATIE and MTN, particularly to Mr G. Noel (F'SATIE), Mr J. De Vries (F'SATIE), Ms S. Gama (MTN), Mr N. Naidoo (MTN) for their assistance and support during the course of this project.

My many thanks to my colleague Mr C. Van Wyk, and the staff of Tshwane University of Technology who assisted me when I needed their help. Particularly Mr B. Mutton, and Mr H. Weinert (Mechatronics Labs).

A lot of thanks to my family and particularly to my Mother for her patience, support and encouragements.

Finally my thanks to the Almighty for everything He made possible…

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY OF TERMS AND ABBREVIATIONS

Abbreviations

ADC	Analog-to-Digital conversion
APN:	Access Point Node
ARIB:	Association for Radio Industry and Business
CEPT:	Conference of European Posts and Telecommunications
DAB:	Digital Audio Broadcast
EDGE:	Enhanced Data rate for GSM Evolution
ETSI:	European Telecommunications Standards Institute
GMSK:	Gaussian Modulation Shift Keying
GPRS:	General Packet Radio Services
GSM:	Global System for Mobile Communication
HSCSD:	High Speech Circuit Switched Data
IMT:	International Mobile Telecommunications
IP:	Internet Protocol
ISP:	In-System Programming
ISDN:	Integrated Services Digital Network
ITU:	International Telecommunications Union
MMS:	Multimedia Messaging Service
MSISDN:	Mobile Station ISDN number
M2M:	Man-to-Machine or Machine-to-Machine
MGM1:	MTN GSM Mascot version 1
MTN:	Mobile Telephone Network
PDP:	Packet Data Protocol
PC:	Personal Computer
PCB:	Printed Circuit Board
PLMN:	Public Land Mobile Network
PUSET:	Public Understanding of Science, Engineering and Technology

SIM:	Subscriber Identity Module
SMS:	Short Message Services
UMTS:	Universal Mobile Telecommunications System
USART:	Universal Synchronous Asynchronous Receiver Transmitter
USSD:	Unstructured Supplementary Service Data
VAS:	Value Added Services
WCDMA:	Wideband Code Division Multiple Access
WAP:	Wireless Application Protocol
8PSK:	8 Phase Shift Keying

Terms

***Advertisement*:** In Collier's Encyclopedia, advertising is defined as "mass, paid communication by means of the printed word, radio, or television, aimed at persuading individuals to take a desired course of action." It says that advertising is also "used to encourage the favorable acceptance of an idea, an institution or a person (Bahr & Johnston, 1992:136).

In this study, advertisement will refer to a technology demonstration whereby MGM1 will show the importance and ease of use of some GSM services to the public with the aim of boosting the use of those services.

***Demonstrator*:** Collins Larousse (1990:250) defines a demonstrator as a person, usually in a shop, who shows people how a machine or device works by operating it themselves and explaining what they are doing.

In this document, the robot (MGM1) is used as a demonstrator to exhibit the use of services such as SMS, MMS, GPRS, HSCSD, voicemail recovery and web surfing using a phone browser.

Distributed microcontroller architecture: This is a microcontroller architecture whereby many microcontrollers on different boards are used to control the whole system instead of a single microcontroller on a single board.

Exhibition: Exhibition is defined in Collins Larousse (1990:331) as the showing of pictures, sculptures, or other things displayed in a public place.

In scientific exhibitions, most of the times new technological developments are displayed, products and artifacts are displayed. MGM1 will be used in that type of event.

Mascot: A mascot is a person, an animal or an object that is believed to bring good luck (Procter, 1996:869).

Modem: In this document modem refers to the device that interfaces the robot to the GSM network allowing the transmission of data via the radio interface from the robot to a mobile phone and vice versa.

Performance: The performance of a person or machine is how well they do a piece of work or activity (Procter, 1996:1048).

Remote Control: A remote control is a system or a device for controlling something such as a machine or vehicle from a distance, by using electrical or radio signals (Procter, 1996:1202).

Robotics: Robotics can be defined as a scientific discipline associated with the design, development, and application of robots (Webster, 1999:202).

CHAPTER ONE

PROJECT OVERVIEW

"A better view is a step ahead and an overview is light in the mind"

1.1 INTRODUCTION

The object of this research is to develop a wheeled robot to be used in the science center of a mobile network service provider as an educational and advertising tool. In the science center, the robot will play the role of a "GSM Technology Demonstrator", showing the public the importance and how to use the wireless technologies and services implemented therein.

On the one hand, the robot will work toward the Public Understanding of Science, Engineering and Technology (PUSET). Awareness and understanding of the impact of science and technology in people's lives is very important both for the people and industries involved in development of new technologies. On the other hand, the robot will promote the services offered by the GSM network service provider and boost the consumption of those services currently not used by the general public.

The "GSM technology demonstrator" project has mainly been initiated to address the challenge of making known certain technologies and services offered by MTN, one of the three GSM network service providers in this country. The project entails the design and construction of a remotely controlled mobile robot, in which most of the latest telecommunication technologies such as GPRS (General Packet Radio Service), MMS (Multimedia Messaging Services), voice recognition and phone web browsing, is implemented.

The robot has an incorporated web-camera for taking pictures and an LCD screen for the display of pictures, image sequences, multimedia presentations or any other information.

This project, initiated by MTN (Mobile Telephone Networks), was inspired by the Star Wars spunky and adventurous astromech droid R2-D2. The aim is to use the GSM technology demonstrator in the MTN science center in Cape Town to educate children and adults about telecommunication technologies. The final prototype developed in this project has been called MGM1 which stands for MTN GSM Mascot version 1. Figure 1.1 illustrates the background of this project.

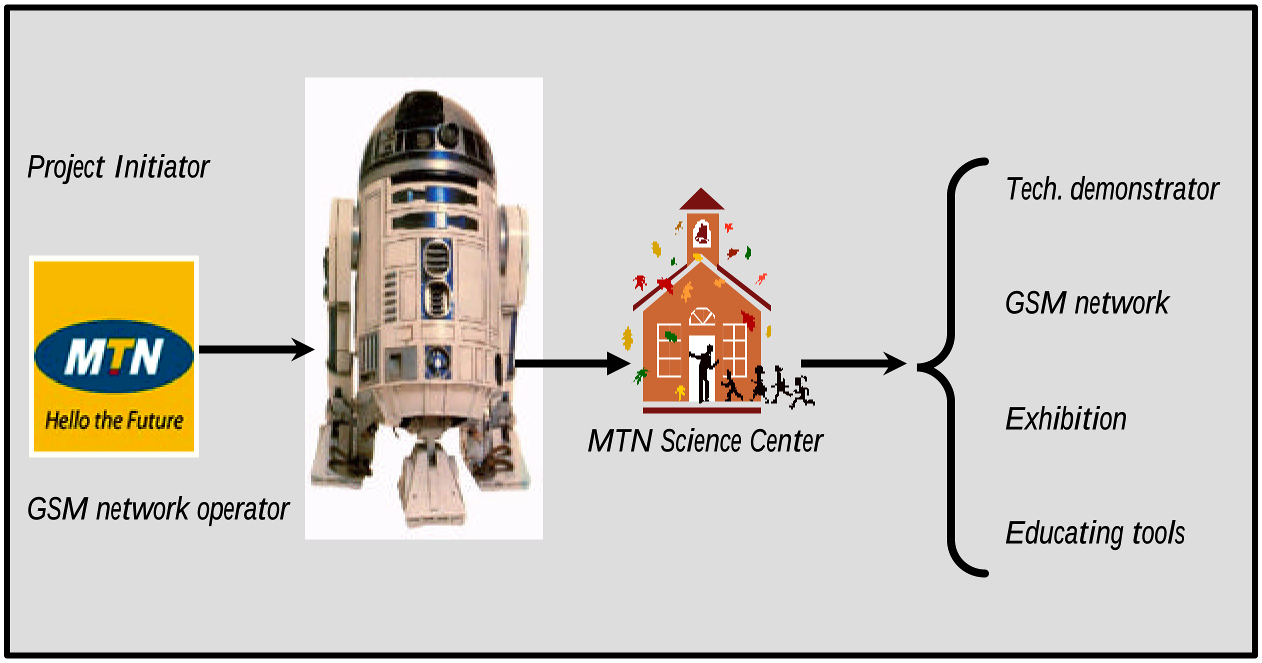


FIGURE 1.1 PROJECT BACKGROUND

1.2 PROBLEM BACKGROUND

The Global System for Mobile Communication known as GSM, is a technology that dominated mobile communication industry over the past decade. GSM originated from Europe and spread into many other countries to become a popular standard for mobile communication. GSM started mainly with voice communication, offering few basic services, then data communication was introduced, but due to slow data rate, it was useful to very few applications. Research and development brought about the different evolutions of GSM, namely HSCSD (High Speed Circuit Switched Data), GPRS (General Packet Radio Service) and EDGE (Enhanced Data rate for GSM Evolution).

Presently, GSM network operators provide a wide range of services, including many new services such as web surfing, MMS (Multimedia Messaging Services), and EMS (Enhanced Messaging Services). There are a lot of useful technological products developed in the world to make life much easier, products which are not used by many simply because these are not well known or not properly understood.

Public understanding of science and technology is a very important aspect of our modern world. Scientific discoveries and new technologies serve the purpose of pushing back the frontiers of human ignorance and make life and the actual way humans do things much simpler. It is therefore of little use keeping them in laboratories and libraries alone. Information on scientific discoveries and new technologies should be disseminated to the general public so that people can be aware of them and make informed choices.

Many companies invest significant capital in the development of new technologies, new products or new services, which they have to commercialize at the end. It is a serious problem for these companies when the public is not aware of the existence of these new products on the market, or when the public is not ready to use the new technology. Therefore educating people about new technologies, their importance and their use is extremely important for such companies. The quicker people can be educated about new technologies, the better for themselves, for the companies developing these technologies, for the government and society in general.

In many industries, the trend is to automate all systems and make use of machines or robots in most of the critical sections of the production chain, for tedious work in order to reduce errors. Since robotics is quite a fascinating and very attractive technology, it is a good tool to use to educate people about the use and importance of a technology through demonstrations.

1.3 THE PROBLEM AND ITS SETTING

1.3.1 PROJECT SPECIFICATIONS

It was specified that the robot should:

- Be mobile
- Be remotely controlled via the GSM network
- Send and receive SMS messages
- Recover personal voicemail
- Incorporate a web cam
- Incorporate a LCD screen
- Take pictures, display and send them as MMS messages
- Perform voice recognition

1.3.2 PROBLEM STATEMENT

The problem of this research is to design and build a wireless (GSM-GPRS) remotely controlled robot, capable of demonstrating in an understandable way certain services offered in a GSM network.

1.3.3 THE SUBPROBLEMS

1.3.3.1 Subproblem 1

The first subproblem is to establish (produce) a conceptual design based on the original idea and then develop a detailed design of each of the different constituents of the system.

1.3.3.2 Subproblem 2

The second subproblem is to develop and build all the components of the robot and finally integrate them so as to produce the final prototype.

1.3.3.3 Subproblem 3

The third subproblem is to test the operation of the built prototype and assess its performance and limitations.

1.4 HYPOTHESES

1.4.1 HYPOTHESIS 1

The first hypothesis is that the implementation of a fast and cost effective remote control using GPRS can be achieved by connecting the robot to a GSM network.

1.4.2 HYPOTHESIS 2

The second hypothesis is that all developed units, mainly electronics, software, and mechanical will be integrated such that a coherent operation may result.

1.4.3 HYPOTHESIS 3

The third hypothesis is that the final prototype will be robust and able to execute commands without ambiguity and perform the requested task immediately after reception of instruction.

1.5 DELIMITATIONS

This work was limited to the design and the construction of the robot. It mainly focuses on the following aspects:

- Mobility
- GPRS remote control
- Sending and receiving SMS and MMS messages
- Take pictures and send them as MMS messages
- Recover its own voicemail

The study will not consider the following aspects:

- Artificial Intelligence for autonomous operation
- Operation on rugged and inclined terrains
- Construction of the outer shell of the robot
- Control accuracy

1.6 ASSUMPTIONS

1.6.1 ASSUMPTION 1

It is assumed that the mobile GSM network operator will provide a proper and reliable service all the time.

1.6.2 ASSUMPTION 2

It is assumed that the GSM coverage in the area of operation of the robot will be acceptable during operation.

1.6.3 ASSUMPTION 3

It is assumed that the PC to be incorporated inside the robot will not be sensitive to vibrations and shocks.

1.6.4 ASSUMPTION 4

It is assumed that the robot will operate indoors and on a flat floor.

1.6.5 ASSUMPTION 5

It is assumed that the person interacting with the robot has a basic knowledge of the implemented functions and their operation, as well as the valid commands and the understood language.

1.7 RESEARCH OUTPUTS

The work done has been **presented and published in two international (IEEE) conferences** :

- MAKAYA, F. D. 2004. Design and Implementation of a GSM-GPRS Controlled Robot. In: EuroSim 2004, Paris, France.

- MAKAYA, F. D. 2004. Design and Performance assessment of a wireless controlled robot. In: EDMO 12th, 2004, Kruger National Park, South Africa.

1.8 DOCUMENT PLAN

This dissertation is the main documentation of the completed project. It contains technical data of the prototype robot MGM1. The document consists of five different chapters.

1.8.1 CHAPTER ONE

The first chapter is the overview of the whole project. It discusses important topics such as project background as well as the definition of the main problem.

1.8.2 CHAPTER TWO

Chapter two is about the literature review. It consists of the history of robotics as a field of research and discusses the main robot design concepts found in the literature.

1.8.3 CHAPTER THREE

Chapter three is the discussion of the complete design of MGM1 and the design implementation. This encompasses all mechanical, electronic, and software designs. The discussion goes from concept to the circuit or detailed design.

1.8.4 CHAPTER FOUR

The fourth chapter is about the tests done and results obtained as well as the discussion of the results obtained.

1.8.5 CHAPTER FIVE

This is the closing chapter where hypothesis are tested against the obtained results. Ideas about future developments and improvements are stated, suggestions for system performance optimization and accessories related to the physical appearance of the robot are given.

CHAPTER TWO

LITERATURE REVIEW

"A complex system that works is invariably found to have evolved from a simple system that works."
John Gaule.

2.1 INTRODUCTION

The desire of making life better and easier is part of human nature. Many centuries before our era, man was already on his quest for automation, mainly for its beauty and also because it could release man from routine and annoying type of work. Building a system or a machine that could do things without human intervention was and is still very fascinating. The first robotic companies and research groups were formed in the 60's. During this period the first industrial robots appeared on scene in some manufacturing plants and since then, robotics has evolved significantly and today we find robots almost everywhere in our everyday lives (Bedini, 1999; William, 2002).

This chapter is a short literature review of robotics design and GSM technology, with the emphasis on the features implemented or that could be implemented in MGM1.

2.2 GSM

The main focus of this study is to establish a wireless man-to-machine communication using the existing PLMN, and also implementing some of the services offered by the PLMN in order to showcase their usefulness.

Currently, all the PLMN in South Africa and most of the countries in the world are GSM networks, therefore the wireless man-to-machine communication achieved in the research is done through (based on) GSM networks.

The section below is a general overview of the GSM technology, from its origin, the type of services available, the network structure and briefly how it operates.

2.2.1 GSM HISTORY

GSM is a standard for digital mobile telephony developed in Europe to substitute the existing analog mobile telephony technology which by that time, was confronted to a number of problems such as increased demand, capacity, and incompatibility with other networks.

In 1982, the Conference of European Posts and Telecommunications (CEPT) established a study group whose objective was to study and develop a public land mobile system for Europe. The group responsible for this work was called "Groupe Speciale Mobile" (GSM). In 1990, phase I of the GSM specifications was published and in 1991 the commercial service was started. From that time, GSM gained worldwide popularity and GSM which originally was an acronym for "Groupe Speciale mobile", was later set to stand for Global System for Mobile Communication.

The GSM recommendations, do not specify the actual hardware requirements, but instead specify the network functions and interfaces in detail, guaranteeing the proper interworking between the components of the system. This allows hardware designers to use their creativity to provide the actual functionality and at the same time makes it possible for operators to buy equipment from different suppliers (Scourias, 1994; Ericsson, 1998).

2.2.2 GSM SERVICES

GSM offers three categories of services. The first category of services is related to the transportation of data to or from an ISDN terminal. These services are referred to as bearer services. The second category of services is referred to as Tele-services. This category includes services such as telephony and SMS. The third category of services is referred to as supplementary services. This include services such as caller identification, call forwarding, call waiting, multiparty conversations, and barring of outgoing calls (Scourias, 1994).

2.2.2.1 Short Messaging Services

Short Messaging Services, in short SMS, is a GSM Tele-service which allows mobile subscribers to send and receive alphanumeric messages up to 160 characters long, using their mobile phones. When non-latin alphabets are used, the message length is reduced to 70 characters. SMS was part of GSM phase 1 standard. It is implemented by adding a node called Short Message Services Center (SMSC) in a GSM network. All messages from senders are directed to the SMSC before they are forwarded to the recipients, hence SMS is said to be a store and forward service. SMS's big advantage is the fact that the message is transmitted via the signaling channel, leaving the bandwidth free for voice, data and fax calls (Portolani, 2004). MGM1 can receive SMS and reply by sending back an SMS. Remote controlling MGM1 can also be done by sending an SMS to it.

2.2.2.2 Multimedia Messaging Services

Multimedia Messaging Services (MMS) is seen as an evolution of SMS from the fact that it supports not only text but more media types such as picture, audio, and video.

MMS therefore allows mobile subscribers to send and receive multimedia messages using their mobile phones. MMS is also a store and forward service but here, the message is transmitted via a data channel. The implementation of MMS in the GSM network requires an additional node, the Multimedia Message Services Center (MMSC) to handle the store and forward function, also the network has to be GPRS and WAP capable.

2.2.3 NETWORK ARCHITECTURE

The GSM network can be divided into three main subsystems: the Mobile Station Subsystem which is used by the subscriber, the Base Station Subsystem which controls the radio link with the Mobile Station, and the Network Subsystem which mainly performs the switching of calls between a mobile and other fixed or mobile network users.

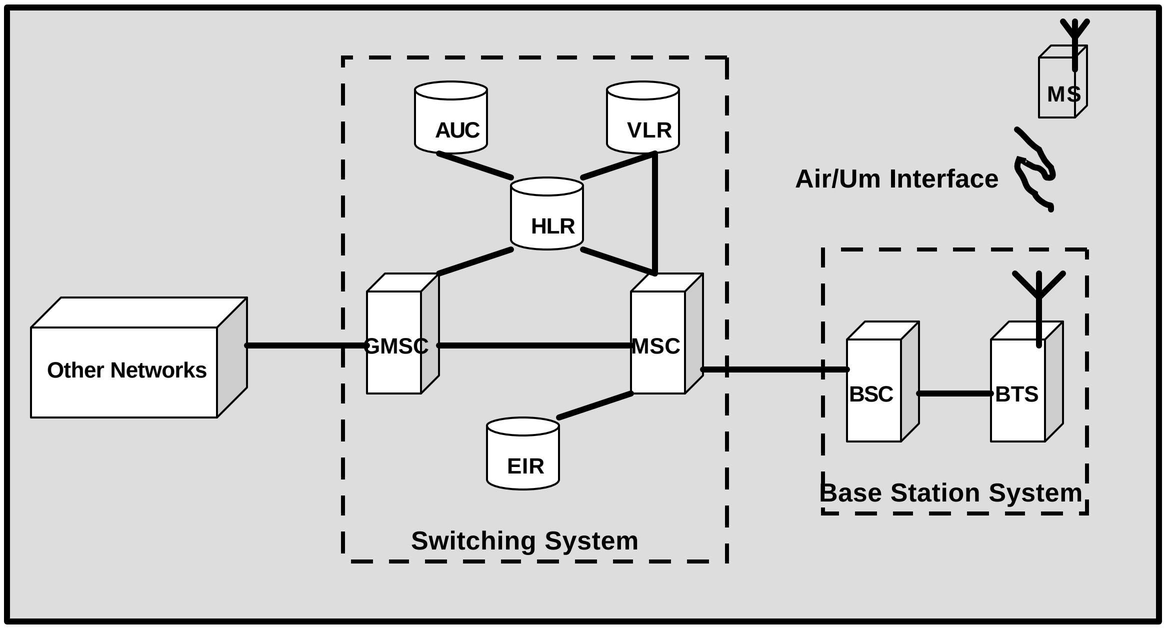


FIGURE 2.1 GSM NETWORK ARCHITECTURE

The Mobile Station subsystem and the Base Station subsystem communicate through the air interface, also known as the air interface or radio link. The Base Station subsystem and the network subsystem communicate through the A interface (Scourias, 1994; Ericsson, 1998).

In MGM1, the remote control function via a mobile phone has been implemented thanks to existing GSM networks. All SMS from a human operator to MGM1 and vice versa, passes through the GSM network. A block diagram of the GSM network is shown in Figure 2.1.

2.2.4 GSM EVOLUTION

ISDN compatibility was one of the GSM specifications, but from the data rate point of view, the radio link imposed to GSM a data rate of 9.6kbps/timeslot, which is much lower compared to the 64kbps/timeslot offered by ISDN. Therefore GSM could only offer low-rate data services.

GSM continued to be developed to improve capacity, quality, coverage, and mainly data rate. Many technologies were then developed to improve GSM data rate, first HSCSD, then GPRS, EDGE and finally WCDMA. The technologies contributing to the evolution of GSM to higher data rate are classified as 2.5 generation or GSM evolution (Scourias, 1994).

2.2.4.1 General Packet Radio Services

In the traditional GSM, data transmission is done using Circuit Switched Data (CSD) techniques whereby the network allocates one radio channel to a MS when data is to be transmitted to the network and the radio channel remain occupied for the connection time.

In GPRS, data is the form of packets and it is transmitted through the air interface only when the need arises. The radio channel does not remain occupied for the data transfer time, but rather can be used by others.

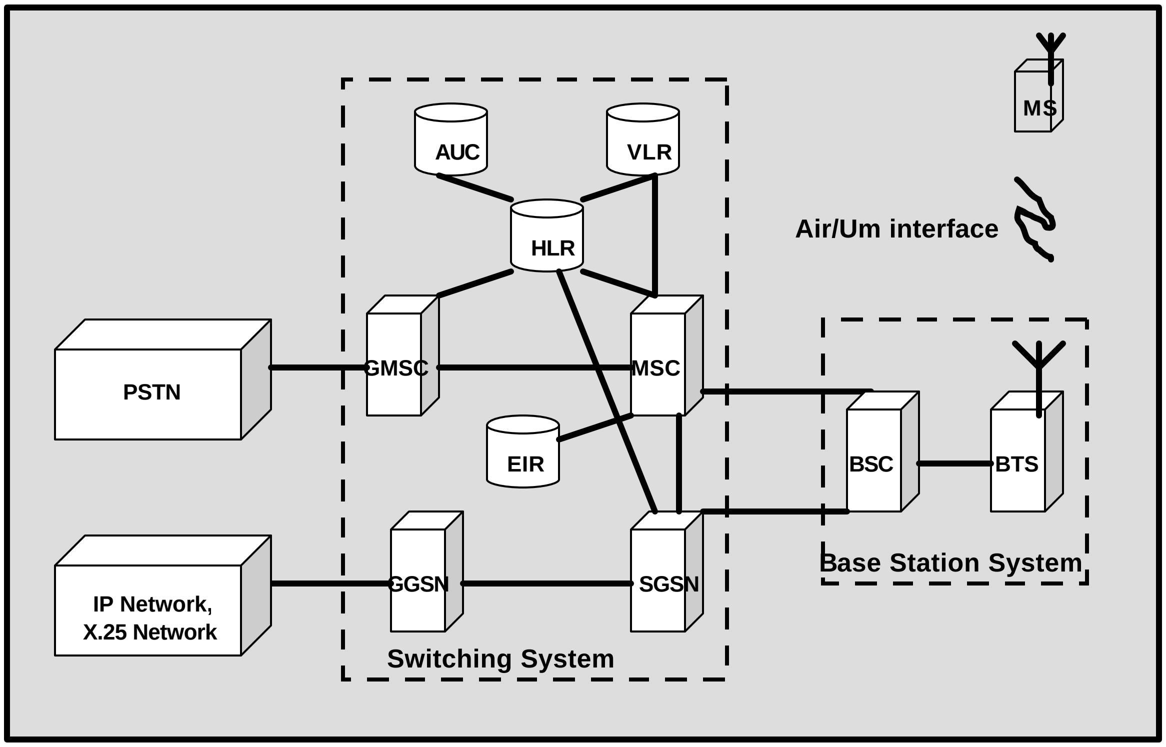


FIGURE 2.2 GSM-GPRS NETWORK ARCHITECTURE

The implementation of GPRS in the GSM architecture can be achieved by adding two new nodes: the Serving GPRS Support Node (SGSN) and the Gateway GPRS Support Node (GGSN). GPRS will be able to offer data rate up to 171.2kbps, but currently it is limited to 48kbps (Portolani, 2004). Figure 2.2 shows a GSM-GPRS network.

GPRS technology is used in this project to connect the MGM1 to the internet. It was also intended to use it for data and pictures transfer from MGM1 to mobile phones.

2.2.5 WIRELESS APPLICATION PROTOCOL

The Wireless Application Protocol (WAP) is a standard developed to link mobile phones to the internet.

It establishes the way a mobile phone should communicate with a server installed in a mobile phone network, in order to make content from the internet easily available on mobile terminals.

WAP actually defines the way content from internet should be filtered for mobile communication (Van der Heyden, 2005).

2.3 ROBOTICS

2.3.1 ROBOTICS HISTORY

The word robot originates from a Czech word “robota” which means tedious labor. In 1920, a Czechoslovakian playwright Karel Capek in his play R.U.R (Rossum’s Universal Robots), introduced the word robot to the world. Long before the word robot was used, a number of mechanical systems or machines were already developed (Dowling, 1996).

In ~350 B.C a brilliant Greek mathematician Archytas of Tarentum built a mechanical bird propelled by steam and ~200 B.C another Greek inventor and physicist Ctesibus of Alexandria designed water clocks that measured time as a result of the force of water falling through it at a constant rate, this revolutionized the way of measuring time from the time glasses that had to be turned after all the sand had run through (Brief history of artificial intelligence, 2004).

In 1495, Leonardo DaVinci designed a mechanical device that looks like an armored knight. In 1738, a French inventor Jacques de Vaucanson started building automata. He built three automata in all. The first one was a player that could play twelve songs, the second one could play a flute, a drum or tambourine, and the third one which was the most famous, was a duck that could move, quack, flap its wings and even eat and digest food.

The duck was Vaucanson's attempt at what he called "moving anatomy" or modeling human or animal anatomy with mechanics. In 1770, the Swiss Pierre Jacquet-Droz and his son Henri-Louis Jacquet-Droz, clock makers and inventors of the modern wristwatch started making automata for European royalty. They created three dolls each, having a unique function. One could write, another could play music, and the third one could draw pictures. In 1898, Nikola Tesla built and demonstrated a remote controlled robot boat at Madison Square Garden (Bedini, 1999).

Then in 1921, a Czech writer Karel Capek introduced the word "Robot" in his play "R.U.R" (Rossum's Universal Robots). In 1926, Fritz Lang released his movie "Metropolis", in which "Maria" the female robot in the film was the first robot to be projected on the silver screen. In 1940, Issac Asimov, a science fiction writer produced a series of short stories about robots for Super Science Stories magazine. The first one was "A strange playfellow" later called "Robbie". It is the story about a robot and its affection for a child that it is bound to protect. Asimov introduced the term "Robotics" which he first used in his book "Runaround" in 1942. But Asimov's most important contribution to the history of the robot is the creation of his three laws of robotics (Dowling, 1996):

- A robot may not injure a human being, or, through inaction, allow a human to come to harm.
- A robot must obey the orders given to it by human beings except where such orders would conflict with the first law.

- A robot must protect its own existence as long as such protection does not conflict with the first or second law.

Asimov later added a “zeroth law” to the list, which is stated as follow:

- A robot may not injure humanity, or, through inaction, allow humanity to come to harm.

In the 60’s a number of robotic research centers and companies were formed, and the first industrial robots appeared on the scene in the automotive industry. Since then, the field of robotics has evolved a lot. Most of the research focus is in the direction of Machine and Artificial intelligence at this actual time (Brief history of artificial intelligence, 2004).

2.3.2 ROBOT DESIGN

2.3.2.1 General system

Generally, the design of a robot requires a main control unit, sensors, input and output interfaces, response units, and a power supply. Figure 2.3 is a general block diagram of a robot.

The mainframe: It is the mechanical housing and the supporting framework for the machine. Much of the machine’s physical appearance is dictated by the nature of the mainframe assembly.

Internal power supply: This is the source of electrical power. It directly supplies all internal electrical circuits and external response mechanisms.

External power supply: This is used to recharge the internal batteries and provide electrical power for test and troubleshooting procedures.

The internal response mechanisms: These are devices that provide responses relevant to the machine's internal operation.

The external response mechanisms: These are devices such as motors and loudspeakers that provide the means for making responses that alter the machine's external environment (Heiserman, 1981).

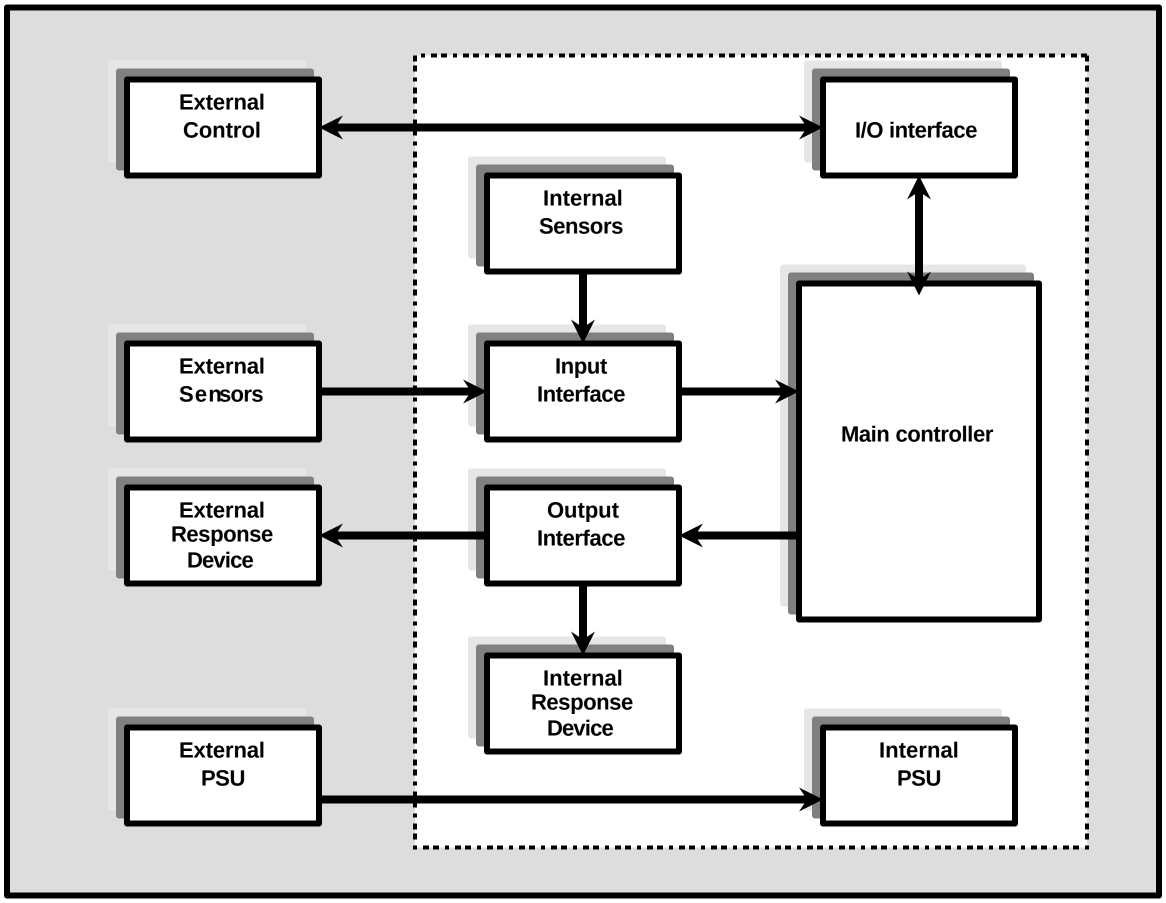


FIGURE 2.3 ROBOT GENERAL BLOCK DIAGRAM

The design of robots in general and of mobile robots in particular, is based on the use of some kind of drive to power the system (motors), sensors, software and control techniques and algorithms. The following section will discuss the devices, tools and techniques most used in the design of robots.

2.3.2.2 Actuators or drive systems used

A drive system is basically a device used to make a robot move and perform tasks such as lifting or moving other objects. There are many types of drive systems in use, the three main types are:

- Hydraulic
- Pneumatic
- Electric motor

2.3.2.2.1 Electric motor drives

Electric motor drives are mostly used where mobility and precision are required rather than big force. Electric motors are by far the most common drive system to be found in mobile (and hobby) robots. These motors can work with very great accuracy, controlling movements up to some fractions of a centimeter but when a very big force is needed, electric motors tend to get very bulky, and the power-to-weight ratio is no longer interesting (Warwick & Garrod, 2001 N°5:3-4).

a) Different types of motors

There are many different types of DC motors. The most used one's in the field of robotics are simple DC motors, stepper motors and servo-motors.

b) Selecting DC motors

In order to implement some mechanical motion in any project, there is a need for drive systems like motors, hydraulic, or pneumatic drives. Motors are the most used except in some specific cases where the size to force ratio for motors is not interesting anymore.

Motors might be fairly large or rather small, depending on the amount of mechanical loading that will be applied to them. One of the important requirements for selecting a motor is to get one that is large enough for the job at hand, but not excessively large. Overly large motors add needless bulk to the project and, in many cases, waste valuable electrical power. Another part of motor selection is to come up with a motor system that runs at the desired speed.

Motors that are not geared down to achieve lower operating speeds are rarely useful in robot projects. The speed adjustment is generally achieved by the use of gear motors.

The ideal situation is to calculate the speed and torque requirements for the motors and then purchase the motors according to those requirements. In practice, it is extremely expensive to buy new motors according to specifications, especially when more than one motor is needed. So it sometimes becomes necessary to use surplus motors or gear motors.

The amount of current that a motor can draw from the voltage source is equal to the voltage divided by the winding impedance. This means that the motor current is directly proportional to the amount of applied voltage. The running speed of a motor is proportional to the applied voltage and inversely proportional to the mechanical loading on the shaft. Also the current demand of the motor is inversely proportional to the running speed and proportional to the mechanical loading.

Torque is the measure of the ability of the motor to do useful work. In other words, it is an indication of the motors twisting force or power. It can be calculated using Equation 2.1:

$$T = F_{\tan} \times l = F \times l \times \sin f \qquad (2.1)$$

Here,

T: Torque in Newton meter (N.m).

F: Force in Newton (N).

$F_{\tan}$: Tangential component of the force in Newton (N).

l: Distance between the force and axis of rotation in meter (m).

In the case of winch or wheel, the force is always tangent, so Equation 2.1 can simply be written as:

$$T = F \times l \qquad (2.2)$$

The first step in selecting a motor is to determine as reasonably as possible the mechanical specifications of the system relevant to the motor: rpm, torque and power. This will help determine the electrical specification for the power supply as well (Heiserman, 1981:15-20).

Equation 2.3 is the relationship between the mechanical power required, the amount of weight to be moved and the linear speed.

$$P = \frac{F \times d}{t} = F \times v \qquad (2.3)$$

Here,

P: required mechanical power in watt (W).

F: Amount of weight to be moved in Newton (N).

t: Time interval required for moving the weight in seconds (s).

d: distance of displacement in meters (m).

v: Linear speed in meter per second (m/s).

2.3.2.3 Sensors

What makes a robot versatile, powerful and fascinating is its ability to collect data, and react or change its behaviour based on that data. Much of the information a robot requires to perform its job comes from sensors. These are devices that collect information about the robot itself or some part of the world around it, and transmit it to the robot's computerized controller. Without sensors a robot would be nothing more than an automated machine. Sight, hearing, touch and other senses, though, give it the means to think for itself (Warwick & Garrod, 2001-6: 9).

Sensors are transducers that convert a certain measurable quantity in the real world into an electric signal. There are many different types of sensors that can be used in robotic applications but they can be divided in 4 main groups: Light sensors, sound sensors, force sensors and position sensors.

2.3.2.3.1 Ultrasound sensors

Ultrasound is very high frequency sound that cannot be heard by humans. Ultrasonic sensors rely on a principle known as echo-location to locate an obstacle. An ultrasonic sensor has two parts, a transmitter and a receiver. The transmitter sends out a signal as continuous pulses of ultrasound. If the pulses hit an obstacle they are reflected back towards the sensor. The time it takes for the signals to bounce back is converted into an exact measure of distance. Ultrasonic sensing depends on the reflective surface or object's density, which affects its ability to reflect sound. Providing an object is dense enough to return the sound signal, ultrasonic sensors can tell whether the object is there, whether it is see-through or not.

So ultrasonic sensors are the best choice for industrial robots designed to work with clear glass or plastic bottles and containers, which lasers, for example, may not detect. Ultrasonic detectors also work in fire or smoke-filled environments. Ultrasonic sensors use sound waves above the range of human hearing. As the sound signal returns after bouncing off an object, it is collected and the time measured for it to return can be easily converted into a measure of distance (Warwick & Garrod, 2001-6:10).

2.3.2.4 Power supply

For a robot to operate properly, there is a need for a power system to supply electrical power to the motors, relays, electronic circuits, and other electrical devices.

The main power supplies for robotics applications are either the standard utility power sources or batteries. The big disadvantage of the utility power source is that the robot will be limited in its motion by a power cable. So the most suited type of power supply for a mobile robot are on-board batteries. Battery operated mobile robots require a power scheme with the following components:

- On-board batteries
- Battery recharging system
- Power distribution and control system

In order to simplify the design of the power supply scheme, it is essential to consider using a battery which voltage rating matches that of the motors and most of the other devices to be powered in the system (Heiserman, 1981:49).

There are two possible configurations when using batteries. The first configuration is to use a single battery to supply the entire system, and the second configuration is to use two or more batteries.

One to supply the high-current electromechanical devices and another supply the noise sensitive electronic circuits. The two main types of batteries suitable for robotics applications are lead-acid and gel-cell batteries. Nickel cadmium (ni-cad) and carbon-zinc batteries can also be use to supply electronic circuits (Heiserman, 1981:20-46).

The battery recharging system needs to match the specifications of the battery, this refers mainly to the charging rate, since charging the battery faster than as specified, reduces the battery's life span.

The power distribution and control system consists of the wiring, protecting circuits (fuses), regulators, voltage step up and voltage step down circuits, and current limiting circuits (Heiserman, 1981:50).

2.4 AVAILABLE STUDY ON ROBOTS AND CONTROL

Many trends and many different approaches have emerged over the years in robot design. It will not be practical not even possible to go through all that has been done in the field of robotics until now, but an overview of some approaches can be given. In this section, examples of a passive system and a very complex and advanced controlled system are discussed.

2.4.1 PASSIVE SYSTEM

The passive system considered here is the "passive dynamic walker". This robot is capable of walking down an incline without any actuation and without control. In other words, there are no motors and there is no microprocessor on the robot; it is brainless, so to speak (Pfeifer et al., 2003).

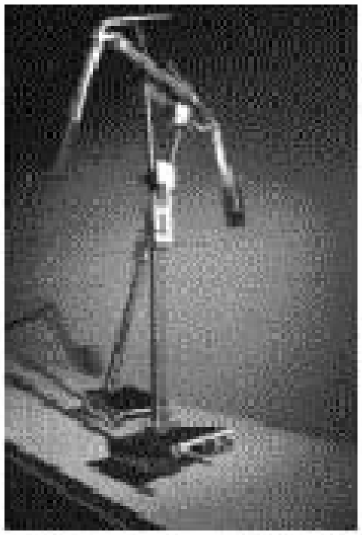


FIGURE 2.4 PASSIVE DYNAMIC WALKER

This passive motion has been achieved by exploiting the dynamics of the robot, its body and its limbs. This kind of walking is very energy efficient and there is an intrinsic naturalness to it. Figure 2.4 shows the passive dynamic walker.

2.4.2 COMPLEX SYSTEM

In this second case, reference will be made of Asimo, designed by Honda's Engineers using a different approach than the passive dynamic walker. Asimo was designed to perform a larger number of different types of movements. This Honda robot is able to do things such as walk up and down stairs, push a cart and open a door.

"The methodology was to record human movements and then to reproduce them on the robot which leads to a relatively natural behavior of the robot.

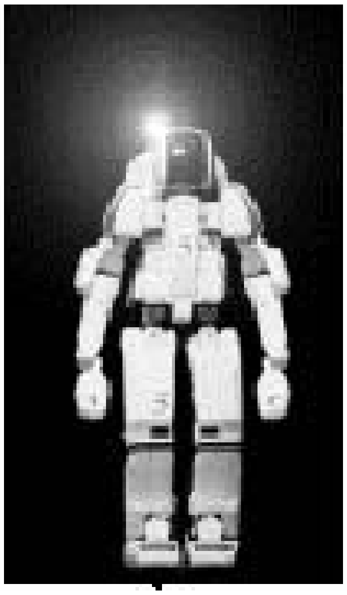


FIGURE 2.5 HONDA ROBOT ASIMO

On the other hand control (or the neural processing if you like) is extremely complex and there is no exploitation of the intrinsic dynamics as in the case of the passive dynamic walker" (Pfeifer et al., 2003). In this specific case, the movement is not energy efficient. Figure 2.5 shows Honda's Asimo.

2.5 RESEARCH FIELDS

Advances in robotics introduced a new field of research called artificial intelligence which aims at developing advanced and intelligent robots.

Artificial Intelligence can be defined as a discipline that deals with programming computers to carry out tasks that would require intelligence if carried out by humans. It deals with tasks considered to require knowledge, perception, reasoning, learning, understanding, and other cognitive abilities (Webster, 1999:205).

EAMI (Evolutionary Adaptive machine intelligence): Refers to a particular process for developing a hierarchy of machine intelligence. Using Evolutionary adaptive Machine Intelligence, it is possible to build an intelligent machine gradually. There are three different classes in the EAMI: Alpha-Class Intelligence, Beta-Class Intelligence, and Gamma-Class Intelligence.

An Alpha-Class machine responds to its environment in a purely random fashion. It learns and remembers nothing. A Beta-Class machine does remember responses to conditions it experienced directly at some time in the past. It responds according to remembered experiences, but it cannot anticipate situations never dealt with on a first-hand basis. The shortcoming of the Beta-Class machine is covered by the third, and final step in the EAMI program which is the Gamma-Class machine (Heiserman, 1981:15 - 17).

2.6 ADVANTAGES AND DISADVANTAGES OF ROBOTICS

The use of machines has a very high initial cost, but it increases production since machines can work faster and for longer periods of time than humans can.

Machines have improved the working conditions of humans by assuming hazardous and monotonous jobs and reduce production costs because they produce fewer “rejects” that humans sometimes produce through fatigue or boredom. Robots improve productivity in a variety of applications from processing raw materials to assembling automobiles. They are especially useful for work in hostile or dangerous environments, such as in outer space or on the ocean floor. Finally, robots are fun to work with. They provide challenging opportunities to everyone from hobbyist to the most advanced robotic designers. Industrial robots have been used in a wide variety of manufacturing applications. Hot, dirty, dangerous foundry work in which molten metal is poured into castings was one of the first jobs in which robots were successfully used.

Welding operations, in which consistency of the spot or seam weld is essential but which also produces a hot, ozone atmosphere annoying or hazardous to humans, has become another widely used application. Hazardous spray painting is another application in which robots are important, because robots can safely apply extremely thin coats of paint consistently, significantly reduces the amount of paint needed per part. Back-breaking, dangerous, and tedious machine loading and unloading is another task to which robots are often applied. Most robots used in these applications are deaf, dumb, blind, and stationary. Thus, these robots are not used so differently from other kinds of automated machines.

However, an entirely new phase in robotics applications has been initiated with the development of “intelligent” robots. An intelligent robot is basically one that is equipped with some sort of sensory apparatus that enables it to sense and respond to variables in its environment. Much of the research in robotics has been and is still concerned with how to equip robots with seeing “eyes” and tactile “fingers”. Artificial intelligence that will enable the robot to respond, adapt, reason, and make decisions in reaction to changes in the robot’s environment are also inherent capabilities of the intelligent robot (Hall, 1985:4-5).

The main disadvantage which may result from the increasing reasoning power of nowadays and future robots is the loss of jobs, and the danger sophisticated robots may become to humanity as predicted by science fiction writers.

CHAPTER THREE

SYSTEM DESIGN AND IMPLEMENTATION

"You know you've achieved perfection in design, not when you have nothing more to add, but when you have nothing more to take away." Antoine de Saint Exupery

3.1 INTRODUCTION

A robot is a mechatronics system, which is made not only of mechanical and electronics components, but also of built-in software constituents for controlling them. Designing such a system requires multidisciplinary expertise since the disciplines involved span from electronics, sensor technology, computers, software development, control engineering, mechanical design, materials and manufacturing. MGM1 was no exception to the rule.

Looking at the robot and the functions to be implemented as a complete system, the design of it was done in two phases. The first phase was to draw a conceptual design of the system as a whole. The system was divided into five different parts: Mechanical, Electronics, Software, Motor control, and Telecommunication. The next phase was to produce a detailed design of each of these subsections of the system.

In this chapter we will discuss the design of MGM1, first the general design and then the detailed design with the various existing options.

3.2 CONCEPTUAL DESIGN

The aim of the project was to develop a wheeled robot, which can interact with people through the GSM network. The shape of the machine was not of the utmost importance but rather the robot had to be able to move forward, backward, turn left and right, reply and execute commands originating from a mobile phone.

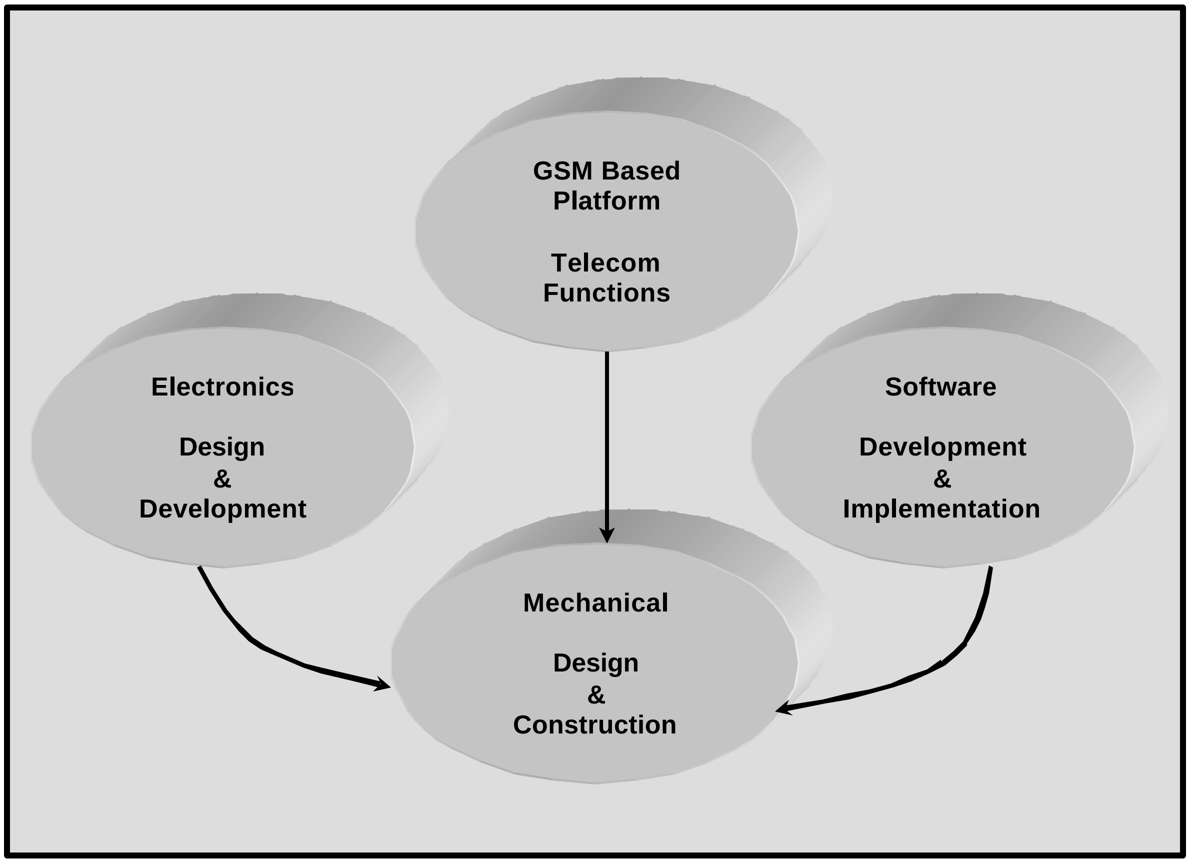


FIGURE 3.1 SYSTEM DESIGN APPROACH

The approach to the design was to develop a GSM based platform, and then build all the telecommunication functions on it, develop the electronics and software for motion and control, and finally construct a mechanical structure or a frame in which all the other parts could be integrated. Before implementing this design procedure, a general view of the final outcome was necessary.

Considering the possible size and the weight of the robot to be developed, it was found more suitable to base the design on a stable wheel configuration. Generally, the greater the number of wheels, the better the stability. A three wheels system is more stable than a 2 wheels system, but four wheels is more stable for most of the ordinary terrains. So the design of MGM1 is based on a four wheel configuration. In order to simplify the control and the steering of the system, two of the wheels are driven by motors and the other two are castor wheels for increased stability. Based on this wheel configuration, MGM1 has two main motors.

For the robot to demonstrate the use of GSM services and especially in in-door environment, there is in fact no need for great physical power, the electric motor drive system is the most suitable in this case. The main parts of the robot to be mobile are the wheels only on this first prototype. Since MGM1 has two driven wheels and two castor wheels, therefore there will be one motor for each driven wheel.

MGM1 needs a powerful processor to manage and control the whole system. Since, a LCD screen has to be incorporated in the robot and a windows based interactive software has to be developed to control the whole system, incorporating a PC in the design of MGM1 became more convenient, because as one package, it offers most of the required accessories.

Many different configurations could be used for the system's internal architecture, linking all the components together.

A centralized architecture could be used whereby all the devices and electronics circuitry are connected and controlled by the main processor of the system.

A decentralized architecture could also be used whereby each subsystem has a microcontroller and makes decisions by itself, this is called distributed microcontroller architecture. The actual architecture implemented in MGM1 is a sort of combination of the above-mentioned architectures.

Considering the number of peripherals and electronics circuits supporting the system and using the PC as the main processing unit, there is a risk of overloading the processor. Also the fact that the PC is windows based and windows not being a real time operating system, assigning the full control of the system to the PC will result in very poor performance.

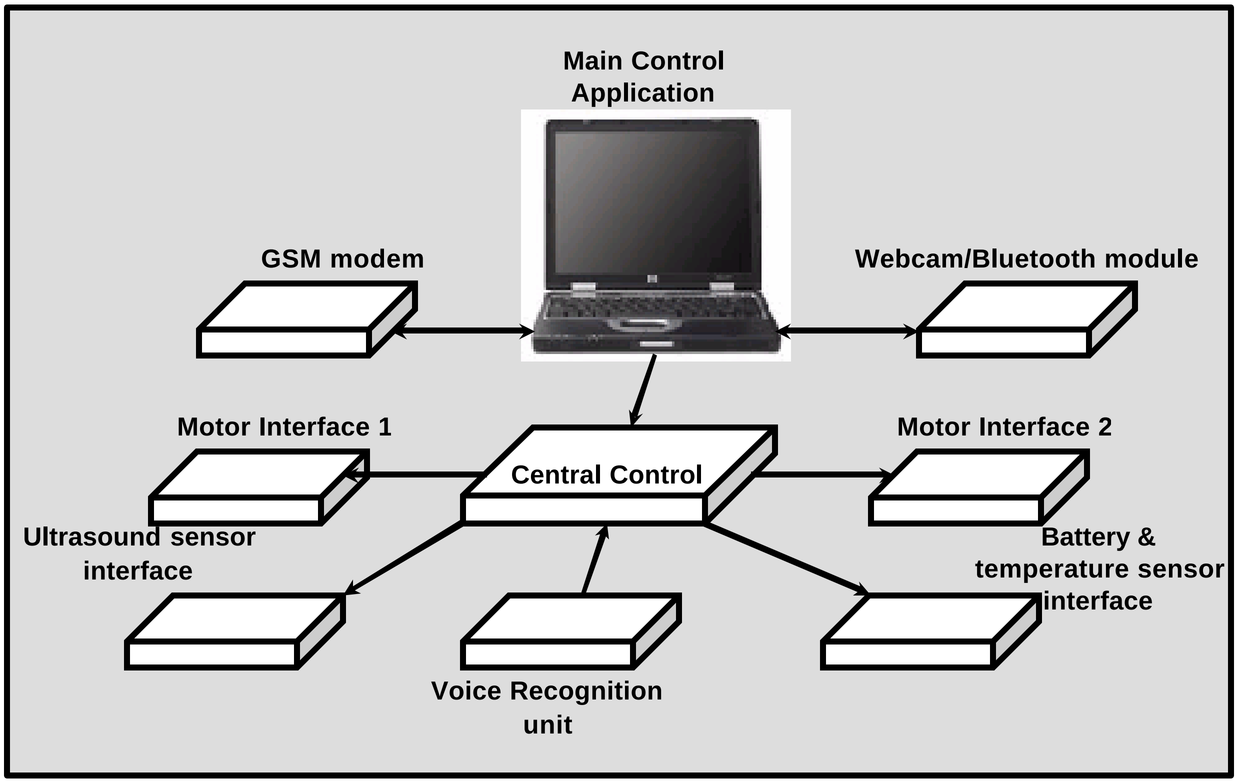


FIGURE 3.2 INTERNAL ARCHITECTURE

The internal architecture of MGM1 is a partially decentralized architecture, using distributed microcontrollers. The different boards communicate through the RS232 interface. **Though RS485, I^2C, or CAN bus could also be used, RS232 was chosen first of all because very high data rate was not required, and secondly because of the simplicity to implement, and also because the GSM module had a build-in RS232 interface.** This internal architecture is shown in Figure 3.2.

This is implemented by using a microcontroller in each subsystem, making it enough intelligent to make the necessary decisions on basic and regular tasks without disturbing the main processor. All the subsystems are then connected to a central control board, which has a more powerful microcontroller. The central control board controls all the subsystems, making decisions at a higher level. One of the main reasons for introducing the central control board is to keep the system real time by limiting the control of sensors and response devices suc h as motors at its level and not involving the PC at all. The central control board is also the bridge between the messages and commands received by the PC and the response devices (motors), ensuring proper data transfer. It is also the bridge between all the subsystems, since it acts as a serial port expander and a router, directing all received data to the right recipient.

3.3 DETAILED DESIGN

This section is an attempt to a detailed design of MGM1 prototype robot. In order to simplify this, the whole system will be divided in 5 subsystems: Telecommunication, Electronics, Motion control, Software and Mechanical. The design of each of these subsystems will be considered at a time.

3.3.1 TELECOMMUNICATION

The telecommunication part of the system consisted in the implementation of the remote control function using GPRS, sending and receiving SMS and MMS, recovering voicemails, and web surfing using a phone browser via a GSM network.

In the architecture of MGM1, the PC is the brain of the whole system, so the PC inside the robot has to be able to connect to the GSM network so that it either can be reached by using a remote mobile phone or can reach any remote mobile phone with a known number. A feasibility study was conducted to determine the possibility of connecting to the GSM network and implementing either GPRS, HSCSD, or USSD technologies for remote control purposes, and finally propose an approach and the technology or equipment to use.

To establish a communication between the robot and a GSM network, there is a need of incorporating a device that can link the robot's brain to the mobile network. A mobile phone is one option but is not easy to use in such a development environment.

Certain Telecommunication equipment manufacturers do commercialize devices such as GSM modules and GSM PC cards, but since the GSM module is the main component around which a mobile phone is built, it is more suitable for development than an actual mobile phone since it is more flexible due to the fact that it provides input/ouput pins, serial communication interface, and allows to customize the settings depending on the type of application. Therefore in this project, the GSM module was found to be the best option.

3.3.1.1 The GSM module

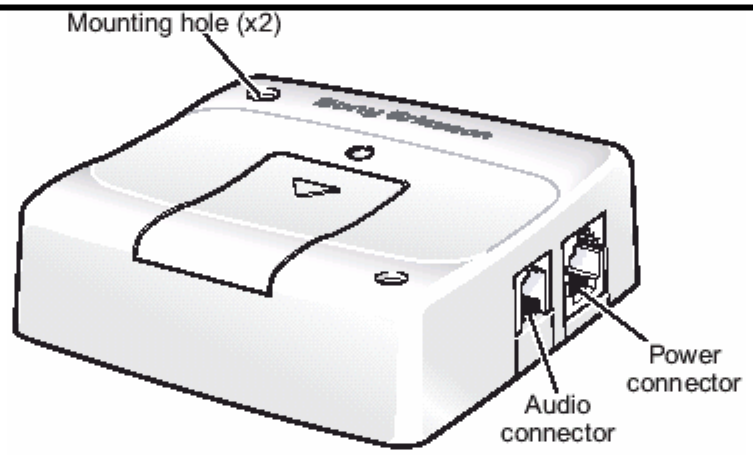


FIGURE 3.3 SONY-ERICSSON GSM MODULE GM29

A GSM module is a serial modem that enables Man-to-Machine and Machine-to-Machine (M2M) wireless communication. It comes as an interface between the machine and the GSM network. Figure 3.3 shows the GSM module used in MGM1.

The GM29 is a plug-and-play dual band (900/1800 MHz) serial modem, capable of sending and receiving data by GPRS, HSCSD, CSD, SMS, and fax. It can also handle voice calls. The modem is controlled by external applications via the RS232 serial interface using AT commands. The AT standard is a line-oriented command language where each command starts with the prefix “AT” which stands for Attention. AT commands are used to interact with the modem and obtain:

- Modem’s general parameters configuration
- Setup and control of the communication to and from the GSM network
- Communication across the RS232 serial interface
- GSM network status information

Initialisation and SMS handling	
Command Description	**Command Syntax**
Insert pin number	AT+CPIN = “pin code”
Set to text mode	AT+CMGF = 1
Work from modem’s memory	AT+CPMS = “ME”
How many new messages	AT+CPMS ?
Sending SMS	AT+CMGS = “n”
Read SMS	AT+CMGR = a
Delete message	AT+CMGD = a

Table 3.1 List of AT commands for initialization and SMS handling

GPRS Connection	
Command Description	**Command Syntax**
Dial	ATD<dial_string>
Select PDP context	AT+CGDCONT = 1, “IP”, “APN”
GPRS attach	AT+CGATT = 1
PDP context activate	AT+CGACT = 1,1
Context activate or enable	AT*E2IPA = 1,1
IP address request	AT*E2IPI = 0
Destination IP definition-Dial up	AT*E2IPO = 1, “IP”, m

Table 3.2 List of AT commands for GPRS configuration

a:	*refers to the position of the message in memory.*
n:	*is the MSISDN number (phone number) of the destination MS.*
m:	*Used protocol port number*

AT commands are provided a document by the manufacturer of the specific modem. In that document, they are classified by technology (e.g: GPRS and HSCSD) or services (SMS). The AT commands mostly used in this project are summarized in Table 3.1 and Table 3.2.

3.3.1.2 Implementation approach

Using the GM29 and the PC, an early experimental platform was mounted to study the connectivity to the GSM network, and how possible it is to use GPRS, and HSCSD, and how to implement services like SMS, MMS in the robot. Below is the basic experimental setup used for that purpose. The first experiment using this setup was done utilizing a terminal program (e.g: Hyperterminal) to send AT commands to the modem via the serial interface. This experimental setup in shown in Figure 3.4.

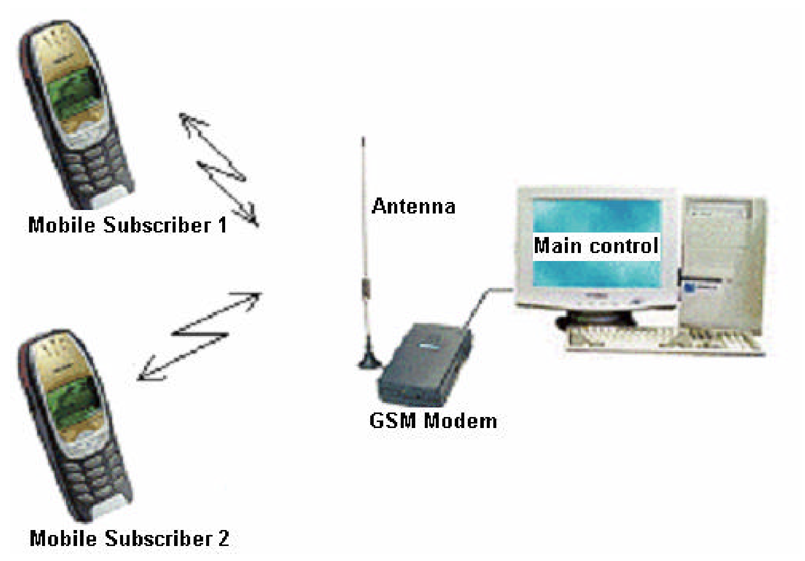


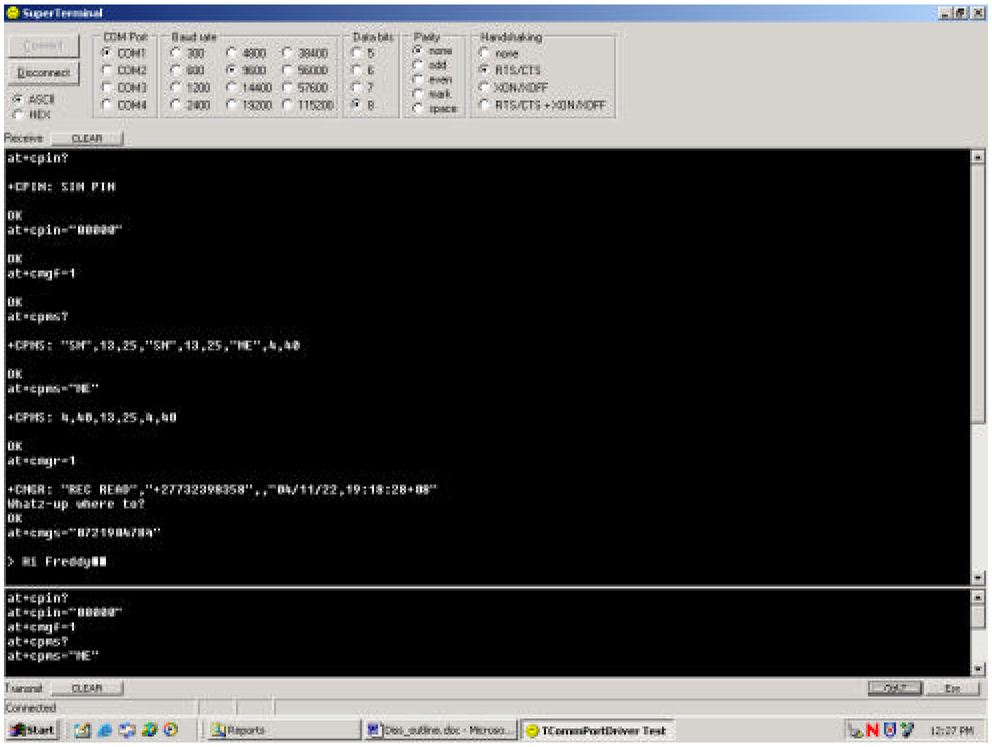
FIGURE 3.4 GM29 CONNECTION AND USE

By default, on power on, the modem is on command mode which means it accepts and understands AT commands. So by using the PC's keyboard, it was possible enter the pin code to enable the modem, set the modem to text mode, select and access the SIM card memory as well as the modem's memory, delete some messages, and more other control tasks.

An interesting thing is that it was possible to send SMS from the PC to a mobile phone and vice versa. It was also possible to configure the modem for GPRS, and HSCSD using AT commands in this fashion. Step by step procedures for configuring the modem with AT commands for most of the above mentioned functions are in Appendix A. Figure 3.5 shows the configuration being done in a terminal application (superterminal).

The next step in the implementation of these telecommunication functions, was to get messages from the mobile phone stored, displayed, and then replied to by the PC without anybody typing on the PC's keyboard.

So an application had to be developed on the PC to handle and manage the flow of messages and commands and automatically reply to them in a predefined way. For this purpose, an application called “MGM1_ControlSoft“ was developed in the Delphi 7 environment.



COM Port: [✓ COM1] COM2 COM3 COM4; [✓ ASCII] HEX
Baud rate: 300 600 1200 2400 4800 [✓ 9600] 14400 19200 38400 56000 57600 115200
Data bits: 5 6 7 [✓ 8]
Parity: [✓ none] odd even mark space
Handshaking: none [✓ RTS/CTS] XON/XOFF RTS/CTS+XON/XOFF

```
at+cpin?

+CPIN: SIM PIN

OK
at+cpin="00000"

OK
at+cmgf=1

OK
at+cpms?

+CPMS: "SM",13,25,"SM",13,25,"ME",4,40

OK
at+cpms="ME"

+CPMS: 4,40,13,25,4,40

OK
at+cmgr=1

+CMGR: "REC READ","+27732398358",,"04/11/22,19:10:28+08"
Whatz-up where to?
OK
at+cmgs="0721984784"

> Hi Freddy
```

```
at+cpin?
at+cpin="00000"
at+cmgf=1
at+cpms?
at+cpms="ME"
```

FIGURE 3.5 GM29 CONTROL USING A TERMINAL PROGRAM

The application is built using AT commands to communicate with the modem, to control and configure it. It was then possible to display a mobile phone originated message on the PC's screen, and send a message (a reply) from the PC to the mobile phone. MGM1_ControlSoft is also responsible for establishing GPRS and HSCSD connections for data transfer.

The developed telecommunication platform was eventually incorporated inside the robot so as to make the robot behave as a mobile station.

The module has a slot where a SIM Card is inserted, this provides the robot with a known MSISDN number that can be dialed to establish the communication or send messages and commands.

The use of a GSM module made the implementation of the telecommunication functions in the robot possible, but this could still not be without the control application developed in Delphi to manage the incoming commands and messages and ensure proper reply and execution.

3.3.2 ELECTRONICS

The robot being an electromechanical system, there are a number of electronics circuits design and used in this project, and this section will be a close look at each of those circuits. The electronics section takes into consideration all the electronic circuits necessary for a proper operation of the robot. The most important circuits developed in this project are the proximity detector circuit, the motor control circuit, the temperature measuring circuit, battery level measuring circuit, the central control circuit, and the position control circuit. **All the circuits discussed here have been designed from scratch, and built especially for this project.**

In the following sections, the design and operation of each of these circuits will be discussed in detail.

3.3.2.1 Ultrasound Proximity Detector Circuit

The purpose of this circuit is to provide the robot with some senses. MGM1 being a remote controlled mobile robot, it needs "eyes" to see the world around it, providing it with a mean of avoiding obstacles. Many types of sensors are available for proximity detection applications but often, infra-red and ultrasound sensors are used, and in this project ultrasound sensors were the best option because in this specific case they work better than infra-red sensors.

This is due to the fact that ultrasound detection is a function of the reflective surface or object's density, meaning that it can even detect transparent objects. Another advantage is that Ultrasound sensors' performance is not influenced by adverse conditions such as smoke. A detailed description of these sensors is given in chapter two.

3.3.2.1.1 Design

The proximity detector circuit has two different parts: the transmitting circuit and the receiving circuit. The transmitting circuit is composed of a 40khz square wave generator, a drive unit and the sensor itself (Ultrasound transmitter). Figure 3.6 is the block diagram of the transmitting circuit.

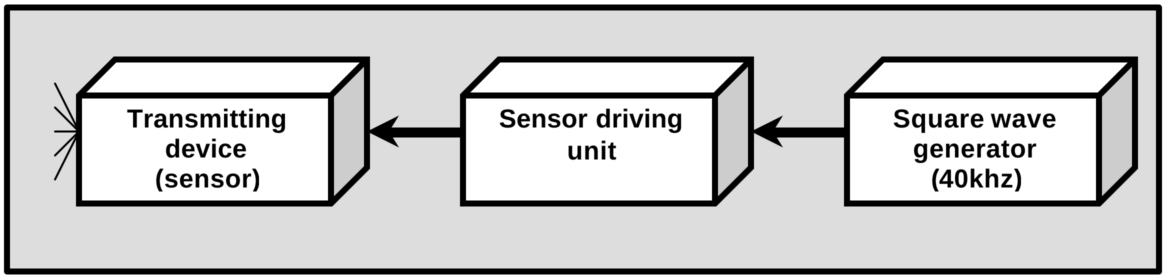


FIGURE 3.6 TRANSMITTING CIRCUIT BLOCK DIAGRAM

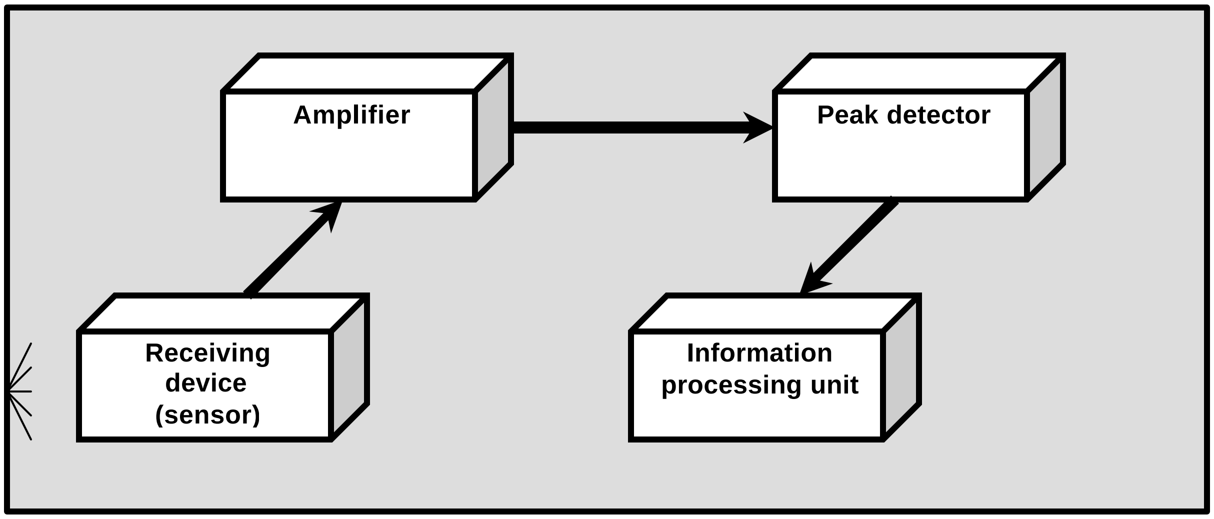


FIGURE 3.7 RECEIVING CIRCUIT BLOCK DIAGRAM

The receiving circuit is composed of the ultrasonic receiving sensor, a double stage amplifier, and a peak detector unit. The block diagram of this part is given in Figure 3.7. The schematic of the amplifier and the peak detector is shown in Figure 3.8.

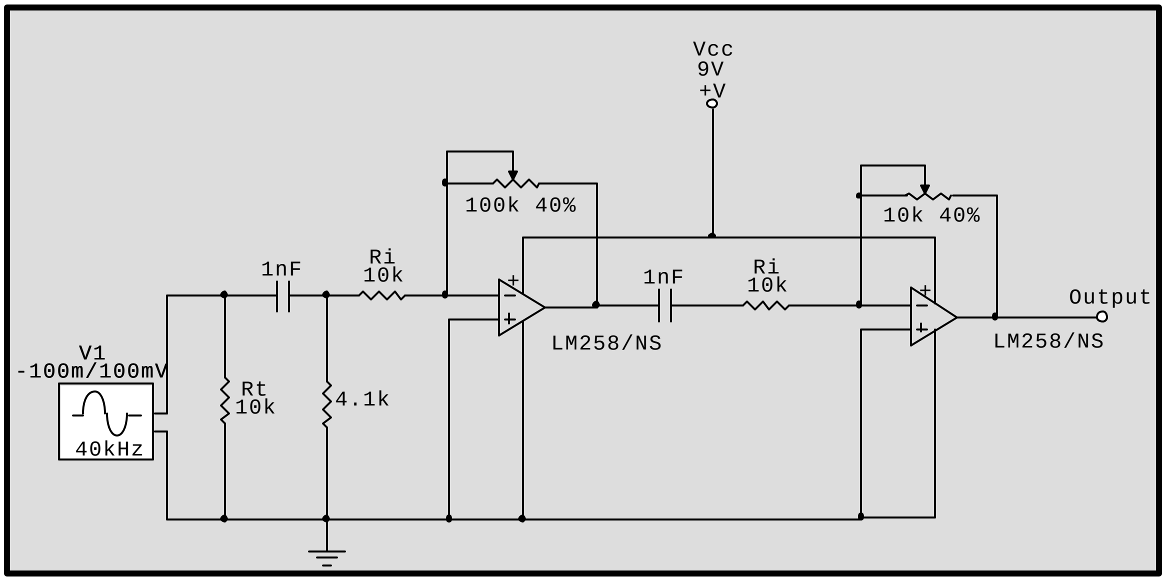


FIGURE 3.8 AMPLIFIER CIRCUIT

The above amplifier circuit was simulated, and the results of the simulation are shown in Figure 3.9. Both the schematic drawing and the simulation have been done using circuitmaker.

The proximity detector circuit had to be an intelligent circuit capable of making decisions at this level, so a microcontroller had to be introduced for that purpose. Using a microcontroller, it became possible to combine the transmitting and the receiving circuits and also reduce the number of on-board components, by using the microcontroller for signal processing, decision making and square wave generator. The block diagram of the combined circuit is shown in Figure 3.10.

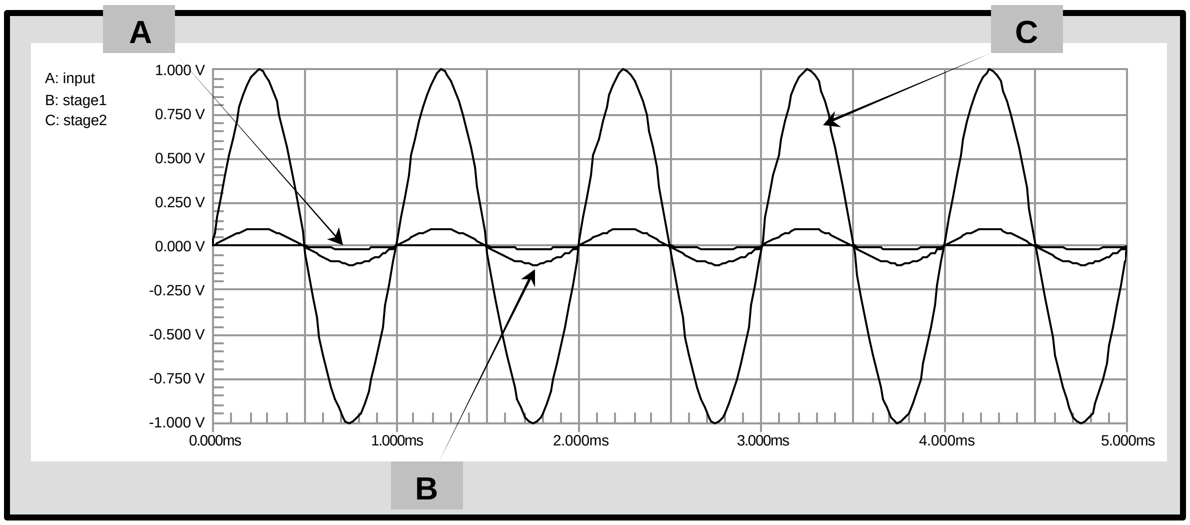


FIGURE 3.9 AMPLIFIER SIMULATION RESULTS

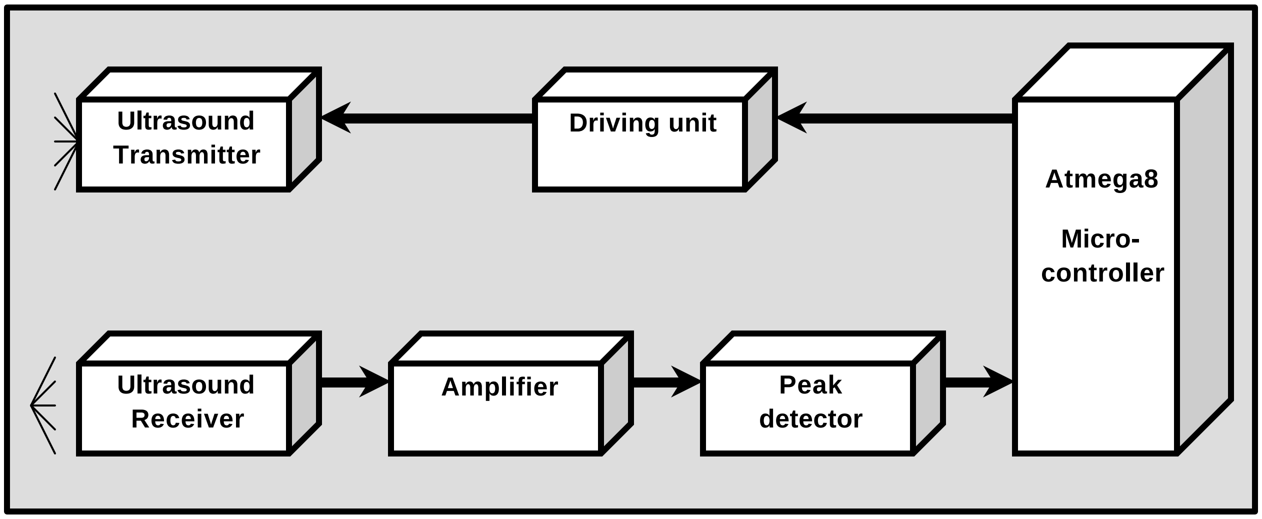


FIGURE 3.10 PROXIMITY DETECTOR CIRCUIT BLOCK DIAGRAM

The driving circuit is built on the RS232 integrated circuit, this converts the 5V signal generated by the microcontroller to a 10V signal that is fed into the ultrasonic transmitting device. The amplifier used in the circuit is a two stages amplifier, where the first stage is an inverting amplifier with a gain of 10 and the second stage a non-inverting amplifier with a gain of 10 as well.

The overall gain of the amplifier is 100, this is enough to boost a signal of about 20mV to an acceptable level of about 2V for further processing. The received signal is a sine-wave, so after amplification, it is rectified to an analog smooth signal which is fed to the microcontroller for analog-to-digital conversion. The PCB design of this circuit is found in Appendix B.

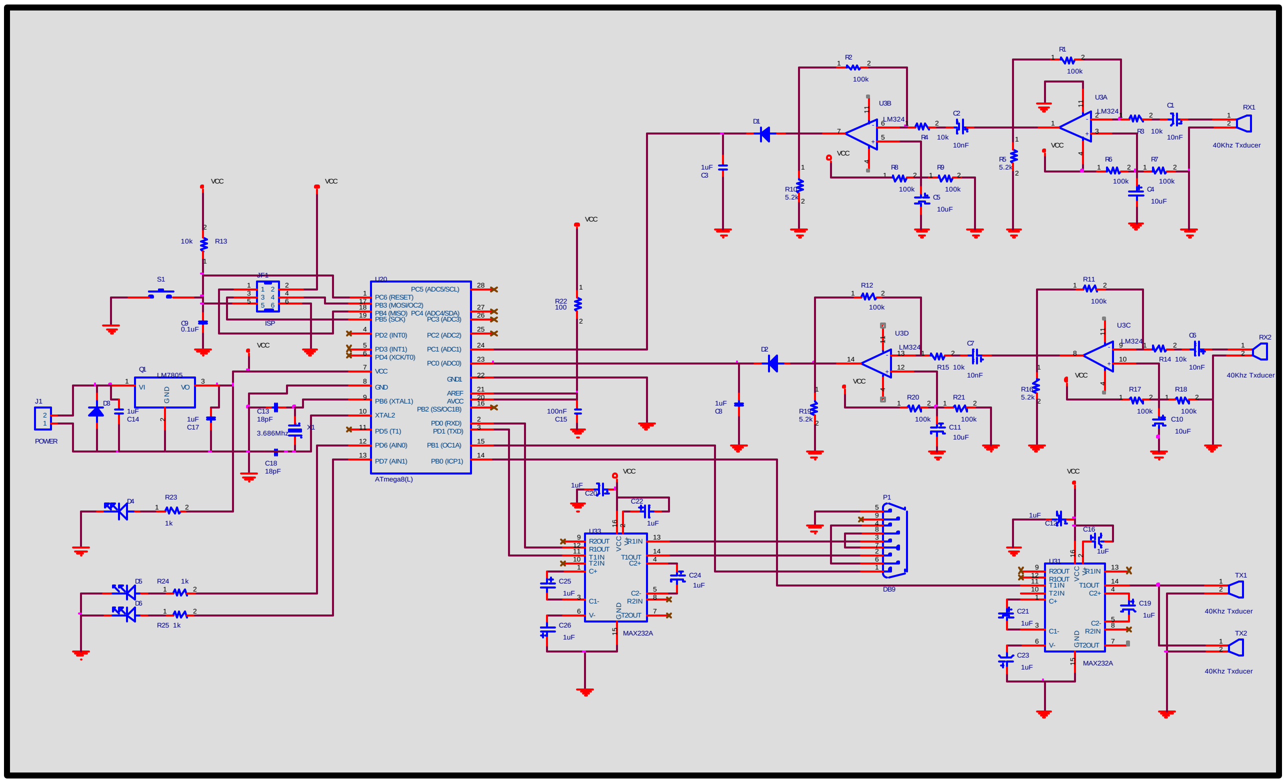


FIGURE 3.11 PROXIMITY DETECTOR UNIT SCHEMATIC

3.3.2.1.2 *Functional Description*

Refer to Figure 3.11. The microcontroller generates a 40kHz square-wave signal which is fed to the ultrasound transmitter driving circuit. The signal enters the driving circuit with an amplitude of 5V and gets out with an amplitude of 10V. This signal is then fed into the ultrasonic transducer (transmitter).

The generated ultrasound propagates through the air and if a surface or an object obstruct the sound wave propagation, the ultrasound will reflect against the object and travel back in the opposite direction and eventually reach the ultrasound receiver, which will detect the ultrasound and convert it into an electric signal of the same frequency but whose amplitude is proportional to the received ultrasound intensity. The received signal, which is in the order of millivolts, is then fed into the amplifier section to boost the signal amplitude to about 3V. This signal is then rectified and smoothened by the peak dectector section before it is finally fed into the microcontroller.

At this stage, the microcontroller starts by converting that analog signal into a digital one, quantifies the signal and then depending on whether the signal is above or below some preset values, the microcontroller is able to transmit an alert signal to the central control board when there is an obstacle or continue normal the analysis if the situation does not depict an obstacle ahead.

3.3.2.3 TELEMETRY FUNCTIONS CIRCUITS

Telemetry is a word coming from “Tele” which means “Distance” and “Metry” which means “Measurement”. In short, telemetry refers to the measurement of some quantities from a distance.

MGM1 is a battery operated robot, using motors and having a number of electronic circuits on-board everything placed within an enclosure made of aluminum profiles and Perspex material. This situation might result in an undesirable rise in temperature, which might affect the proper operation and even the life span of circuitries and other devices there incorporated. Also, the batteries used in the robot are exhaustible resources, which means that they will supply the system only for a certain period of time before they become flat.

It is undesirable for this to happen suddenly during a demonstration or at a time it is the least expected. Hence the need to monitor both the temperature and the battery voltage level in the robot and give a report about the current status of the measured quantities to a distant operator only when necessary. This will enable the operator to take the necessary precautions before the system is brought to a stand-still or before any damage occurs.

3.3.2.3.1 Design

The telemetry unit board is a combination of two different circuits, the battery voltage measuring circuit and the temperature measuring circuit.

3.3.2.3.1.1 Temperature measuring circuit

This circuit is mainly composed of a temperature sensor, and a microcontroller. The temperature sensor used is the LM35 and the microcontroller is the Atmega8. An amplifier could be used between the sensor and the microcontroller to boost the signal but in this case it was not found to be necessary since the signal produced by the sensor could be processed by the microcontroller in the A/D conversion.

3.3.2.3.1.2 Battery level measuring circuit

The battery level measuring circuit is a very simple circuit designed to assess the level of the supply battery. It uses a resistive voltage divider circuit as the sensing end and the microcontroller for analysis and data process. This circuit also does not need an amplifier since the signal measured by the sensing components is between 3V to 5V.

The temperature measuring circuit and the battery level measuring circuit are built on the same board so as to have a single telemetry unit and reduce the number of components to be used. The block diagram of the complete telemetry unit is illustrated in Figure 3.12.

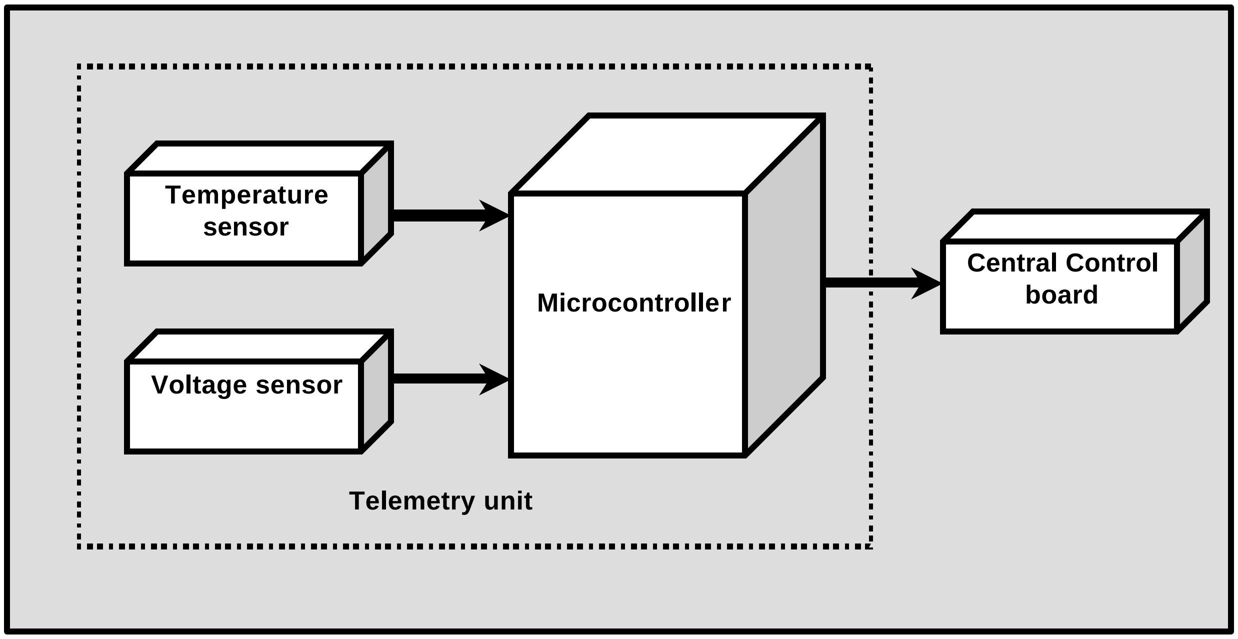


FIGURE 3.12 TELEMETRY UNIT BLOCK DIAGRAM

3.3.2.3.2 Functional Description

3.3.2.3.2.1 Temperature measuring circuit

The LM35 converts the ambient temperature into voltage, typically 10mV per degree Celsius. This voltage is then amplified by the non-inverting amplifier and the amplified signal is then fed into the microcontroller. Since the output of the transducer is an analog voltage, it needs to be converted into a digital signal which can be processed by the microcontroller. The used microcontroller (Atmega8) has an internal ADC (analog-to-digital) converter.

So the analog signal is fed to the microcontroller, analog to digital conversion is implemented by software. The code loaded in the microcontroller converts the analog voltage into a real temperature reading.

This reading is then compared with a thresholds temperature and if the reading is higher than the set limit, the microcontroller has to transmit that temperature reading to the central control board via the serial port. And if the temperature reading is lower than the preset limit, then the temperature is considered to be within the acceptable limits, so there is no need of transmitting the reading to the central board.

The report on the status of the temperature has to be delivered to a distant operator via the GSM network. For that purpose AT Commands are added to temperature reading in the code before the serial transmission. In this manner, the GSM modem will recognize the incoming data as commands and therefore will execute them. The software can be modified so as to report continuously about the status of the temperature on a constant interval of time.

3.3.2.3.2.2 Battery level measuring circuit

The system operates from a 12V battery, so for processing purposes, it is reduced using the voltage divider circuit. The resistor ratio of the divider is 0.4 so the read voltage becomes 4.9V. This voltage is fed straight into the microcontroller. This microcontroller is loaded with a code that does ADC conversion and further data processing. The circuit operates using the same principle as the temperature measuring circuit. The threshold voltage is a voltage 20% lower than the voltage at the start. So, when the measured voltage is greater than 9.6V, the battery is considered charged and no data is transmitted, and as the measured voltage approaches 9.6V, the microcontroller transmits the reading to the central control board via the serial interface.

Since this reading will later be transmitted to the GSM modem, AT Commands are added to the reading to enable the GSM modem to transmit that voltage reading to a distant operator via the GSM network.

3.3.2.2 MOTOR CONTROL CIRCUIT

The name motor control circuit refers to the electronic interface between the central control board and the actual motors. This interface is needed for the basic control of the motor, managing the PWM for soft start or stop, the forward and reve rse direction, and high current supply to the motors.

3.3.2.2.1 Design

A number of different technologies are available for high current supply to the motors. This can be achieved using BJT or FET transistors in a half bridge configuration, or by using half bridge IC's such as LD293. But in this project, due to the estimated weight of the robot and therefore the expected current, the option selected was the use of relays and a single high current FET in the circuit. In all, the circuit is made of a microcontroller, two relays, and driving transistors. The block diagram of the motor control interface is shown in Figure 3.13.

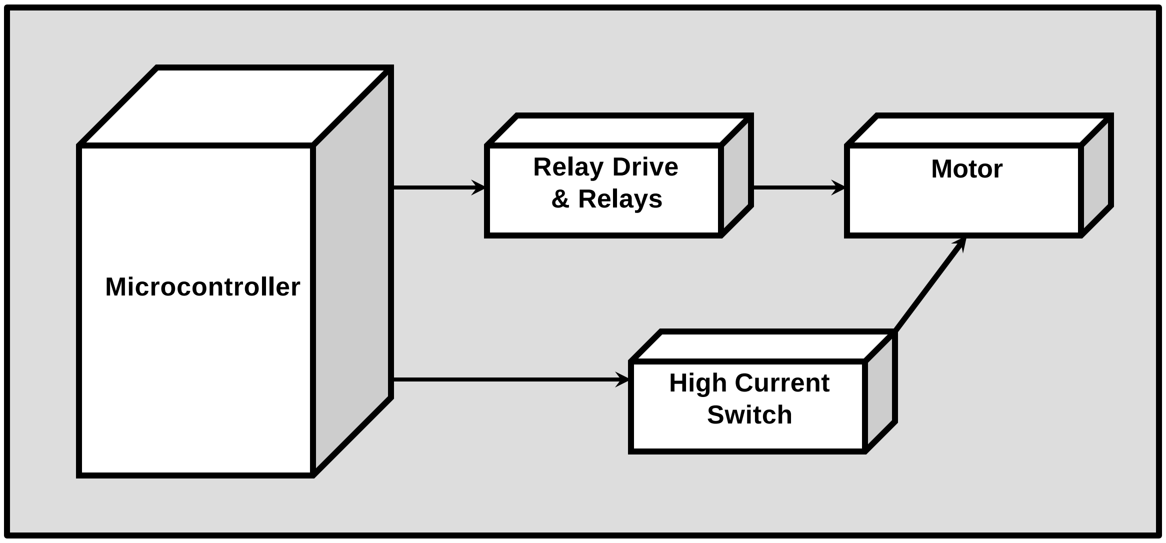


FIGURE 3.13 MOTOR CONTROL INTERFACE

The schematic diagram of the circuit is shown in Figure 3.14.

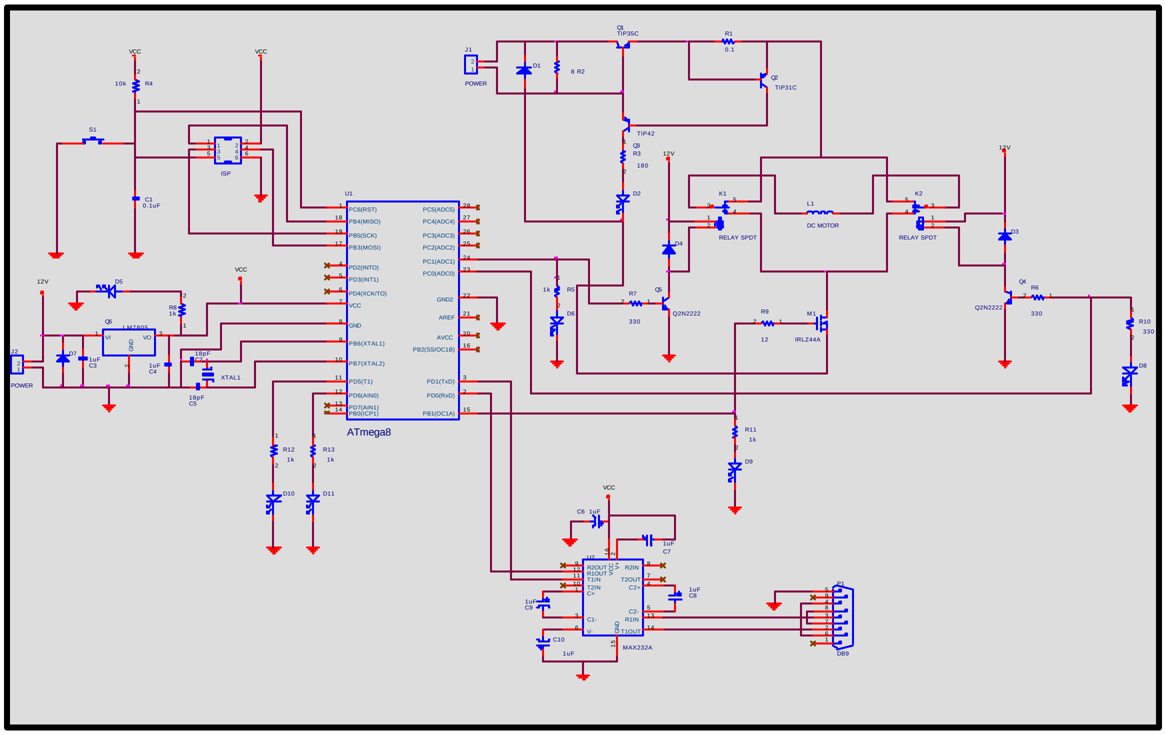


FIGURE 3.14 MOTOR CONTROL INTERFACE SCHEMATIC DIAGRAM

3.3.2.2.2 Functional Description

Refer to Figure 3.14. In this circuit, the microcontroller's function is to receive commands from the central control board and then execute them. The signal to manipulate the motors originates from the remote control mobile phone, goes through the GSM module, the PC and then to the central control board, from where it is forwarded to the motor control serial port. At reception of these commands, the microcontroller executes by driving the motor accordingly. If the received command is to run the motor forward for example, the microcontroller generates a PWM signal to start the motor slowly, gradually ramp it up to maximum speed, keeps it running for a certain time (set to 3 seconds for tests) and then reduces the speed gradually until the motor stops. It remains in that state until another command is sent.

If the received command is to reverse, the microcontroller changes the relay switches positions so that the current through the motor flows in the opposite direction. Then it generates the PWM to start the motor, run it for 3 seconds and stops it in a similar fashion as described above.

3.3.2.4 VOICE RECOGNITION CIRCUIT

Voice recognition is one the technologies MGM1 has to exhibit. MGM1 is already remotely controlled by a mobile phone, the aim here is to make it respond to vocal commands as well. At this stage, only a few predefined commands will be used.

3.3.2.4.1 Design

The voice recognition unit is built around the voice direct 364 module. Voice direct 364 is a voice recognition module capable of recognizing a maximum of 15 words while giving a BCD code of the position of the recognized word in memory. The idea is to use this module in conjunction with a microcontroller (ATmega8) so that the words recognized by the module be assessed by the microcontroller and later transmitted serially to the central control board before being forwarded to the motor control board. The block diagram of the voice recognition board is shown in Figure 3.15.

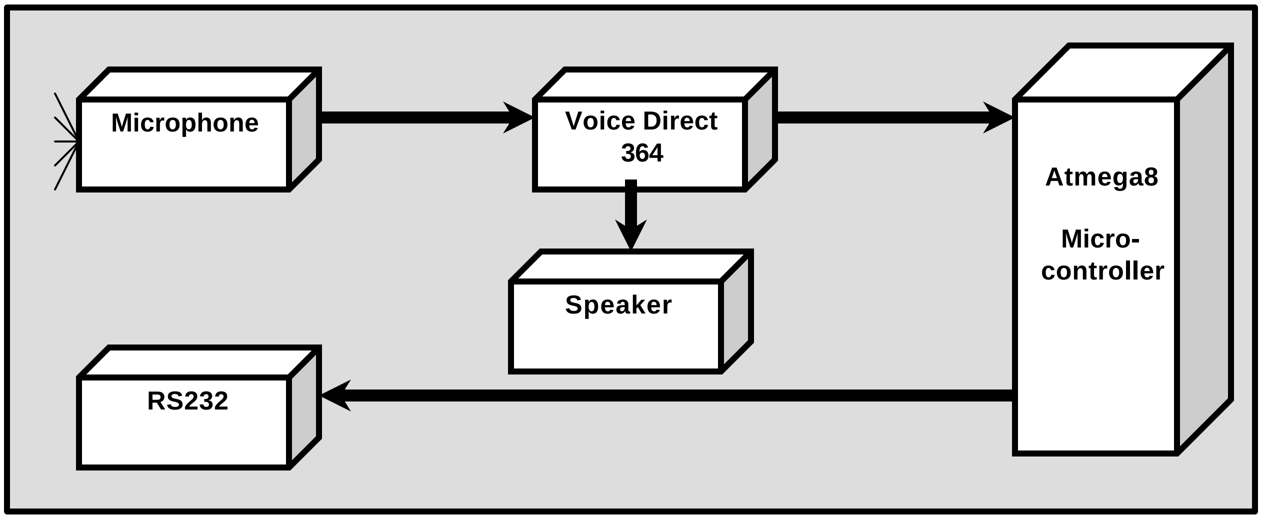


FIGURE 3.15 VOICE RECOGNITION UNIT BLOCK DIAGRAM

3.3.2.4.2 Functional description

Refer to Figure 3.16. At power ON, the LED connected to the module is ON if the module has been trained or OFF if the module has not been train yet. When the module is trained, the trained words are stored in the module's memory in the trained order. To train the module, the "train" button has to be pressed down, the module will then give a vocal prompt through the speaker saying: "Say a word".

If the word said is clear enough, the module saves it and waits for the “train” button to be pressed again. After a couple of seconds if the button is not pressed, the module will consider the training has completed.

If the “Continuous listening” button is pressed, the module will be waiting for a word to be said. If said is one of the trained words, the module will sent a signal on one of the 8 output lines corresponding to the position of the word in memory and the speaker will output it loud: “Word one” for instance.

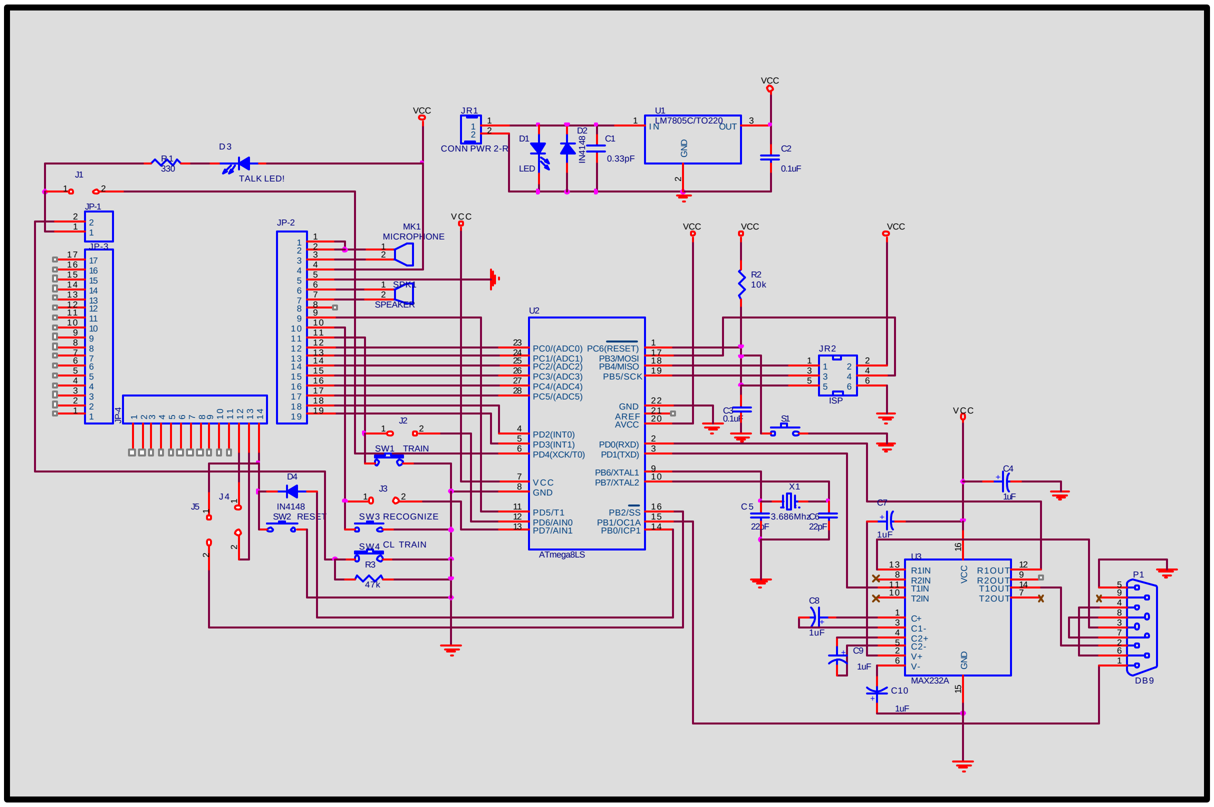


FIGURE 3.16 VOICE RECOGNITION UNIT SCHEMATIC

The words in the microcontroller are programmed in the same sequence as in the module so that when the microcontroller reads the output line from the module, it is able to identify the word that has been said. The microcontroller will then transmit the first letter of that word to the central control unit via the RS232.

The circuit has a 5V regulator on board and a 6 pins JTAG connector for programming the microcontroller on-board.

3.3.2.5 CENTRAL CONTROL CIRCUIT

The idea of introducing a central control unit in the system is due to a number of reasons critical to the proper operation of the robot. The first main reason is to have a secondary control unit that is not windows based but rather with a real time operating system in order to keep the motion control, obstacle detection and other parts of the system real time. The second main reason is that the number of ports on the PC is not enough to connect more than five different boards, hence the need of some kind of a router receiving data from different sources and directing them to the right destination.

This board expands the number of serial ports on the system making it possible for many other units to be connected. The other reason for having a central control is to avoid overloading the computer with data from all the peripheral units.

3.3.2.5.1 Design

The problem of this design was to use a single microcontroller having only one USART, to send and receive data from 8 different serial ports. The central control unit is made of a microcontroller, a multiplexer, a demultiplexer and RS232 components. The microcontroller used is the Atmel AVR Atmega16 a 40pins microcontroller with a USART and ADC conversion capabilities. The multiplexer is the 74LS151 and the demultiplexer is the 74LS138. Pin number one in the RS232 DB9 connector has been used as the interrupt line on all the serial ports. Figure 3.17 is the block diagram of the Central Control unit.

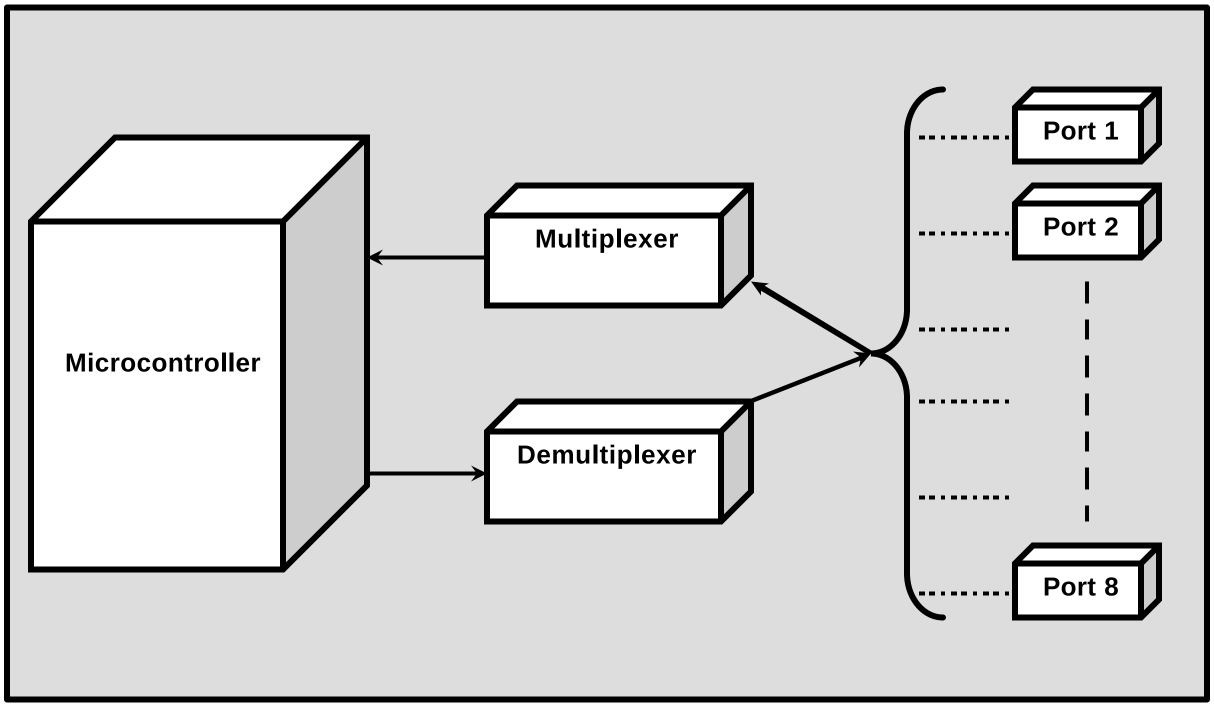


FIGURE 3.17 CENTRAL CONTROL UNIT BLOCK DIAGRAM

3.3.2.5.2 Functional Description

Figure 3.18 shows the schematic of the central control unit. This circuit has 8 serial ports to which 8 peripherals are connected. The peripherals are electronic circuits that interact with the central control circuit. The main function of the central control unit is to route all the command from the PC to the right serial port and data from any peripheral circuit to the PC or another peripheral. All the ports are numbered from 1 to 8, and as mentioned before, all the pin 1 of the DB9 connectors are used as request lines, giving peripherals the opportunity to send data to the central control or to the PC when the need arises.

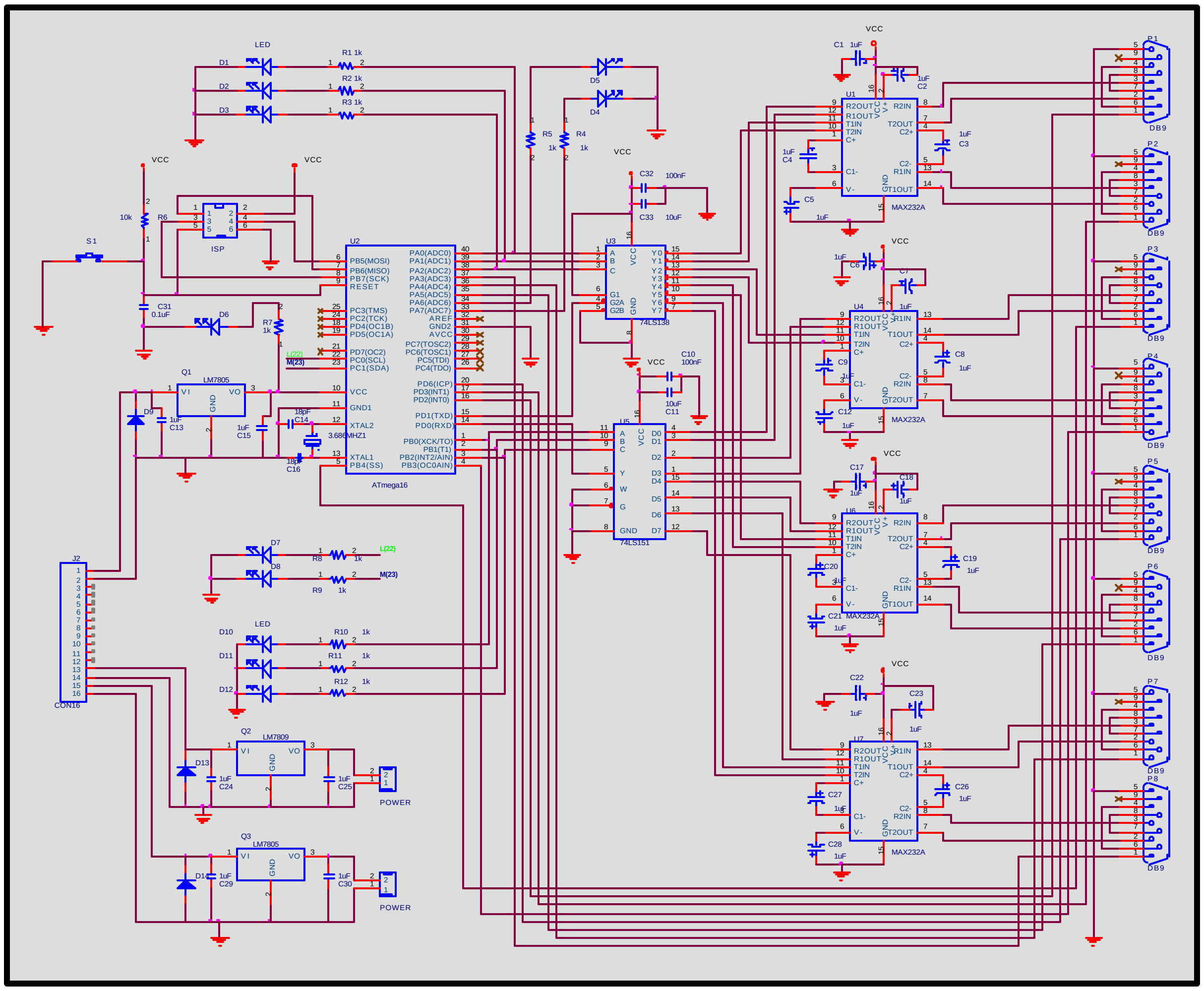


FIGURE 3.18 CENTRAL CONTROL UNIT SCHEMATIC

In fact, the microcontroller in the central control unit polls the request lines all the time with a certain priority order, and if one of the lines goes high, the microcontroller knows automatically the peripheral requesting for the transmit resources, and therefore uses the three select lines of the multiplexer to select or enable that specific port. Once the connection is established, data is transmitted from the peripheral to the central control unit's microcontroller for analysis and decision-making.

Also when the microcontroller receives a command from the PC, knowing the destination port, it uses the three select lines of the demultiplexer to select the destination port, and once the connection is established, the microcontroller transmits data to the peripheral via that specific port.

The microcontroller in the central control unit also keeps information it receives from the proximity detector unit about the current status of the environment, such that, if it receives a command to move forward while it had stored an "obstacle ahead" message from the proximity detector, it does not send such a message to the motors. This makes MGM1 obey to the command only when it does not conflict with its knowledge of the environment around it.

3.3.3 MECHANICAL

This section is about the mechanical aspect of the project. The robot needed a body of some shape in order to be useful especially as a technology demonstrator. The ideal height of the robot was specified to be ideally 0.96m, but the shape and the appearance were not specified.

Due to the scope of the work and the time allocated, it was found necessary to focus more on the implementation of the GSM functions to be demonstrated, so on the mechanical side, the idea was to build a basic frame on which the various electronics circuits and other equipments could be supported and carried. This frame can serve for experimental purposes as the first prototype, and later be improved for future versions of the robot. The actual body for MGM1 is made of two different parts:

- The main frame
- The chassis

The concept was first developed and then a tool called Solidworks was used to make the drawing of the designed part. The actual assembly or construction is then based on the drawing. The design and assembly of each of these parts will be discussed separately.

3.3.3.1 Design

3.3.3.1.1 Main Frame Design

The main frame is a 410 x 410 mm square base rising to the height of 720 mm. The structure has three different compartments topped by a cubic head of dimensions 173 x 235 x 214 mm. The idea was to provide space to store the equipment and materials constituting the payload of the robot, and being part of the robot themselves.

Considering the height of the robot, the weight distribution is critical, since it directly influences the stability of the whole structure. Therefore, the heaviest components were placed at the bottom to keep the center of gravity of the robot as low as possible. So the bottom compartment was designed to contain two lead acid batteries weighing 14Kg each and two motors, the second compartment to contain all the electronic circuits and devices, and the top compartment designed to contain the LCD screen and the other parts of the PC. The main frame designed in Solidworks is shown in Figure 3.19.

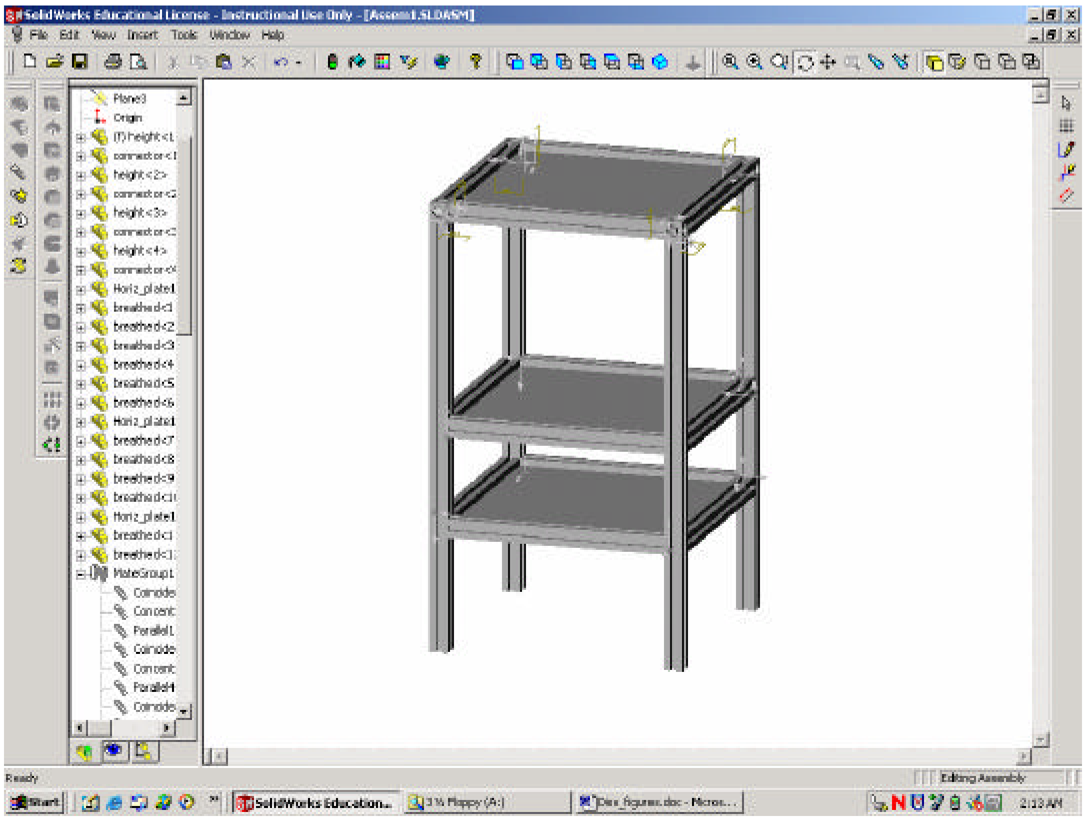


FIGURE 3.19 MAIN FRAME DESIGN

3.3.3.12 Chassis Design

The chassis is the base that carries the main frame. It is a square structure with two wheels on the rear-left and rear-right sides and two smaller wheels at the bottom-front. The big wheels on the sides are 200mm diameter, and the small wheels at the bottom are 30mm diameter. The chassis was first designed in solidworks before construction. Figure 3.20 shows the chassis design in Solidworks.

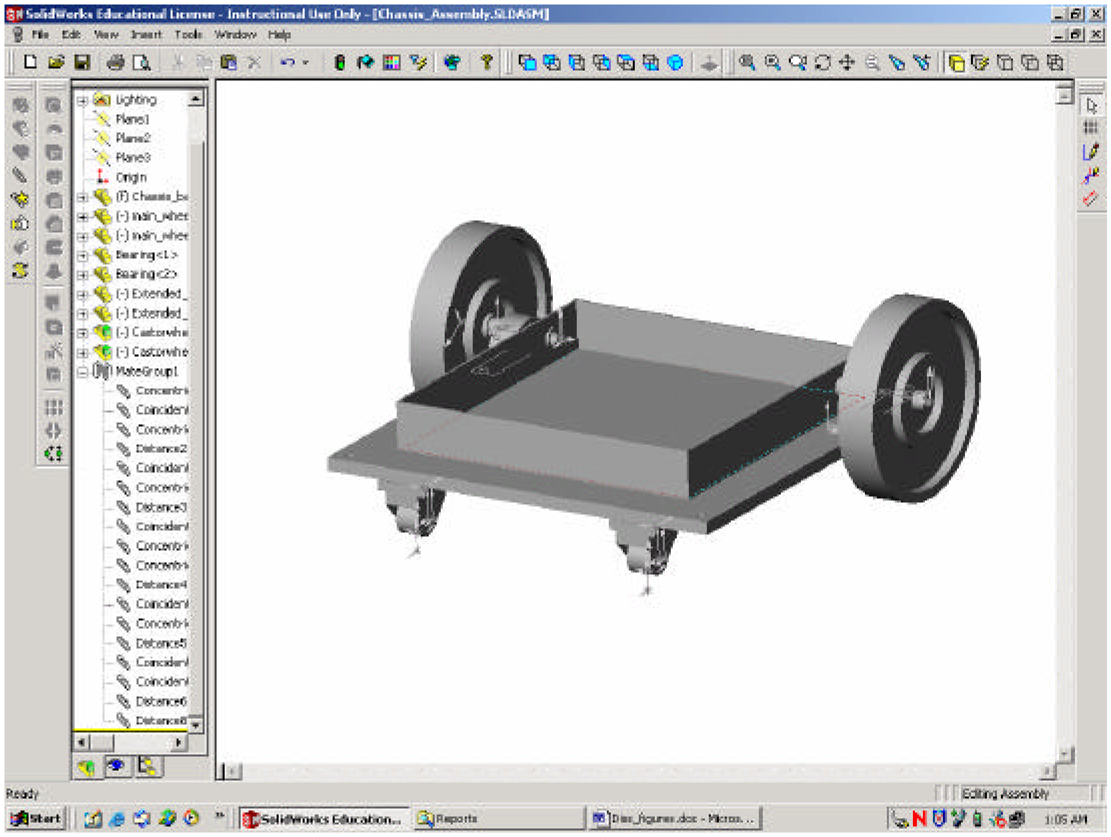


FIGURE 3.20 CHASSIS DESIGN

The expected weight of the robot was between 50 to 60Kg, main reason to build a robust chassis assembly capable of supporting a weight within that range of magnitude. The wheels used are 200mm diameter with a 20mm hole at the center for the driving shaft. The shaft of the motor is 12mm long and has a diameter of 10mm.

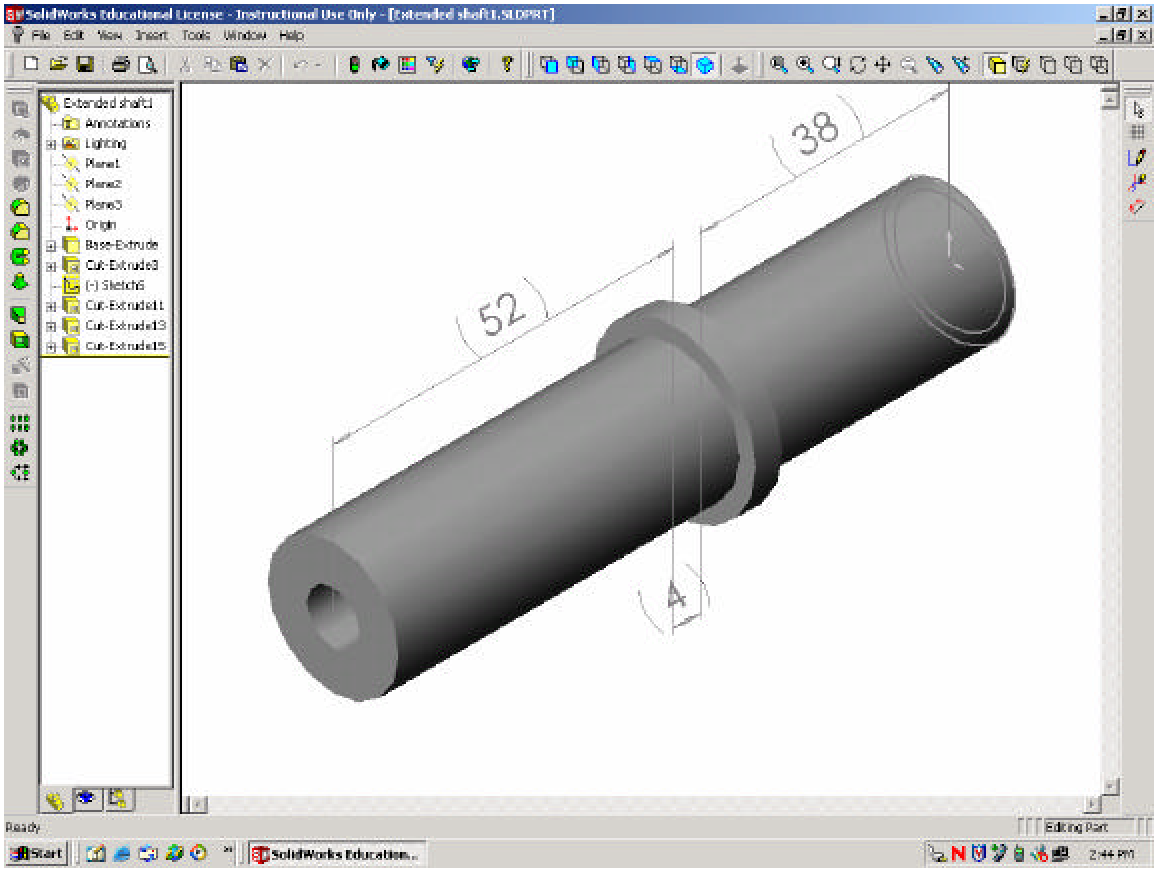


FIGURE 3.21 SHAFT EXTENDING PART

Since the hole in the wheel and the motor shaft are not compatible in size, connecting the motor to the wheel was quite a difficult exercise that needed an additional metallic piece, fitting the two sizes to be made. The fitting piece was machined such that it also be a shaft extension to compensate for the distance between the wheel and the motor through the chassis frame. Figure 3.21 shows the design of the machined shaft extension and fitting piece in Solidworks.

Two bearings were introduced to support the extended shaft against the weight of the robot, and also to reduce the wobbling in case there was a small misalignment in the fitting process.

3.3.3.2 Construction

3.3.3.2.1 Main Frame construction

The main frame is constructed using aluminum profiles from Bosch Rexroth and the 8 mm thick clear Perspex. The aluminum profiles have a square cross sectional area of 30mm x 30mm and each of the 4 sides has a 8mm groove in the middle. The Perspex fits in the profiles grooves to close the open spaces in the frame. The constructed Main frame for MGM1 is shown in Figure 3.22.

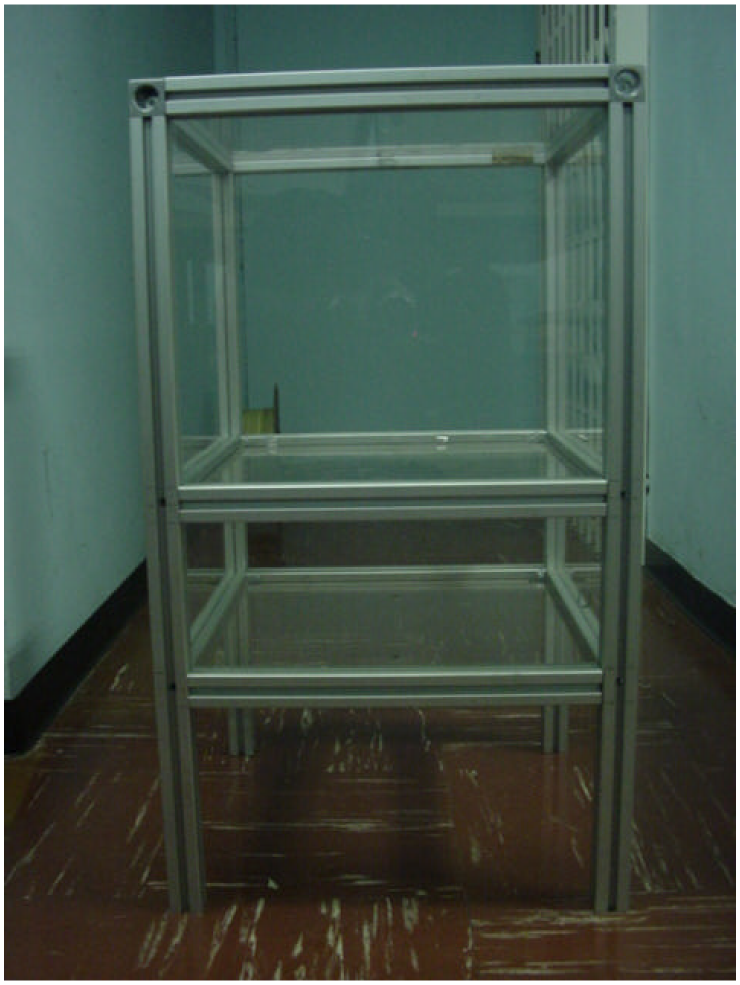


FIGURE 3.22 CONSTRUCTED MAIN FRAME

3.3.3.2.2 Chassis construction

The chassis is built using 60mm triangular steel bars, and nylon wheels. The driven wheels are positioned right at the back and the castor wheels are in front. The extended shaft of the motors passes through the bearings mounted on the triangular steel frame via two holes drilled to size. The constructed chassis is shown in Figure 3.23.

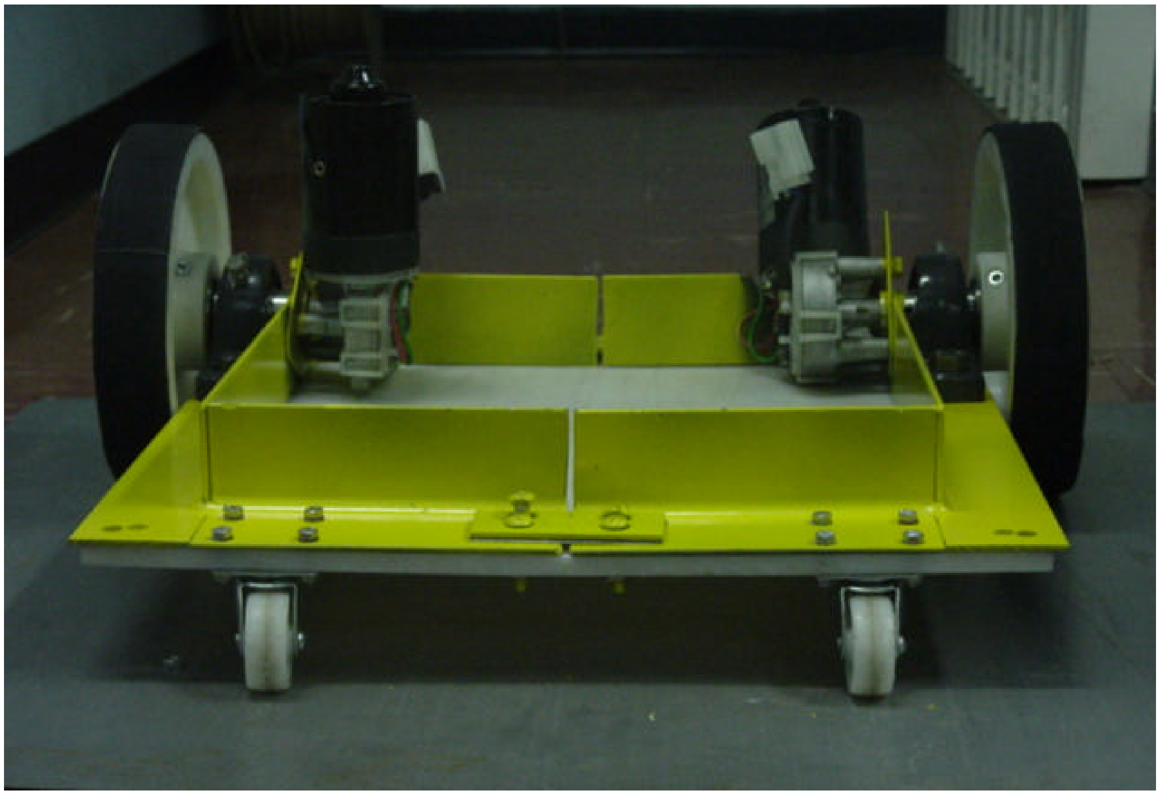


FIGURE 3.23 CONSTRUCTED CHASSIS

The complete assembly of the mechanical parts of MGM1 as designed in Solidworks is shown in Figure 3.24, and the constructed assembly is shown in Figure 3.25.

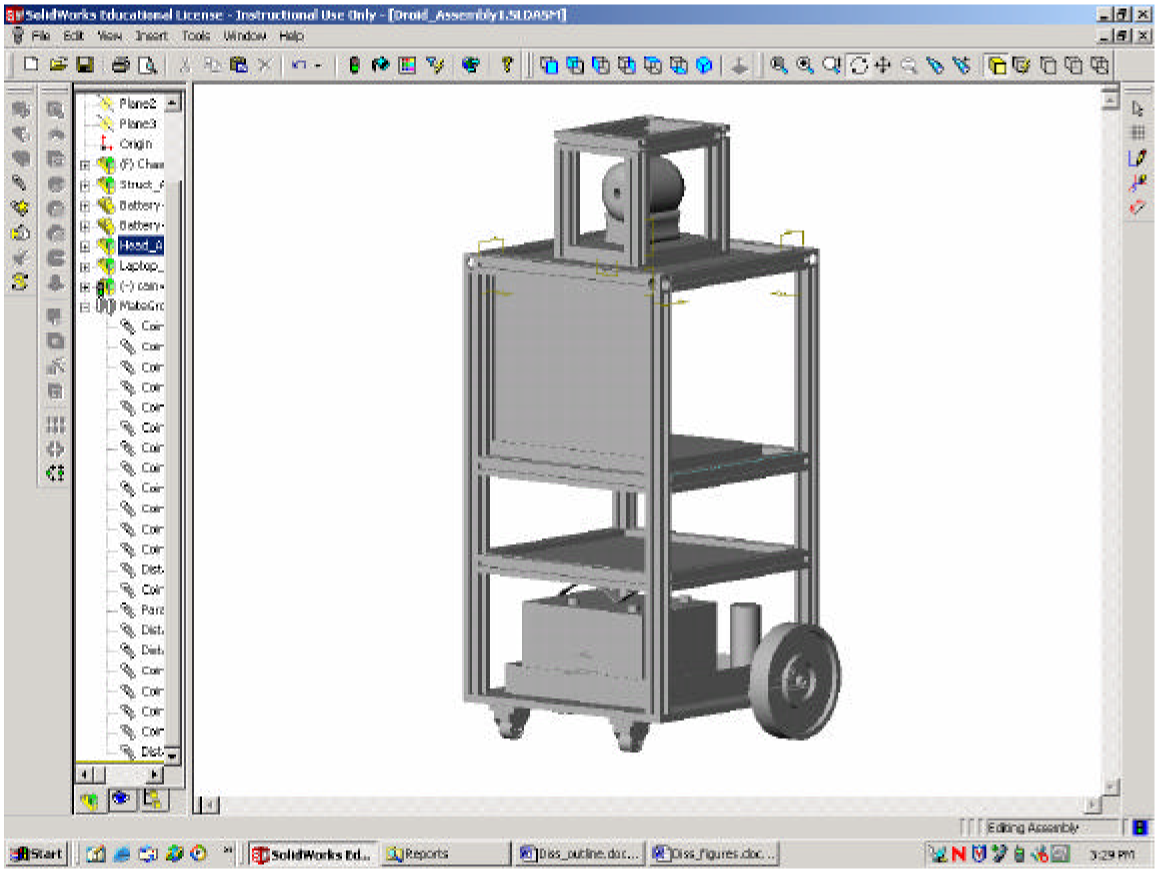


FIGURE 3.24 DESIGNED MGM1

3.3.4 SOFTWARE DESIGN

The implementation of most of the functionalities of MGM1 required some software development. From the internal architecture given in Figure 3.2, there are three levels of control and decision making in the whole system. Most of the control is done by the main processing unit (PC), which is the highest processing level. To make this possible, an application had to be developed in the PC to handle the control and decision-making. The second level of control is the central control unit. This is an intermediary level between the highest and the lowest levels. There was also a need for software at this level. The third level of control is all the peripherals and devices attached to the system.

This is the lowest level of control, where software is needed to control response mechanisms (motors), internal and external sensors and the other circuits and devices attached.

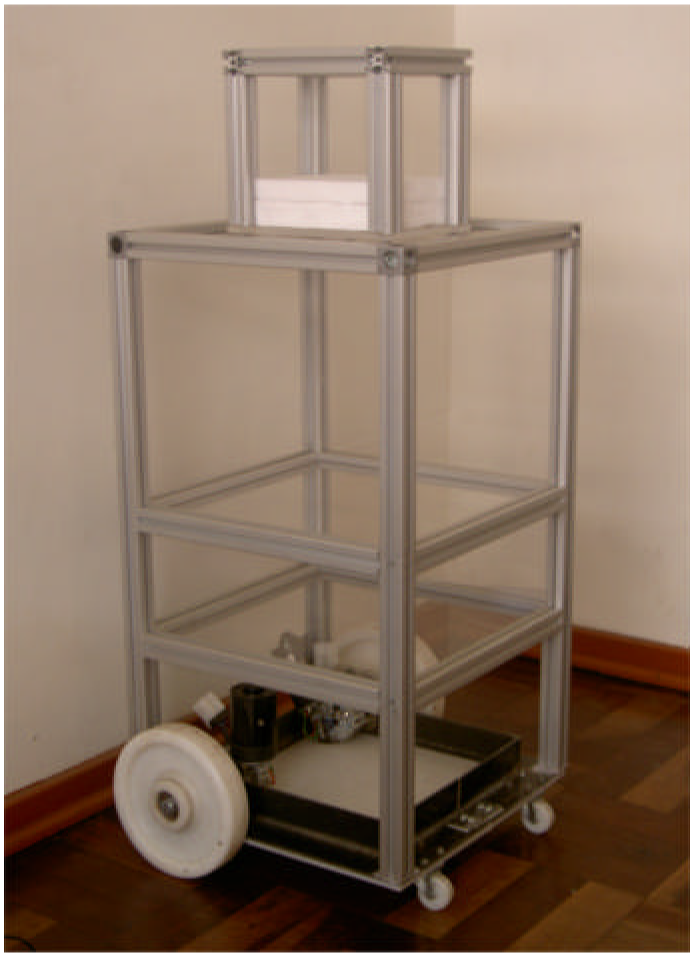


FIGURE 3.25 MGM1 ASSEMBLY

The following sections will cover the discussion of the different software developed in this project. There have been mainly two types of software developed in this project:

- A windows based application for the PC
- Embedded software for the microcontrollers

3.3.4.1 Windows based application development

The internal architecture of MGM1 in Figure 3.2, and the experimental setup in Figure 3.4, show that the GSM module is connected to the PC and other devices such as the web camera and the Bluetooth device. This means that data from these devices have to be handled and processed by the PC and the most obvious way to achieve that is to develop an application on the PC to control the flow, analyze and process the data.

An application called MGM1_ControlSoft was therefore developed using Delphi as the programming language and the IDE (Integrated Development Environment) used is Delphi 7.

3.3.4.1.1 MGM1_ControlSoft Design

By essence, MGM1_ControlSoft has to use the Com port of the PC to send and receive data. It also has to control the webcam for scanning and taking pictures. Therefore for rapid development of this application, two components were used:

- Cport 3 for the serial communication and,
- Delphi Twain for controlling the webcam

Cport 3 is a library component that takes care of the serial communication between an application (Delphi or C++ builder) and the physical serial port on the PC. Delphi Twain is another library component that enables any Delphi application to use the Twain features. Twain (Technology Without An Interesting Name) is a standard developed for the control of multimedia devices developed by a group of major manufacturers. The block diagram of MGM1_ControlSoft is given in Figure 3.26 and the flowchart is shown in Figure 3.27.

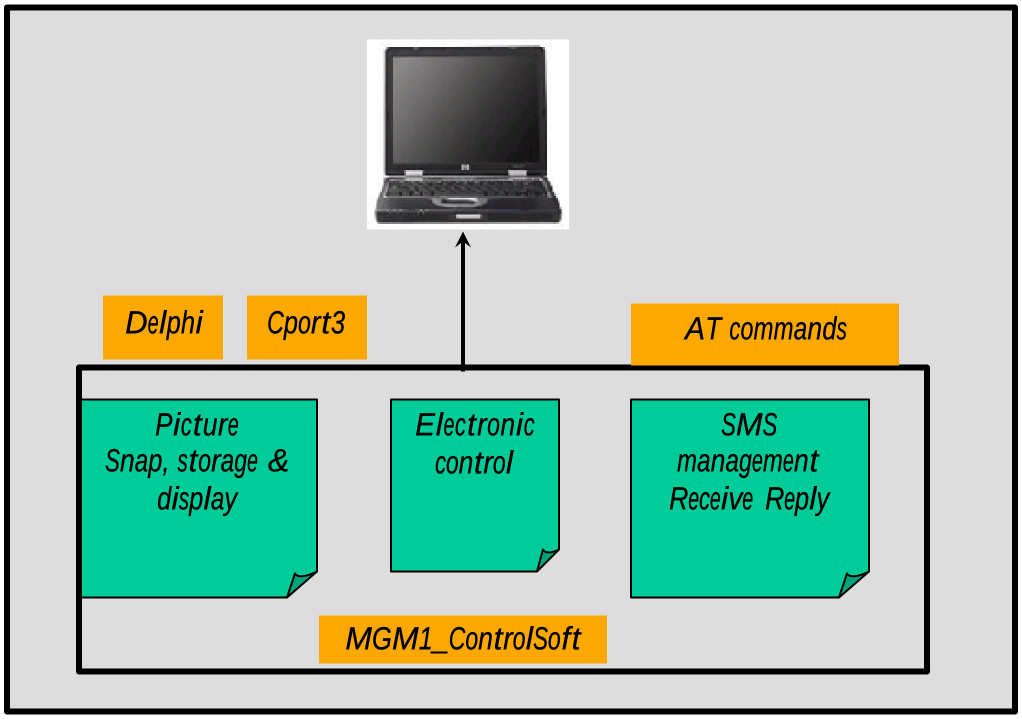


FIGURE 3.26 MGM1_CONTROLSOFT BLOCK DIAGRAM

3.3.4.1.2 *MGM1_ControlSoft functional description*

MGM_ControlSoft is an application that is launched automatically as soon as Windows setup is complete. At its start, MGM1_ControlSoft initializes the GSM module and the Webcam. The GSM module initialization consists in entering the SIM card pin code, setting the modem to text mode, and selecting the modem's memory instead of the SIM memory and the webcam is initialized in the scan mode. Immediately after initialization, the camera starts scanning and the images are displayed on a small window at the center of MGM1_ControlSoft. Then, the application is ready and in charge.

If it receives a message from a mobile phone, it saves the message, and then checks whether it is coming from the remote control phone or another phone.

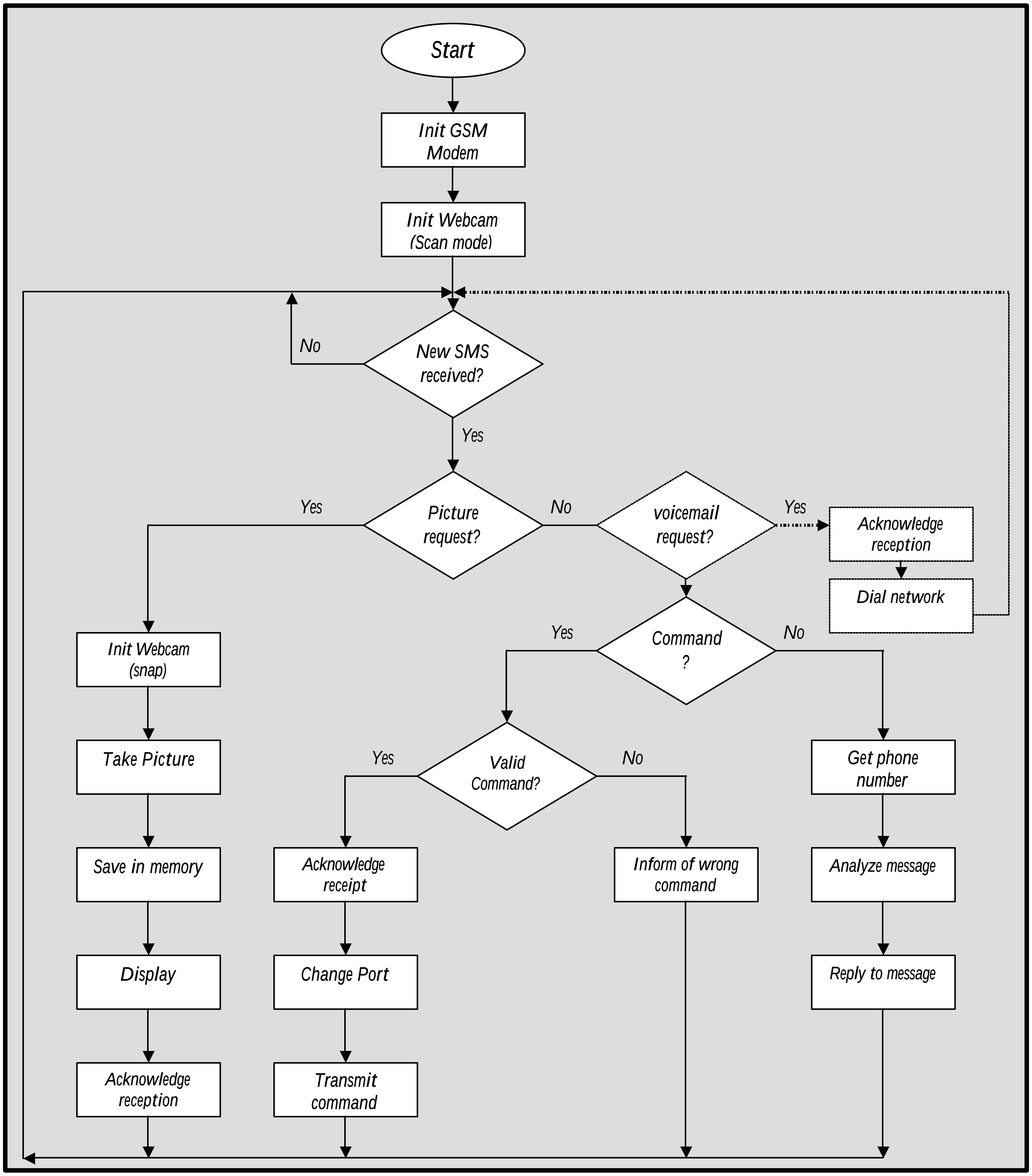


FIGURE 3.27 CONTROL_SOFT FLOWCHART

Only messages from the remote control phone are considered to be commands while others are just normal messages. In case of the received message being a command, MGM1_ControlSoft checks the validity of the command from the command list and sends back an acknowledgement message, then it transmits the message to the Central control unit for execution if the command was valid. If the received message is from any other mobile, only an acknowledgement message is sent back. There are a number of standard replies stores, such that depending on the type of message received a reply is associated.

If the received message is requesting a picture, Control_Soft first acknowledges receiving the message and then initializes the webcam in snap mode, takes a picture, saves into the current directory and then displays it on a small window as part of the main application window.

3.3.4.2 Embedded software

This section discusses the codes written for the various microcontrollers in all the electronics boards developed in this project. All the microcontrollers used are AVR microcontrollers from Atmel, all the codes are written in embedded C, and the tools used are ImageCraft and AVR Studio. The code written for each of the electronics units will be discussed separately in the following sections.

3.3.4.2.1 Central control unit code

This is the code that does the routing of the received data from the PC or an electronic circuit to the correct destination, it partially manages obstacle avoidance, and introduces priorities among the peripherals.

3.3.4.2.1.1 Design

This code is written for the ATmega 16 Atmel AVR. The aim of the code is to manage the transfer of data between the PC and the electronic peripherals using a single USART. This means that all the connected units have to share the same transmit and receive lines, and since two devices cannot transmit or receive at the same time, a regulating method had to be implemented. In this code, the unit that wants to transmit has to request the right to do so from the central control unit microcontroller.

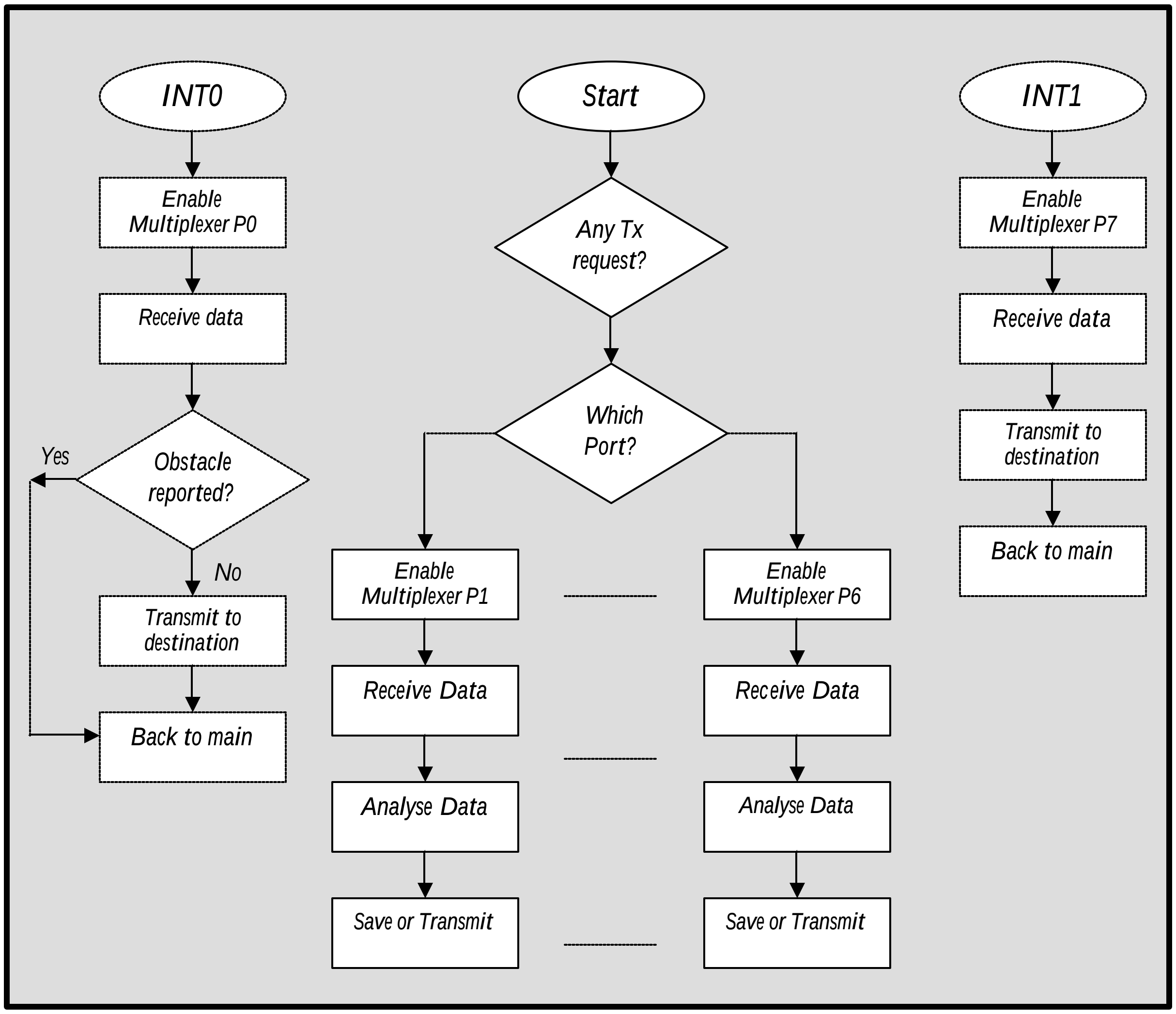


FIGURE 3.28 CENTRAL CONTROL UNIT CODE FLOWCHART

The device or the peripheral has to wait until the transmit right is granted, only then would the device be able to transmit data. The transmit priority is set by the interrupts and also by internal code. There are two interrupts in this code, INT0 and INT1. Interrupt 0 is used to give transmitting priority to the PC and Interrupt 1 is used to give the second priority to the proximity detector unit.

Since only two of the three interrupt lines provided by the microcontroller could be used, the other ports could not be put on interrupt but rather they are polled all the time in a predefined order. Figure 3.28 shows the flowchart of the central control unit code.

3.3.4.2.1.2 Functional Description

The central microcontroller gives “transmit priority” to the peripherals connected to port 0 or port 1 by enabling line 0 or line 1 of the multiplexer, when one of the interrupt lines INT0 or INT1 is triggered. This will allow one of these ports to transmit data to the central control unit. The central microcontroller analyses the received data and transmit it to another peripheral, precisely the peripheral data was addressed to. If the data is just informative, the central microcontroller makes a decision or stores it for future use.

When no interrupt line is triggered, the central microcontroller does the polling of all the other peripherals “request lines” in a certain order and gives the “transmit right” to the device requesting before it continues the polling. Here again, the central microcontroller analyses the received data and transmits it to the peripheral it has been addressed to. The central microcontroller makes a decision or stores the data if it is not addressed to a specific peripheral.

3.3.4.2.2 Motor control unit code

3.3.4.2.2.1 Design

This code is written for the motor control unit. It generates the PWM to start the motor softly, it controls the steering (turn left or turn right), as well as the direction (forward or reverse). The microcontroller used (Atmega8) has three PWM channels, but only one (PB1=OC1A) is used. Two output pins (PORTC 0 and PORTC 1) are used to control the relay switches which change the running direction of the motors. One output pin (PORTB 5) is used to control the MOSFET switch. When this switch is closed, only then that current starts flowing through the motor.

The code uses the microcontroller's USART to communicate and accepts single characters such as O (On), L (Left), R (Right), B (Backward), F (Forward), S (Stop) as commands. The analysis of the received data is done by means of a case statement. Since there is a control board for each motor, there are also two versions of this code, one for the left motor and the other for the right motor.

The difference between the two codes is mainly in the steering. In the "Turn right" subroutine, the left motor version of the code drives the left motor while the right motor version of the code keeps the right motor off. The "Turn left" subroutine the opposite happens. The left motor version of the code keeps the left motor off while the right motor version of the code drives the right motor. Therefore, only the movement of one motor at a time makes the robot turn in a certain direction depending on which motor has been active. The flowchart of the motor control unit code is shown in Figure 3.29.

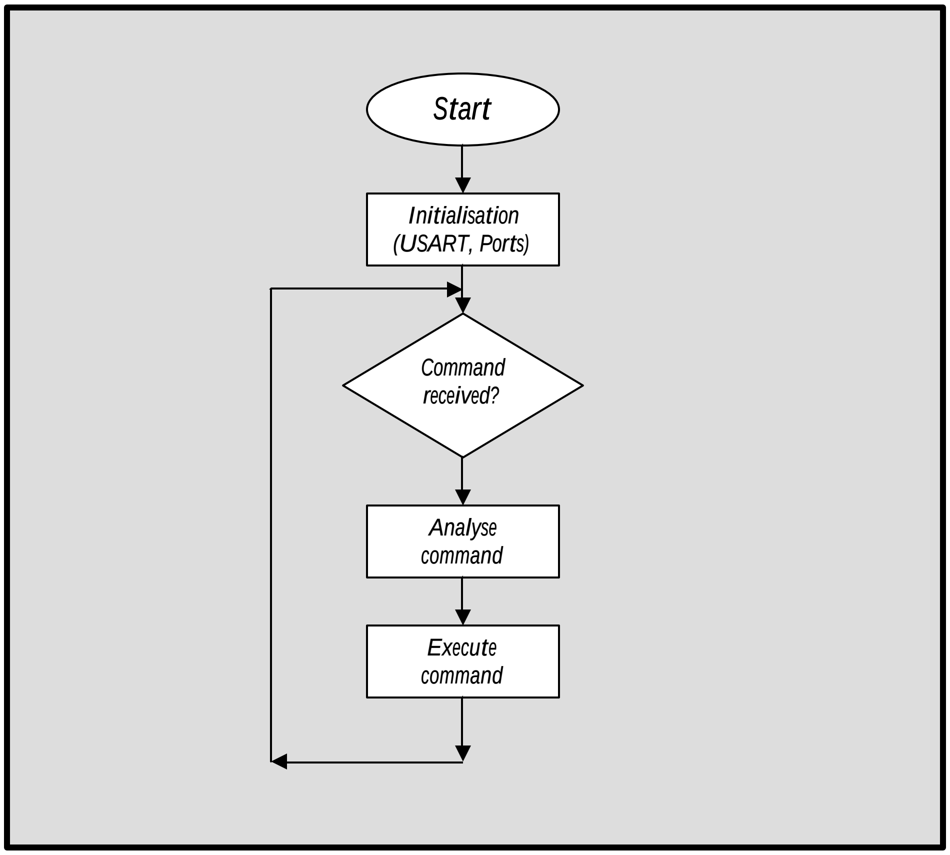


FIGURE 3.29 MOTOR CONTROL UNIT CODE FLOWCHART

3.3.4.2.2.2 Functional Description

When using the mobile phone as a remote control, one has to send the message O (On) first to switch the robot on before it starts executing the motion commands. So at power on, the code in its main routine checks first whether it has to be active or not. It does that by waiting for data from the USART and as long as the character O (On) is not received, it keeps on waiting. After receiving the character O, it starts executing motion commands such as L (Left), B (Backward) and F (Forward). The execution of each of these commands is implemented in a “Case statement” and in each case there is a “If statement” where the code checks whether it should execute the routine or not. Refer to Appendix C-C1

When the character L (Left) is received, the microcontroller on the left unit does not generate the PWM signal keeping the left motor off and the microcontroller on the right unit starts generating the PWM signal to run the right motor. When the character R (Right) is received, the microcontroller on the left unit starts generating the PWM to run the left motor and the microcontroller on the right unit does not output the PWM signal keeping the right motor off. This is how the steering of the robot is achieved in this project. If the message F (Forward) is received, the microcontrollers in both units will each generate a PWM signal to start and run both motors.

When the character B (Backwards) is received, the microcontrollers in both units will set the relays switches so that the flow of current makes the motor run on reverse. The switching signal to the relays is output on PORTC 0 and PORTC 1. When the character S (Stop) is received, the codes in both units switch to the passive mode (OFF) waiting for the character O (On) to be received again.

3.3.4.2.3 Telemetry unit code

3.3.4.2.3.1 Telemetry unit code design

This code uses the microcontroller ADC pins to collect data from the sensors. ADC0 assesses data from the voltage divider representing the battery level and ADC1 is used to assess the temperature readings from the temperature sensor LM35. The code also monitors two red LEDs indicating the status of the battery level and the temperature. The LEDs are OFF as long as the readings from the sensors are below the threshold and as soon as they go above the threshold, the LEDs go ON. The flowchart of the telemetry unit code is shown in Figure 3.30.

3.3.4.2.3.2 Functional Description

This code assesses data from the voltage divider and the temperature sensor, it does the analog to digital conversion and does the necessary calculations to get the value of the reading. It then compares the calculated value with a preset threshold value and decides on whether to warn the operator or not. The operator will only be warned if the temperature exceeds the preset value or if the battery voltage level goes below a preset value. The warning is done by transmitting data via the serial port to the central control unit, which directs it to the PC before it is sent to the GSM modem for transmission over the air.

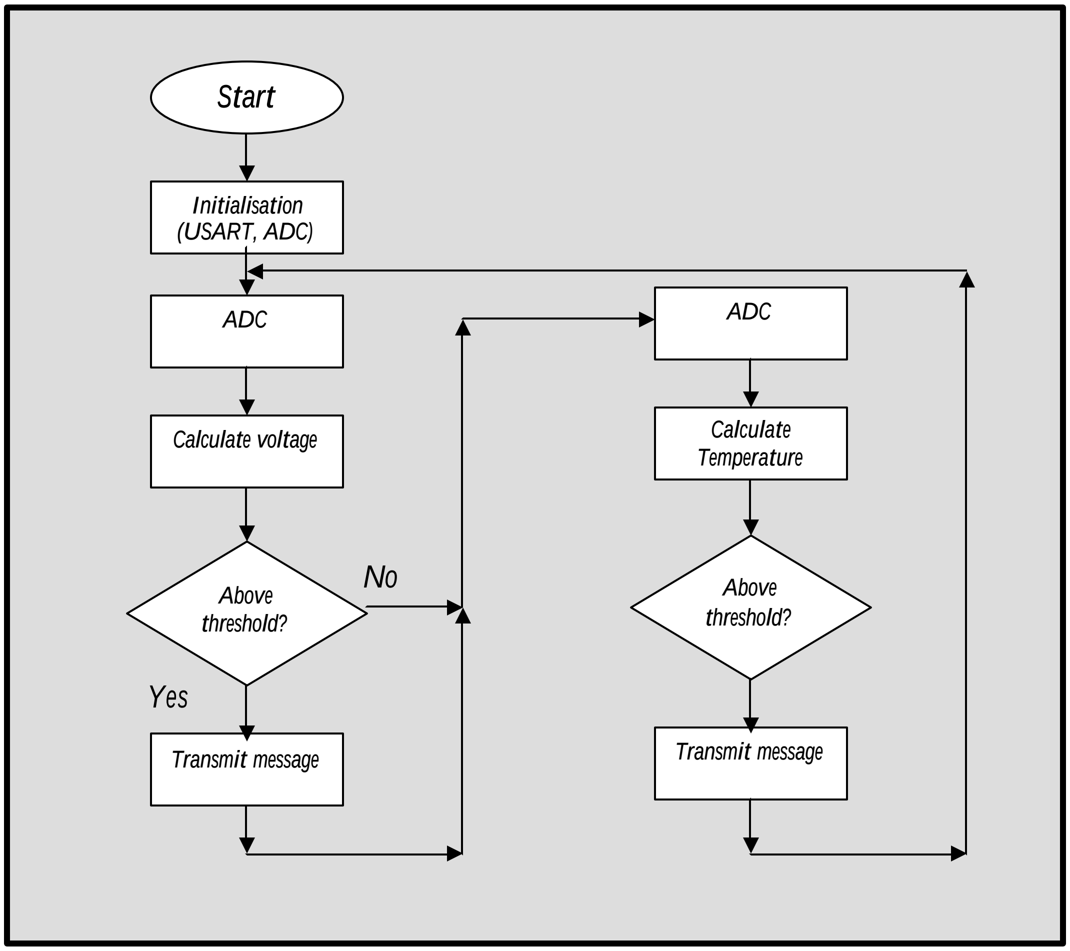


FIGURE 3.30 TELEMETRY UNIT CODE FLOWCHART

3.3.4.2.4 Proximity detection unit code

3.3.4.2.4.1 Design

This code generates a 40KHz signal to be fed to the ultrasound transmitter and analyses all the data captured by the receiving transducer to determine whether there is an obstacle within the range or not.

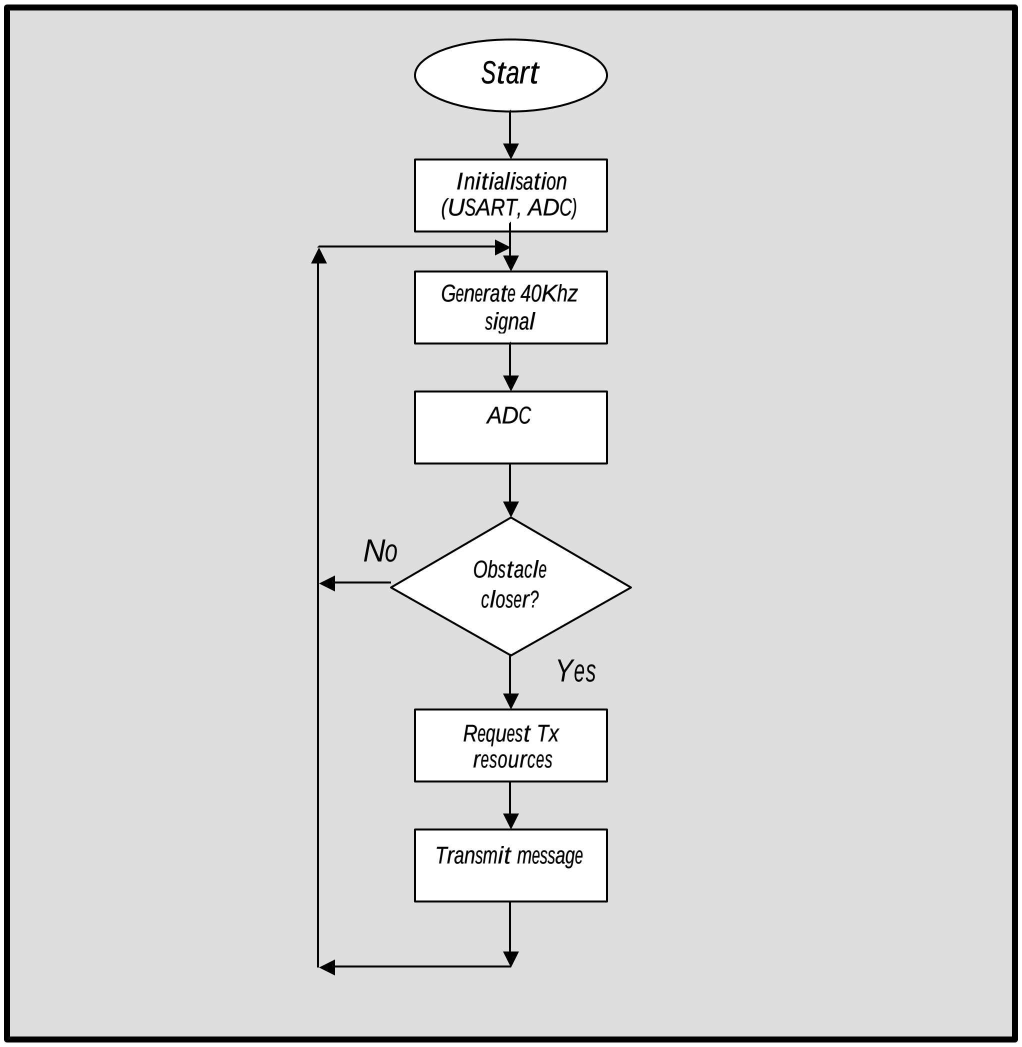


FIGURE 3.31 PROXIMITY DETECTOR UNIT FLOWCHART

The 40KHz signal is generated by making one input /output pin high for 12.5 µs then making it low for the same period of time. This generates a 50 % duty cycle signal which is fed to the circuit driving the transducer. The received data is converted into a number which is then compared with a preset threshold value before a decision is taken. The flowchart of this code is shown in Figure 3.31.

3.3.4.2.4.2 Functional Description

This code controls both the transmitter and the receiver circuits. It first generates the pulse signal to drive the transmitter and then analyses the signal from the receiver circuit. The code samples the signal from the peak detector and then converts it into a voltage level. The obtained voltage level is then compared with a preset threshold value. If the reading is above the threshold value then the obstacle is very close, no move toward that direction will be taken. If the reading is not above the threshold value, then the microcontroller checks how far is the obstacle, from the threshold line. This allows the microcontroller to keep track of the obstacles position.

3.3.5 PROGRAMMING AND PROGRAMMING TOOLS

3.3.5.1 Embedded programming

There is code written for each of the microcontrollers. After assembling the boards, the codes were loaded into the respective microcontrollers. The microcontrollers were programmed serially in system using the Atmel STK500 development board. The STK500 is shown in Figure 3.32.

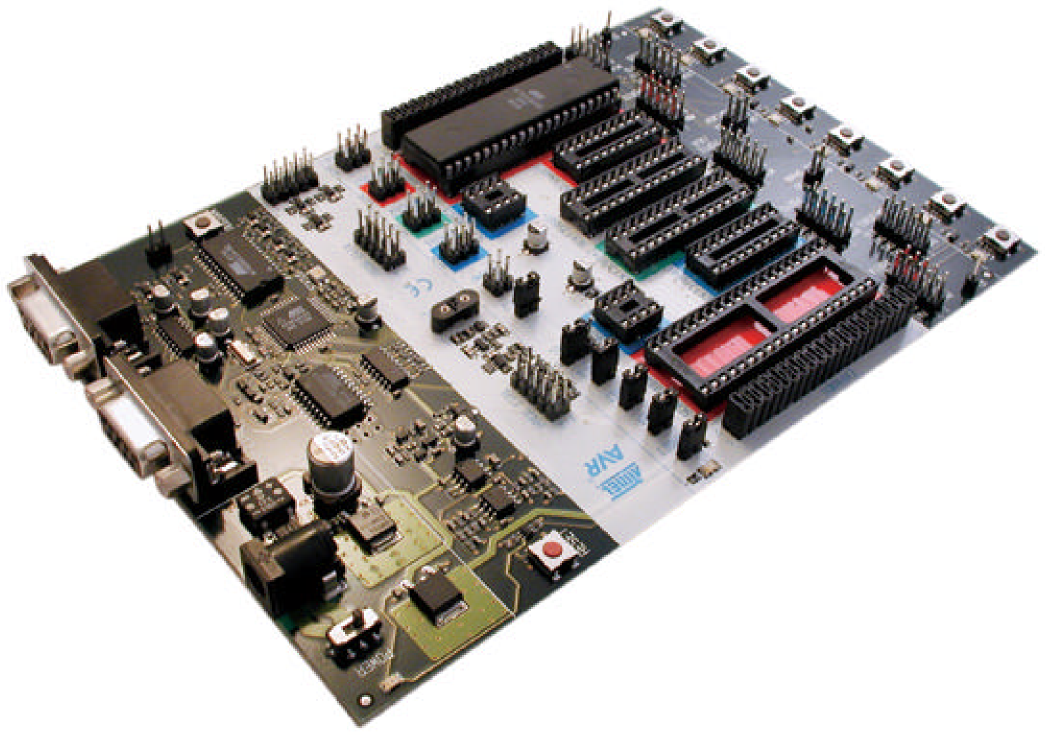
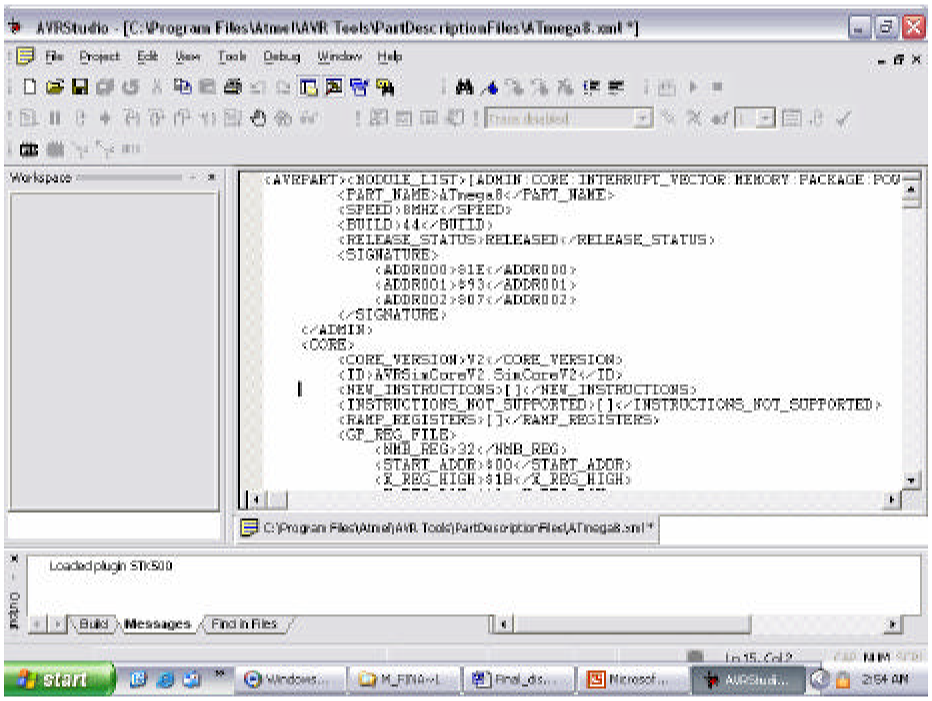


FIGURE 3.32 STK500 DEVELOPMENT BOARD

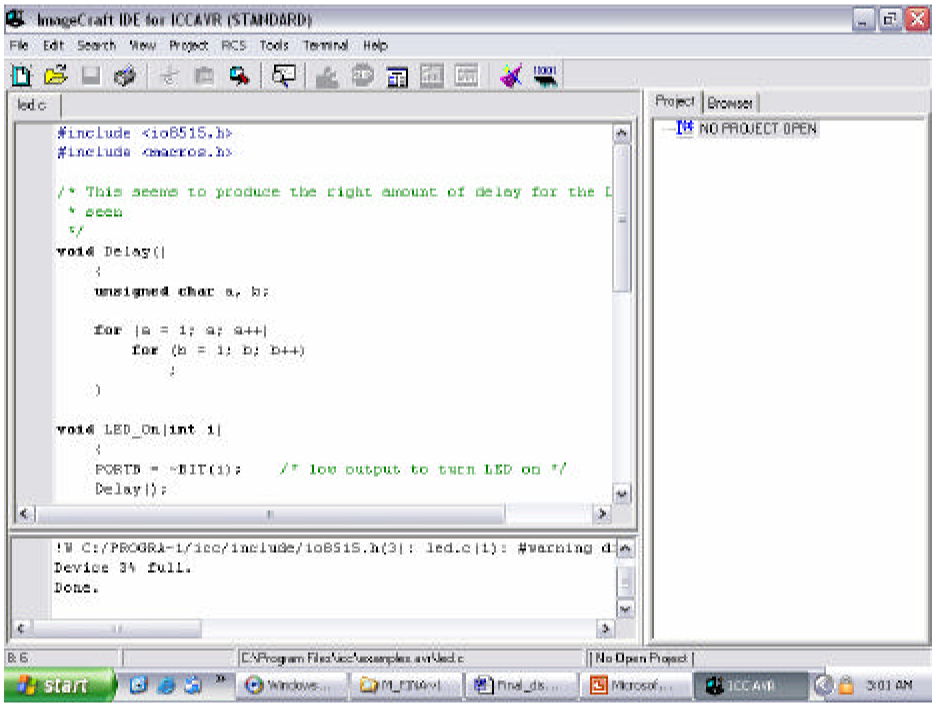
All the embedded software has been written in C. Two applications were used as development environment, ImageCraft and AVR studio 4. ImageCraft was used as an editing and compiling environment, the used version could not debug nor program microcontrollers in system. AVR studio 4 could not compile C code, it was used only as debugging and programming environment. So the code was written and compiled in ImageCraft and then exported to AVR studio 4 for debugging and programming. Figure 3.33 shows the AVR Studio graphic user interface and Figure 3.34 shows the ImageCraft graphic user interface.


AVRStudio - [C:\Program Files\Atmel\AVR Tools\PartDescriptionFiles\ATmega8.xml *]
File Project Edit View Tools Debug Window Help
Workspace
C:\Program Files\Atmel\AVR Tools\PartDescriptionFiles\ATmega8.xml *
Build Messages Find in Files
Ln 15, Col 2
start

```
<AVRPART><MODULE_LIST>[ADMIN:CORE:INTERRUPT_VECTOR:MEMORY:PACKAGE:POW
        <PART_NAME>ATmega8</PART_NAME>
        <SPEED>8MHZ</SPEED>
        <BUILD>44</BUILD>
        <RELEASE_STATUS>RELEASED</RELEASE_STATUS>
        <SIGNATURE>
            <ADDR000>$1E</ADDR000>
            <ADDR001>$93</ADDR001>
            <ADDR002>$07</ADDR002>
        </SIGNATURE>
    </ADMIN>
    <CORE>
        <CORE_VERSION>V2</CORE_VERSION>
        <ID>AVRSimCoreV2.SimCoreV2</ID>
        <NEW_INSTRUCTIONS>[]</NEW_INSTRUCTIONS>
        <INSTRUCTIONS_NOT_SUPPORTED>[]</INSTRUCTIONS_NOT_SUPPORTED>
        <RAMP_REGISTERS>[]</RAMP_REGISTERS>
        <GP_REG_FILE>
            <NMB_REG>32</NMB_REG>
            <START_ADDR>$00</START_ADDR>
            <X_REG_HIGH>$1B</X_REG_HIGH>
```

```
Loaded plugin STK500
```

FIGURE 3.33 ATMEL AVR STUDIO WINDOW


ImageCraft IDE for ICCAVR (STANDARD)
File Edit Search View Project RCS Tools Terminal Help
led.c
Project Browser
NO PROJECT OPEN
8:6
C:\Program Files\icc\examples.avr\led.c
No Open Project
start

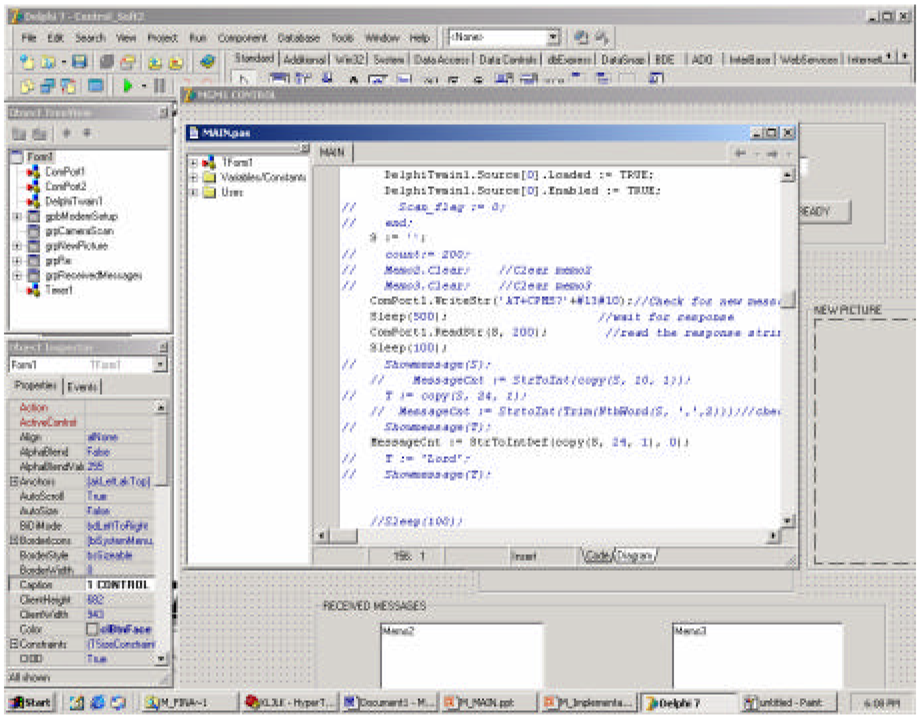
```
#include <io8515.h>
#include <macros.h>

/* This seems to produce the right amount of delay for the L
 * seen
 */
void Delay()
    {
    unsigned char a, b;

    for (a = 1; a; a++)
        for (b = 1; b; b++)
            ;
    }

void LED_On(int i)
    {
    PORTB = ~BIT(i);    /* low output to turn LED on */
    Delay();
```

```
!W C:/PROGRA~1/icc/include/io8515.h(3): led.c(1): #warning d
Device 3% full.
Done.
```

FIGURE 3.34 IMAGECRAFT WINDOW

3.3.5.2 Computer programming

The computer programming consisted in developing a graphic user interface for MGM1 screen. The application developed is called MGM1_ControlSoft. It has been developed in Delphi 7 environment. Delphi 7 window is illustrated in Figure 3.35.



```
    DelphiTwain1.Source[0].Loaded := TRUE;
    DelphiTwain1.Source[0].Enabled := TRUE;
//      Scan_flag := 0;
//    end;
  S := '';
//    count:= 200;
//    Memo2.Clear;     //Clear memo2
//    Memo3.Clear;     //Clear memo3
  ComPort1.WriteStr('AT+CPMS?'+#13#10);//Check for new mess
  Sleep(500);                      //wait for response
  ComPort1.ReadStr(S, 200);         //read the response stri
  Sleep(100);
//    Showmessage(S);
  //     MessageCnt := StrToInt(copy(S, 10, 1));
//    T := copy(S, 24, 1);
  //  MessageCnt := StrtoInt(Trim(NthWord(S, ',',2)));//che
//    Showmessage(T);
  MessageCnt := StrToIntDef(copy(S, 24, 1), 0);
//    T := 'Lord';
//    Showmessage(T);


  //Sleep(100);
```

Action	
ActiveControl	
Align	alNone
AlphaBlend	False
AlphaBlendVal	255
Anchors	[akLeft,akTop]
AutoScroll	True
AutoSize	False
BiDiMode	bdLeftToRight
BorderIcons	[biSystemMenu,
BorderStyle	bsSizeable
BorderWidth	0
Caption	1 CONTROL
ClientHeight	682
ClientWidth	943
Color	clBtnFace
Constraints	(TSizeConstraint
Ctl3D	True

FIGURE 3.35 DELPHI 7 WINDOW

3.3.6 STEERING SYSTEM DESIGN AND SIMULATION

The steering of a wheeled robot can be achieved in many different ways. The internal architecture of MGM1 influences the steering method a lot. In this section two of the possibilities considered will be discussed.

The main aim is to find a good way of making the robot move straight, turn left, right or around when instructed to do so. Therefore, considering a four wheeled robot having two driving wheels and two castor wheels, two different methods were retained for close examination. These two methods of steering have been studied and simulated in the following sections.

3.3.6.1 Path modelling and simulation

This section is going to focus on the control of the motors. The aim is to find the best signals to control the motors according to their characteristics and to predict the path of the robot. It will be considered that the variations of the speed are slow enough such that the transient region of the motors is negligible.

Considering only the two main wheels of the robot, the left wheel may be denoted M1 and the right wheel M2. If V1 and V2 are considered to be their respective translation speeds and *d* the space between the wheels, then the infinitesimal path of the robot can be illustrated by Figure 3.36.

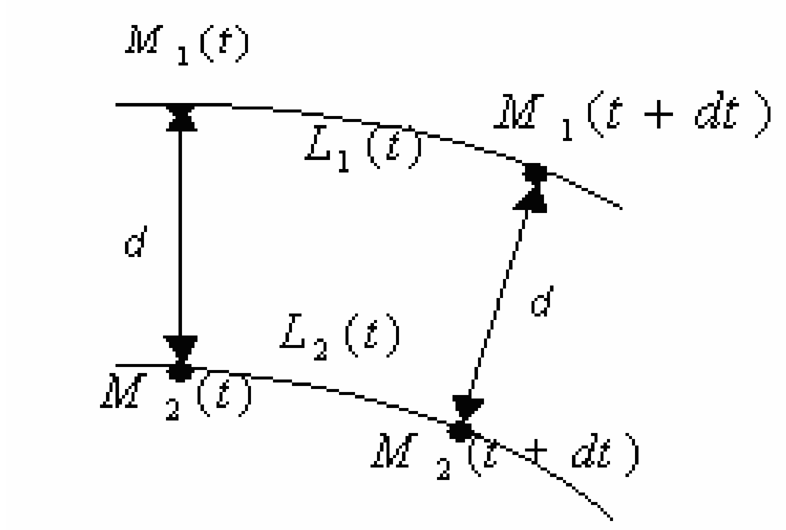


FIGURE 3.36 INFINITESIMAL PATH OF THE ROBOT

Let us denote *b(t)* the angle of rotation between *t* and *t+dt*. If we approximate $L_1(t)$ and $L_2(t)$ by straight lines, using trigonometric relations, the rotation angle can be expressed by:

$$b(t) = Arc\sin(\frac{|L_1(t) - L_2(t)|}{d}) \approx \frac{|L_1(t) - L_2(t)|}{d} \qquad (3.1)$$

Equation 3.1 allows the updating of the positions of the wheels if we know $L_1(t)$ and $L_2(t)$.

On the other hand, without assuming any shape for the path, the modelling of an α degrees turn gives

$$\int_0^T b(t)dt = a \qquad (3.2)$$

where, *T* is the time to execute the turn.

3.3.6.1.1 Speed profiles for a 90° turn

To make the robot turn, the designer can adjust the values of the speed for the motors. A BIBO system excited by a signal will converge to a solution after a period of time. The behaviour of the system during this period is called transient region and is different from the stationary solution.

Equation 3.3 gives the value of the speed for a 90° turn with constant speed for the two wheels:

$$\begin{cases} v_1(t) = v_1 \\ v_2(t) = v1 - \dfrac{pd}{2T} \end{cases} \qquad (3.3)$$

To achieve this profile, the motor M2 would be exited by a non-continuous signal at *t=0* (beginning of the turn) and *t=T* (end of the turn). It will result in the appearance of a transient region. To avoid this phenomenon, the speed of the wheels of the robot will be at least continuous functions (Condition 1) and could also have its derivative continuous (Condition 2). We are going to investigate these conditions through two profiles denoted linear variations (Equation 3.4, Figure 3.37a) and sinusoidal variation (Equation 3.5, Figure 3.37b). The linear variation complies with the first condition whereas the sinusoidal variation complies with both Condition1 and Condition2.

Linear variation (Condition 1):

$$\begin{cases} v_1(t) = v_1 \\ v_2(t) = v1.(1 - \frac{2t}{T}) \end{cases} \quad \text{for } t<T/2 \qquad (3.4)$$

$$\begin{cases} v_1(t) = v_1 \\ v_2(t) = v1.(\frac{2t}{T} - 1) \end{cases} \quad \text{for } t>T/2 \qquad (3.5)$$

Sinusoidal variations (Condition 1+ Condition 2) :

$$\begin{cases} v_1(t) = v_1 \\ v_2(t) = \frac{v_1}{2}.(1 + \cos(\frac{p}{T} t)) \end{cases} \qquad (3.6)$$

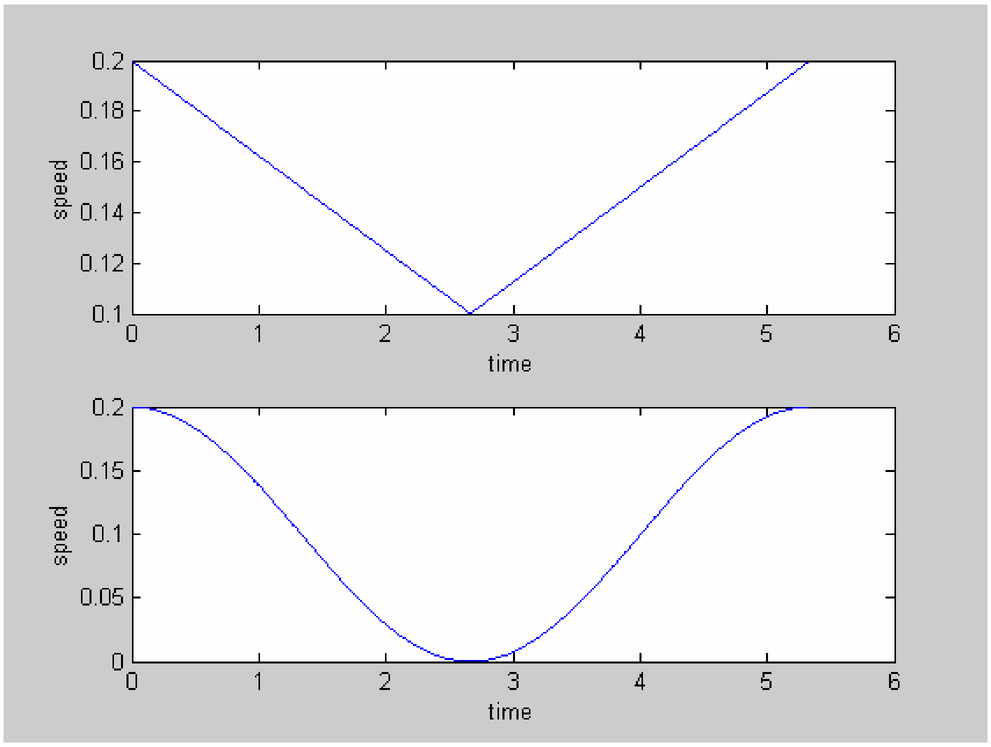


FIGURE 3.37 A) LINEAR SPEED PROFILE B) SINUSOIDAL SPEED PROFILE

The two following paragraphs are focusing on two parameters, which can lead to the choice of a speed profile: time for a turn and space required for this turn.

Time for a turn

The solution of the time for a 90° turn is given by Equation 3.7, where $|L_1(t) - L_2(t)| << d$. Therefore:

$$\int_0^{T/2} Arc\sin(\frac{|L_1(t)-L_2(t)|}{d})dt = \frac{p}{4} \qquad (3.7)$$

For the linear variation, the result is $T_{linear} = \frac{dp}{v_1}$

For the sinusoidal variation, the result is: $T_{\text{sinusoidal}} = \frac{dp}{v_1} \cdot \frac{p}{p-2}$

The linear variation leads to a turn at least two times quicker than sinusoidal variation.

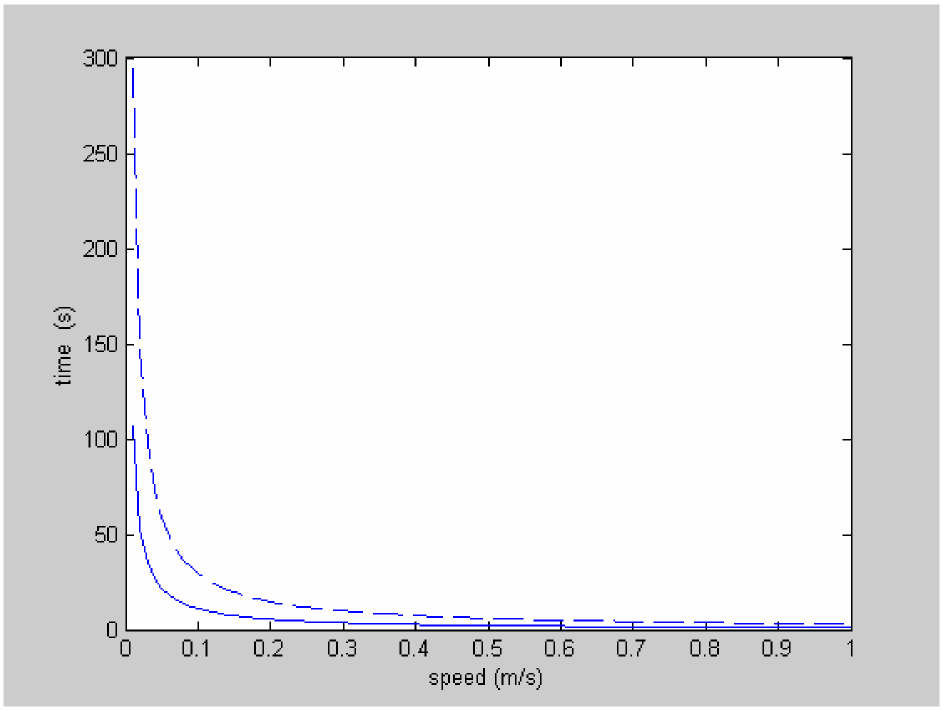


FIGURE 3.38 TIME FOR A 90° TURN FOR LINEAR VARIATIONS (PLAIN) AND SINUSOIDAL VARIATIONS (DASHED)

Path of the robot

Using the equations presented in the mathematical model, it is possible to simulate the path of the two wheels. In this simulation, the wheel spacing is 34cm and the initial speed 0.2 m/s.

From the analysis of Figure 3.39 and 3.40, we can see that the space required for a turn is twice as much using a linear variation than a sinusoidal variation.

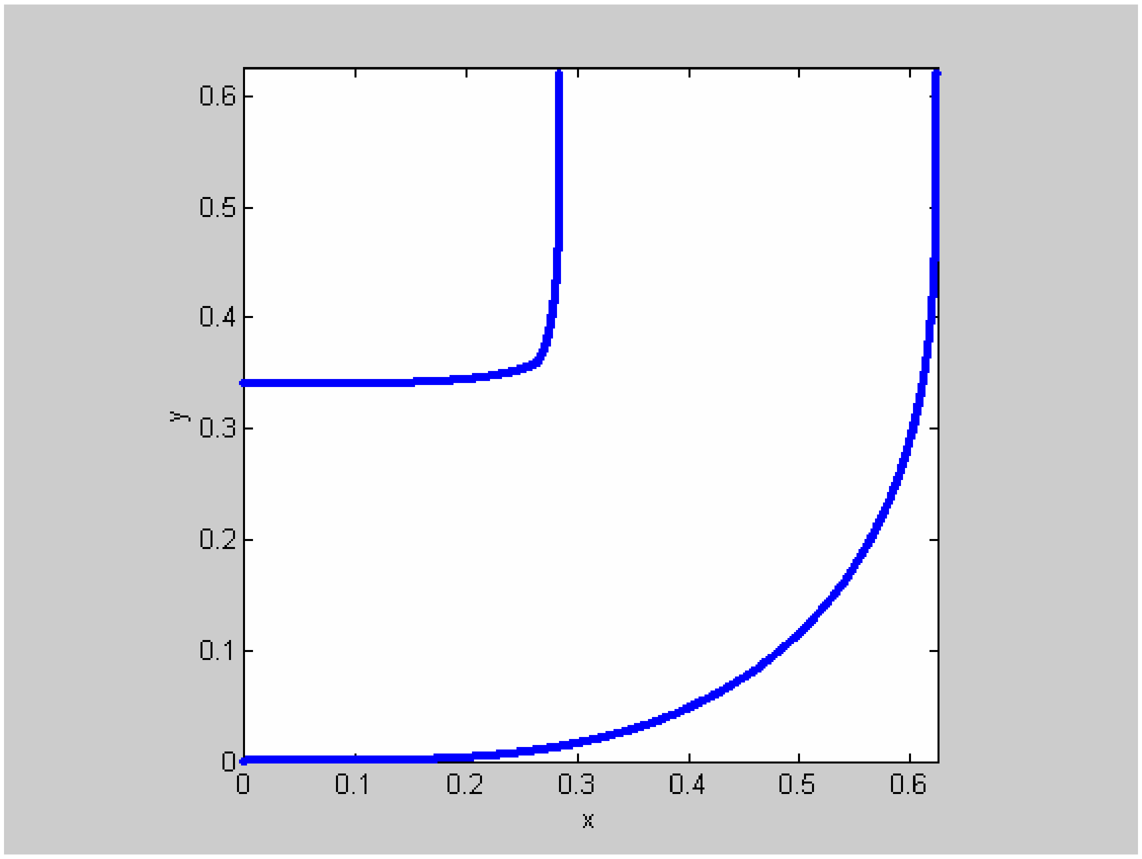


FIGURE 3.39 WHEELS PATH FOR THE SINUSOIDAL VARIATION

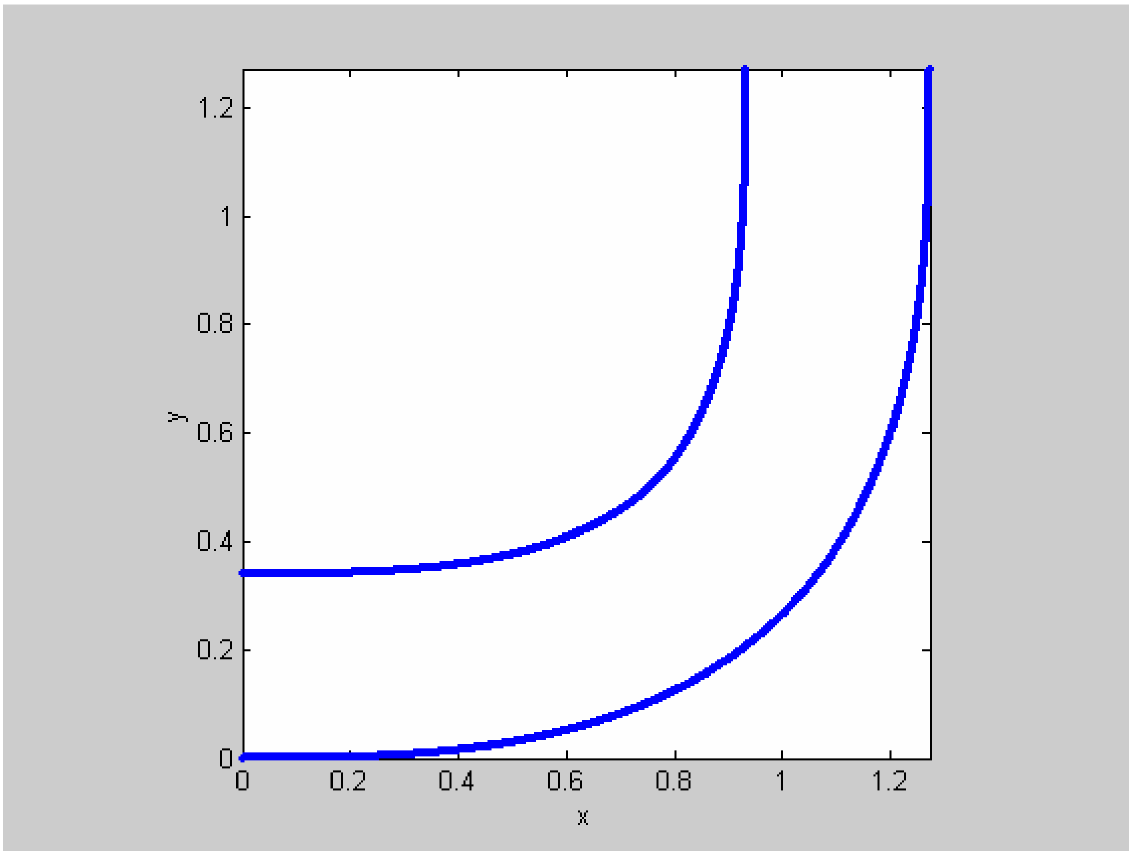


FIGURE 3.40 WHEELS PATH FOR THE LINEAR VARIATION

CHAPTER FOUR

TESTS AND RESULTS

"The ultimate result is the most obvious scale…"

"Work sown…result harvest…"

4.1 INTRODUCTION

This chapter covers the integration of the different building blocks, and the main tests of the system as initiated to validate the main objective of the project. The final results are also presented and discussed.

The system has been designed as already discussed, and each unit designed was tested as a stand-alone unit, and all the working units were then integrated together into a full system. Integration is one of the most interesting and difficult parts of any project, in the sense that it reveals a lot of crucial issues and sensitive aspects necessary for the operation of the system.

4.2 DESIGN OUTPUTS

The electronics design resulted in a number of PCBs and electronic cards. In this section, the different boards will be discussed and displayed.

4.2.1 MOTOR CONTROL BOARDS

In the final prototype called MGM1, there are two identical motor control boards. The difference between the two is in the software implemented. After design, simulation and implementation, the motor control board produced is shown in Figure 4.1.

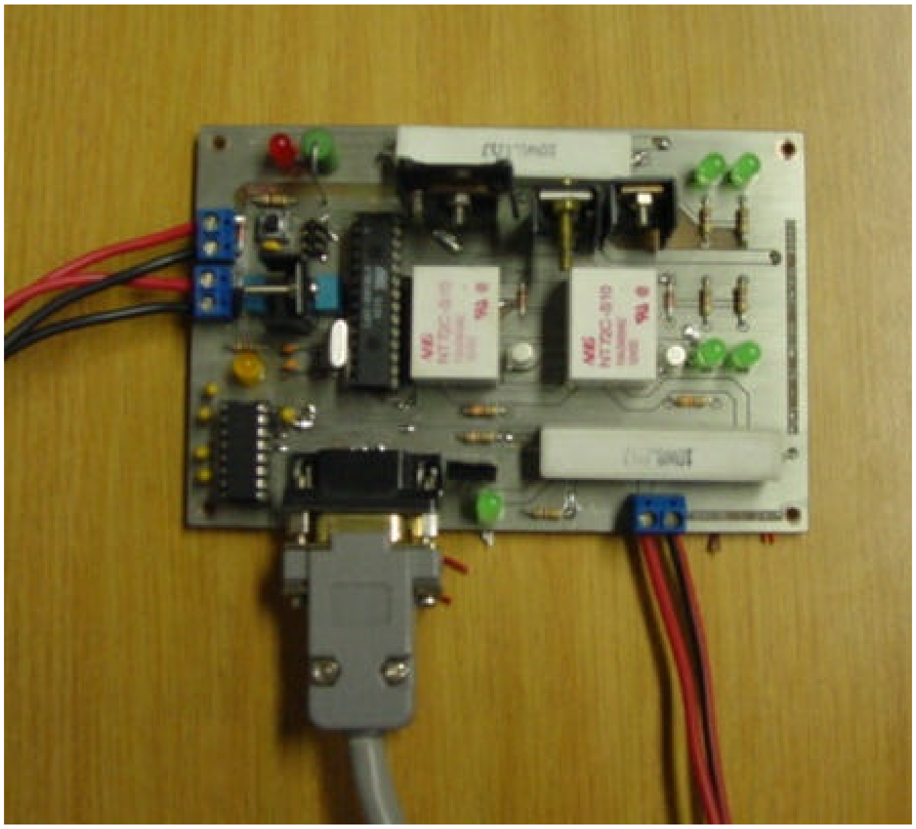


FIGURE 4.1 MOTOR CONTROL PCB

During testing, the PWM signal generated by the microcontroller ATmega8 to drive the relay switching circuit was measured. The measurements were taken on pin 11 of the microcontroller which is connected to the base of the two transistors switching the relays. The purpose of this signal is mainly to avoid brutal motor start or stop. With this signal the motor can start slowly and gradually increase the speed until it reaches its full speed. Figure 4.2 shows the measured signal at its start and at a later stage with a higher duty cycle.

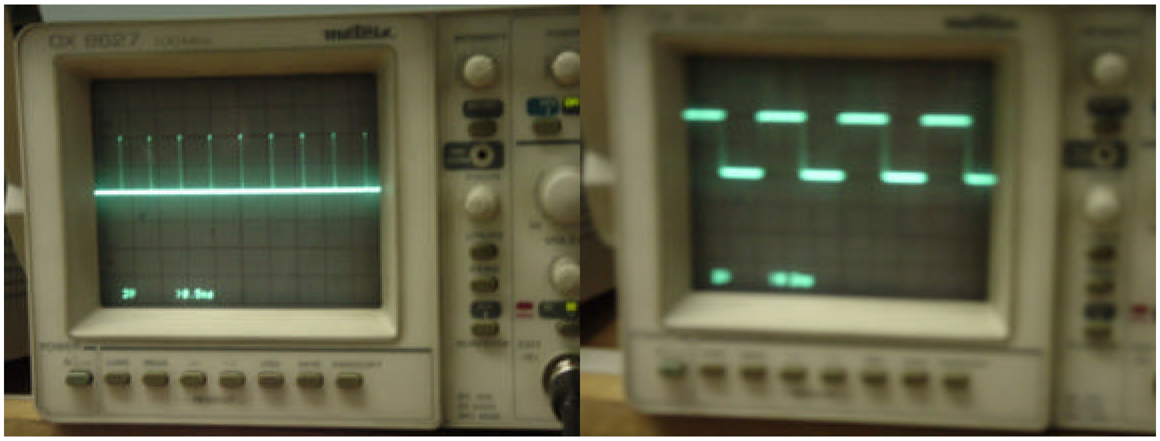


FIGURE 4.2 PWM SIGNAL DRIVING MOTORS

The board performance has been assessed practically. The test consisted in connecting the board to a computer and to a motor. By typing characters from the keyboard as commands, the circuit could turn the motor in forward and reverse directions.

PROXIMITY DETECTOR BOARD

The proximity detector board is an independent unit in the sense that it doesn't receive commands, but rather performs obstacle detection and then reports about the outcome of its task to the central control. The PCB of this unit is the one shown in the Figure 4.3.

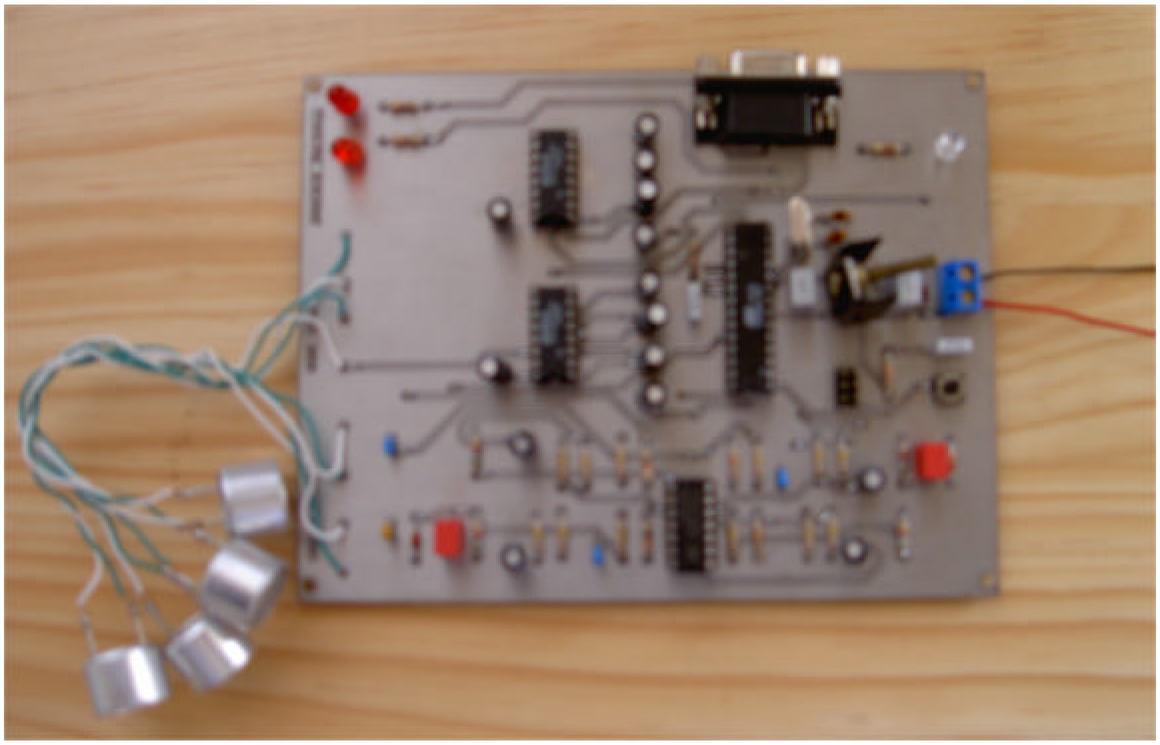


FIGURE 4.3 OBSTACLE DETECTION BOARD

When an object is found within the range of one meter, the transmitted signal will reflect back to the receiver. In this case, the received signal will have a high enough amplitude to be processed.

4.2.3 TELEMETRY FUNCTIONS BOARD

The telemetry function unit combines both the battery and temperature measuring circuits. The telemetry PCB is shown in Figure 4.4.

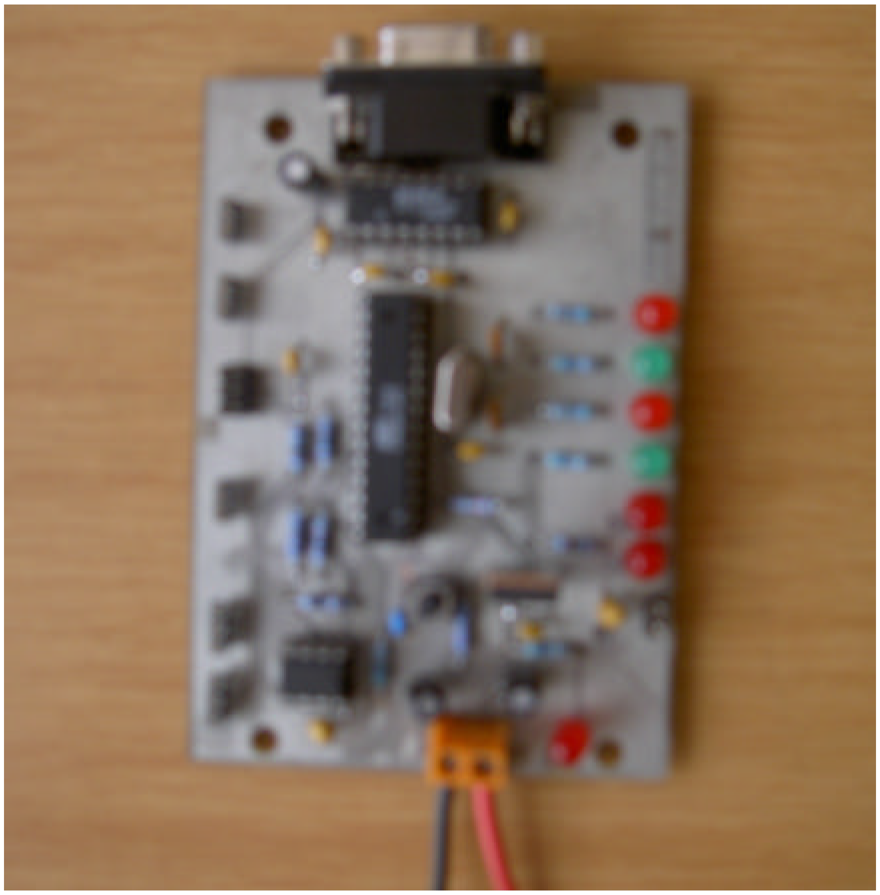


FIGURE 4.4 TELEMETRY FUNCTIONS PCB

4.2.4 CENTRAL CONTROL BOARD

The central control board performs the function of a router with decision making on system operation, depending on the type of messages received. The PCB designed and built is as shown in Figure 4.5.

For test purposes, the board was connected serially to a laptop and to the left and right motor control boards. The motor control boards were each connected to a motor.

By typing characters on the keyboard as commands, the central control board could route the commands to the motor control board and the motors could start running. The test was conducted for on, forward, backward, left, right and stop.

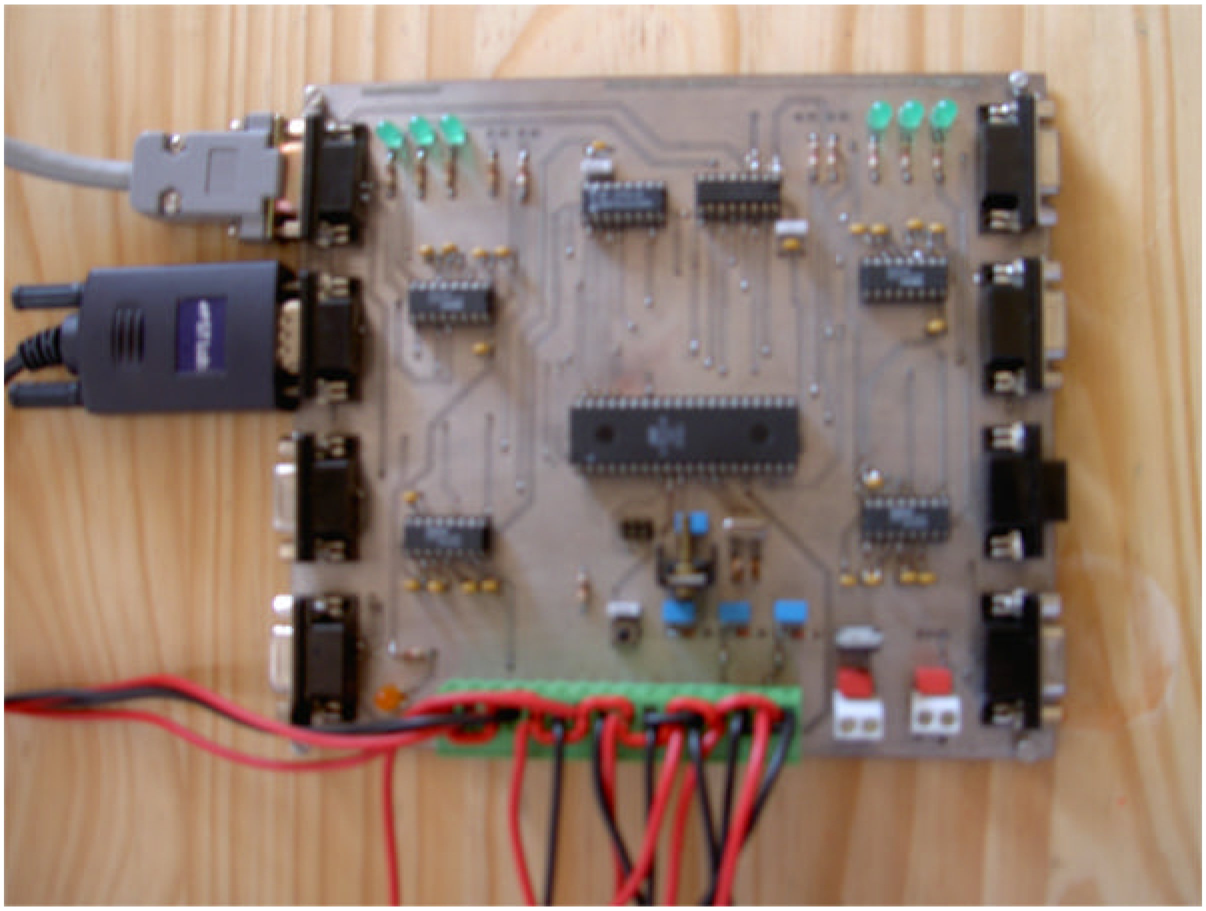


FIGURE 4.5 CENTRAL CONTROL UNIT PCB

4.3 SYSTEM INTEGRATION

4.3.1 ELECTRONIC AND SOFTWARE INTEGRATION

All the different functional units were connected together as planned in the conceptual design (internal architecture) during integration. This integration was done in three different phases.

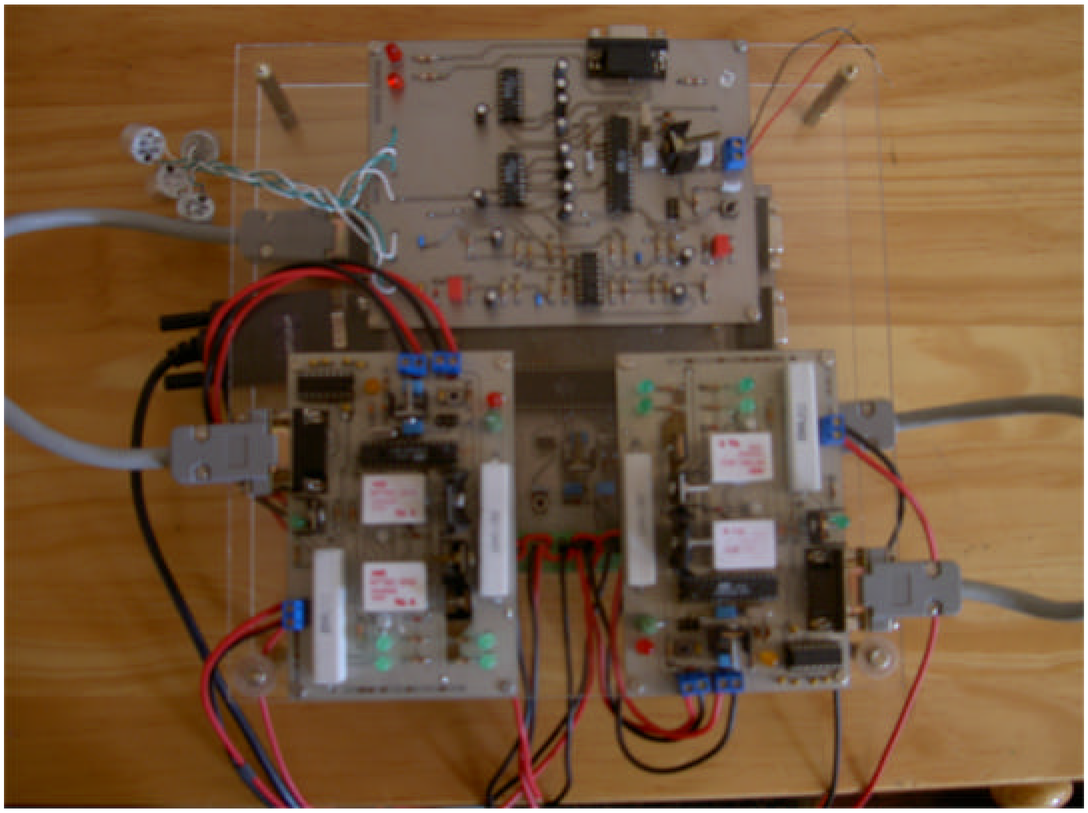


FIGURE 4.6 FIRST PHASE OF INTEGRATION

The first phase of integration consisted in interfacing all the peripherals to the central control board through the RS 232 interface and testing the operation of the assembly.

The peripherals mentioned above are the different electronic boards that connect to the central control board such as Motor control boards, proximity detector board, telemetry board and the voice recognition board. This integration is shown in the Figure 4.6.

The second phase of integration was to interface the electronic assembly just tested in the first phase, to the system main control and test the data flow from one point to another. Figure 4.7 shows how the different boards fit at different levels on the stand.

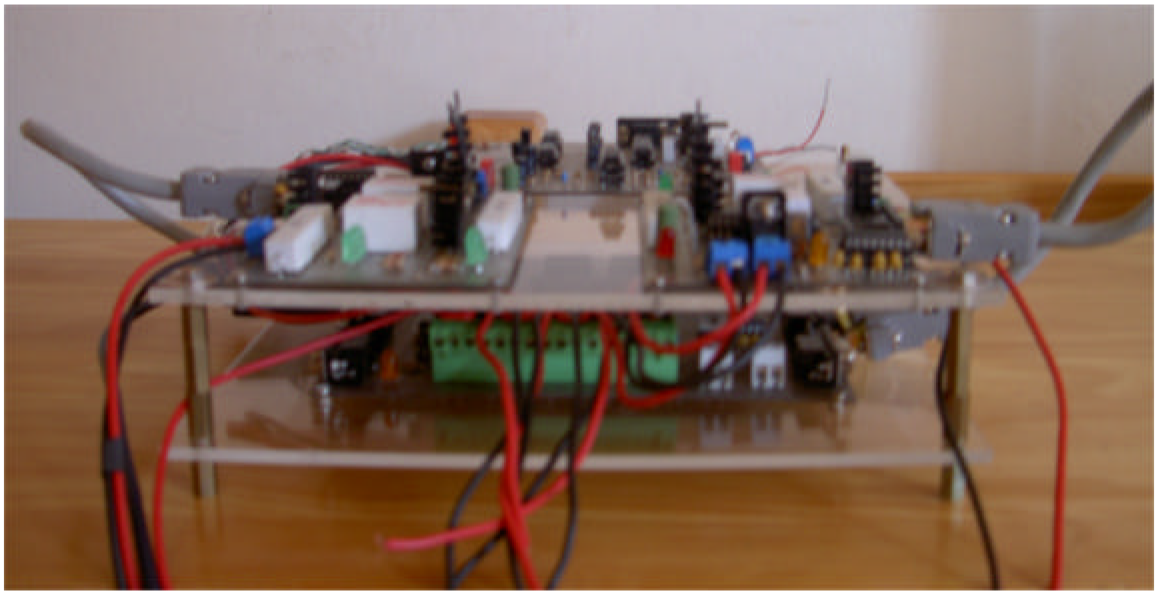


FIGURE 4.7 ELECTRONIC CIRCUITS ON STAND

The third phase of integration was to bring all the mechanical parts together into the main frame which will support all the other units that make the payload of the robot. The assembled main frame is shown in Figure 4.8.

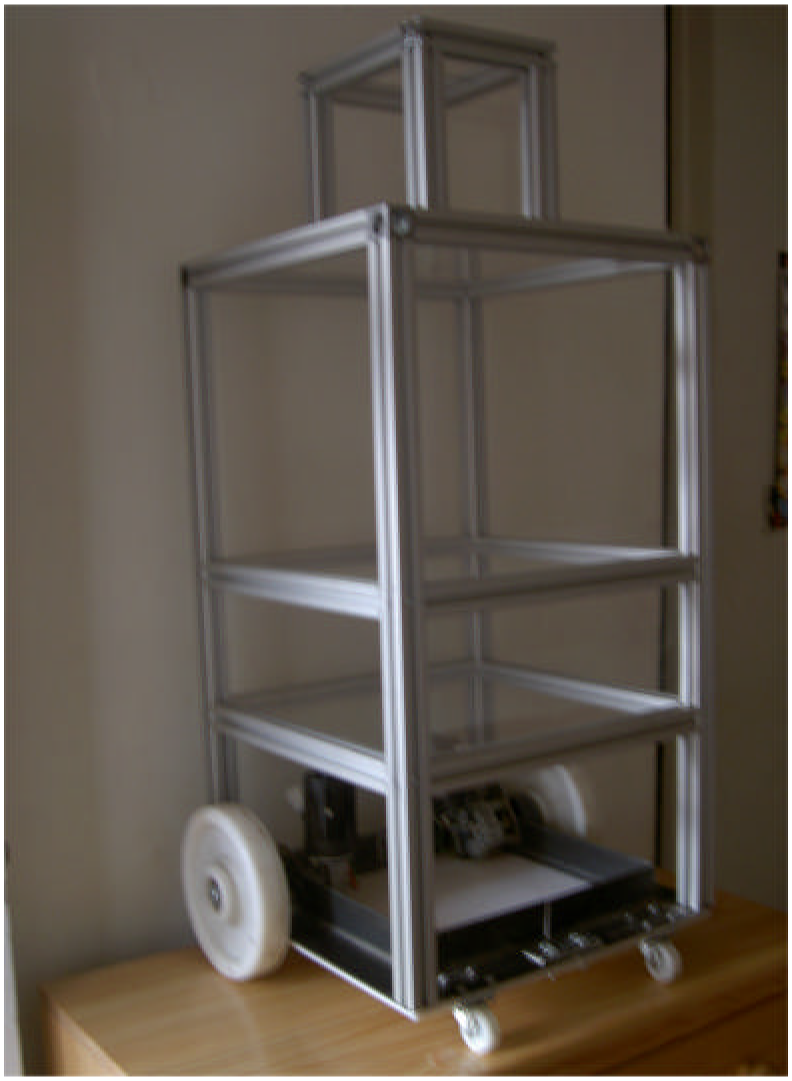


FIGURE 4.8 THIRD PHASE OF INTEGRATION

4.3.2 COMPLETE SYSTEM INTEGRATION

The last phase of integration was to integrate the electronic assembly and the other devices that constitute the robot such as the batteries and the camera into the mechanical frame built to support it. At this stage, the prototype looked as shown in the Figure 4.9.

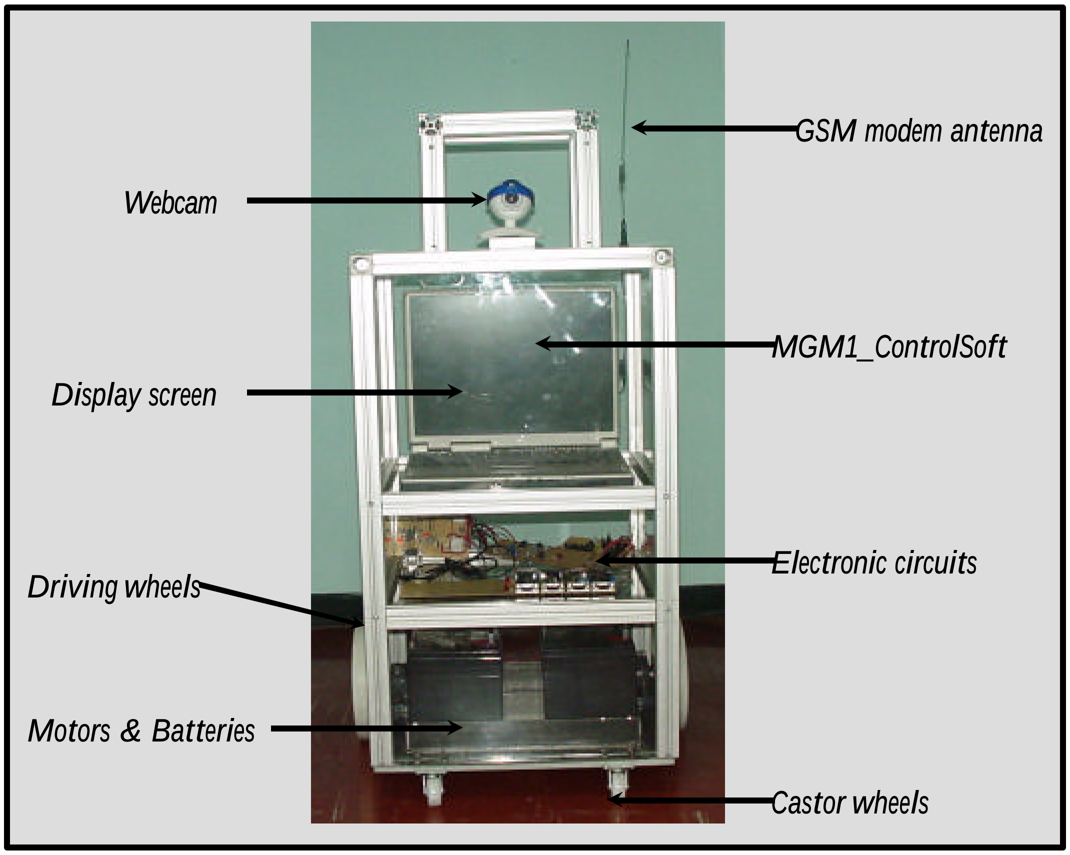


FIGURE 4.9 COMPLETE SYSTEM INTEGRATION

4.4 TESTS

This section details the validation tests of the main functions implemented in the robot. Most of these functions are requirements found in the initial specifications of the project. Some validation tests where conducted at some stages during integration and others after the complete integration. Each of the tests will be discussed separately.

4.4.1 REMOTE CONTROL TEST

The remote control function using a mobile phone involves the GSM network, the GSM modem, the PC, the main control application, and the central control board for command analysis, data transfer, and command execution. The first phase of validation test was done using the experimental setup shown in Figure 4.10

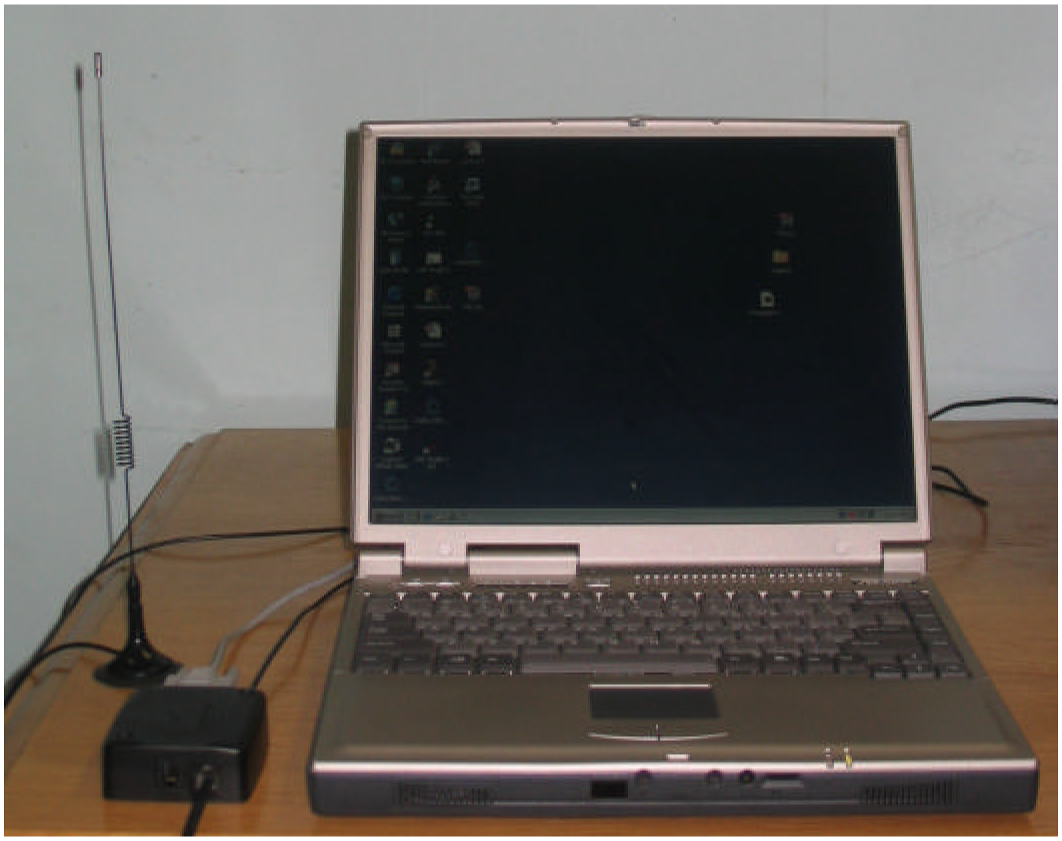


FIGURE 4.10 REMOTE CONTROL TEST EXPERIMENTAL SETUP

The aim of this test was to prove that it was possible using a mobile phone to remotely control the robot. The mobile phone is used in this test as the originating tool for SMS, which are commands to be sent to the robot.

The command is supposed to travel through the GSM network, be received on the robot side by the GSM modem, analyzed by the MGM1_ControlSoft application, transmitted to the central control board and then to the motor control boards to finally drive the motors.

So, to validate this test, the command path had to be followed through the whole chain, until the final output is observed through the motor's behaviour.

This test was successfully conducted, and the response of the motors to the message transmitted by the mobile phone was observed approximately 9 seconds after the message was sent. When the command transmitted by the mobile phone is received, the motors start running, moving the robot in the instructed direction and after an arbitrary time set in the code for tests purposes, the robot stops and waits for the next command.

4.4.2 PICTURE CAPTURING AND DISPLAY TEST

A picture capturing function is implemented in the robot, capable of displaying a captured picture. The robot main control unit, the web-camera and the mobile phone are devices used to achieve this function. The preliminary test carried out to validate the proper operation of this feature was done using the experimental setup shown in Figure 4.11.

The aim of the test is to show that once integrated, MGM1 will be able to take pictures on request and display them on the LCD screen. The motor behind the above experimental setup is the application developed on the PC to control the major activities in the robot, the application is called “MGM1_ControlSoft”. This application is built to recognize messages such as “Picture”, “Take a picture”, or “Take me a picture”.

Once “MGM1_ControlSoft” receives such a command from the mobile phone via the GSM modem, it activates the camera and takes a picture. The picture is stored in memory and then displayed on the application window. This test was conducted successfully and is fully operational in the system.

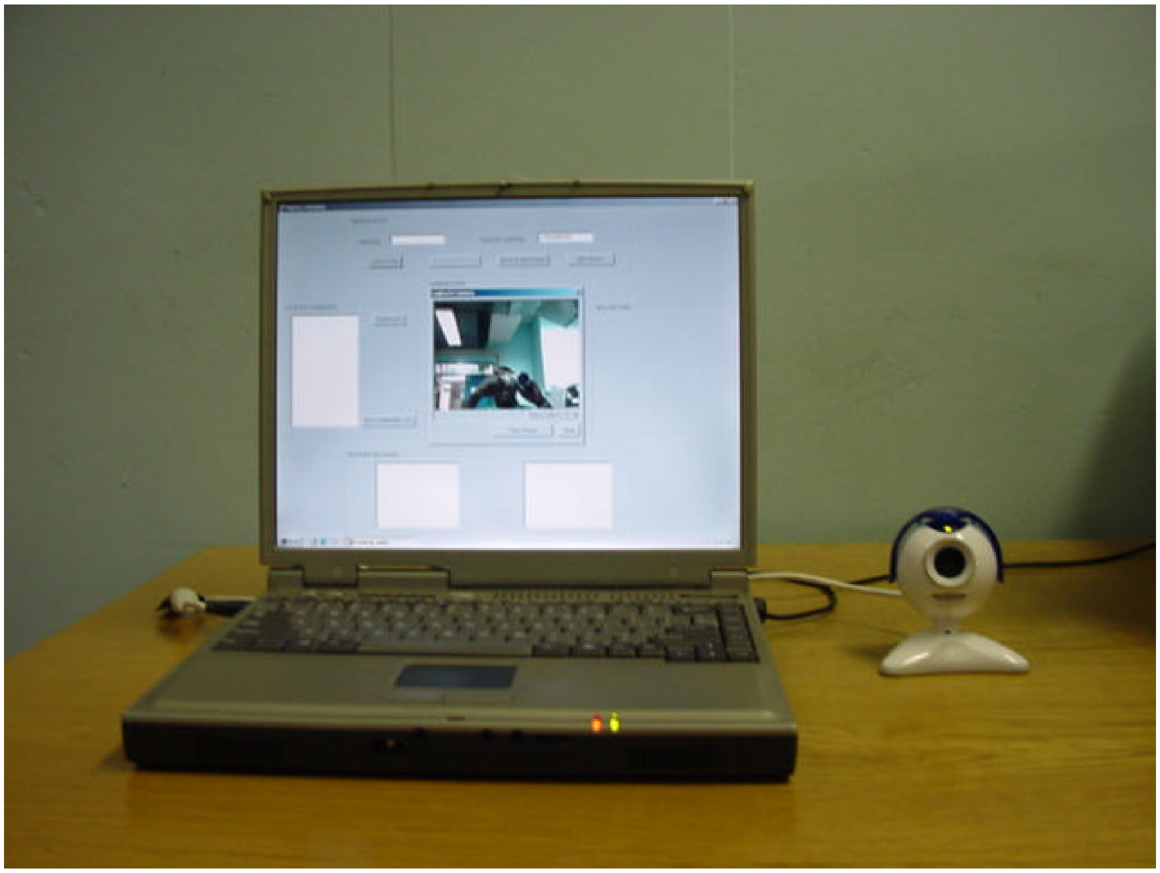


FIGURE 4.11 PICTURE CAPTURING EXPERIMENTAL SETUP

4.4.3 SMS REPLY TEST

MGM1_ControlSoft has been developed to receive commands and messages from the GSM modem and to reply to them via the same channel. So using an experimental setup similar to the one in Figures 4.10 and 4.11, it was possible to validate the implementation of this feature.

When an SMS is sent from the mobile phone, it is received by the GSM modem which transmits it serially to the PC. The MGM1_ControlSoft receives that message or command, analyses its content and executes it if necessary.

Depending on the type of message, MGM1_ControlSoft has a number of pre-defined answers to the received messages such "Command received", "Command executed", "MGM1 received your message", allowing it to always reply to any message received.

4.4.4 VOICEMAIL RECOVERY TEST

The voicemail recovery function was implemented in MGM1_ControlSoft. The GSM modem was connected to the laptop via the serial port and a telephone set connected to the RJ11 port on the modem.

A message requesting "voicemail recovery" from the remote control phone is the command to initiate voicemail recovery. Any SMS containing the word voicemail such as "retrieve voicemail", "get voicemail", or simply "voicemail" is considered to be the message requesting voicemail recovery. If such a message is sent to the MGM1, the control application (MGM1_ControlSoft) dials "111" for instant access to voicemail in the MTN network. If the connected telephone set is off hook, the voicemail is heard through the phone's speaker.

4.4.5 INTEGRATED SYSTEM TEST

After validating the operation of all the implemented functions (motion, SMS reception and reply, voicemail recovery, internet connection) and integrating all the tested units into the mechanical frame, a general test of the robot was conducted to validate the operation of the complete system.

Testing the prototype was done step by step in order to test each of the implemented functions. First of all, MGM1 has to be powered through the main switch.

Once all the power indicating LED on the electronic boards are on, MGM1 is able to receive any message addressed to it and reply to those messages. It can also take pictures but it will not execute motion commands. This is due to the fact that there is a software key implemented in MGM1_ControlSoft such that, the remote control alone can initiate motion. For MGM1 to become active, it has to receive an "ON" command first. Once it becomes active, it starts responding to motion commands from the remote control phone.

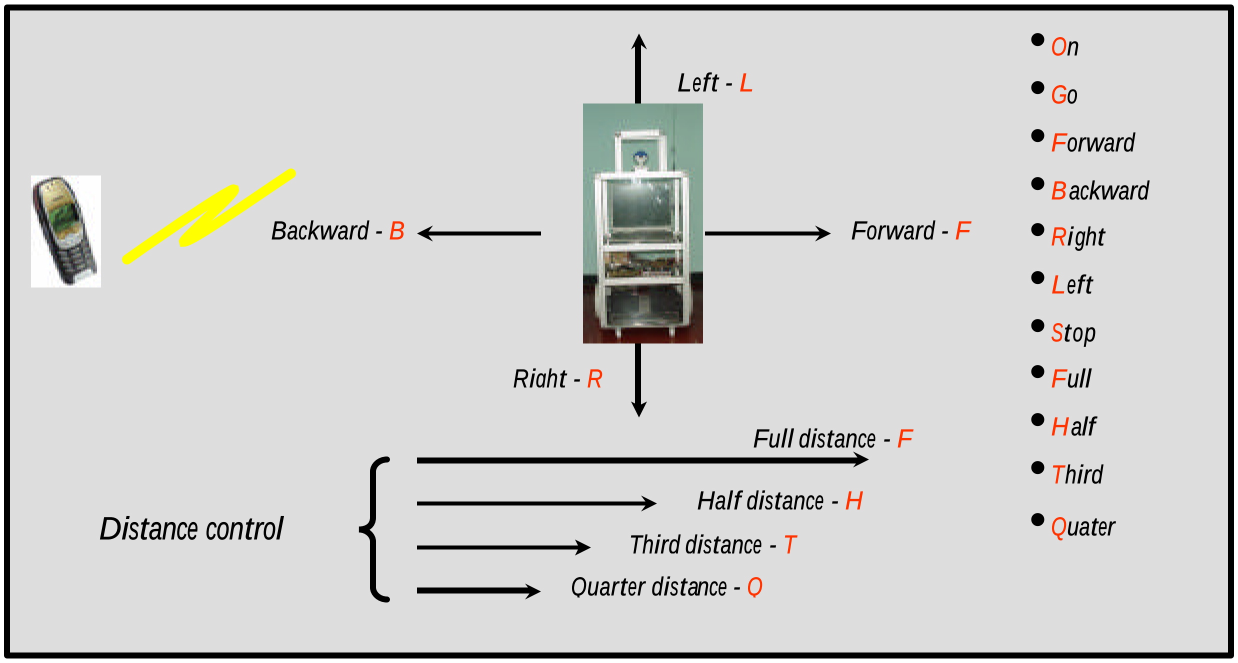


FIGURE 4.12 COMMAND USED IN MGM1

For test purposes, MGM1_ControlSoft and microcontrollers codes were written to recognize only the first letters of instructions. During system testing, messages such as "O" for On, "G" for Go, "F" for Forward, "B" for Backward, "L" for Left, "R" for Right, "H" for Half distance, "T" for Third distance, "Q" for Quarter distance and "S" for Stop were sent to MGM1 and the it executed the command as expected. This idea and all the commands used are illustrated on Figure 4.12.

MGM1 recognizes any message containing the word “Picture” as a picture request message. During system testing, MGM1 activated the camera and took pictures each time messages such as “Take a picture”, “Take me a picture” or simply “Picture” were sent.

	Minimum time	Maximum time
SMS	6-7 sec	9 sec- Undefined
Pictures	13-14 sec	17 sec-Undefined
Motors	8-9 sec	12 sec-Undefined

Table 4.1 Response times

This table shows the delay between the time a command is sent and the time a corresponding response is observed. The time is in seconds and undefined refers to the case where there is congestion in the network and the message is not forwarded immediately to the destination.

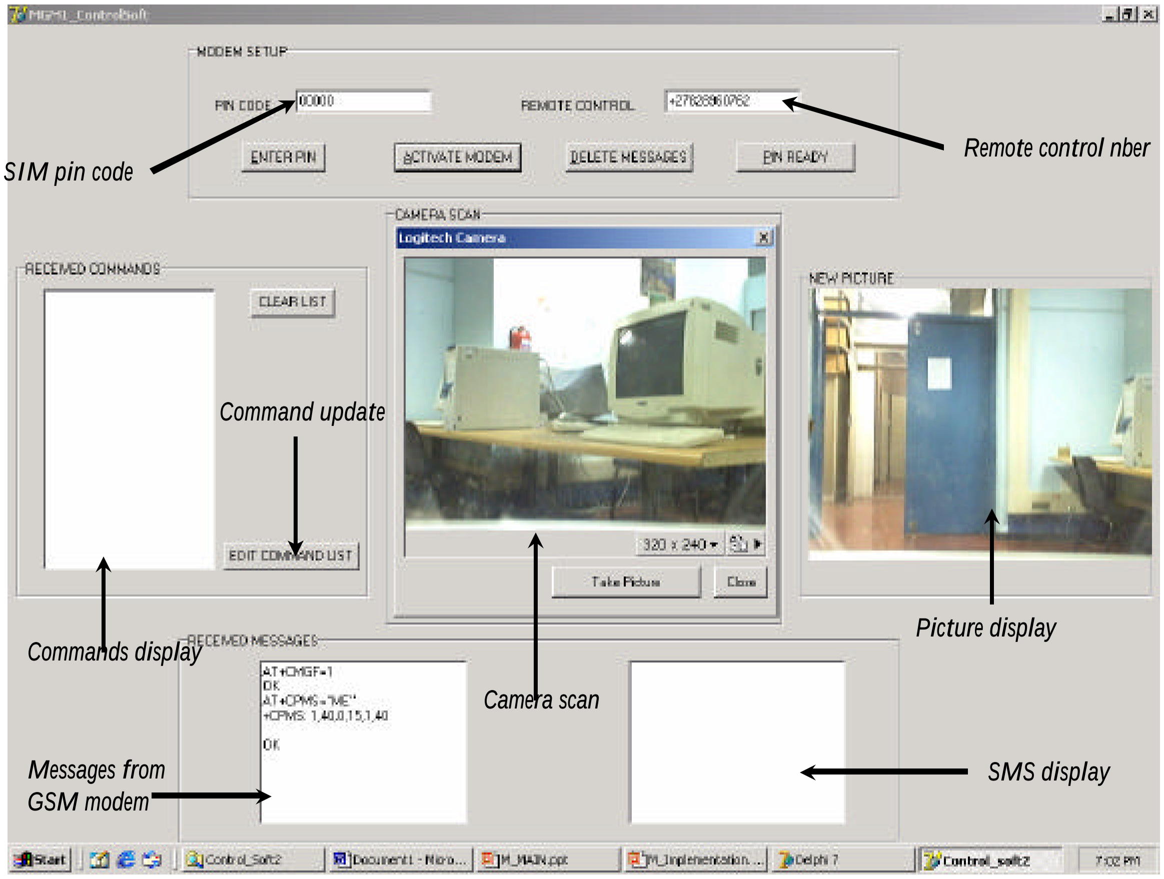
4.5 RESULTS

The aim of this project was to design and build a GSM Technology demonstrator robot, and the main output of this research project is the prototype mobile robot referred to here as MGM1. In this section the prototype and some important tools like the main control software (MGM1_ControlSoft) and some aspects of the performance of the robot were presented.

4.5.1 MGM1_CONTROLSOFT

One of the great achievements in this project is “MGM1_ControlSoft”, a windows-based application developed to be the main brain of the system, which manages the other tasks to be carried out by the robot.

MGM1_ControlSoft is the main interface between the GSM network and the electronic and mechanical parts of MGM1. It has been designed to be a user-friendly Graphic User Interface. On the MGM1 LCD display, this application is open and maximized on the screen. Figure 4.13 shows the GUI of MGM1_ControlSoft.



```
AT+CMGF=1
OK
AT+CPMS="ME"
+CPMS: 1,40,0,15,1,40

OK
```

FIGURE 4.13 MGM1_CONTROLSOFT GRAPHICAL USER INTERFACE

At power ON, the application is automatically launched and becomes immediately operational, but a few things such as initializing the GSM modem by entering a pin code for the SIM card used in MGM1 have to be done before the system is fully functional. MGM1_ControlSoft has been developed using Delphi7.

4.5.2 FINAL PROTOTYPE

The final prototype presented as the output of this project is as shown in Figure 4.14.

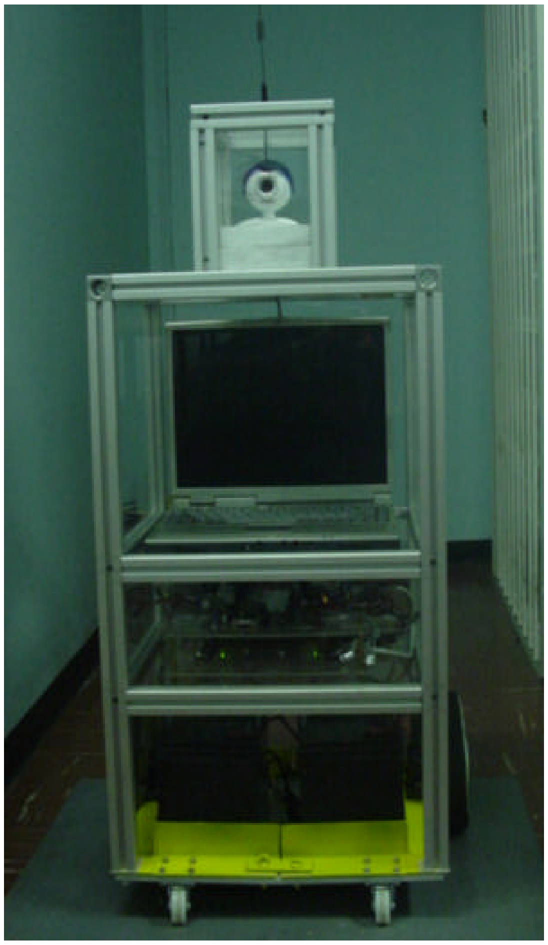


FIGURE 4.14 MGM1 PROTOTYPE

MGM1 is connected to the GSM network, and remotely controlled by a mobile phone. MGM1 does move and can perform a number of movements, change directions and go specific distances as specified by commands. MGM1 can take pictures each time one is requested through a SMS, and it can recover its own voicemails.

4.5.3 PROTOTYPE PERFORMANCE

Functions	Features	Study	Implementation	Observation
Motion	Forward	√	√	Small problems may be encountered on a slippery floor due to the lack of strong grip.
	Backward	√	√	
	Turn right	√	√	
	Turn left	√	√	
	Speed change	√	√	
Remote control		√	√	2 to 3 seconds delay
Pictures	Snap	√	√	Picture size is small. The display is also small
	Save	√	√	
	Display	√	√	
SMS reception		√	√	All messages sent are received
SMS reply		√	√	If airtime is finished, no reply
Voicemail recovery		√	√	Need good speaker to be heard
Telemetry	Temperat ure	√		Not fully implemented
	Battery level	√		
Obstacle detection		√	√	Only in front
Bluetooth	Data trsfer	√		No application
Web surfing	Internet connect	√	√	Not part of the Control_Soft.

Table 4.2 Implemented functions and performances

The functionality of MGM1 has already been discussed in the previous sections. In this section, a number of aspects of MGM1 operation and performance that were not highlighted before, will be discussed and presented here in summarized form.

Table 4.1 summarizes the project outputs in terms of the functions that had to be implemented. The table shows the topics that have been designed and implemented, describing the performance for each of them and some general comments.

During the testing, MGM1 travelled approximately 4 metres in 5 second. Using this data, the actual speed of MGM1 can be calculated using the following formula:

$$S = \frac{d}{t} \qquad (4.1)$$

Here, *d is the distance traveled and t is the time taken to travel that distance.*

The speed is given by:

$$S = \frac{d}{t} = \frac{4}{5} = 0.8 \, {}^{m}\!/_{s} \qquad (4.2)$$

On average, the time it takes a command from the mobile phone, to the MGM1_ControlSoft is measured to be approximately 6 seconds in average for 10 tests done.

The time taken by MGM1_ControlSoft to process that command is approximately 2 seconds. During that time, the application analyses the received message, replies to it, and instructs the response unit of interest to execute the requested task. This gives an idea of the total time taken a command sent is finally executed. For 10 tests conducted, the average of the measured time-delays was found to be 8 seconds.

4.6 RESULTS DISCUSSION

A lot has been said about the design, and the implementation approach taken for the fulfillment of the main goal of this project, mainly the building of a mobile robot that can act in exhibitions as a GSM Technology Demonstrator.

The implementation of the remote control function in this project has been achieved by SMS. The use of GPRS technology for such a purpose is left for future work. Remote control through SMS works relatively well. There is a noticeable delay problem but it is not as serious as anticipated at the start of the project.

4.6.1 Remote control via GPRS

In fact, using the current infrastructures, it is not possible for a mobile terminal to connect directly to another mobile terminal via a GPRS network alone. It has to connect to an IP network where a server has to reply to its requests.

In order to transfert data via GPRS, the system has to connect to the GSM-GPRS network first, then connect to a wireless network gateway, and then connect to an IP network.

This can be achieved by developing an application using AT commands and TCP/IP protocols, but the requirements for the development of such an application (mostly regarding time) were beyond the scope of this project.

4.6.2 Sending pictures as MMS

Just as GPRS, MMS technology has not been implemented due to the complexity involved in accessing all the Gateways in the MTN network, and also to the lack of information on this specific topic and the poor support from the GSM modem supplier.

Due to the distributed microcontroller architecture implemented, the system does not show any sign of data congestion, contrary to the initial hypothesis. There is still room for other peripherals to be added to the central control board, and it appears that even then, the system will still be able to run smoothly without any congestion.

The trajectory of MGM1 is not perfect. The robot can move straight but not in an optimum way. This is due to slippery wheels and floors. Since the microcontroller executes instructions sequentially and when starting the motors, it cannot address both motors simultaneously, but rather address one and then the other. However, this microcontroller shortcoming is not clearly noticeable.

The shape of the robot was not a critical issue for this prototype. There probably are many other ways of handling this particular aspect of the robot, but it was decided to build a neat frame for experiments to be conducted with relative ease. Still, there is a possibility of building a more aesthetic housing to the whole structure, and to give the robot a better shape and physical appearance.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 INTRODUCTION

Designing a mobile robot to be used as a "GSM technology demonstrator" and assessing its performance was the main objective of this project. The design of the complete system was discussed in chapter 3 and the keys elements to the performance of the robot were evaluated in chapter 4. In this chapter, all the conclusions drawn from the design and implementation of subsystems as well as the integration of the complete system are described. The current status of the MGM1's performance is discussed and finally some recommendations in the design as well as the performance of the system as a whole, are discussed.

5.2 DESIGN CONCLUSIONS

The design of MGM1 is based on the concept developed during the early stage of the project, inspired by both the nature and the goal of the project. As a demonstrator to be used in exhibitions, certain basic requirements such as being transparent to display the technology used in the design, easy access of internal components for maintenance purposes, a simple architecture for ease of maintenance and modular design to make the system as independent as possible were of great importance. The aim was to cater for as many of these aspects as possible in the design process.

The design concept of MGM1 is supported by two different architectures brought together, mainly the electronic architecture and the mechanical architecture.

The electronic architecture of the system is semi-centralized, meaning that the main control of the system controls only some critical devices like motors while others like sensor units are quite independent and only communicates with the main control for data transfer. This has been achieved by using distributed microcontroller architecture. All electronic circuits have been implemented on PCB, and have all been mounted on the circuit panel to keep everything inside the robot neat and robust. The wiring has also been taken care of in order to keep the interior clear and presentable. In terms of maintenance, this has great value, since a faulty unit can just be removed and changed without necessarily affecting the operation of the whole system.

The mechanical design is adapted to facilitate mobility of the robot. It is a simple design made of three main portions: the head, the body, and the chassis.
The three units can be assembled together and dismantled with relative ease. The body and the head can also be dismantled further by separating the aluminum profiles from the Perspex and therefore reducing considerably the size of the whole system for transportation purposes.

The frame is made of clear Perspex, therefore transparent and revealing the robot interior. It has three different compartments where mechanical, high power, electronics and other equipments can be stored depending on their categories. Also, each compartment is easily accessible from an open door at the back of the robot. All these aspects make the system easy to understand, use and maintain.

5.3 PERFORMANCE CONCLUSIONS

MGM1 performs a number of communication and mechanical functions. Mobility is one of the most important requirements of the project, and MGM1 does move forward, backward and turn left as well as right at the operator's command. The performance as far as the motion is concerned is satisfactory. The electronics and mechanics used to implement motion in MGM1 are far from being perfect. Aspects such as the "used technology" were discussed in chapter 3 with the trade-off attached to each of the various options.

The concept of controlling the robot with a mobile phone, has also been implemented with great success. Though the reliability of the use of a mobile phone as a remote control depends on many factors some of which are some, external to the designed system such as the GSM network, no failure was observed during the test period. Over a period of six months, the control of MGM1 via a mobile phone proved to be robust and quite reliable.

An obstacle detection system has been implemented in MGM1 to detect obstacles within the range of one meter in front of the robot. This detection system can be improved to a higher level of sensitivity to detect obstacles all around the robot. In this document, two different methods of increasing the sensitivity of MGM1 are proposed. The simplest way to achieve this is to build three more proximity detector boards similar to the implemented board, so that there is at least one board with two ultrasound transmitters and two ultrasound receivers on each of the four sides of the robot. Another way is to develop a new proximity detector board with a microcontroller that will manage all the ultrasound sensors positioned on the four sides of the robot. The main disadvantage of the first method is the fact that it will require the use of four different serial ports just for obstacle detection. The main control application which runs on the PC, coordinates all the activities in the robot as expected.

When MGM1_ControlSoft receives a picture request message, it activates the camera to take a picture and stores that picture in a defined directory in the hard drive and finally, it displays the picture taken in the picture frame on the application window (center right). MGM1_ControlSoft communication with the central control board is quite reliable. Though the MGM1_ControlSoft may be optimized to cater for more sophisticated functions, and a better GUI (Graphic User Interface), it has proven to be quite stable and reliable over the 6 months test period.

The complete system as integrated is working properly. MGM1 responds to all the implemented commands performing all the implemented functions, but one of the visible problems is the fact that the process of sending a command and the command execution are time consuming, making the system operation a bit slow.

The conclusions drawn above about the performance of the robot and the different functional units that are part of the system are given within the context of the tests conditions. An average test lasted about an hour.

5.4 PROBLEMS ENCOUNTERED AND APPROACH TO SOLUTIONS

Many problems were encountered during the course of this project and in each case, they were dealt with in a way or another. In this section a number of the problems encountered and how they were attended to will be discussed.

5.4.1 PROBLEMS WITH MOTORS

The choice of the motor is based on its torque and the torque is a function of the weight to be moved and the real weight of the robot was not known at the start.

Wiper motors are suitable for positioning on the same side, either left or right. Using them in the robot as left motor and right motor makes them run in opposite directions.

The chassis was a single metal structure to which both motors were connected. The motors could not run both at the same time, there was a short circuit problem.

The first problem was solved by estimating the final weight of the robot and calculating the minimum required torque. For the second problem, the polarity across the motors was reverse to make them move in the same direction. The third problem was solved by cutting the chassis in the middle and electrically isolating the two parts.

5.4.2 PROBLEMS WITH MOTOR CONTROL BOARDS

The FET switching the high current to the motors and the high current transistor in the current limiti ng circuit blew up a couple of times.

Using test equipment probes for measurements and troubleshooting on the powered circuit was a hazard, often blowing components and burning tracks.

The first problem was solved by acquiring components with a higher current rating and adding heat sinks to them. No solution was found for the second problem except increasing care during measurements.

5.4.3 PROBLEMS WITH MICROCONTROLLERS

It happened that with the same code and the same baud rate it was not possible to obtain the same results from two identical microcontrollers (same part number) when transmitting data to a terminal application and generating the PWM signal.

A sort of oscillation was noticed on the motor control boards. When a command was received the board could start transmitting characters without stopping.

The first problem was not solved because not well understood, since the various tested microcontrollers were all in good working conditions. The oscillation problem was solved by re-soldering joints of some components such as the microcontroller and the RS232.

5.4.4 PROBLEMS WITH INTEGRATION

Cables between the units were often a problem, sometimes just loose and sometimes not responding all.

Data transfer between the different units was not always working like when tested with the PC.

Problems with the common power supply did rise. Need for more current that the supply can deliver and need for different supplies for different units.

Complexity of troubleshooting, debugging and test all the functions when everything is put together.

The problems with cables was solved by keeping the connections tight and checking the drivers installation when a problem arises.

The problem with the supply was encountered when using bench power supply units, but when a lead acid battery was used the problem was not encountered. The problem of complexity of troubleshooting and debugging an integrated system can only be improved by developing methods and techniques specific to each problem but it cannot be solved.

5.5 RECOMMENDATIONS AND FUTURE WORK

5.5.1 ELECTRONICS

The motor control circuit is relay-based. The technology is said to be old, mechanical and slow but still works fine. In this application, the speed and the fact that it is a mechanical device were not critical issues.

If the weight of the robot is reduced such that the motor requirement for current is not above 5A, one of the good options will be to use H-bridge integrated circuits. This will simplify the design and reduce considerably the size of the boards.

The motion of MGM1 can be better controlled if a feedback system is added to the motion control circuitry.

This can be implemented by introducing speed sensors at the wheel or current sensors in the motor supply circuits in order to determine the error between the speed or the current of the two motors and use the error signal to adjust the speed. Though the microcontroller is quite reliable, the speed control feedback system will increase the efficiency and the accuracy of the robot motion.

The central control board communicates with all the peripherals using the RS232 interface. Built around a single USART microcontroller, the circuit uses multiplexers and Demultiplexers to communicate with all the pheripheral boards attached to it.

The use of other standards such as RS485, I^2C or CAN bus can also be considered. Their implementation may require a number of nodes at each peripheral. In this case, the microcontroller code will also have to be changed, and will mainly depend on the way chosen to address each peripheral.

The general architecture of the robot can be modified by developing a FPGA board which will replace all the peripheral boards as well as the central control board. All the functions performed by the existing boards will be implemented by software in the FPGA using VHDL or Verilog as a programming language. This drastic change in architecture will have numerous advantages such as simplified internal architecture, reduced number of electronic circuits in the system, size and cost improvement, reduced complexity, faster integration and therefore improved development time.

The implementation of this option will require expertise necessary to the development of FPGA based circuits as well as a good knowledge of at least one of the programming languages for FPGA.

5.5.2 MECHANICAL

A lot can also be done to improve the mechanical part of MGM1. One of the most important aspects is to reduce the weight of the whole system by possibly using lighter mate rials in the construction of the main frame.

The use of spherical castor wheels is very much advised here since it will reduce motion problems encountered by using the cylindrical castor wheels.

The head of the robot may be made to rotate, so that it may be turned at a command at a specific time interval. In the same line of idea, mobile arms may also be added to the robot. The main frame may be covered with an outer shell to give a more attractive look to the robot.

5.5.3 SOFTWARE

As it has always been the case, software is the most flexible part of this system. Both the main control software and the embedded software may be improved to achieve higher levels of performance.

The PC based system control application developed is performing very well within the context in which it was developed. The internal architecture of the system was designed to stand for a support to this windows based application. A better option will be to develop a real time application on a computer having a real time operating system such as linux.

The “MGM1_ControlSoft” can also be improved. New functions can be implemented, and links of interaction between MGM1_ControlSoft and other applications can also be introduced so as to make the launch of those applications and the switch between them possible by request commands from the mobile phone.

The Graphic User Interface can be rearranged if possible in order to make it more presentable and more user-friendly. Since the aim is to use a touch-screen LCD, the positioning of all the window buttons and other components of the application window are very critical.

The embedded software may also be optimized as more functions are implemented. Each unit containing a microcontroller has a different code, so these upgrades should be done for each board separately.

Features such as Bluetooth connection to a phone or a video projector, MMS, connection to a public address system, and Web surfing demonstration using a phone browser are left for future work. Features such as voice recognition and obstacle detection have been designed, implemented and tested at circuit level but haven't been integrated in the final prototype. About MMS, all the implementation procedure and approach have been studied, the required equipment are also known. This makes future implementation of MMS more interesting.

5.6 CONCLUSION

MGM1 is the first prototype of the "GSM Technology Demonstrator" concept. A number of the units implemented are not really optimized as such, but rather, they have been laid as the foundation in this research process, on which, finer works may rise.

The most interesting thing is the fact that from the concept and nothing else, a basic "GSM robot" platform has been developed. The platform will be used for further research in order to develop better techniques, better designs and better models of the "GSM Technology Demonstrator".

LIST OF SOURCES CONSULTED

BAHR, L.S. & JOHNSTON, B. 1992. *Collier's Encyclopedia*. New York. Maxwell MACMILLAN.

BEDINI, S.A. 1999. *The role of automata in the history of technology* [Online]. Available from: http://xroads.virginia.edu/~DRBR/b_edini.html [Accessed: 10/11/2004].

BOBROW, L.S. 1985. *Fundamentals of Electrical Engineering.* New York: Holt, Rinehart and Winston.

Brief history of artificial intelligence [Online]. 2002. Available from: http://www.aaai.org/AITopics/bbhist.html [Accessed: 11/11/2004].

COBUILD ENGLISH LEARNER'S DICTIONARY. 1990. London: William Collins Sons & Co

COX, I.G & WILFONG, G.T. 1990. *Autonomous robot vehicles*. MC. USA: Edwards Brothers.

DOWLING, K. 1996. *What is robotics?* [Online]. Available from: http://www.frc.ri.cmu.edu/robotics-faq/1.html [Accessed: 11/11/2004].

DOWLING, K.J. 1997. *Limbless locomotion: learning to crawl with a snake robot.* Ph. D. dissertation, Carnegie Mellon University.

ERICSSON. 1998. Introduction to Mobile Telecommunication and GSM. Ericsson radio systems AB.

ERICSSON. 2000. GPRS System Survey. Ericsson radio systems AB.

FLOYD, T.L. 1999. *Electronic devices.* 5th ed. New Jersey: Prentice Hall

GLENN, D. 2002. *Van Nostrand's Scientific Encyclopedia*, 9th edition. New York: Wiley-interscience.

HALL, E.L. & HALL, B.C. 1985. *Robotics. A user-friendly introduction.* New York: CBS college publishing.

HEISERMAN, D.L. 1981. *How to design and build your own custom robot.* USA: Tab Books Inc.

HYSTORY OF MOBILE ROBOTS [Online]. Available from: http://www.amigobot.com/amigo/history.html [Accessed: 23/02/2004].

HOROWITZ, P & HILL, W. 1995. *The art of electronics.* 2nd Ed. Cambridge: Cambridge University Press.

LOOMIS, M.E.S. 1983. *Data communications.* Englewood Cliffs, NJ: Prentice Hall.

MALCOLM, Jr.D.R. 1988. *Robotics an introduction.* 2nd ed. PSW-KENT: Wadsworth.

PFEIFER, R., IIDA, F. & BONGNARD, J. 2003. New robotics: Design principles for intelligent systems. *Artificial life on new robotics* [Online]. April 6. Available from: http://www.mae.cornell.edu/bongard/papers/2004_ALife_Pfeifer.pdf [Accessed: 22/11/2005].

PORTOLANI, M. *SMS and remote controls* [Online]. Available from: http://www.dpspro.com/tcs_sms.html [Accessed: 20/01/2004].

PORTOLANI, M. *GPRS basics* [Online]. Available from: http://www.dpspro.com/tcs_gprs.html [Accessed: 11/02/2004].

PROCTER, P. 1996. *International dictionary of English.* New Dehli: Cambridge University Press.

ROBOTS [Online]. Available from: http://inventors.about.com/library/inventors/blrobots.htm [Accessed: 23/02/2004].

SCOURIAS, J. 1994. *History of GSM* [Online]. Available from: http://kbs.cs.tu-berlin.de/~jutta/gsm/js-intro.html [Accessed: 20/01/2004].

SHERMAN, K. 1985. *Data communication: A user's guide.* 2nd ed. Virginia: Prentice Hall.

SURJOO, S. 2003. GPRS call setup. MTN: Internal document.

TAYLOR, P.M. 1990. *Robotic control.* Hong Kong: Macmillan

WEBSTER, J.G. 1999. *Wiley Encyclopedia of Electrical and Electronics Engineering*. Vol. 7. New York: John Wiley & Sons, Inc.

WILLIAMS, M. 2002. *History of robotics* [Online]. Available from: http://www.bsu.edu/web/MAWILLIAMS/history.html [Accessed: 23/02/2004].

WHAT is MMS [Online]. Available from: http://mms.aip.org/arlo/faq.html#What_is_MMS_ [Accessed: 21/01/2004].

WHAT is MMS [Online]. Available from: http://www.gsmworld.com/technology/mms/whatis_mms.shtml [Accessed: 21/01/2004].

VAN DER HEYDEN, M. Wap [Online]. Available from: www.wap.net/devkit [Accessed: 22/02/2005].

WARWICK, K & GARROD, R. 2001. *Ultimate real robots 4*. London: Eaglemoss International Ltd.

WARWICK, K & GARROD, R. 2001. *Ultimate real robots 5*. London: Eaglemoss International Ltd.

WARWICK, K & GARROD, R. 2001. *Ultimate real robots 6*. London: Eaglemoss International Ltd.

APPENDICES

APPENDIX A

A.1 STEP BY STEP AT COMMANDS TO INITIALIZE MODEM

A.1.1 Typed commands

```
AT                                  #Are AT commands supported?
AT+cpin?                            #Need pin to be inserted or not?
AT+cpin="relevant pin"              #Insert pin code
AT+cpms="ME"                        #Work with modem’s memory
AT+cmgf=1                           #Activate Text mode operation
```

A.1.2 Replies

```
AT

OK                                  #AT commands supported
AT+cpin?

+CPIN: SIM PIN

OK                                  #+CPIN supported, insert it
AT+cpin="0000"

OK                                  #Pin code accepted
AT+cpms="ME"

+CPMS: 0,40,5,20,0,40               #Space: 40 and saved: 0
OK                                  #Switched to modem memory
AT+cmgf=1
```

```
OK                                        #Text mode activated
```

A.2 STEP BY STEP AT COMMANDS TO SEND AND READ SMS FROM THE MODEM

A.2.1 Typed commands

```
AT+cmgs="0721904784"              #Send message to following number
This is a test message            #Message content
AT+cmgr=1                         #Read first message stored
AT+cmgd=1                         #Delete first message
```

A.2.2 Replies

```
AT+cmgs="0721904784"

> This is a test message

+CMGS: 225

OK                                #Message has been sent
AT+cmgr=1

+CMGR: "REC UNREAD","+27721904784",,"05/03/28,14:04:39+08"
Have you received it? Freddy
OK                                #Message read from memory
AT+cmgd=1
OK                                #Message deleted
```

A.3 STEP BY STEP GPRS CONFIGURATION

A.3.1 Terminal configuration

A.3.1.1 Typed commands

```
AT+cgdcont=1,"ip","internet"
ATD<*99**1#>
```

A.3.1.2 Replies

```
AT+cgdcont=1,"ip","internet"

OK
ATD<*99**1#>

CONNECT
```

A.3.2 Windows configuration

1. You must be disconnected from any modem calls.
2. Click on START - SETTINGS – CONTROL PANEL.
3. Double – Click the “Phone and Modems Options” icon.
4. On the Dialling rules tab, select the GSM radio button.
5. On the Modems tab, highlight the modem that you want to connect with, like Infrared modem, serial cable or bluetooth modem.
6. Now click on the PROPERTIES button.
7. Check and change the maximum port speed set at 115200 kbps.
8. Click on the ADVANCED tab.

9. In the Extra Initialisation Commands box, enter the AT string that you desire for your GPRS connection. (Eg. At+cgdcont=1,"IP","internet","",0,0)
10. Verify that your phone can support all the commands that you wish to implement.
11. Note that the command might not need the "at" part of the string for some Windows versions, just start with a "+" sign.
12. Click on OK, then APPLY, and then OK again to close each tab.
13. Double – Click your shortcut icon that you use to make another dialup modem connection.

14. The computer will attempt to dialup to the number given using the GPRS modem on your mobile phone. In the dial string box, enter the number ***99***1#** and use your username and password given to you (you may also use guest and guest respectively).
15. You should see a little window showing the modem status, like dialling…verifying username and password…registering your computer to the network … authenticated!
16. You should then see the icon on the bottom right of your screen saying "Connected @" a certain speed.
17. In future dialups you can just open the dialup connection box and then dial the GPRS connection number. You do not have to enter the "at" command every time after your first connection, as it has already been written to the modem. Your phone will recognise it as a GPRS call by the special number you dial.

APPENDIX B

B.1 PRINTED CIRCUIT LAYOUTS

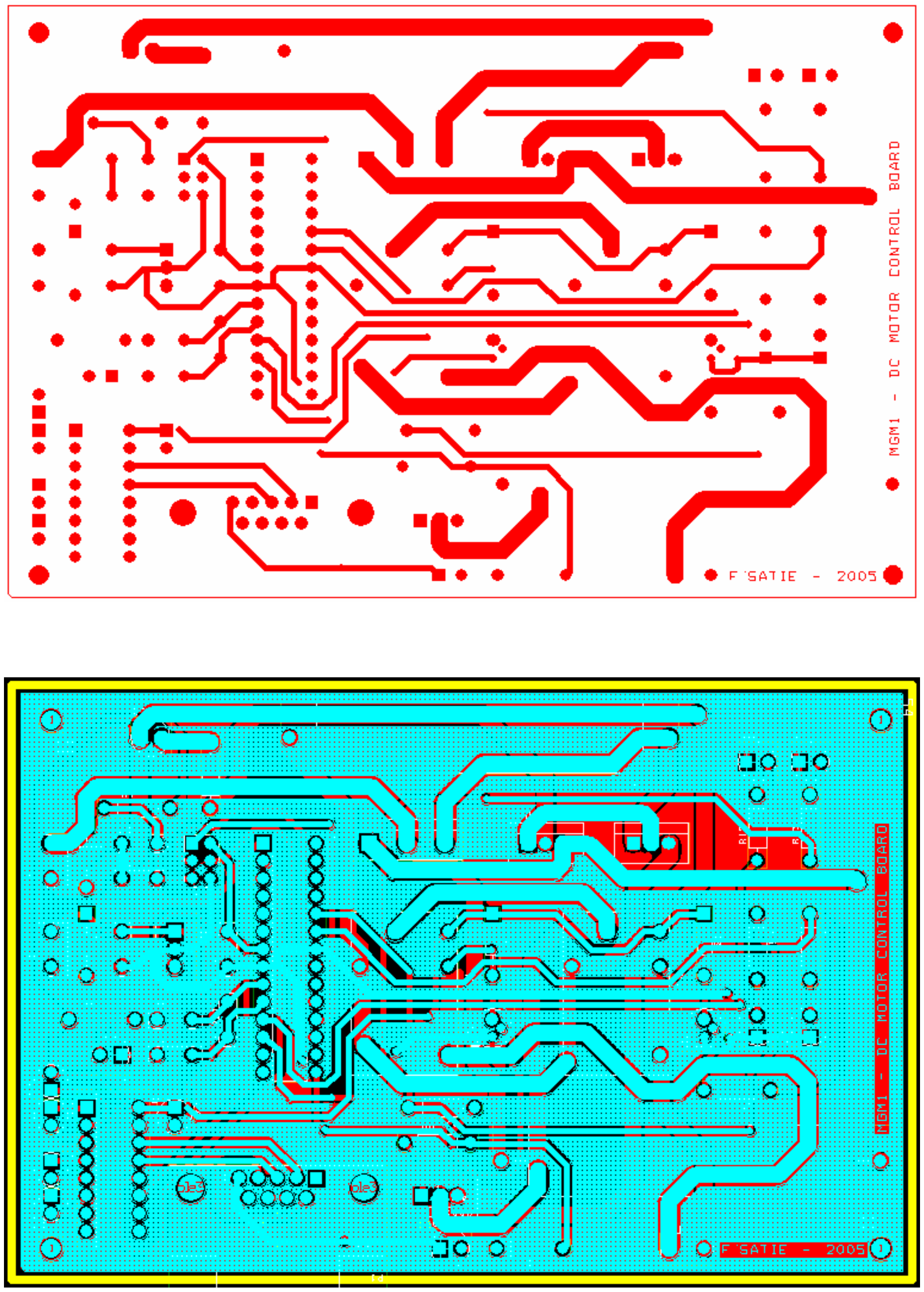
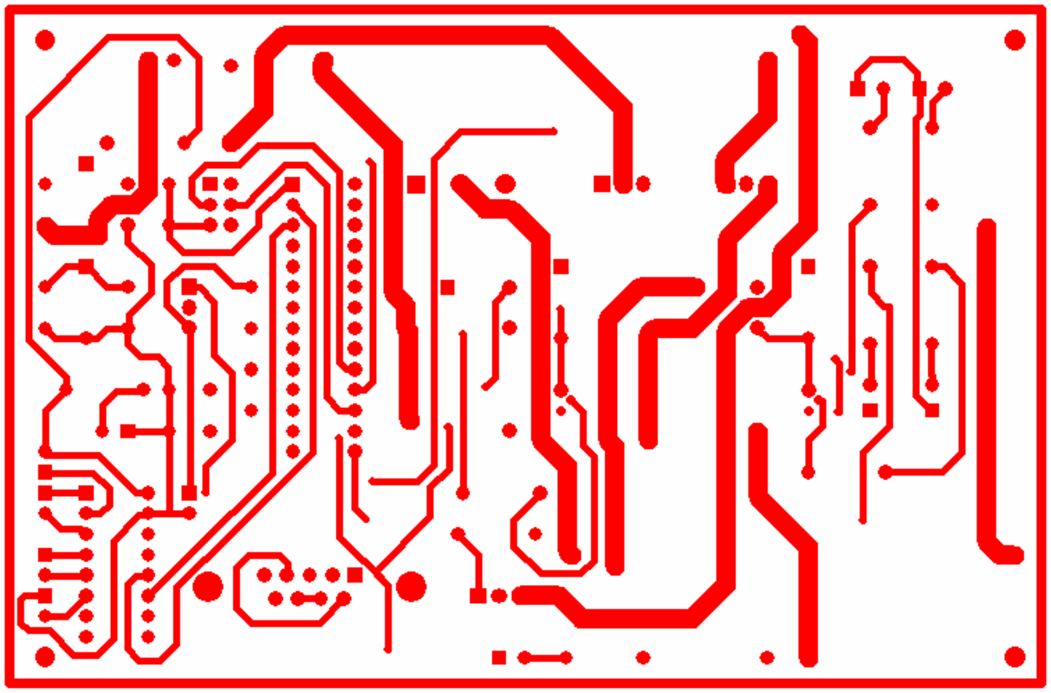


FIGURE B.1 MOTOR CONTROL BOARD TOP LAYER



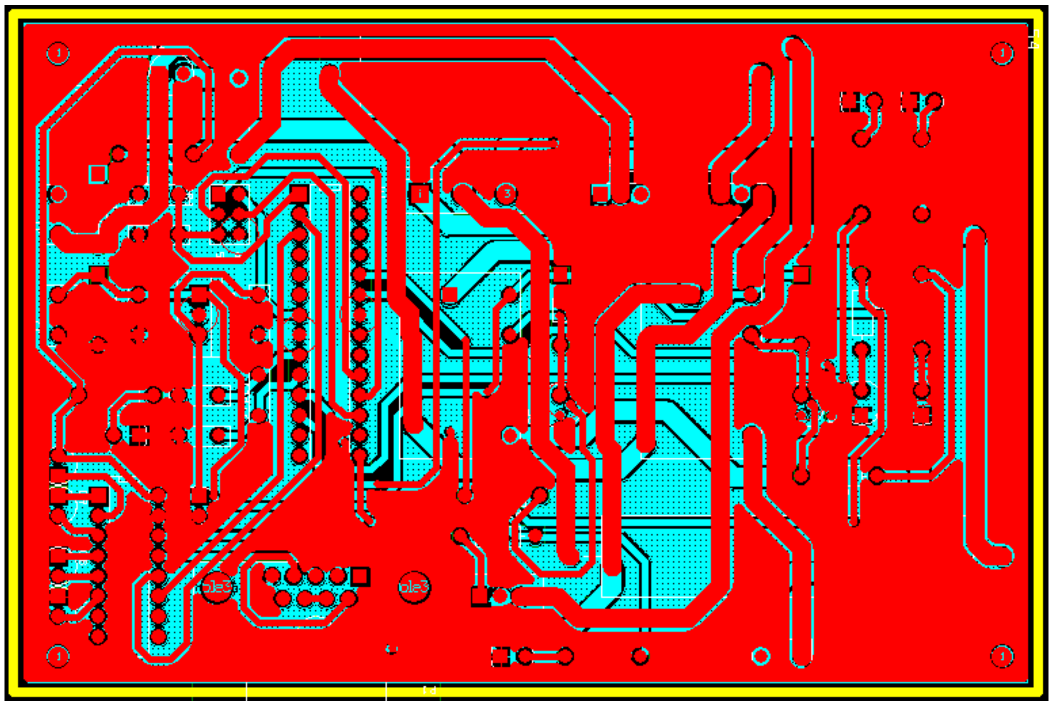
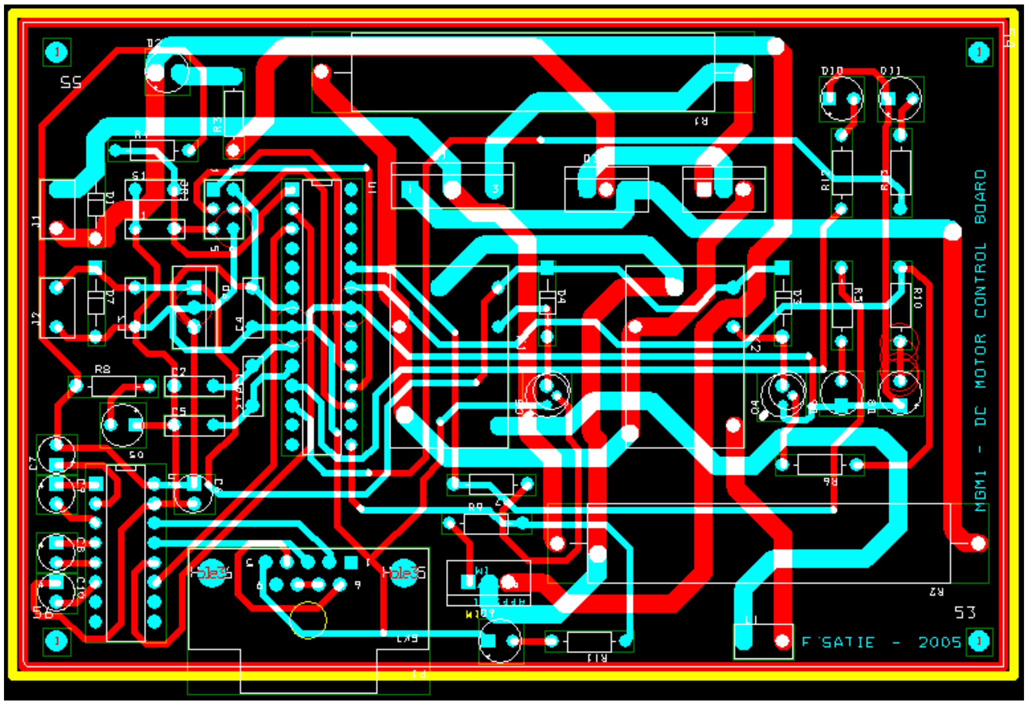


FIGURE B.2 MOTOR CONTROL BOARD BOTTOM LAYER



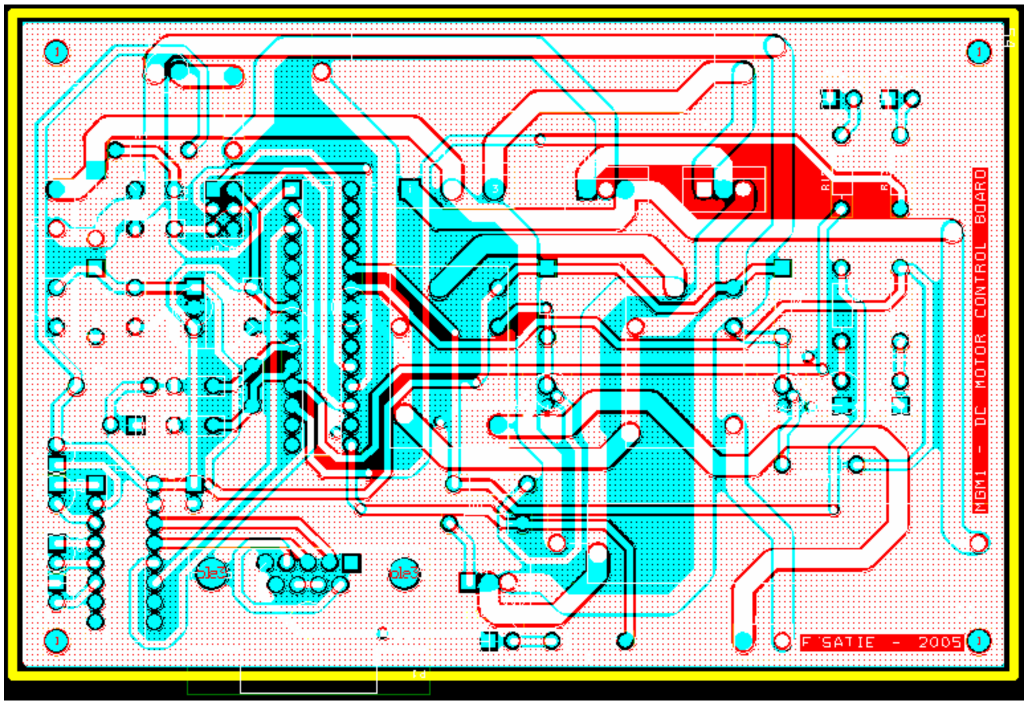


FIGURE B.3 COMPLETE MOTOR CONTROL BOARD LAYOUT

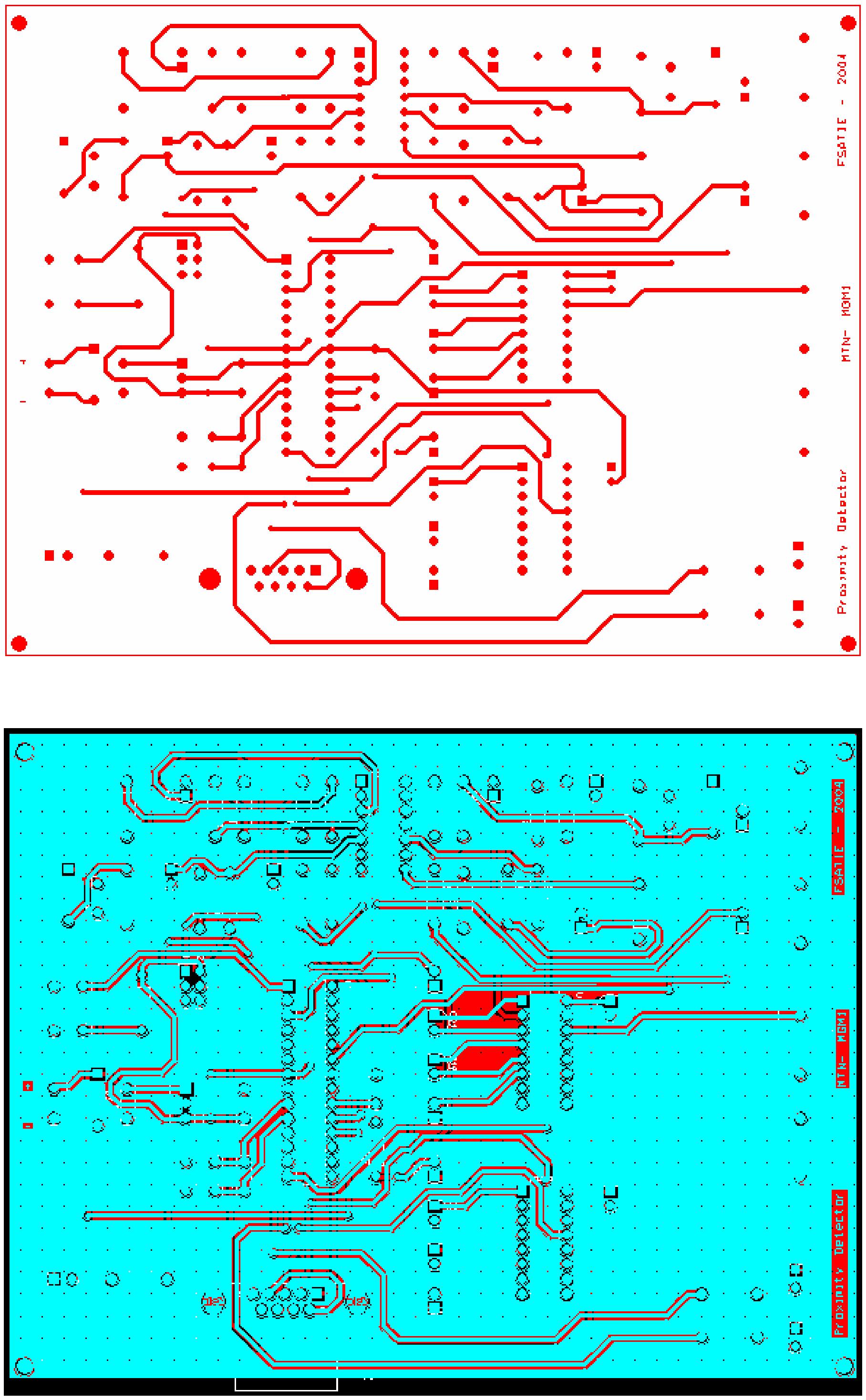
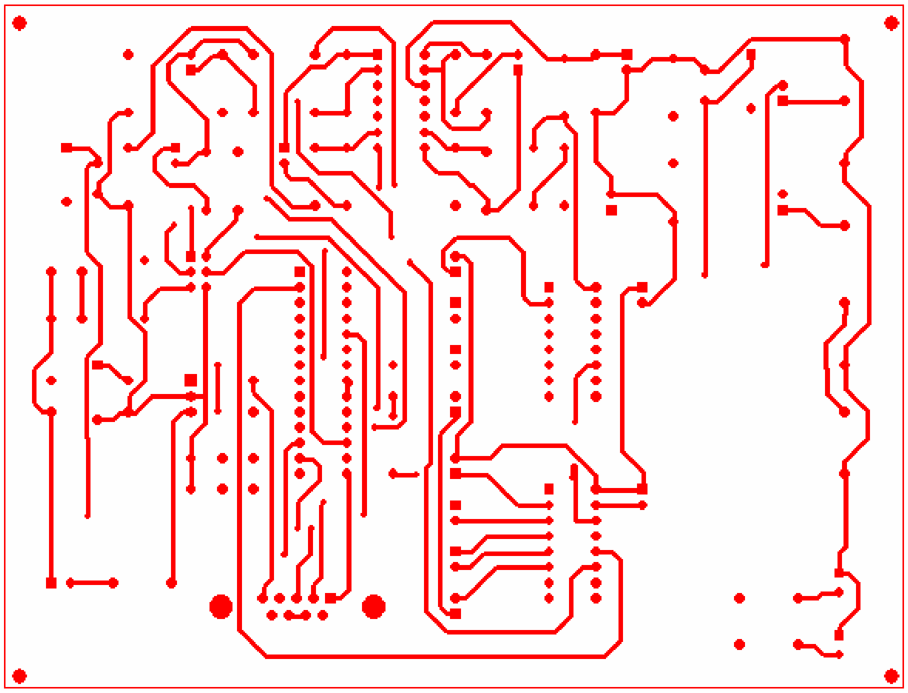


FIGURE B.4 PROXIMITY DETECTOR BOARD TOP LAYER



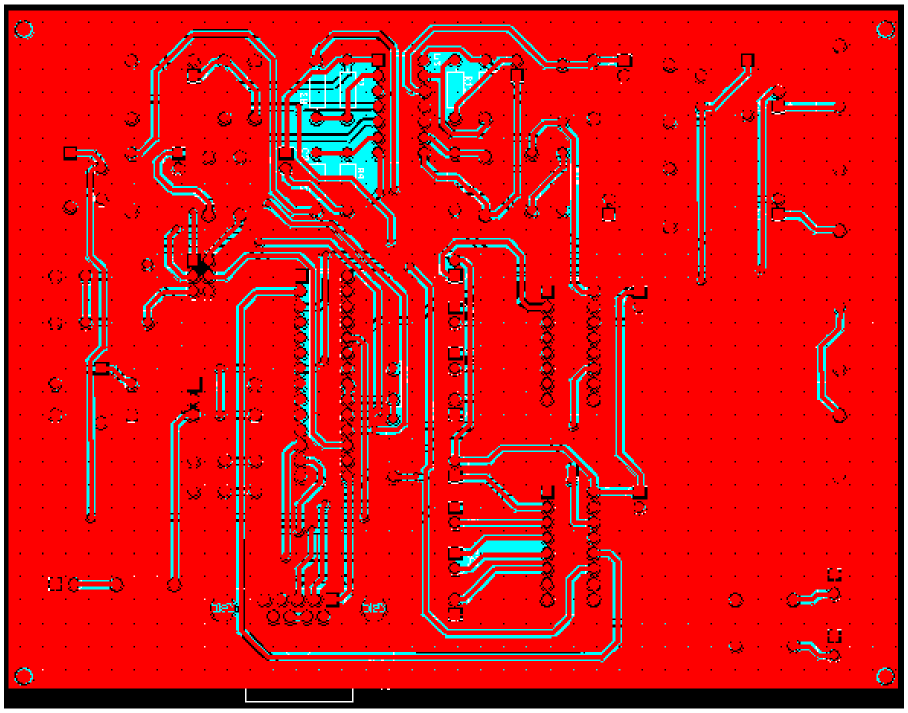
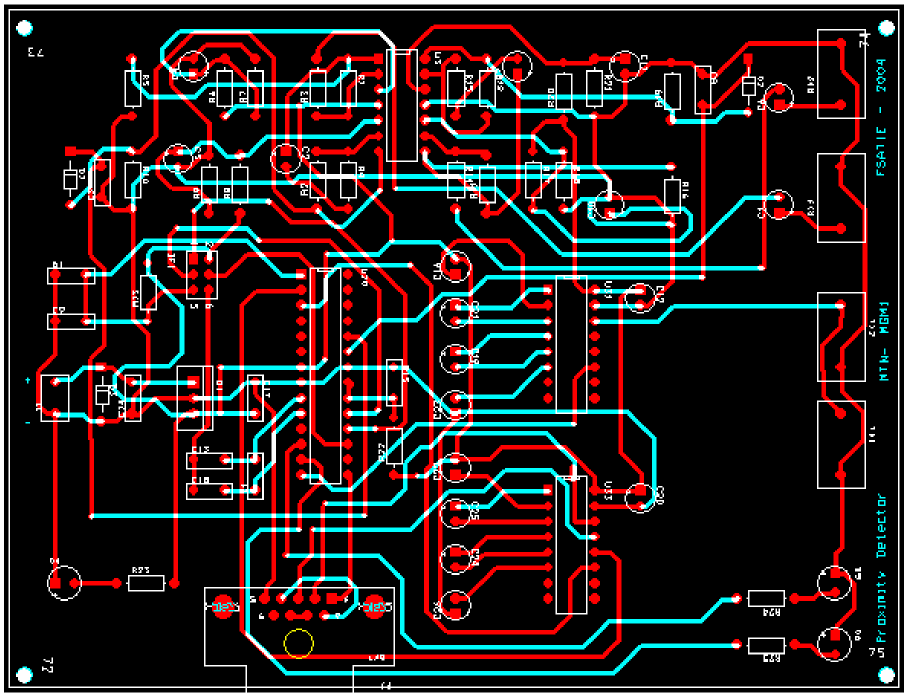


FIGURE B.5 PROXIMITY DETECTOR BOARD BOTTOM LAYER



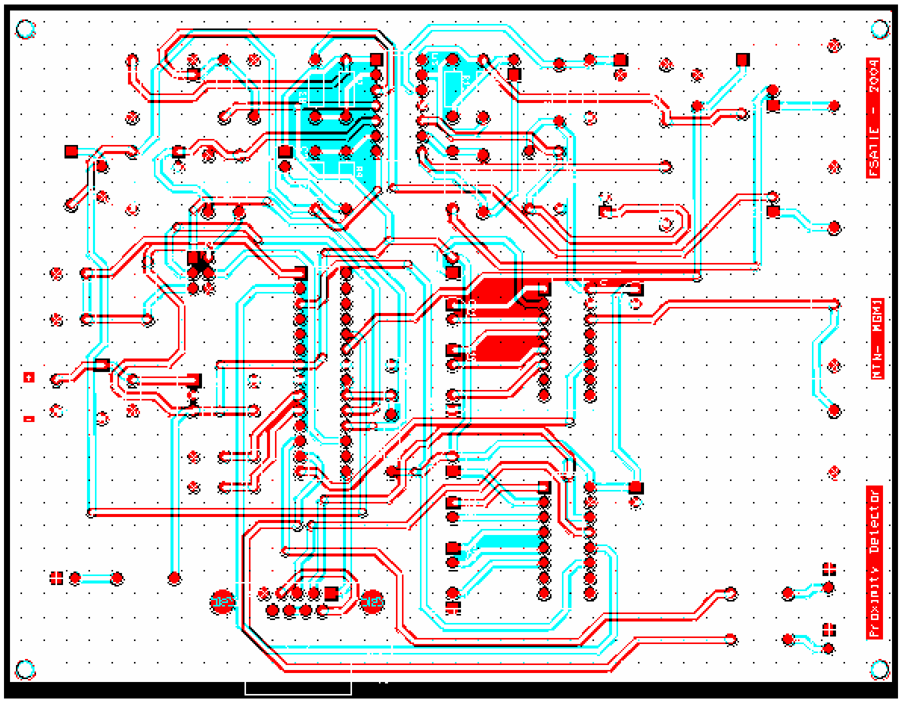


FIGURE B.6 COMPLETE PROXIMITY DETECTOR BOARD LAYOUT

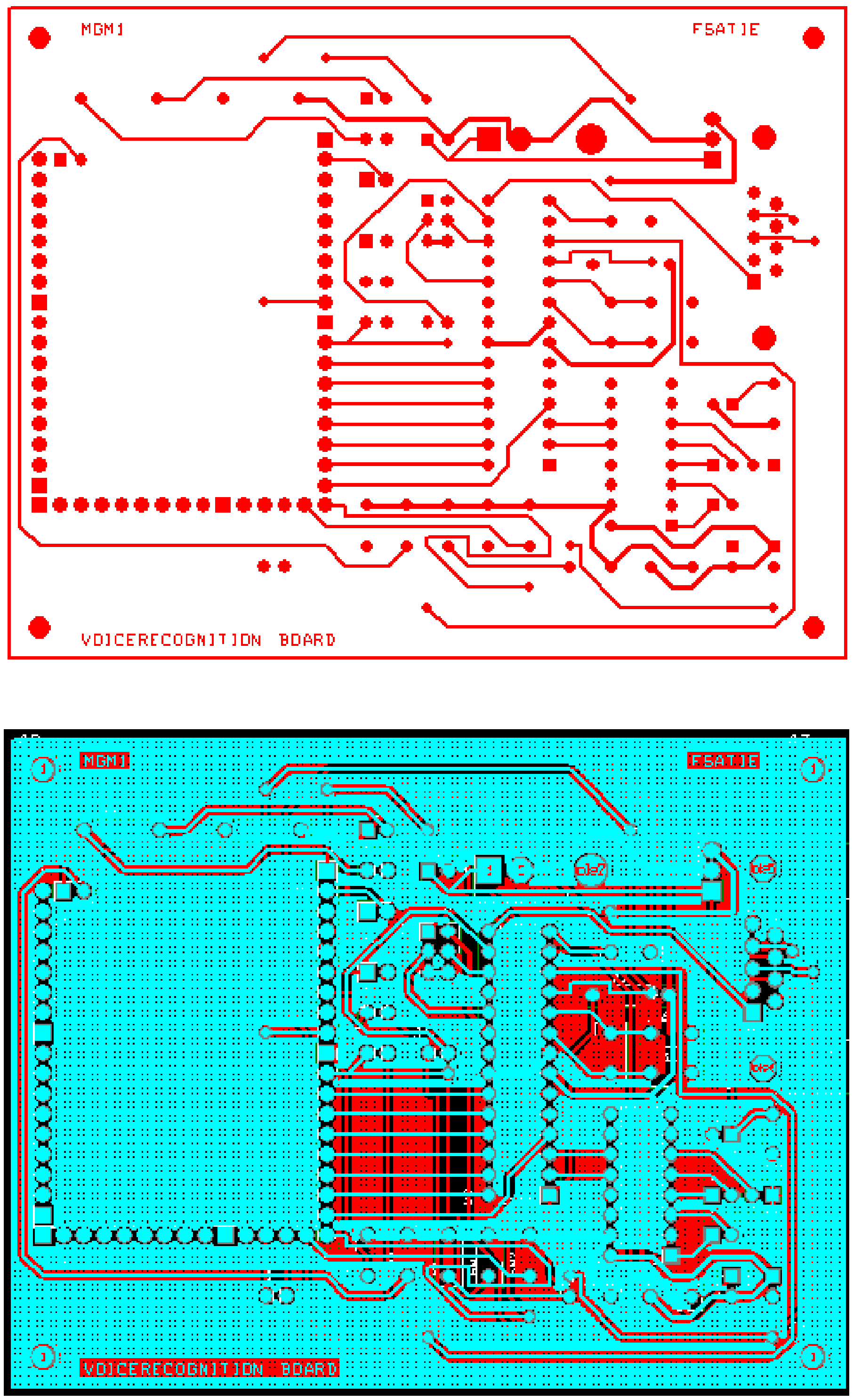
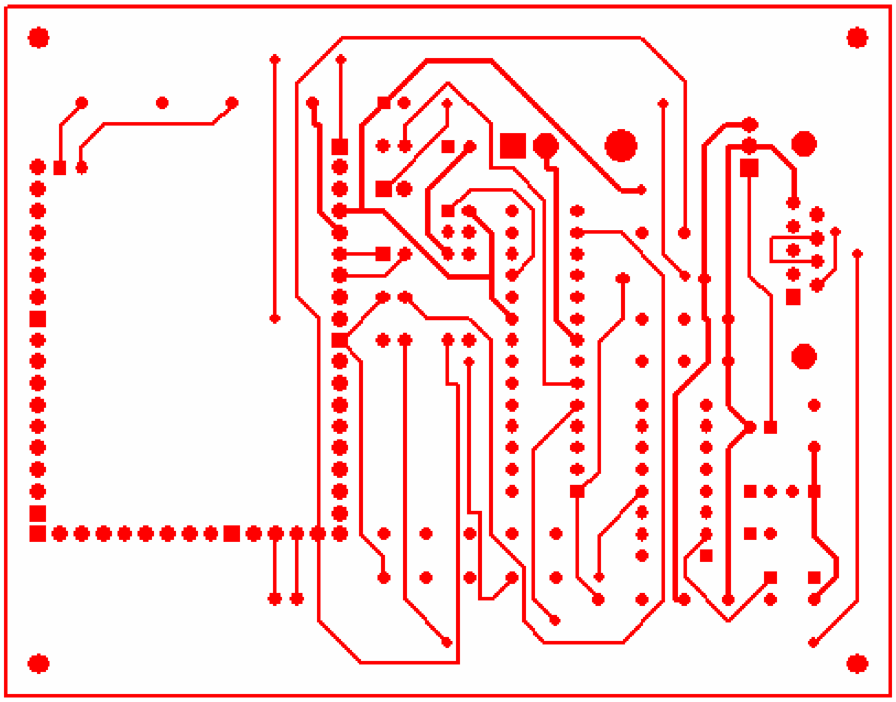


FIGURE B.7 VOICE RECOGNITION BOARD TOP LAYER



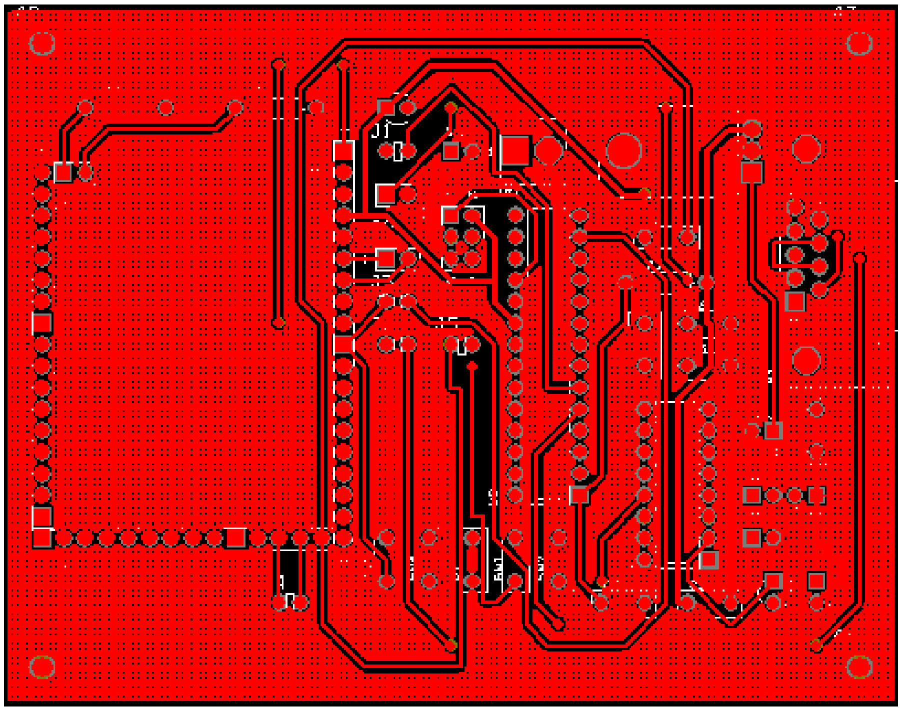
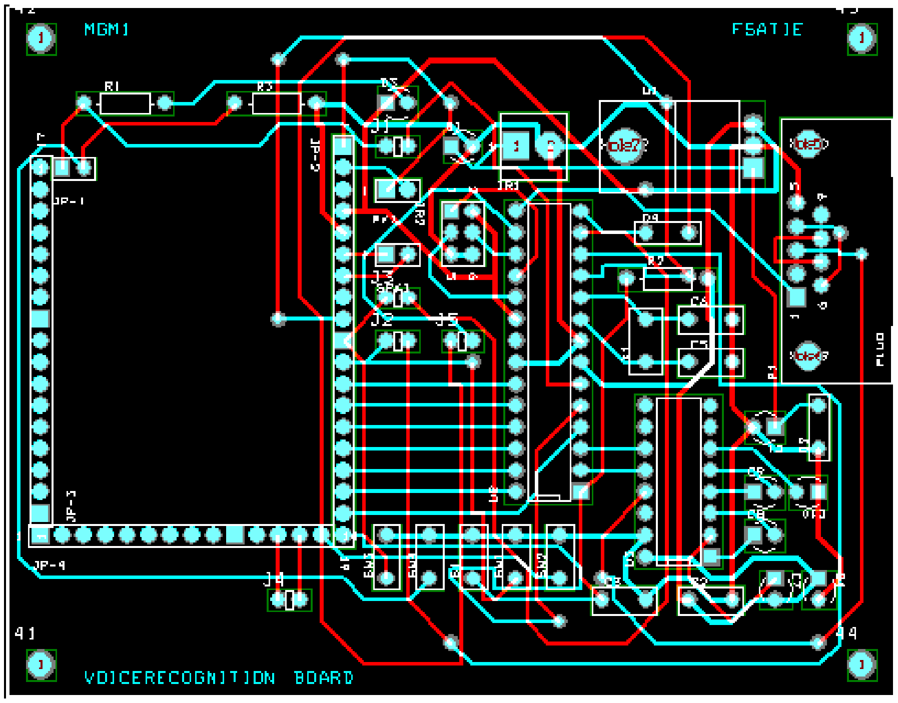


FIGURE B.8 VOICE RECOGNITION BOARD BOTTOM LAYER



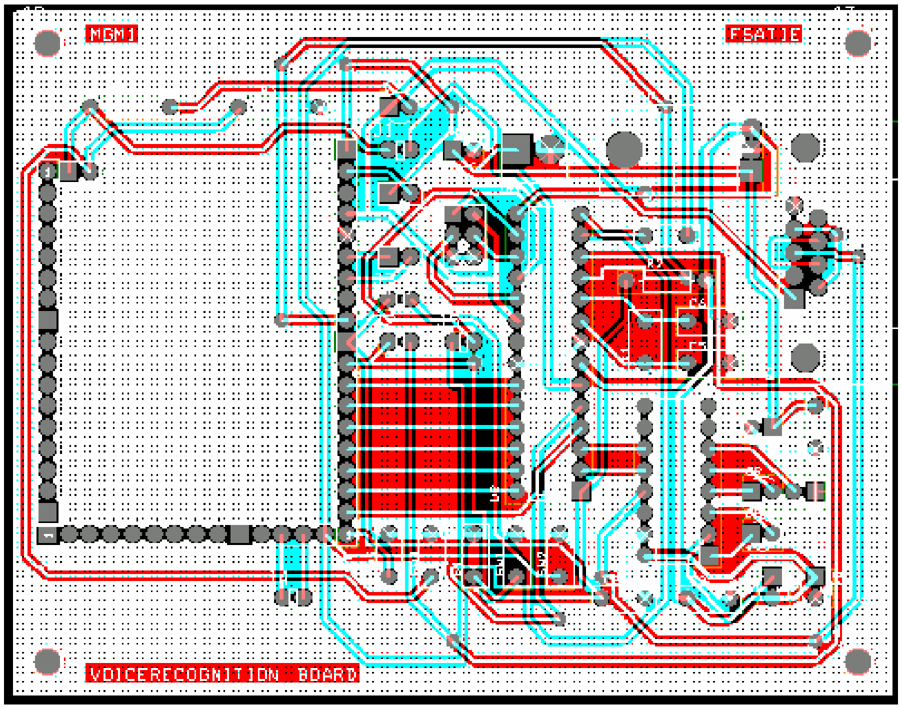
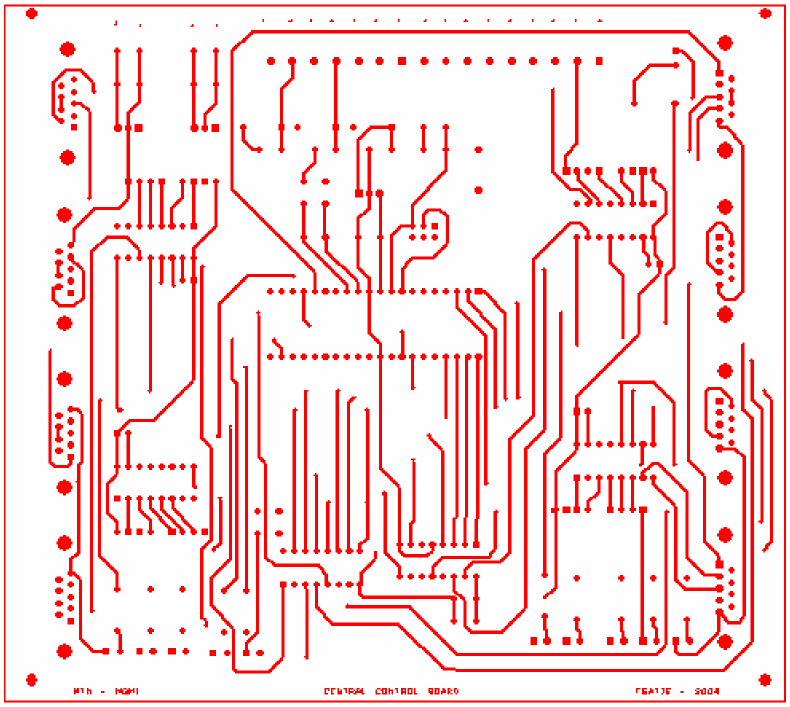


FIGURE B.9 COMPLETE VOICE RECOGNITION BOARD



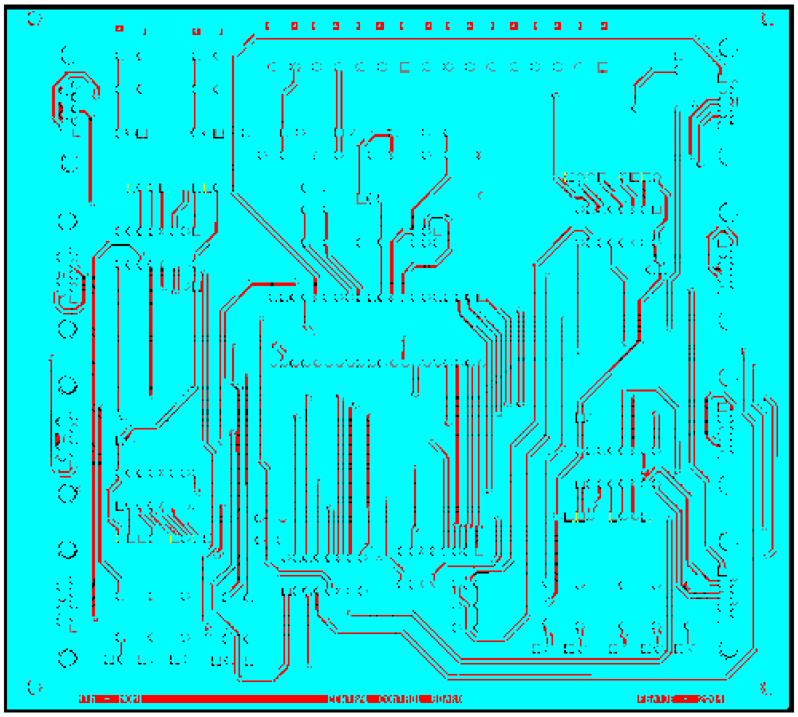
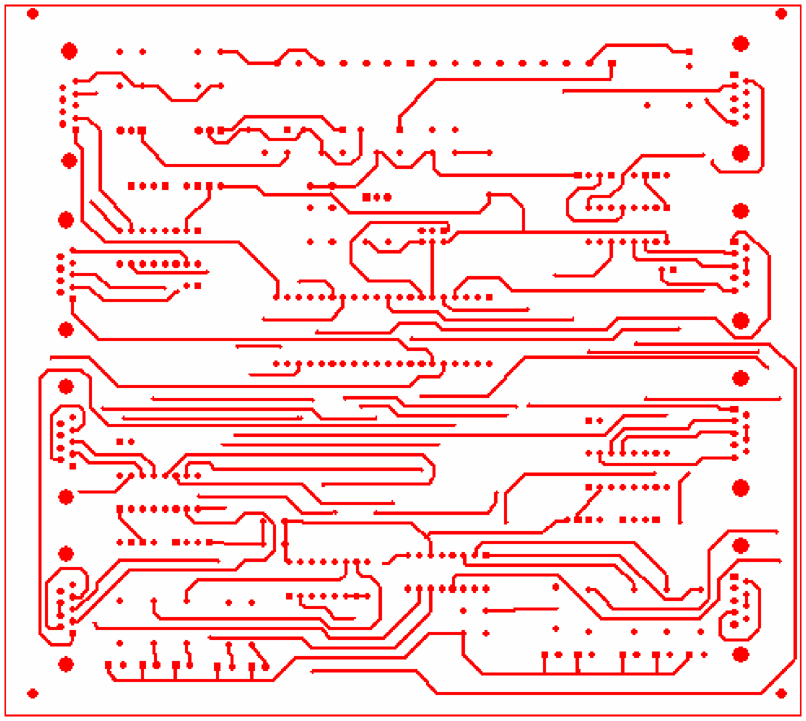


FIGURE B.10 CENTRAL CONTROL BOARD TOP LAYER



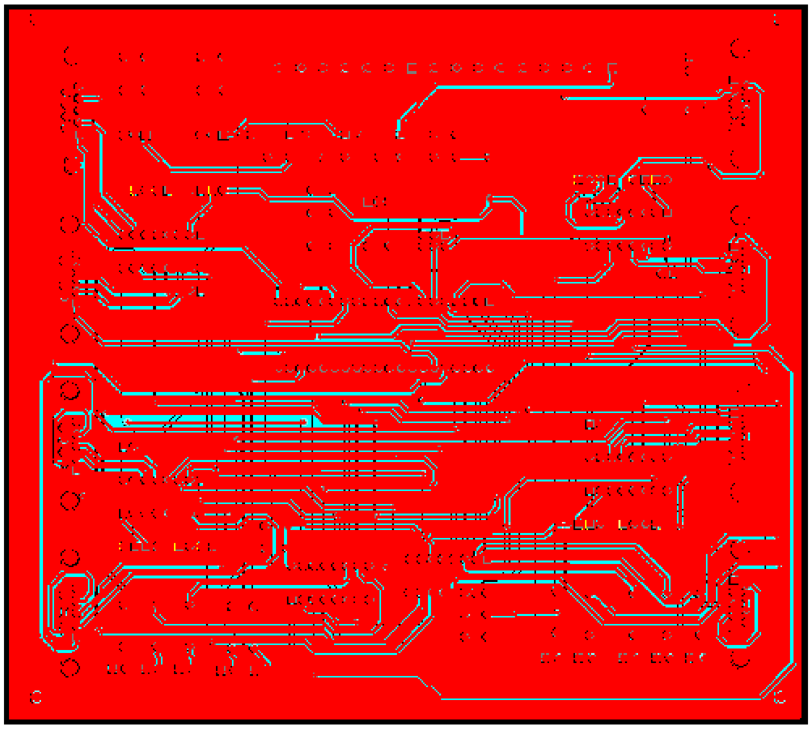
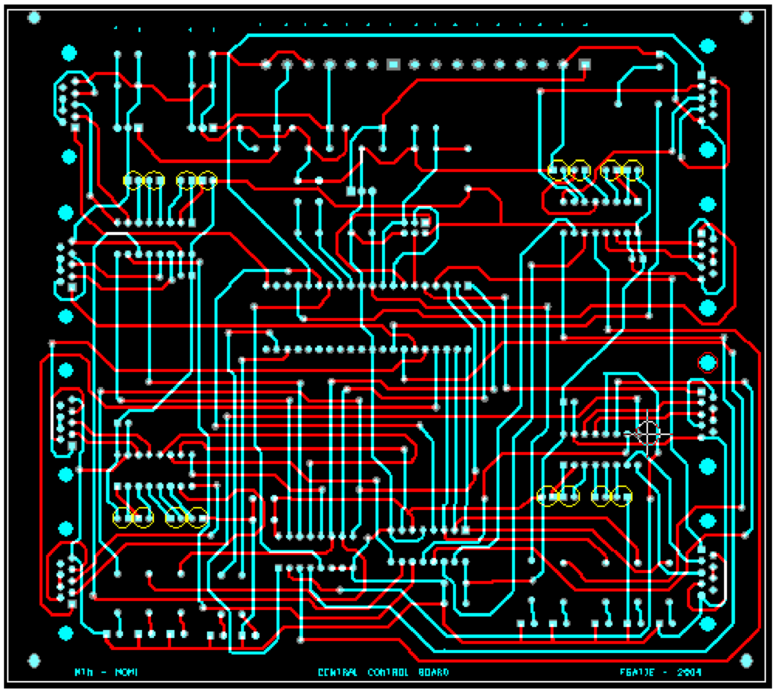


FIGURE B.11 CENTRAL CONTROL BOARD BOTTOM LAYER



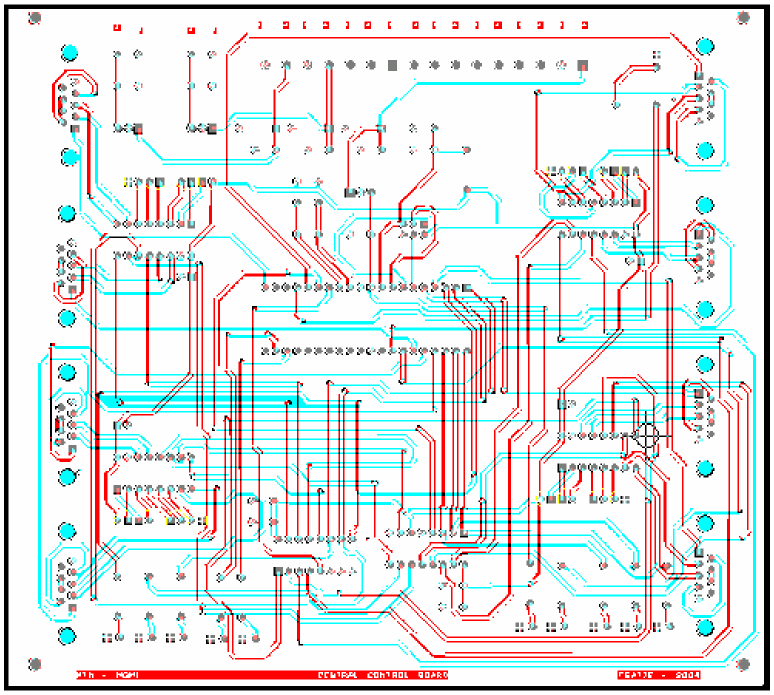


FIGURE B.12 COMPLETE CENTRAL CONTROL BOARD LAYOUT

APPENDIX C

C.1 Motor control board embedded software

```
//======================================================================
//Company: F'satie
//Project: Design and Implementation of a GSM-GPRS controlled robot
//Author: F. D Makaya Ondengue
//Simulator: AVR studio
//Compiler: ImageCraft
//Date: 30/06/2004
//Version: 0.0.1
//======================================================================
//This code has been developed to drive the main wheels of the robot.
//MGM1 has two driven wheels and two castor wheels. The steering of
//the system is done by this software.
//======================================================================
//This controls the motor driving circuit board which has two
//relay transistors and one MOSFET. Two inputs to 2 switching
//transistors to control the running direction (forward or backward)
//and one input to the MOSFET generating a PWM for soft start and stop
//for the motors. In this specific case, turning left and right is
//achieved by running the left or right motor while the other is off.
//======================================================================
#include <macros.h>
#include <iom8.h>

void InitUART(unsigned int baudrate);
void Init_PortB ();
void ReceiveByte();
void TransmitByte(unsigned char data);
void CompareByte();
void Run_right_motor(unsigned int pwm);
void Run_right_motor_short(unsigned int pwm1);
void Run_Half_distance(unsigned int pwm2);
void Run_Third_distance(unsigned int pwm3);
void Run_Quater_distance(unsigned int pwm4);
//void Turn_Left();
//void Turn_right();
void Go_forward();
void Go_backward();
void Delay();
unsigned char Rxdata;
unsigned char Txdata;
//unsigned int baudrate = 23;
unsigned int delay,delay1;
unsigned int width;
unsigned int Active;
unsigned int Direction_1, Direction_2;
long count;

//=======================================================================
/*Main program*/
//=======================================================================

void main (void)
{
       InitUART(23);    //Set baudrate to 9.6Kb/s using a 4MHz crystal
       Init_PortB ();   //Initialize PB0-PB3 as inputs and PB4-PB7 as
outputs
       for (;;)
       {
       ReceiveByte ();              //Receive data from main controller
      TransmitByte (Rxdata);              //Transmit back the received
       CompareByte();
       }
}

//=======================================================================
/*Initialization of the USART*/
//=======================================================================
void InitUART (unsigned int baudrate)
{
 UBRRH = (unsigned char)(baudrate>>8);    //Set the baud rate
 UBRRL = (unsigned char)baudrate;
 //UBRR = baudrate;
 //UCSRB = (UCSRB | 0x018);
 UCSRB = ((1<<RXEN)|(1<<TXEN));           //Enable UART receiver and
transmitter
// UCSRB = ((1<<URSEL)|(1<<USBS)|(3<<UCSZ0));
 }

//======================================================================
/*Initialization of PortB and PWM*/
//======================================================================

void Init_PortB ()
{
DDRB = (DDRB | 0x02);               //Configure PB1 as output pins
DDRC = (DDRC | 0x03);               //Configure PC0-PC1 as output pins
TCCR1A =(TCCR1A | 0x0A1);           //Set 8 bit fast PWM and other
settings
TCCR1B =(TCCR1B | 0x0A);            //Set 8 bit fast PWM and clock
source select
}
//======================================================================
/*Read function*/
//======================================================================

void ReceiveByte()
{
             while (!(UCSRA & (1<<RXC)))     //Wait for incomming data
             ;
             Rxdata = UDR;
}

//======================================================================
/*Write function*/
```

```
//=======================================================================

void TransmitByte (unsigned char data)
{
       while (!(UCSRA & (1<<UDRE)))             //Wait for empty
transmit buffer
       ;
       UDR = data;                              //Start transmittion
}

//=======================================================================
/*This function decodes the message received from the PC: The function
compares the*/
/*received charactere with a list of known characteres in order to
decode the*/
/*the message behind it*/
//=======================================================================

void CompareByte()
{
 switch (Rxdata)
         {
          case 'O':
                  TransmitByte (Rxdata);  //Transmit back the received
                  Go_forward();                 //Set running direction
                  DDRB = (DDRB | 0x02);   //Configure PB1 as output pin
                  width = 0;  //Set the starting duty cycle of the PWM
                  Active = 1; /Flag for system activation when ON is
pressed.
            break;

            case 'G':
                  if (Active)
                  {

                   TransmitByte (Rxdata);
                   Run_right_motor(width);           //Call function
to ramp to full speed
```

```
                  }
            break;

            case 'L':
                  if (Active)
                  {
                   TransmitByte (Rxdata);
                   Run_right_motor_short(width);      //Call function
to ramp to full speed
                  }
            break;
            case 'R':
                  if (Active)
                  {
                   TransmitByte (Rxdata);
                  }
            break;
            case 'F':
                  if (Active)
                  {
                   TransmitByte (Rxdata);
                   Go_forward();          //Call function to go forward
                   Run_right_motor(width);            //Call function
to ramp to full speed
                  }
            break;
            case 'B':
                  if (Active)
                  {
                   TransmitByte (Rxdata);
                   Go_backward();   //Call function to go backward
                   Run_right_motor(width);            //Call function
to ramp to full speed
                  }

            break;

            case 'H':
                  if (Active)
                  {
                   TransmitByte (Rxdata);
                   if (Direction_1)
                   {
                    Go_forward();         //Call function to go forward
                   }
                   else
                   {
                   Go_backward();         /Call function to go backward
                   }
                   Run_Half_distance(width);          //Call function
to ramp to full speed
                  }
            break;

            case 'T':
                  if (Active)
                  {
                   TransmitByte (Rxdata);
                   if (Direction_1)
                   {
                    Go_forward();         //Call function to go forward
                   }
                   else
                   {
                   Go_backward();        //Call function to go backward
                   }
                   Run_Third_distance(width);         //Call function
to ramp to full speed
                  }
            break;



            case 'Q':
                  if (Active)
                  {
                   TransmitByte (Rxdata);
                   if (Direction_1)
                   {
                    Go_forward();         //Call function to go forward
                   }
                   else
                   {
                   Go_backward();        //Call function to go backward
                   }
                   Run_Quater_distance(width);        //Call function
to ramp to full speed
                  }
            break;


            case 'S':
                  if (Active)
                  {
                   TransmitByte (Rxdata);            //Transmit back
the received

                   DDRB = (DDRB & 0x0FD);           //Configure PB1 as
input pin: stop Pwm
                   Active = 0;            //Flag for system activation
when ON is pressed.
                  }
            break;

            default:
                  break;
            }
}


//=======================================================================
/*Function to slowly start, run and stop the motor for three seconds
during straight motion*/
//=======================================================================
void Run_right_motor(unsigned int pwm)
{
 for (pwm = 90; pwm<=255; pwm++)
//Increase pulse width until max of 255
      {
      OCR1A = pwm;                  //Assign to register for output
      for (count=0; count<=1000; count++)      //Time high
            {
            delay--;
            }
      }
      for (count = 0; count <= 500000; count ++)
            {
            delay--;
            }
  for (pwm = 255; pwm >0; pwm --)
      {
      OCR1A = pwm;
      for (count=0; count<=1000; count++)      //Time high
            {
            delay--;
            }
      }
}

//======================================================================
/*Function to slowly start, run and stop the motor for one second when
changing direction*/
//======================================================================
void Run_right_motor_short(unsigned int pwm1)
{
 for (pwm1 = 90; pwm1<=255; pwm1++)


//Increase pulse width until max of 255
      {
      OCR1A = pwm1;                 //Assign to register for output
      for (count=0; count<=1000; count++)      //Time high
```

```
            {
            delay--;
            }
      }
       for (count = 0; count <=125000; count ++)
            {
            delay--;
            }
  for (pwm1 = 255; pwm1 >0; pwm1 --)
      {
      OCR1A = pwm1;
      for (count=0; count<=1000; count++)       //Time high
            {
            delay--;
            }
      }
}

//======================================================================
/*Function to slowly start, run and stop the motor for one second when
changing direction*/
//======================================================================

void Run_Half_distance(unsigned int pwm2)
{
 for (pwm2 = 90; pwm2<=255; pwm2++)
//Increase pulse width until max of 255
      {
      OCR1A = pwm2;                  //Assign to register for output
      for (count=0; count<=1000; count++)       //Time high
            {
            delay--;
            }

      }
       for (count = 0; count <=250000; count ++)
            {
            delay--;
            }
  for (pwm2 = 255; pwm2 >0; pwm2 --)
      {
      OCR1A = pwm2;
      for (count=0; count<=1000; count++)       //Time high
            {
            delay--;
            }
      }
}

//======================================================================
/*Function to slowly start, run and stop the motor for one second when
changing direction*/
//======================================================================
void Run_Third_distance(unsigned int pwm3)
{
 for (pwm3 = 90; pwm3<=255; pwm3++)
//Increase pulse width until max of 255
      {
      OCR1A = pwm3;                   //Assign to register for output
      for (count=0; count<=1000; count++)       //Time high
            {
            delay--;
            }
      }
       for (count = 0; count <=167000; count ++)
            {
            delay--;
            }
  for (pwm3 = 255; pwm3 >0; pwm3 --)
      {
      OCR1A = pwm3;

      for (count=0; count<=1000; count++)       //Time high
            {
            delay--;
            }
      }
}



//======================================================================
/*Function to slowly start, run and stop the motor for one second when
changing direction*/
//======================================================================
void Run_Quater_distance(unsigned int pwm4)
{
 for (pwm4 = 90; pwm4<=255; pwm4++)
//Increase pulse width until max of 255
      {
      OCR1A = pwm4;                  //Assign to register for output
      for (count=0; count<=1000; count++)       //Time high
            {
            delay--;
            }
      }
       for (count = 0; count <=125000; count ++)
            {
            delay--;
            }
  for (pwm4 = 255; pwm4 >0; pwm4 --)
      {
      OCR1A = pwm4;
      for (count=0; count<=1000; count++)       //Time high
            {
            delay--;
            }
      }
}


//======================================================================
/*This function generates PWM to slow down the motor before changing
direction*/
//======================================================================
/*void Slow_down(unsigned int pwm1)
{
 for (pwm1; pwm1>=100; pwm1--)
      {
      OCR1A = pwm1;
      for (count=0; count<=5000; count++)
            {
            delay--;
            }
      }
 width = pwm1;
}

*/

/*void Halt (unsigned int pwm2)
{
 for (pwm2; pwm2>=1; pwm2--)
      {
      OCR1A = pwm2;
      for (count=0; count<=5000; count++)
            {
            delay--;
            }
      }
}
*/
/*
void Stop_right_motor (unsigned int pwm2)
{
 for (pwm2; pwm2>=1; pwm2--)
      {
      OCR1A = pwm2;

      for (count=0; count<=1000; count++)
            {
            delay--;
            }
      }
}
*/
//======================================================================
/*This function sets PB1 and clears PB2 in order to reverse*/
/*the supply across the motor so that it turns forward*/
//======================================================================
void Go_forward()
{
Direction_1 = 1;
PORTC = (PORTC | 0x01);            //Set PB4 to Open relay A switch
PORTC = (PORTC & 0x0FD);           //Clear PB5 to Close relay B switch
Direction_2 = 0;
}
//======================================================================
/*This function clears PB1 and sets PB2 in order to reverse*/
/*the supply across the motor so that it turns backwards*/
//======================================================================
void Go_backward()
{
Direction_2 = 1;
PORTC = (PORTC & 0x0FE);           //Clear PB4 to close relay A switch
PORTC = (PORTC | 0x02);            //Set PB5 to open relay B switch
Direction_1 = 0;
}
//======================================================================
/*This function stops the right motor to enable the robot to turn
left*/
//======================================================================
//void Turn_Left()
//{
//PORTB = (PORTB & 0x0EF);                            //Clear PB5 to
stop motor*/

//delay;                                                    //Delay*/
//PORTB = (PORTB | 0x010);                            //Set PB5 to
start motor*/
//}
```

```
//====================================================================
/*This function drives the left motor in order to turn the robot
right*/
//====================================================================
//void Turn_right()
//{
/*void Slow_down ();*/
//PORTB = (PORTB | 0x010);                            //Set PB5 to
drive the motor at full speed
//}

//====================================================================
 /*Delay*/
 /===================================================================
//void Delay()
//    {
//    unsigned char a, b, c;

//    for (a = 1; a<5; a++)
//          for (b = 1; b<5; b++)
                  //for (c = 1; c; c++)
//                ;
//    }
//====================================================================
//====================================================================
```

C.2 Voice recognition board embedded software

```
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Company: F'satie
//Project: Design and Implementation of a GSM-GPRS controlled robot
//Author: F. D Makaya Ondengue
//Version: 1.1.0
//Simulator: AVR studio
//Compiler: ImageCraft
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//This code has been developed to interact with the voice
//recognition module in order to identify and transmit in a
```

```
//character format to central control board, the words recognized
//by the module.
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

#include <macros.h>
#include <iom8.h>

void InitUART(unsigned char baudrate);
void Init_PortB ();
void TransmitByte(unsigned char data);
void Readword();
void Train_module();
unsigned char Rxdata;

/*Main program*/

void main (void)
{
       InitUART(25);                                       // Set the
baudrate to.... using a 3.686MHz crystal
       Init_PortB ();                                    //Initialize
PB0-PB3 as inputs and PB4-PB7 as outputs
       for (;;)
       {
        if (PORTD==32)
        {
         Readword();


            }
       else
       {
       Train_module();
       }
      }
/*Initialization of the USART*/
}
void InitUART (unsigned char baudrate)
{
```

```
       UBRRH = (unsigned char)(baudrate>>8);    //Set the baud rate
       UBRRL = (unsigned char)baudrate;
       UCSRB = (UCSRB | 0x018);
       //UCSRB = ((1<<RXEN)|(1<<TXEN));         //Enable UART receiver
and transmitter

}

/*Initialization of PortB and PWM*/

void Init_PortB ()
{
DDRB = (DDRB | 0x0F0);                                  //Configure PB4-PB7
as output pins
DDRC = (DDRC | 0x0FF);                                  //Configure PB4-PB7
as output pins
}

/*Read function*/

/*Write function*/

void TransmitByte (unsigned char data)
{
       while (!(UCSRA & (1<<UDRE)))        //Wait for empty transmit
buffer
       ;
       UDR = data;                                       //Start
transmittion
//     return;
}

/*This function decodes the message received from the PC: The function
compares the*/
/*received charactere with a list of known characteres in order to
decode the*/
/*the message behind it*/
```

```
void Readword()
{
 switch (PORTC)
         {
          case 0:
                  Rxdata = 'G';
                  TransmitByte (Rxdata);           //Transmit character
                  break;
          case 1:
                  Rxdata = 'F';
                  TransmitByte (Rxdata);           //Transmit character
            break;
            case 2:
                  Rxdata = 'B';
                  TransmitByte (Rxdata);           //Transmit character
                  break;
            case 3:
                  Rxdata = 'R';
                  TransmitByte (Rxdata);           //Transmit character
            break;
            case 4:
                  Rxdata = 'L';
                  TransmitByte (Rxdata);           //Transmit character
            break;
            case 5:
                  Rxdata = 'S';
                  TransmitByte (Rxdata);           //Transmit character
            break;
            case 6:
                  Rxdata = 'H';
                  TransmitByte (Rxdata);           //Transmit character
            break;
            case 7:
                  Rxdata = 'Q';
                  TransmitByte (Rxdata);           //Transmit character
            break;
            case 8:
                  Rxdata = 'T';
                  TransmitByte (Rxdata);           //Transmit character
            break;
/*          case 9:
                  Rxdata = '';
                  TransmitByte (Rxdata);           //Transmit character
            break;
            case 10:
                  Rxdata = 'B';
                  TransmitByte (Rxdata);           //Transmit character
            break;
            case 11:
                  Rxdata = 'B';
                  TransmitByte (Rxdata);           //Transmit character
            break;
            case 12:
                  Rxdata = 'B';
                  TransmitByte (Rxdata);           //Transmit character
            break;
            case 13:
                  Rxdata = 'B';
                  TransmitByte (Rxdata);           //Transmit character
            break;
            case 14:
                  Rxdata = 'B';
                  TransmitByte (Rxdata);           //Transmit character
            break;
            case 15:
                  Rxdata = 'B';
                  TransmitByte (Rxdata);           //Transmit character
            break;
*/
            default:
                  break;
            }
}

void Train_module ()
{
```

```
PORTD = (PORTD & 0xvalue);                                    //Put a low on
train

}
```

C.3 Central control board embedded software

```
//**********************************************************************
//Project: Design and Implementation of a GSM-GPRS controlled robot
//Author: F. D Makaya Ondengue
//Company: F'satie
//Simulator: AVR studio
//Compiler: ImageCraft
//Date: 30/06/2004
//Version: 001
//**********************************************************************
//This code is written for the central control board in the control
// architecture of the system. The board plays the role of a serial
// port expander as well as the role of a router to channel the
// incoming data to the right destination. Three ports are interrupt
//triggered and five are polled. So this code is continually polling
//the different pins to check if one of the ports wants to transmit
//data, it will then enable that specific channel to do
//so. It analyzes that data if necessary, and decide whether to passe
//the data to the target board or ignore it.
//**********************************************************************
#include <macros.h>
#include <iom16v.h>
#include <stdio.h>
//**********************************************************************
//Interrupt handlers and vectors declarations
//**********************************************************************

//#pragma interrupt_handler   INT0_handler:2
//#pragma interrupt_handler   INT1_handler:3
```

```
//**********************************************************************
//Functions and variables declarations
//**********************************************************************

//void Interrupt_setup ();
void InitUART(unsigned int baudrate);
void Init_Ports ();
void ReceiveByte();
void TransmitByte(unsigned char data);
/*void Motion_decision();
void Motion_decision();
*/
unsigned char Command;
unsigned char Rxdata;
unsigned char Tele_status;
unsigned char Obstacle_ahead = 0;

//**********************************************************************
//This interrupt handler receives the obstacle position from the
//proximity detector and store it in a variable
//**********************************************************************

/*void INT0_handler(void)
       {
         PORTA = (PORTA & 0x00);                //clear port A
         PORTA = (PORTA | 0x00);                //Select channel 0
         ReceiveByte();
         Obstacle_ahead = Rxdata;
       }

//**********************************************************************
//Interrupt handler to receive commands from the Laptop and passes them
//to the motors
//**********************************************************************

void INT1_handler(void)
      {
         PORTA = (PORTA & 0x00);                //Clear portA
         PORTA = (PORTA | 0x01);                //Select channel 1
         ReceiveByte();
         Tele_status = Rxdata;
         PORTB = (PORTB & 0x00);
         PORTB = (PORTB | 0x01);                //Select channel 1
         TransmitByte(Tele_status);

      }
*/
//**********************************************************************
//Main program
//**********************************************************************

void main (void)
 {
 InitUART(23);          //Baudrate = 9.6kbps using a 3.66MHz crystal
 Init_Ports ();         //PB0-PB3 as inputs and PB4-PB7 as outputs

 for (;;)
 {

        PORTB = (PORTB & 0x00);
        PORTB = (PORTB | 0x02);     //Select channel 2

        PORTA = (PORTA & 0x00);
        PORTA = (PORTA | 0x02);     //Select channel 2 Demultiplexer

        ReceiveByte();
        TransmitByte (Rxdata);

        PORTA = (PORTA & 0x00);
        PORTA = (PORTA | 0x03);     //Select channel 4 Demultiplexer
        PORTB = (PORTB & 0x00);
        PORTB = (PORTB | 0x03);     //Select channel 4 multiplexer

        TransmitByte (Rxdata);
        PORTA = (PORTA & 0x00);
        PORTA = (PORTA | 0x03);     //Select channel 4 Demultiplexer
        PORTB = (PORTB & 0x00);
        PORTB = (PORTB | 0x03);     //Select channel 4 multiplexer

        TransmitByte (Rxdata);

        PORTA = (PORTA & 0x00);
        PORTA = (PORTA | 0x04);     //Select channel 4 Demultiplexer
        PORTB = (PORTB & 0x00);
        PORTB = (PORTB | 0x04);     //Select channel 4 multiplexer
        TransmitByte (Rxdata);

   }
}
//*********************************************************************
//Initialization of the USART
//*********************************************************************

void InitUART (unsigned int baudrate)
{

 UBRRH = (unsigned char)(baudrate>>8);    //Set the baud rate
 UBRRL = (unsigned char)baudrate;
 UCSRB = ((1<<RXEN)|(1<<TXEN));           //Enable UART receiver and Tx
// UCSRB = ((1<<URSEL)|(1<<USBS)|(3<<UCSZ0));
}
//*********************************************************************
//Initialization of PortB and PWM
//********************************************************************

void Init_Ports ()
{
 DDRA = (DDRA | 0x07);              //Configure PA0-PA2 as output pins
 DDRB = (DDRB | 0x07);              //Configure PB0-PB2 as output pin
 DDRD = (DDRD | 0x03);              //Configure PD1-PD0 as output pins
```

```
}

//*********************************************************************
//Function to receive data from the serial port
//*********************************************************************

void ReceiveByte()
{
             while (!(UCSRA & (1<<RXC)))  //Wait for incomming data
             ;
             Rxdata = UDR;
}

//*********************************************************************
//Function to transmit data through the serial port
//*********************************************************************

void TransmitByte (unsigned char data)
{
       while (!(UCSRA & (1<<UDRE)))  //Wait for empty transmit buffer
       ;
       UDR = data;                   //Start transmittion
}

//*********************************************************************
//Function to check whether an obstacle was already reported on the
//direction of the incoming command or not and then clears the the
//obstacle position flag. If an obstacle was reported in that
//direction, it ignores the command and if no obstacle was reported, it
//transmits the command to the right motor control board by selecting
//the right channel.
//*********************************************************************
/*
void Motion_decision()
{
      if (Obstacle_ahead != Command)
       {
        PORTA = (PORTA & 0x00);
        PORTA = (PORTA | 0x03);                 //Select channel 3
        TransmitByte(Command);
        PORTA = (PORTA & 0x00);
        PORTA = (PORTA | 0x02);                 //Select channel 2
        TransmitByte(Command);
        PORTA = (PORTA & 0x00);
        PORTA = (PORTA | 0x04);                 //Select channel 4
        TransmitByte(Command);
        PORTA = (PORTA & 0x00);
        PORTA = (PORTA | 0x02);                 //Select channel 2
        TransmitByte(Command);
        Obstacle_ahead = 0;
       }
 }
//**********************************************************************
//
//**********************************************************************
```

C.4 PC main application (MGM1_ControlSoft)

```
unit MAIN;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, CPort, ExtCtrls, StrUtils, DelphiTwain;

type
  TForm1 = class(TForm)
  Timer1: TTimer;
  ComPort1: TComPort;
  ComPort2: TComPort;
```

```
    gpbModemSetup: TGroupBox;
    btnEnterPin: TButton;
    btnActivateModem: TButton;
    lblPinCode: TLabel;
    edtPinCode: TEdit;
    lblRemoteControl: TLabel;
    edtControlNumber: TEdit;
    grpRe: TGroupBox;
    btnClearList: TButton;
    Memo1: TMemo;
    btnEditCommandList: TButton;
    grpCameraScan: TGroupBox;
    btnPinReady: TButton;
    DelphiTwain1: TDelphiTwain;
    grpReceivedMessages: TGroupBox;
    grpNewPicture: TGroupBox;
    Memo2: TMemo;
    Memo3: TMemo;
    Image1: TImage;
    btnDeleteMessages: TButton;
    procedure FormCreate(Sender: TObject);
    procedure btnEditCommandListClick(Sender: TObject);
    procedure Port1RxChar(Sender: TObject; Count: Integer);
    procedure btnEnterPinClick(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure btnClearListClick(Sender: TObject);
//    procedure TTNotifyEvent(Sender: TObject);
    procedure btnActivateModemClick(Sender: TObject);
    procedure btnPinReadyClick(Sender: TObject);
    procedure TSourceNotify(Sender: TObject; const Index: Integer);
    procedure TOnSourceFileTransfer(Sender: TObject; const Index: Integer;
     Filename: TW_STR255; Format: TTwainFormat; var Cancel: Boolean);
    procedure TOnTwainAcquire(Sender: TObject; const Index: Integer;
      Image: TBitmap; var Cancel: Boolean);
    procedure btnDeleteMessagesClick(Sender: TObject);
//    procedure Button2Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
    Form1: TForm1;
    CDIRECTS : String;
    SMS : String;
    CELLNUMBER : String;
    COMMAND : String;
    COMMAND_EXIST : Boolean;
    Scan_flag: integer = 0;

implementation

uses
   CMD;//Include header files

{$R *.DFM}
const
  WM_ALWAYSONTOP = 99;

procedure TForm1.FormCreate(Sender: TObject);
```

```
//var
//MyHandle : THandle;
begin
   CDIRECTS := GetCurrentDir();

   DelphiTwain1.LibraryLoaded := TRUE;
   DelphiTwain1.SourceManagerLoaded := TRUE;
   DelphiTwain1.Source[0].ShowUI :=TRUE;
   DelphiTwain1.Source[0].TransferMode := ttmMemory;
   DelphiTwain1.Source[0].Loaded := TRUE;
   DelphiTwain1.Source[0].Enabled := TRUE;
   Memo1.Clear;
   Memo2.Clear;
   Memo3.Clear;
//    MyHandle := FindWindow(nil, 'Logitech camera');
//    SendMessage(MyHandle, 99, 0, 0);

end;

procedure TForm1.btnEditCommandListClick(Sender: TObject);
begin
   COMMANDS.Show;
end;

procedure TForm1.Port1RxChar(Sender: TObject; Count: Integer);
var
Str: String;
begin
   ComPort1.ReadStr(Str, Count);
   Memo2.Text := Memo2.Text + Str;
end;
```

```
procedure TForm1.btnEnterPinClick(Sender: TObject);
begin
   Memo2.Clear;
   ComPort1.WriteStr('AT+CPIN="' + edtPinCode.Text + '"' + #13#10);
//    ComPort1.WriteStr('ATQ=1' + #13#10);
   btnActivateModem.Enabled := True;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
var
F : Textfile;
S, RxMessage : String;
MessageCnt,I : Integer;
Comp : Integer;
Count : Integer;
Word, Greeting : String;
SenderName : String;
Talk : Integer;
Picture_Request : Boolean;
Pin_code: string;
begin
//    if (Scan_flag = 1)then
//    begin



     DelphiTwain1.LibraryLoaded := TRUE;
     DelphiTwain1.SourceManagerLoaded := TRUE;
     DelphiTwain1.Source[0].ShowUI :=TRUE;
     DelphiTwain1.Source[0].TransferMode := ttmMemory;
     DelphiTwain1.Source[0].Loaded := TRUE;
     DelphiTwain1.Source[0].Enabled := TRUE;
//      Scan_flag := 0;
//    end;
   S := '';
//    count:= 200;
//    Memo2.Clear;    //Clear memo2
//    Memo3.Clear;    //Clear memo3
   ComPort1.WriteStr('AT+CPMS?'+#13#10);//Check for new message
   Sleep(500);                   //wait for response
   ComPort1.ReadStr(S,  200);                //read the response string (+CPMS:
"ME",x,40,"SM",0,20,"ME",3,40)
   Sleep(100);
//    Showmessage(S);
   //    MessageCnt := StrToInt(copy(S, 10, 1));
//    T := copy(S, 24, 1);
    //  MessageCnt := StrtoInt(Trim(NthWord(S, ',',2)));//check the message count
fro the response string (x)
//    Showmessage(T);
   MessageCnt := StrToIntDef(copy(S, 24, 1), 0);
//    T := 'Lord';
//    Showmessage(T);



   //Sleep(100);
     if (MessageCnt = 1) then//if there is 1 new message then begin, else goto
***X***
   begin
      ComPort1.WriteStr('AT+CMGR=1'+#13#10); //send read message no: 1
command
      Sleep(700);                  //wait for response
      ComPort1.ReadStr(SMS, Count);      //read response string (AT+CMGR=1
                                //+CMGR:                                   "REC
UNREAD","+27731503857",,"04/02/13,11:46:03+08"
                                //Left
                                //OK
                                //**Now the message is in string 'SMS'
     word := 'Picture';
     Picture_Request := AnsiContainsText(SMS, word);
      if (Picture_Request = true) then                         //If A is true then
setup camera and take picture
     begin
        DelphiTwain1.Source[0].Enabled := False;
        DelphiTwain1.LibraryLoaded := TRUE;
        DelphiTwain1.SourceManagerLoaded := TRUE;
        DelphiTwain1.Source[0].ShowUI :=False;
        DelphiTwain1.Source[0].TransferMode := ttmFile;
        DelphiTwain1.Source[0].Loaded := TRUE;
        DelphiTwain1.Source[0].Enabled := TRUE;
//         DelphiTwain1.Source[0].Enabled := True;
//         DelphiTwain1.Source[0].ShowUI :=True;
//         T := 'Lord';
//         Showmessage(T);
     end;

       Memo2.Text := SMS;              //display the message in string 'SMS in
memo2
     ComPort1.WriteStr('AT+CMGD=1'+#13#10);//delete the new message in
MODEM
     Sleep(700);                //Wait for the message to be deleted
     //Memo3.Text := Trim(NthWord(SMS, "", 7)); //Display the received SMS
     //Memo3.Text := Trim(copy(SMS, 7,)); //Display the received SMS
     //Memo3.Text := Trim(Copy(SMS, 1, Length(SMS) - 2)); // Take out "OK"
```

```
      RxMessage := StrRScan(Pchar(SMS), '/');
      RxMessage := Trim(Copy(RxMessage, 17, length(RxMessage)));
//Isolate message containt
      RxMessage := Trim(Copy(RxMessage, 1, length(RxMessage) - 2));
//Suppress "OK"
      Memo3.Text := RxMessage;

//      Showmessage (SMS);
      CELLNUMBER := copy(SMS, 34, 12);
//      Showmessage(CELLNUMBER);
//              CELLNUMBER := Trim(copy(SMS, 21, length(SMS)));//get the
CELLNUMBER from the SMS string
      Comp := 1;                //Reset the COMPARE status integer
      Comp              :=              StrComp(PChar(CELLNUMBER),
PChar(edtControlNumber.Text));//Compare the CELLNUMBER with the number
in Edit4

      //Received message is from the remote control mobile phone

        if (Comp = 0) then             //if they are the same then begin, else goto
***Y***
      begin
           COMMAND := UpperCase(RxMessage); //get the COMMAND from the
SMS String
//         COMMAND := Trim(Copy(COMMAND, 1, Length(COMMAND) - 2));
            COMMAND_EXIST := False;              //reset the COMMAND_EXIST
status integer.
         Assignfile(F, CDIRECTS + '\COMMANDS.txt');//Assign the file
         Reset(F);                   //Reset the file to the beginning of the file
```

```
          while not eof(F) do                   //while it is not the end of the file read
every line
         begin
            Comp := 1;                    //reset the COMPARE status integer.
            Readln(F, S);//Read a line 'S' out of the file 'F'
            Comp := StrComp(PChar(COMMAND), PChar(S));//Compare the SMS
Command with the File Command
             if (COMP = 0) then//if the SMS command match with the file commnad
then
            begin
               COMMAND_EXIST := True;//the COMMAND exist
            end;
         end;
      Closefile(F);//we are done with the file, so close the file.
      Sleep(100);
      if (COMMAND_EXIST = True) then//if we found a match then start
      begin
         Memo1.Text := Memo1.Text + COMMAND + #13#10;//Display the
COMMAND
         Sleep(400);
         ComPort1.WriteStr('AT+CMGS="'+CELLNUMBER+'"'+#13#10);// send
start SMS command to the MODEM.
         Sleep(400);                           //Wait for '>' character.
         ComPort1.WriteStr('MGM1 VALID COMMAND
RECEIVED'+Char(26)+#13#10);//Send 'VALID COMMAND' message to
MODEM.
         Sleep(1000);
//          OpenFile(AnsiPchar(CDIRECTS + '\image.bmp'));
//Transmit the received command to the motor board

         Comport1.Close;
```

```
          Comport2.Open;
          Comport2.WriteStr(COMMAND);
          Sleep(400);
          Comport2.Close;
          Comport1.Open;
       end
       else //***B***
          if (COMMAND_EXIST = False) then//if we didn't find a match then, else
goto **B**
       begin
          ComPort1.WriteStr('AT+CMGS="'+CELLNUMBER+'"'+#13#10);//send
start SMS command to the MODEM.
          Sleep(400);                    //Wait for '>' character
          ComPort1.WriteStr('INVALID                         COMMAND
RECEIVED'+Char(26)+#13#10);//send 'INVALID COMMAND!' message to the
MODEM.
          Sleep(400);
          end;
       end //***Y***
       else
               //The coversation starts here!
       if (Comp <> 0) then
       begin
          SenderName := StrRScan(Pchar(SMS), '"');
          SenderName := Trim(Copy(SenderName, 8, length(SenderName) - 11));


           //SenderName := UpperCase(Trim(copy(SMS, 7, Length(SenderName) -
7))); //get the Sender Name from the SMS String
          //SenderName := Trim(Copy(SenderName, 6, Length(SenderName) - 7));
          //Showmessage(SenderName);
//           Showmessage(SenderName);
```

```
         Word := 'I AM'; //Trim(Copy('I AM',1,4));
         Greeting := UpperCase(Trim(copy(RxMessage, 1,4))); //get the SMS
String
          //Greeting :=Trim(Copy(Greeting,1,4));//get the fisrt 4 characters from the
SMS
         //Showmessage(Greeting);
         Talk :=1;
         Talk := StrComp(PChar(Word), PChar(Greeting));//Compare the SMS
Command with the File Command
         if (Talk = 0) then
         begin
            ComPort1.WriteStr('AT+CMGS="'+CELLNUMBER+'"'+#13#10);
            Sleep(1000);                        //Wait for '>' character.
             ComPort1.WriteStr('HELLO '+SenderName+' HOW ARE YOU? HOPE
YOU ARE ENJOYING THE SHOW'+Char(26)+#13#10);//Send 'Hello' message
to MODEM.
            Sleep(400);
         end;
         if (Talk <> 0) then
         begin
         //Showmessage(CELLNUMBER);
            ComPort1.WriteStr('AT+CMGS="'+CELLNUMBER+'"'+#13#10);
            Sleep(1000);                        //Wait for '>' character.
            ComPort1.WriteStr('THANKS FOR THE MESSAGE.
MGM1'+Char(26)+#13#10);//Send 'VALID COMMAND' message to MODEM.
            Sleep(400);
         end;
      end;

   end; //***X***
end;
```

```
procedure TForm1.btnClearListClick(Sender: TObject);
begin
  Memo1.Clear;
end;

//procedure TForm1.TTNotifyEvent(Sender: TObject);
//begin
    //Setting the webcam in scan mode
//   DelphiTwain1.LibraryLoaded := TRUE;
//   DelphiTwain1.SourceManagerLoaded := TRUE;
//   DelphiTwain1.Source[0].ShowUI :=TRUE;
//   DelphiTwain1.Source[0].TransferMode := ttmMemory;
//   Delp hiTwain1.Source[0].Loaded := TRUE;
//   DelphiTwain1.Source[0].Enabled := TRUE;
//end;


procedure TForm1.btnActivateModemClick(Sender: TObject);
begin
  if (edtControlNumber.Text = '') then
  MessageDlg('NO SENDER NUMBER', mtERROR, [mbOK], 0)
  else
  begin
    ComPort1.WriteStr('AT+CMGF=1' + #13#10);
    Sleep(500);
    ComPort1.WriteStr('AT+CPMS="ME"' + #13#10);
    Sleep(500);
    Timer1.Enabled := True;
    Memo2.Clear;
```

```
   end;

end;

procedure TForm1.btnPinReadyClick(Sender: TObject);
begin

  Timer1.Enabled := True;
  Memo2.Clear;
  Memo3.Clear;

end;



procedure TForm1.TSourceNotify(Sender: TObject; const Index: Integer);
begin
   delphitwain1.source[index].SetupFileTransfer(
   IncludeTrailingBackslash(getcurrentdir) +
   'image.bmp',tfBMP);

end;

procedure TForm1.TOnSourceFileTransfer(Sender: TObject;
  const Index: Integer; Filename: TW_STR255; Format: TTwainFormat;
  var Cancel: Boolean);
//  var T: string;
//  Scan_flag : integer;
  begin
//    ShowMessage('File saved to ' + filename);
   Image1.Picture.LoadFromFile(CDIRECTS + '\' + 'image.bmp');
//      Image1.Picture.Assign(Image);
//    DelphiTwain1.LibraryLoaded := TRUE;
//    DelphiTwain1.SourceManagerLoaded := TRUE;
//    DelphiTwain1.Source[0].ShowUI :=TRUE;
//    DelphiTwain1.Source[0].TransferMode := ttmMemory;
//    DelphiTwain1.Source[0].Loaded := TRUE;
//    DelphiTwain1.Source[0].Enabled := TRUE;
//    T := 'Lord';
//    Showmessage(T);
//      Scan_flag := 1;

   //    DelphiTwain1.Source[0].Enabled := True;
end;

procedure TForm1.TOnTwainAcquire(Sender: TObject; const Index: Integer;
  Image: TBitmap; var Cancel: Boolean);
begin
  //Copies the Image parameter to the TImage
  Image1.Picture.Assign(Image);
  //We only want the first image
  Cancel := TRUE;

end;

//procedure TForm1.Button2Click(Sender: TObject);
//var
//MyHandle : THandle;
//begin
//    MyHandle := FindWindow(nil, 'Logitech Camera');
//    SendMessage(MyHandle, WM_ALWAYSONTOP, 0, 0);

//end;
```

```
procedure TForm1.btnDeleteMessagesClick(Sender: TObject);
begin
    ComPort1.WriteStr('AT+CMGD=3'+#13#10);//delete the extra (third)
message in MODEM memory
    ComPort1.WriteStr('AT+CMGD=2'+#13#10);//delete the extra (second)
message in MODEM memory
end;

end.
```

APPENDIX D

D.1 Mechanical assembly

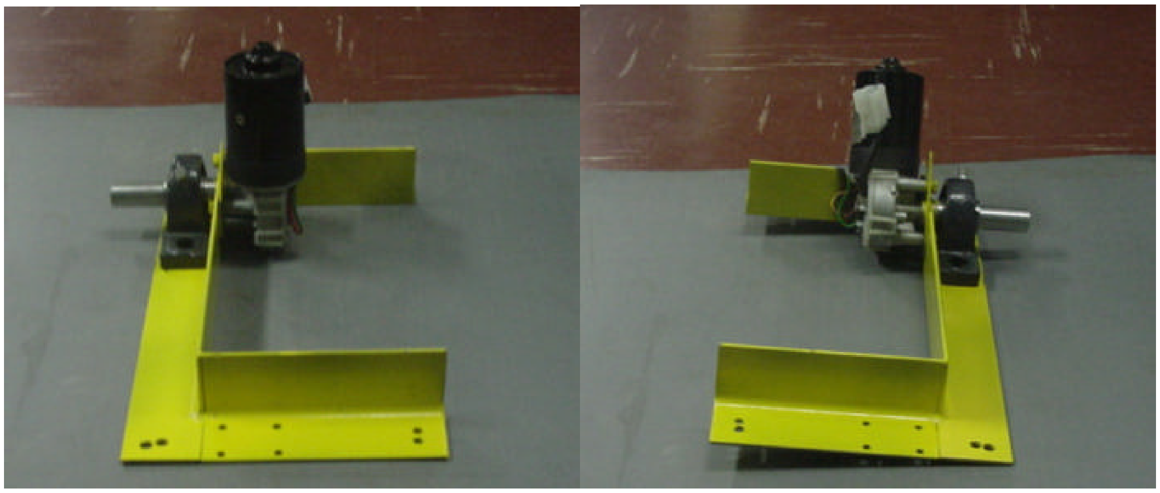


FIGURE D.1 PARTS OF THE CHASIS FRAME

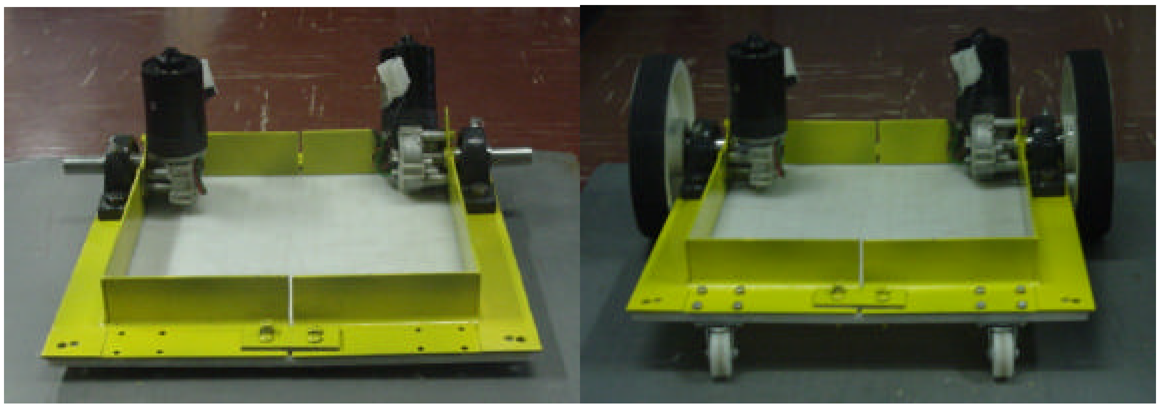


FIGURE D.2 CHASIS ASSEMBLY

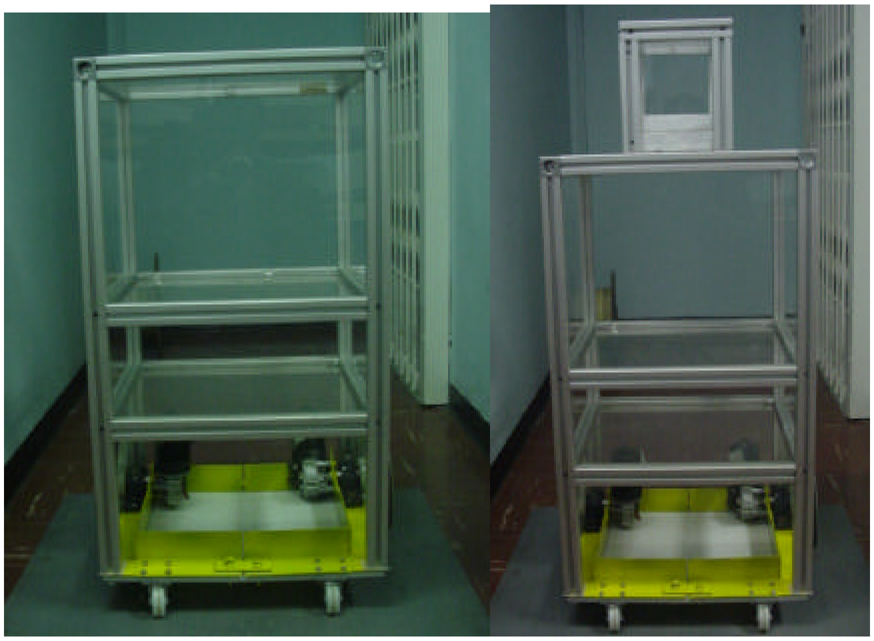


FIGURE D.3 MAIN FRAME ASSEMBLY

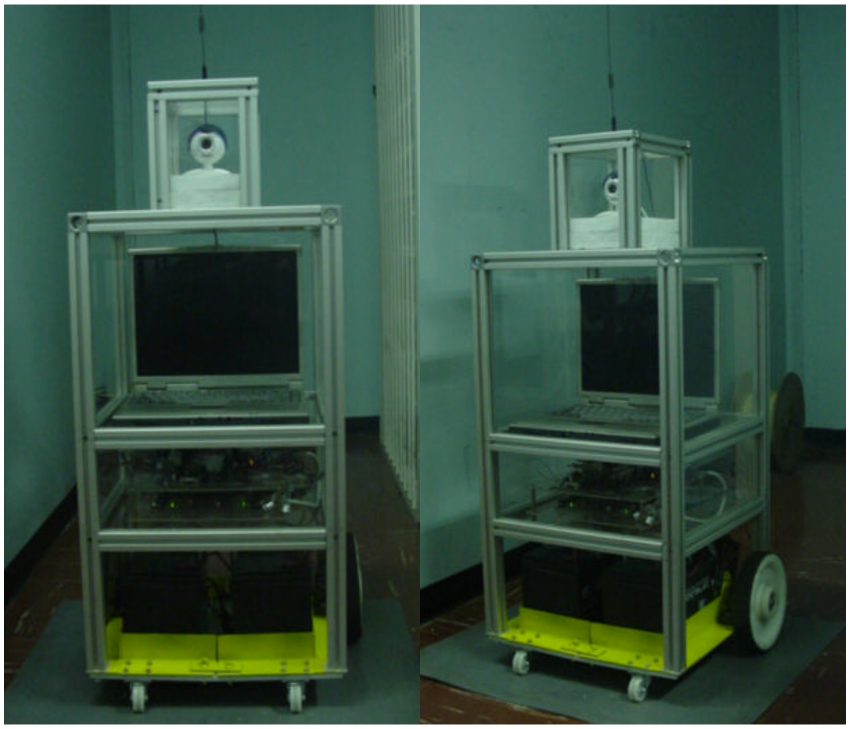


FIGURE D.4 INTEGRATED SYSTEM

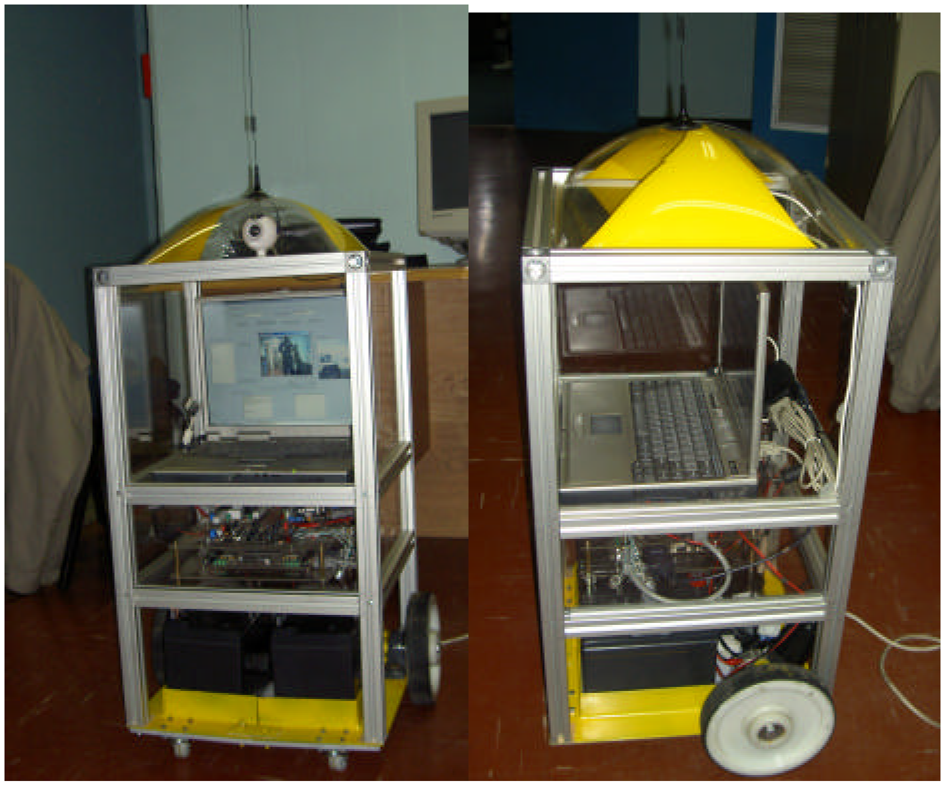


FIGURE D.5 FRONT AND SIDE VIEW OF FINAL PROTOTYPE

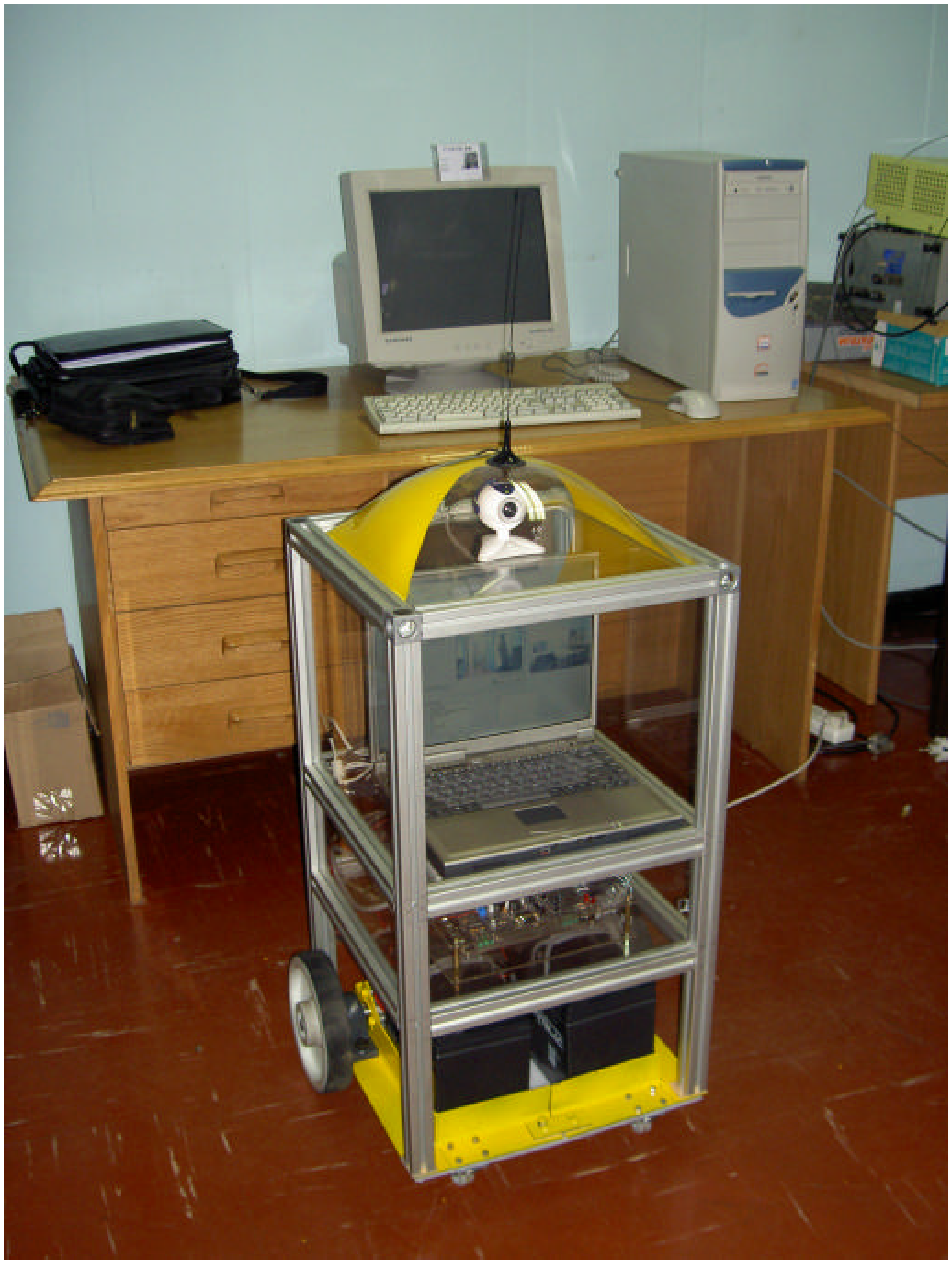


FIGURE D.6 MGM1 IN LAB 1

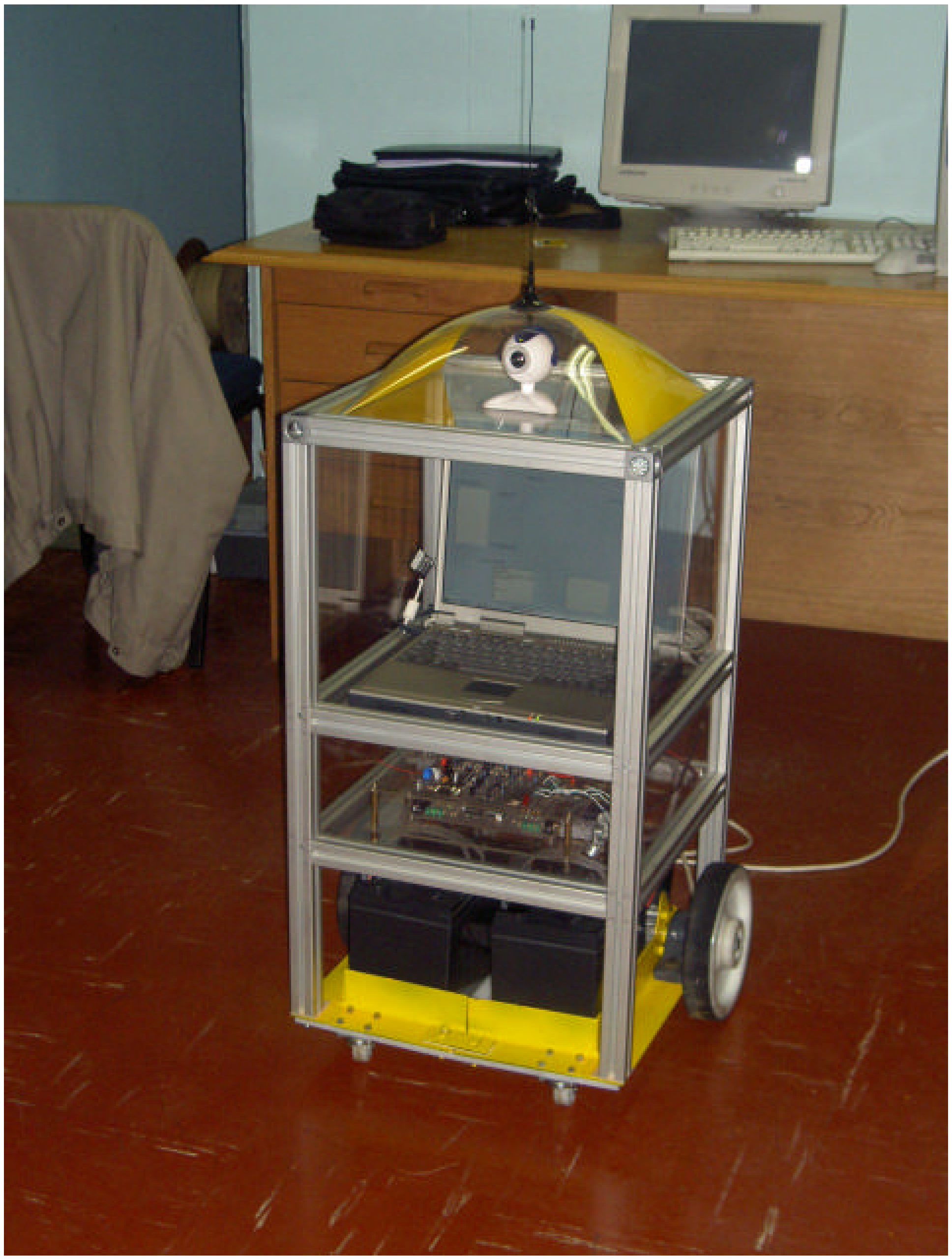


FIGURE D.7 MGM1 IN LAB 2

Design and Implementation of a GSM-GPRS Controlled Robot

Freddy Destin Makaya Ondengue