Palestine Polytechnic University

College of Engineering & Technology

Mechanical Engineering Department

Graduation Project

"Web-Based Mobile Robot Control system"

Project team

Ahmad Amro	In fulfillment For Bachelor Degree in Computer Engineering
Lama Amro	In fulfillment For Bachelor Degree in Communication Engineering
Sabreen Sbeitan	In fulfillment For Bachelor Degree in Communication Engineering
Abdallah Al-Barghothi	In fulfillment For Bachelor Degree in Mechatronics Engineering

Project Supervisors

Dr. Momen Sughayyer

Dr. Murad Abusubaih

Hebron- Palestine

June , 2010

Palestine Polytechnic University

Hebron- Palestine

College of Engineering & Technology

Mechanical Engineering Department

"Web-Based Mobile Robot Control System"

Project team

Ahmad Amro

Abdallah Basel Al-Barghothi

Lama Amro

Sabreen Sbeitan

Project Supervisors

Dr. Momen Sughayyer

Dr. Murad Abusubaih

Supervisor Signature

_____

Testing Team Signature

_____ _____ _____

Department Manager Signature

_____

Abstract

For the increasing interest in the remote controlling systems, especially through the internet, it was decided to work on an internet controlling system named Web-Based Mobile Robot Control System.

The system allows the user to control a mobile robot wirelessly through the internet, using software on a web page. This robot is semiautonomous, meaning that the user and the robot will share the control process, where the user controls and supervises the movement of the robot through sending new coordinates to the robot. And in the absence of the network coverage the robot will execute the orders according to an implemented intelligent procedure, this procedure will give, the robot the ability of avoiding obstacles and then proceed its primary objective. The robot will have an onboard wireless transceiver to allow control signal to flow from and to the robot, and it will have an on board camera, so that the user can get a video feed from the robot to supervise and explore the surrounding, The most important feature of this system is the mobility of the robot, the robot has no wires attached to it, portable power supply and an implemented intelligent.

The system has the ability to operate in ad-hoc mode, having only the robot system and a client device to perform the controlling operation, no need for access point coverage or internet access.

Dedication

إلى من رضاهم من رضا ربي ..

إلى من نور بالحنان دربي ..

أمي.....أبي

إلى كل اسم يبدأ بالألف و ينتهي بالياء ..

إلى الشموع التي احترقت لتضيء دربي ..

إلى العيون التي سهرت معي الليالي ..

إلى القلوب التي غمرتني بالمحبة ..

إلى الشفاه التي تمنت لي النجاح ..

إلى الذين رسموا بدمائهم حدود الوطن ..

الشهداء ... الذين لهم المجد يركع ..

إلى الذين يقبعون خلف القضبان .. عمالقة الثورة ..

إلى كل الأحبة والأصدقاء ..

إلى كل من سال دمه من جرحى و شهداء ..

الذين ما زالت فلسطين في اذهانهم ألى الأبد ..

إليهم جميعا نهدي هذا العمل ..

Acknowledgment

We wish to thank all instructors at the college of Engineering and technology for their efforts and advices during our five years of studying.

Special thanks for our supervisors *Dr.Momen sughayyer* and *Dr.Murad Subieh* for their knowledge, assistance and wise supervision.

Special thanks for *Eng.Khalid Al-Tmezi* and *Eng.Sami Al-Salamin* for their advices and assistance with instruments and tools inside and outside laboratories.

Special thanks for *Prof.Dr.Kareem Tahboub* for his wise advices, wide knowledge, useful resources and methodologies to complete and success our project.

And finally I would like to thank my friend, brother and godfather, *Eng.Ayman Sobih* for his assistance, and his wide and useful practical knowledge that helped me to complete my part, who without him I would not have reached this stage.

"Thank you Ayman from all the teams in this project, your friendship has increased our practical knowledge and made difference in our lives"

Special Words

Ahmad Amro

I would love to thank my team including my lovely fiancé Lama who was my best companion in this project and has been so in my life, Abdallah, who brought the idea of this project to life and his fiancé Sabreen, both were the best colleagues I could ever ask for.

Lama Amro

A special thank I would love to give to my colleagues, for being the best colleagues anyone can ask for. My fiancé Ahmad who has been a great supporter during this whole year, I would never have the ability to do such work without him being by my side, my family who did all what they could to support me in this five years long journey. I would thank my teachers who gave me a priceless wealth from which I will benefit my whole life.

Sabreen Sbeitan

I would like to thank my family for the support they provided me through my entire life, thanks to all my friends, huge thanks to my great team, in particular my fiancé and my best friend Abdullah without his love and encouragement I would not have done this work.

Table of Contents

List of Figures

List of Tables

List of Symbols

Symbole	Description
	current gain
n	number of batteries
x’	wheel speed in x direction
r	wheel raduis
θ	angel of rotation
v	linear velocity
´	angular velocity
R	circumference radius of the wheel.
¥	curvature
¤	initial orientation of angle
p	point in the space
q	vector
p'	jacobian expression
$\acute{}_{L}$	angular velocities of the left wheel
$\acute{}_{R}$	angular velocities of the right wheel
T_m	the torque developed by the motor
T_L	Load torque
T_d	friction torque
V_f	field voltage
V_a	armature voltage
I_f	field current
I_a	armature current
$V_b(s)$	back electromotive-force voltage

K_m	the motor constant
G(s)	system transfer function
$^{2}{}_{a}$	time constant of the armature
L_a	inductance of armature windings
R_a	resistance of armature windings
λ	angle contribution

Part Two Computer and Communication Part

List of Tables

Table of Figures

Part One

Mechatronics Part

Chapter One

Introduction

1.1 Overview

In recent years, the development of computer and reduction in size and cost of integrated circuits lead to create intelligent systems like mobile robot, which has a lot of mechatronics applications and uses; among them entertainment and educational purposes. In addition to that, wireless networks have become lately attractive to engineering applications, since the implementations of wireless networks have rabidly increased everyday in order to meet up the needs of wireless engineering applications which gives the advantage of mobility, reliability, and efficiency compared to any similar projects with wired networks.

One of the interesting application comes through the integration of mobile robot field with wireless field, so we can get a **Web-Based Mobile Robot Control System ,** which is the idea of this project, it meets the needs of mobility and controllability of the robot from anywhere in the world at any time through a web page.

The robot design is intended for entertainment and educational purposes since it has a small size, no wiring limitation, implemented intelligence, high mobility and maneuverability.

The robot in this project will be controlled and driven wirelessly through the internet by receiving coordinates (x, y) from the user, the robot system uses a built on camera to monitor the surrounding environment, where it sends the video feed to allow user to modify the movements of the mobile robot, and the onboard camera wirelessly through the internet using simple webpage software. In the absence of the wireless network coverage, the robot will be able to reach the given coordinates autonomously with the ability of avoiding obstacles using a group of ultra sonic sensors.

1.2 Project Idea

The project idea is to design and build an educational robot system that could be involved in local or international robotic competitions. The robot will be completely controlled by a human user; wirelessly through the internet (World Wide Web) using a web page, see figure (1.1) which describes the whole process.

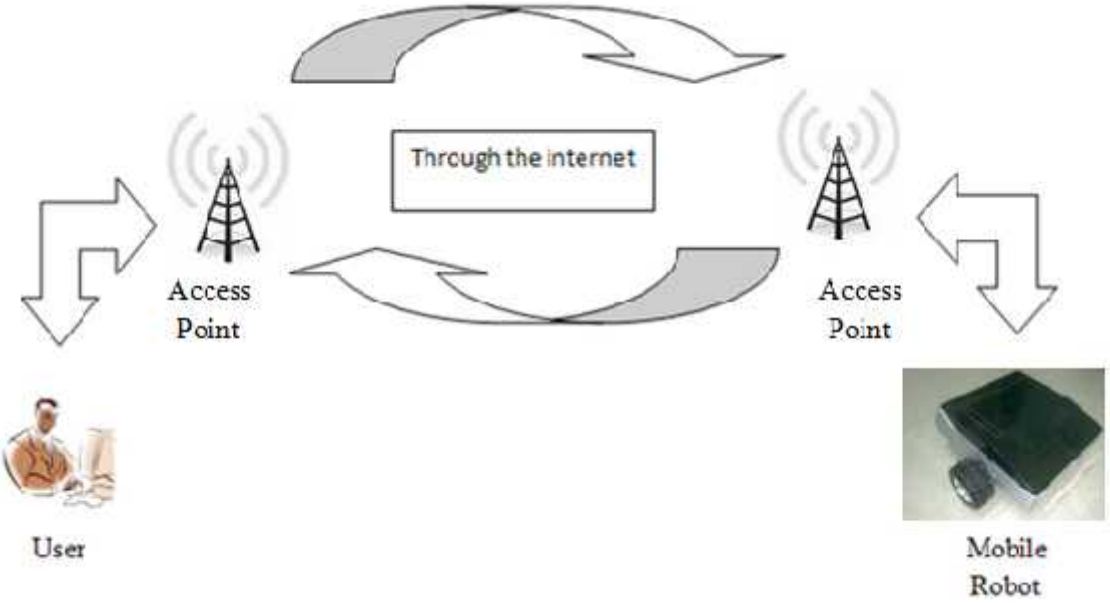


Figure (1.1) System General Diagram

We can say that this robot is somehow works like the nervous system in humans for many reasons; first of all, let us discuss how the human nervous system works in general.

Let us say that a human wants to move his arm, the brain sends moving orders to the arm muscles. These orders will not be directly transmitted to muscles, they have to go through a network of nerves that connects the brain with muscles, and then will be sent to the right muscle through a complex network of nerves in the arm itself; according to the direction that the brain wants, and finally an external feedback (Eyes) and internal feedback (nerves) will tell the brain whether the operation is completed or not.

But what if a flame source interrupted the arm while moving it in the absence of vision, will the muscle wait until this interruption is transmitted to the brain?!

In fact, no, the muscle will send this interruption to the spinal cord which will deal with it directly and sends new orders to move the arm away from this interruption, until new orders are received from the brain.

Now let us compare the operation of human nervous system with the robots operation. The user (brain) wants to move the robot (arm) to a specific direction and position (x, y); he will give the coordinates to the software (nerves) that he deals with on his personal computer to make the move, these orders are transmitted as control signals through the internet and networks (nerves) to the transceiver (nerves) that is implemented on the robot, then this transceiver passes the control signals to the PIC (complex nerves in the arm) which run the motors to reach the new coordinates, and a feedback data will be returned from the robot to the user all the way back through the internet to inform him if the operation is done or not. These data can be visual video feedback (Eyes) and electrical signals from the sensors (nerves) implemented on the robot.

Now for the interruption case, if the robot faces a disturbance while performing the operation like a wall for example in the absence of network coverage (no controllability or video feed), it will not wait until this interruption is transmitted to the user to make the right decision, it will follow a **Intelligent** procedure and send the disturbances directly from the sensors to the PIC controller (spinal cord) which will make the right decision for avoiding the obstacle and then return to its original path, and if the robot faces a dead end, it will follow a **Failsafe** procedure, meaning that it will hold still in its place until the network coverage is back again, so the user will make the right decision.

So this project improves the performance of the mobile robots through eliminating the limitation of wired connection which will allow the user to control the robot from any place in the world at any time. And also the ability of autonomous driving and obstacles avoidance through the implemented intelligence.

1.3 Methodology

Since this project is a mechatronics system, which is integration between multi-disciplinary sciences, it can be represented by figure (1.2), which shows the synergistic integration of three engineering fields. Three teams will be working individually and as a group on these fields to complete the project, they are a mechatronics engineer, communication engineer and a computer engineer.

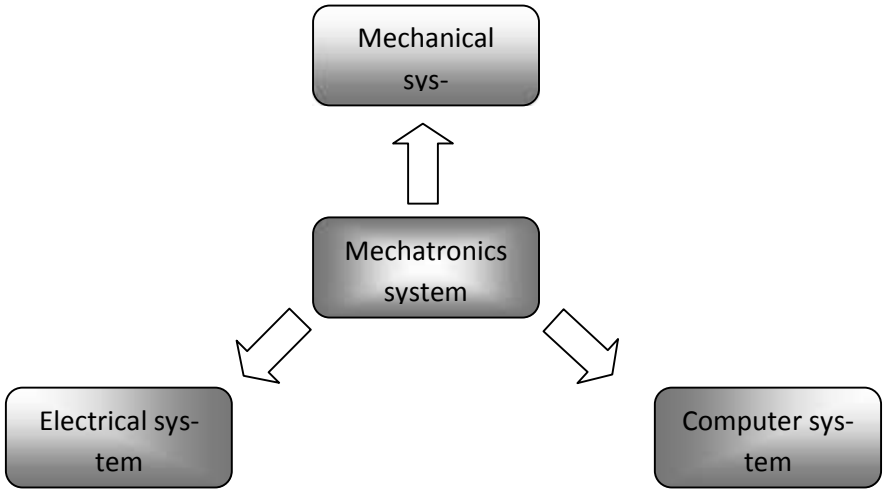


Figure (1.2) Synergic Integration

- Mechanical System:

  Designing and building the mechanical structure of the robot and choosing the appropriate dimensions and shape in order to fit in all the internal component of the robot and achieve the best mobility and maneuverability, also insuring a full traction between the wheels and the ground all the time, then designing the steering mechanism in order to get the best mobility, maneuverability and stability. This includes implementing the wheels, gears and motors.

- Electrical System (Control System):

  Implementing the motors for the movement of the robot and the onboard camera, also designing the control circuit of the robot to suit the application, and then implementing the battery back that will supply the entire component with power.

  Also building and programming the transceiver circuit which will allow the flow of control signals from the web into the robot and vice versa.

- Computer System:

  Implementing the micro-controller (PIC), program it with C language, and implement an intelligent avoiding obstacle procedure in it in order to give the robot the ability of avoiding obstacles, and then interface it with the electromechanical component and wireless transceiver.

1.4 Motivation

The system has some important points that come from the following:

- The big picture of this project is to be able to control a mobile robot wirelessly through the internet, despite the mechanical design simplicity and concentrating more on the idea of wireless mobile controlling.
- The design process which includes a multi-disciplinary teams of mechatronics, communication and computer sciences.
- The implementation of an intelligent web-based mobile robot.
- To provide an example bases for local or international robotic competitions.
- Can be used as a toy for entertainment.
- A programmable mobile robot that could be programmed easily according the designer ideas for many entertainment options.

1.5 Literature Review

1.5.1 Overview

According to Robotics Industrial Association (RIA),"a robot is a reprogrammable, multi-function, manipulator designed to move materials, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks."[1]

And the classification of robots can be According to the Japanese Industrial Robot Association (JIRA):

- Class 1: *Manual-Handling Device:* A device with multiple degrees of freedom that is actuated by an operator.
- Class 2: *Fixed-Sequence Robot:* A device that performs the successive stages of a task according to a predetermined, unchanging method and is hard to modify.
- Class 3: *Variable-Sequence Robot:* Same as class 2, but easy to modify.
- Class 4: *Playback Robot:* A human performs the task manually by leading the robot, which records the motions for later playback. The robot repeats the same motions according to the recorded information.
- Class 5: *Numerical Control Robot:* The operator supplies the robot with a movement program rather than teaching it the task manually.
- Class 6: *Intelligent Robot:* A robot with the means to understand its environment and the ability to successfully complete a task despite changes in the surrounding conditions under which it is to be performed.

Since the mobile robot in the project is controlled by a human operator, and also has intelligent to avoid obstacles, so it can be classified under Class 5 and 6.

Mobile robots could be found in many aspects of life and industrial applications, especially in tasks that are dangerous for human; since machines are less sensitive than people to radiation and toxic encountered during some tasks such as; repairing nuclear plant equipment and exploring or working in dangerous environment. In addition to this, mobile robots could be found in repetitive tasks such material handling systems, and storing systems. Add to this and that the scientific researches and entertainment purposes.

1.5.2 Application of Mobile Robot

1.5.2.1 Exploring and performing operations in Dangerous Environment

Some mobile robots are used for performing operations or exploring dangerous environment which humans can't perform or reach so instead of doing the job, they use mobile robots to perform operation from safety distance, and some of these robots are:

I. CASTOR– explosive ordinance disposal robot

Is a small, remote controlled mobile robot (cable and radio) designed to carry out inspection and intervention in hostile environments providing complete safety for the operator, figure (1.3). It is dedicated to handling and neutralizing explosives (EOD and IEDD) as well as carrying out operations in nuclear, bacteriological or chemical (NBC) fields. Operational 24 hours thanks to its battery pack that can be replaced within less than 10 seconds. Its small size allows it to operate easily in towns, airports and in the areas most difficult access: trains, tubes, buses, aircraft aisles.

Figure (1.3) Castor Robot

II. COBRA miniature inspection robot

Is a remote controlled mobile vehicle designed to perform inspection and neutralization in environments inaccessible by field operations, figure (1.4). It is capable of carrying accessories such as a small disruptor, laser, and sensors. Four oversized tires and invertible operation allow for mobility on any type of terrain or conditions. Low weight and high shock resistance allow the unit to be deployed even through a window so that rapid and safe inspection may be done with minimal human involvement.

Figure (1.4) Cobra M.R

III. INBOT

It is designed for audio and video inspection, surveillance and reconnaissance missions in confined zones such as pipelines, vehicle underbodies and buildings. Weighs only 5.6 lbs. (2.1kg) and can be carried by a single person. The robot's engineered protective packaging has been specifically designed for fast deployment in harsh and dangerous environment. The robot is literally “throw able” for rapid, safe deployment, Figure (1.5).

Figure (1.5) INBOT

1.5.2.2 Robots for special needs people

People who care for physically handicapped persons have great burdens in mind and body, the problem of care for them will become a social problem. To solve this problem:

I. A small mobile robot system is developed. The purpose of this system is to pick up and bring daily using objects putting them somewhere indoors semi-automatically. This mobile robot system consists of a manipulator, a mobile unit, a visual sensor, a control computer, an operator console and a wireless LAN modem. This system gives information about the surrounding area to the operator through the visual sensor and the operator console. The operator can control the mobile robot on an internet browser; it can be controlled from any type of computer and any place. The operator console using internet browser is very simple and easy to operate the robot, the communication method using computer network makes it possible to extend the service area of mobile robot.

II. Wheelchair mobile robot is found. It is a holonomic vehicle with a continuous transmission; this vehicle has four ball wheels. Independently actuated by DC motors, enables it for moving the vehicle in any direction within the plane and rotating it around its center [3]. The balls are fixed at the terminals of two beams, these two beams are intersecting at a joint which hold a chair upon it. This joint is provided with a differential gear mechanism. The angle between the two beams can be changed to continuously varying the gear ratio. This wheelchair is provided with a joystick to determine the direction of motion.

1.5.2.3 Teaching Robots

An example of such a robot is Lego-mindstorm robots. The mindstorm robot is a programmable 9 bit computer housed within a large Lego brick. It has 3 input ports, for touch and light sensors, and three output ports for driving motors. The robot kit comes with software that runs under windows operating system and enable the user to program the robot to perform an endless variety of tasks. The programs can be downloaded to the robot using an infrared transmitter that attaches to the computer serial port.

1.5.2.4 Scientific research

I. Tutebot

It is a very simple robot, that can follow a wall and when it bumps into something, a signal from the bump sensor directs two motors to reverse direction, and then it will backup and turn, what makes it turn is an element of state, or timing, in the system that is implemented with a resistor-capacitor (RC) circuit, one for each wheel. Tuebots brain is an analog computer, which is programmed only by adjusting potentiometers. Tutebots consists of switches, relays, motors, and discrete electronic components, all of which can be assembled rather easily, Figure (1.6) [2].

Figure (1.6) Tutebot Robot

II. C.R.I.S – Computerized Robot Intelligence System

CRIS is a fully autonomous mobile; it was created as a flexible, easily programmed research platform for the testing of various AI algorithms. At the time that it was entered in the 1997 Fort Worth Regional Science Fair, the robot's main function was to travel through an unknown space for a given amount of time and then fin its way back to its origin based solely on sensor readings of its environment. CRIS implements with three bump switches, and interface card. The robot is driven by two incredibly strong 6Vdc power motors which each have their own Yuasa 6V, 10AH batteries. CRIS also uses another 12V, 7AH lead acid battery for the computer. The robot is approximately 48" high and has a diameter of 18". The construction time of the robot was around 13 months after 14 months of research and planning, Figure (1.7).

Figure (1.7) CRIS

1.5.2.5 Micro - Inspection Robot

One type of these robots is a robot that is 23mm in diameter and 110mm in length. It can travel through both vertical pipes and curved sections. Its rate of travel is 6 mm/s. several micro devices and micro-mechanism are built in this micro-robot such as planetary wheel mechanism, micro camera, and micro hand for manipulating small objects in pipes.

The need to carry out inspections inside small pipe lines has grown recently with particular demand relating to the 1-in pipe lines often found in chemical plants, heat exchangers, and gas or water supply systems. The robot is controlled manually by an operator observing the camera images and handling a joystick and bottoms on control pendant.

1.5.2.6 Material Handling

The purpose of material handling is to transfer raw materials, tools, medicines, and supplies from one location to another to facilitate the overall operations of manufacturing or services. The handling of material must be performed safely, efficiently (at low cost), accurately, and without damage to materials.

1.5.2.7 Past projects

At 2002 in *Palestine polytechnic university* a wired mobile robot graduation project was introduced to the mechanical department of engineering. The robot can be operated in an automatic and manual operation. In automatic operation it can follow a predetermined path which consists of moving in forward for a programmed distance, then turning right or left through some angle (programmed angle), then moving again in forward direction for another programmed distance. After these motions the robot can reverse its orientation and stop for a period of time for the purpose of loading and unloading, and then it can return back following the same path to its starting position. This automatic loop can be repeated for a desired number if times.

In manual operation the user can determine where the robot will go through entering the desired distances and angles to allow the robot to move maximally ten movements, which are combinations of forward, backward, left and right. The user can determine a desired velocity that must be less or equal to .5(m/s), and through the pulse width modulation the speed of the robot could be controlled [4].

1.6 Time Planning

The following tables explain the expected timing plan for the first and second semester.

Table 1.1 Project Timing for the 1st semester Plan Table

Tasks \ Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Choosing project															
Literature review															
Mechatronics Design – Mechanical															
Mechatronics Design – Electrical															
Kinematic Modeling															
Control Design															
Documentation															

Table 1.2 Project Timing for the 2nd semester Plan Table

Tasks \ Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Building the mechanical structure															
Driving and steering mechanism															
Motors Implementation															
Micro-Controller															
Sensors Implementation															
Designing Control Circuit															
Documentation															

1.7 Estimated Cost

The following table shows estimated hardware costs.

1	Mechanical Structure	150
2	PIC Microcontrollers	100
3	Video Camera	340
4	WLAN Transceiver + Antenna	540
5	Motors	50
6	Boards & Integrated Circuits	160
7	Battery	70
8	Sensors	2350
Total (NIS)		**3760**

Table 1.3 Operating hardware costs

1.8 Contents

- **Chapter one:** Introduction to mobile robots, and the reason of choosing this project, literature review, methodology, project idea, cost of project, time tables.

- **Chapter two:** This chapter introduces the design process of the robot, where the mechanical, electrical and micro-controller parameters will be identified and designed.

- **Chapter three:** The kinematic analysis of the mobile robot and the wheels configuration will be discussed, motion restriction and degrees of freedom the robot has in a plane.

- **Chapter four:** This chapter will discuss the Control parameters, methods to control mobile robots, computer control process, type of controllers and the design of control circuit.

- **Chapter five:** This chapter introduces all the experiments and results which had been tested on the robot and its components.

- **Chapter six:** Recommendation and Future Work.

- **Appendix A:** Robots board connection diagram**.**

- **Appendix B:** Stepper and DC motors.

- **Appendix C:** DC motors and stepper motor drive circuits.

- **Appendix D:** PIC code.

- **Appendix E:** Encoders and ultra sonic sensors.

- **Appendix F:** Testing Codes.

- **References.**

Chapter Two

Design Concept

2.1 Introduction

This chapter discusses the design concept of the project as an input for the mechatronics design. This part includes the mechanical, electrical, telecommunication and computer components implemented in the project.

2.2 General Design Concept

The design of the robot must fit the requirements and the applications which had been discussed in chapter one, this means that there is some parameters that must be identified in order to choose the best configuration of mechanical, electrical, telecommunication, computer components.

2.2.1 System parameters

This robot will be designed for driving on flat terrain, which means that it will have a constant speed up to (0.7 m/s), and in order to achieve this speed, a suitable motors must be implemented, this motor must be small, light and an acceptable power consumption.

And since this project uses small motors, the total mass of the robot must suit the motors abilities (torque, power, and speed), which means that we must calculate the total inertia of the robots structure and add it to the motors inertia so we can get an equivalent inertia and then construct the transfer function of the system to make sure that this motor suits the robots operation and check the dynamic response of the system whether it's appropriate or not.

Then power consumption rate will be calculated in order to find the amount of power required to operate the robot. Also the number of analog I/O and digital I/O will be defined in order to use a specific PIC with the suitable number analog/digital I/O ports.

2.3 Mechanical Design

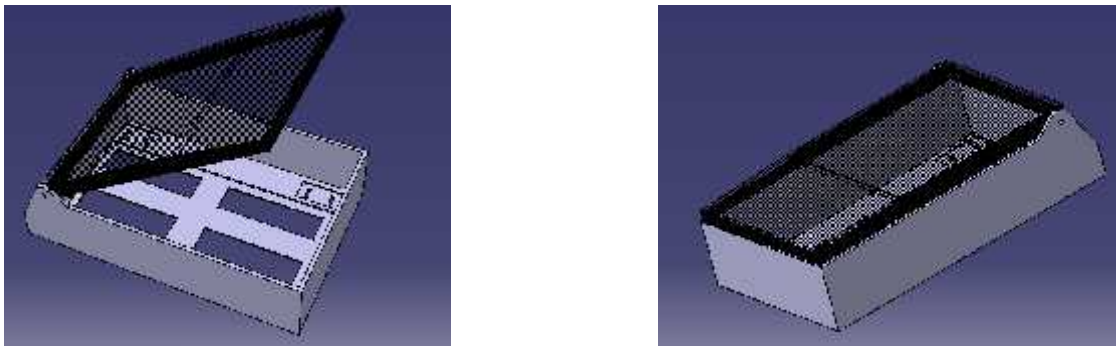
2.3.1 Mechanical structure

One of the simplest yet most effective ways to build a small robot is to use a body made of formed structure of plastic. Plastic is easy to machine and can produce a lightweight, rugged chassis.

In mechanical design there are two design strategies; the first one is assuming that all dimensions, locations of connectors, and kinds of material are known, then fixing some of them and see the best dimensions and locations after doing the analysis. The second strategy is seeing what is available of materials, dimensions, and expects the best locations of connectors and supporters, then check for a desired factor of safety [5]. The second strategy will be used.

This project is concerned with the design of mobile robot whose wheel arrangement will be a *differential configuration,* with two rolling wheels (ideal) for stability, according to the comparison we made in section (2.3.2). In this comparison we studied almost all wheels configurations in order to get the best mobility maneuverability and stability.

The basic structure of the robot was chosen after identifying all the components of the robot in order to fit all of them inside. We brought a plastic box, and then we machined it to fit the projects parameters. The structure of the robot is simple: a plastic box covered with a plastic surface, the weight of the box was 2.7 kg, and the dimensions were (approximately 40*30*13cm^3). The basic shape is shown in figure (2.1).



(a) Left view (b) Right view

Figure (2.1) Basic shape of the mobile robot

2.3.2 Wheel arrangement

While people and many animals walk on legs, most mobile machines roll on wheels. Wheels are simpler to control, pose fewer stability problems and can go faster than legs. Stability is maintained by ensuring that the center of gravity of the vehicle is always within the triangle formed by three points touching the ground [1].

The most familiar wheel layouts for vehicles are shown in figure (2.2):

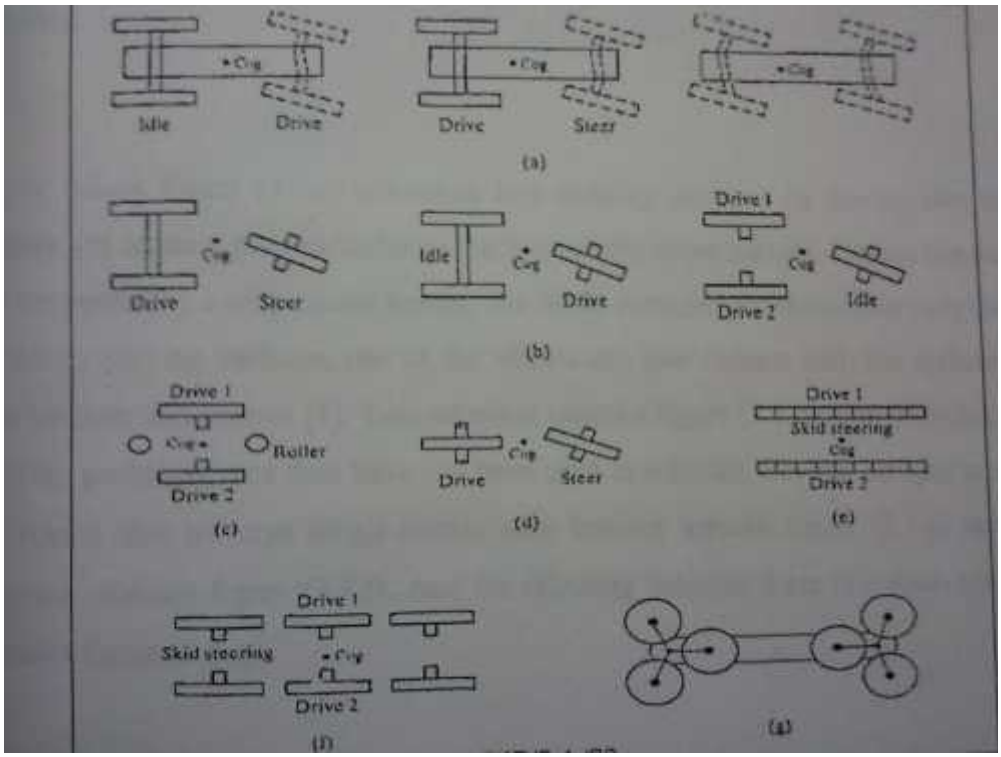


Figure (2.2) wheel arrangement [4]

In figure (2.2.a) four wheels are placed at the corners of a rectangle. Normally the two rear wheels are used for driving, and the forward two for steering. Alternative arrangements include front-wheel drive, four-wheel drive, and four-wheel steering allowing some limited sideways motions. Most four-wheel vehicles have limited maneuverability because they have to move a forward direction in order to turn [1]. Also, a wheel suspension system is required to ensure that the wheels are in contact with the ground at all times. When moving in a straight line all wheels rotate at the same velocity. When turning corners, the inside wheel rotate slower than the outside wheels, to avoid scrubbing because the contact distance traveled by the inside wheel is shorter. In a mobile robot, these requirements are met by a good mechanical design and by controlling of the drive of the wheels independently.

One way to reduce the problems of four-wheeled vehicles is to replace the coupled steering wheels with one wheel, figure (2.2.b). The problem still remains that for accurate control turning, the two drive wheels must rotate wheels must rotate at slightly different speeds. Three-wheeled vehicles have the advantages that wheel-to-ground contact can be maintained on all wheels without a suspension system. Here the single wheel is the drive wheel as well as the steering wheel. Combining drive and steering mechanisms in one wheel results in a more complex mechanical design. In other three-wheeled vehicles, two wheels are driven independently and the third is idle. Steering is accomplished by driving a wheel at different speeds. For the robot to follow straight lines and curves accurately, motor speed must be controlled precisely.

However, inaccuracies in achieving the desired trajectory can be caused by mechanical factors like wheel slip. In this design, the vehicle turns around a point along the axis between the driven wheels. If this axis passes through the center of the vehicle then the center of gravity has to be offset toward the idle wheel to maintain stability.

Two-wheeled vehicles figure (2.2.d), like bicycles, have stability problems, and thus have not been used in robotics. And for climbing vehicles there is a stair-climbing vehicle figure (2.2.g). Skid-steer locomotion figure (2.2.e) is commonly used on tracked vehicles such as tanks and bulldozers, but is also used on some four- and six-wheeled vehicles. Robots that make use of tread have much larger ground contact patches, and this can significantly improve their maneuverability in loose terrain compared to conventional wheeled designs. However, due to this large ground contact patch, changing the orientation of the robot usually requires a skidding turn, wherein a large portion of the track must slide against the terrain. The disadvantage of such configurations is coupled to the slip/skid steering. Because of the large amount of skidding during a turn, the exact centre of rotation of the robot is hard to predict and the exact change in position and orientation is also subject to variations depending on the ground friction. Controlling straight line travel can be difficult to achieve. Therefore, dead reckoning on such robots is highly inaccurate. This is the trade off that is made in return for extremely good maneuverability and traction over rough and loose terrain. Furthermore, a slip/skid approach on a high-friction surface can quickly overcome the torque capabilities of the motors being used. In terms of power efficiency, this approach is reasonably efficient on loose terrain but extremely inefficient otherwise.

This project uses the wheel arrangement showed in Figure (2.2.c). This Turtle robot configuration was chosen because it overcomes the stability problem by having two idling castors, on an axis perpendicular to the axis of the drive wheels and adds maneuverability and mobility to the movement of the robot. Unless the castors are supported by a suspension system, the turtle arrangement is suitable only for flat or slowly varying surfaces; one of the wheels can lose contact with the surface, and lose traction and control [4].

Figure (2.3) shows the wheel configuration and the implementation of the wheels in the structure of the robot.

Figure (2.3) wheel configuration

Where the diameter of the front and rear wheels (castors) are 4cm and they have bearings between them and the bottom of the robot in order to reduce the friction while turning and maneuvering as much as possible, so they will not affect the maneuverability and mobility of the robot, while the middle wheels (drive wheels) has 10cm diameter of rubber material.

Since the diameters of the castors and the drive wheels are not the same, we made some modification during the construction phase, where we used metal rings and screws to get a balanced contact with the ground for all of the wheels all the time.

2.4 Electrical Design

2.4.1 Energy supply

In mobile robot, energy supply is needed to perform the specific task we want autonomously. This energy is stored in accumulators or normal batteries. In fact there is a difference between these terms, the first ones are rechargeable, and the others are not. In this project we will use rechargeable batteries.

Choosing a battery configuration is a key feature in the design process. The dimensions, voltage, weight and recharge method of the battery configuration form restrictions on the entire design process and determine directly the autonomy of the robot.

2.4.2 Battery Pack

The Robot has many different components, and each component may need a different voltage value, meaning that the robot must have different voltage values onboard. So the robot carries a ‘sealed lead’ 12v-4Ah and a power rate of (48Wh), see figure (2.4).

Figure (2.4) 12 voltage-4Ah Battery

This main voltage source where used to supply two type of regulators onboard the robot, one of them is TS317 variable regulator that gives a variable voltages according to the connected resistors, this regulator where used to supply the robots transceiver and camera.

All of the connection circuits are in appendix A.

The other regulator was 7805 fixed regulator; it gives a fixed (5v) used to supply the PIC, H-bridge, ICs and ultrasonic sensors, all of these components will be discussed in the coming sections.

Table (2.1) shows the power consumption rate of all the components of the robot, to check whether the battery is capable for the project or not.

Component	**Power Consumption(W)**
PICs	2
Encoders	0.2
Motors	15
Ultrasonic	.08
Camera	0.198
Power Transistors	12
Logic ICs	0.2
Transceiver	0.3
Regulators	6
Total	**36.538**

Table 2.1 Component Power Consumption

Since the battery specifications are:

12V, 4Ah, 1.5kg and dimensions 100 *85*65mm^3.

➔The total power that the battery can supply:

$$p = V * I$$
$$p = 12v * 4Ah = 48Wh \quad (2.1)$$

➔If we considered the average power rating for the battery, since its voltage will drop to 10.5v when the current capacity is almost zero, the power rating may be considered as 40Wh.

➔So if power consumption of the robot including all its internal components equals 36.538W, we can calculate the operation time of the robot:

$$40(Wh) / 36.538(W) = 1h \quad (2.2)$$

Other specification of the battery back:

➔ Specific Energy $40Wh / 1.5(kg) = 27(Wh / kg)$. (2.3)

➔ Energy Density $40(Wh) * 0.100 * 0.85 * .065 = 22.1(Wh / m^3)$. (2.4)

2.4.3 Battery charging

In order to keep the robot at high performance capabilities, the battery must be recharged after using it for more than one hour, this value were estimated through experiment were the robot suffered lack of power when it was operated for this time long.

So to charge the battery all we need is a (13.5v-13.7v) power source and a diode to prevent the current from flowing back to the power supply after the battery is fully loaded. The charging time must be between 6-9hours.

2.5 Motors

Motors are the parts that are responsible for the movement operation of the robot; they convert the electrical energy they are supplied with into kinematic energy on their shafts, this energy is used to move the robot and its camera base.

Two types of motors were used in this project, DC motors and stepper motor; they are discussed in the next section.

2.5.1 DC Motor Specification

Since this project uses a *differential configuration*, then we had to use two motors for steering and driving, each one is connected to a middle wheel at each side, so if want to drive the robot in a straight line all we have to do is to equal the voltage on the two motors, and if we want to turn left or right, we can make one of the motors to turn in the opposite direction of the other, and if we want to drive the robot in an arc direction, we can vary the voltage of the motors where one of them will be faster than the other according to the direction we want.

And in order to be able to control and vary the speed of the motors, which will mostly be about (0.7 m/s), we chose the Compound-Wound RS445PA DC Motors; see figure (2.5); where it's a combination of the shunt wound and series wound types combining the characteristics of both. Characteristics may be varied by varying the combination of the two windings. These motors are generally used where severe starting conditions are met and constant speed is required at the same time. This motor needs one voltage source, and is fed from the drive circuit which will be discussed later in this chapter.

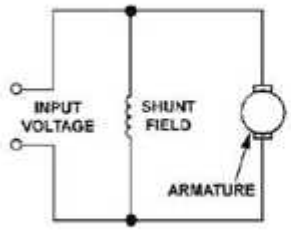


Figure (2.5) compound RS445PA DC Motors

This motor has an acceptable power rate (7.87w), small size, light weight (127g), high efficiency and wide operating voltage range, see appendix B.

2.5.2 DC Motor Driving Circuit

And to be able to control the direction of the each motor separately (cw or ccw) and rate of voltage transmitted to each motor, LM298 H-bridge were used on each motor, see figure (2.6), it consists of four switches connected in the topology of H. Where all switches are opened and closed so as to put a voltage of one polarity across the motor for current to flow through it in one direction, or a voltage of the opposite direction, causing current to flow through the motor in opposite direction for reverse rotation.

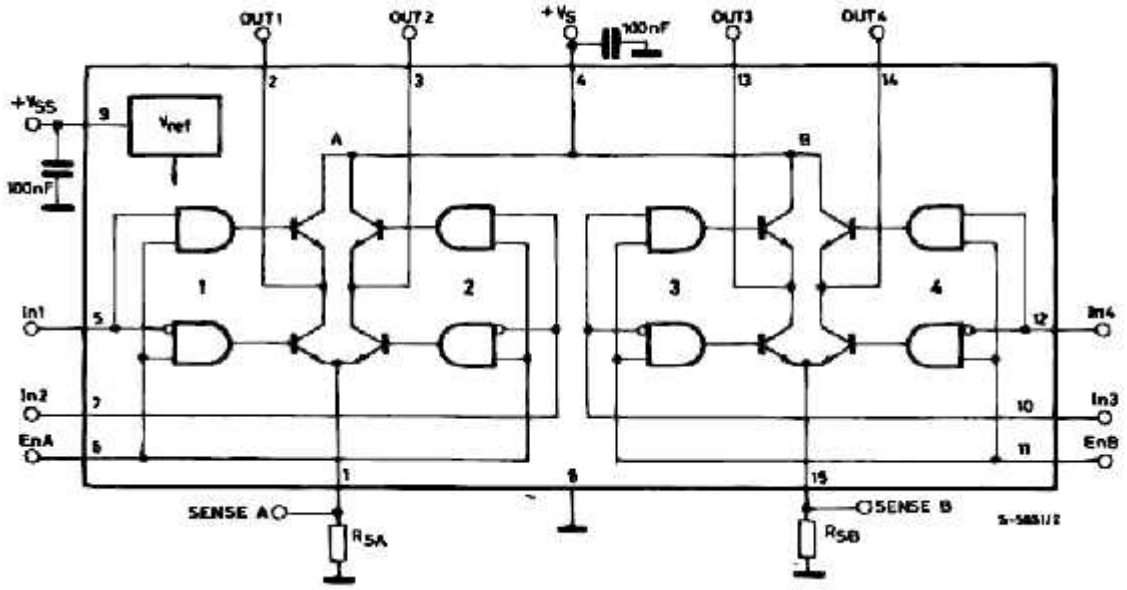


Figure (2.6) H-bridge using transistors

The previous figure shows the wiring connection of the LM298 dual H-bridge, where input (1, 2, 3 and 4) receives the Pulse Width Modulation (described in section 2.6.2) and direction signals from the PIC through an external Logic Circuit that is described in the next section, which passes these signals to the motors through output (1, 2, 3 and 4). The H-bridge is supplied with two voltages, one is 5v used to operate the H-bridge and the other is 24v used as supply voltage for the motors, also it has an enable signal that enables or disables it, these enable are connected to the sensing element described below.

The connection diagram is described briefly in appendix C.

In order to protect the motors and the H-bridge from high current flow, two pins in the H-bridge are used as sensing elements for current sensing, they are connected to a 1A fuse, where it allows only the current below 1A to flow through the circuit to since the motors can't hold more than (1.47A) at stall moment, so as long as the current is below 1A, these pins outputs will be

high logic voltage meaning that the H-bridge is enabled, and at the moment the current exceeds 1A, the H-bridge is disabled.

2.5.3 External logic Circuit

The robot as mentioned before uses two DC motors, and this requires four signals for each motor to control its speed and direction. And since we have one control PIC on the robot which has only two PWM pins in its configuration, we had to build an external logic circuit that can supply each motor with a direction and speed signal, this logic circuit consists of two IC's, the first one is 7408 AND gate and the second is 7404 INVERTER gate, where the AND gate receives the PWM and direction signals from the PIC and passes them to the H-bridge, so each motor has four signals for controlling direction and speed.

Much more details are mentioned in the Appendix C.

2.5.4 Gear Reduction

As mentioned before, the voltage of the two motors will be controlled by the PIC, but though, gears had to be implemented in order to reduce the speed of wheels and also to increase the motors torque on the wheel, to allow the wheels to overcome internal motors friction, or friction between the wheels and ground.

The ratio of the gear reduction was chosen in a way that reduces the speed of the wheel with respect to the motors in acceptable way, and also to satisfy the torque required for overcoming the friction.

The gear reduction ratio is as follow:

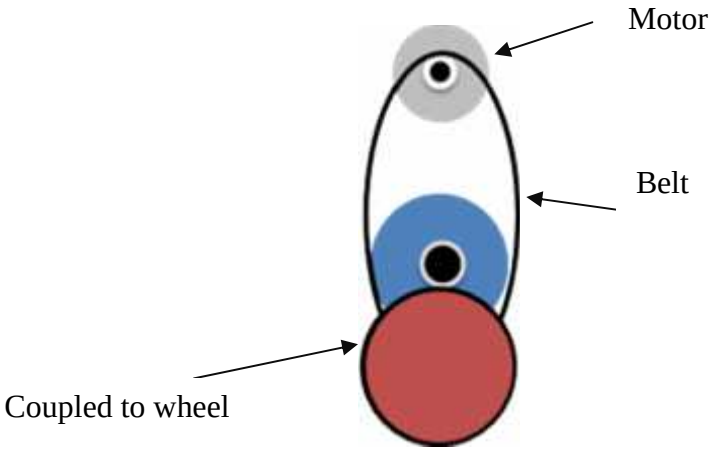


Figure (2.7) Gear reduction

Figure (2.7) shows the gear reduction that is coupled to the motors with 1:11 ratio, meaning that the shaft of the motor must rotate eleven revolutions in order to rotate the wheel one revolution.

2.5.5 Stepper Motor Specification

Stepper motor is used in this project due to its flexibility, high output torque, high performance with the PIC and wide range of step degree, this project has a stepper motor with (3.5°) degree, input voltage of (12-24v) that suits the power rate see appendix B.

Figure (2.8) Bipolar Stepper Motor

Where the stepper motor in figure (2.8) were used to move the camera base that carries the robots camera, and since the camera had to be rotated a full rotation for more than one time, the cable connection would have been a problem, so instead of connecting the cameras wires directly to the robots board, a **handmade** base where constructed to allow the camera to rotate 360° freely with no restriction, see figure (2.9).

This **handmade** base consists of two plastic gears, where the lower gear has four circular shape copper wires that is fixed into it, and these wires are connected to the robots board, where as for upper gear, it has four copper pins, that is fixed into it, and the cameras wires were connected to them.

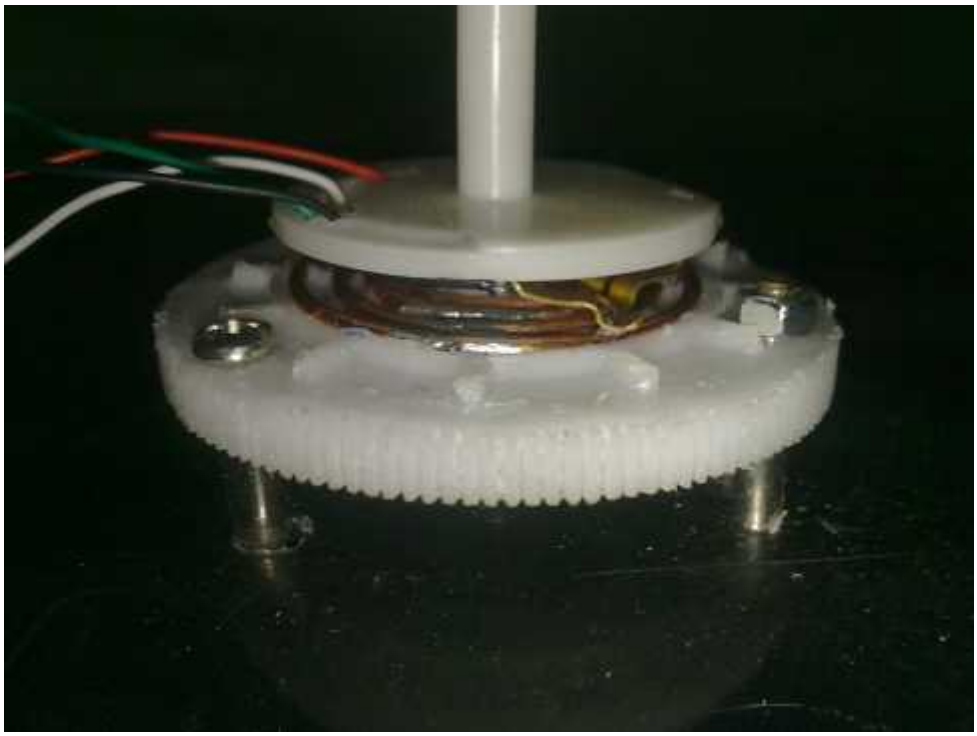


Figure (2.9) the handmade camera base

The copper on the two gears remains in contact all the time, whether the camera rotates (10°) degrees or (720°) degree.

The coupling with the stepper was with the lower gear, where the stepper was fixed on the internal side of the robots cover and connected through the robots board to the drive circuit of the stepper.

2.5.6 Stepper Motor Drive Circuit

Since the stepper motor requires a high power signal that the PIC cannot supply, an external drive circuit had to be interfaced between the motor and the PIC, this drive circuit consists of four 120TIP NPN Darlington Power Transistor, these transistor are used to amplify the signal coming from the PIC, and passes it to the stepper, the transistor works as a switch that connects the supply voltage connected to the collector voltage (12v); to the motor when its base receives a high logic signal.

Schematic of the connection diagram is in appendix C, also appendix D includes for the code for moving the stepper.

2.6 Microcontroller Design and Programming

A microcontroller (also microcontroller unit, MCU or µC) as defined in Wikipedia is "a small computer on a single integrated circuit consisting of a relatively simple CPU combined with support functions such as a crystal oscillator, timers, watchdog timer, serial and analog I/O etc. [8]

Microcontrollers have long been a convenient interface for the embedded systems; they represent the core of the control system for the electronic devices in dedicated applications. Thus, in contrast the microprocessors that are used in general purpose applications like personal computers that need high-performance. Microcontrollers contain data and program memory, serial and parallel I/O, timers, and external and internal interrupts [9], and many more peripherals, made them a strong choice when implementing control systems.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, remote controls, office machines, appliances, power tools, and toys. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems. [8].

Microcontrollers exist in many types and forms, for different applications and costs. And here is a list of some microcontrollers' types that are known in the market:

1) MIPS (32-bit PIC32)
2) ARM processors
3) 8051
4) PIC (8-bit PIC16, PIC18, 16-bit dsPIC33 / PIC24)

In this project the PIC microcontrollers is the type that will be discussed.

The PIC microcontrollers are a family from Harvard architecture microcontrollers that is manufactured by Microchip Technology. The name PIC at the beginnings stood for "Programmable Interface Controller" and shortly after that was replaced with the name "Programmable Intelligent Computers".

PICs are popular with both industrial developers and hobbyists alike due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability.[10]

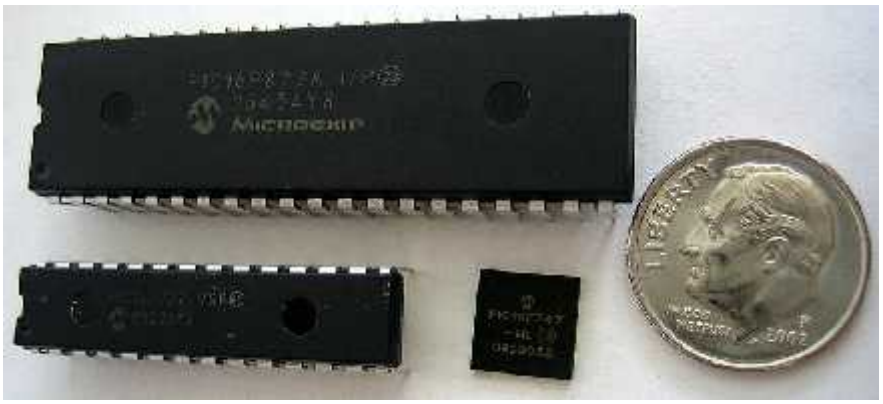


Figure (2.10) PIC Microcontrollers

The PIC microcontroller is part that requires much of programming, because it combines all the hardware of the robot system, the interfacing with them, communicating with them, and performs processing needed for the robot movement, the intelligent system to avoid obstacles, see figure (2.11).

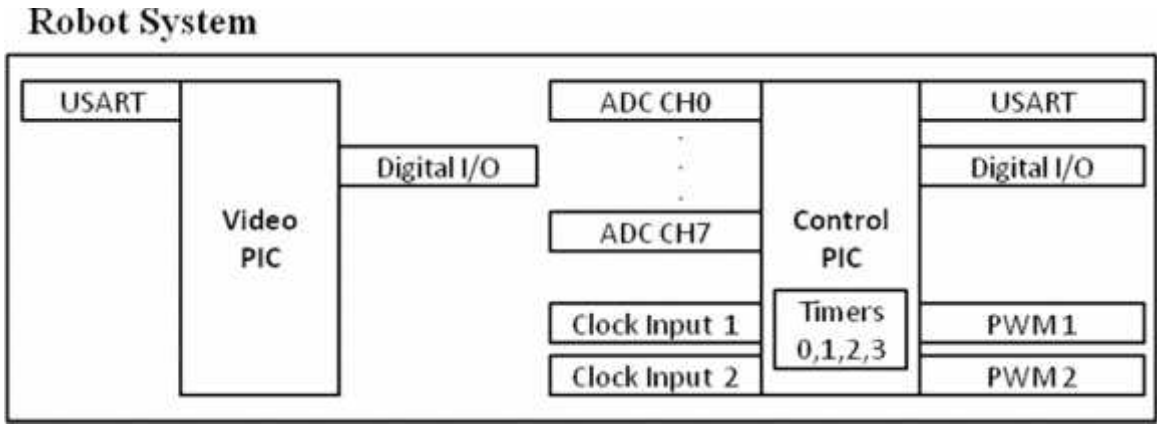


Figure (2.11) PIC Microcontroller Components that requires programming

2.6.1 USART Programming.

USART stands for Universal Synchronous Asynchronous Receive Transfer, which is a very important component in the system, because it is responsible for the serial communications, in which is used between the PICs, the WiFly transceiver and the camera.

2.6.2 PWM Programming

The Pulse Width Modulation (PWM) is used in this system to drive the two DC motors that is responsible for the robot movement. Two motors requires two PWM outputs which are provided by the PIC18F4550.

The frequency of PWM must be calculated carefully to match the DC motors and H-bridge frequencies, which is 20 KHz for both of them.

➔➔PWM period formula:

$$\text{Period} = ((1/\text{Frequency})/4 \text{ x } (1/F_{osc}) \text{ x Timer2 Prescaler}) -1 \quad (2.5)$$

Period = ((1/20 x 103)/4 x (1/8 x 106) x 1)-1

Period = 99;

After calculating the frequency of the output PWM signal, it is essential to determine the value of the Duty Cycle for each signal; this value determines the value of the voltage of the PWM signal.

Duty Cycle Formula:

$$\text{Duty Cycle} = (\text{Required Voltage}/ \text{Source Voltage}) \text{ x } 4 \text{ x PWM Period.} \quad (2.6)$$

The duty cycle value differs in this system by the function it is needed for, for example if the motors are required to move in full speed, this requires the full scale of the voltage signal from the PIC which is 5V, this can be accomplished by a value of 396 for the duty cycle.

Much more details are mentioned with the code in appendix D.

2.6.3 Timers Programming

Timers are very helpful modules in the PIC, they execute the required action in parallel with the PIC execution, this was needed because the controlling of the robot movement must work in parallel with sensors readings, and USART readings.

There are four timers in the PIC18F4550, Timer0, Timer1, Timer2, and Timer3. All of them were needed in the code to implement the system functions. In this section a brief description of each timer will be discussed, why it was needed, and how it was programmed.

- Timer0: This timer was configured to read the encoder (which is discussed in the coming sections) of the left wheel, the output voltage of the encoder will act as the input clock of the timer, making the timer working as a counter, reading the number of pulses from the encoder and calculating the relationship between the number of pulses and actual distance moved is described in section (2.7.3).

- Timer1: Same as Timer0, but it is connected with the right wheel encoder.

- Timer2: Used for the PWM operation, working as the counter of the period.

- Timer3: Used also for the PWM operation, working as the counter of the Duty cycle.

More details and codes are described in the Appendix D.

2.7 Sensors

The mobile robot in this project is controlled from distance, and probably most of the time the user will not have a sight on the robot even if the user can supervise the driving through the camera, yet he can't have a full view at one time for the surrounding area, and so sensors are used to assist the user in controlling the robots movement accurately, or to interrupt the driving operation before any damage can harm the robot, which is called a failsafe. Sensors in this project can be classified as:

2.7.1 External Sensors

External sensors measure the interaction of a robot with its environment. The range of external sensors are very large, an examples of them are touch, proximity, sound, gas-mixture, pressure, temperature and vision camera sensor. In this project LV-MaxSonar-EZ0 sensors are used, see figure (2.12).

Figure (2.12) LV-MaxSonar-EZ0

These ultrasonic sensors can range objects up to 6.45m (even when ranging moving objects), it has small size, low power and high sensitivity. The ranging process is described in the next section.

Eight LV-MaxSonar-EZ0 sensors are implemented around the robot in a way that helps the robot to recognize the surroundings easily, this configuration was chosen also to fit the intelligence procedure the robot will follow when it find an obstacle. And since the sensors have a large beam width, when they recognize the ground as an obstacle when they were implemented on the robot, so we had to reduce the range of 6 sensors (right, left and back side) to (20cm) where as the front sensors were implemented at (15°) up to be able to set their range to (1.5m) to give the robot the ability to check where it's going before it start the operation, see figure (2.13), also see appendix E.

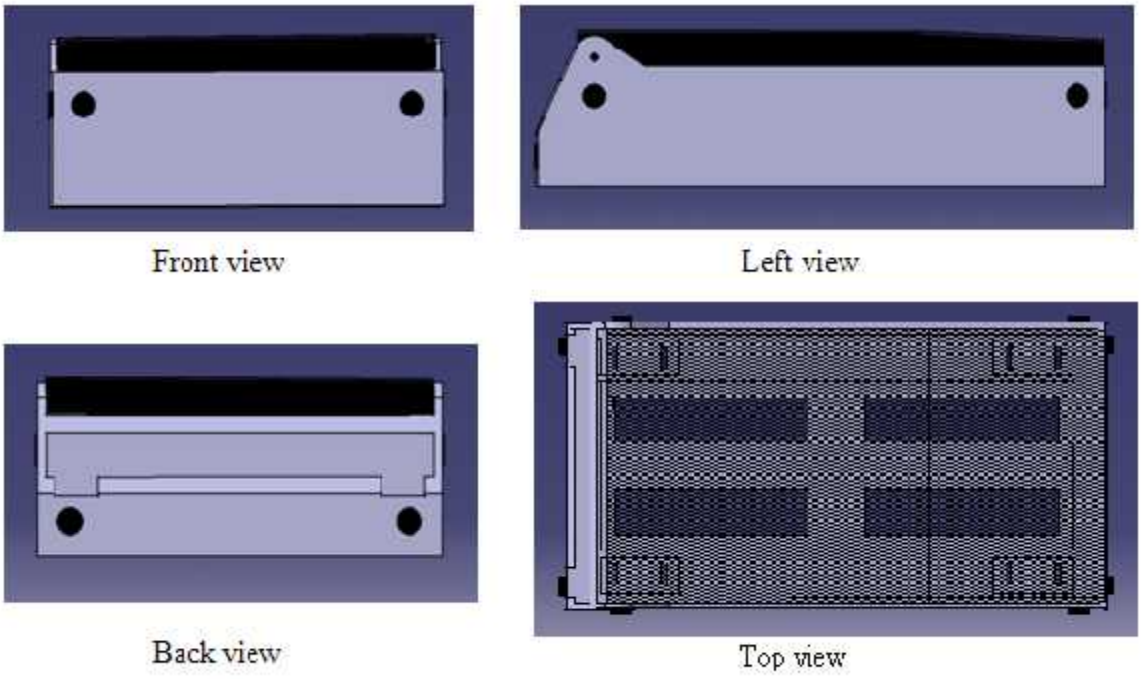


Figure (2.13) Ultrasonic Implementation

2.7.2 ADC Programming

The programming of the Analog-to-Digital converter module is very simple but essential in this system, the ultrasonic sensors (discussed in the next section) are connected to this module, they output a voltage signal, and its value determines the range of the closest obstacle.

To program the ADC to read the range, it is important to determine the Least Significant value of the ADC, which is the value of voltage that is detected by the module referred to as the resolution.

$$LSB = \text{Input Voltage}/2^{\text{ADC-reselution}}. \quad \text{Where ADC-resolution} = 10 \text{ bit} \qquad (2.7)$$

$$LSB = 5V/1024 = 4.8828125mV$$

Then to calculate the value of the ADC that represents the range in inches the sensors sensitivity is important. So, the distance of the obstacle formula is:

$$\text{Value of ADC (integer)} = \text{Distance (inches)} \times \text{Sensitivity} / LSB \qquad (2.8)$$

Sensors sensitivity: A supply of 5V yields ~9.8mV/in.

For example if the robot is needed to detect obstacles close than 30cm. the value of the ADC will be:

$$ADC = ((30cm/2.5)(inches) \times 9.8)/ 4.8828125 = 24. \quad (2.9)$$

More techniques to deal with eight analog inputs one for each sensor will be fully discussed in the Appendix D.

2.7.3 Internal Sensors

As we said, this mobile robot is controlled from distance, and so the controller must ensure perfect tracking for the motors in order to control their rotation and position, and to be able to do that, sensors must be implemented in the feedback branch of the controller, some of the sensors that could be used for this job, is the optical encoder see figure (2.14), it consists of transistor (LED) and receiver (phototransistor) and a disc with many small holes or slots near the outside circumference. The disc is attached to a motor shaft. The light from LED flashes on and off each time the slots pass by as the disc rotates, producing digital signal which can be counted and then converted into distance traveled.

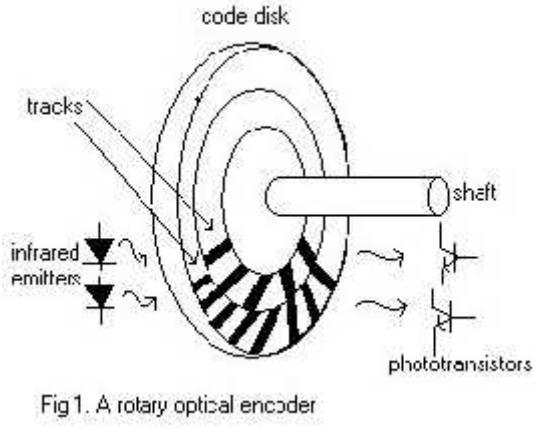


Figure (2.14) Optical Encoder

The CE incremental rotary encoder is used in this project, where it has (60pulse/rev) for resolution, and has two phases in order to know the movement direction of the robot, see figure (2.15).

Figure (2.15) CE rotary encoder

And since the direction of movement is extremely important for the robot, an external logic circuit was implemented in the robot to identify the movement direction (forward or backward). It's important because the robot will sum the distance reached to the value saved in its memory, or subtract the value of the present data from the past data in order to calculate the remaining distance to reach the new coordinates.

This logic circuit consists of a 7474 Flip-Flop, where phase A of the encoders were connected to the Q pin, and phase B was connected to the Clock pin in the flip flop.

Much more details about the connection diagram and code are mentioned in the appendix C and D.

The encoders are coupled to the motors back shafts directly, which means they are before the gear reduction (ratio 11 motor: 1 wheel), so if the wheel makes a one revolution, the encoder must rotate 11 revolutions, meaning that the relation of the pulses and the wheel rotation will be as follows:

$$\rightarrow\rightarrow \frac{D}{N} = \frac{2*®*R}{11*60} \tag{2.10}$$

$$N = \frac{D*11*60}{2*®*R} \tag{2.11}$$

$$N = \frac{D*11*60}{2*®*5} \tag{2.12}$$

$$N = \frac{D66}{®} \tag{2.13}$$

Where N: number of pulses.

D: distance reached (cm).

R: radius of wheel.

For example: if the robot is ordered to move 1m the timer0 reading must reach 955.

So after we found the relation between the number of revolution of the wheel and the number of the revolution of the encoder, we can easily give the robot an order to move a specific distance, which will be as we said (x, y) coordinates, where these values will be transmitted through the web server then it will be converted in the PIC into distance and an angle according to the below equations, and then sends them to the microcontroller on the robot through the transceiver.

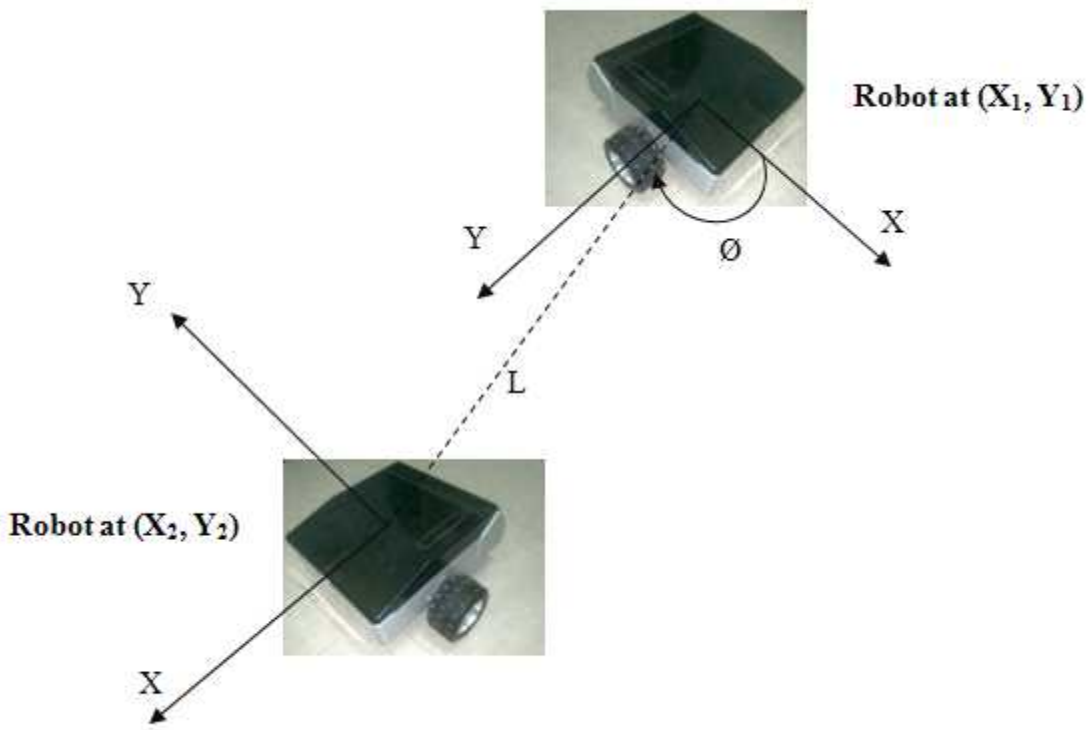


Figure (2.16) calculating new coordinates

The PIC will receive the new coordinates through the transceiver and it will start computing the angle to rotate and the distance to move with respect to the reference axis at (X_1, Y_1) through the following equation that will be programmed inside it.

$$L = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2} \tag{2.14}$$

$$¤ = \tan^{-1} \frac{(y_2 - y_1)}{(x_2 - x_1)} \tag{2.15}$$

And after the PIC calculates these values, they will be defined in a register inside the PIC and they will be compared with the encoders reading to indicate whether the robot has reached its destination or not.

Much more explanation about encoder's, moving functions and codes are in the appendix D.

Chapter Three

Kinematic Modeling

3.1 Introduction

This chapter discusses the kinematic modeling of the robot and wheels. *Kinematic* is the branch of classical mechanics that describes the motion of objects without consideration of the causes leading to the motion [15].

In order to find the kinematic model of the robot and the wheels, we need to specify the coordinate frames of them. Coordinate frames are assigned at both ends of each link. The links of a wheeled mobile robot are the floor, the robots body and the steering link. The relationship between the body of the robot and the floor is modeled by two coordinate frames, one is attached to the center of the robot and the other is attached to the floor. The degrees of freedom of the robot are three (x, y, z), while a conventional wheel has only two degree of freedom, because the third degree of freedom in model is eliminated when setting the z-component to zero since the wheel will move on a flat surface.

3.2 Modeling assumptions

a) The robot moves in a planer surface.

b) The guidance axis are perpendicular to floor.

c) Wheels rotate without any slippery problems.

d) The robot does not have flexible part.

e) During small amounts of time, which direction is maintained constant, the vehicle will move from one point to other following a circumference arc.

f) The robot is considered as a solid rigid body, and any movable parts are the direction wheels, which are moved following a command control position.

3.3 Kinematic Restrictions

Consider the wheel of the mobile robot as shown in figure (3.1), wheel movement (speed) in direction x is calculated by radius r and angle speed rotation ¯ by:

$$\dot{x} = r\bar{\dot{}} \tag{3.1}$$

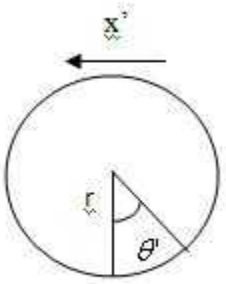


Figure (3.1) speed of wheels

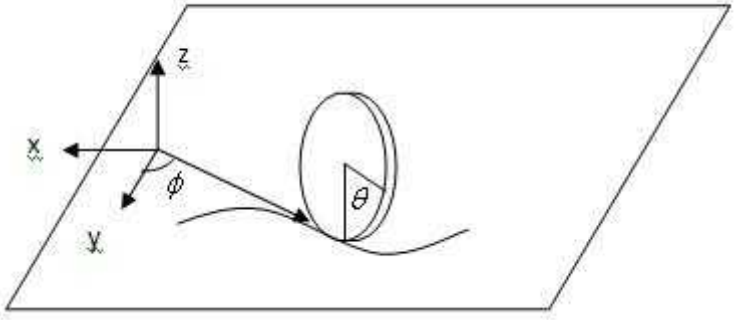


Figure (3.2) Kinematic restriction of wheels in 2D plane

$\dot{x}$ is the speed in the x direction, it is directly proportional by the angular velocity of the wheel. However, other restrictions appear in wheels when the movement is restricted to a 2D plane (x, y). Assume the angular orientation of a wheel is defined by angle¤ . Then, while the wheel is following a path and having no slippery conditions, the velocity of the wheel at a given time, which is set by r¯ ', has the following restrictive velocity components $(\dot{x}, \dot{y})$with respect to coordinate axes X and Y.

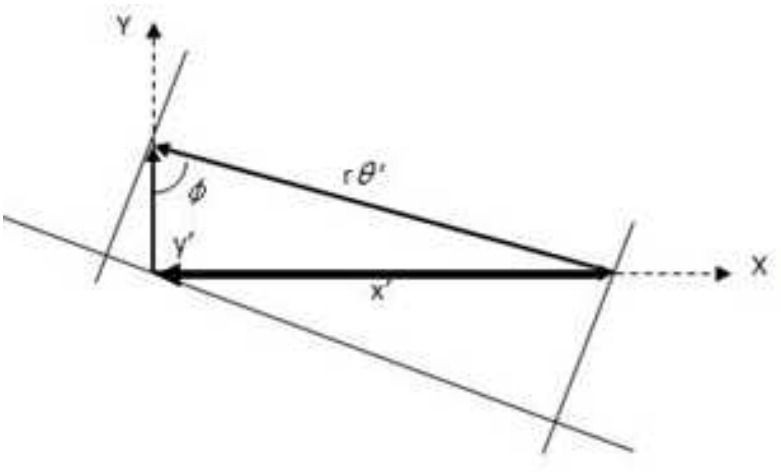


Figure (3.3) Remark restrictions for 2D plane movement

$$\vec{r} = -\dot{x}\sin¤ + \dot{y}\cos¤ \tag{3.2}$$

Derive equation (3.2), for a constant speed the acceleration will be zero, there for:

$$0 = \dot{x}\cos¤ + \dot{y}\sin¤ \tag{3.3}$$

Consider now that the mobile robot follows a circular trajectory as shown in figure (3.4), notice that the lineal and angular velocities of the robot are given by

$$v = \Delta s / \Delta t \tag{3.4}$$

$$´ = \Delta¤ / \Delta t \tag{3.5}$$

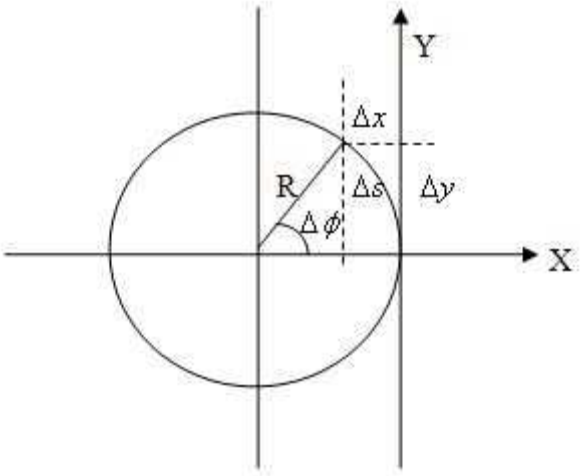


Figure (3.4) Circumference movement of the vehicle

Where Δs and $\Delta ¤$ are the arc distance traveled by the wheel, and its respective orientation with respect to the global coordinates. The arc distance Δs traveled in Δt time is obtained by:

$$\Delta s = R\Delta ¤ \tag{3.6}$$

Where R is the circumference radius of the wheel.

The curvature is defined as the inverse of the radius R as:

$$¥ = 1/R = \Delta ¤ / \Delta s \tag{3.7}$$

The movement equations in the initial position are given by the following expressions:

$$(\Delta x) = R(\cos(\Delta ¤) - 1) \tag{3.8}$$

$$(\Delta y) = R\sin(\Delta ¤) \tag{3.9}$$

An extension of the later equation is provided in the next expression, considering a specific initial orientation of angle ¤ . This is accomplished by rotating the earlier initial coordinates (3.8) and (3.9).

$$\Delta x = R[\cos(\Delta ¤) - 1]\cos ¤ - R\sin(\Delta ¤)\sin ¤ \tag{3.10}$$

$$\Delta y = R[\cos(\Delta¤) - 1]\sin¤ + R\sin(\Delta¤)\cos¤ \tag{3.11}$$

Assuming now that the control interval is sufficiently small $\Delta t = 0$, then we can assume the orientation change would be small enough, and

$$\cos(\Delta¤) \cong 1 \tag{3.12}$$

$$\sin(\Delta¤) \cong \Delta¤ \tag{3.13}$$

Substituting (3.12) and (3.13) into (3.10) and (3.11), we get

$$\Delta x = -R\Delta¤\sin¤ \tag{3.14}$$

$$\Delta y = R\Delta¤\cos¤ \tag{3.15}$$

Now, considering (3.6), we get

$$\Delta x = -\Delta s\sin¤ \tag{3.16}$$

$$\Delta y = \Delta s\cos¤ \tag{3.17}$$

Dividing both sides of equations (3.16) and (3.17) by Δt, and considering also (3.4), if Δt tends to zero, we finally get

$$\dot{x} = -v\sin¤ \tag{3.16}$$

$$\dot{y} = v\cos¤ \tag{3.17}$$

Also, using (3.5) we can obtain the complimentary equation

$$\dot{¤} = ´ \tag{3.18}$$

3.4 Jacobian Model

Assume that p represents a point in the space having n generalized coordinates, and q a vector of m actuation variables (for n>m), and assume p' and q' are the respective derivatives of such vectors, then the direct model is obtained by the jacobian matrix, J(p) by

$$\dot{p} = J(p)\dot{q} \tag{3.19}$$

The jacobian expression can be written in the form (Zhoa and Bennet):

$$\dot{p} = f(p) + \sum_{i=1}^{m} g(p)_i \dot{q}_i \tag{3.20}$$

$$\dot{p} = \begin{bmatrix} -\sin ¤ \\ \cos ¤ \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} ´ \tag{3.21}$$

Where v is the linear velocity of the vehicle, and ´ is its angular velocity. These equations can also be changed to the form of equation (3.19) as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{¤} \end{bmatrix} = \begin{bmatrix} -\sin ¤ & 0 \\ \cos ¤ & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ ´ \end{bmatrix} \tag{3.22}$$

For $\dot{q} = [v \quad ´]^T$

Notice that combining the first two equations from (3.22), and eliminating v, we get back the restricted relationship (3.3) by

$$\dot{x}\cos ¯ + \dot{y}\sin ¤ = 0 \tag{3.23}$$

This is due that the vehicle can only move along its longitudinal axis by

$$\tan ¤ = -\dot{x}/\dot{y} \tag{3.24}$$

In other words, the vehicle position (x,y) and its orientation ¤ are not independent.

The inverse model of the system involves the inverse of the jacobian. When the jacobian is not a square matrix, it is necessary to calculate its pseudo inverse, by multiplying both sides by J^T, and solving for q', to obtain:

$$\dot{q} = \{J(p)^T J(p)\}^{-1} J(p)^T \dot{p} \tag{3.25}$$

Then, for model (3.22) and using (3.19), we obtain

$$\begin{bmatrix} v \\ \acute{} \end{bmatrix} = \begin{bmatrix} -\sin ¤ & \cos ¤ & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{¤} \end{bmatrix} \tag{3.26}$$

From the first relationship, we obtain the earlier restricted condition (3.3) by

$$v = -\dot{x}\sin ¤ + \dot{y}\cos ¤ \tag{3.27}$$

3.5 Configuration of mobile robot

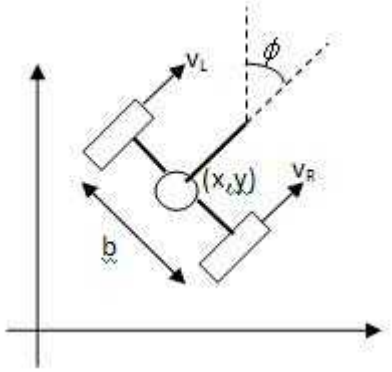


Figure (3.5) Differential configuration of mobile robot

Figure (3.5) shows the differential configuration of mobile robot, it use independent velocities in both wheels left and right ($v_{L,}$ and v_R respectively) to move in the 2D plane to a specific point (x,y) and specific orientation ¤ .

Assume for differential model, that ´ $_L$ and ´ $_R$ are the corresponding angular velocities of the left and right wheels. Given the radius of the wheels as r, the corresponding linear and angular velocities of the robot are given by

$$v = (v_R + v_L)/2 = r(´_R + ´_L)/2 \tag{3.28}$$

$$´ = (v_R - v_L)/b = r(´_R - ´_L)/b \tag{3.29}$$

Where b is the bias of the robot (separation of the two central wheels). Also, if the linear and angular velocities are provided, then the angular velocities of the wheel can be obtained by

$$´_L = (v - (b/2)´)/r \tag{3.30}$$

$$´_R = (v + (b/2)´)/r \tag{3.31}$$

Substituting equations (3.28) and (3.29) into model of mobile robots (3.21), we find

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{¤} \end{bmatrix} = \begin{bmatrix} -(r\sin¤)/2 \\ (r\cos¤)/2 \\ -r/b \end{bmatrix} ´_L + \begin{bmatrix} -(r\sin¤)/2 \\ (r\cos¤)/2 \\ r/b \end{bmatrix} ´_R \tag{3.32}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{¤} \end{bmatrix} = \begin{bmatrix} -(r\sin¤)/2 & -(r\sin¤)/2 \\ (r\cos¤)/2 & (r\cos¤)/2 \\ -r/b & r/b \end{bmatrix} \begin{bmatrix} ´_L \\ ´_R \end{bmatrix} \tag{3.33}$$

3.6 Force Interaction with Ground

Since the total weight of the robot is balanced consider the following free body diagram of the robot, figure (3.6), where the center of gravity (G) of the robot was computed using CA-TIA, and its located on the driving axis.

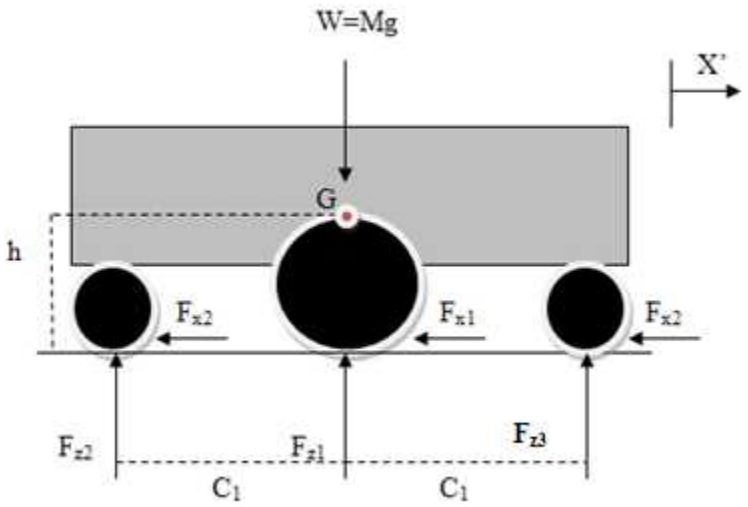


Figure (3.6) free body diagram (a)

Where

M: mass of the robot=3.95kg	G: center of gravity
$F_{x1,}$ F_{x2}: friction force with ground	F_{z1}, F_{z2}, F_{z3}: Normal Force from ground
C_1=12.5cm	g= 9.81m/s^2
h=10cm	a= 0.7m/s^2

The previous figure shows the free body diagram of the robot, we can see the force interaction between the robot and the ground in vertical and horizontal direction, where we have two situations, the first one when moving forward where the rear wheel loses contact with ground (F_{z2}=0), the second situation when moving backward where the front wheel loses contact with ground (F_{z3}=0).

The following equation expresses the interaction with ground, through calculating the moments around the center of gravity of the robot according to vehicle dynamic equations [13], these equations will be as follow:

For the Vertical forces:

$$\rightarrow F_{z1} = 2Mg\left[\frac{C_1}{C_1 + C_1}\right] - Ma\frac{2h}{C_1} \quad (3.34)$$

$$F_{z1} = Mg - \frac{2h}{C_1}Ma$$

$$\rightarrow F_{z2} = 2Mg\left[\frac{C_1}{2C_1}\right] - Ma\frac{h}{2C_1} \quad (3.35)$$

$$F_{z2} = Mg - Ma\frac{h}{2C_1}$$

$$\rightarrow F_{z3} = 2Mg\left[\frac{C_1}{2C_1}\right] - Ma\frac{h}{2C_1} \quad (3.36)$$

$$F_{z3} = Mg - Ma\frac{h}{2C_1}$$

After substituting the values of the variables in (3.34), (3.35) and (3.36), we get:

$F_{z1} = 34.276N$

$F_{z2} = 37.594N$ While F_{z3} will be zero when moving backward

$F_{z3} = 37.594N$ While F_{z2} will be zero when moving forward

➔➔Since Mg= 3.95*9.81 (3.37)

=38.7495N

This value is sufficient enough compared to the normal forces that are exerted on the robot, meaning that the robot will not flip over when it starts moving in either direction.

3.7 Equivalent Inertia

The equivalent inertia of the robot is a very important and critical branch in the robot system, since its value affect the performance of the robot during movement action. Also it's important because it's substituted in the system over all transfer function.

Transfer function is a relation between the input and the output of the system, through it we can consider the system stable or unstable and all other properties of the system are involved in the transfer function formula.

As we said in order to find the transfer function of the system, the equivalent inertia must be calculated. In this project the equivalent inertia consists of a summation of the motors inertia and the structures inertia, we can use the kinematic energy to find its value as in the next section.

3.7.1 Kinetic Energy

The kinetic energy is defined as the work needed to accelerate a body of a given mass from rest to its current velocity.

For our system the general kinematic equation is:

$$KE = \frac{1}{2} J_{eq} \dot{\theta}_m = \frac{1}{2} J_m \dot{\theta}_m^{\,2} + \frac{1}{2} J_1 \dot{\theta}_m^{\,2} + \frac{1}{2} J_{st} \dot{\theta}_w^{\,2} + \frac{1}{2} J_2 \dot{\theta}_w^{\,2} + \frac{1}{2} J_w \dot{\theta}_w^{\,2} \qquad (3.38)$$

Where:

J_{eq}: Equivalent Inertia J_m: Motors Inertia

J_1: Motors gear inertia J_{st}: Structures Inertia

J_2: Wheels gear Inertia J_w: Wheels Inertia

And since the gear reduction ratio is

$$\frac{\dot{\alpha}_w}{\dot{\alpha}_m} = \frac{N_1}{N_2} \quad \rightarrow\rightarrow \quad \dot{\alpha}_w = \frac{11}{1}\dot{\alpha}_m \tag{3.39}$$

We get

$$J_{eq} = J_m + J_1 + 11J_{st} + 11J_2 + 11J_w \tag{3.40}$$

This value contains all the inertias of the robots components. And since the gear box consists of two small gears of plastic as mentioned before, their inertias can be neglected. So that the formula will be:

$$J_{eq} = J_m + 11J_{st} + 11J_w \tag{2.41}$$

We can divide equation (3.41) into two main components, the first one is the inertia at the motors branch and the other one is the structures inertia.

We will find them separately in the next two sections, and then they will be combined together.

3.7.1.1 Motors Inertia

Let us start with motors branch inertia, see figure (3.7) which shows the components at the motors branch, and consider $J = J_m + 11J_w$.

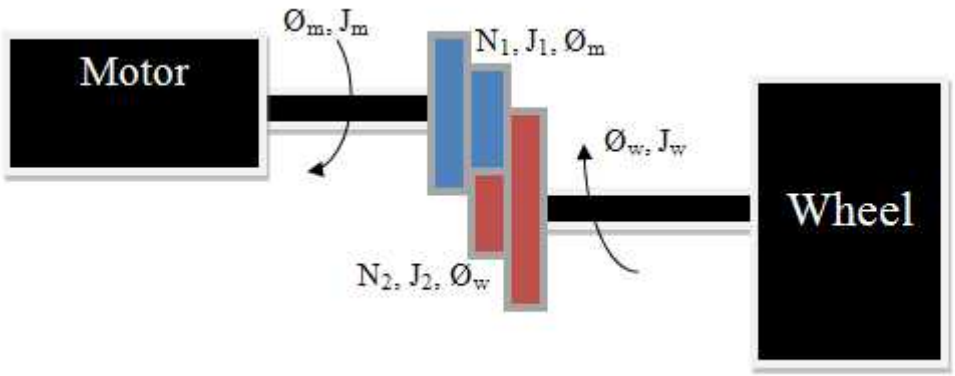


Figure (3.7) Motor branch inertia

The value of (J) were obtained through an experiment using the XPC Target, since the motors are coupled to gears and wheels; we were able to measure the motors response through the encoders that are coupled to the motors; using the DAQ which stands for Data Acquisition Card.

Note:

We mention that the XPC Target experiment was made before changing the gear box. We changed the gear box due to a fatal damage we encounter while testing the robot on ground, this fatal damage forced us to modify the gear box ratio and change it from (1:6) to (1:11).

So we recommend replaying this experiment with the new configuration to get an accurate response. Though we assumed no difference between the two configurations due to shortage of time we had.

3.7.1.1.1 XPC Target Experiment

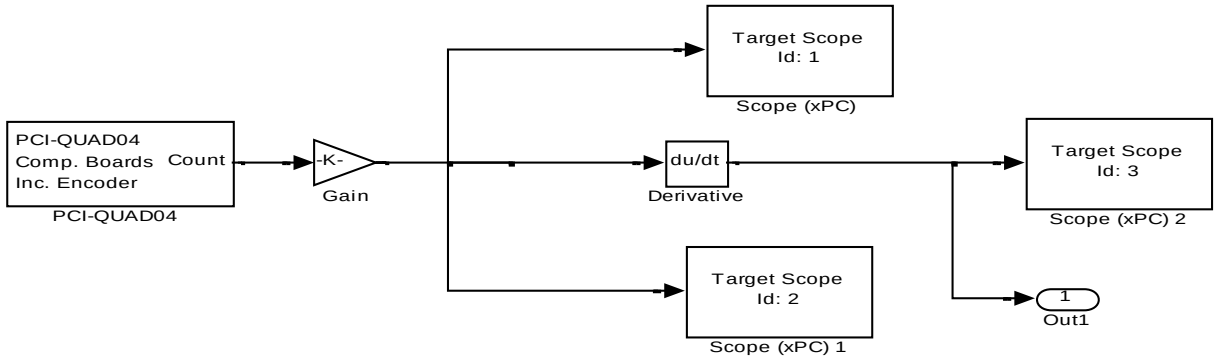


Figure (3.8) XPC target model

XPC target is a technology that can be used to analyze data and systems through running the experiment on two computers, the first one captures the data from the tested system, and the other computer analyze it.

The robot were flipped over on it back, and the encoders were connected to the first computer through the DAQ, then we ran the robots motor for ten seconds and captured the encoders signal, then sent them to the XPC target that analyzed the data that has been quantized, figure (3.9) shows the first order response captured from this experiments.

The following figure represents the data obtained through the XPC target experiment.

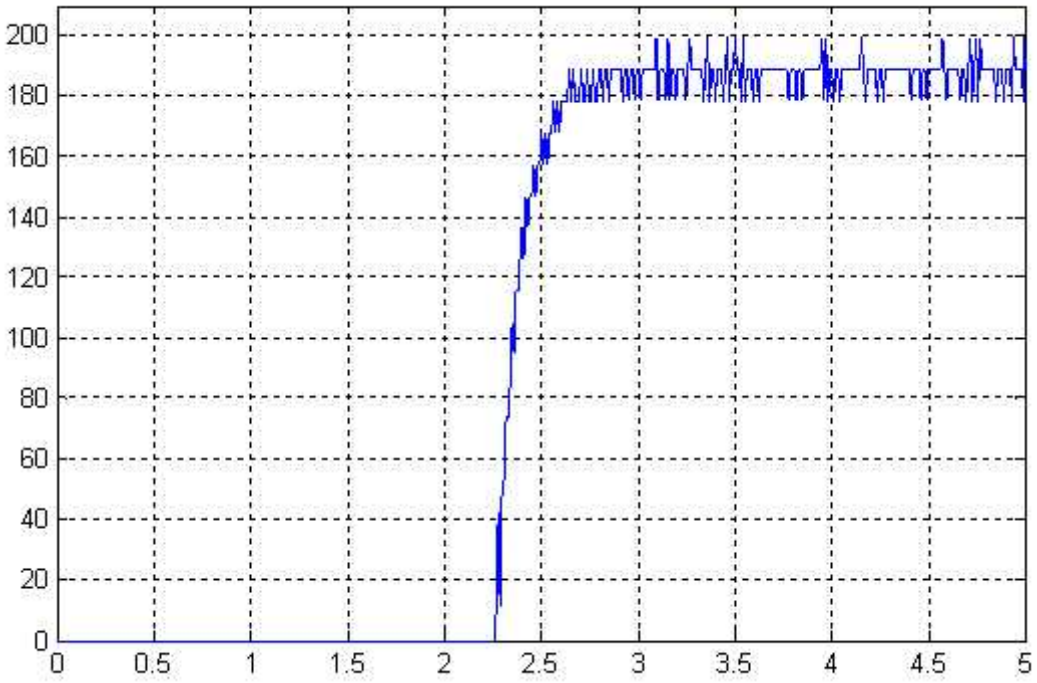


Figure (3.9) Speed response of the motors

From this figure (3.9), we know that the system is a first order system, we can see that there is some noise signal combined with the encoders signal, and this is due to the 50Hz electricity supplied to the laboratory the experiment were made at.

3.7.1.1.2 Deriving J the first component of J_{eq}

Since the previous figure shows the response of the motors, wheels and gears, we can use the following equation to find the equivalent inertia of these components.

General first order motor transfer function is derived in chapter four, section three, equation (4.19):

$$C(s) = \frac{s^{-}(s)}{V_a(s)} = \frac{K}{s+a} = \frac{K_m/(R_a b + K_b K_m)}{^{2}{}_{m}s+1}$$

Where $^{-}(s)$: position of the motor

$V_a(s)$: armature voltage

K_m, K_b: motors constant

$^{2}{}_{m}$: Time constant

R_a: armature resistance of the motor

b: friction constant (neglected)

K: gain

a: 1/time constant

And from the data acquired, the actual transfer function can be found, according to the following points.

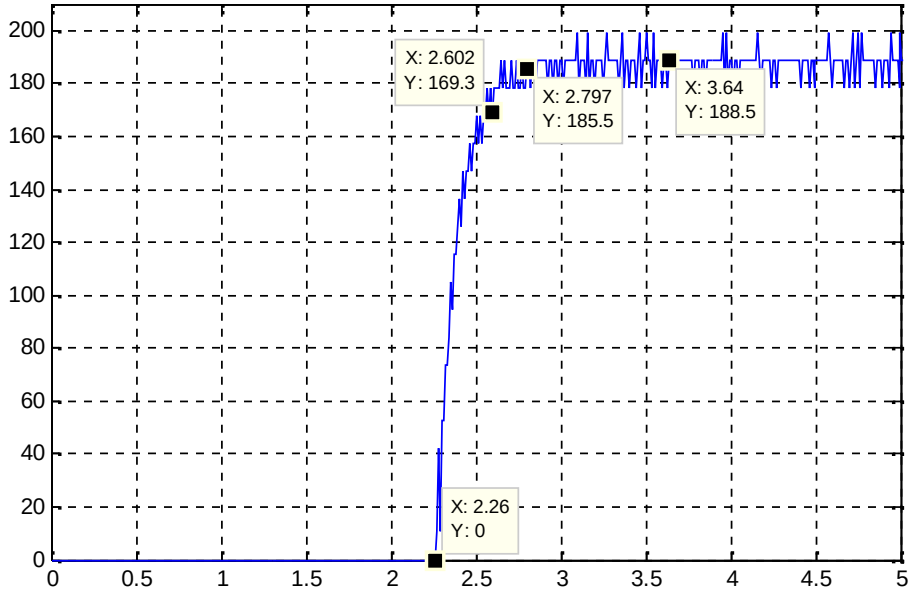


Figure (3.10) speed response

From figure (3.10), we can estimate:

1) The amplitude is 188.5
2) Time constant= 63%*Amplitude

=118.755 at t=2.383-2.25=0.133 (3.42)

3) t=1/a ➔➔ a=7.52 (3.43)
4) K=amplitude*a K=1417.3 (3.44)

Now the transfer function formula can be acquired:

$$C(s)=\frac{\bar{s}(s)}{V_a(s)}=\frac{K}{s+a}=\frac{1417.3}{s+7.52}*\frac{1}{6} \quad (\textbf{\textit{Due to old gear box}}) \tag{3.45}$$

$$=\frac{236.2}{s+7.52}$$

$$=\frac{31.4}{0.133s+1}=\frac{K_m/(R_a b+K_b K_m)}{^2{}_m s+1}$$

And to find the value of (J_{eq}), we need to find the motors constants, using the following equations:

$$\rightarrow K_b=\frac{e_a}{\mu_{noload}} \tag{3.46}$$

$$\rightarrow K_m=\frac{T_{stall}R_a}{e_a} \tag{3.47}$$

Where e_a: armature voltage μ_{noload} : speed at no load

T_{stall}: stall torque

The values of these variables can be acquired from the motors data sheet in appendix, where the torque, speed, efficiency and rated voltage are specified in the specification figure of the motor, see figure (3.11).

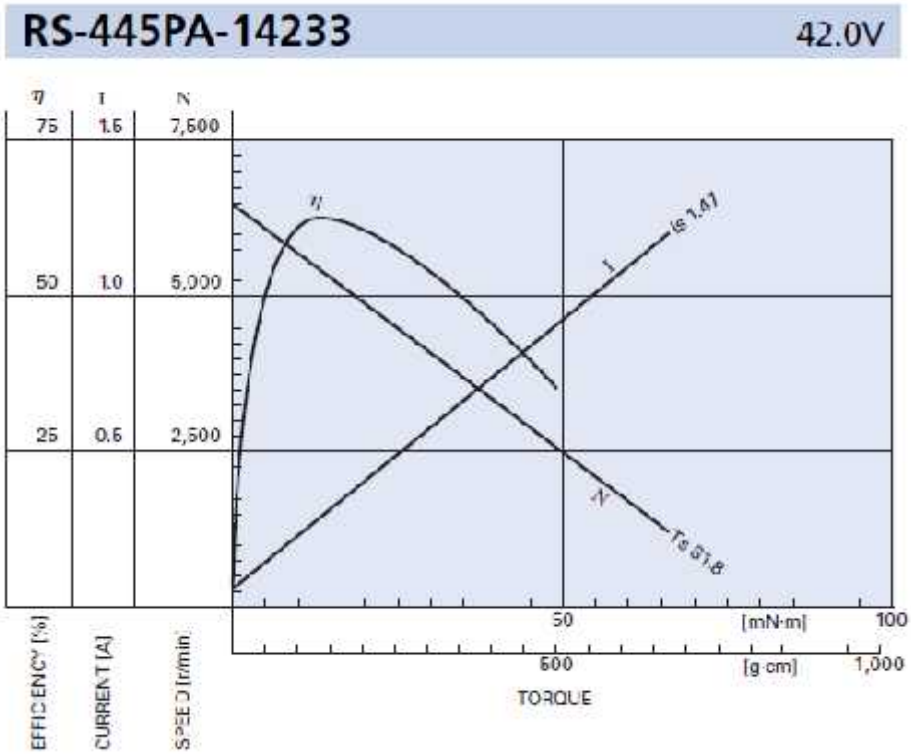


Figure (3.11) Motor specification

After substituting the variables values we get:

➔ $K_b = 0.06$ ➔ $K_m = 77.9 * 10^{-3}$ ➔ R_a=$40*10^{-3}$kilo-ohm

So now the value of first component inertia can be found through equation (4.20):

$$J = \frac{^2{}_m (K_b K_m)}{R_a} = \frac{0.133(0.06 * 77.9 * 10^{-3})}{40 * 10^3} \quad (3.48)$$

$$J = 1.55 * 10^{-5}\, kg.m^2$$

Now that we found the J value, all we have to do is to find the robot structure inertia, and add it to the motors inertia to get the equivalent inertia of the system.

3.7.1.2 Robot Structure inertia

The robot in this project can be considered under vehicles systems, meaning that vehicle dynamic can be applied on it.

In order to find the structures inertia, the traction force with ground must be taken under consideration, this leads to the following equations, figure (3.12).

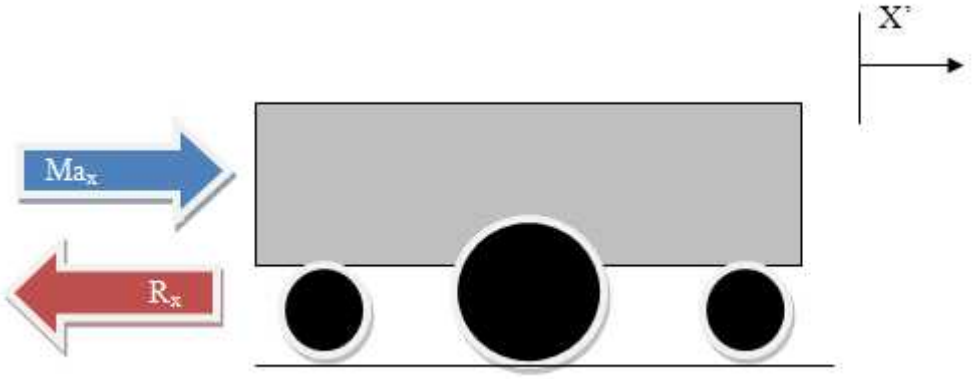


Figure (3.12) free body diagram (b)

Using Newton's law:

$$\rightarrow \sum F_x = Ma_x \qquad (3.49)$$

$$\rightarrow F_x = Ma_x + R_x$$

Where:

a_x: acceleration$= \frac{0.7-0}{1-0} = 0.7 m/s^2$; R_x: Rolling Resistance Force

$$F_x = \frac{w}{g} a_x + f_r wg \qquad (3.50)$$

f_r: friction factor for low friction surface

$$F_x = \frac{3.95}{9.81}0.7 + 0.002 * 3.95 * 9.81 = 0.35N$$

And also, $$F_x = \frac{T_e N n}{r} - \frac{J_{structure} a_x}{r^2} \quad (3.51)$$

This leads to $$J_{structure} = \frac{T_e N n r - F_x r^2}{a_x} \quad (2.52)$$

$$J_{structure} = \frac{0.01 * 6 * 0.85 * 6 * 10^{-2} - 0.35 * (6 * 10^{-2})^2}{0.7} = 0.0034 \text{ kg.m}^2$$

Now that found the structures inertia, we can sum it to the motors inertia,

$$J_{eq} = J_m + J_{structure} \quad (3.53)$$

$$J_{eq} = 1.55 * 10^{-5} + 3.4 * 10^{-4} = 3.4155 * 10^{-3} \text{kg.m}^2$$

So now the overall transfer function formula will be like this after modifying equation (4.20), and we can neglect the internal damping (b) since its value is too small:

$$^2 = \frac{R_a J_{eq}}{(K_b K_m)}$$

$$= \frac{40 * 10^{-3} * 3.4155 * 10^{-3}}{(0.06 * 77.9 * 10^{-3})} = 0.028$$

So the transfer function will be according to equation (4.19):

$$C(s) = \frac{s^{-}(s)}{V_a(s)} = \frac{K}{s + a} = \frac{K_m / (R_a b + K_b K_m)}{^2 s + 1}$$

$$= \frac{1/(K_b)}{^2 s + 1} = \frac{1/0.06}{0.028s + 1}$$

$$\rightarrow\rightarrow C(s) = \frac{\bar{s}(s)}{V_a(s)} = \frac{16.667}{0.028s + 1}$$

Now that we found the overall transfer function, we can use to design the controller in chapter four.

Chapter 4

Control design

4.1 Introduction to Control System

The purpose of control is to force or change the behavior of a system to follow a desired criterion.

In this project, two control loops will be used; the first one consists of the operator and a high level software, the operator supervises the whole operation through the video feedback transmitted from the robot through the internet, and controls it through the high level software which will be programmed by LABVIEW, this loop depends on the operator himself and how fast he will be in the reaction with deferent situations.

The second loop is an autonomous control in the robot itself, this loop consists of the sensors (discussed in chapter three) and the controller that controls the position of the robot. The design of this controller will be discussed later in this chapter.

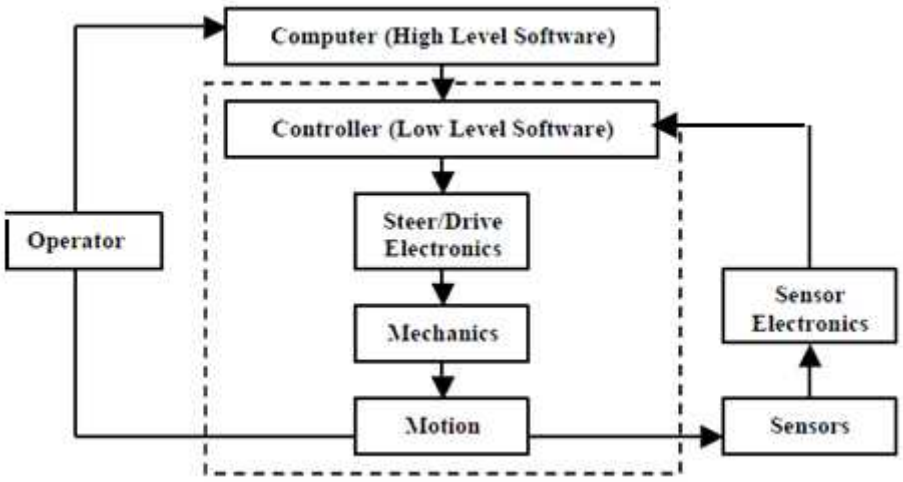


Figure (4.1) Control Loops:

Left loop with operator Right loop autonomous control with sensors

4.2 Closed Loop Control System

To obtain accurate control of a process during the execution of an action, feedback control is used where the variables that are being controlled are continually measured by feedback then compared to reference (error calculation), and the action is modified according to a control law implemented in the PIC controller to overcome the error.

When speaking about a motor in open loop control, there is no feedback from the motor, telling the robot program how fast the wheels are turning or how far the robot has gone, and so to implement true velocity- or position-control algorithm, the robot needs sensors on wheels, such as the shaft encoder for position or velocity feedback which was mentioned earlier. Such feedback enables what are known as closed loop control algorithms. Figure (4.2) represents the main configuration of the closed loop system for a DC motor.

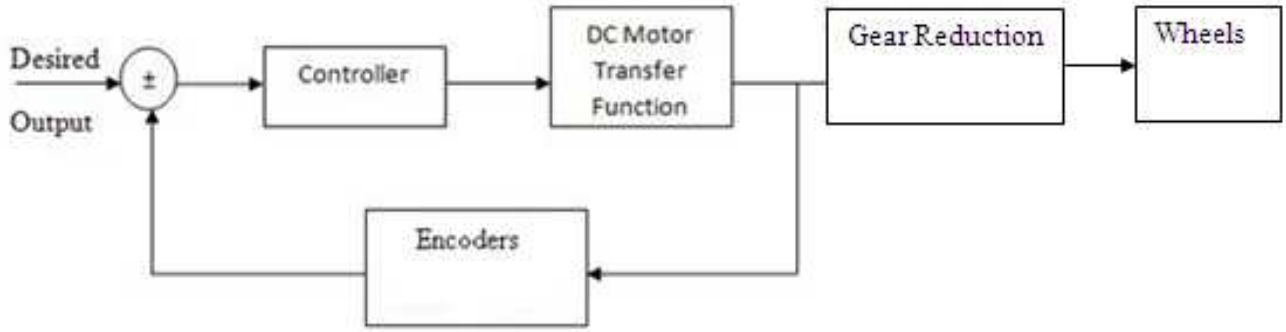


Figure (4.2) Closed loop configuration

The fundamental reason for using feedback, despite its cost and additional complexity are [12].

1. Decrease in the sensitivity of the system to variations in the parameters of the process.
2. Ease of control and adjustment of the transient response of the system.
3. Improvement in the rejection of the disturbance and noise signals within the system.
4. Improvement of transient response and reduction of steady-state error of the system.

4.3 DC motor Transfer function

The dc motor is a power actuator devise that delivers energy to a load, as shown in figure (4.3) [14].

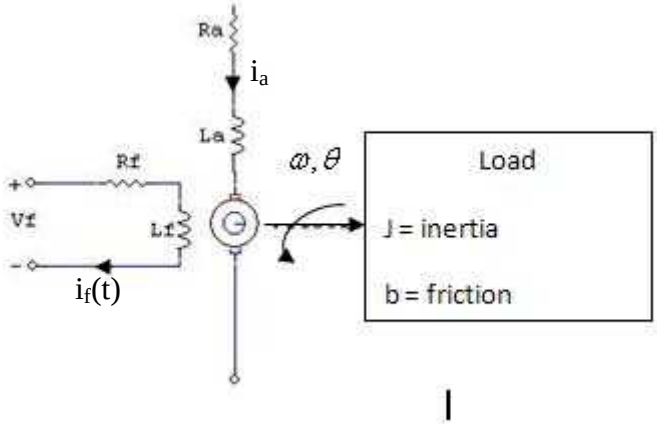


Figure (4.3) dc motor wiring

The dc motor converts direct current (dc) electrical energy into rotational mechanical energy. A major fraction of the torque generated in the rotor (armature) of the motor is available to drive and external load. Because of feature such as high torque, speed controllability over wide range, portability, well-behaved speed-torque characteristic, and adaptability to various types of control methods, dc motors are still widely used in numerous control application, including robotic manipulators, tape transport mechanism, disk drives, machine tools, and servo valve actuators.

The transfer function of the dc motor will be developed for a linear approximation to an actual motor, and second-order effects, such as hysteresis and the voltage drop across the brushes, will be neglected. The input voltage may be applied to the field or armature terminals. The air-gap flux of the motor is proportional to the field current, provided the field is unsaturated, so that

$$¤ = K_f i_f \tag{4.1}$$

The torque developed by the motor is assumed to be related linearly to ¤ and the armature current as follows

$$T_m = K_f i_f(t)¤ = K_1 K_f i_f(t) i_a(t) \tag{4.2}$$

It is clear from eq. (2) that, to have a linear element, one current must be maintained constant while the other current becomes the input current. First, we shall consider the field current con-

trolled motor, which provides substantial power amplification. Then we have in Laplace transform notation

$$T_m(s) = (K_1 K_f I_a) I_f(s) = K_m I_f(s) \tag{4.3}$$

Where $i_a = I_a$ is a constant armature current, and K_m is defined as the motor constant. The field current is related to the field voltage as

$$V_f(s) = (R_f + L_f s) I_f(s) \tag{4.4}$$

The motor torque $T_m(s)$ is equal to the torque delivered to the load. This relation may be expressed as

$$T_m(s) = T_L(s) + T_d(s) \tag{4.5}$$

Where $T_L(s)$ is the load torque and $T_d(s)$ is the disturbance torque, which is often neglected. However, the disturbance torque often must be considered in system subjected to external forces such as antenna wind-gust forces. The load torque for notating inertia as shown in figure (4.3) is written as

$$T_L(s) = Js^{2\bar{}}(s) + bs^{\bar{}}(s) \tag{4.6}$$

Rearranging eq. (4.3) and (4.5), we have

$$T_L(s) = T_m(s) - T_d(s) \tag{4.7}$$

$$T_m(s) = K_m I_f(s) \tag{4.8}$$

$$I_f(s) = \frac{V_f(s)}{R_f + L_f s} \tag{4.9}$$

Therefore the transfer function of the motor-load combination, with $T_d(s) = 0$, is

$$\frac{\bar{}(s)}{V_f(s)} = G(s) = \frac{K_m}{s(Js+b)(L_f s + R_f)} = \frac{K_m / JL_f}{s(s + b/J)(s + R_f / L_f)} \tag{4.10}$$

The block diagram model of the field-controlled dc motor is shown in figure (4.4).

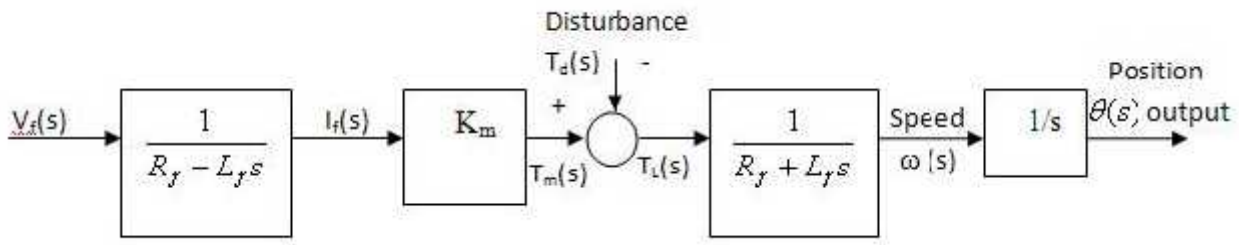


Figure (4.4) Block diagram model of field controlled dc motor

Alternatively, the transfer function may be written in terms of the time constant of the motor as

$$\frac{\theta(s)}{V_f(s)} = G(s) = \frac{K_m / bR_f}{s(\tau_f s + 1)(\tau_L s + 1)} \tag{4.11}$$

Where $\tau_f = L_f / R_f$ and $\tau_L = J / b$. Typically, one finds that $\tau_L > \tau_f$ and often the field time constant may be neglected.

The armature-controlled dc motor uses the armature current i_a as the control variable. The stator field can be established by a field coil and current or a permanent magnet. When a constant is established in a field coil the motor torque is

$$T_m(s) = (K_1 K_f I_f) I_a(s) = K_m I_a(s) \tag{4.12}$$

When a permanent magnet is used we have

$$T_m(s) = K_m I_a(s) \tag{4.13}$$

Where K_m is a function if the permeability of the magnetic material.

The armature current is related to the input voltage applied to the armature as

$$V_a(s) = (R_a + L_a s) I_a(s) + V_b(s) \tag{4.14}$$

Where $V_b(s)$ is the back electromotive-force voltage proportional to the motor speed. Therefore we have

$$V_b(s) = K_b \mu(s) \tag{4.15}$$

And the armature current is

$$I_a(s) = \frac{V_a(s) - K_b \mu(s)}{(R_a + L_a s)} \tag{4.16}$$

Equation (4.6) and (4.7) represent the load torque so that

$$T_L(s) = Js^2\overline{\ }(s) + bs\overline{\ }(s) = T_m(s) - T_d(s) \tag{4.17}$$

The relation for the armature-controlled dc motor is shown schematically in figure (4.5).

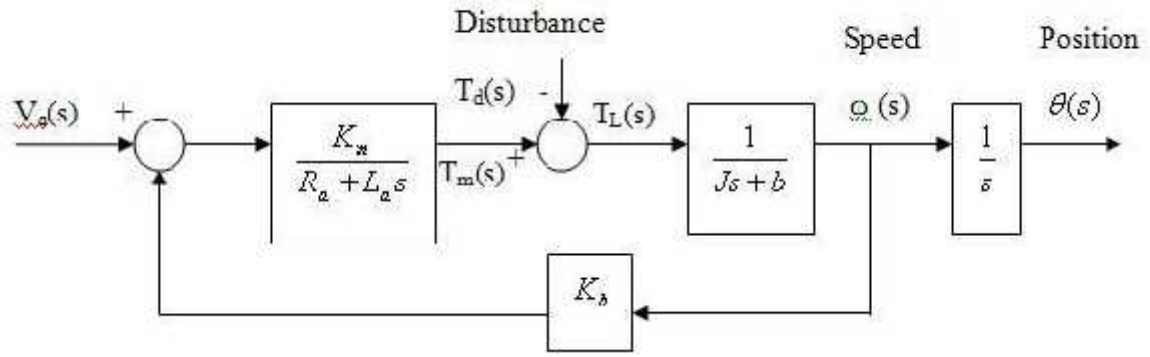


Figure (4.5) Armature-controlled dc motor

Using equation (4.12), (4.16) and (4.17), or, alternatively, the block diagram in figure (4.5), we obtain the transfer function (with $T_d(s) = 0$)

$$\begin{aligned} G(s) &= \frac{\overline{\ }(s)}{V_a(s)} = \frac{K_m}{s[(R_a + L_a s)(Js + b) + K_b K_m]} \\ G(s) &= \frac{\overline{\ }(s)}{V_a(s)} = \frac{K_m}{s(s^2 + 2\P\mu_n s + \mu^2)} \end{aligned} \tag{4.18}$$

However, for many dc motors, the time constant of the armature, $^2{}_a = L_a / R_a$, is negligible, and therefore

$$G(s) = \frac{\overline{\ }(s)}{V_a(s)} = \frac{K_m}{s[(R_a(Js + b) + K_b K_m]} = \frac{[K_m /(R_a b + K_b K_m)]}{s(^2{}_1 s + 1)} \tag{4.19}$$

Where $$^2{}_1 = R_a J /(R_a b) + K_b K_m) \tag{4.20}$$

It is of interest to note that K_m is equal to K_b. this equality may be shown by considering the steady-state motor operation and the power balance when the rotor resistance is neglected. The power input to the rotor is $(K_b\omega)i_a$, and the power delivered to the shaft is T_w. In steady-state condition, the power input is equal to the power delivered to the shaft so that $(K_b\omega)i_a = T\omega$; since $T=K_m i_a$ equation (12), we find that $K_b= K_m$.

And after the motor transfer function is calculated, all we have to do next semester is an experiment to find the values of Motor constant (K_m), equivalent inertia of the rotor shaft added to the robot structure inertia (transferred to the rotor shaft) (J_m), field time constant (τ_f), Rotor time constant (τ).

4.4 Speed Control of DC motor

The control law in this project is in involved with position control of the motors as described in section one. But though, speed of motors will be fixed to some values according to the direction of motion as described in section (2.6.2), and since the user will send (x, y) coordinate to the robot, which means that the robot might rotate for a calculated value of angle, and so the motors velocity will be reduced to (0.4m/s) in order to reduce the slip and the effect of the structures inertia.

The most famous method of controlling the speed of DC motor is armature terminal voltage. The speed is easily controlled from zero to maximum safe speed in either forward or reverse directions using this method.

In application in which power source is a battery, various circuits provide variable armature terminal voltage to DC motor, as a means of speed control. It must be noted that the DC value of a voltage or a current is its average value. The driving circuit is essentially a switch that turns on the battery for short time intervals. It may vary the average or the DC value of the terminal voltage by varying the pulse width (pulse width modulation, or PWM described in chapter 2), or pulse frequency (PFM) see figure (4.6).

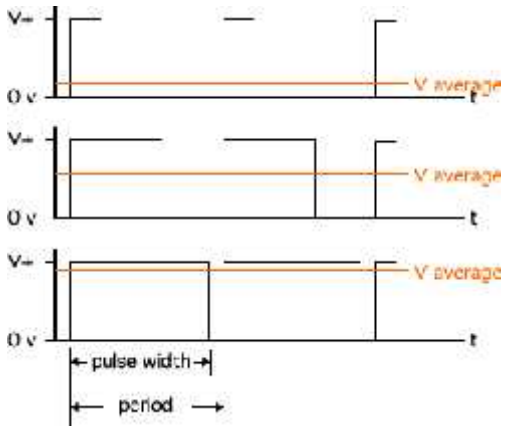


Figure (4.6) PWM

$$V_{avg} = (t_{on}/T)*V \tag{4.21}$$

Where t_{on} is the time where the pulse equals V,

V is the maximum voltage could be applied,

T is the time for one cycle and equals ($t_{on}+t_{off}$),

t_{off} is the time where the voltage equal zero.

4.5 Control Circuit Design

4.5.1 Overview

As we said in section one that the purpose of control is to force or change the behavior of a system to follow a desired criterion, and to do so a suitable controller must be designed and implemented to get the desired stable criterion.

In this project this criterion is position, this is what we want to control, where the motion of the robot must be controlled during moving to the given coordinates as we mentioned before in order to be able to perform the driving and steering operation as needed, so we need a position controller.

In general, let us talk about stability, and what affects it. Stability is defined as to maintain the system at the desired criterion that we want, what affects it, is the poles of the process transfer function. Where the addition of stable poles on the open loop Transfer function increases the stability, and the addition of stable zeros increases the stability and speeds up the system of the closed loop response.

There are various types of controllers can be used for position control, such as [12]:

1) P controller:

It's also called static compensator, where it makes a change to the output that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant K_p, called the proportional gain.

$G_c(s) = K$ Where K is constant

2) PD controller:

It adds a zero to the transfer function of the motor, and it's used to improve the transient response problems.

$G_c(s) = K_p + K_D * s$ Where K_p, K_D are constants

3) PI controller:

It adds a zero and a pole to the transfer function of the motor, and it's used to improve the steady state error in the process.

$G_c(s) = (K_p*s + K_i)/s$ Where $K_{p,}$ K_i are constants

4) PID controller:

It adds two zeros and one pole to the transfer function of the motor, and it's used to improve transient response and steady state errors in the process.

$G_c(s) = (K_i + K_p*s + K_D*s^2)/s$ Where $K_{p,}$ K_i , K_D are constants

5) Lead controller:

It's a generalization of PD controller, used to improve transient response.

$G_c(s) = K_c \sim (T_s+1)/\sim T_s+1$, $0<\sim<1$

Where $K_{c,}$ ~ and T are constants

6) Lag controller:

It's used to improve steady state errors.

$G_c(s) = K_c \quad (T_s+1)/ \quad T_s+1$, $1<$

Where is constant

7) Lag lead controller:

It's used to improve both transient and steady state errors response.

$G_c(s) = G_{lag}(s)*G_{lead}(s)$

8) State feedback, or pole placement:

Is a method employed in feedback control system to place the closed-loop poles of a plant in pre-determined locations in the s-plane. Placing poles is desirable because the location of the poles corresponds directly to the eigenvalues of the system, which control the characteristics of the response of the system [17].

4.5.2 Controller Design

4.5.2.1 P Controller Design

P controller was chosen for many reasons, listed in the following points:

1) Due to its simplicity and that it does not require a huge control effort.
2) It was experimented on matlab and showed a perfect performance and results.
3) The final reason for choosing this type of controller is that we suffered shortage of internal timers on PIC18F4550. The number of internal timers is four as mentioned before, and they were all used to operate the motors and the encoders, so it was impossible for us to use any of these timers to implement a PD, PI or State feedback controllers since they all need timers to capture a sampling period of time required for differentiating or integrating.

Now back to our controller, Through the transfer function we got from chapter three, we can start designing our controller using matlab, were the transfer function were build inside the matlab, and then we used the SISO TOOL to calibrate our controller to get the best performance, but first we had to calculate the gain voltage ratio between the PIC and the H-bridge, were there is a ratio between the output voltage of the PIC and the output voltage of the H-bridge.

This value was calculated using matlab, were we made an experiment through rising the duty cycle in a step of 20 from its minimum value to its maximum as described before (0-396), and while we was rising the duty cycle value, we recorded the voltage values of PWM that is going from the PIC to the H-bridge, and also we recorded the output of the H-bridge, then we transferred the data to matlab and we got the relation between the PIC voltage and H-bridge voltage which is represented by a gain between the plant and the input. The experiment data is placed in appendix F.

The battery voltage when the experiment was made was 19v; the following table shows the captured data.

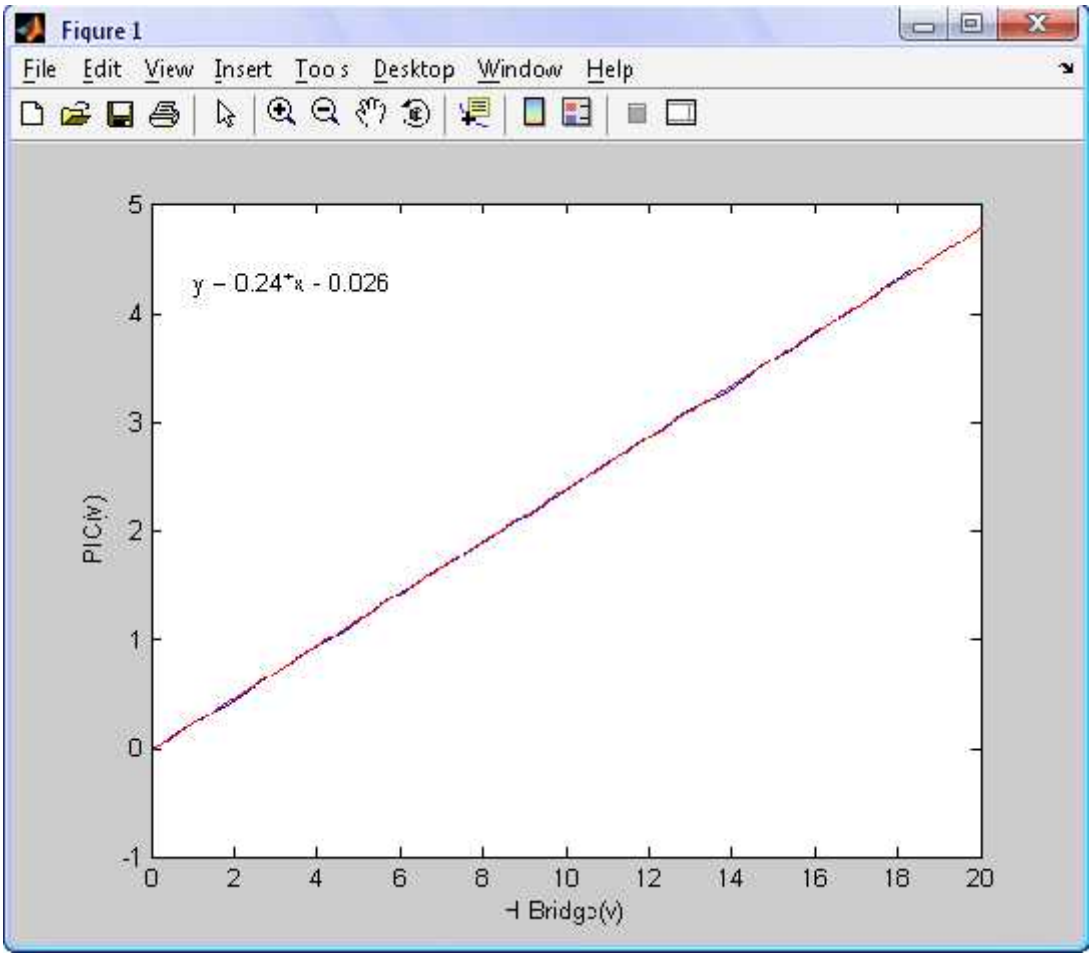


Figure (4.7) Relation between PIC voltage and H-bridge voltage

So we can see from figure (4.7) that the relation is linear between them, and the gain4 in the next figure can be considered (0.24) since (0.026) would not have a large effect on the gains value.

Now since we calculated the gains value, and we have already found the transfer function from chapter three, we can start the controller design.

Open loop model:

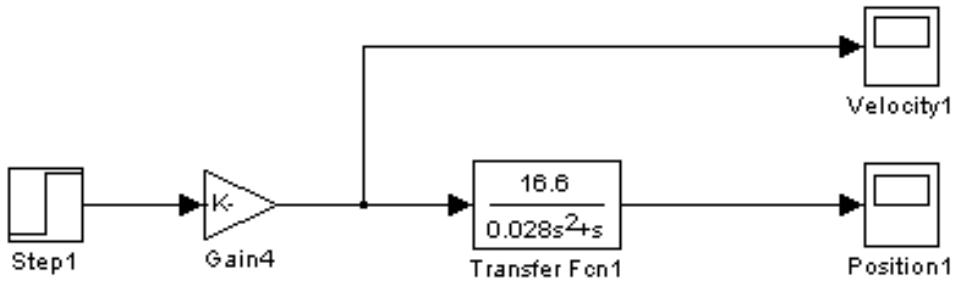


Figure (4.8) open loop model

Where the step input for this model was 15 rad≈1meter and time is in seconds.

Response of the open loop model:

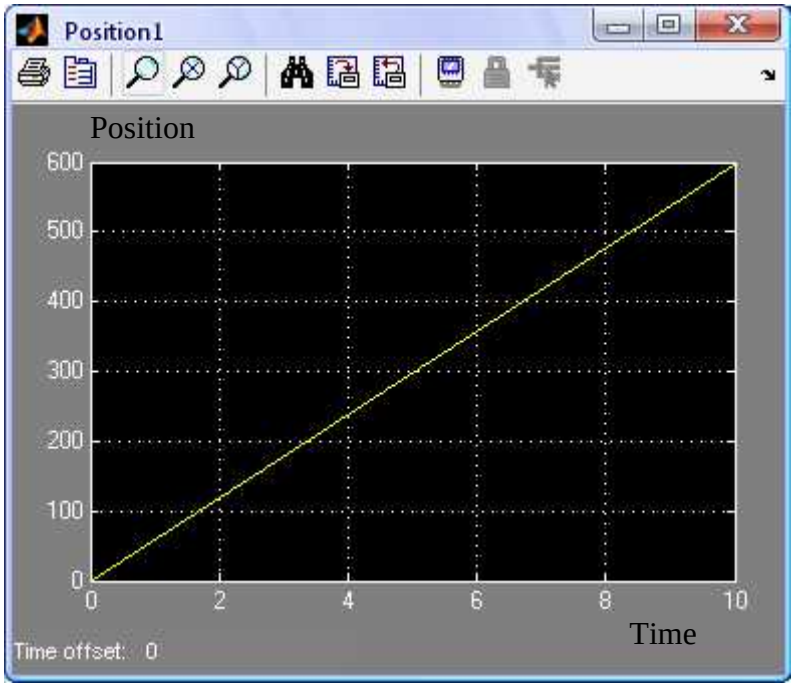


Figure (4.9) open loop response of position

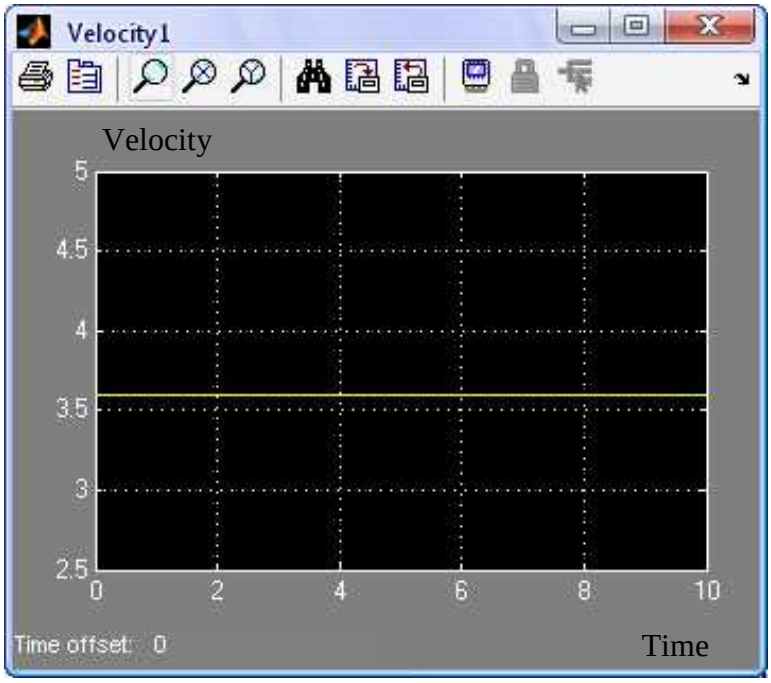


Figure (4.10) open loop response of speed

We can see from the previous figures that the position of the robot will not reach steady state and will increase with time as long as the PWM is opened, and also the speed will not decay to zero unless the PWM is closed from the PIC meaning that the step input becomes zero.

So now we need a controller to ensure an accurate tracking for the input for the position of the robot, and also a speed that decays to zero when the robot reaches its new destination.

Closed loop model:

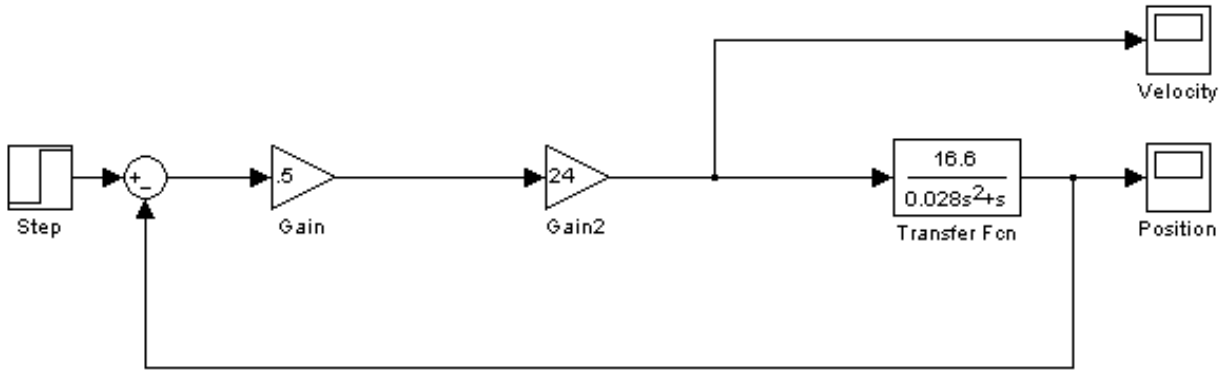


Figure (4.11) closed loop model

For the same input value we can monitor the velocity response and the position response were:

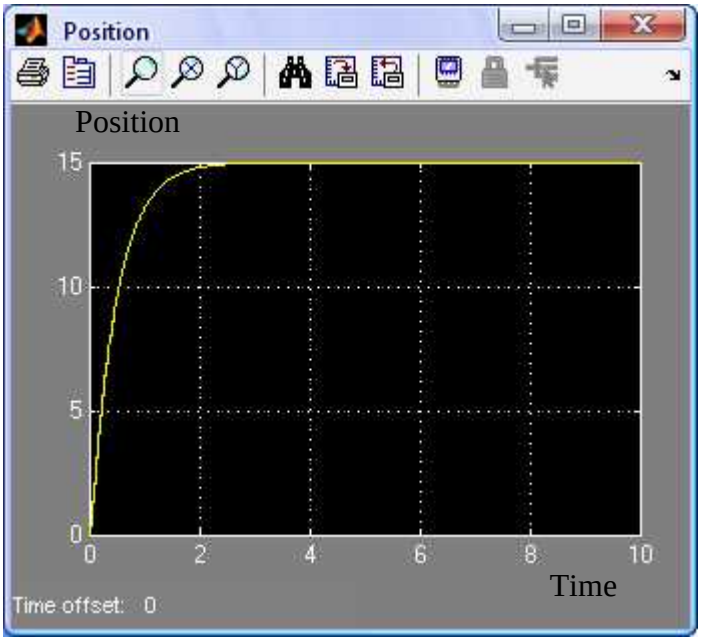


Figure (4.12) closed loop response for position

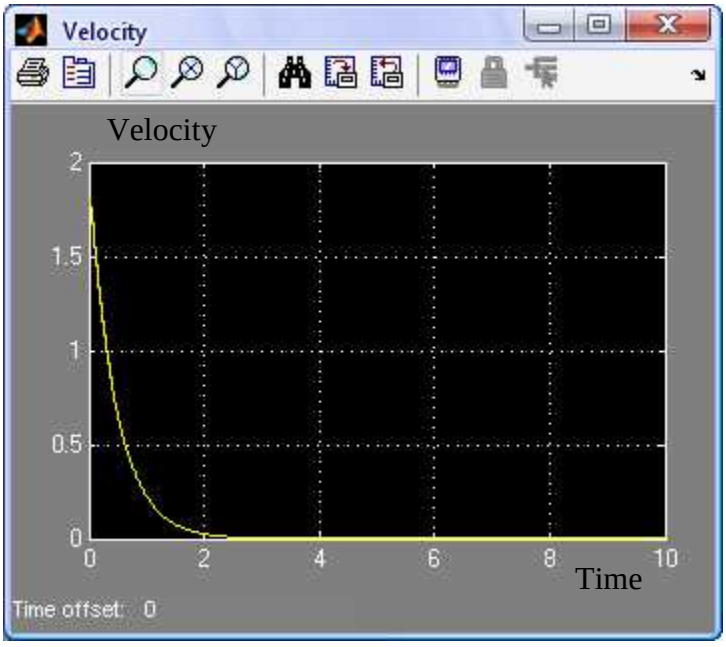


Figure (4.13) closed loop response for velocity

We can see from figure (4.12) and figure (4.13) that the position of the robot has reached the input value in an acceptable time, whereas the velocity decayed to zero after short time, meaning that the robot has stopped without the need of closing PWM.

The value of gain K was calibrated using the on matlab as follow until we got an acceptable response for the system, were we took in our consideration an acceptable time to complete a task, avoid overshooting and an acceptable velocity that would not harm the hardware components.

This gain is used to correct the error in the system through multiplying it with the encoders reading as a deference equation programmed on the PIC.

Now what if a disturbance affected the system, such disturbance may be from one motor on the other one, since they are both supplied from the same source; also the disturbance may be due to low performance of regulators or H-bridge due to heat effect.

Such disturbances will affect the response of the system, and to show there effect a simulation had been experimented on matlab as follow:

Closed loop with disturbance:

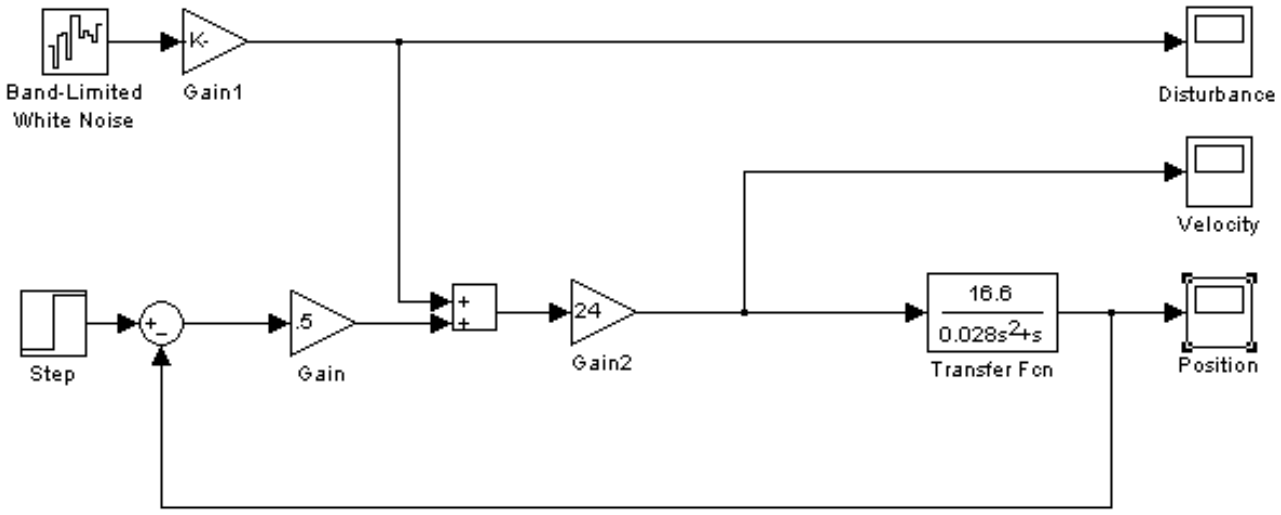


Figure (4.14) closed loop model with disturbance

Noise signal generated using matlab, it represent any losses or disturbances that can affect the system.

Let us try a noise value of (0.01), assuming that this is the effect of heat losses in ICs, see figure (4.15).

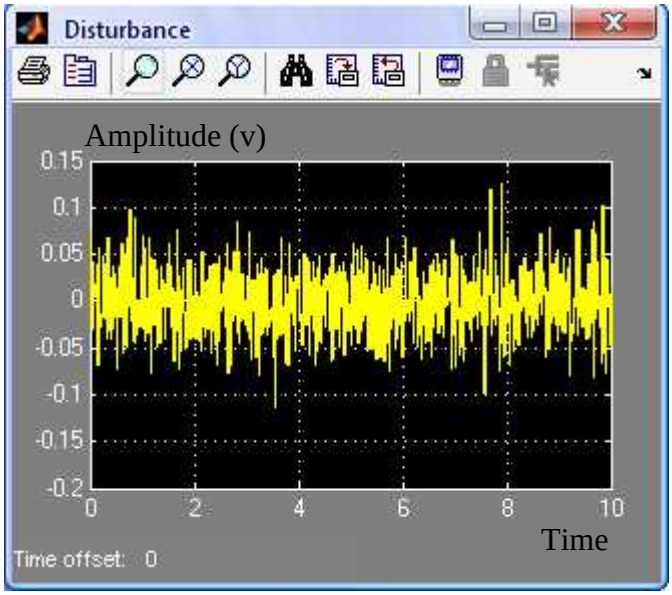


Figure (4.15) generated noise signal with gain (0.01)

Response of system with disturbance:

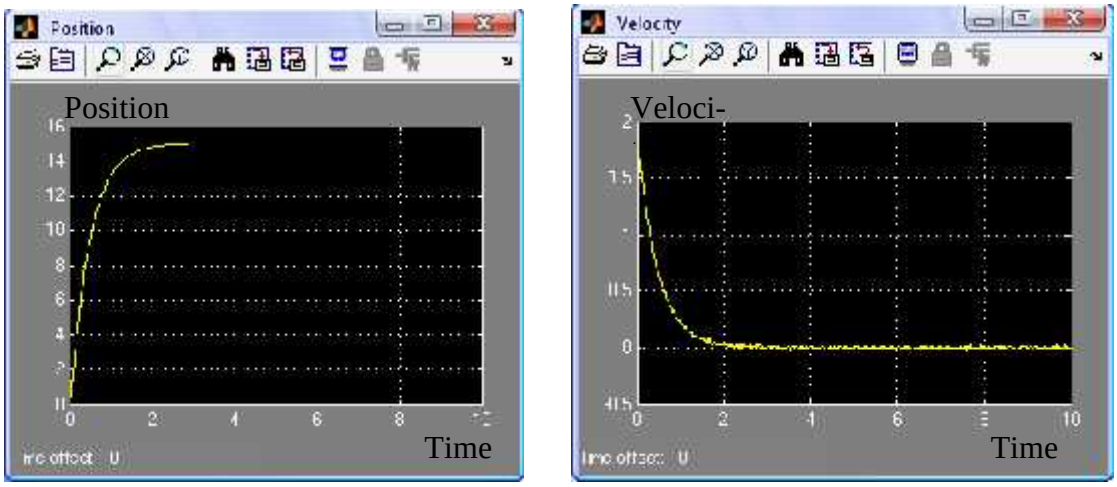


Figure (4.16) Response of system with disturbance of (0.01) position to the left and velocity to the right

We can see from the previous figure that the effect of the generated noise did not affect the response that much, meaning that the controller was able to compensate for any value that equals or is less than this value.

Now what if the system is affected by a larger value of noise, let us assume a noise gain that equals (0.1), the system response would be:

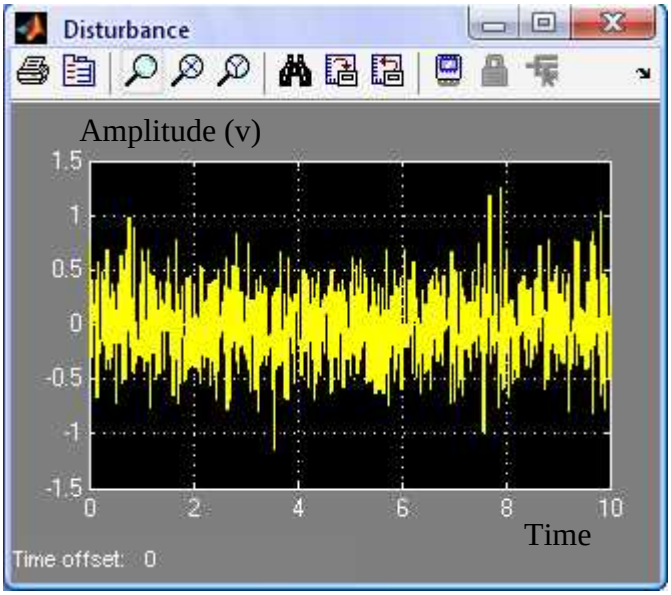


Figure (4.17) generated noise signal with gain (0.1)

Response of system with disturbance:

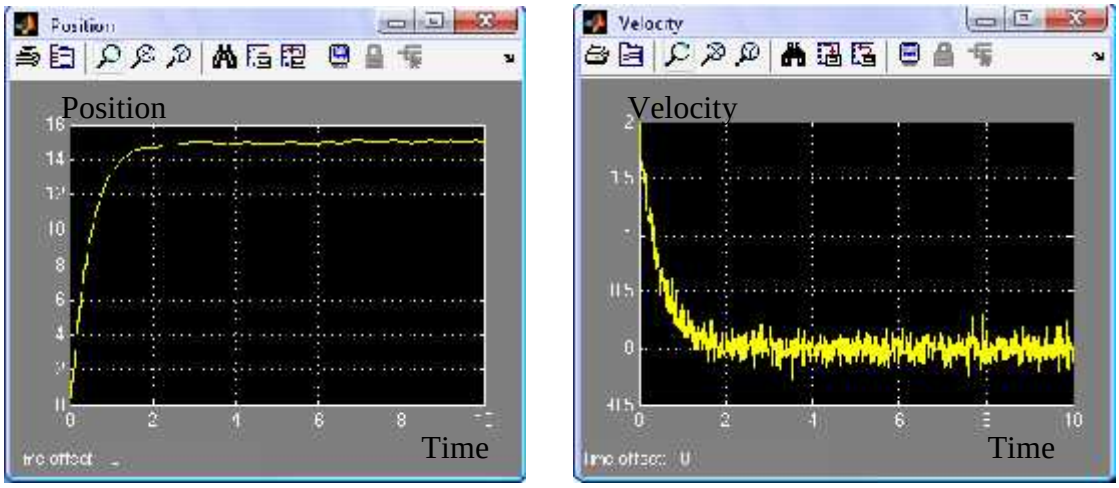


Figure (4.18) Response of system with disturbance of (0.01) position to the left and velocity to the right

We can see from the previous figures that such a disturbance with this value would make the job difficult for the controller to compensate the error, especially the velocity error, and this is due to high frequency the system inhabits during such a disturbance.

And here comes the need for the other type of controllers, so they would be able to track the input and keep the response of the system in a perfect way. Such controllers that might be used for this case is the state feedback controller and robust tracking controller, since they have the ability of overcoming the disturbances and insure a stable and fast response.

4.6 Computer Process Control

Computer process control is defined as the use of a stored program in digital computer to control an industrial process. For the computer to be used in process control; the computer must collect data from the operation to process. If the computer is utilized to control process directly, data must be communicated to the process. The data flowing back and forth between the computer and the process can be classified into three types:

1. Continuous analog signals.

   A continue variable is one that assumes a continuum of values over time. The variable is uninterrupted as time proceeds; an example of it is force, temperature, pressure, velocity.

2. Discrete binary data.

   Which are the data that can either of two possible values, such as on or off, open or close.

3. Pulse data or discrete that is not restricted to binary; in other words, more than two values are possible. Pulse data are a train of pulse signals. This type is used in digital sensors as optical encoders.

And since this project has a PIC micro-controller, and the gain of the controller had been found in the previous section, the difference equation and its programming is under development.

4.7 Implementation of intelligent system

During the operation of the robot, the network coverage might be lost, and at this stage, the robot will run autonomously, which means it will need a routine program to execute during absence of network coverage and other status. This program will be programmed inside the PIC controller to help the robot to avoid obstacles during the absence of supervision, it will depend on the sensors signals and according to these signals, the robot will find his way to avoid the obstacle and then return to its original path and complete it operation. If the robot faces a dead end, it will hold still in its place until the network is back again, so that the user is back in the process again to solve the problem and get the robot away from disturbance.

4.7.1 Problem Definition

Since a two independent motorized middle wheel is implemented in this project, and each one has one input, we can control two variables on the plane (as discussed before), in this case we will use (x_f, y_f) to define the final position of the robot, and then leave the path planning techniques which is under development in the mean while [18].

The problem to direct the robot from an initial posture to a final position (goal) while avoiding an obstacle is defined as:

The robot is at a specific posture (x_i, y_i, $\bar{}_i$), and it's given new coordinates to reach (x_f, y_f) while avoiding obstacles if they are in the robots path, this problem can be described for the following aspects:

4.7.1.1 Reaching the Goal

The problem of reaching the goal in figure (4.20) is defined as the direction and velocity in which the robot should be directed in order to reach the goal.

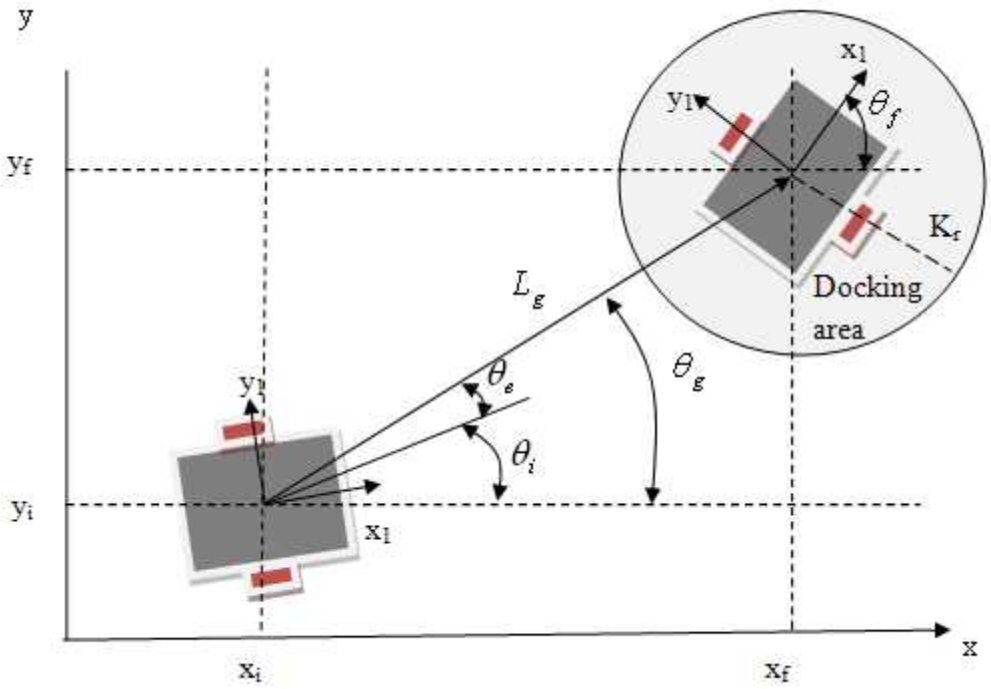


Figure (4.19) problem definition of reaching the goal

First of all, let us recall the equations (2.14/15) which we derived in chapter 2, section (2.7.3), and they are:

$$L = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$$

$$- = \tan^{-1}\frac{(y_2 - y_1)}{(x_2 - x_1)}$$

And we can modify them according to our problem definition, so they would look like:

$$L_g = \sqrt{(y_f - y_i)^2 + (x_f - x_i)^2} \tag{4.22}$$

$$-_g = \tan^{-1}\frac{(y_f - y_i)}{(x_f - x_i)} \tag{4.23}$$

And the error angle would be:

$$\bar{}_e = \bar{}_g - \bar{}_i \tag{4.24}$$

Then the robot is directed from initial posture $\P_i = (x_i, y_i, \bar{}_i)$, to the final position using the following rules:

$$v = \begin{bmatrix} v_{\max} & if \left| d_g \right| > K_r \\ \dfrac{v_{\max}}{K_r} & f \left| d_g \right| \le K_r \end{bmatrix} \tag{4.25}$$

Where v_{max} is the maximum heading velocity (0.7m/s) and K_r (equals 1 meter) is the radius of the docking area.

$$w = w_{\max} \sin(\bar{}) \tag{4.26}$$

Where w_{max} is the maximum angular velocity of the robot platform.

With these rules (4.25/26), the robot is directed to the final position at the maximum linear velocity when it is outside the docking area, and it will be approaching slowing down when its inside the docking area.

On the other hand the robot will be correcting the heading angle as a function of the sine of the error angle $\bar{}_e$, where the maximum angular velocity $w=\pm w_{max}$ will be achieved at $\bar{}_e = \pm 90°$, and the minimum angular velocity w=0 will be $\bar{}_e = 0°$.

4.7.1.2 Obstacle Modeling

Since the robot may encounter obstacles while its executing operation, so it must have an intelligent procedure that it can follow in order to avoid that obstacle, this procedure is under development for the mean while. And it's going to be programmed in the PIC in a procedure that responds to the sensor that interrupts the operation indicating that an obstacle is reached or in the docking area of around the robot.

4.8 Delay in control process

In its general sense, delay refers to a lapse of time. In this project delay must be involved in the system in order to be able to deal with it and implement it in the controller and compensate it when required.

This project faces three components or delays ($T_{total}=T_m+T_n+T_h$):

1. (T_m) The processing time consumed by the micro controller during execution of instructions, which had been reduced to its minimum value through the interrupts technique, were the PIC deals with objects and operation only if it happens, without the need of checking for change every period of time, and for the time consumed during the ADC operation, it was reduced also through using the same concept which is interrupts.

2. (T_n) Delay in the network, this value varies according to the pressure the network has, it's discussed in the computer and communication part. But since the robot will get the coordinates from the user, this component of delay will not affect the performance, because the new coordinate will be transmitted to the robot, and the robot will immediately start calculating the angle and the distance to move, then executes them, and so it will not depend on the user to complete the job.

3. (T_h) Human interaction delay, which is a period of time elapses between the recognition of the object, and making the decision to give new orders [13]. And in order to be able to find a specific value for this component, we must study the reaction time of humans in general and take the maximum value. See table (4.1).

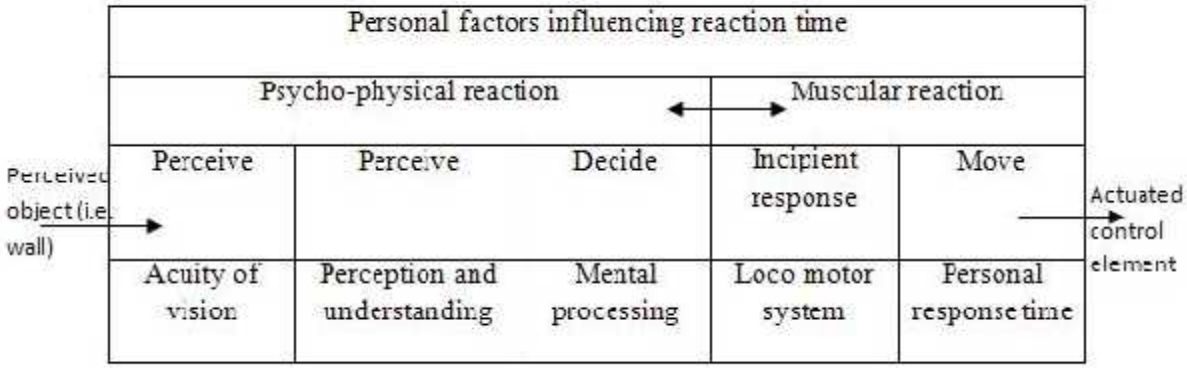


Table 4.1 Reaction Time

The reaction time is not fixed value; it ranges from 0.3 to 1.7 s, depending upon the user and on external factors. See table (4.2).

	Reduction down to 0.3 s ← → Extension up to 1.7 s	
A) Personal factors	Trained, reflexive response	Inexperienced response
	Good frame of mind, optimum performance potential	Poor frame of mind e.g. fatigue
	Highly skilled user Youth	Low level of user skill Advanced age
	Anticipation	Inattentiveness, distraction
	Physical and mental health	Health disorders in response groups
B) Extraneous factor (obstacle situation)	Uncomplicated, easily comprehended, predictable, familiar.	Complicated, difficult to comprehend, unpredictable, rarely encountered
Type of perceived object	Explicit, Conspicuous	Equivocal, inconspicuous
Location of perceived object	Within field of vision	At edge of field vision
Nature of control element	Logical layout of mechanic control	Poor layout of mechanical controls

Table 4.2 effect of personal and extraneous factors on reaction time

Chapter Five

Testing & Experiments

5.1 Introduction

This chapter shows all the results and experiments that had been experimented on the robot were the wheel arrangement, open loop test, closed loop test, sensors, transceiver, micro controller and all other components were tested separately then all together to ensure a perfect performance of the robot.

5.2 Wheel Arrangement

In this experiment, we tested the alignment of the wheels and whether they are arranged in the right place or not.

This test was made after the wheels and gears were implemented on the robot, and that was through connecting the battery directly to the motors, and see whether the robot will approximately move in a straight line or not to ensure that there is no misalignment in the structure and the wheel configuration.

This experiment was configured on three types of terrain, which are carpet, asphalt and flagstones.

On flagstones, the robot showed a fast but inaccurate performance due to slip of wheels on this type of terrain. So we made a modification on wheels through exchanging them with rubber wheels instead of the old plastic one, and after that the robot showed fast and accurate performance.

On asphalt and carpet, the robot moved accurately, but there was lack of power, since the first battery back supplied 12v 4Ah, so we increased the power of the robot through connecting another battery back of 12v 1Ah on series with the first one, so that the total power would be 24v 5Ah.

After that the robot showed fast, accurate and reliable performance.

5.3 PIC Programming

In this experiment we tried multiple codes on the PIC to ensure a good response during execution of code.

Some of these codes were simple yet important to adjust the timing of the internal clock of the PIC, and other codes to make sure that all output and inputs are working properly. LEDs were used to simulate the output of the PIC.

Other experiments on the PIC was with DC motors, where multiple codes were programmed on the PIC to monitor the DC motor performance, and it capabilities of changing direction suddenly and continuously with variable values of voltage.

These codes are in Appendix F.

5.4 Voltage ratio between the PIC and the H-bridge

In this experiment we estimated the ratio of voltage between the PIC and the H-bridge, in order to calculate the gain ratio between these components.

We set a step of 20 for duty cycle from (0-396), and recorded the voltage value of the PIC and the H-bridge, and then we derived an expression to represent the ratio between them as Follow:

PIC(v)	H-Bridge(v)	Duty Cycle
0	1	20
0.42	1.9	40
0.64	2.8	60
0.86	3.7	80
1.08	4.7	100
1.3	5.5	120
1.52	6.4	140
1.74	7.4	160

PIC(v)	H-Bridge(v)	Duty Cycle
1.96	8.25	180
2.17	9.2	200
2.39	10.1	220
2.62	11	240
2.84	11.9	260
3.06	12.8	280
3.28	13.9	300
3.5	14.7	320
3.72	15.7	340
3.94	16.5	360
4.16	16.5	380
4.41	18.3	396

Table 5.1 Data collected from the voltage of the PIC & H-bridge experiment

The derivation is in chapter four, section (4.5.2.1)

5.5 Open Loop Experiment

In this experiment we programmed the robot to move in a pre-determined path, to check the PIC performance with timers and DC motors in a multi task operation.

Where the robot was programmed to follow a pre-determined path through moving forward for some value of delay, then turn left for other value of delay, and then move backward, then turn right and repeat the program, code is in appendix F.

The following figures show the actual and experimental paths:

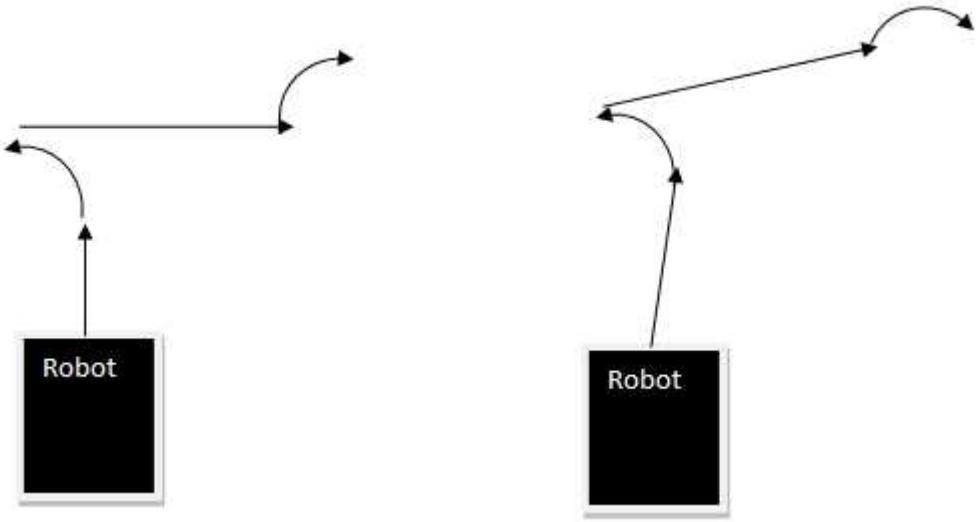


Figure (5.1) programmed path to the left and actual path to the right

Through this experiment we made sure that the timers of the PIC work properly, and the DC motors responds to sudden and normal changes in voltage value and direction despite the error of destination.

5.6 Encoders

In this experiment we made sure that the encoders give a true value reached, and checked its sensitivity, through connecting it to the flip flop circuit explained in Appendix E. then we moved the wheel manually without any supply, and the encoders showed high sensitivity to direction change whether the displacement was big or small.

After that we interfaced the encoders with the PIC through the flip flop circuit, to get a pulse and direction reading, code is in appendix F.

5.7 Ultra Sonic Sensors

In this experiment we checked the ultra sonic sensors sensitivity and accuracy, through interfacing them with the PIC, and we monitored the angle of sonar to realize the best position for sensors on the structure of the robot.

After that we implemented the sensors on the robot and monitored the stability performance of the sensors, to ensure that they will not detect any obstacle that passes in front of the robot for less than 1 sec.

Then we modified the angle of the sensors in order to prevent them from detecting the ground as an obstacle, also to make sure that we can detect the maximum distance around the robot.

After experiment the detection range was calibrated for the sides and rear sensors up 30cm, and for the front sensors up to 1.5m.

After that we checked the delay time during the ADC conversion for the eight sensors, to modify the speed of robot if necessary.

All codes and the procedure to deal with eight sensors are in appendix F.

5.8 Transceiver

The computer team checked the receiving rate of the transceiver and whether data is transmitted accurately to robot and it's all described in part two of this documentation

5.9 Moving the Robot According to a Pre-determined Path with the Assist of Encoders and Sensors

This experiment is divided into six parts, each part is involved with a particular operation of the robot, and they are as follow:

5.9.1 Closed Loop Response

In this experiment we tested the motors and encoders performance with the PIC, were we programmed a code to read from encoders and then compares the value of encoders with the goal distance, and check whether the robot reached that distance or not.

The code is appendix F.

The robot showed very good response in this experiment despite the absence of controller. The pre-determined path was programmed as follow:

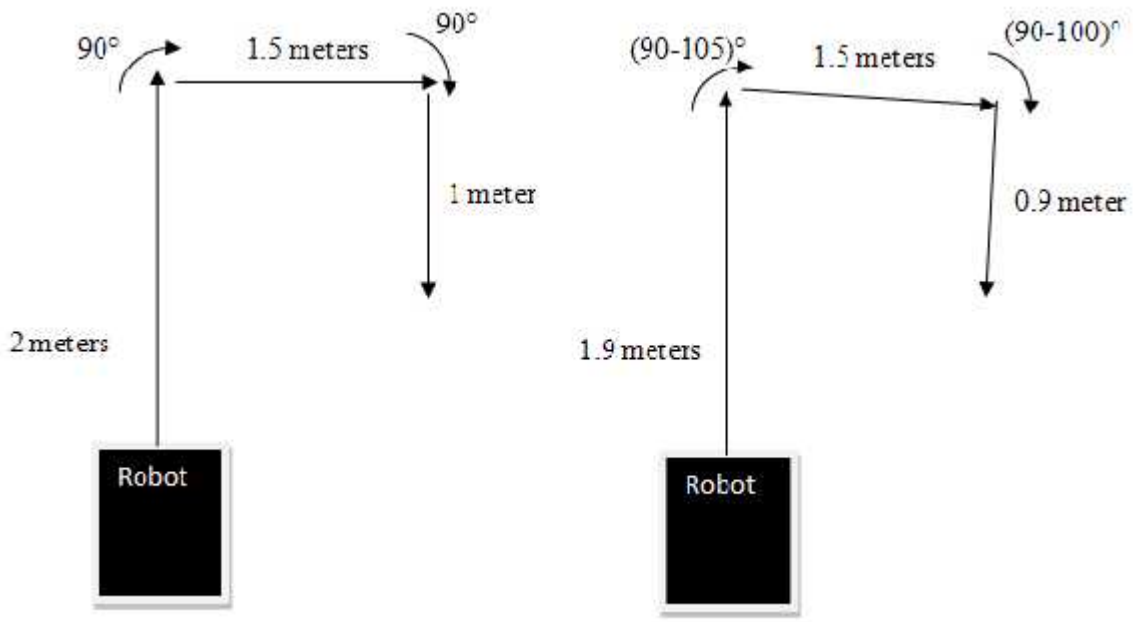


Figure (5.2) Pre-determined path to the left and actual path to the right

5.9.2 Checking for Obstacle

In this experiment we tested the ranges that the robot can detect, and also checked whether the speed of the robot suits the sudden stop after checking the obstacle.

We programmed the robot to search for a path that has no obstacle for a range of one meter, so that if the robot finds this path it will move toward it, after that if the obstacle exists, the robot will research for another path and repeat the sequence.

After this experiment the robot were able to find the path and move toward it for many times, were we but some obstacle in its way suddenly and normally, so that if the obstacle exists for more than one second, the robot will consider it as a real obstacle and will starts searching for another path.

Otherwise the robot stops for one second, and if the obstacle is no more in front of it, the robot will consider it as a negligible obstacle and resumes its operation.

5.9.3 Combining the Encoders with Ultra Sonic

In this experiment we tested the performance of the robot when it has encoders and ultra sonic sensors assistance.

The two codes of encoders and sensors were merged together into one code, see appendix F.

This code enabled the robot for searching for a path that has no obstacle in a range of one meter, and if this path exists the robot will execute the program of the closed loop response described before.

The robot showed a very good speed and position performance, though it has drifted a little distance from its destination, so the controller must be implemented to eliminate this drifting as much as possible.

5.9.4 Controller Assistance

After programming the controller in the PIC, the robot were able to reach the given distance accurately without any error, the encoders were used to measure the distance reached and then compared with the desired distance, then multiplied by 0.5 which is the gain of the P-Controller derived in chapter four.

5.9.5 Obstacle Avoiding Procedure

The robot in this experiment were able to avoid many types of obstacles, U shape, L shape and square shape obstacle, where the robot was given a distance to reach, and while moving, it faces the obstacle mentioned above, the robot showed a very intelligent and fast decision and performance of avoiding these obstacles

5.9.6 Receiving Coordinates from Internet

This experiment is described in the computer and communication part of this documentation.

5.9.7 Final Test

In this experiment the robot were given a distance to reach wirelessly, starts moving to reach the goal distance with the assistance of the encoders, ultra sonic sensors and the controller.

The robot responded accurately to the given commands, and performed immediately without any delays.

The video results of this experiment are included with the CD of this project.

Chapter Six

Recommendation and Future work

6.1 Introduction

This chapter introduces the recommendations for our project for the mechatronics part; in order to get better results and response.

Also it introduces the future work that might enhance the operation of the robot.

6.2 Recommendation

1) It would be better and much more reliable if the wheel configuration of the robot were modified in a way that ensures a full contact with ground for all wheels, whether the robot encountered a small stone or a rough road. One of the solutions might be through designing a suspension mechanism that ensures the required contact despite the terrain varying.

2) We recommend using another type of micro controllers that may have more than four timers, to be able to program a state feedback, robust tracking or PID controller, as we mentioned in chapter four. PIC18F8X20 and PIC18F6X20 can be used since they have five timers (timer0, timer1, timer2, timer3, timer4).

3) It would be more acceptable and much more accurate if the controller were replaced by state feedback or robust tracking controller, to overcome any disturbance that may affect the system as we described in chapter four.

4) We recommend using a covered gear box to protect it from any environmental effect that can damage it or damage the wheel configuration, since we encounter such damage that broken the gear box and the motors with their drive circuits due to stall state the motor reached when the gear box was broken, this lead to current overflow in the electronic circuit then damaging it.

5) We also recommend using a comparator circuit besides the fuses to protect the drive circuit and all other electronic components from current overflow.

6.3 Future Work

Due to the fast development in technology field, and especially in the energy field, it would really be a fantastic idea to implement a solar panel on the surface of the robot instead of the solid cover it has, so that the robot will not need any charging operation if it has an electric system that can collect the solar energy and store it in accumulators onboard the robot.

Also since the robot has an onboard camera, it would be a brilliant idea to program an image processing system that would help the robot to identify its environment easily, meaning that it will have the assistance of image processing beside the sensors, making it much more easier to search for a new path that is free of obstacles.

And finally, since the robot has a PIC micro controller, and it executes multiple tasks in the same moment, such as moving to new coordinates while rotating the camera to check for the environment, this mechanism can be updated to a lifting and grabbing mechanism, that can be used to move parts or object from place to place according to the users need.

References

1. Philip Johan Mckerrow, Introduction to Robotics, Addison-Wesley Puplishing Company, Sydny 1995.
2. Joseph L.jones, Antia M.Flynn, Mobile Robots Spiration to Implementation, 2nd edition, A K Peters, Massachuseffs 1999.
3. karim A.Tahboub, Harry H. Asad, Dynamic Analysis and control of a Holonomic Vehicle With a Continuously Variable Transmission, journal of Dynamic Systems, Measurement, and Control, Transaction of the ASME, 118/vol.124, march 2002, Transaction of the ASME.
4. Jaser abdel-ghafar al-muhtaseb, iyad waheed al atrash, Mobile robot, graduation project in Palestine polytechnic university, September 2002.

5. Joseph Edward Shigley, Charles R.Mischkle, Mechanical Engineering Design, 5th edition, McGraw-Hill Book Company, New York.

6. Dirk Lefeber, Hichem Sahli, Autonomous Mobile Robot Mechanical Design, Vrije University Brussel, Academiejaar 2004-2005.

7. Datasheet, “Charge Terminal 3000”, www.conrad.com

8. http://en.wikipedia.org, Microcontroller.

9. Dogan Ibrahim, Advanced PIC Microcontroller Projects in C, Elsevier Publishing, March 2008.

10. http://en.wikipedia.org, PIC microcontroller.

11. John Iovine, A Beginner’s Guide to Robotics Projects Using the PICmicro, McGraw-Hill Publishing, 2004.

12. Richard C,dorf, Robert H.Bishop, Modern Control Systems, 8th edition, Addison-wesley, New York, 1998.

13. Vehicle Dynamics Text Book.

14. Richard C,dorf, Robert H.Bishop, Modern Control Systems, 9th edition, 2000.

15. Joseph Stiles Beggs (1983). *Kinematics*. Taylor & Francis

16. http://en.wikipedia.org, PID Controller.

17. http://en.wikipedia.org/wiki/State_feedback_controller.

18. Victor Villela, Robert Parkin, Marcelo Lopez parra, Jesus M.Dorador, WHEELED MOBILE ROBOT WITH OBSTACLE AVOIDANCE CAPABILITY, Sociedad Mexicana de Ingenieria Mecanica, 2004.

Part Two

Computer and Communication Part

Chapter One

Introduction

1.1 Overview
1.2 Motivation
1.3 Literature Review
1.4 Time Planning
1.5 Cost Estimation
1.6 Report Content

Chapter One

Introduction

1.1 Overview

Wireless networks have lately become a huge field for engineering projects, its applications are enormous. The implementations of wireless networks have increased over the last decade upon the new technologies discovered or developed everyday to meet up with the needs of wireless applications that gives the advantage of mobility, reliability, and efficiency compared to any similar applications with wired networks.

One of the fields of wireless communications is controlling systems. This is the main idea of this report, **Web-Based Mobile Robot Control System**. The System meets the needs of mobility, and performing control process from anywhere. The robot system includes a built on camera to monitor the surrounding territory. The video feed of the camera can be stored online, through a web page with member's access control.

1.2 Motivation

Mobile robots have become a wide area for development due to their effective applications like the ones that can reach places that are hard to reach by humans and other applications that needs inspecting, surveillance etc.

The robot is mobile, which means that it has no wires attached and no limitations on its movement compared to wired robots, so it can be used indoors and also outdoors.

Remote control of the system makes it much easier for the user to control the robot wherever it is. This system can be used as a mobile surveillance camera in buildings, banks, universities, etc. So the security staff can observe the surroundings from anywhere they are. Another application of this system is inspecting sewer systems which are places that hard to be reached by workers, the inspector of the sewers can inspect the whole system while setting on his comfortable office.

1.3 Literature Review

Many projects have been developed in systems control field, but few had used wireless technology. An example is the project entitled by "**Internet- Based Robot Control System Using Matlab.**"[1] This project was developed at PPU, Matlab software was applied as the programming language to perform remote control operations via the internet. The robot was connected directly to the web server through the parallel port and other hardware devices in order to perform the controlling operations. This project lacks the ability to control mobile robot wirelessly.

In this system LabVIEW program was chosen instead of Matlab that lacks speed, also due to the simple programming requirements using LabVIEW program, because most of the functions are implemented not by writing a code, but building the algorithm using graphical blocks acting as the code. Additionally, using PIC microcontrollers instead of video acquisition card, due to the programmable characteristics of the PIC's. Further, a wireless transceiver was implemented in this system instead of the parallel port, because the robot is mobile. Beside all that, the robot in this system has the ability to avoid obstacles by an implemented intelligent system.

Another project in the robots controlling field is the "**Security Guard Robot (SG-Robot)"** [2] which was developed by a group of departments in a number of Korean universities. This project operates using CDMA networking. The controlling is done through cellular phone software with video feedback from the SG-Robot. An ARM microcontroller was used in this project. A PXA255 controller was used for image processing. The controllers exchange commands through an ad-hoc LAN network.

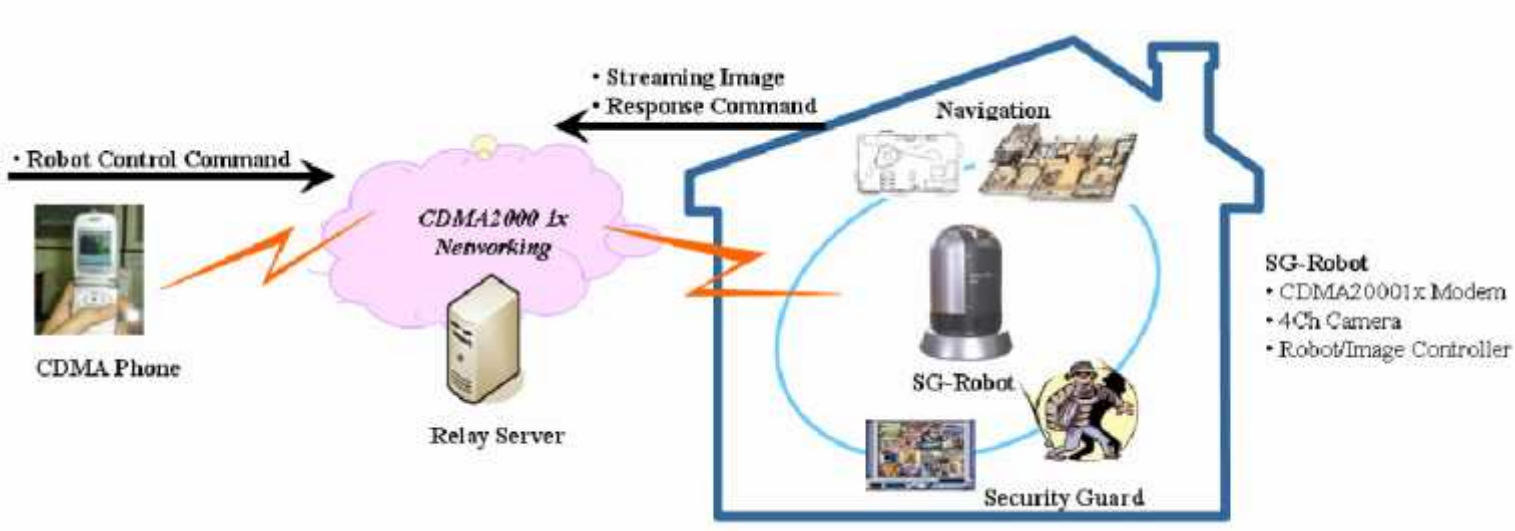


Figure 1.1 Configuration of SG-Robot using CDMA networking[2].

There are number of differences between **Web-Based Mobile Robot Control System** project, which was developed throughout this report and the **SG-Robot program** project that applies the CDMA infrastructure which is incompatible to use for such projects in Palestine, rather GSM is the operating network for cellular phones which has a limited bandwidth. In addition to that, applying GSM-based controller would cost much more money compared to a web-based controller due the charge per second system in the GSM network, while the internet network costs per month. Another difference is using the PIC microcontroller instead of using the ARM microcontroller and PXA255 controller. PIC's systems are much simpler than dealing with two controllers.

1.4 Time Planning

This section summarizes the expected timing plan for this project.

1.4.1 Time Schedule

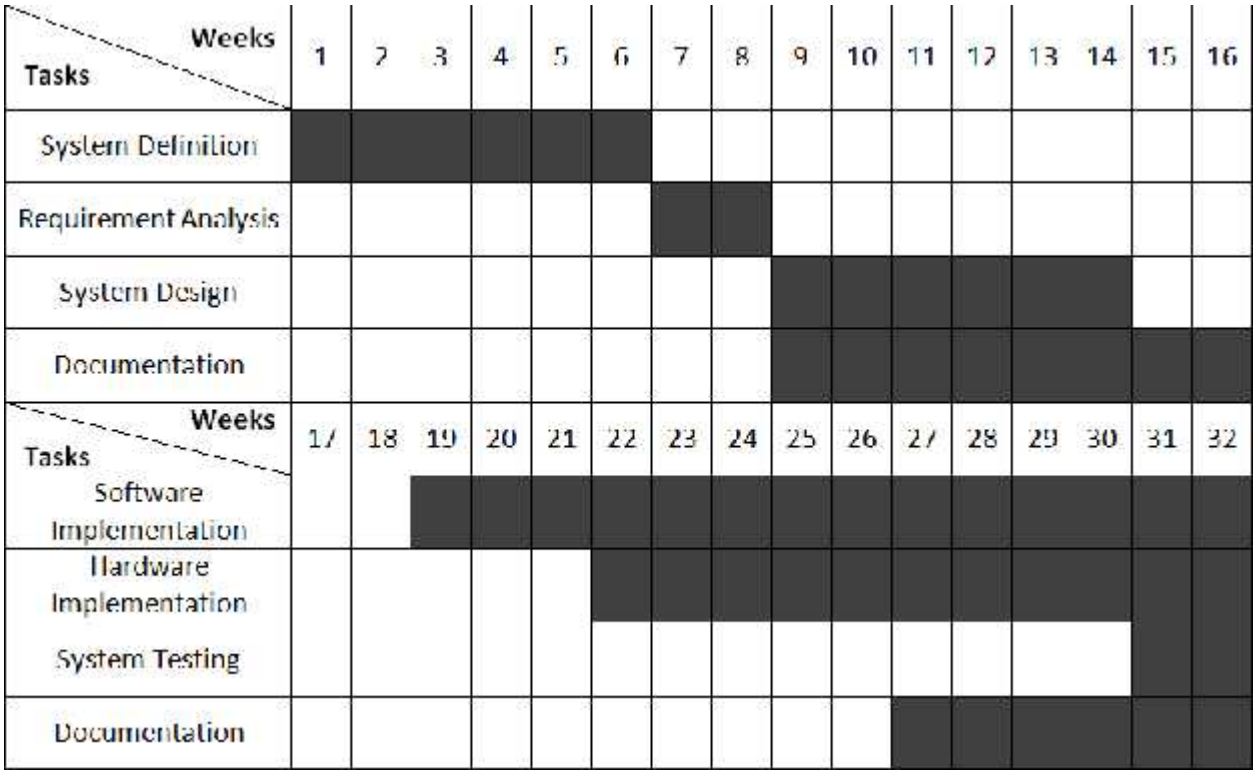
The following table explains the expected timing plan.

Table 1.1 Project Timing Plan Table.

System Definition	6 Weeks
Requirement Analysis	2 Weeks
System Design	6 Weeks
System Implementation	13 Weeks
Testing and Modification	2 Weeks
Documentation	16 Weeks
Total	**32 Weeks**

1.4.2 Schedule Table

Table 1.2 Schedule Table.



1.5 Cost Estimation

In this section, estimated hardware and software costs are presented.

1.5.1 Hardware Cost

The following table shows estimated hardware costs.

Table 1.3 hardware costs.

1	PIC Microcontroller	15$ x2
2	Video Camera	100$ x1
3	WLAN Transceiver + Antenna	150$ x1
4	DC Motor	20$ x2
5	Stepper Motor	10$ x1
6	Battery	20$ x1
7	Proximity Sensor	45$ x8
8	Encoders	130$ x2
	Total	**870$**

1.5.2 Software Cost

The following table shows estimated software costs.

Table 1.4 Software Cost.

1	LabVIEW 7.1	$100
2	Microchip MPLAB	120$
3	XAMPP Web Server	40$
4	Hosted Server Online with 2GB Capacity and 1MB Traffic	25$ Annually
	Total	**258$**

1.5.3 Human Resources Costs

The following table shows the estimated human resources costs.

Table 1.5 Human Resources Costs

1	105 Days x 50$	5250$ x 4 Persons
	Total	**21000$**

1.6 Report Contents.

This report consists of a number of chapters. Each discusses a subject related to the project.

Chapter one discusses the idea, the motivation, the time scheduling, and the requirements of the project, to give readers general view about the project.

Chapter two discusses the theoretical background. It starts with general information about the system, and then discusses the important aspects of the system including PIC microcontroller, wireless transceivers, video camera and sensors.

Chapter three presents the general system design concepts. It includes system objectives, general system block diagram, system flow control, system and brief description of system operation.

Hardware design implementation is discussed in chapter four, it includes a general description for the system interfacing, detailed interfacing techniques and requirements to connect the system components together, a division between both the mechanical part and the electrical part is also discussed.

Chapter five discussed the software design implementation. It contains a detailed block diagrams to show how the software requirements in this system were designed and implemented to perform the required function of the robot.

Chapter six presented the testing operation of the system. Testing time schedule, testing procedure and testing strategies presented through black box testing and white box testing.

Chapter seven suggests future work that could be added to this system.

Chapter Two

Theoretical Background

2.1 Introduction

2.2 Mobile Robots

2.3 Microcontroller

2.4 WLAN Transceivers

2.5 Sensors

2.6 Serial Video Cameras

Chapter Two

Theoretical Background

2.1 Introduction

As stated in chapter one, the system under development is a mobile robot that can be controlled via the Internet. It consists of two main parts. The first is the robot system, and the second is the LabVIEW program that controls the robot.

The system depends on an online web server that is configured by installing the necessary web server software. The web server runs software that connects the user to the robot and performs some processing on the control operations that come from the user and send them to the robot. The robot is connected to the internet through a wireless transceiver module, and contains other hardware devices necessary for the controlling operations.

The controlling process will give the system's user the ability to control the robot according to his needs. The system will give the user the option to record the video feed from the camera on the robot, and store it online. This camera acquires the video from the surrounding environment and sends it to the PIC microcontroller, which in its part transfer it to the web server through the wireless transceiver. According to the video, the user can modify the commands and send them to the web server for processing, and then the web server sends them to the robot.

2.2 Mobile Robots

In recent years, the development of computer and reduction in size and cost of integrated circuits led to create intelligent systems like mobile robots, which has a lot of applications and uses, one of them is surveillance and another one is exploring of some environments that humans can't reach.

Mobile robot is "an automatic machine that is capable of movement in a given environment ". Mobile robots could be found in many aspects of life and industrial applications, especially in tasks that are dangerous for humans, since machines are less sensitive than people to radiation and toxic encountered during some tasks, such as repairing nuclear plant equipment and exploring or working in dangerous environment. In addition to this, mobile robots could be found in repetitive tasks, such as material handling systems, and storing systems. Add to this the scientific researches and entertainment purposes.

There are many types of mobile robot control [3]:

1. Manual remote or tele-op

This robot is controlled by a driver using a joystick or other control device which could be plugged directly to the robot, a wireless joystick or an accessory to a wireless computer. As the idea of the **Web-Based Mobile Robot Control System,** the tele-op is used to help the operator to stay out from the harm way and to reach places that is hard to be reached by humans.

2. Guarded tele-op:

The **Guarded tele-op** robot can sense any obstacle in its way and avoid it, also it could be a driver.

3. Line-following robot

This robot is an early Automated Guided Vehicles (AGVs). It has the ability to follow a line painted or embedded on the ground or even an electrical wire in the floor. Most of these robots use a "keep the line in the center sensor" algorithm to do this. If any obstacle faced them, they can't avoid it and move around it as **Guarded tele-op**, they just stop and wait until this obstacle is removed.

4. Autonomously randomized robot

Such kind of robots can bounce off walls. The walls can be sensed by electronic sensors or by physical bumpers. The algorithm used in these robots is a simple "bump and turn 30 degrees", which enabled the robot to cover most of or all of the floor or the surface to be covered.

5. Autonomously guided robot

This kind of robots knows some information about it is position and its waypoint to reach various goals. "Localization" or knowledge of its current location, is calculated using sensors such as motor encoders, vision, lasers and global positioning systems. Positioning systems often determine the location and orientation of the platform from which it can plan a path to its next waypoint or goal using triangulation, relative position and/or Monte-Carlo/Markov localization. It can gather sensor readings that are time- and location-stamped.

6. Sliding autonomy

These robots combine multiple levels of navigation under a system called sliding autonomy. Most autonomously guided robots, such as the Help Mate hospital robot, also offer a manual mode. The MOBILE ROBOT Sinside guidance system, which is used in the ADAM, Patrol Bot, Speci -Minder, MapperBot and a number of other robots, offers full sliding autonomy, from manual to guarded to autonomous modes.

2.3 Microcontrollers

A microcontroller (also microcontroller unit, MCU or µC) is "a small computer on a single integrated circuit consisting of a relatively simple CPU combined with support functions such as a crystal oscillator, timers, watchdog timer, serial and analog I/O etc. Program memory in the form of EEPROM (Electrically Erasable Programmable Read Only Memory) or ROM (Read Only Memory) is also often included on chip, as well as a typically small amount of RAM (Random Access Memory)."[4]

Microcontrollers have long been a convenient interface for embedded systems, they represent the core of the control system for electronic devices in dedicated applications. Thus, in contrast the microprocessors that are used in general purpose applications like personal computers that need high-performance, and multitasking. Microcontrollers contain data and program memory, serial and parallel I/O, timers, external and internal interrupts [4], and peripherals. These make them a strong choice when implementing control systems.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, remote controls, office machines, appliances, power tools, and toys. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems. [5].

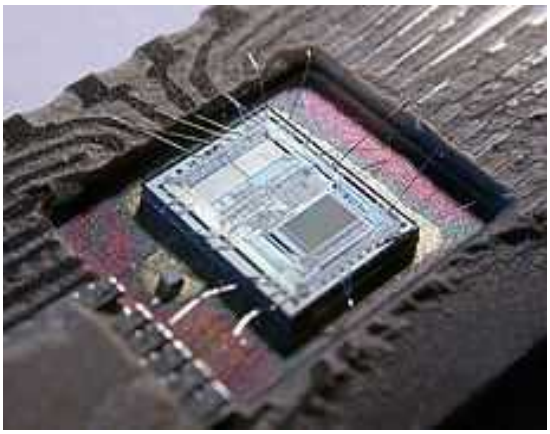


Figure 2.1 Intel 8-bit microcontroller

Microcontrollers exist in many types and forms, for different applications and costs. And here is a list of some microcontrollers' types that are known in the market:

1. MIPS (32-bit PIC32)
2. ARM processors
3. 8051
4. PIC (8-bit PIC16, PIC18, 16-bit dsPIC33 / PIC24)

In this system the PIC microcontrollers are used.

The PIC microcontrollers are a family from Harvard architecture microcontrollers that is manufactured by Microchip Technology. The name PIC at the beginnings stood for "Programmable Interface Controller" an shortly after was replaced with the name "Programmable Intelligent Computers".

PICs are popular with both industrial developers and hobbyists alike due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability.[6]

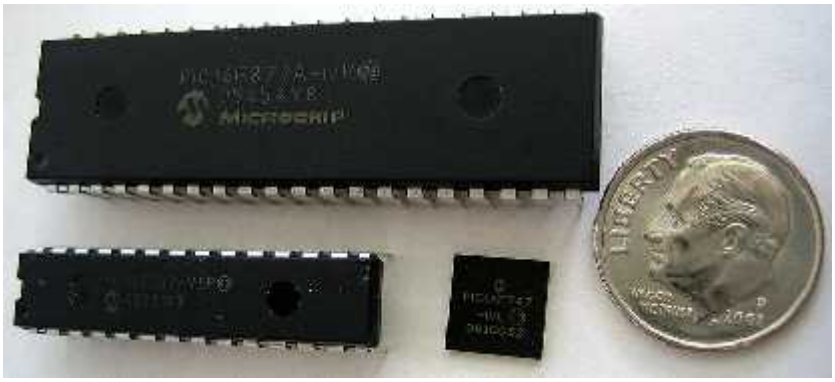


Figure 2.2 PIC Microcontrollers

The PIC chips (or PICmicro chips) are as stated above, programmable chips, programmed to perform a special operations for dedicated applications in embedded systems, they provide strong interfacing abilities with many peripherals and other microcontrollers in other embedded systems.

Programming PIC microcontrollers is a simple three steps process, write the code, compile the code, and upload the code into a microcontroller. Writing the code can be developed in many Integrated Development Environments (IDE's) for example, MPLAP IDE, which is a software developed for the Microchip appliances like the PIC microcontrollers. Compiling the code can be done by the compiler of the MPLAP IDE. There are different compilers associated to work with PIC chips, C compiler, or assembler for assembly language codes, and many more. The decision of which compiler to use, is a developer choice, depending on the application which the PIC is a part of. The final step of programming the PIC chip is uploading the code into the microcontroller. This can be done also in MPLAP IDE or in different programs that are connected to the PIC kit (figure 2.3). The uploading process can be done through a USB cable or other connecting technique depending on the kit that contains the PIC chip.

Figure 2.3 PICDEM™ FS-USB Evaluation Kit for PIC18F4550

The PIC microcontroller architecture makes interfacing most peripherals with the PIC a far from hard task, the I/O's are organized as ports, PORTA, PORTB, etc. each port can be treated as a unit or as single I/O pins, 8-bit, 5-bit or other organization. Each port was initially configured to

do a specific operation, serial data operations, Analog-to-Digital conversion, and many more, but it's not always necessary to stick with the initial configuration, each port or single I/O pin can be configured to do different operation from what initially configured.

In this project the PIC18F4550 is used. This is due to its availability, cheap cost, easy programming (75 Instructions single-word instruction), and the most important factor, suitability to be used in robotic applications as in this system. The PIC18F4550 has all what is needed for the implementation of this project, enough I/O Ports, ADC, timers and counters, and variable operating frequency up to 48MHz.

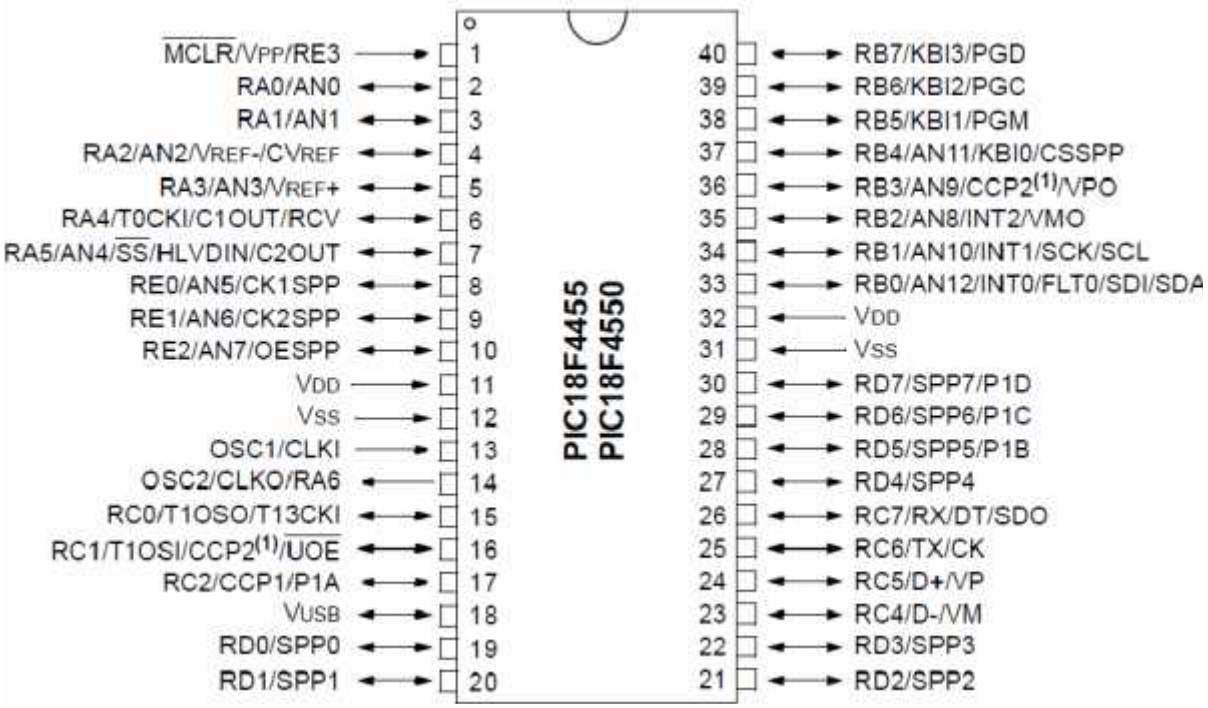


Figure 2.4 PIC18F4550 Pin Diagram

As shown in figure 2.4, the PIC18F4550 provides 33 single I/O pins, divided into 5 ports, PORTA 6 pins, PORTB, PORTC, and PORTD 8 bits, and finally PORTE 3 pins.

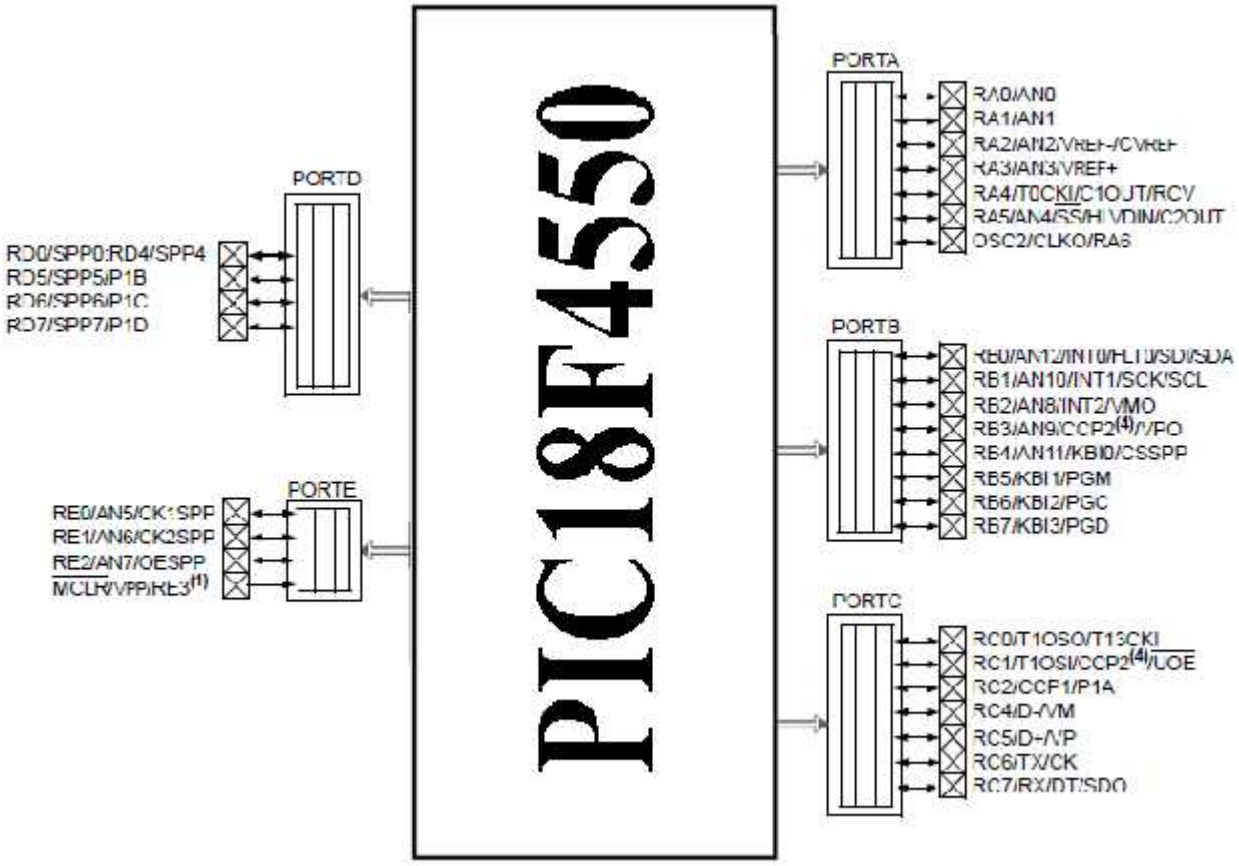


Figure 2.5 PIC18F4550 I/O Organization

The PIC18F4550 as shown in figure 2.6, contains a built on Analog-to-Digital converter (ADC) with 10-bit resolution, up to 13 channels, and three timers. Figure 2.5 shows the interfacing of the ADC with the 13 input analog channels. The ADC is used to connect the 8 ultrasonic sensors which give analog readings for the distance of the obstacles in the range of the sensors.

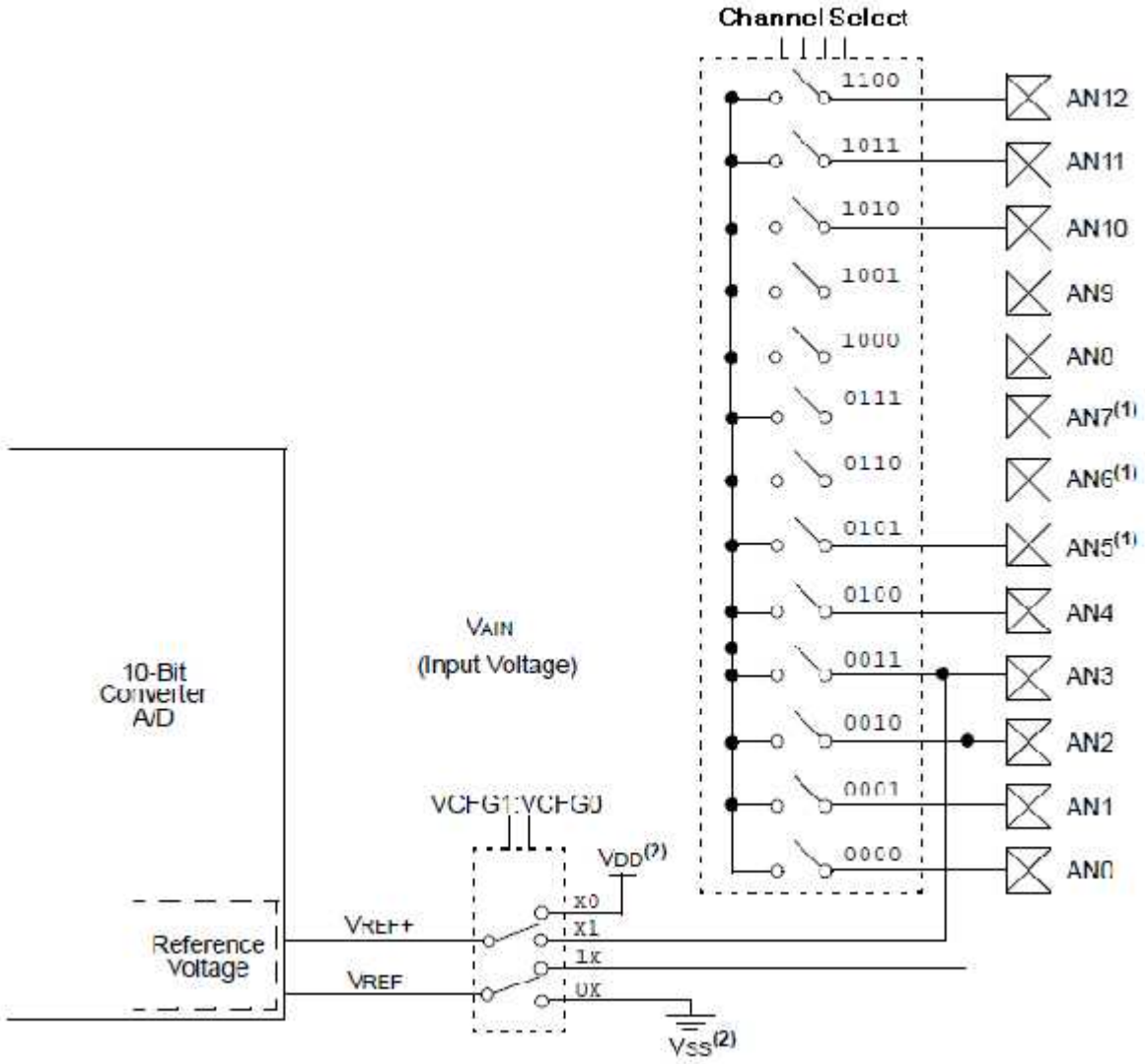


Figure 2.6 PIC18F4550 ADC Inputs

2.4 Wireless Transceivers

The wireless transceiver is a device that contains both transmitter and receiver which are combined in a common circuitry, it transmits and receives data wirelessly. In this project, the transceiver connects the PIC microcontroller with the internet through an access point. It receives the video feed from the PIC, modulates it and sends the modulated signal to the access point. Also, the transceiver receives the commands coming from the access point, demodulates them and passes them to the PIC.

After researching, it was found that the **WiFly GSX** transceiver is the most suitable wireless transceiver available in the markets that could be used in this project.

- **Why WiFly GSX**

The **WiFly GSX** was chosen to be used in this project due to a number of factors, it has an extremely small size and light weight, which make it more appropriate to be implemented on

the robot without slowing down the robot movement. Also its low cost and availability in the market, and the most important factor is its suitability for this project, its characteristics meet the project objectives.

One of the important factors of choosing the wireless transceiver was the consideration whether the transceiver has the ability to associate with an access point. This is because the system depends on the access point to receive the commands, and to transmit the video feed, this transceiver can associate with an 802.11g\b access points. It can be also connected to PIC microcontroller, and support serial communication between them. Further interfacing details will be discussed in chapter 4.

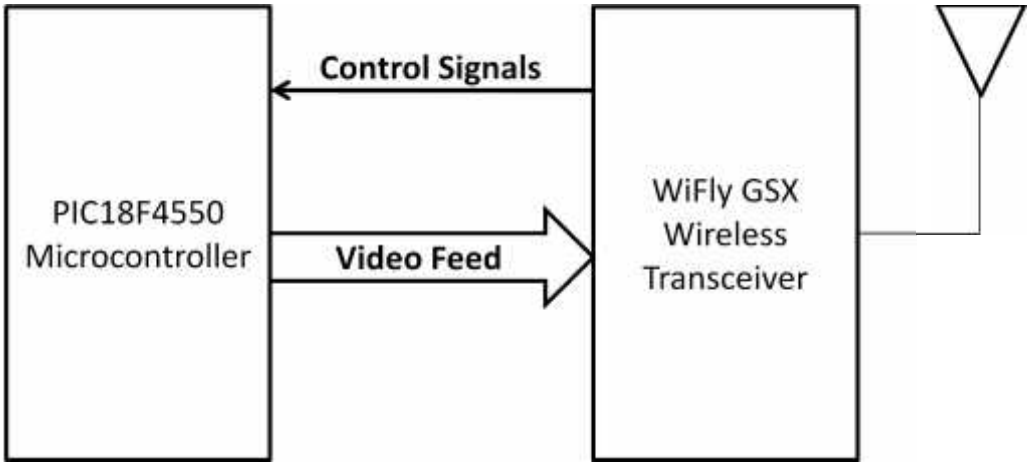


Figure 2.7 WiFly GSX with host microcontroller

It is very important to think about the power consumption for the system. As described earlier, this system is a mobile system, there are no wires attached to it, so the power consumption has to be limited as much as possible. The **WiFly GSX** transceiver was considered as a breakthrough in the wireless communication, because it has the least power consumption between the wireless transceivers, it has three modes, sleep mode of 4µA, standby of 15mA and active mode 40-212 mA. With this little power consumption the **WiFly GSX** can live on the regular AA batteries for many years.

The **WiFly GSX** wireless transceiver is a programmed module, it has a specific commands to programmed with. The system designer can communicate with the transceiver through the CMD on any computer with windows on it, or a software can be downloaded to the computer, such as the **TeraTerm** software. This software creates a connection between the computer and the transceiver wirelessly, and after the connection is completed, the transceiver will be able to receive from the developer. Or it can be connected to a PIC microcontroller and receives ASCII code form it.

Figure 2.8 WiFly GSX RN-131G

As shown on figure 2.9, the WIFLY transceiver has 44 pins. One of these pins is left unconnected (PIN 35), some of them are connected to the ground (PIN 19) and (PIN 36-44) these pins must be connected for proper antenna performance. Some pins are used for a sensor interface, with analog input.

There are two sources for the input power to the transceiver, it could be from a battery (2-3.3 V) which could be connected into (VDD-BATT PIN20), or a (3.3–3.7 V) voltage source which could be connected into (VDD-IN PIN 21).

There is a dedicated pin for forcing the transceiver to wake up from the sleep mode which is (FORCE_AWAKE PIN 9).

The pins that are used to connect the transceiver with the PIC microcontroller are the UART-RX and the UART-TX (PINs 12 & 13). The UART-TX sends data from the module to the PIC, and vice versa for UART-RX.

This transceiver has an on-board ceramic antenna that operates at the 2.4GHz for transmission and reception, and other U.FL connector for optional external antenna.

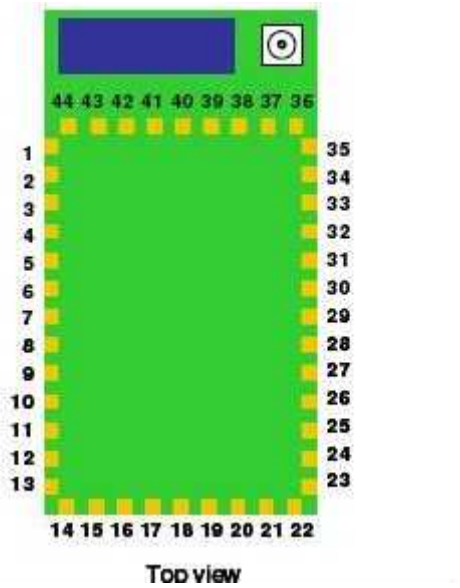


Figure 2.9 WiFly GSX Pin Diagram

2.5 Sensors

Sensors are devises used to transform a physical state to a form of data that can be understood by electrical devices, circuits, etc. There are many types of sensors, temperature, humidity, chemical, and many more. In this system ultrasonic sensors are used.

Ultrasonic sensors can be used to detect the presence of something in the range of the sensor and the distance between the sensor and that body, sometimes called range finders. There are many types of these sensors, the type implemented in this system is the LV-MaxSonar-EZ0 (see figure 2.10).

Figure 2.10 LV-MaxSonar-EZ0 Range Finder.

The LV-MaxSonar-EZ0 sensor operates at range of voltage between 2.5V to 5.5V provide ranging up to 6.45 meter (255 inch) with a controlled beam width. This sensor output ranges in three manners, as a Pulse Width Modulation (PWM) representing the range with scaling factor of 147uS per inch. Another output as a serial data represented in ASCII code. The third output as an analog voltage with sensitivity depending on the voltage source, for example with a supply of 5V yields ~9.8mV/in. and 3.3V yields ~6.4mV/in. The analog output is the type

used in this system, due to the simplicity of reading the range in analog signal using the PIC microcontroller that provides up to 13 analog inputs, and the most important reason, the serial module in the PIC is connected to the WiFly transceiver and the Video Camera, while the PWM module is connected to the DC motors. More about interfacing the LV-MaxSonar-EZ0 with the PIC microcontroller is described in chapter 4.

2.6 Serial Video Cameras

In this system there are many hardware components, connected to the PIC microcontroller, so, the intention was to minimize the wires and the complexity of interfacing. Serial communication was the much suitable to implement in this system, much less wires than any equivalent hardware in parallel communication, a single wire to transfer data and another to receive.

The video camera is needed in this system to capture the surrounding, and transfer video data to the PIC microcontroller through serial connection between the USART in the PIC, and the UART in the video camera.

In this system the ITM-C-328 serial camera was chosen, it performs as a JPEG compressed still camera, or video camera. It was chosen due to its cheap price, small size, low power consumption, programmable, and suitable to the project requirement.

As mentioned above, the ITM-C-328 is a programmable camera, the user can choose the baud rate, the frame size, shooting mode (video or still, compressed or uncompressed), and many more configurations.

The Transfer rate is 115.2Kbps for the JPEG still pictures, and 160X128 preview at 8bpp with frame rate from 0.75 to 6 fps.

Chapter Three

System Design

3.1 Project Objectives.

3.2 General Block Diagram.

3.3 System operation.

3.4 System Flow Control.

3.5 System Operating Requirements.

3.6 Delay Calculations.

3.7 System Security.

Chapter Three

System Design

This chapter discusses the design concept of the system. It describes the project objectives, the system block diagram as well as presenting system entities.

3.1 Project Objectives

Project objectives can be summarized in these points:

- To provide a mobile robot that can move freely with no wired attached.
- To provide a robot control system that performs a set of motion actions, for both the robot and the camera base.
- To provide an option of controlling the robot in both ad-hoc network as well as infrastructure network.
- To provide suitable and easy web application for robot control.
- Enable the user to configure the robot wireless module over a TCP connection.
- To give the user the option of recording the video feed through the controller program on the web page, and storing it on the client.
- To give the robot the ability to associate with the best access point that has the strongest signal power.
- To provide the robot with an intelligent system to protect it from obstacles for safe movement.
- To ensure a respective level of security in the system.

3.2 General Block Diagram

In this section, a general system block diagram is presented. The general block diagram is shown in figure 3.1:

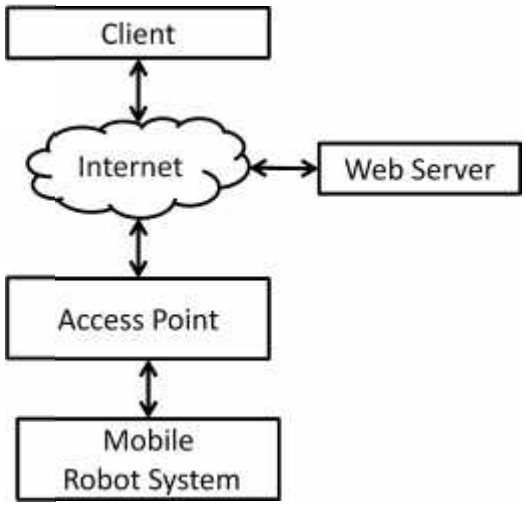


Figure 3.1 System General Block Diagram.

3.3 System Operation

The system user logs in with his account on the web page to gain access to the robot controller software. When he does so, the video feedback is transmitted from the carried camera on the mobile robot to the controller software. The user sends the control command from the controller software to control the movement of both the robot and the camera base. The robot has an intelligent system implemented to avoid obstacles.

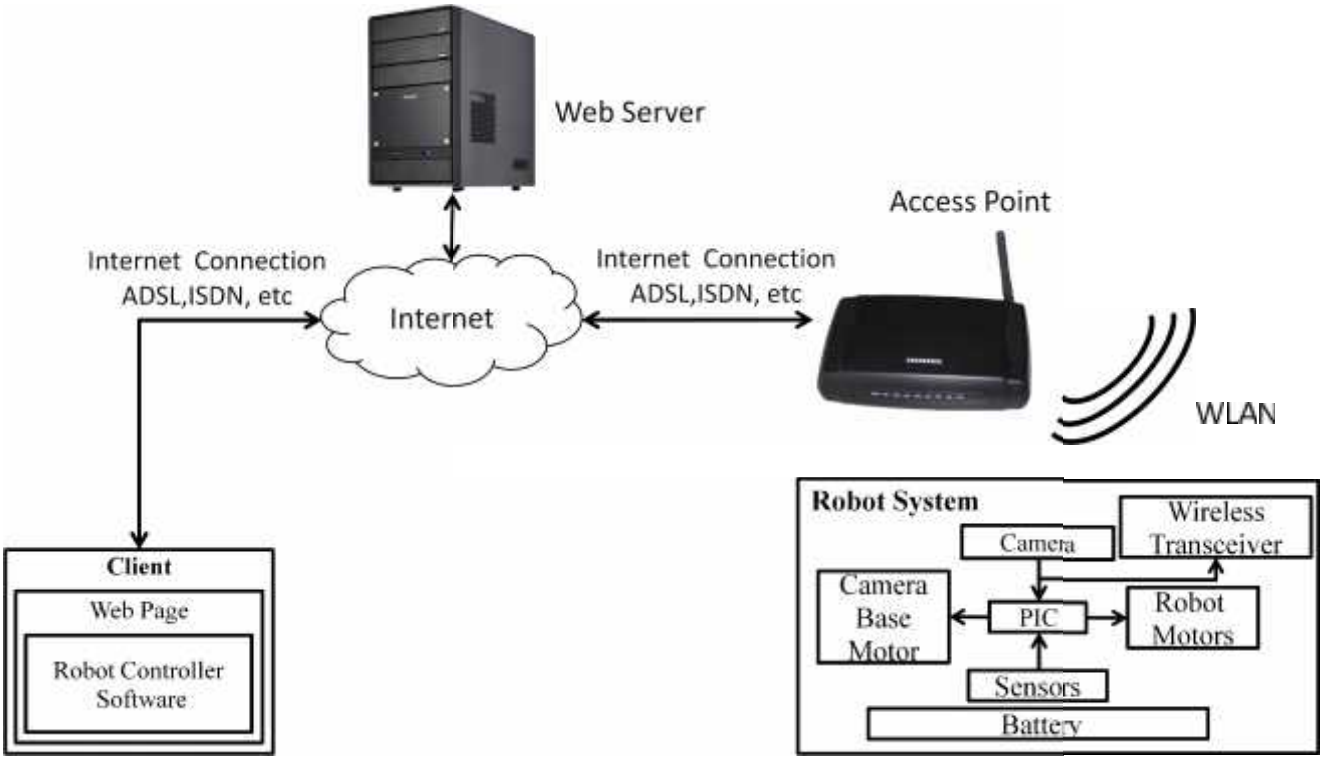


Figure 3.2 System Operation.

Figure 3.2 shows the main system components. In this section, a detailed description of each component is included.

In order to understand each component, the components are specified into two categories, software components, and hardware components.

3.3.1 Software Components

The components that have a software form, code or graphical blocks or applications or any other form of programming, these components are very important because they program the hardware to function as required.

1. Client

The client is categorized as a software component, because the user uses the applications to enter and use the system.

The client is the remote controlling side of the system. Any device in the world with an internet access and a web browser and has the authorization to control the system can be called a client, whether it's a computer, PDA, cell phone, or any other device.

As shown in figure 3.3, the client runs the robot control software. This software is a part of a web page, the system user logs in with a username and password to gain access to the software as shown in figure 3.3. When the software is running the user can perform the control operations on the robot and the camera base. The video feed can be recorded optionally.

The system users can control the system anywhere they are with no geographical limitations, which means that the user and the robot can be distant from each other.

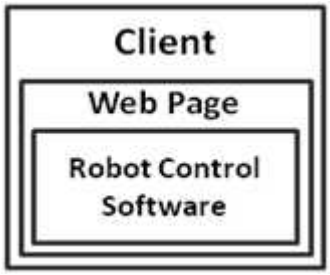


Figure 3.3 Client

2. Internet

To access the web page that runs the control software, the user obviously must connect to the internet, and then use the software to perform control operations. As the internet bandwidth increased the control commands and the feedback from the robot will be faster and more effective for the user, also the video feed will be in a better quality.

The functionality of the internet in this project is connecting the client to the robot system, as shown in figure 3.4, through the internet the controller software connects to the mobile robot through the home network's access point.

Figure 3.4 Internet Connects Client to the Robot System

3. Web Server

The dedicated web server is hosted on a domain online to have a better bandwidth, and enough capacity to store the videos online for the user to check them out whenever he desires to.

As shown in figure 3.5, the web server consists of three main parts. The first one is the server program it should support PHP pages. For that reason XAMPP server (Apache Server)

software is installed to open the web pages. The second part is LabVIEW software which is needed to run the third part which is the robot control software that is described below. Additionally a storage space is needed to store the client's recorded video feeds from the robot's camera. Also to store the client's information and accounts side by side with the web site.

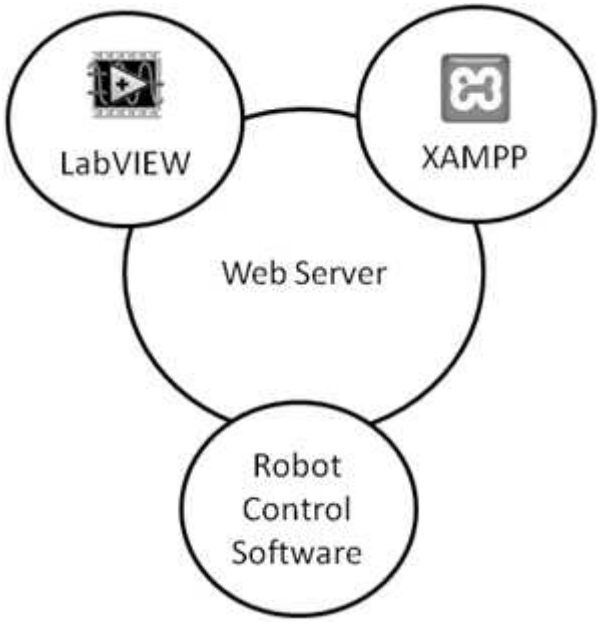


Figure 3.5 Web Server

The web site that the client uses to log into the robot control software is hosted on a web server. The web site was developed using Adobe Dreamweaver CS4, written in PHP language. This was due to the strength and simplicity of the PHP scripts when dealing with databases needed for the web site. The databases used in the system will be developed using PHPMyAdmin that is in the XAMPP server installation package. This system deals with databases for retrieving and storing the system users' accounts, and storing or watching recorded videos from the robot's camera.

4. Robot Control Software

Whenever the robot is powered up it will automatically try to connect to the web server through the IP address of the web server and the connection is done through a port which number is determined. The web server address and the port number is programmed in the WiFly transceiver which is built on the robot, this allocates the robot's location, in which network it exists.

When the web page that contains the software is opened by the system user, the user deals directly with the user interface that has the options of moving the robot and observing the video, etc.

The robot control software maps every command the user makes from a LabVIEW event to an ASCII characters command so that the PIC microcontroller can later decode, then the command is sent to the mobile robot through a TCP connection.

The robot controller software has two options regarding the connection with the robot:

1. Automatic Connection: when the robot is powered up it will automatically try to connect with the web server. This option requires that the robot control software is always up and dunning so the robot can connect to whenever it is powered up.
2. Manual Connection: the user can connect directly to the robot using a known IP address and a port number of the robot's WiFly transceiver. This option is helpful in debugging the transceiver, if a system developer trying to connect to the robot without the web server, the developer must have the robot controller software installed in his device.

   Navigating the robot is done through sending coordination of the required destination, or sending the distance and the direction of the required movement.

   The robot control software can also be used to program the wireless transceiver over a TCP connection.

   The robot controller software is developed using LabVIEW program. The decision to use LabVIEW was made due to its strong functionality when dealing with TCP functions (TCP open connection, TCP Read, TCP Write, etc), which was helpful in the implementation of this project, beside that, LabVIEW provides a simple web server capabilities to any application that is built using the software.

3.3.2 Hardware Components

Physical components that is programmed or interfaced with each other to provide the system functionality.

1. Wireless Access Point

As shown in figure 3.6, the wireless access point is essential for the system operation. The robot searches for a wireless access point to connect to through the robot's wireless transceiver. When the best access point (strongest signal) is found, the transceiver try to authenticate with it, if it succeeded, association occur to make the robot ready for controlling.

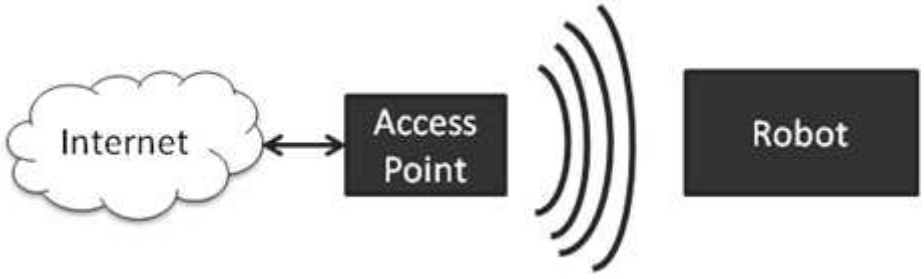


Figure 3.6 Wireless Access point Rule

2. Mobile Robot System

The mobile robot is the main part of this project, it contains the parts that helps the user doing the control operations, motors, camera, wireless transceiver, ultrasonic sensors, and a rechargeable battery. In addition to all that, the main part of the controlling system, the PIC microcontroller.

- **Motors**

The robot contains three motors, two DC motors for driving the robot (figure 3.7), and one stepper motor to turn the camera base around as shown in figure (3.8).

Figure 3.7 Driving DC Motors.

Figure 3.8 Stepper Motor.

- **Video Camera**

A visual feedback for the systems user is required for the controlling operation. So, a video feed is needed from the robot to the client to help the user move around as if he was in the area. The video camera used in this system is the ITM-C-328 Serial Camera which has a built on compression engine.

Figure 3.9 ITM-C-328 Serial Camera.

The main advantage of the ITM-C-328 is that it is serial, which leads to easy implementation and programming, also, its small size makes it easy to be built on the stepper motor so that it can be rotated 360 degree.

The video camera was programmed to operate at a baud rate of 38.4kbps in the serial connection with the PIC microcontroller.

- **WLAN Transceiver**

The robot is mobile, no wires attached to it, the power source is portable, so connecting it to the internet is an application for wireless LAN (WLAN). This is done using the wireless transceiver.

The wireless transceiver is connected to the PIC microcontroller, the transceiver is responsible for searching for networks, choosing the network with strongest signal power, sending and receiving both control signals and data from and to the robot, forward them to the PIC microcontroller or forward them from PIC to the access point which sends them to the web server.

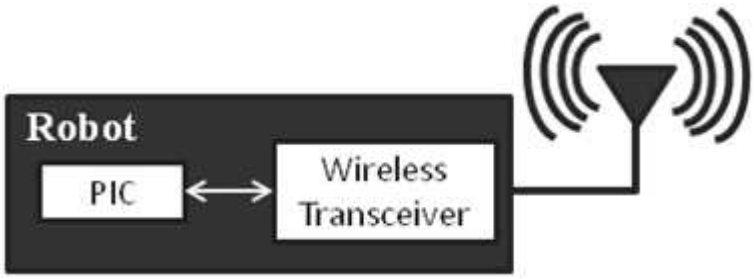


Figure 3.10 Wireless Transceiver Rule

The WLAN transceiver used in this system is the WiFly GSX. It was programmed to operate at a baud rate of 38.4kbps in the serial connection with the PIC microcontroller.

- **Encoders**

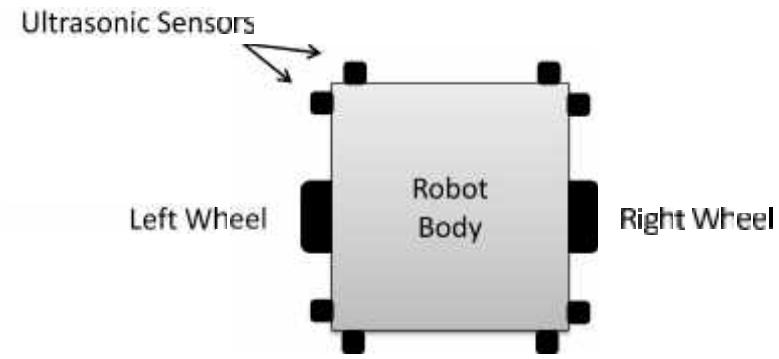
Its very important for the robot movement to be calculated every step, the direction its going and how many steps it has moved, this is done using two encoders one at each motor, they are both connected to the PIC microcontroller as an input, the PIC receives a frequency signal from the encoders and calculate the number of times this signal became high (pulse), every time the signal became high it means that the encoder detected a step from the motor. The relationship between number of pulses and the steps the motor moved is not fixed and is related to the type of encoder used and the gear ratio of the gears the motor is connected to.

Figure 3.11 CE Rotary Encoder.

As shown in figure 3.11, the encoders used in this system are CE rotary encoders.

- **Sensors**

There are some measurements done by the PIC microcontroller, distance from objects to avoid crashing. For this reason, eight sensors are needed, two at each side of the robot, figure 3.12. This is needed to check if there is a close object to insure safety movement for the robot, and to help implementing the intelligent system.



3.12 Sensors Distribution.

- **Battery**

The Robot has many different components, and each component may need a different voltage value, meaning that the robot must have different voltage values onboard. So the robot carries a 'sealed lead' 24v-5Ah and a power rate of (80Wh)

Figure 3.13 24V-5A Battery.

- **PIC Microcontroller**

The whole robot is built around the PIC microcontrollers. The PICs connect all the robots' components together, receives the instruction signals from the wireless transceiver, and sends the video feed back to it.

The PIC microcontrollers used in this system are both PIC18F4550, programmed using MPLAP software.

Since the internal oscillator of the PIC microcontroller has an 8MHz frequency, the baud rate cannot exceed 38.4kbps at which it was programmed.

Figure 3.14 shows the components of the mobile robot system, the PIC microcontroller connects them all together.

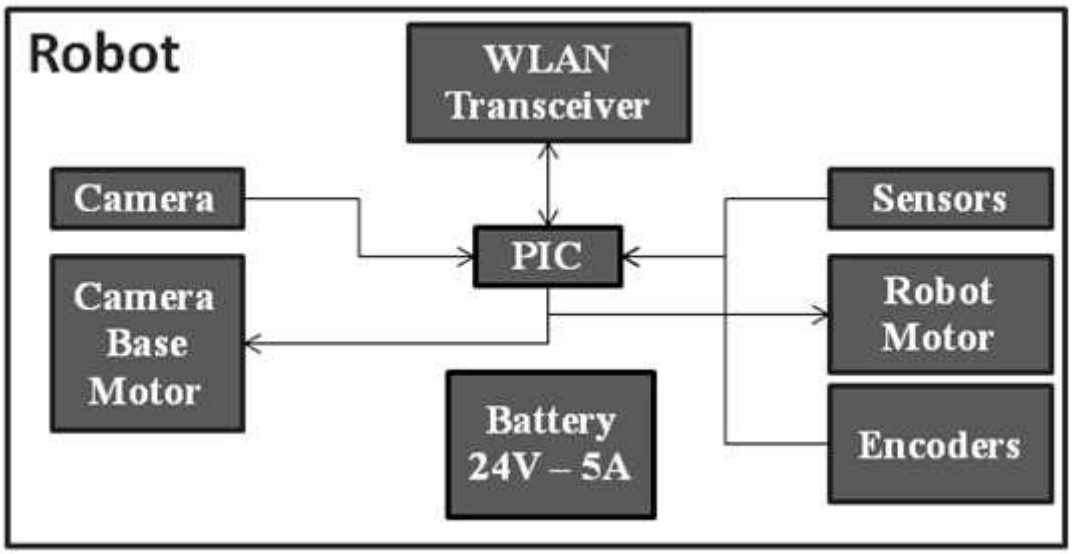


Figure 3.14 Mobile Robot System.

3.4 System Flow Control

Figure 3.15 shows the system flow control, it helps to understand and analyze the steps and the problems might appear and needs to be solved, in order to make the system as flawless as possible.

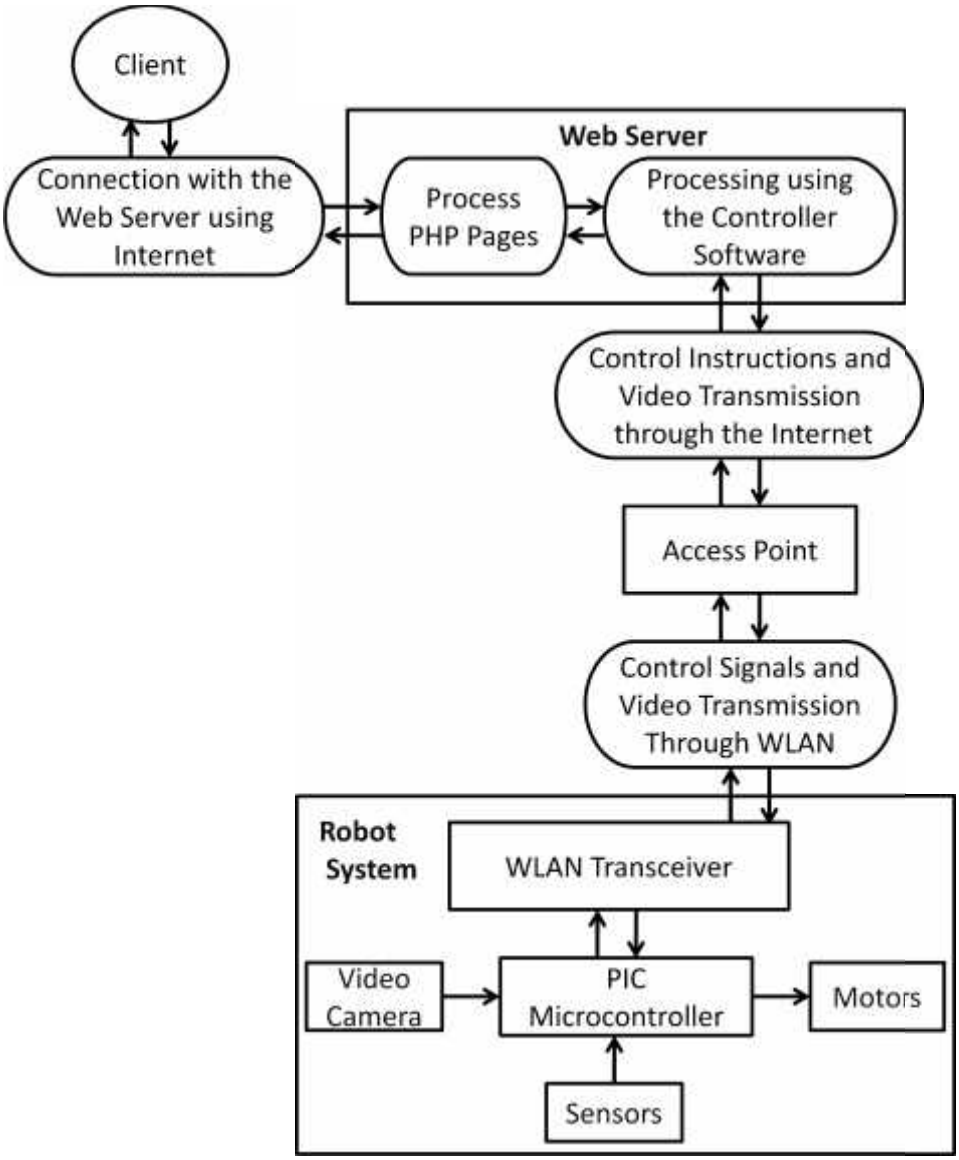


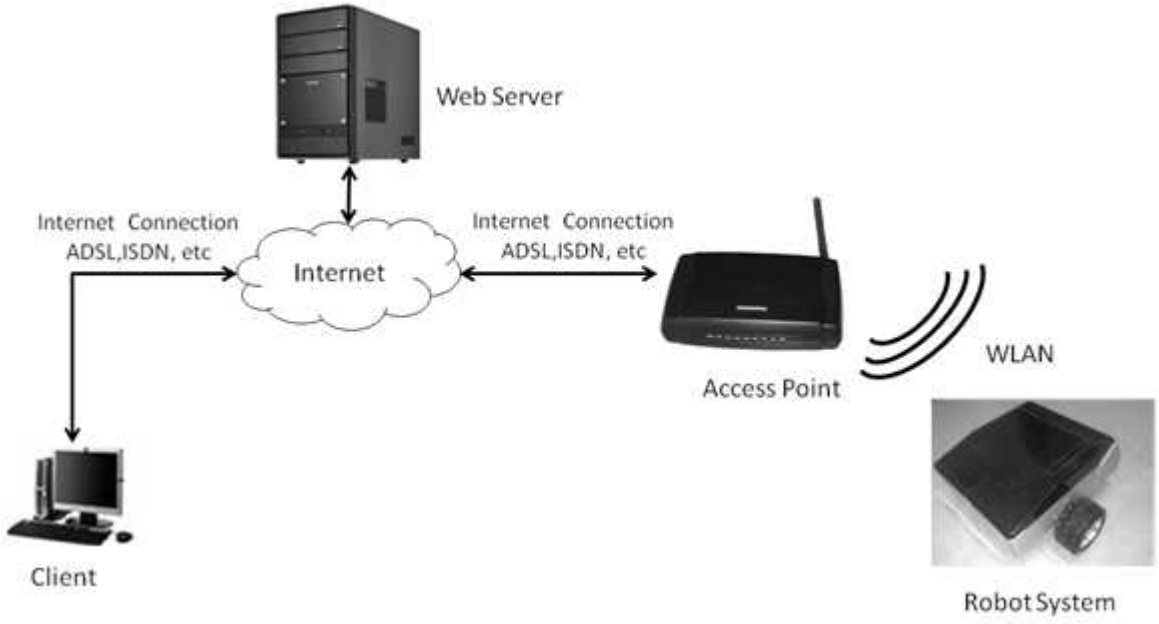
Figure 3.15 System Flow Control Diagram

3.5 System Operating Requirements

In order to guarantee an efficient operation for this system, a number of requirements must be available. Some are essential for the system to work properly, and others to increase the systems performance.

Some of the systems components represent the core of systems operation, without them, the system can't run at all. Each one of these components must meet up with certain requirements in order to operate the system without any delay or any unanticipated problems, this system can be operated in an infrastructure network or adhoc network.

3.5.1 Infrastructure Networks



3.16 System Operating in Infrastructure Network

To operate the system in an infrastructure network, figure 3.16, which is any network with access point, the existence of an internet network running on the IEEE standards is essential, in both the client side, and the robot side. In the client side it is essential to run the system in an internet bandwidth with capacity suitable to view a video feed which is about **38.4kbps**, and send control instructions to the robot which doesn't require a bandwidth more than **few number of bytes**. These two operations occur at the same time. So the minimum bandwidth required for this system to work is **39kbps.**

The user must have web browser software installed in the client device. It is important that the web browser supports **ActiveX Controller**, and **LabVIEW Run-Time Engine 7.1** or above for the robot control software.

The web server must have a **suitable traffic bandwidth** to run the system properly. The client bandwidth requirement applies also on the server.

The robot system needs to run at a **Wireless LAN** network operating at the **IEEE 802.11g/b** standards, with bandwidth requirement just like the client and the web server. Using access point with suitable range to enable the robot to move freely is also a very important factor. **A range of 150-300 ft** is fairly enough for almost every indoor application. For other applications, the range depends on the coverage area of the robot. A stronger access point is needed if the robot reaches for more than 300 ft.

Further requirements maybe needed in order to increase the system performance. Increasing the Internet bandwidth in the client side, the web server, and the robot system side, all this, decreases the possibilities for a delay in the system. Also running the client side on a wired network is more

efficient than using a WLAN network. This is because the wired networks have a good Quality of Service (QoS) while wireless networks are known with lacking at this matter.

Summarizing the above, the essential requirements to operate the system at an infrastructure network are:

- Minimum assigned bandwidth of **39kbps.**
- At the client side, Web browser software supporting **ActiveX Controller,** and **LabVIEW Run-Time Engine 7.1,** the system was tested and worked efficiently using Internet Explorer 8.0 Browser.
- Access point with **range of 150-300 ft**, and operating at the **IEEE 802.11g/b** standards.

3.5.2 Ad-hoc Networks

3.17 System Operating in Ad-hoc Network

For the system to operate in ad-hoc network, figure 3.17, all what is needed at the robot side, is to run the transceiver in the ad-hoc mode, this can be done simply by turning the ad-hoc switch on. On the client side, the user must associate with the ad-hoc network which the transceiver has creates, and the robot control software installed on the client device with LabVIEW software. The user must be in the transceiver's range for this mode to operate successfully.

3.6 Delay Factors

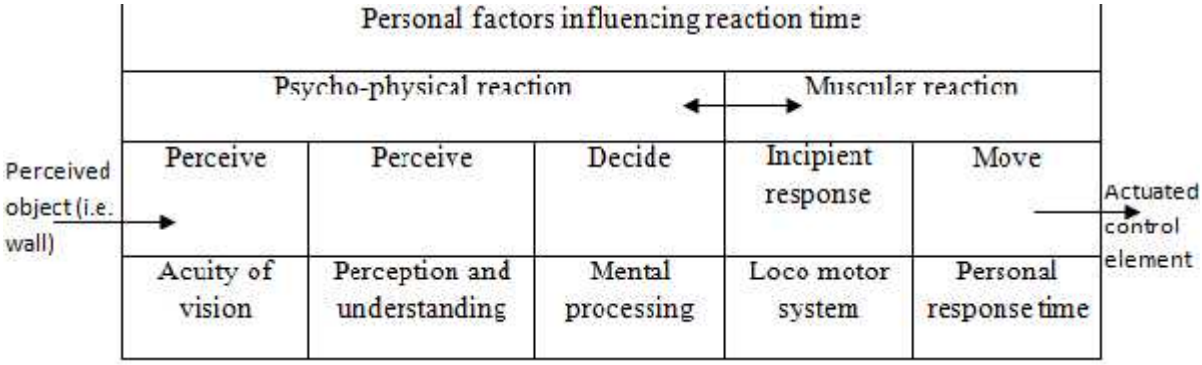
Delay is a very important factor for the operation of the system, it should be considered to avoid system instability.

There are four main sources of delay.

3.6.1 Human Delay

Human interaction delay is the period between the recognition of the object, and making the decision to give new orders [7]. In order to find a specific value for this component, we must study the reaction time of humans in general. See table (3.1).

Table 3.1 Reaction Time



The reaction time is not fixed value; it ranges from 0.3 to 1.7 s, depending upon the user and on external factors. See table (3.2).

Table 3.2 Effect of Personal and Extraneous Factors on Reaction Time

	Reduction down to 0.3 s ← → Extension up to 1.7 s	
A) Personal factors	Trained, reflexive response	Inexperienced response
	Good frame of mind; optimum performance potential	Poor frame of mind e.g. fatigue
	Highly skilled user	Low level of user skill
	Youth	Advanced age
	Anticipation	Inattentiveness, distraction
	Physical and mental health	Health disorders in response groups
B) Extraneous factor (obstacle situation)	Uncomplicated, easily comprehended, predictable, familiar.	Complicated, difficult to comprehend, unpredictable, rarely encountered
Type of perceived object	Explicit, Conspicuous	Equivocal, inconspicuous
Location of perceived object	Within field of vision	At edge of field vision
Nature of control element	Logical layout of mechanic control	Poor layout of mechanical controls

3.6.2 Wired Network Data Rate

Most networks these days uses internet with bandwidth shared among users. Each user benefits from a part of the bandwidth, and this part is not fixed, sometimes the user may get the whole bandwidth as he subscribed, and sometimes uses his minimum sharing amount which is called the share ratio.

Assuming that the minimum share ratio in ADSL lines is 1/32, and assuming that the client and the robot uses a bandwidth of 128 kbps which is the minimum bandwidth available in ADSL, and also assuming that the network is full with users, in this rare case, both the robot and the client will have the minimum bandwidth of 128 kbps divided by 32, and that equals 4 kbps which is very limited bandwidth which may cause a respective amount of delay in the system.

Figure 3.18 shows the relationship between the Internet bandwidth, and the number of users sharing this bandwidth.

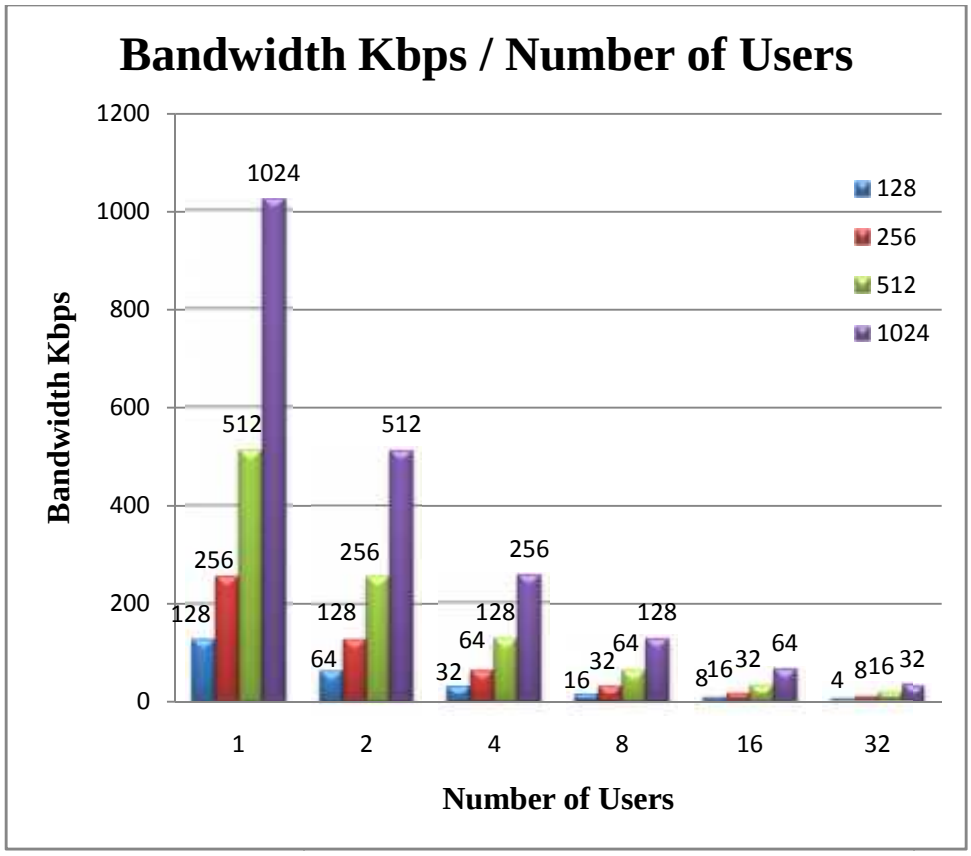


Figure 3.18 Bandwidth relation with the number of users

3.6.3 Wireless Network Data Rate

As the robot system runs wirelessly, the delay in the robot's network must be calculated. The delay in the wireless network increases with the increasing of the network users and the increasing of the robot distance with from the access point. As mentioned in the system operating requirements, the bandwidth requirement needed for the robot's network is to provide a data rate of 9kbps.

When operating the robot at a wireless network with data rate of 1Mbps which is rarely used these days due to its very limited bandwidth, the delay would be at maximum value when the network is full with users, and the robot reaches the maximum range of the access point.

There are many IEEE wireless technology standards on the market today, like Bluetooth, 802.11b, and 802.11a, and 802.11g, etc.

The 802.11b standard operates in the 2.4 GHz frequency range but uses the Direct Sequence Spread Spectrum (DSSS) modulation method. This results in a maximum data rate of 11 Mbps. It has a range of 60 -150 feet indoors. As the distance between the access point and robot increases, the data rate falls to 5.5, 2, and 1 Mbps, respectively.

802.11a operates in the 5 GHz frequency range (UNII band) and uses Orthogonal Frequency Division Multiplexing (OFDM). Due to OFDM, the maximum data rate of 54 Mbps can be obtained within a distance of 60 feet between wireless devices. As the distance increases, 802.11a devices will auto scale the data rate down to 48, 36, 24, 18, 12, 9, and 6 Mbps. A comparison of distance versus data rate is illustrated in Figure 3.19 below for the 802.11a, 802.11b, and 802.11g technologies.

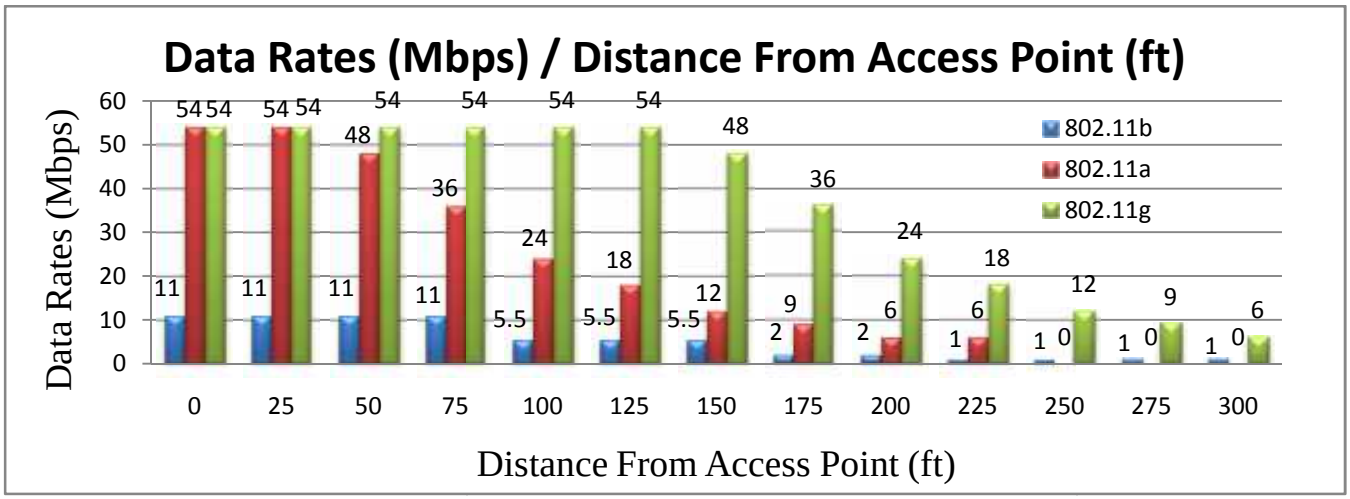


Figure 3.19 comparison of distance versus data rate

The 802.11g standard combines the best features of 802.11a and 802.11b. It operates in the 2.4 GHz range using OFDM which results in a maximum data rate of 54 Mbps at a range of 150 feet. The data rate will decrease incrementally from 48 Mbps to 6 Mbps as the distance between access point and client increases.

Figure 3.20 shows the decreasing of the bandwidth of the 802.11g when the number of the networks users increase.

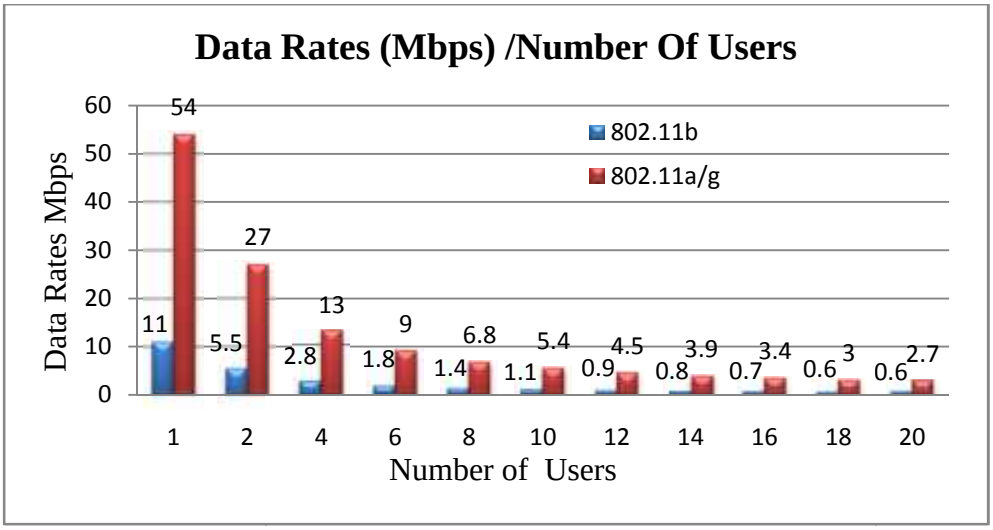


Figure 3.20 802.11g Data Rates Mbps for Number of Users

The tradeoff with 802.11g is in a lower capacity, versus 802.11a, to serve a large number of high speed WLAN users. The OFDM modulations allow for higher speed but the total available bandwidth in the 2.4 GHz frequency band remains the same because 802.11g is still restricted to three channels in the 2.4 GHz band, unlike the eight that are available in the 5Ghz band.

The primary factors are the data rate and the distance or area the robot can cover and still maintain connectivity. Below in Table 3.3 is a comparison of each standard.

Table 3.3 Wireless Technology Standards Comparison

Technology	Frequency	Data Rate	Distance (indoors)	Modulation Scheme
802.11b	2.412 – 2.462 GHz	11 - 1 Mbps	150 - 300 ft	DSSS
802.11a	5.15 - 5.825 GHz	54 - 6 Mbps	60 - 180 ft	OFDM
802.11g	2.4 GHz	54 Mbps	150 - 300 ft	OFDM

The decision of which technology to apply will depend on the requirements of the system. Each access point can practically service between 32 to 64 users, depending upon manufacturer. The bandwidth will be shared by all users. The number of radio channels associated with each technology will vary by technology and product manufacturer. Multiple channels allows for multiple access points to be collocated in order to provide access for a larger number of users.

Assuming that 802.11b is the applied wireless standard, and the robot is 200ft far from the access point, in a network with 20 users, the data rate at this far would be 2Mbps and the whole 11Mbps bandwidth is shared among 20 users. So, even if we assume that the 2Mbps is assigned to the robot, the throughput of this bandwidth will not be enough to operate the system properly.

In order to limit the delay as much as possible, it is recommended to run both the client side and the robot on a network with bandwidth that guarantees an assigned bandwidth of at least 39 kbps which is fairly suitable for this project, because the bandwidth requirement for this system as discussed before is about 39kbps, and also using 802.11g access point with bandwidth of 54Mbps with rang of 150-300 ft.

There is only one unavoidable case related to network delay may cause a robot to stop receiving instructions and sending the video, it would be losing the connection with the ADSL network. To avoid unstable state of the robot in this case, the robot can be programmed to freeze and stop any running component.

3.6.4 Processing Delay

This system deals with a respective number of peripherals connected to the PIC microcontroller, three motors, video camera, wireless transceiver, and sensors. This may cause a noticeable delay when executing two operations at one time (sending the video feed, and receiving the control instructions), the PIC can't perform multitasking such as PC's or other complex microcontrollers. To overcome this obstacle two PICs were used, one for the video feed, and the other for the control instructions.

Each PIC operates at **8MHz** frequency or **0.125 µS** cycle duration, and as known that PIC's are from the RISC architecture which means, one instruction per cycle, except branching instructions.

The processing delay is not related only to the PICs, there is Analog-to-Digital converter (ADC) which consumes time when doing conversions.

There is a minimum acquisition time for the ADC that must be taken into consideration. Figure 3.21 shows a snipped image form the PIC18F4550 datasheet contains the calculations for the minimum acquisition for the ADC.

$$
\begin{aligned}
T_{ACQ} &= \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \\
&= T_{AMP} + T_C + T_{COFF} \\
&= 2\,\mu s + T_C + [(\text{Temperature} - 25^\circ C)(0.05\,\mu s/^\circ C)] \\
T_C &= C_{HOLD}(R_{IC} + R_{SS} + R_S)\ln(1/2047) \\
&= -120\,pF\,(1\,k\Omega + 7\,k\Omega + 10\,k\Omega)\ln(0.0004885) \\
&= 16.47\,\mu s \\
T_{ACQ} &= 2\,\mu s + 16.47\,\mu s + [(50^\circ C - 25^\circ C)(0.05\,\mu s/^\circ C) \\
&= 19.72\,\mu s
\end{aligned}
$$

Figure 3.21 Minimum Acquisition Time in the PIC18F4550 ADC.

The ADC acquisition time value is essential when writing the code, to avoid incorrect readings for the ADC.

3.7 System Security

Security in this system is very essential. It protects the system from being hacked, or spying on the transferred data, it protects the control system from being entered by anyone who is not authorized. In this system, there are two levels of security, the first level is on the web site (user-level), and the other one is between the robot control software on the web site and the transceiver (connection-level).

At the web site, each user wants to use the system he should have an account, with username and password. Any user that tries to enter to the robot control software web page and he does not have an account, he will not be able to access to that web page, he will be redirected away. Also if the user has an account, but he typed the user name and password incorrectly, he will also be redirected away.

If a user has an account with user name and password, and he entered both of them correctly, then he will access to the robot control software web page, but still does not have the authority to control the robot yet. At first he must be sure that no other user is using the control software. If the software is already in use, a message will appear to the user that the control of the software has been granted for another user. This prevents controlling the robot by more than one user.

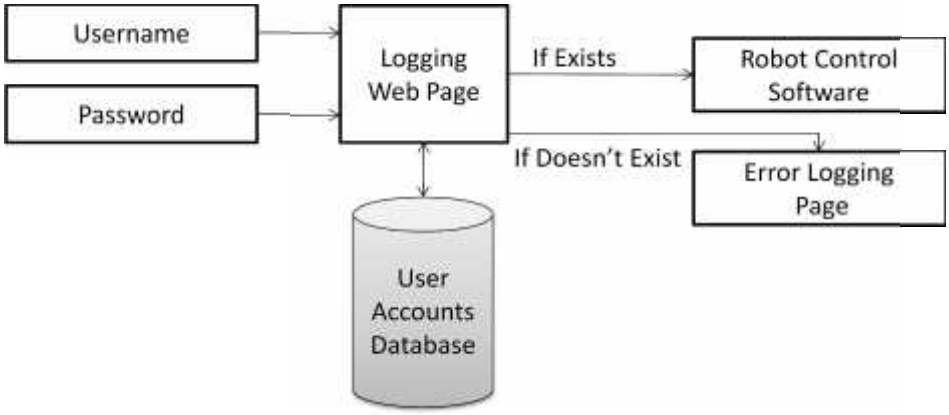


Figure 3.22 User-Level Security Diagram

The previous security level was to secure who have the privilege of controlling the robot. Another security level that is crucial and had to be applied in this system, it is the security on the connection itself, the connection between the WiFly transceiver and the web server, including the transferred data, devices ID, users, etc. This level of security is applied in the transceiver by a password that is sent by the web server and must be accepted by the transceiver at the beginning of every connection. The strength of applying the password, is that it must be sent in one TCP packet, and if the first character sent does not match the stored password in the transceiver, the connection will be dropped.

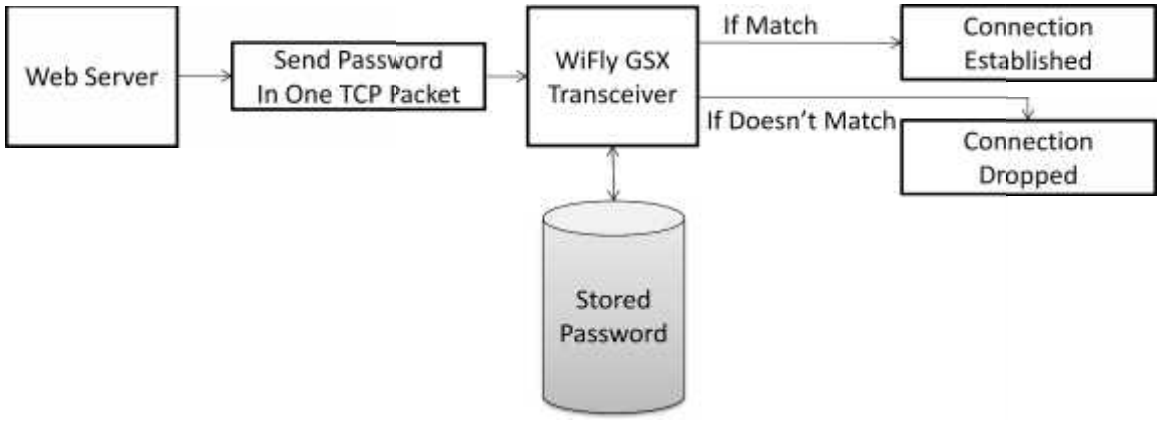


Figure 3.23 Connection-Level Security Diagram

More security in this system is related to the security methods implemented in the WLAN networks, between the transceiver and the access points. The video and the commands that are transferred must be encrypted before transmitted, and decrypted at the receiving side. The encryption and decryption occur at the wireless phase of the transmission, which is between the access point and the transceiver.

The WiFly GSX transceiver uses the Direct Sequence Spread Spectrum (DSSS) as an encryption method as well as a modulation method. The access point also well use the same techniques for communicating with the transceiver since a handshaking process happened between them at the authentication and the connecting phase.

These techniques make it almost impossible for anyone to know the transmitted data between the access point and the transceiver unless the code for the DSSS is known.

Chapter Four

Hardware Design Implementation

4.1 Introduction.
4.2 Electrical Part
4.3 Mechanical Part

Chapter Four

Hardware Design Implementation

4.1 Introduction

Building robotic systems have always been a multipart projects, mechanical part, electrical part, maybe chemical part, or even biological part, depending on the robot application that it was built for. A division must be made between these parts in order to build the system.

In this project, the hardware design will be divided into two parts, the mechanical part and the electrical part. The mechanical part includes the mechanical components of the robot, the DC motors, the stepper motor, and the sensors. The electrical part includes the electrical circuits of the system, the wireless transceiver, the PIC microcontrollers, the video camera, and the battery.

In figure 4.1, a general system interfacing block diagram is shown, it represents the interfacing of the whole system. In the coming sections, detailed interfacing techniques and requirements will be discussed in both the mechanical and electrical parts.

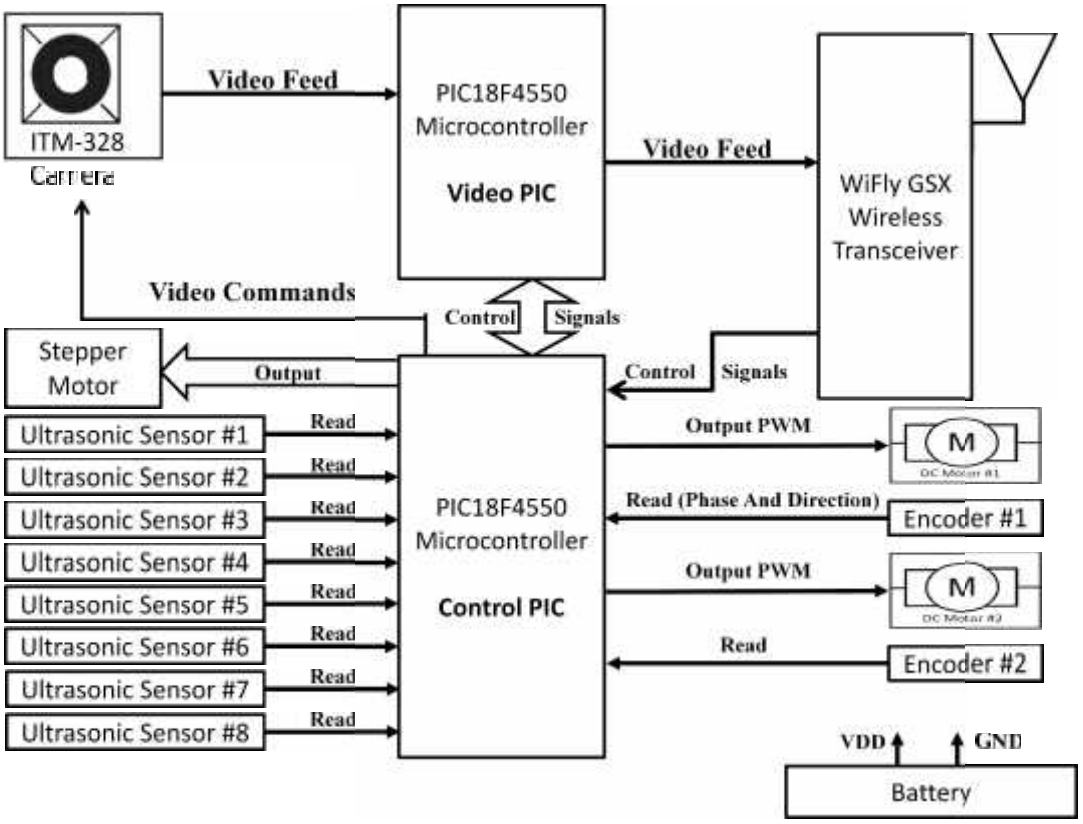


Figure 4.1 General System Interfacing Block Diagram.

4.2 Electrical Part

In this section, the electrical part of the system will be discussed. This includes interfacing techniques, requirements, and ideas to connect the electrical parts together, the wireless transceiver, the video camera, and the battery all with the PIC microcontroller, figure 4.6.

4.2.1 Interfacing Two PICs

This system requires multitasks, which means two or more tasks at the same time, sending the video feed while receiving the control instructions, and reading the sensors. Implementing this on a single PIC would produce a great amount of delay that would make the system unstable.

A solution of using two PICs is produced in this section. Using two PICs for this system would increase the performance, one PIC is connected to the camera and is responsible for sending the video to the WiFly GSX. This PIC is called the **Video PIC**. The other PIC is responsible for the movement of the robot, rotation of the camera base, and the sensors. This PIC is called the **Control PIC**.

As shown in figure 4.2, the two PICs communicate with each other through three digital I/O pins. For example (RB0, RB1, and RB2), this is needed to exchange some data between the control PIC and the video PIC needed for the configuration of the serial camera.

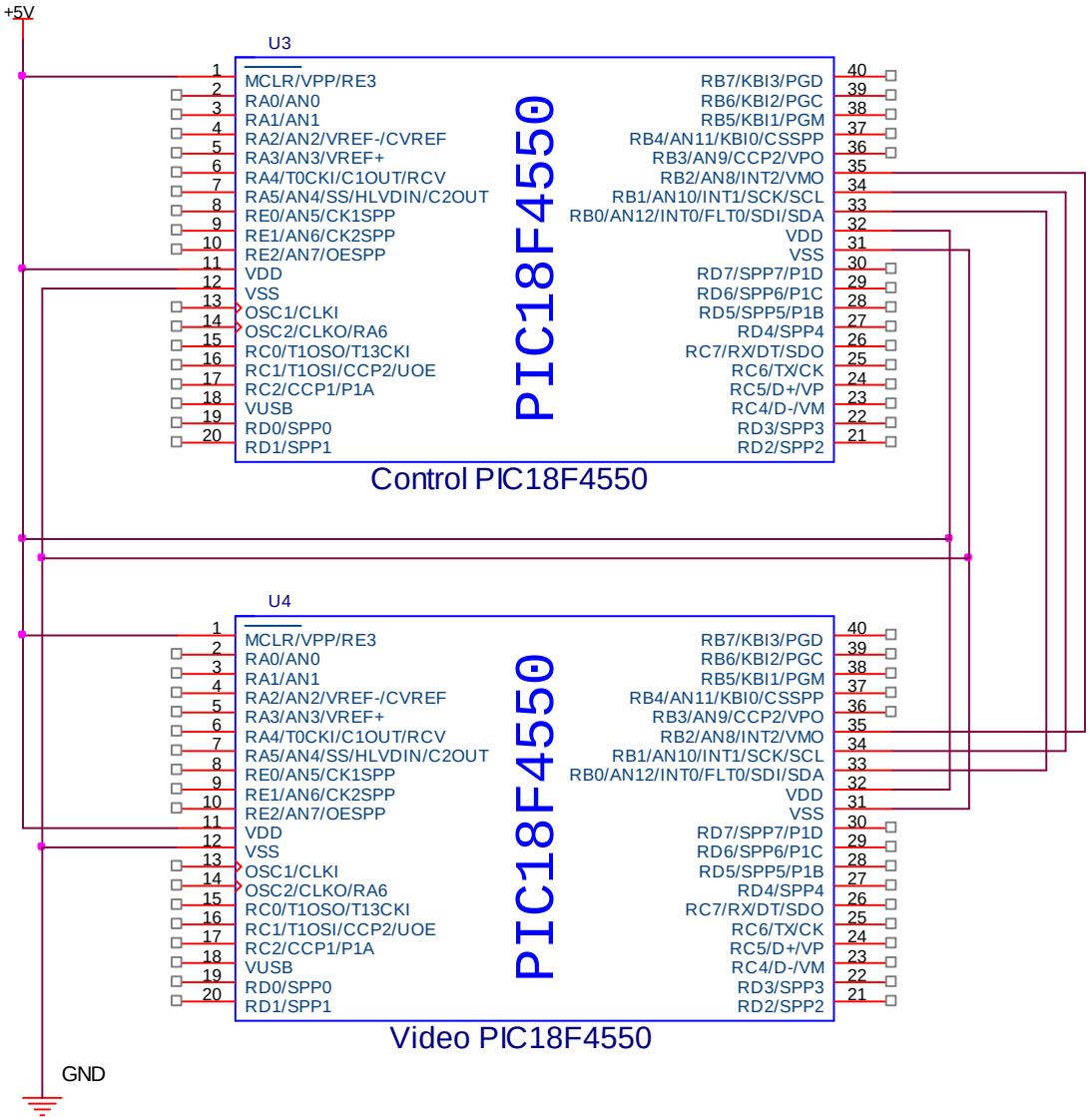


Figure 4.2 Interfacing Two PICs

4.2.2 Interfacing WiFly GSX Wireless Transceiver

In order to connect the wireless transceiver to the PIC microcontroller, a brief study was made about the pins for both the PIC and the WiFly GSX wireless transceiver, and the idea was that serial communication can be applied between them. The PIC18F4550 supports serial communication through the built in USART, two pins are available, one serial data input (RC7/RX), and the other is the serial data output (RC6/TX). The WiFly GSX also supports serial communication through the UART_TX (output) and UART_RX (input) pins.

The WiFly GSX is connected to both PICs (figure 4.3). The connection to the video PIC is done through the UART_RX pin connected with the RC7/RX pin, because the video is an output from the PIC to the WiFly GSX, and connection to the control PIC is done through the UART_TX pin connected with the RC6/TX pin, in order to send control instructions to the PIC.

Due the difference in the voltage levels between the PIC Microcontroller (5V) and the WiFly GSX transceiver (3.3V), connecting the USART_TX of the PIC with the UART_RX of the transceiver is done through three 10KΩ resistors connected serially, and the wire of the UART _RX is taken from a 20KΩ. The values of these resistors were found using the voltage divider rule.

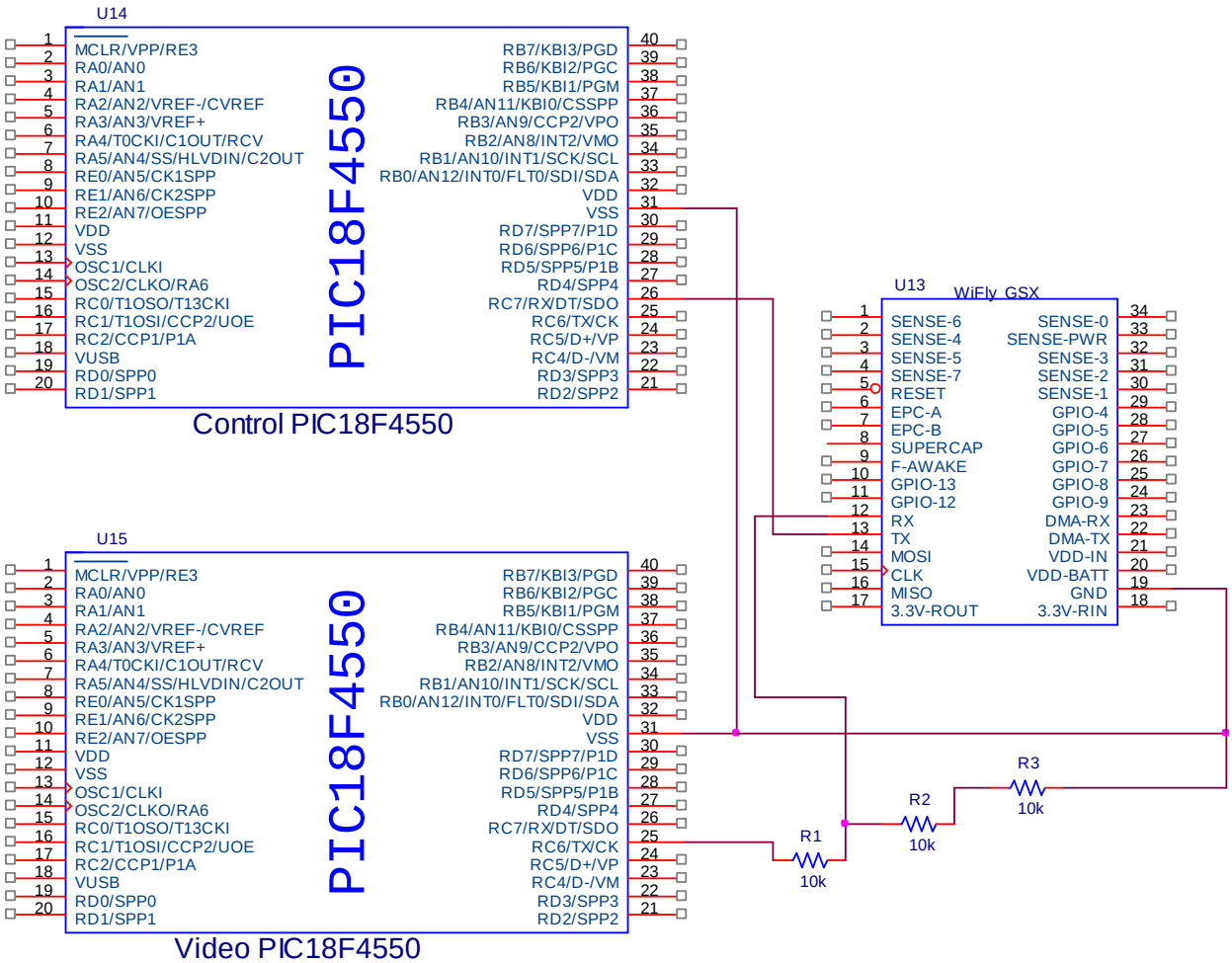


Figure 4.3 Interfacing WiFly GSX Wireless Transceiver

The WiFly GSX baud rate can be configured to be from 2400 bps, to 1Mbps which is suitable to the project requirements.

4.2.3 Video Camera Interfacing

The video camera is a very important component in this system. It is responsible for the visual feedback for the user while doing the control operations.

The ITM-C-328 camera which is the one used in this system has only four pins, voltage pin (Vcc), ground pin (GND), transmit data pin (Tx) ,and the receive data pin (Rx).

Connecting the video camera to the PIC is done through the serial data line from the camera transmit data pin (Tx) to the video PIC through the USART input pin (RC7/Rx), figure 4.4. And the data pin (Rx) is connected with the Control PIC with the USART output pin (RC6/Tx), through a voltage divider circuit as the WiFly, because the camera operates at 3.3V.

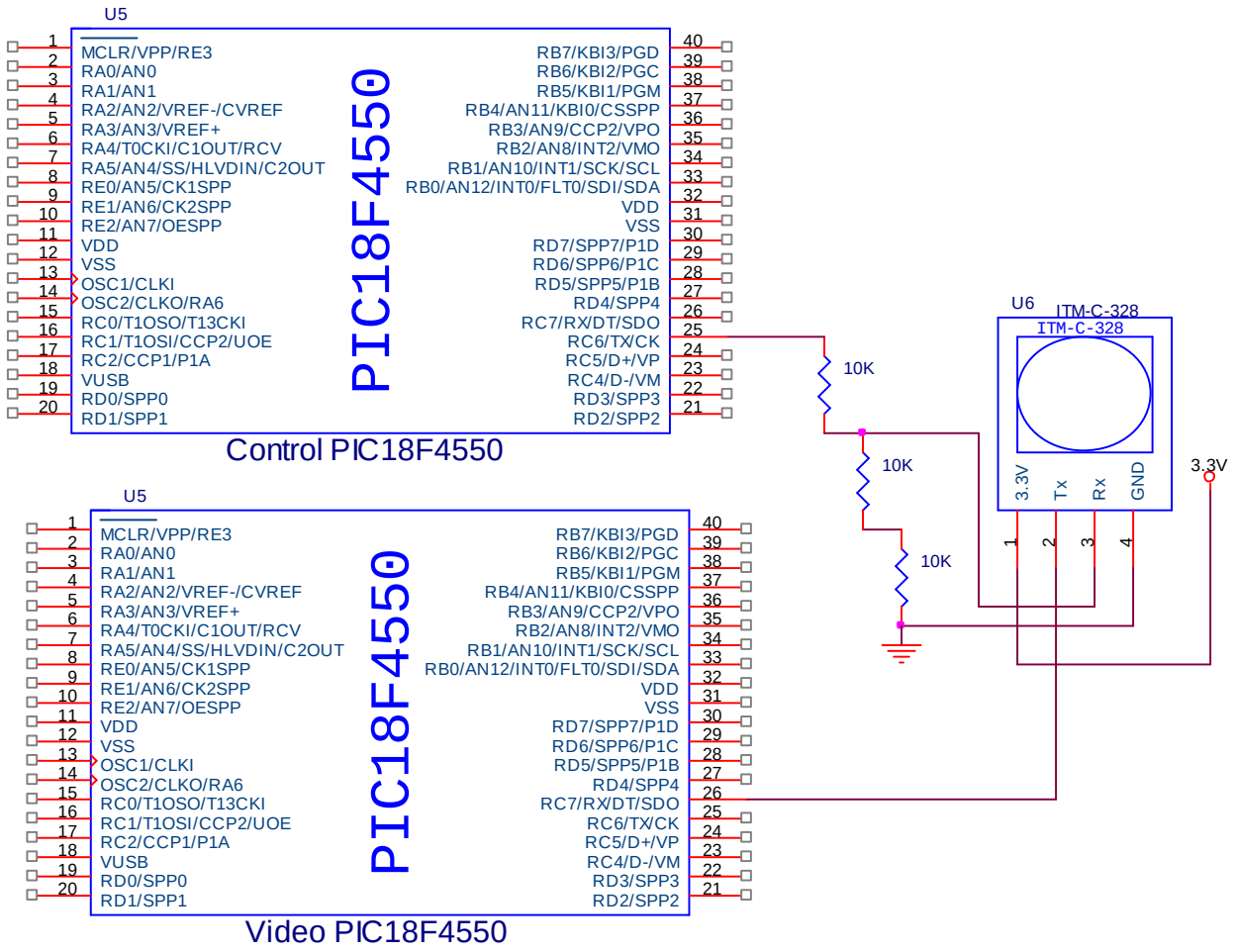


Figure 4.4 Video Camera Interfacing.

The Camera is connected to both PIC's because it is programmable and a connection must be established with it using the Rx and Tx pins, and because the Tx pin in the Video PIC is occupied by the WiFly Transceiver, the camera receives the commands to start connection from the control PIC and sends the video feed to the Video PIC.

4.2.4 Battery

The battery is the portable power source for the system, it feeds all the systems' component with power.

In order to connect some components with the battery, regulators are needed to control the power entering the components, to avoid any problems of overloading the components with power, and burning them.

Figure 4.5 presents the connection of the electrical components with the battery.

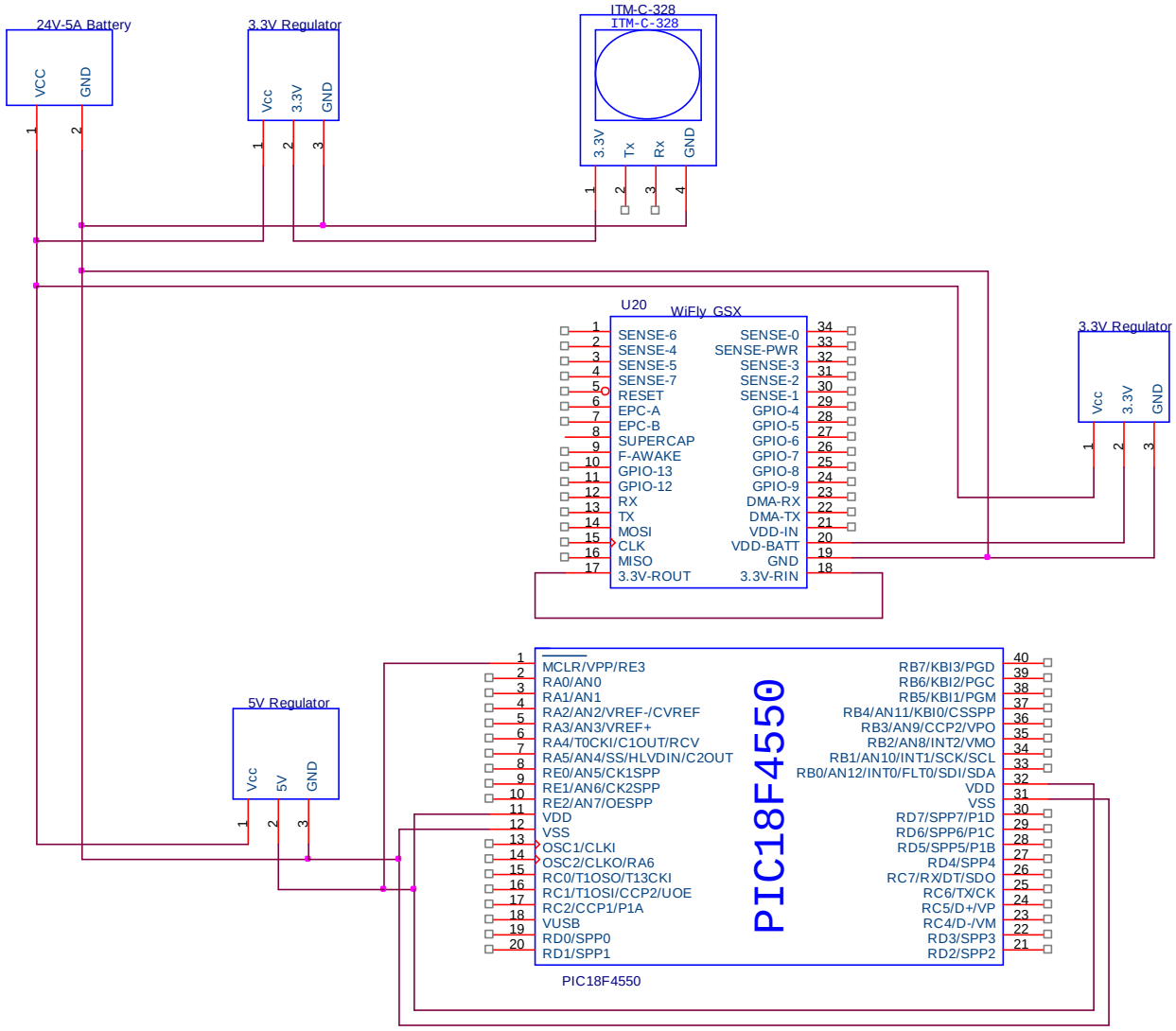
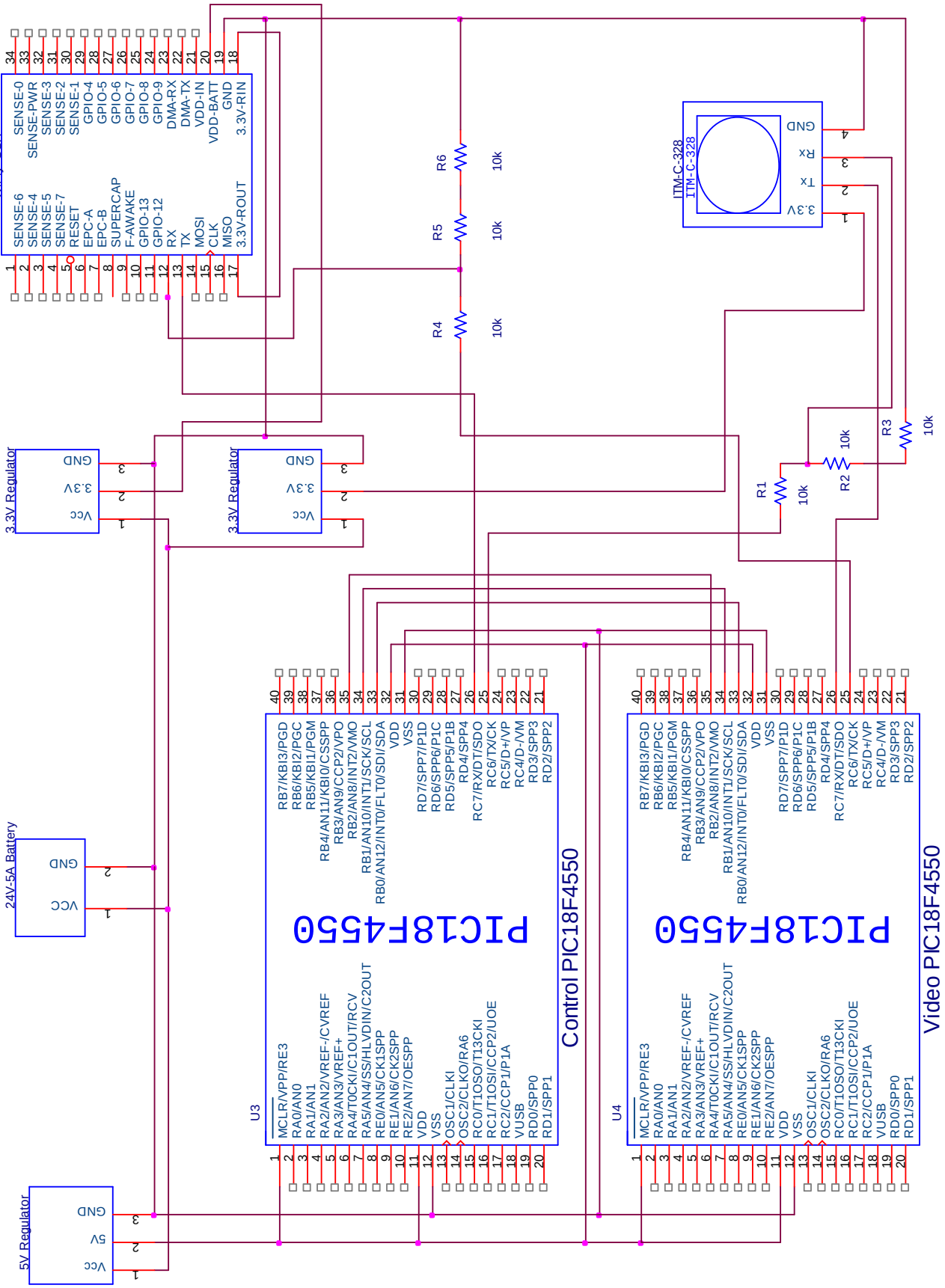


Figure 4.5 Battery Interfacing.



4.6 Electrical Components Interfacing

4.3 Mechanical Part

In this section interfacing the mechanical part of the system will be discussed, the DC motors, the stepper motor, and the sensors. Mechanical part means the part that is performing mechanical operations despite the fact that the components are electrical components in their nature, for example, the stepper motor is an electrical component in the system that performs mechanical operation, which is rotating the camera base.

For more discussion about the mechanical components see Part One the Mechatronics Part Chapter Two, (page 16).

Chapter Five

Software Design Implementation

5.1 Software System Design

5.2 Detailed Design Description

5.3 System Configuration

5.4 Web Site Design and Implementation.

Chapter Five

Software Design Implementation

5.1 Software System Design

The implementation of a web based control system requires a number of tools and methods provided by some applications such as the LabVIEW software. This software supports powerful methods to handle TCP/IP communications, which is the mean of communication between the robot and the user.

In this chapter, a detailed discussion about the system will be covered. Figure 5.1 shows the software detailed system diagram.

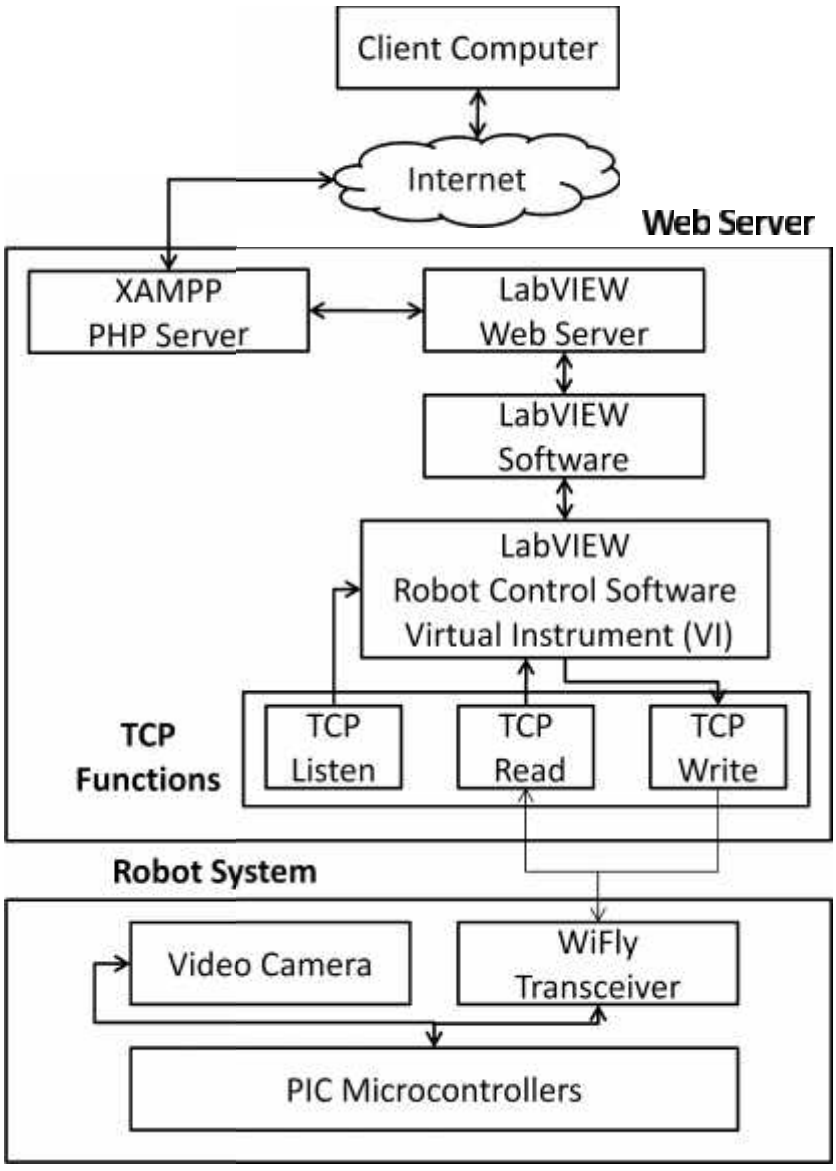


Figure 5.1 Software Detailed System Diagram.

5.2 Detailed Design Description

In this section a detailed description about each software component is presented. It discusses the importance of each component and its rule.

5.2.1 Client

The client is the remote controlling side of the system. Any device in the world with an internet access and a web browser and has the authorization to control the system can be called a client, whether it's a computer, PDA, cell phone, or any other device.

In the client side, in order to perform the controlling operations, only a web browser with ActivX controller is needed, and LabVIEW Run-Time Engine 7.1 which will be provided for the user to download when the page is opened. ActiveX controller is installed into most known browser, Internet Explorer, Firefox, and many more. It is needed to run the LabVIEW Virtual Instrument (VI) that is recognized by the browser as a Plug-in. So, when the user opens the robot control software page, the only thing the user needs to do is enabling the plug-in pointed by the ActiveX controller as shown in figure 5.2.

Figure 5.2 Enabling ActiveX controller.

5.2.2 Web Server

The web server requires two different web server applications, the XAMPP web server to handle the PHP pages, and the LabVIEW web server which enables remote access to the Virtual Instrument (VI) that runs the robot control software.

- **XAMPP Web Server**

The XAMPP web server is needed to run Apache web server, which is generally recognized as the world's most popular Web server (HTTP server). Beside the Apache server XAMPP supports MySQL, and PHP. All of them are need in this system.

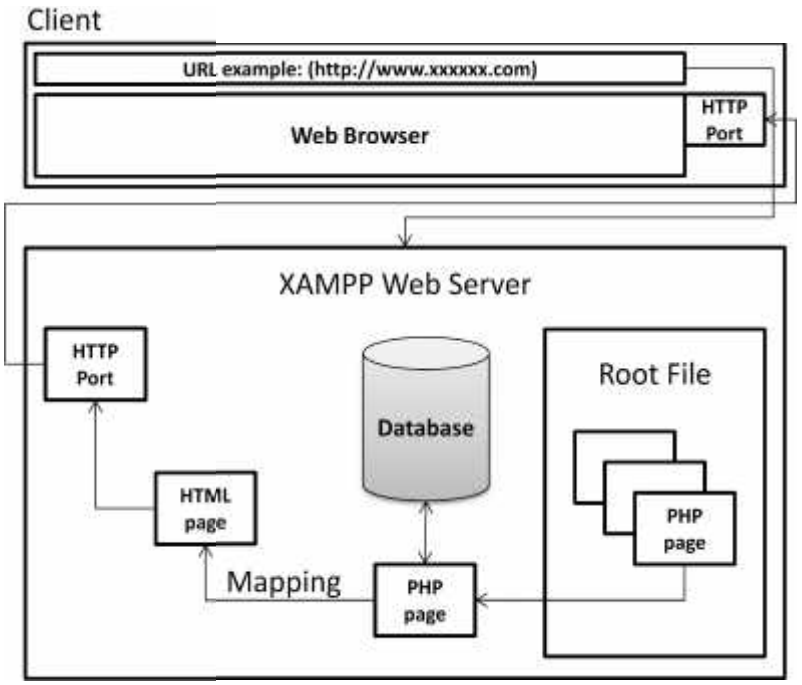


Figure 5.3 XAMPP Web Server Rule.

As shown in figure 5.3, the XAMPP web server receives requests to open web pages. It fetches the required page from the root file of the server, which is the file containing the files of the web site (pages, connections, images, etc), then maps the PHP page to its HTML equivalent, so that any browser can compile the HTML code and present the user with the required web page.

- **LabVIEW Web Server**

The LabVIEW development environment and any applications built from VIs contain an integrated Web server. This web server enables remote controlling of the VI in a specific port number. The web server makes publishing the VI over the internet very easy, as the VI will be hosted in an HTML web page with a specific address, and the access to this web page is done through any HTTP port.

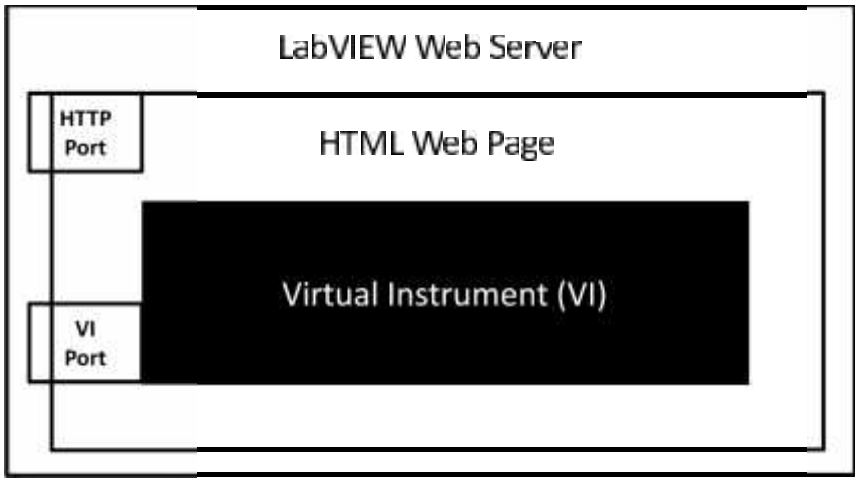


Figure 5.4 Hosting VI in HTML webpage

The combination between two web server applications requires, splitting ports among them and uniting the IP address of both web servers so that they can run as one unit, figure 5.5.

Since the user uses the HTTP protocol to log into the web site, and that protocol occupies specific port numbers like 80 and 6060, the XAMPP server can output the HTML pages into port 80, and the LabVIEW server can publish the VI into an HTML page through port 6060, and the VI access can go through port 3363. In this way ports are specified for each server and no conflict in ports is suspected to occur.

Each web server requires a static IP address so it can be accessed through it. The existing of two web servers requires two IP addresses, but in this system the two web server applications must have only one IP address so that the user can connect with the robot through a single web server. This is accomplished by using the same root file for both of the web server applications.

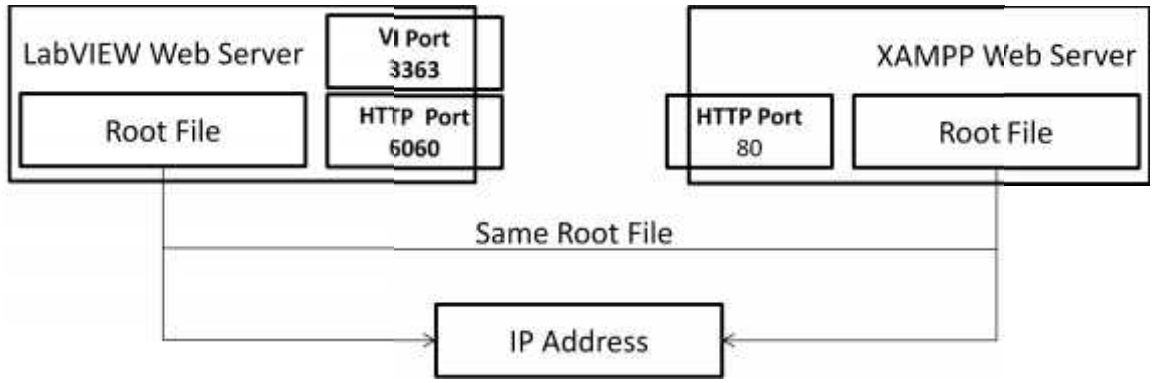


Figure 5.5 Combining Two Web Server Applications In on Web Server.

5.2.3 Robot Control Software

As described in section 3.3, the robot control software is the user interface of the controlling process over the robot.

In this section the main concepts of programming the software will be mentioned, but the developed VI's will be presented in the Appendix.

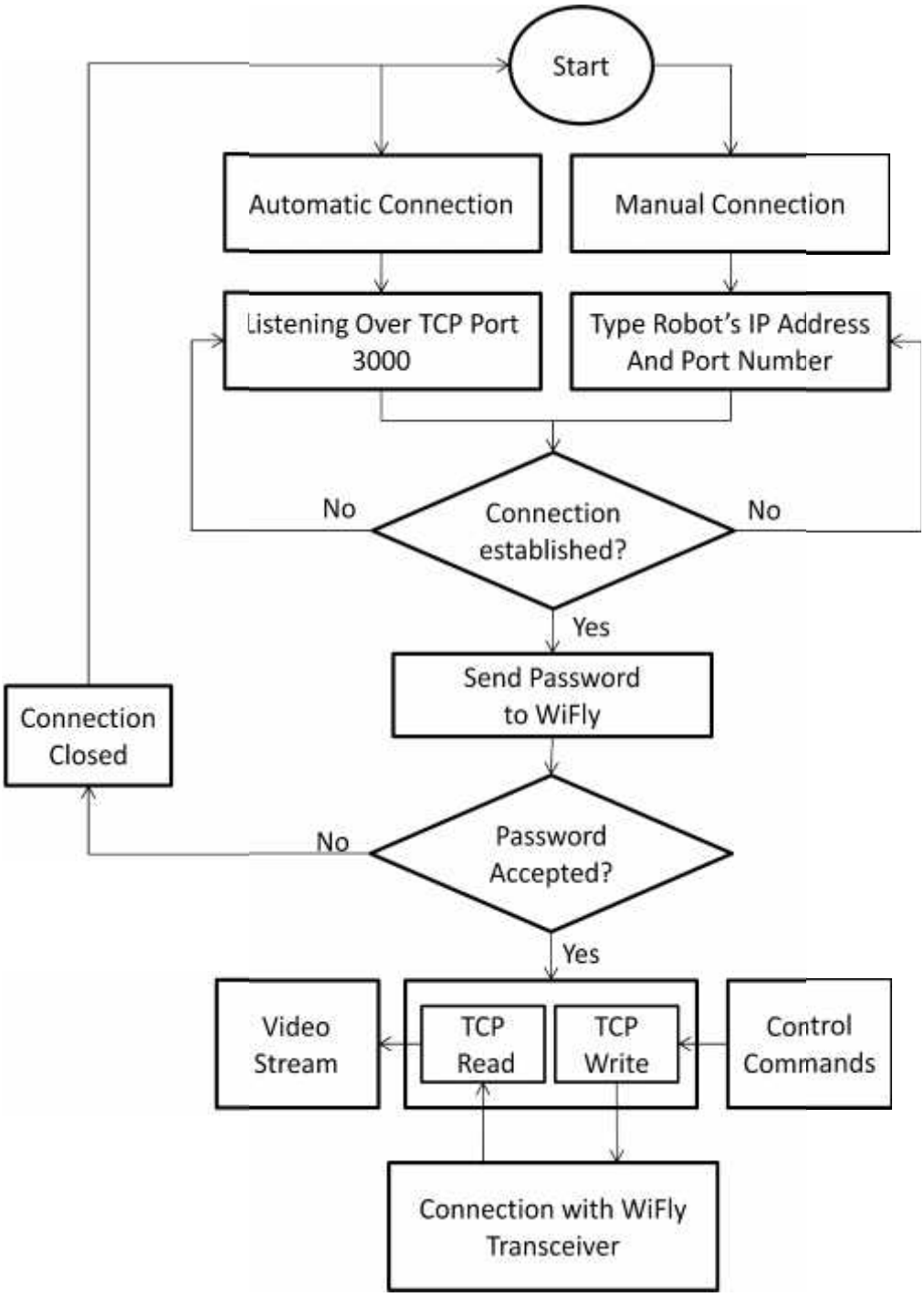


Figure 5.6 Robot Control Software Flow Chart.

As shown in Figure 5.6, the controller software can connect to the robot automatically upon power up of the robot. If this option is not needed, a connection using the IP address of the WiFly transceiver and the port number can be done. Anyway, after a connection is established, the transceiver request the password from the web server in order to authenticate the connection, if the authentication completed successfully, the software is ready to receive the video feed from the camera, and to send the user commands to the robot.

Figure 5.7 shows the Robot Control Software user interface, with all the options and feedbacks from the robot. The Block diagram which holds the code of the robot control software will be presented in the Appendix.

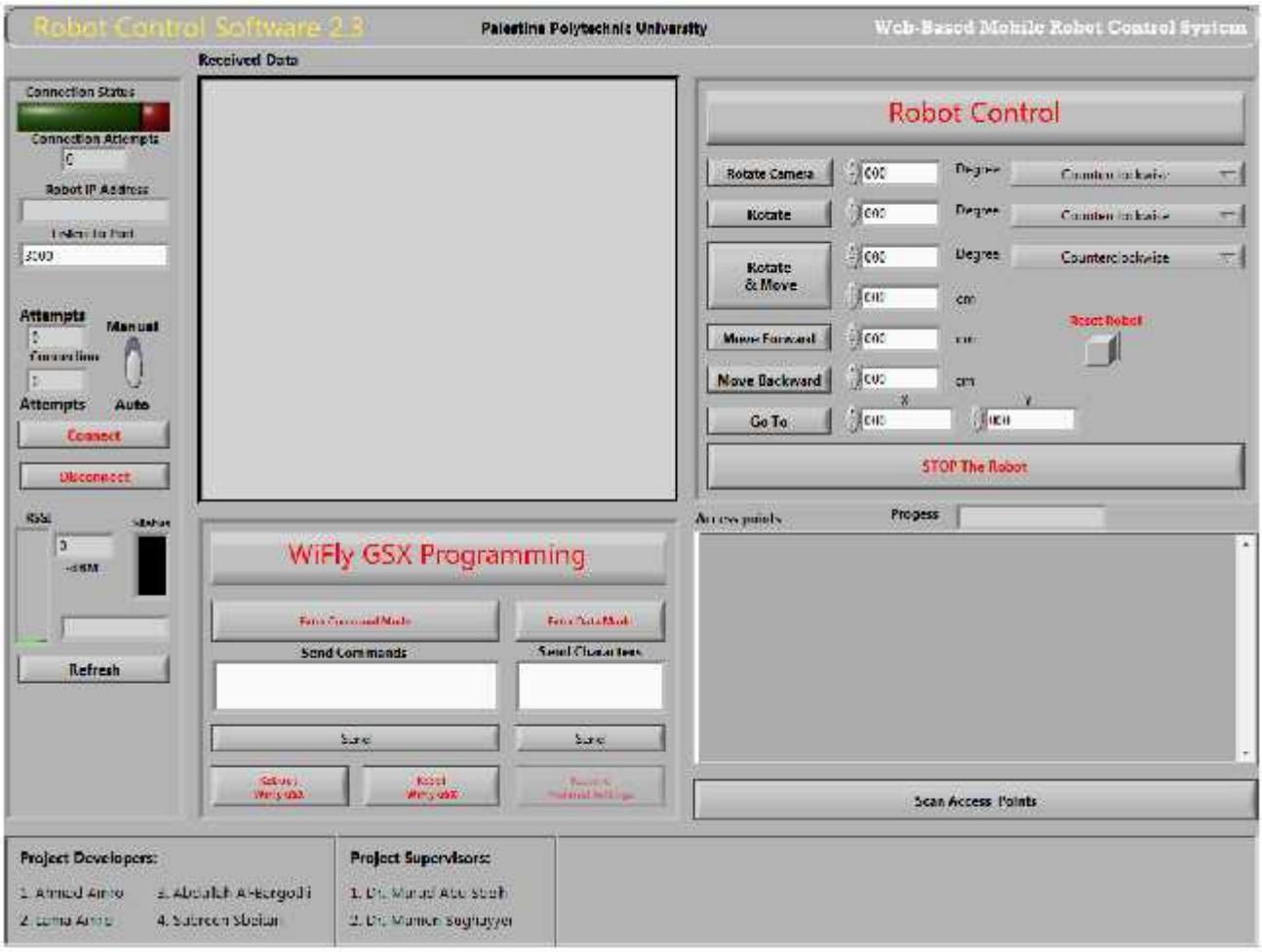


Figure 5.7 Robot Control Software User Interface.

The Robot Control Software VI Block Diagram (code) is presented in Appendix H.

5.2.4 WiFly GSX Transceiver Programming

Upon power up, the transceiver's behavior is shown in figure 5.8. The first thing the transceiver does when it's turned on is looking for a wireless networks, it has 13 channels at which scan for WLAN's occurs. After the scan is completed, the transceiver will try to associate with the network that has the strongest Received Signal Strength Indicator (RSSI). If a connection is established, the transceiver will try to connect with the web server using the stored IP address of the server, and the connection is done through the stored port number which is 3000 in this system. The authentication is done as follows. The transceiver will challenge the web server for a password. The web server is programmed to reply with the correct password. After a successful authentication, the transceiver will be ready to exchange data with the web server.

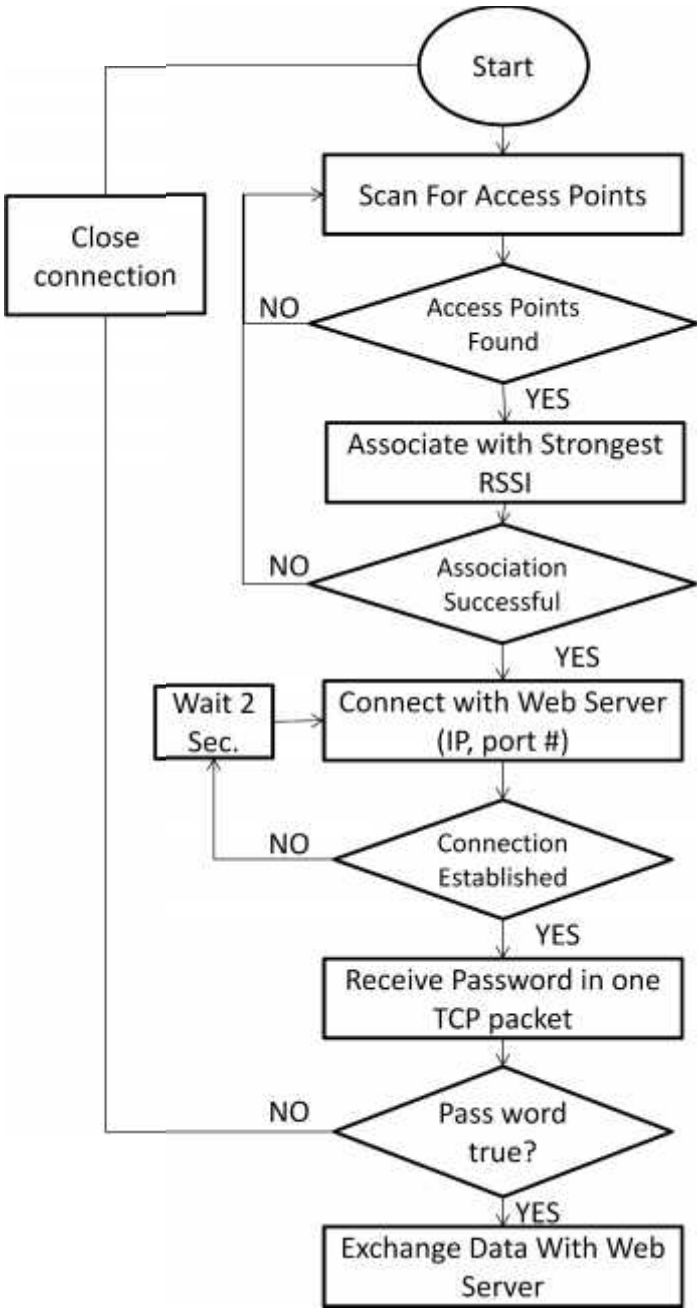


Figure 5.8 WiFly GSX Behavior Upon Power Up.

For the transceiver to function as required, it must be programmed by typing a number of commands in a terminal program using Telnet service, for example TeraTerm software. When the transceiver is programmed and the configurations are set, it should be able to access Wi-Fi networks and transfer or receive data.

In this project connecting with the WiFly transceiver at the first time was done wirelessly by connecting with the transceiver in the adhoc mode. Enabling adhoc mode via hardware was done by connecting PIO9 pin to the 3.3V line at power up. Later, the adhoc mode was no longer needed. Connecting with the transceiver was done through an access point the transceiver automatically associated with.

When the module powers up in adhoc mode the WiFly module creates an adhoc network with the following information:

SSID: WiFly-GSX-7d where 7d are the final two bytes of the transceiver's MAC address, which is 00:06:66:00:3c:7d.

IP address: 169.254.1.1

Starting a Telnet session using TeraTerm to set up a connection with the module was done as follows:

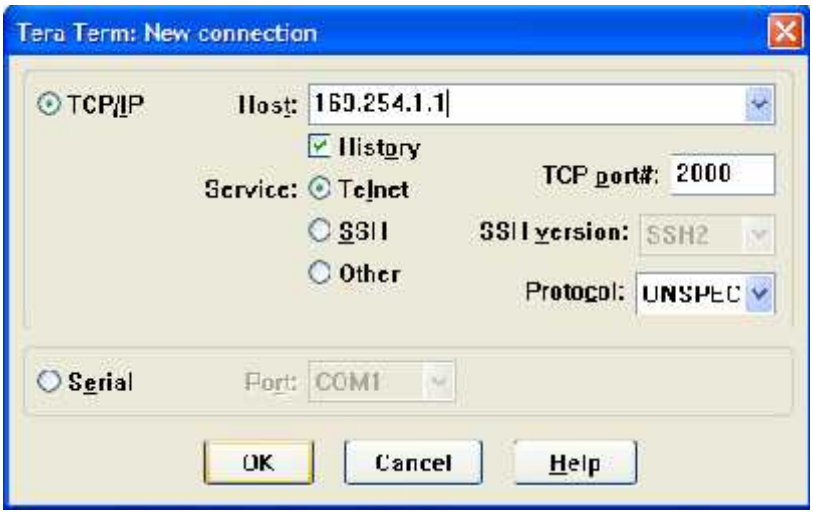


Figure 5.9 Starting a Telnet session using TeraTerm in adhoc

As shown in figure 5.9 the module replied with the string *HELLO* indicating that the connection has been established. In terminal window, the WiFly GSX module was entered into command mode by typing $$$. A CMD reply confirmed that the transceiver was running in the command mode.

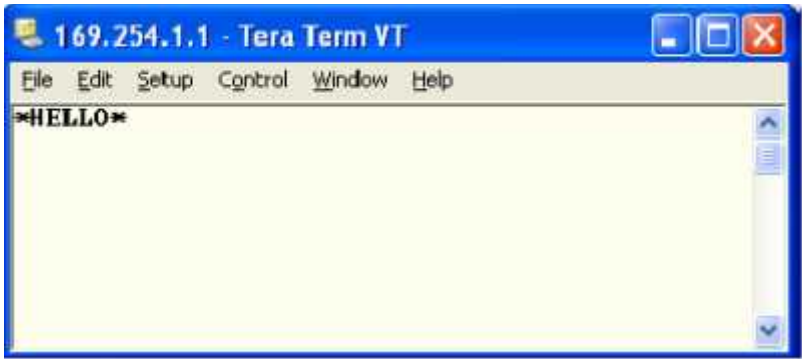


Figure 5.10 A Telnet session using TeraTerm

The message which indicates that a connection with the transceiver has been established *HELLO*, was modified in this project and replaced with the *PPU* message. This was done by this command (**set comm remote *PPU***).

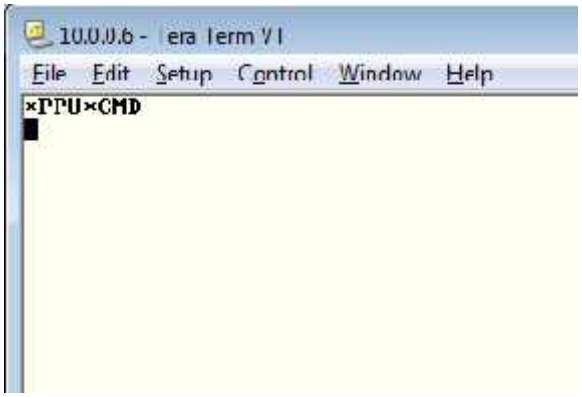


Figure 5.11WiFly in Command Mode

The (**set wlan chan**) command is used to determine in how many channels between the 13 channels of the transceiver the scan can be performed. If this command was followed by a "**0**", this means that the scan will be performed across all the 13 channels returning the SSID for the WLAN's in range. In this project the (**set wlan chan 0**) command is used to perform the scan at all the 13 channels.

The (**set wlan join**) command is used to determine the policy for association with network access point. If the user wants to join with a network manually, the command must be followed by"**0**", this prevents the transceiver to join a network automatically. If the user wants to join a network with the stored SSID, then the command must be followed by "**1**". A very useful feature for this transceiver is that it can automatically scan and connect with the network which has the strongest signal, this can be done by writing the command followed be "**2**". In this project the (**set wlan join 2**) is used, which makes the transceiver more powerful cause it connects with the strongest access point.

The transceiver must have an IP address to be a part of any network, this IP could be assigned manually using the (**set ip address**) command followed by the wanted IP, or, if the DHCP client is turned on, the transceiver will be granted an IP address automatically from the network during association. The DHCP client can be configured using the (**set ip dchp**) command. If this command was followed by "**0**", then the DHCP client is off, and the transceiver will use the stored static IP. If the command was followed by "**1**", this means that the DHCP client is on and the transceiver will be given an IP address and use the gateway address of the access point. In this project, the (**set ip dchp 1**) is used, because it is essential for the transceiver to use an IP address in the range of the network, since the user and the robot will be in different places many times, and the fixed IP does not always match the IP range of all networks.

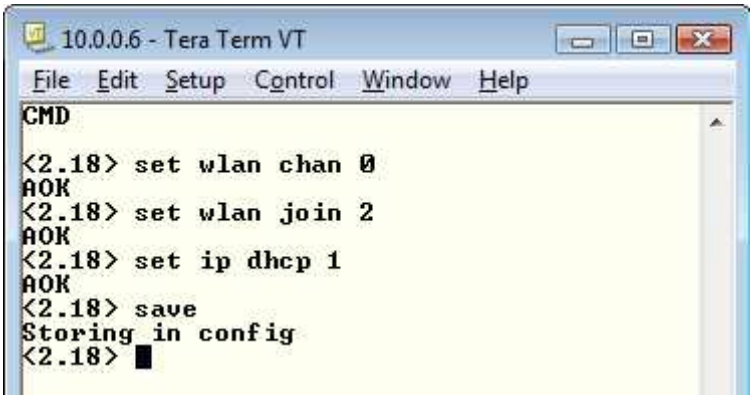


Figure 5.12 Programming Auto-association Commands.

Since in this project the sleep mode will not be implemented, the auto sleep configuration must be disabled, this is done by using the (**set sys autosleep 0)** and the sleep timer must be zero, this is done using the command (**set sys sleep 0**). In this case the transceiver will always be in the active mode.

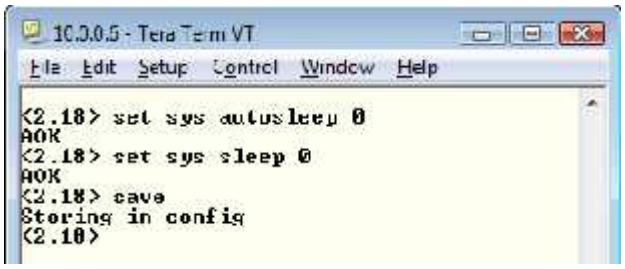


Figure 5.13 Disabling Autosleep Mode.

The (**set sys autoconn**) command is used in TCP mode to determine the value of the auto connect timer. The value that follows the command determines how often the transceiver should try to connect to the stored remote host (web server), if the value is "**1**", the transceiver will only connect once upon powering up. If the value is "**2**", which is the value that is used in this project, or more, then if the connection is closed the auto connect command will make the transceiver open the connection after the number of seconds defined.

To configure the web server IP address and Port number of the connection these two commands were used, (**set ip host WEB_SERVER_IP**) and (**set ip remote 3000**).

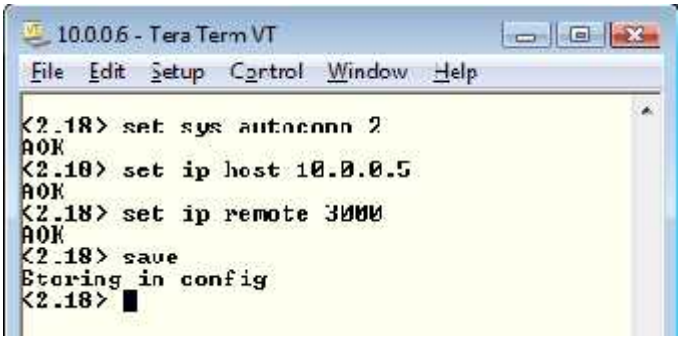


Figure 5.14 Programming Web Server Connection Commands.

The WiFly GSX has an option that if the transceiver has been idle for a specific time, the connection will be disconnected. The command (**set comm idle**) sets the idle disconnect timer. If the command was followed by "**0**", the idle disconnect will be disabled, as the case in this system.

The default case for the transceiver is to receive a flush size value before forwarding, this flush size value is the number of bytes that should be received on the UART of the transceiver before starting to forward to the TCP connection, this value is determined by the command (**set comm size**), the default value for this command is 64 bytes, and the maximum is 1420 bytes. If this command can be followed by "**0**" or "**1**" as the case in this system, this case makes the transceiver start forwarding immediately without the need of the flush size.

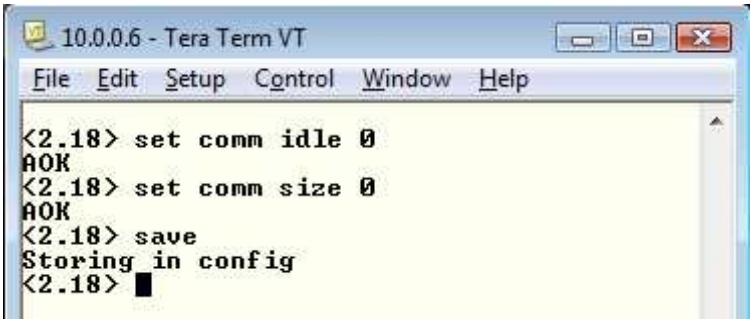


Figure 5.15 Programming Transferring of Data commands.

For a successful serial communication between the transceiver's UART and the PIC microcontroller USART, they must have the same baud rate. The command that sets the transceiver's baud rate is (**set uart baud**) followed by the value of the baud rate that is the same in both the transceiver and the PIC microcontroller which in this system is (**38400**) bytes per second (Bps).

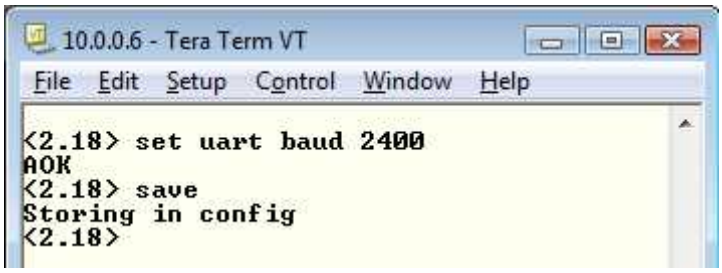


Figure 5.16 Programming UART Sittings.

For each command to be successfully accepted by the transceiver, a (**save**) command must be used after any new commands. If the commands were successfully handled by the transceiver, it will reply with (**AOK**), if any error in the syntax of the command, the transceiver will reply with (**ERR: ?- Cmd**) , and (ERR: Bad Args) in case of error in the parameters.

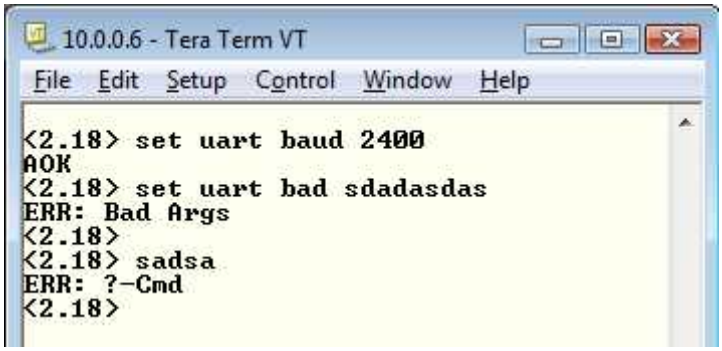


Figure 5.17 Errors in Commands.

All the commands that were mentioned above are from the **set** category of the commands. There are another four categories, **get**, **status**, **action**, and **IO** commands.

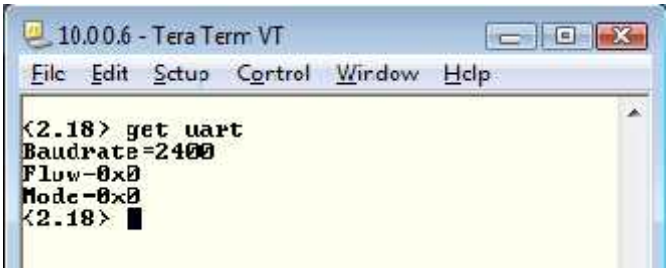


Figure 5.18 A GET Command Example.

5.2.5 PIC Microcontroller Programming

The PIC microcontroller is part that requires much of the programming, because it combines all the hardware of the robot system, the interfacing with them, communicating with them, and performs processing needed for the robot movement, the intelligent system to avoid obstacles.

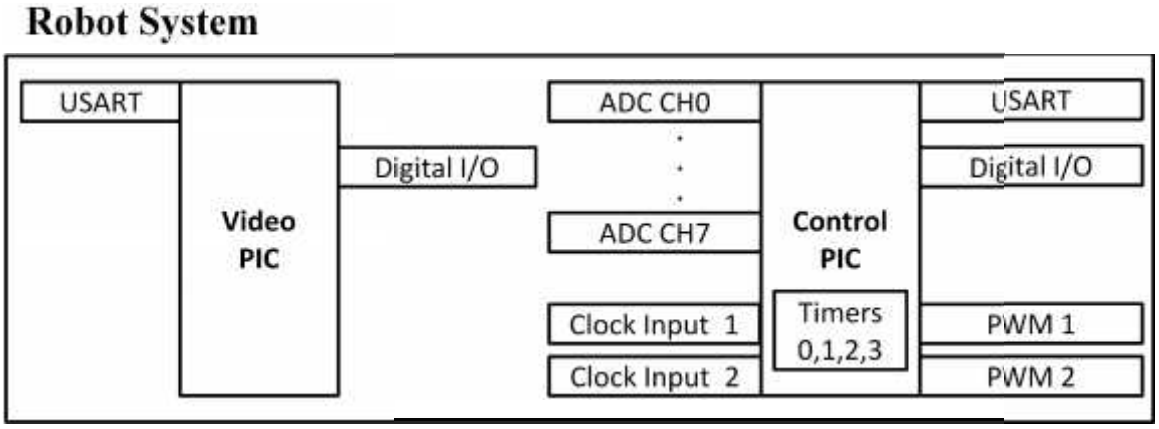


Figure 5.19 PIC Microcontroller Components that requires programming.

- **USART Programming**

USART stands for Universal Synchronous Asynchronous Receive Transfer, which is a very important component in the system, because it is responsible for the serial communications, in which is used between both the WiFly transceiver and the camera with the PIC.

The first step in programming the USART is configuring the **Baud rate** to make sure that the data is sent and received correctly. In order to configure the baud rate correctly the value of the SPBRG register must be calculated. The SPBRG is the baud rate generator.

Formulas for SPBRG

SPBRG = (Fosc/(16 x Baud rate)) -1, in the case of BRGH=1

Fosc: Frequency of Oscillator of the PIC microcontroller which is 8MHz.

BRGH: High Baud Rate Select bit, 1 = High speed 0 = Low speed, which is (1) in this system

Baud rate implemented in this system is 38400 Bps

SPBRG = $(8x10^6/(16 \times 38400)) -1= 12$

The second step is to enable the interrupt on receive option **(USART_RX_INT)**, so that whenever the PIC receives commands from the web server, it decodes that command and perform the requested operation.

More USART programming techniques and concepts are fully described with the code in Appendix D.

- ## PWM Programming

The Pulse Width Modulation (PWM) is used in this system to drive the two DC motors that is responsible for the robot movement. Two motors requires two PWM outputs which are provided by the PIC18F4550.

The frequency of PWM must be calculated carefully to match the DC motors frequency, which is 20KHz for both of the motors.

PWM period formula:

Period = ((1/Frequency)/4x(1/Fosc)xTimer2 Prescaler) -1

Period = $((1/20x10^3)/4x(1/8x10^6)x1)-1$

Period = 99;

After calculating the frequency of the output PWM signal, it is essential to determine the value of the Duty Cycle for each signal, this value determines the value of the voltage of the PWM signal.

Duty Cycle Formula:

Duty Cycle = (Required Voltage/ Source Voltage) x 4 x PWM Period.

The duty cycle value differs in this system by the function it is needed for, for example if the motors are required to move in full speed, this requires the full scale of the voltage signal from the PIC which is 5V, this can be accomplished by a value of 396 for the duty cycle.

Much more details are mentioned with the code in Appendix D.

- ## ADC Programming

The programming of the Analog-to-Digital converter module, is very simple but essential in this system, the ultrasonic sensors are connected to this module, they output a voltage signal its value determines the range of the closest obstacle.

To program the ADC to read the range, it is important to determine the Least Significant value of the ADC, which is the value of voltage that is detected by the module.

LSB= Input Voltage/$2^{\text{ADC-reselution}}$. ADC-resolution = 10 bit

LSB = 5V/1024= 4.8828125mV

Then to calculate the value of the ADC that represents the range in inches the sensors sensitivity is important. So, the distance of the obstacle formula is:

Value of ADC (integer) = Distance (inches) x Sensitivity / LSB;

Sensors sensitivity: A supply of 5V yields ~9.8mV/in.

For example if the robot is needed to detect obstacles close than 30cm. the value of the ADC will be:

ADC = ((**30cm**/2.5)(inches) x 9.8)/ 4.8828125 = 24.

More techniques to deal with eight analog inputs one for each sensor will be fully discussed in Appendix D.

- ## Timers Programming

Timers are very helpful modules in the PIC, they execute the required action in parallel with the PIC execution, this was needed because the controlling of the robot movement must work in parallel with sensors readings, and USART readings.

There are four timers in the PIC18F4550, Timer0, Timer1, Timer2, and Timer3. All of them were needed in the code to implement the system functions. In this section a brief description of each timer will be discussed, why it was needed, and how it was programmed.

- Timer0: This timer was configured to read the encoder of the left wheel, the output voltage of the encoder will act as the input clock of the timer, making the timer working as a counter, reading the number of pulses from the encoder and calculating the relationship between the number of pulses and actual distance moved is described in this formula:

  Number of Pulses = (Distance x **66)/ π**

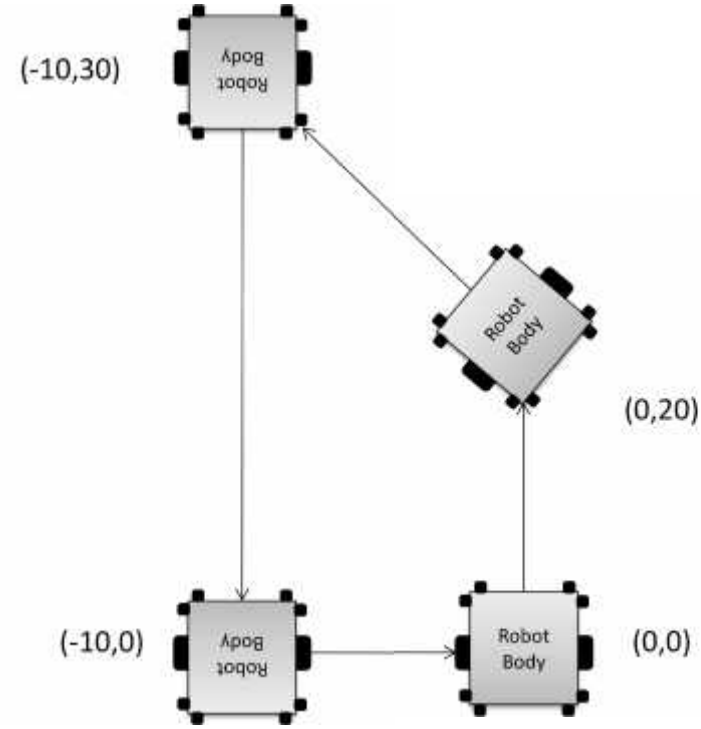
  For example: if the robot is ordered to move 1m the timer0 reading must reach 2102.
- Timer1: Same as Timer0, but it is connected with the right wheel encoder.
- Timer2: Used for the PWM operation, working as the counter of the period.
- Timer3: Used also for the PWM operation, working as the counter of the Duty cycle.

More details and codes are in Appendix D.

- **Intelligent System Programming**

To implement an intelligent capability in the robot, the robot is programmed to calculate the coordinates of its position at every point. At system start-up the robots coordinates are (X=0, Y=0) and at every step the robot calculates the coordinates of new position through a programmed code in the control PIC. Navigating the robot according to coordinates is an option for the system user. The user can send the coordinates to the robot to go to and the robot is expected to move to those coordinates. The unit of the coordinates is in square centimeter, so a transition from point (X=0, Y=0) to the point (X=0, Y=1) is a 1 cm step forward.

When the robot reaches its destination, it stores the coordinates of that point into a two dimensional array containing the path coordinates as shown in figure 5.20. The robot also stores the coordinates when it stops after detecting an obstacle, so it can use that coordinates to calculate the path to avoid that obstacle and try to reach the destination point.



Path Two-Dimensional Array

X	Y
0	0
0	20
-10	30
-10	0
0	0

Figure 5.20 Robots Path Storing Technique

The robot is also programmed to avoid obstacles by detecting them using eight ultrasonic sensors, if the robot was ordered to move to some point and in its path to that point an obstacle was detected, the robot is programmed to avoid that obstacle and try to return to its original path.

The C code to implement this system is under development and expected to be finished at the end of this semester.

5.3 Web Server Configurations

To configure the web server, a number of applications must be installed, and a few configurations must be chosen. In this section an explanation for the steps to configure the web server will be described.

At first the server must have an operating system installed and ready for applications to be installed.

Step One: Install XAMPP software version 1.7.2 or above.

After installing the XAMPP software, run the XAMPP Control Panel and enable the Apache server, and MySQL, as shown in figure 5.21. For the web site to operate, this step must be done.

Step Two: Install the web site files (pages, connections, databases, etc) in the XAMPP root directory, which is normally:

C:\xampp\htdocs\(Name Of the Website folder)

The web site files must include the **Published Web Page of the Robot Control Software VI,** see figure 5.22.

Step Three: Install LabVIEW software version 7.1 or later.

After installing the LabVIEW software, the LabVIEW web server must be enabled. This is done as the following, figure 5.23:

In the software menu >> **Tools** >> **Options** >> from the drop list Web **Server: Configuration** >> **Enable Web Server** >> **Choosing the Root Directory** the same as web site folder >> choosing the **HTML port number 6060** to avoid conflict with the XAMPP software which occupies the HTML port number 80.

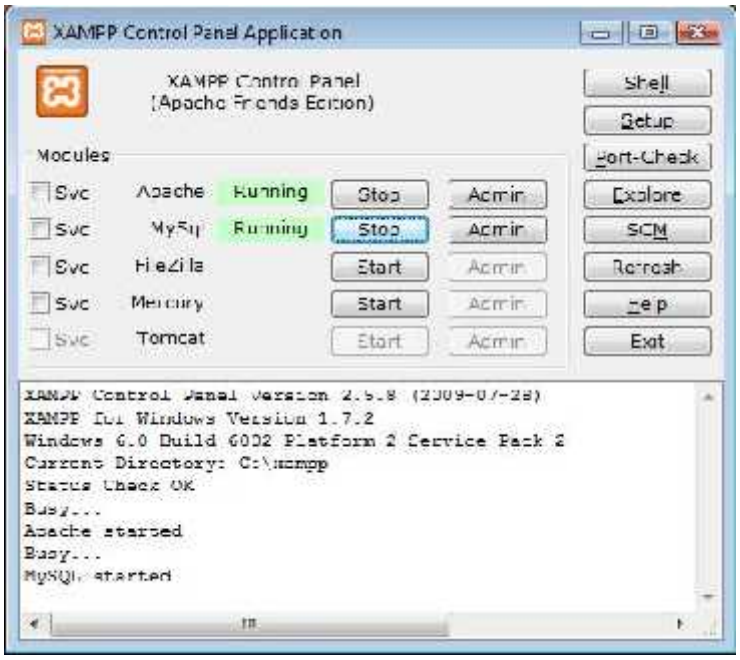


Figure 5.21 XAMPP Control Panel

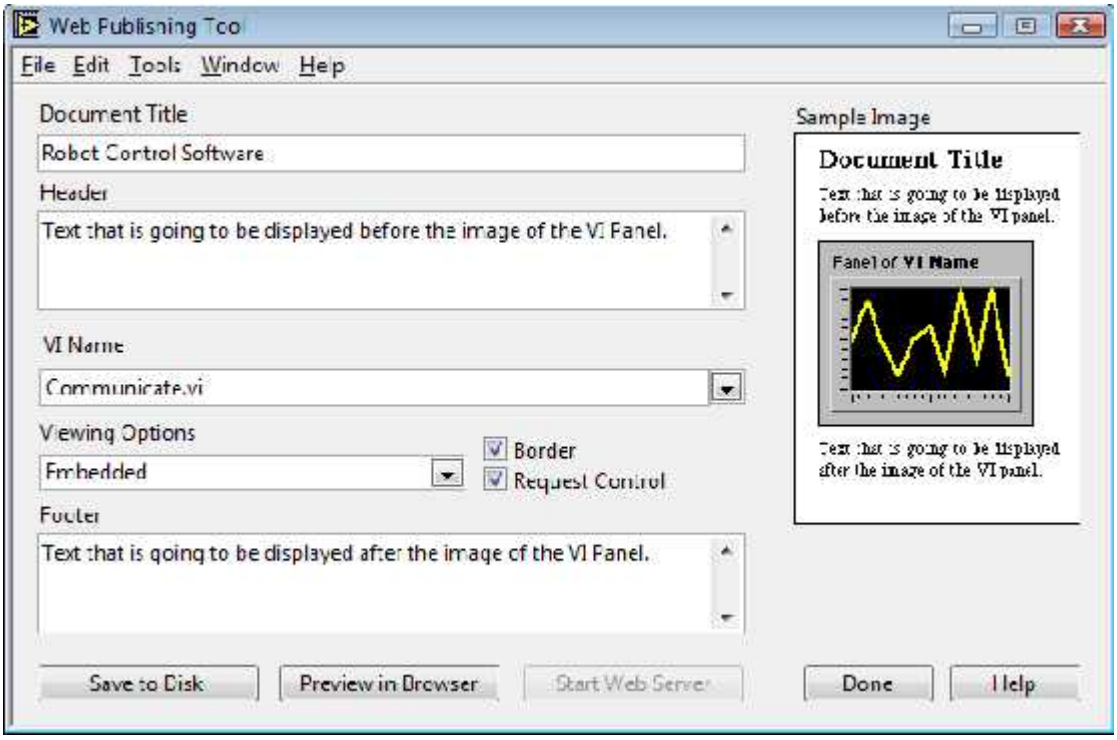


Figure 5.22 Web Publishing Tool in LabVIEW 7.1.

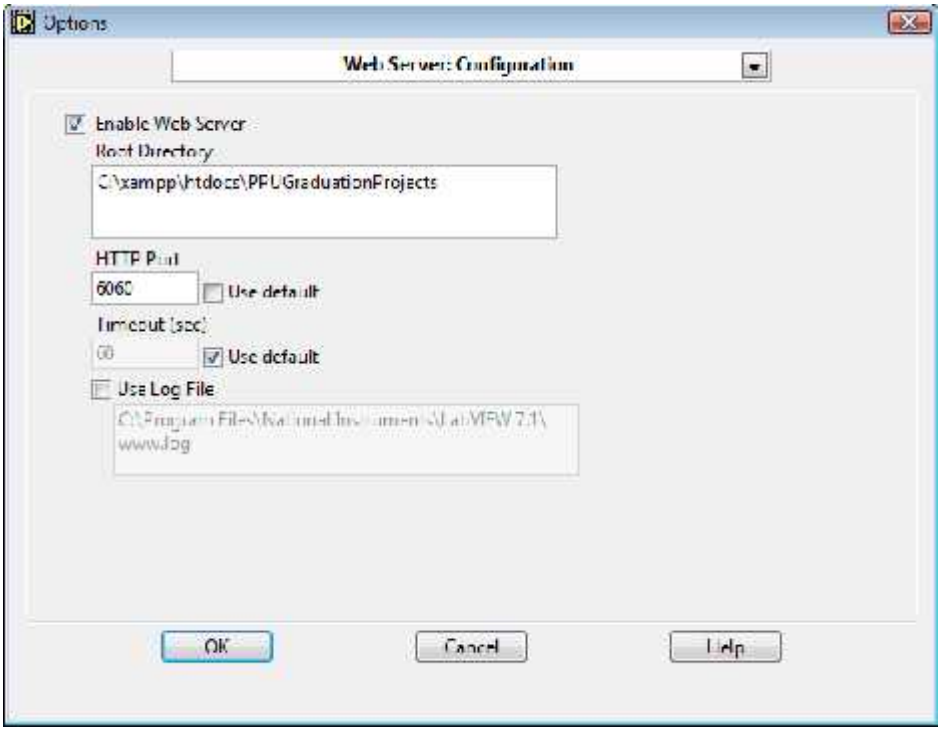


Figure 5.23 LabVIEW Web Server Configuration.

Step Four: To prepare the web server to be accessed by users, all of the above applications must be running, and also the Robot Control Software must be opened.

5.4 Web Site Design and Implementation.

As shown in figure 5.24, that shows a flowchart of the steps the user must do to use the Robot Control Software to control the robot. The user must first log into his account in the web site. If the authentication completes, when the user go to the robot controller webpage, if another user is using the software, a message will appear to the user that the software is busy. But if the user is the first to use the software, the permission will be granted to him and he can use the software and try to connect with the robot.

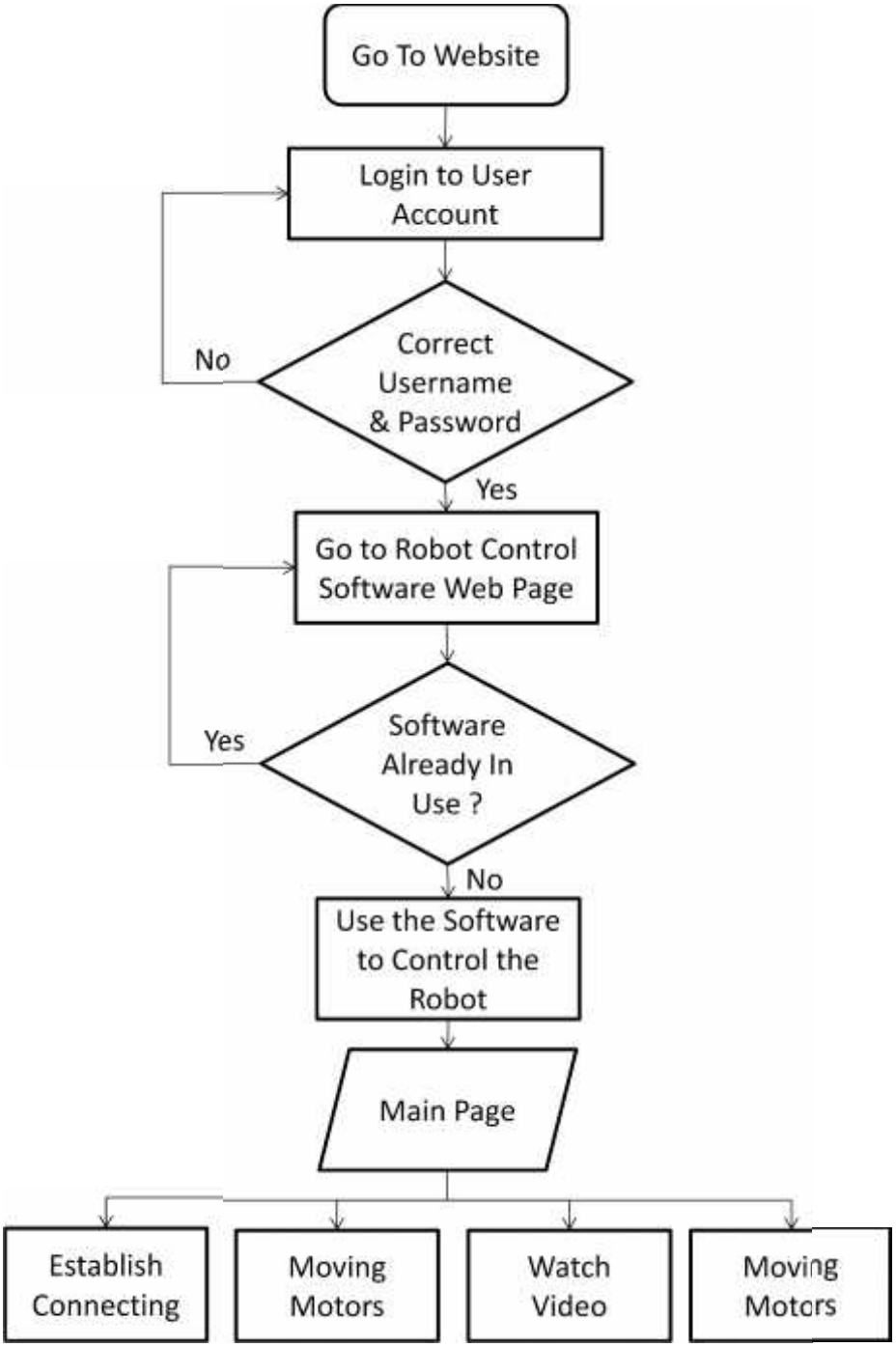


Figure 5.24 Web Site Design Flow Chart.

Chapter Six

System Testing

6.1 Introduction
6.2 Testing Scheduling
6.3 Testing Procedure

Chapter Six

System Testing

6.1. Introduction

The testing steps are very important to the system. The system after all the works has been done, was placed under spot to see if it is working as expected, and to find out the mistakes, problems.

6.2 Testing Scheduling

The table below shows testing time schedule.

Table 6.1 Testing Schedule

Days / Testing Process	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Unit Testing	■	■	■	■	■	■								
Sub-System Testing						■	■	■	■	■				
System Testing											■	■	■	■
Acceptance Testing													■	■

6.3 Testing Procedures

The system testing procedures are shown in table 6.2.

Table 6.2 Testing procedures.

Unit testing:

To implement such testing, each unit must be tested, so each task of the project would be tested individually. The WiFly GSX transceiver was tested alone, the PIC microcontroller, ITM-C-328 serial camera, DC and stepper motors each was tested separately. Additionally, the connecting cables and all the pins were tested. Finally, testing each part of the robot's physical construction was done.

The software components also were tested. The web server, the LabVIEW web server, the XAMPP web server, and the robot control software, the connection of all of them with the internet. The consistency of the web site and the flow of data in it were tested.

Sub-System Testing:

The main aim of this testing part is to test the main operations in the system. There are six main operations in the system. The first one is the robot movement. The second operation is the camera base horizontal movement. The third operation is exchanging of data between the WiFly transceiver and the PIC microcontroller. The fourth one is sending the video camera streaming to the PIC. The fifth operation is the sensors readings transfer to the PIC. The last one is the connection between the WiFly transceiver and the web server. All the operations mentioned before fulfilled the expectations, except the video streaming, it needs more time to be fulfilled.

The major software components were tested also, The connection between the web server and the transceiver using TCP connections function, the web site security and user authentication, the robot controlling. What is left to be tested is capturing video streaming, and recording the video.

System Testing:

This section is to test system work. It should be tested by using a complete process of connecting the electrical system components, the robot's physical components, moving the robot, rotating the camera, capturing video and avoiding obstacles. By implementing this process, the system works very well except the video streaming which is not finished yet.

The software component in the system was tested. User authentication, starting the robot control software, configuring the system as required, controlling the robot and the camera base movement, and closing the system. Only viewing video streaming has not been tested yet.

Acceptance testing:

In this section, the system will be tested to see if it meets the requirements the system was built for, and at this point the system did meet the needed task of the system. Except the video streaming part.

6.4 Testing Strategies

This part of the project shows the system Testing Strategies.

6.4.1 Black Box Testing

This type of testing depends on suggesting specific problems, starting the system and then checking the behavior of the system in these suggested problems.

Table 6.3 Black Box Testing.

Inputs	Expected Values	Actual Values	Notes
Client running XP, Vista, 7 Windows operating systems, or Linux.	The system should work properly.	The system worked properly.	Not recommended to operate using Unix operating system, (errors in .dll files)
Client using Chrome browser.	The system should work properly.	Chrome browser did not support LabVIEW Run-Time Engine 7.1.	The system was tested using Internet Explorer 8.0 and worked properly.
Invalid user login information.	Access denied.	Access denied.	The user will be redirected to the Error Login Page.
Two users login to the Robot Control Software at the same time.	Allow the first one, and deny the other.	Grant control to the first user, and show a message to the second user that the control is already granted to another user.	The second user will see a snap shot of the Control Software.
The robot is not turned on.	Failed attempt to establish connection.	Failed attempt to establish connection.	A message will appear to the user.
Robot Shutdown, Access Point shutdown.	Connection with the control software should be disconnected.	Connection with the control software will be disconnected.	A message to the user will be displayed.
Web server shutdown.	The robot should operate in the idle mode*, and the user should not be able to browse the web site.	The robot will operate in the idle mode, and the user will not be able to browse the web site.	The robot can be controlled in ad-hoc mode
No Access Points were found, or no association or authentication with Access point.	Failed attempt to establish connection, and the robot should operate in the idle mode.	Failed attempt to establish connection and the robot will operate in the idle mode.	If the network has a security constrains, the transceiver could be programmed to associate with that network using SSID, the PASS Key, or any other security parameters
An attempt to connect with the WiFly transceiver without web server or not running in ad-hoc mode, using invalid password.	The WiFly transceiver should deny the connection.	The WiFly transceiver denies the connection.	None

The user sends two or more consecutive commands.	The robot should execute them in order.	If the robot started executing the first one, and the second one arrived, the robot will stop executing the first and execute the second.	None
User disconnect	Robot should return to operate in idle mode.	Robot returned to operate in idle mode.	None

* Idle Mode: when the robot is not connected with the web server and not running in ad-hoc mode, it will wait for a remote connection or to be operated in the ad-hoc mode by the ad-hoc mode switch.

6.4.2 White Box Testing

This test will be done according to the critical paths on the flow chart. A value will be expected, and then an observation will occur to see the real value.

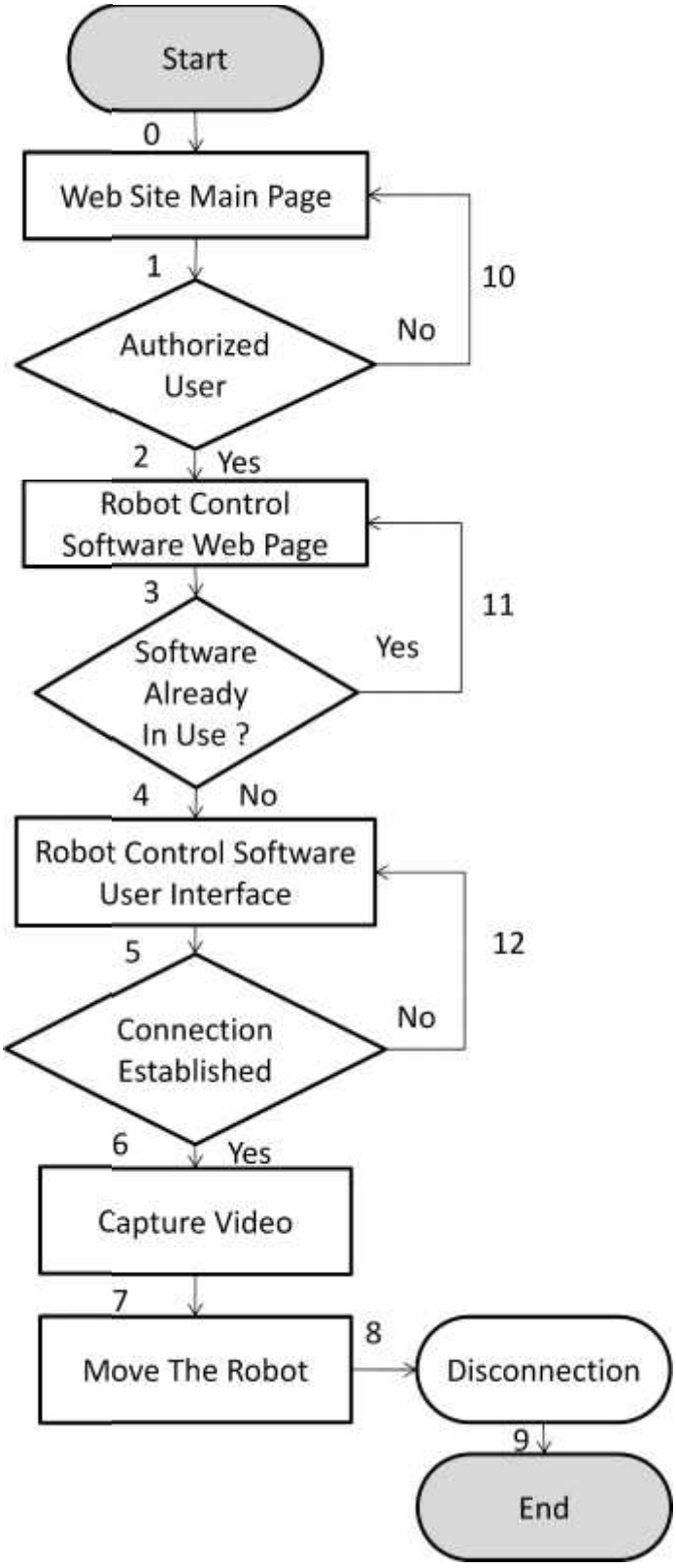


Figure 6.1 White Box Testing.

Table 6.4 White Box Testing.

Branch	Inputs	Expected Values	Actual Values
0,1	Web Server Running.	User Browse Properly.	User Browse Properly.
0,1,10	Unauthorized user.	Redirected to login page.	User redirected to login page.
0,1,2	Authorized user.	Enter Robot Control Software web page.	User entered Robot Control Software web page.
0,1,2,3,11	Software already in use.	Deny access to robot control software.	Show a message to the user that the control is already granted to another user.
0,1,2,3	Software not in use.	Grant control to the user.	Control granted to the user.
0,1,2,3,4,5,12	Failed attempt to establish connection.	A message will appear to the user.	A message appears to the user.
0,1,2,3,4,5,6	Capture Video.	Receive video stream.	Not yet accomplished.
0,1,2,3,4,5,6,7	Move the motors.	Robot Respond to commands.	Robot Responded to commands.
0,1,2,3,4,5,6,7,8	User disconnects.	Drop connection.	Connection dropped.

Chapter Seven

Future Work

7.1 Future Work

Chapter Seven

Future Work

7.1 Future Work

In this project, there are some ideas that could be done or added to improve its performance, or add some capabilities, some techniques that are efficient and meet worldwide needs. Some of these ideas are mentioned below:

- System could be improved to handle number of robots, controlled by a number of users at the same time. This could be accomplished by updating the robot control software for that manner by listening to a number of ports in which each robot uses.
- The robot could be improved apply handover capability between the networks. This could be applied by writing a certain code in the PIC to send commands to the WiFly transceiver serially, to scan for networks periodically and force it to associate with the strongest taking into consideration the Ping Pong effect.
- A special controller device with LCD and keypad… etc, could be designed, and used to replace the client computer.
- A new capability could be added to the system, to enable the user to listen to the voices around the robot, using a microphone, and enable the user to speak through a speaker on the robot.
- A memory could be added to the robot system to store videos on board in case of connection dropped. This could be done through serial connection between the memory and the SPI module in the PIC microcontroller.
- The robot could be programmed to receive commands via voice recognition, this could be applied by a voice recognition program on the robot control software, converting voice commands into text commands to send them over TCP connection to the robot.
- An improvement to the video quality could be applied, by using a high resolution camera, or any other microcontroller or microprocessor with higher frequency.
- The two PIC microcontrollers could be replaced with a single microcontroller or microprocessor that supports multitasking, and high performance.
- More commands and operations could be added to the robot, like searching for objects (using image processing or certain sensors).

References

[1] Alaa Al Tamimi, Bashar Al Takrouri, Internet- Based Robot Control System Using Matlab, Palestine Polytechnic University, June 2004

[2] Je-Goon Ryu, Hyeon-Min Shim, Se-Kee Kil, Eung-Hyuk Lee, Heung-Ho Choi, Seung-Hong Hong, Design and Implementation of Real-Time Security Guard Robot using CDMA Networking. http://www.roboticslab.co.kr/...../CI2006-1.pdf

[3] http://en.wikipedia.org , Mobile Robot.

[4] Dogan Ibrahim, Advanced PIC Microcontroller Projects in C, Elsevier Publishing, March 2008.

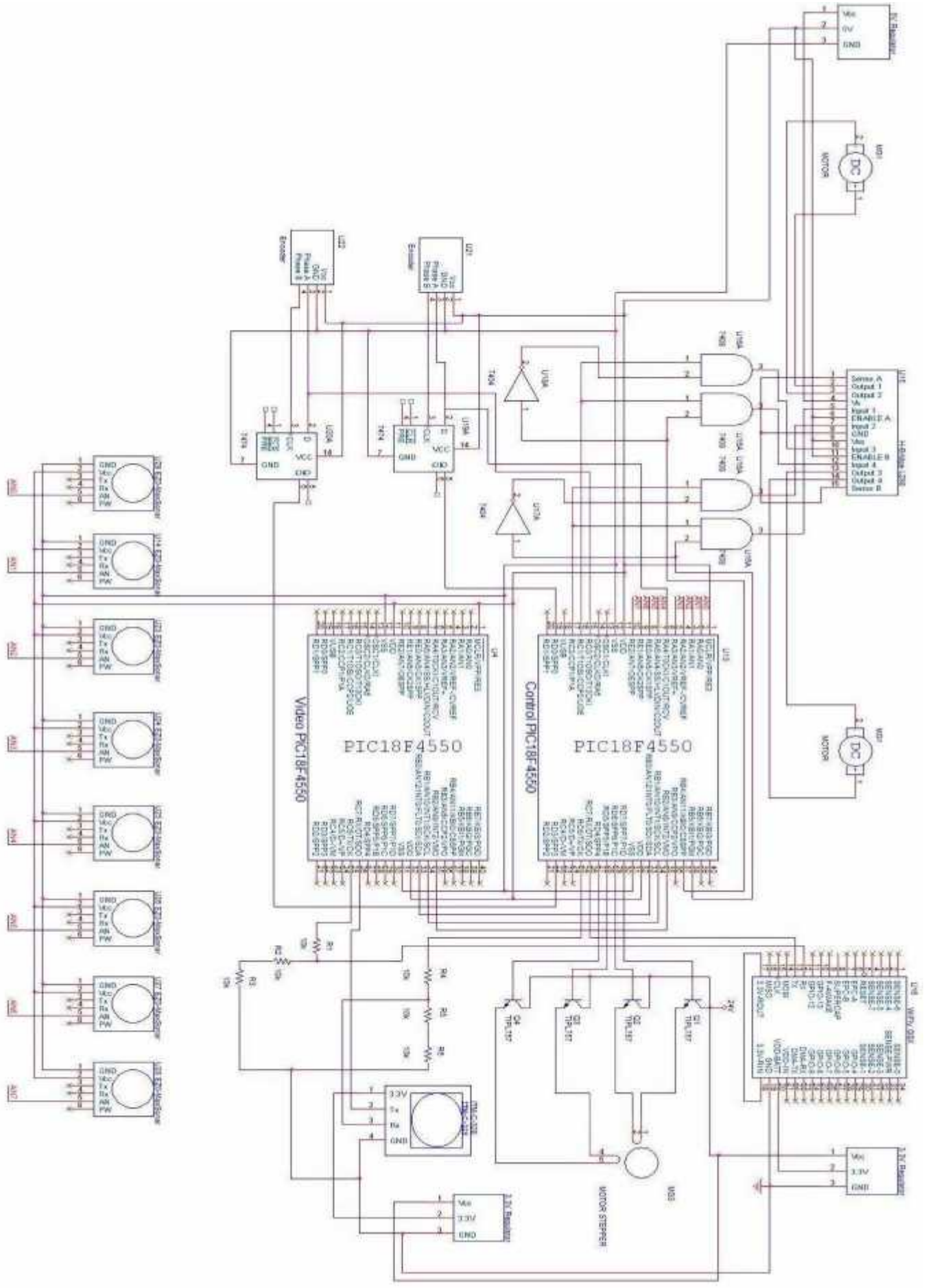
[5] http://en.wikipedia.org, Microcontroller.

[6] http://en.wikipedia.org, PIC Microcontroller.

[7] Vehicle Dynamic Textbox (reference)

Appendix A

- **Robots Board Connection**


Video PIC18F4550
PIC18F4550
Control PIC18F4550
PIC18F4550
MOTOR
MOTOR
MOTOR STEPPER
TIP127
TIP127
TIP127
TIP127

Appendix B

- **DC Motor RS-445PA**
- **Stepper Motor**

RS-445PA/PD

MABUCHI MOTOR
Carbon-brush motors

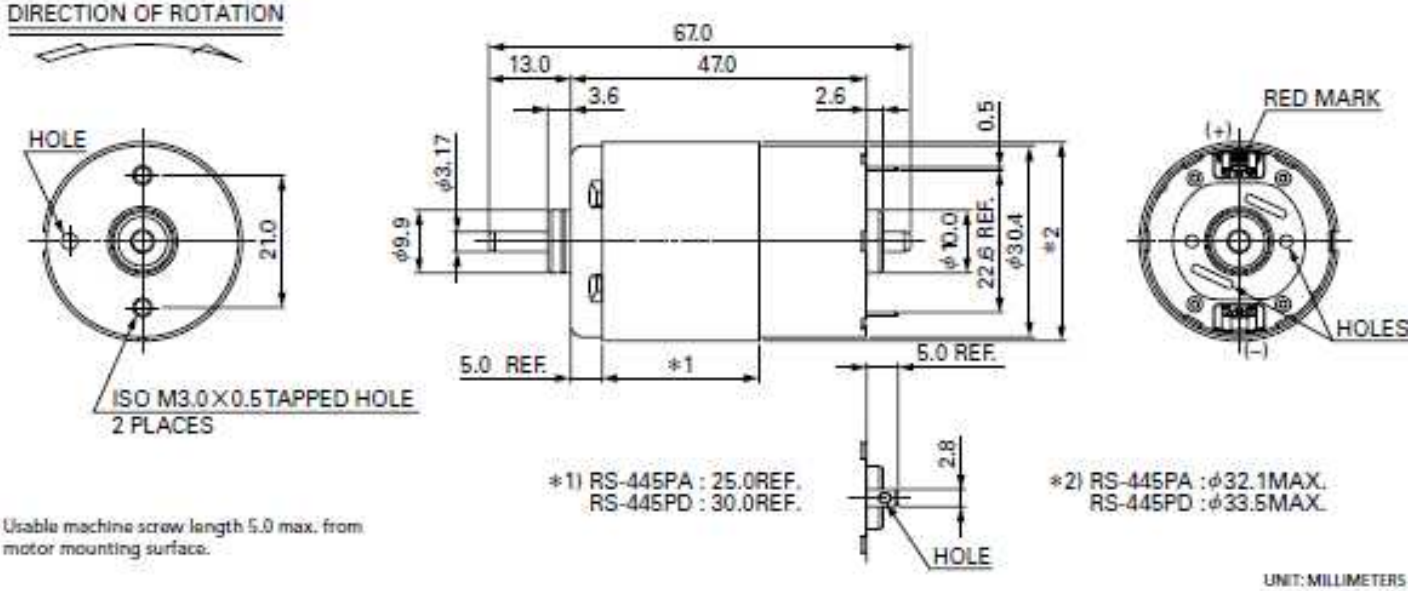
OUTPUT : 3.0W~55W (APPROX) WEIGHT : 127g (APPROX)

Typical Applications **Precision Instruments** : Printer

MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
			r/min	A	r/min	A	mN-m	g-cm	W	mN-m	g-cm	A
RS-445PA-14233	12~42	*42V CONSTANT	6500	0.060	5410	0.30	13.7	140	7.78	81.8	834	1.47
RS-445PA-15200	12~42	*42V CONSTANT	7600	0.067	6420	0.36	14.6	148	9.77	93.7	955	1.98
RS-445PD-16175	12~42	*42V CONSTANT	7400	0.055	6410	0.35	15.3	156	10.3	114	1162	2.28

*This operating voltage indicates the average or r.m.s. In case of pulse-width modulation (PWM) power supply. Besides, as withstand voltage, 1800V for one second can not be guaranteed.

DIRECTION OF ROTATION



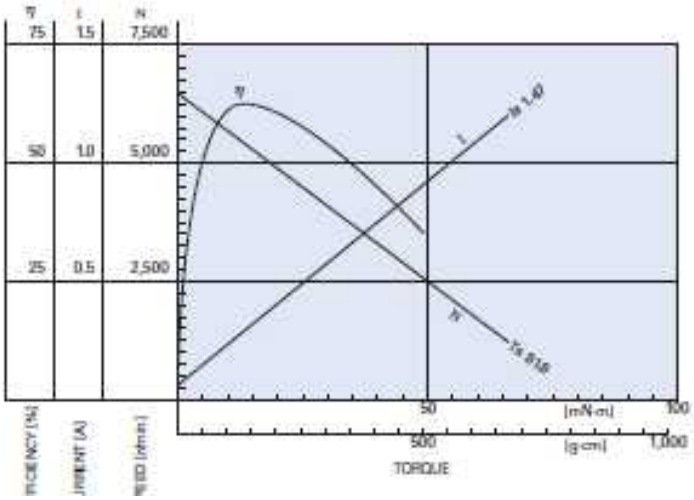
*1) RS-445PA : 25.0REF.
RS-445PD : 30.0REF.

*2) RS-445PA : ϕ32.1MAX.
RS-445PD : ϕ33.5MAX.

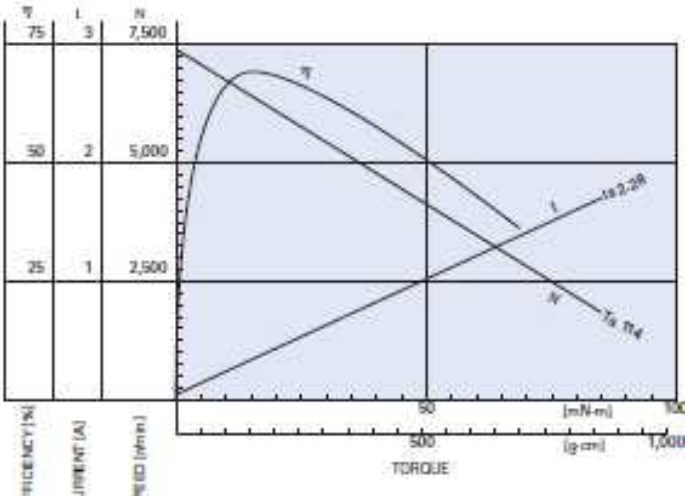
Usable machine screw length 5.0 max. from motor mounting surface.

UNIT: MILLIMETERS

RS-445PA-14233 42.0V



RS-445PD-16175 42.0V



Stepping Motors
M35SP-11NK

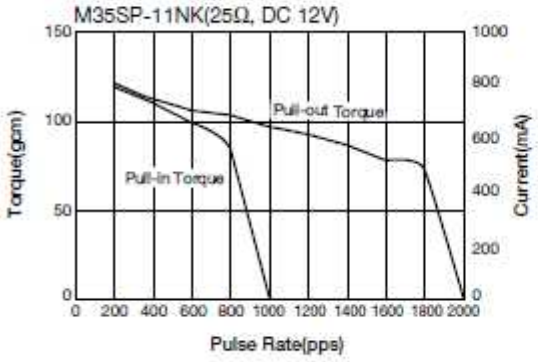
OUTLINE

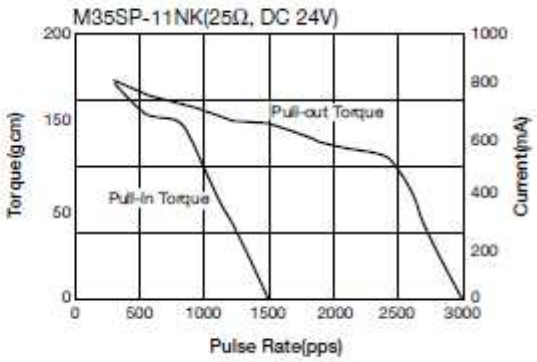
M35SP-11NK, which has the thinnest body among the model series with ø35 diameter, realizes step angle of 3.75°, high resolution and high speed responsiveness.
It is ideal for use in the small electronic equipment of the future.

SPECIFICATIONS

Items	M35SP-11NK	
Rated Voltage	DC 12V	DC 24V
Working Voltage	DC 10.8~13.2V	DC 21.6~26.4V
Rated Current/Phase	200mA(PEAK)	300mA(PEAK)
No. of Phase	2 Phase	2 Phase
Coil DC Resistance	25Ω/phase±7%	25Ω/phase±7%
Step Angle	3.75°/step	3.75°/step
Excitation Method	2-2 Phase excitation(Bipolar driving)	
Insulation Class	Class E insulation	Class E insulation
Holding Torque	14.7mN-m	20mN-m
Pull-out Torque	8.9mN-m/1200pps	12.1mN-m/1800pps
Pull-in Torque	10.8mN-m/400pps	13.8mN-m/600pps
Max. Pull-out Pulse Rate	1900pps	3300pps
Max. Pull-in Pulse Rate	920pps	1250pps

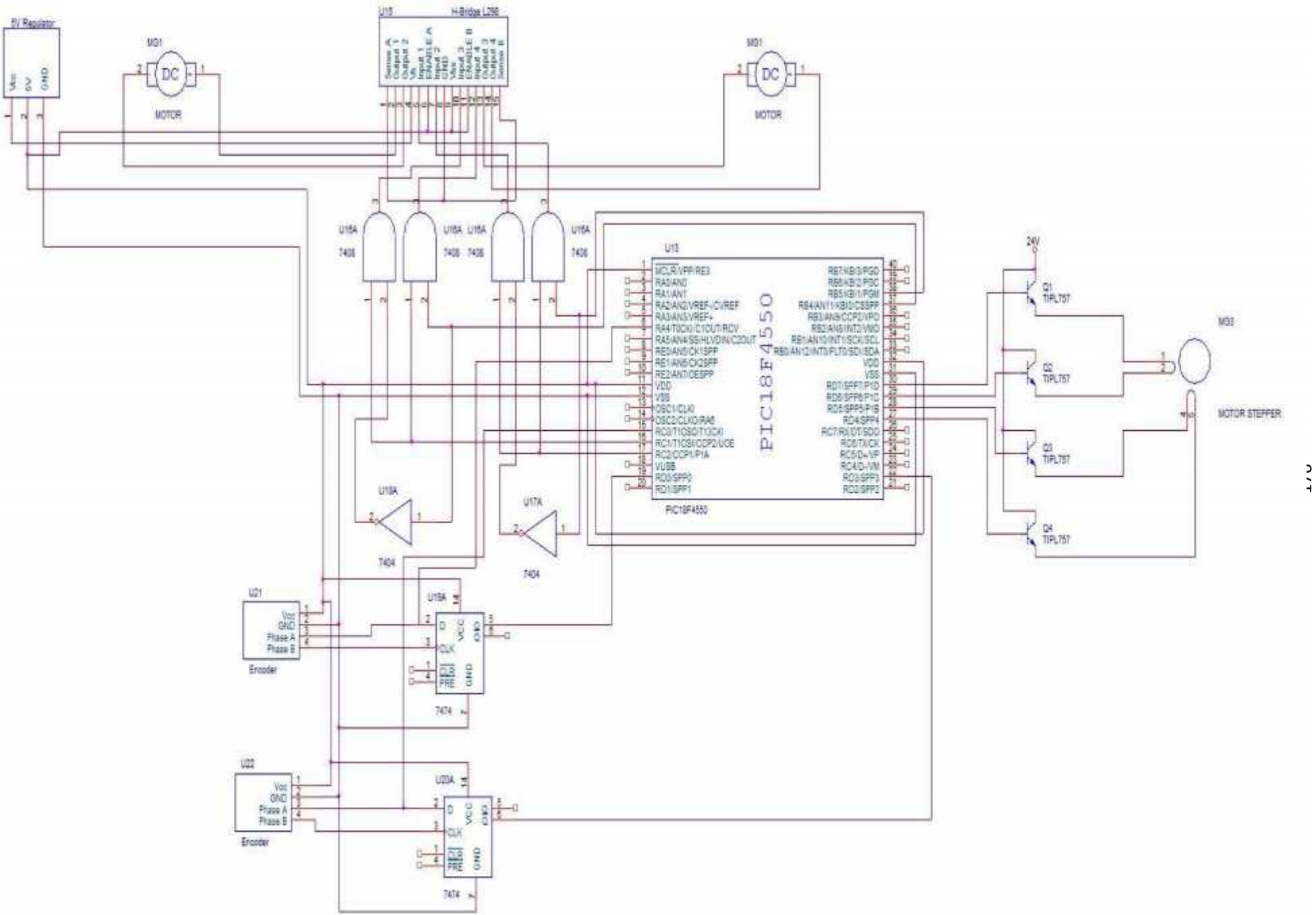
CHARACTERISTICS





Appendix C

- **H-Bridge L298**
- **Drive Circuit Implementation**
- **Logic Circuit Implementation for DC motors**
- **Stepper motor drive circuit Implementation**


5V Regulator
VIN
5V
GND
U15
H-Bridge L298
MG1
DC
MOTOR
MG1
DC
MOTOR
U16A
7408
U16A
7408
U16A
7408
U16A
7408
U13
PIC18F4550
MCLR/VPP/RE3
RA0/AN0
RA1/AN1
RA2/AN2/VREF-/CVREF
RA3/AN3/VREF+
RA4/T0CKI/C1OUT/RCV
RA5/AN4/SS/HLVDIN/C2OUT
RE0/AN5/CK1SPP
RE1/AN6/CK2SPP
RE2/AN7/OESPP
VDD
VSS
OSC1/CLKI
OSC2/CLKO/RA6
RC0/T1OSO/T13CKI
RC1/T1OSI/CCP2/UOE
RC2/CCP1/P1A
VUSB
RD0/SPP0
RD1/SPP1
RB7/KBI3/PGD
RB6/KBI2/PGC
RB5/KBI1/PGM
RB4/AN11/KBI0/CSSPP
RB3/AN9/CCP2/VPO
RB2/AN8/INT2/VMO
RB1/AN10/INT1/SCK/SCL
RB0/AN12/INT0/FLT0/SDI/SDA
VDD
VSS
RD7/SPP7/P1D
RD6/SPP6/P1C
RD5/SPP5/P1B
RD4/SPP4
RC7/RX/DT/SDO
RC6/TX/CK
RC5/D+/VP
RC4/D-/VM
RD3/SPP3
RD2/SPP2
U18A
7404
U17A
7404
24V
Q1
TIPL757
Q2
TIPL757
Q3
TIPL757
Q4
TIPL757
MG3
MOTOR STEPPER
U21
Vcc
GND
Phase A
Phase B
Encoder
U19A
7474
U22
Vcc
GND
Phase A
Phase B
Encoder
U20A
7474

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

Multiwatt15

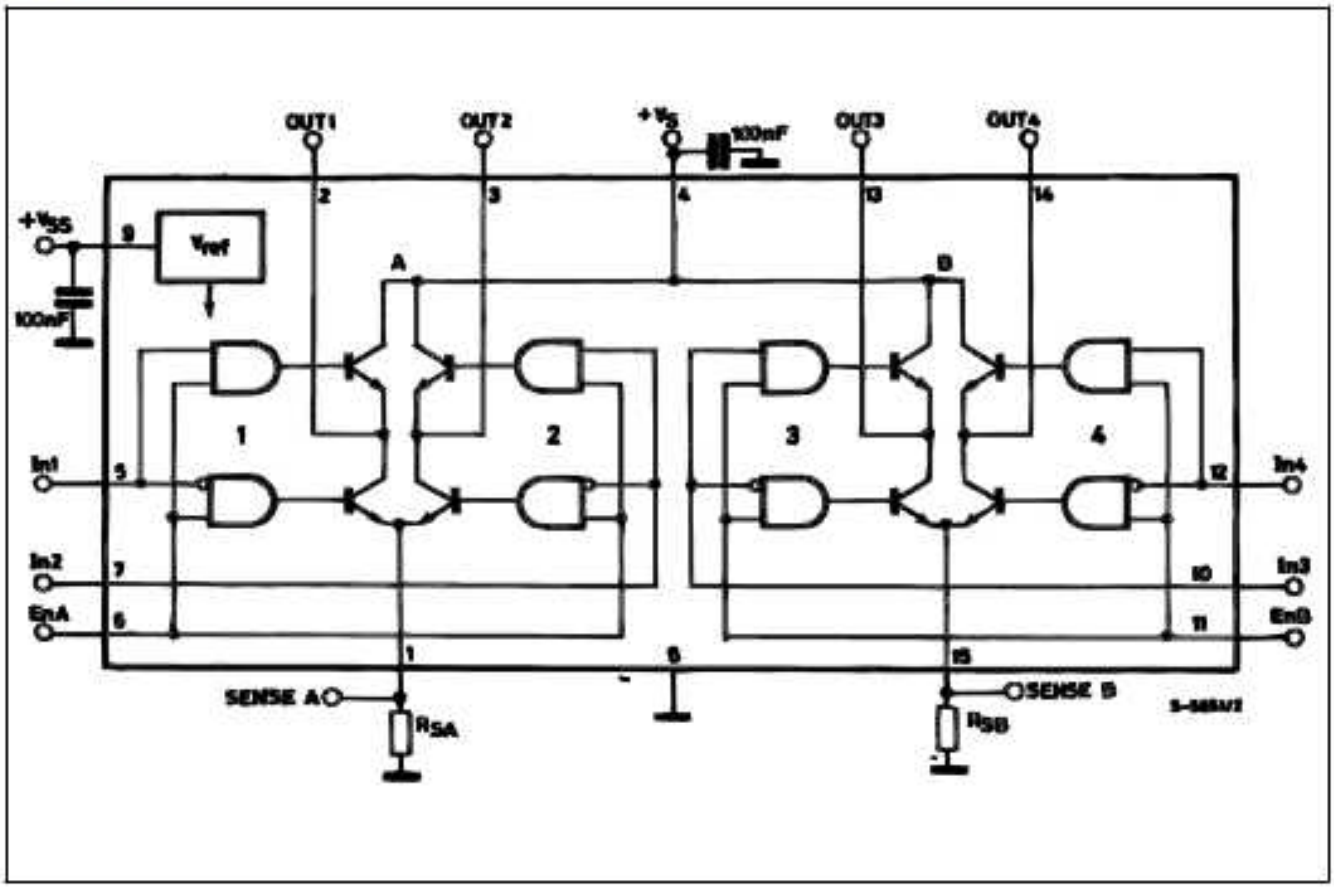
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

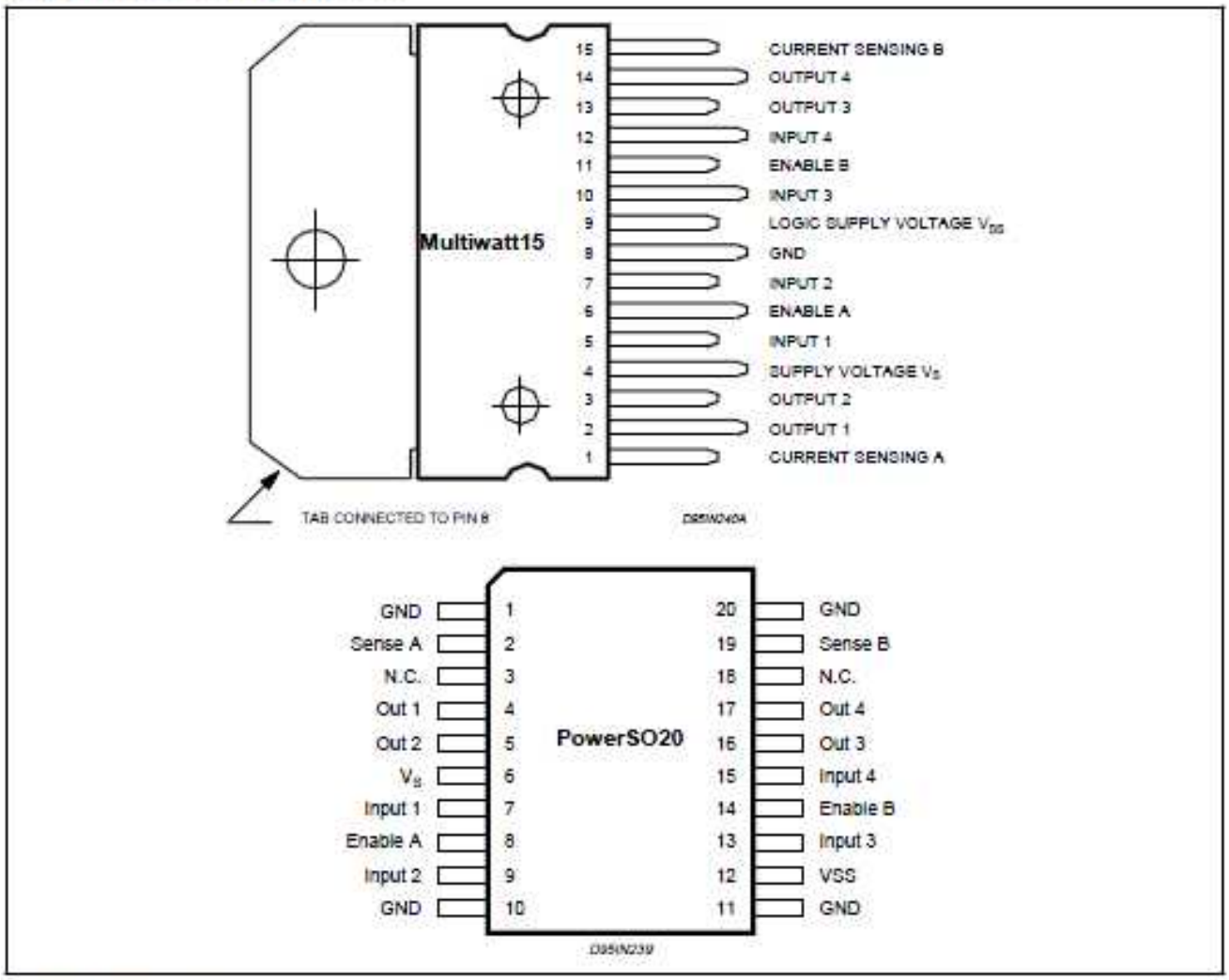
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition		V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)			4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0	V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L	V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0	V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L	V_i = X			6	mA
V_{iL}	Input Low Voltage (pins 5, 7, 10, 12)			–0.3		1.5	V
V_{iH}	Input High Voltage (pins 5, 7, 10, 12)			2.3		VSS	V
I_{iL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L				–10	µA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V			30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)			–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)			2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L				–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V			30	100	µA
$V_{CEsat\ (H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A		0.95	1.35 2	1.7 2.7	V V
$V_{CEsat\ (L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)		0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)		1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)			–1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for $t \leq 50$ µsec; in steady state V_{sens} min ≥ -0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

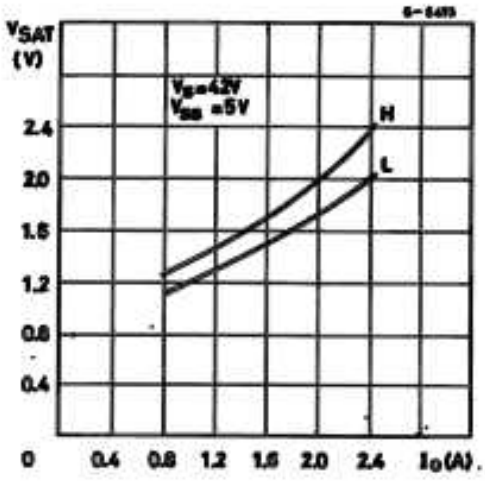
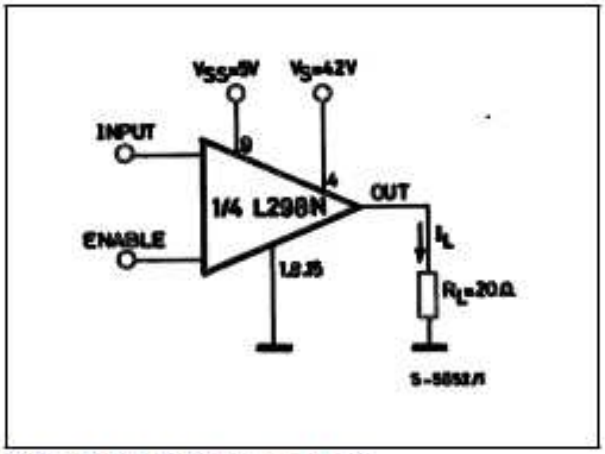


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.

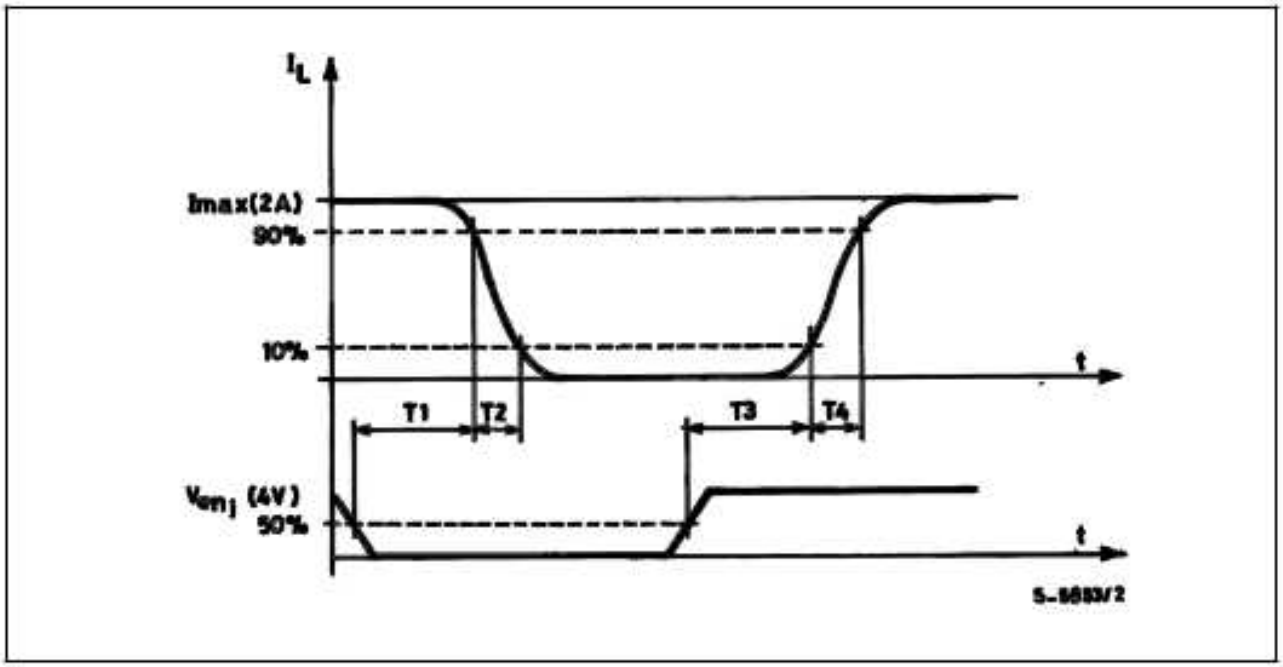
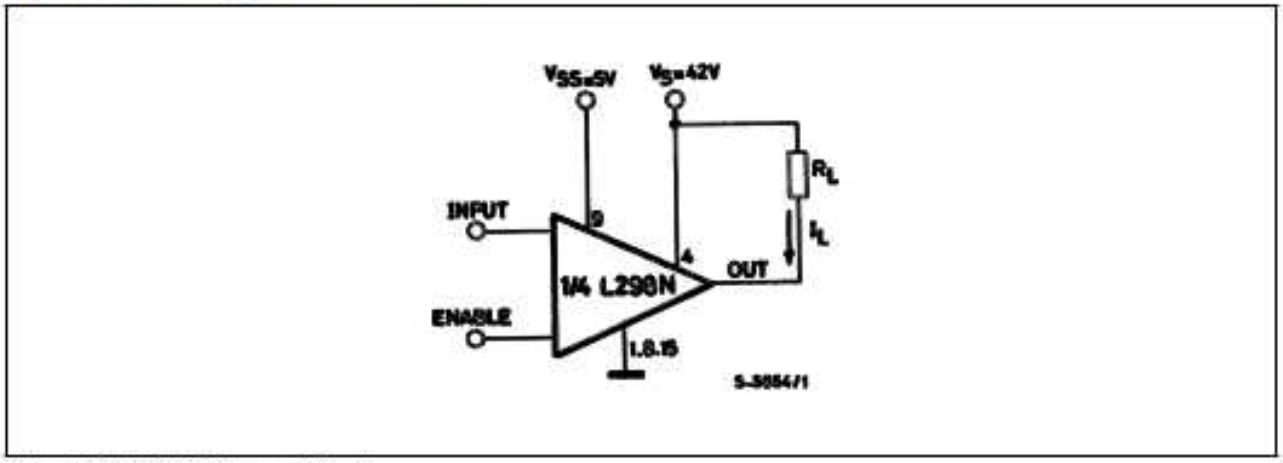


Figure 4 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

Figure 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.

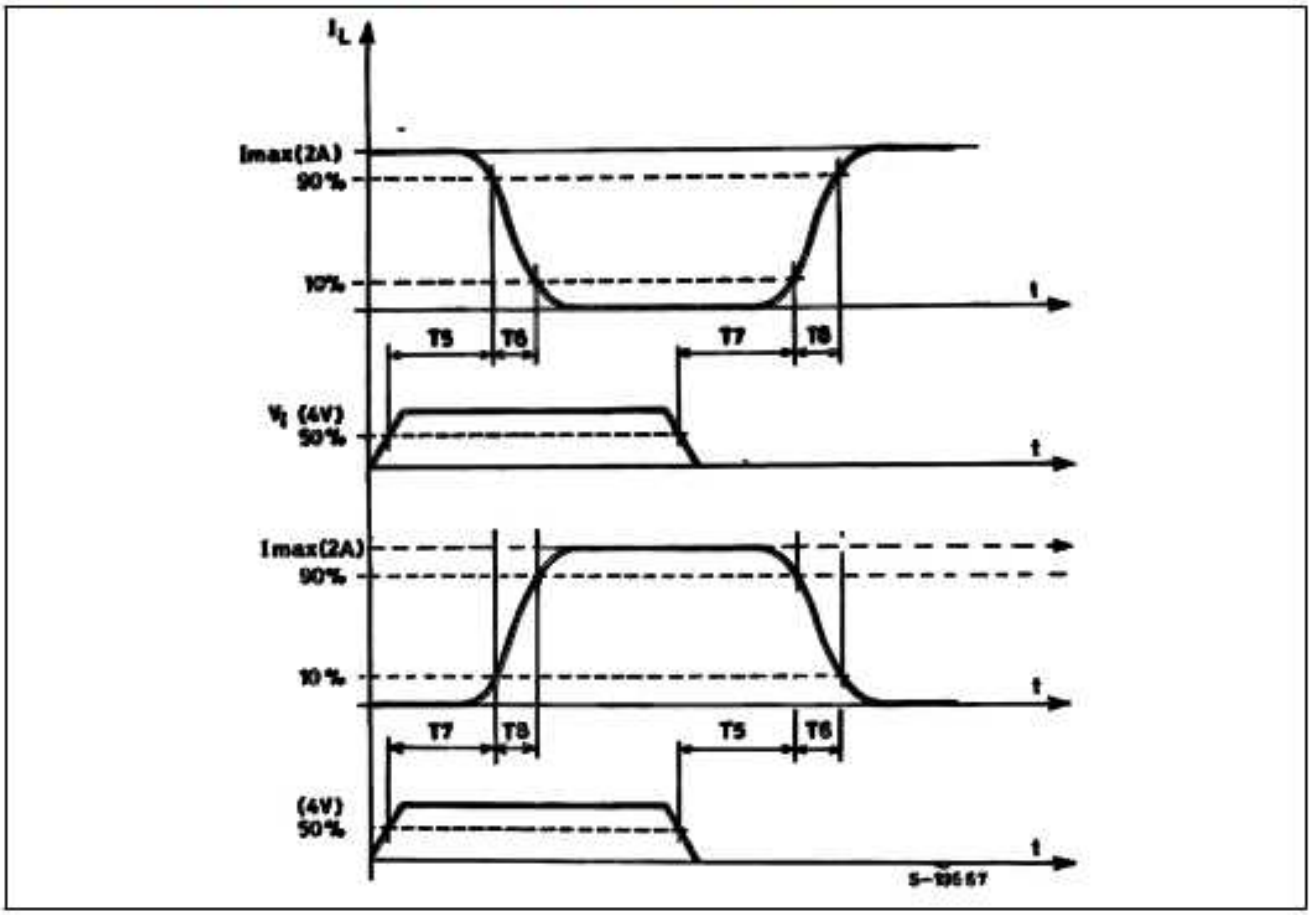
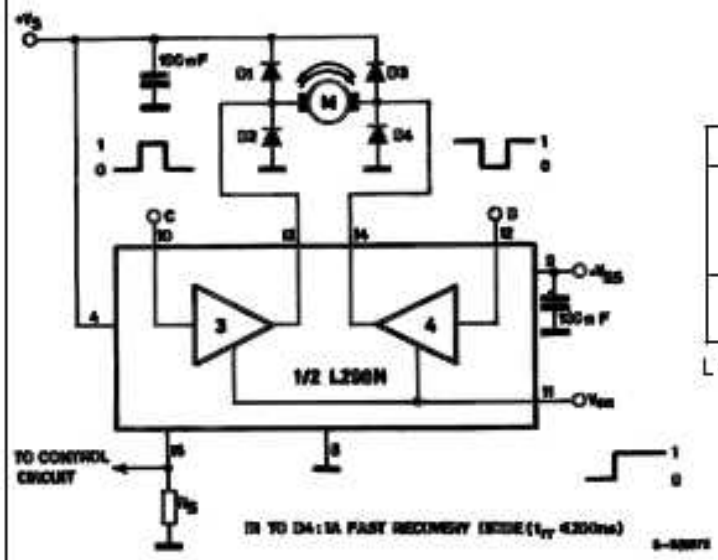


Figure 6 : Bidirectional DC Motor Control.

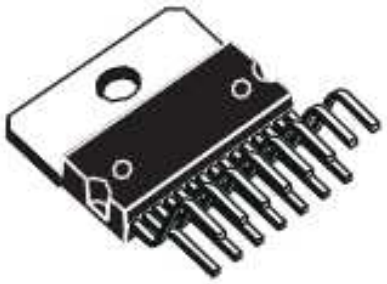


Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

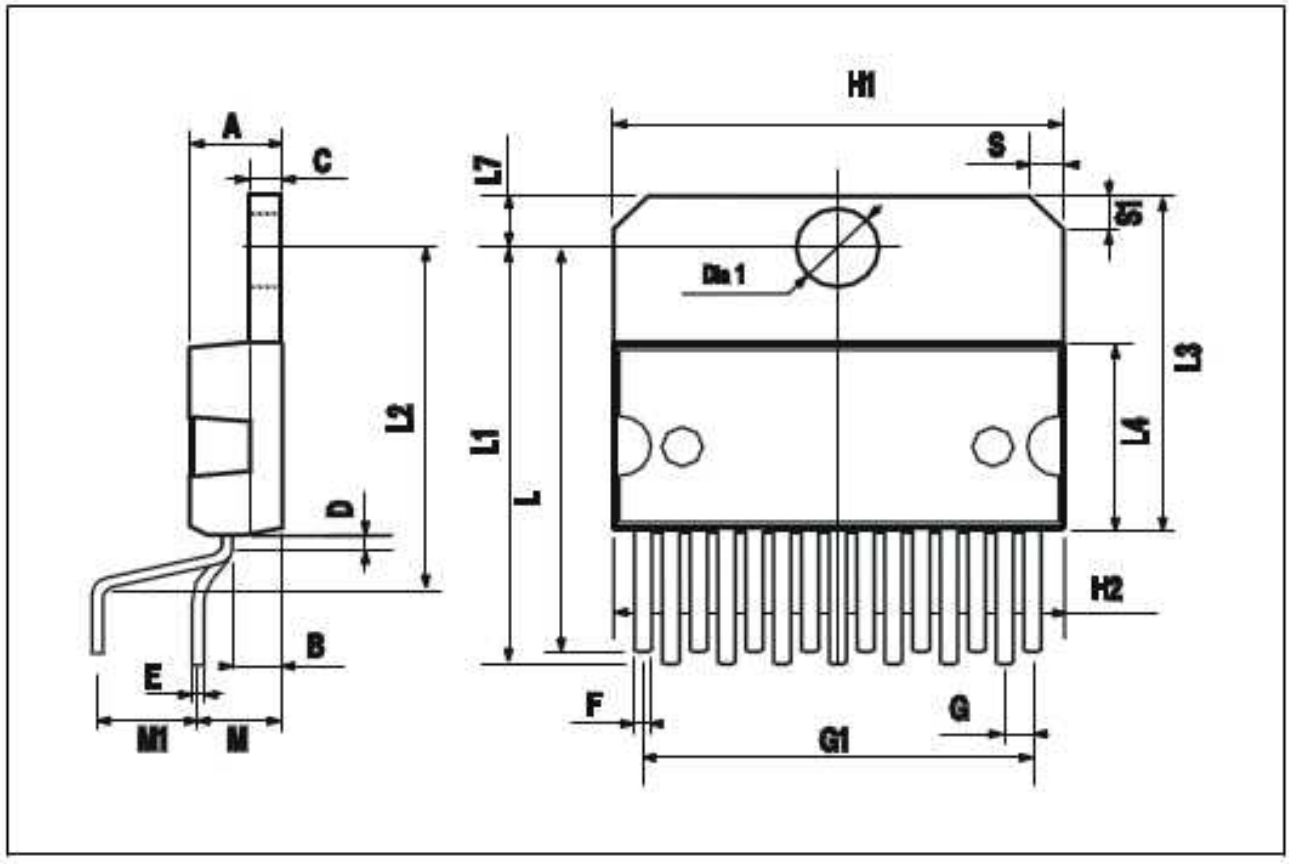
L = Low H = High X = Don't care

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
D		1			0.039	
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.02	1.27	1.52	0.040	0.050	0.060
G1	17.53	17.78	18.03	0.690	0.700	0.710
H1	19.6			0.772		
H2			20.2			0.795
L	21.9	22.2	22.5	0.862	0.874	0.886
L1	21.7	22.1	22.5	0.854	0.870	0.886
L2	17.65		18.1	0.695		0.713
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L7	2.65		2.9	0.104		0.114
M	4.25	4.55	4.85	0.167	0.179	0.191
M1	4.63	5.08	5.53	0.182	0.200	0.218
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA



Multiwatt15 V



FAIRCHILD SEMICONDUCTOR™

August 1986
Revised February 2000

DM7408
Quad 2-Input AND Gates

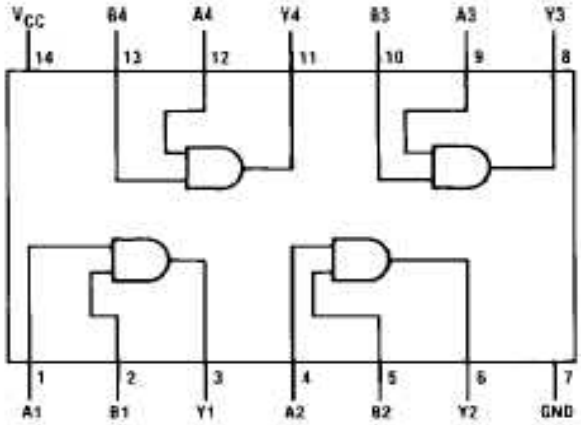
General Description

This device contains four independent gates each of which performs the logic AND function.

Ordering Code:

Order Number	Package Number	Package Description
DM7408N	N14A	14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Connection Diagram



Function Table

$Y = AB$

Inputs		Output
A	B	Y
L	L	L
L	H	L
H	L	L
H	H	H

H = HIGH Logic Level
L = LOW Logic Level

Absolute Maximum Ratings(Note 1)

Supply Voltage	7V
Input Voltage	5.5V
Operating Free Air Temperature Range	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note 1: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V_{CC}	Supply Voltage	4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage	2			V
V_{IL}	LOW Level Input Voltage			0.8	V
I_{OH}	HIGH Level Output Current			−0.8	mA
I_{OL}	LOW Level Output Current			16	mA
T_A	Free Air Operating Temperature	0		70	°C

Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 2)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −12 mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max	2.4	3.4		V
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IH} = Min		0.2	0.4	V
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 5.5V			1	mA
I_{IH}	HIGH Level Input Current	V_{CC} = Max, V_I = 2.4V			40	µA
I_{IL}	LOW Level Input Current	V_{CC} = Max, V_I = 0.4V			−1.6	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 3)	−18		−55	mA
I_{CCH}	Supply Current with Outputs HIGH	V_{CC} = Max		11	21	mA
I_{CCL}	Supply Current with Outputs LOW	V_{CC} = Max		20	33	mA

Note 2: All typicals are at V_{CC} = 5V, T_A = 25°C.

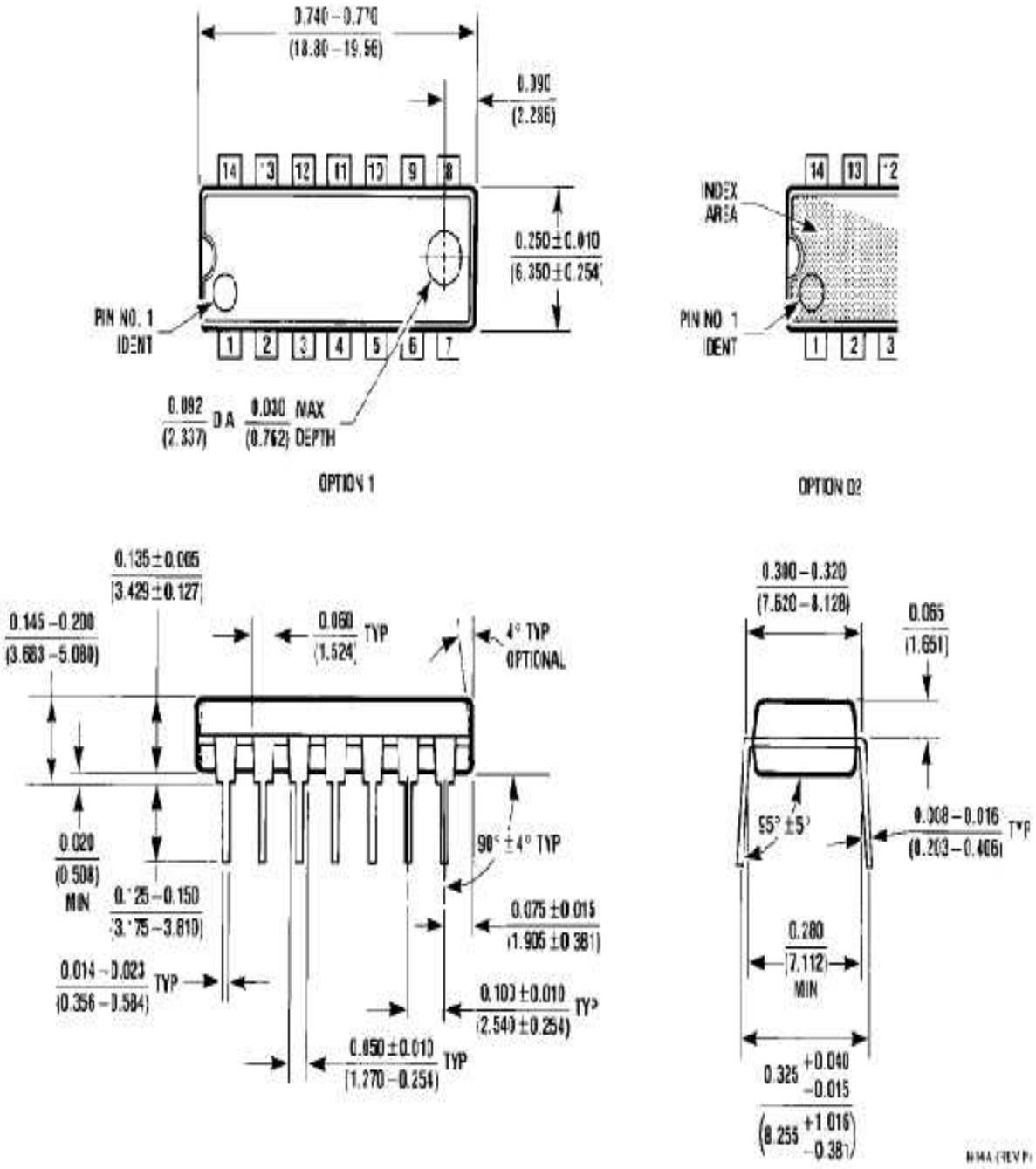
Note 3: Not more than one output should be shorted at a time.

Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

Symbol	Parameter	Conditions	Min	Max	Units
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	C_L = 15 pF, R_L = 400Ω		27	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output			19	ns

Physical Dimensions inches (millimeters) unless otherwise noted



14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide
Package Number N14A

SN5404, SN54LS04, SN54S04, SN7404, SN74LS04, SN74S04 HEX INVERTERS

DECEMBER 1983—REVISED MARCH 1988

- **Package Options Include Plastic "Small Outline" Packages, Ceramic Chip Carriers and Flat Packages, and Plastic and Ceramic DIPs**
- **Dependable Texas Instruments Quality and Reliability**

description

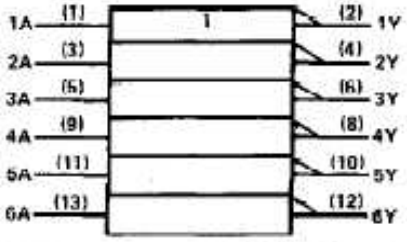
These devices contain six independent inverters.

The SN5404, SN54LS04, and SN54S04 are characterized for operation over the full military temperature range of −55°C to 125°C. The SN7404, SN74LS04, and SN74S04 are characterized for operation from 0°C to 70°C.

FUNCTION TABLE (each inverter)

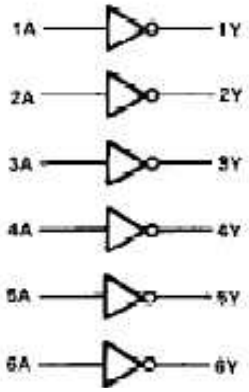
INPUTS A	OUTPUT Y
H	L
L	H

logic symbol†

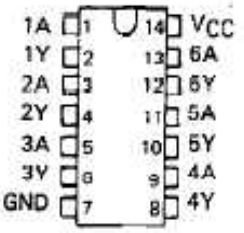


†This symbol is in accordance with ANSI/IEEE Std. 91-1984 and IEC Publication 617-12.
Pin numbers shown are for D, J, and N packages.

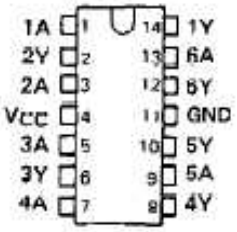
logic diagram (positive logic)



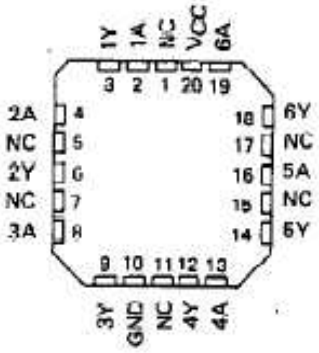
SN5404 . . . J PACKAGE
SN54LS04, SN54S04 . . . J OR W PACKAGE
SN7404 . . . N PACKAGE
SN74LS04, SN74S04 . . . D OR N PACKAGE
(TOP VIEW)



SN5404 . . . W PACKAGE
(TOP VIEW)



SN54LS04, SN54S04 . . . FK PACKAGE
(TOP VIEW)



NC – No internal connection

DARLINGTON COPLEMENTARY SILICON POWER TRANSISTORS

...designed for general-purpose amplifier and low speed switching applications

NPN	PNP
BDX53	BDX54
BDX53A	BDX54A
BDX53B	BDX54B
BDX53C	BDX54C

FEATURES:

* Collector-Emitter Sustaining Voltage-
  $V_{CEO(SUS)}$ = 45 V (Min) - BDX53,BDX54
  = 60 V (Min) - BDX53A,BDX54A
  = 80 V (Min) - BDX53B,BDX54B
  = 100 V(Min) - BDX53C,BDX54C
* Monolithic Construction with Built-in Base-Emitter Shunt Resistor

8 AMPERE
DARLINGTON
COMPLEMENTARY SILICON
POWER TRANSISTORS
45-100 VOLTS
60 WATTS

TO-220

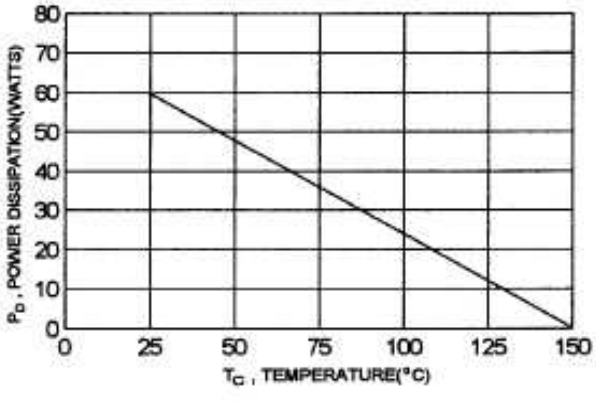
MAXIMUM RATINGS

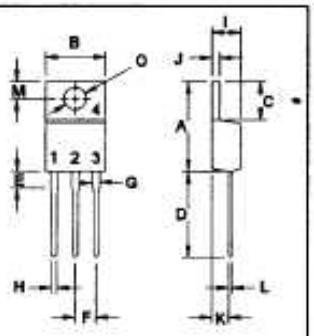
Characteristic	Symbol	BDX53 BDX54	BDX53A BDX54A	BDX53B BDX54B	BDX53C BDX54C	Unit
Collector-Emitter Voltage	V_{CEO}	45	60	80	100	V
Collector-Base Voltage	V_{CBO}	45	60	80	100	V
Emitter-Base Voltage	V_{EBO}	5.0				V
Collector Current - Continuous Peak	I_C I_{CM}	8.0 12				A
Base Current	I_B	0.2				A
Total Power Dissipation @T_C = 25°C Derate above 25°C	P_D	60 0.48				W W/°C
Operating and Storage Junction Temperature Range	T_J , T_{STG}	-65 to +150				°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Max	Unit
Thermal Resistance Junction to Case	$R_{\theta jc}$	2.08	°C/W

FIGURE -1 POWER DERATING





PIN 1.BASE
2.COLLECTOR
3.EMITTER
4.COLLECTOR(CASE)

DIM	MILLIMETERS MIN	MAX
A	14.68	15.31
B	9.78	10.42
C	5.01	6.52
D	13.06	14.62
E	3.57	4.07
F	2.42	3.66
G	1.12	1.36
H	0.72	0.96
I	4.22	4.96
J	1.14	1.38
K	2.20	2.97
L	0.33	0.55
M	2.48	2.98
O	3.70	3.90

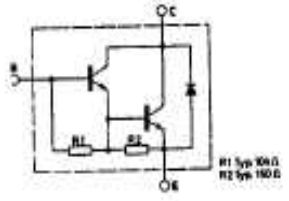
ELECTRICAL CHARACTERISTICS (T_C = 25°C unless otherwise noted)

Characteristic		Symbol	Min	Max	Unit
OFF CHARACTERISTICS					
Collector-Emitter Sustaining Voltage(1) (I_C= 100 mA, I_B= 0)	BDX53, BDX54 BDX53A, BDX54A BDX53B, BDX54B BDX53C, BDX54C	$V_{CEO(sus)}$	45 60 80 100		V
Collector Cutoff Current		I_{CEO}			mA
(V_{CE}= 22 V, I_B= 0)	BDX53, BDX54			0.5	
(V_{CE}= 30 V, I_B= 0)	BDX53A, BDX54A			0.5	
(V_{CE}= 40 V, I_B= 0)	BDX53B, BDX54B			0.5	
(V_{CE}= 50 V, I_B= 0)	BDX53C, BDX54C			0.5	
Collector-Base Cutoff Current (V_{CB}= Rated V_{CB}, I_E= 0)		I_{CBO}		200	uA
Emitter-Base Cutoff Current (V_{EB}= 5.0 V, I_C= 0)		I_{EBO}		2.0	mA
ON CHARACTERISTICS (1)					
DC Current Gain (I_C= 3.0 A, V_{CE}= 3.0 V)		hFE	750		
Collector-Emitter Saturation Voltag (I_C= 3.0 A, I_B= 12 mA)		$V_{CE(sat)}$		2.0	V
Base-Emitter Saturation Voltage (I_C= 3.0 A, I_B= 12 mA)		$V_{BE(sat)}$		2.5	V
Diode Forward-Voltage (I_F= 3.0 A)		V_F		2.5	V

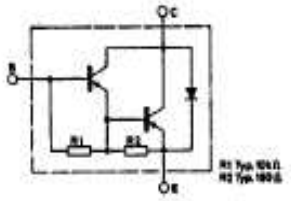
(1) Pulse Test: Pulse Width =300 us,Duty Cycle ≤ 2.0%

INTERNAL SCHEMATIC DIAGRAM

BDX53 Series NPN



BDX54 Series PNP



- Package Options Include Plastic Small-Outline Packages, Ceramic Chip Carriers, and Standard Plastic and Ceramic 300-mil DIPs

description

These devices contain two independent positive-edge-triggered D-type flip-flops. A low level at the preset ($\overline{PRE}$) or clear ($\overline{CLR}$) inputs sets or resets the outputs regardless of the levels of the other inputs. When $\overline{PRE}$ and $\overline{CLR}$ are inactive (high), data at the data (D) input meeting the setup time requirements is transferred to the outputs on the positive-going edge of the clock pulse. Clock triggering occurs at a voltage level and is not directly related to the rise time of the clock pulse. Following the hold-time interval, data at the D input may be changed without affecting the levels at the outputs.

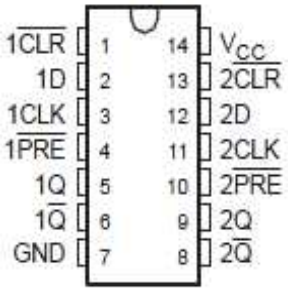
The SN54F74 is characterized for operation over the full military temperature range of –55°C to 125°C. The SN74F74 is characterized for operation from 0°C to 70°C.

FUNCTION TABLE

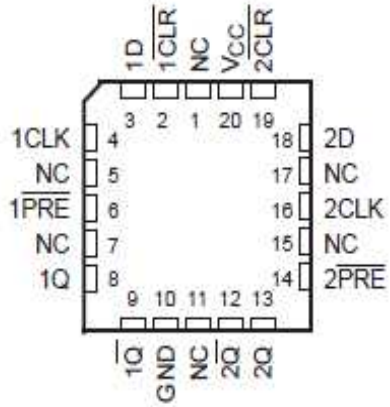
INPUTS				OUTPUTS	
$\overline{PRE}$	$\overline{CLR}$	CLK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H†	H†
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q}_0$

† The output levels are not guaranteed to meet the minimum levels for V_{OH}. Furthermore, this configuration is nonstable; that is, it will not persist when $\overline{PRE}$ or $\overline{CLR}$ returns to its inactive (high) level.

SN54F74 . . . J PACKAGE
SN74F74 . . . D OR N PACKAGE
(TOP VIEW)

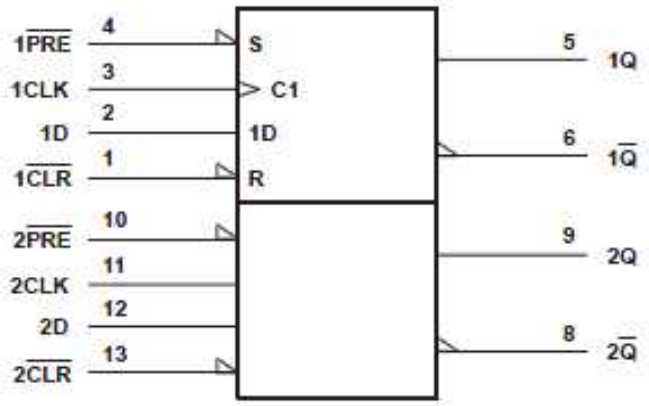


SN54F74 . . . FK PACKAGE
(TOP VIEW)



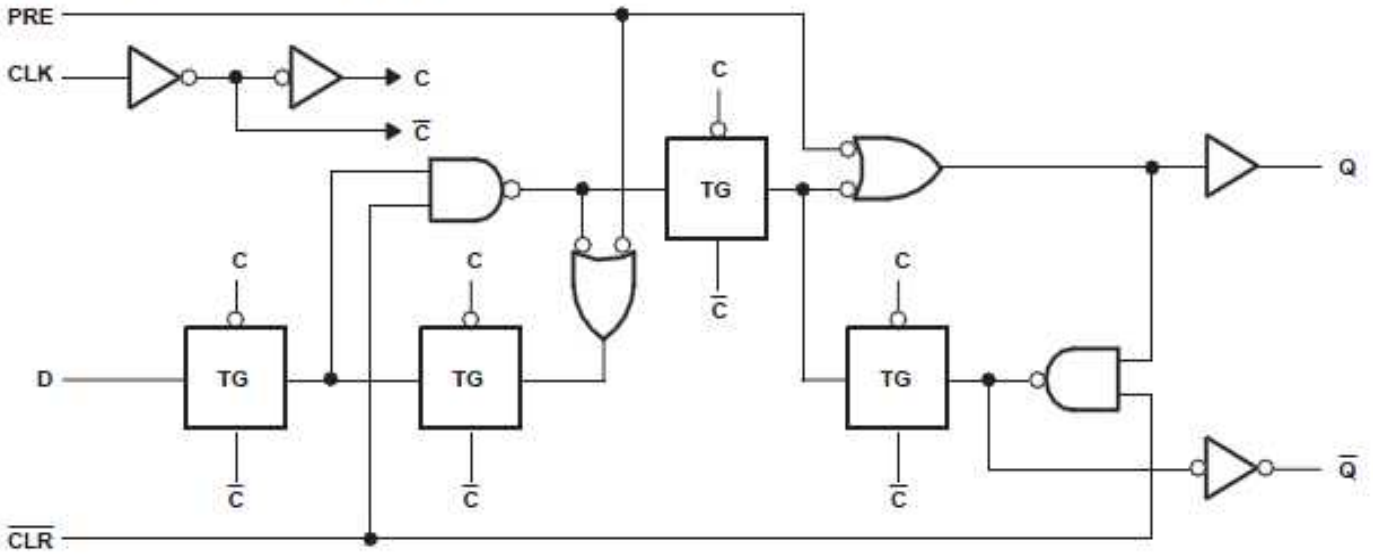
NC – No internal connection

logic symbol†



† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.
Pin numbers shown are for the D, J, and N packages.

logic diagram, each flip-flop (positive logic)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)‡

Supply voltage range, V_{CC}	−0.5 V to 7 V
Input voltage range, V_I (see Note 1)	−1.2 V to 7 V
Input current range	−30 mA to 5 mA
Voltage range applied to any output in the high state	−0.5 V to V_{CC}
Current into any output in the low state	40 mA
Operating free-air temperature range: SN54F74	−55°C to 125°C
SN74F74	0°C to 70°C
Storage temperature range	−65°C to 150°C

‡ Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: The input voltage ratings may be exceeded provided the input current ratings are observed.

recommended operating conditions

		SN54F74			SN74F74			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.5	5	5.5	V
V_{IH}	High-level input voltage	2			2			V
V_{IL}	Low-level input voltage			0.8			0.8	V
I_{IK}	Input clamp current			–18			–18	mA
I_{OH}	High-level output current			–1			–1	mA
I_{OL}	Low-level output current			20			20	mA
T_A	Operating free-air temperature	–55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS		SN54F74			SN74F74			UNIT
				MIN	TYP†	MAX	MIN	TYP†	MAX	
V_{IK}		V_{CC} = 4.5 V,	I_I = –18 mA			–1.2			–1.2	V
V_{OH}		V_{CC} = 4.5 V,	I_{OH} = –1 mA	2.5	3.4		2.5	3.4		V
		V_{CC} = 4.75 V,	I_{OH} = –1 mA				2.7			
V_{OL}		V_{CC} = 4.5 V,	I_{OL} = 20 mA		0.3	0.5		0.3	0.5	V
I_I		V_{CC} = 5.5 V,	V_I = 7 V			0.1			0.1	mA
I_{IH}		V_{CC} = 5.5 V,	V_I = 2.7 V			20			20	µA
I_{IL}	Data, CLK	V_{CC} = 5.5 V,	V_I = 0.5 V			–0.6			–0.6	mA
	$\overline{PRE}$ or $\overline{CLR}$					–1.8			–1.8	
I_{OS}‡		V_{CC} = 5.5 V,	V_O = 0	–60		–150	–60		–150	mA
I_{CC}		V_{CC} = 5.5 V,	See Note 2		10.5	16		10.5	16	mA

† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ Not more than one output should be shorted at a time, and the duration of the short circuit should not exceed one second.
NOTE 2: I_{CC} is measured with D, CLK, and $\overline{PRE}$ grounded then with D, CLK, and $\overline{CLR}$ grounded.

timing requirements over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)

			V_{CC} = 5 V, T_A = 25°C		SN54F74		SN74F74		UNIT
			'F74						
			MIN	MAX	MIN	MAX	MIN	MAX	
f_{clock}	Clock frequency		0	100	0	80	0	100	MHz
t_w	Pulse duration	CLK high, $\overline{PRE}$ or $\overline{CLR}$ low	4		4		4		ns
		CLK low	5		6		5		
t_{su}	Setup time, data before CLK↑	High	2		3		2		ns
		Low	3		4		3		
	Setup time, inactive-state before CLK↑§	$\overline{PRE}$ or $\overline{CLR}$ to CLK	2		3		2		
t_h	Hold time, data after CLK↑	High	1		2		1		ns
		Low	1		2		1		

§ Inactive-state setup time is also referred to as recovery time.

KEC
KOREA ELECTRONICS CO.,LTD.

SEMICONDUCTOR
TECHNICAL DATA

KIA7805AP/API~ KIA7824AP/API
BIPOLAR LINEAR INTEGRATED CIRCUIT

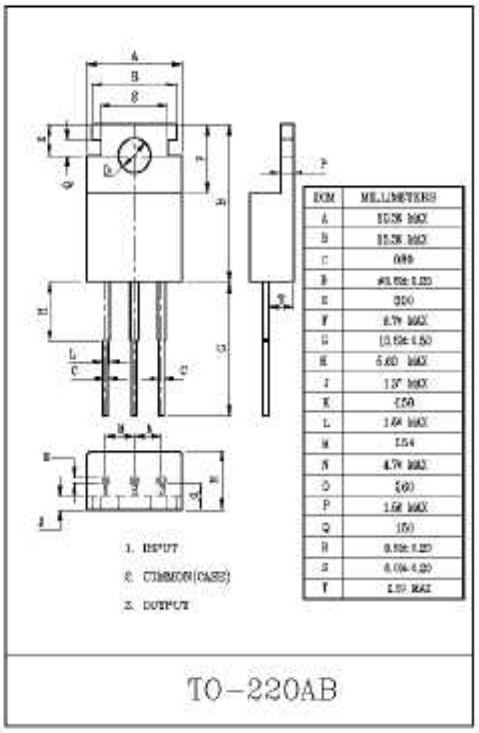
THREE TERMINAL POSITIVE VOLTAGE REGULATORS
5V, 6V, 8V, 9V, 10V, 12V, 15V, 18V, 20V, 24V.

FEATURES

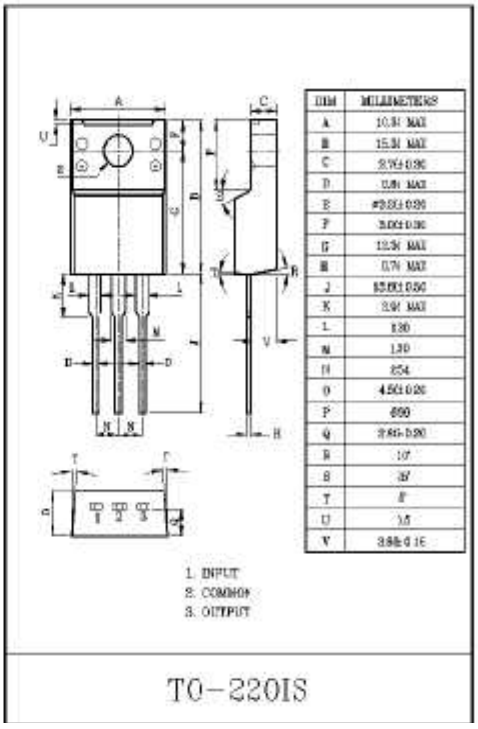
- Suitable for C-MOS, TTL, the Other Digital IC's Power Supply.
- Internal Thermal Overload Protection.
- Internal Short Circuit Current Limiting.
- Output Current in Excess of 1A.
- Satisfies IEC-65 Specification. (International Electronical Commission).

MAXIMUM RATINGS (Ta=25℃)

CHARACTERISTIC		SYMBOL	RATING	UNIT
Input Voltage	KIA7805AP/API~ KIA7815AP/API	V_{IN}	35	V
	KIA7818AP/API~ KIA7824AP/API		40	
Power Dissipation (Tc=25℃)		P_D	20.8	W
Power Dissipation (Without Heatsink)	KIA7805API~ KIA7824API	P_D	2.0	W
Operating Junction Temperature		T_j	-30~150	℃
Storage Temperature		T_{stg}	-35~150	℃



TO-220AB

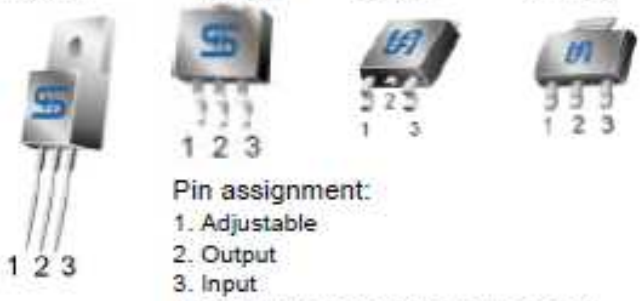


TO-220IS

TS317

3-Terminal Adjustable Output Positive Voltage Regulator

TO-220 TO-263 TO-252 SOT-223



Pin assignment:
1. Adjustable
2. Output
3. Input

(Heatsink surface connected to pin 2)

Output Voltage Range From 1.25V to 37V
Output Current up to 1.5A

General Description

The TS317 is adjustable 3-terminal positive voltage regulator capable of supplying in excess of 1.5A over an output voltage range of 1.25 V to 37 V. This voltage regulator is exceptionally easy to use and require only two external resistors to set the output voltage. Further, it employs internal current limiting, thermal shutdown and safe area compensation, making it essentially blow-out proof.

The TS317 is offered in 3-pin TO-220, TO-263, TO-252 and SOT-223 package.

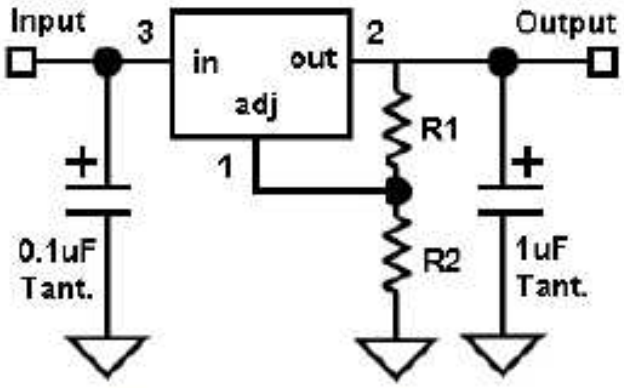
Features

- Output current up to 1.5A
  - * TO-220/TO-263 for 1.5A
  - * TO-252/SOT-223 for 500mA
- Output Adjustable between 1.25 V and 37 V
- Internal Thermal Overload Protection
- Internal Short-Circuit Current Limiting Constant with Temperature
- Output Transistor Safe-Area Compensation
- Floating Operation for High Voltage Applications
- Eliminates Stocking Many Fixed Voltages
- Output voltage offered in 4% tolerance

Ordering Information

Part No.	Operating Temp.	Package
TS317CZ	-20 ~ +150°C	TO-220
TS317CM		TO-263
TS317CP		TO-252
TS317CW		SOT-223

Standard Application



$V_{out} = 1.25\ V * (1 + R2 / R1) + I_{adj} * R2$

Since I_{Adj} is controlled to less than 100 μA, the error associated with this term is negligible in most applications A common ground is required between the input and the output voltages. The input voltage must remain typically 2.0V above the output voltage even during the low point on the Input ripple voltage.

* = Cin is required if regulator is located an appreciable distance from power supply filter.

** = Co is not needed for stability; however, it does improve transient response.

Appendix D

- **PIC18F4550 Microcontroller**
- **Datasheet & C Code**

MICROCHIP

PIC18F2455/2550/4455/4550

28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 1-Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 µA typical
- Sleep mode currents down to 0.1 µA typical
- Timer1 Oscillator: 1.1 µA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 µA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes, including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal Oscillator Block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Dual Oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-Current Sink/Source: 25 mA/25 mA
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 5.2 ns (TCY/16)
  - Compare is 16-bit, max. resolution 83.3 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI (all 4 modes) and I²C™ Master and Slave modes
- 10-bit, up to 13-channel Analog-to-Digital Converter module (A/D) with Programmable Acquisition Time
- Dual Analog Comparators with Input Multiplexing

Special Microcontroller Features:

- C Compiler Optimized Architecture with optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide Operating Voltage Range (2.0V to 5.5V)

Device	Program Memory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I²C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

© 2007 Microchip Technology Inc.

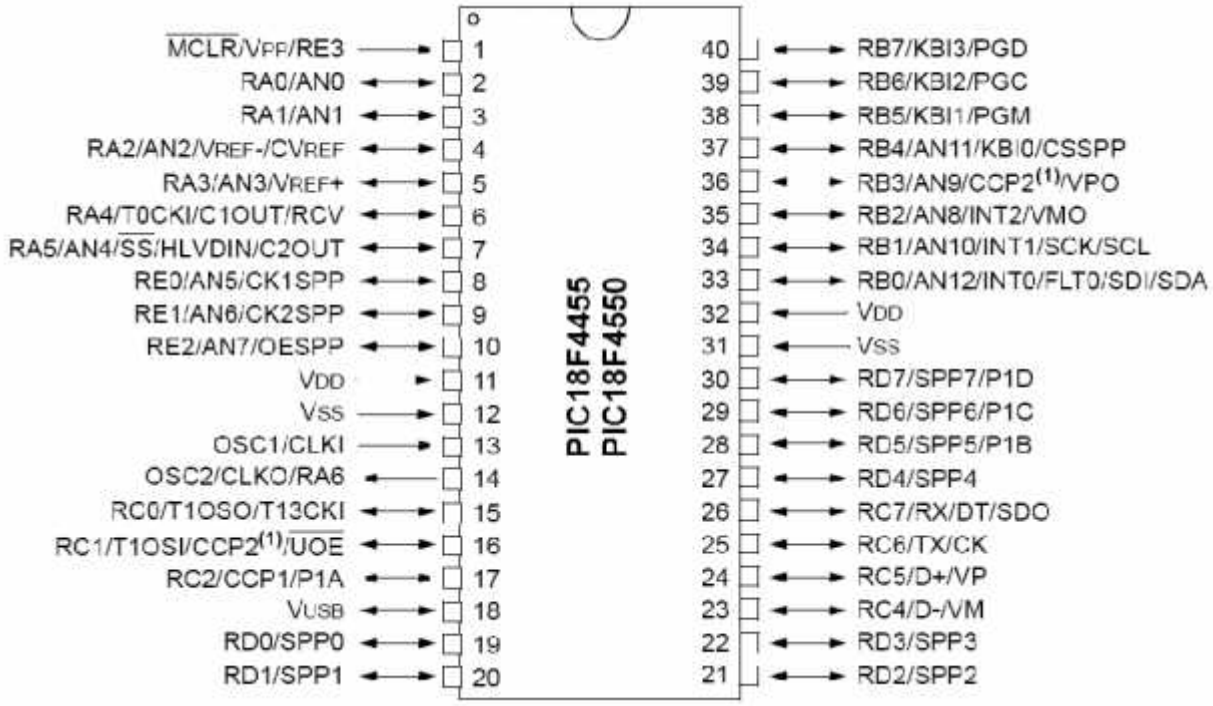


TABLE 1-3: PIC18F4455/4550 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
MCLR/VPP/RE3	1	18	18			Master Clear (input) or programming voltage (input).
MCLR				I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
VPP				P		Programming voltage input.
RE3				I	ST	Digital input.
OSC1/CLKI	13	32	30			Oscillator crystal or external clock input.
OSC1				I	Analog	Oscillator crystal input or external clock source input.
CLKI				I	Analog	External clock source input. Always associated with pin function OSC1. (See OSC2/CLKO pin.)
OSC2/CLKO/RA6	14	33	31			Oscillator crystal or clock output.
OSC2				O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO				O	—	In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6				I/O	TTL	General purpose I/O pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

3: These pins are No Connect unless the ICPRT Configuration bit is set. For NC/ICPORTS, the pin is No Connect unless ICPRT is set and the DEBUG Configuration bit is cleared.

TABLE 1-3: PIC18F4455/4550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTA is a bidirectional I/O port.
RA0/AN0	2	19	19			
RA0				I/O	TTL	Digital I/O.
AN0				I	Analog	Analog input 0.
RA1/AN1	3	20	20			
RA1				I/O	TTL	Digital I/O.
AN1				I	Analog	Analog input 1
RA2/AN2/VREF-/CVREF	4	21	21			
RA2				I/O	TTL	Digital I/O.
AN2				I	Analog	Analog input 2.
VREF-				I	Analog	A/D reference voltage (low) input.
CVREF				O	Analog	Analog comparator reference output.
RA3/AN3/VREF+	5	22	22			
RA3				I/O	TTL	Digital I/O.
AN3				I	Analog	Analog input 3.
VREF+				I	Analog	A/D reference voltage (high) input.
RA4/T0CKI/C1OUT/RCV	6	23	23			
RA4				I/O	ST	Digital I/O
T0CKI				I	ST	Timer0 external clock input.
C1OUT				O	—	Comparator 1 output.
RCV				I	TTL	External USB transceiver RCV input.
RA5/AN4/$\overline{SS}$/HLVDIN/C2OUT	7	24	24			
RA5				I/O	TTL	Digital I/O.
AN4				I	Analog	Analog input 4.
SS				I	TTL	SPI slave select input.
HLVDIN				I	Analog	High/Low-Voltage Detect input
C2OUT				O	—	Comparator 2 output.
RA6						See the OSC2/CLKO/RA6 pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

3: These pins are No Connect unless the ICPRT Configuration bit is set. For NC/ICPORTS, the pin is No Connect unless ICPRT is set and the DEBUG Configuration bit is cleared.

TABLE 1-3: PIC18F4455/4550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/AN12/INT0/FLT0/SDI/SDA	33	9	8			
RB0				I/O	TTL	Digital I/O.
AN12				I	Analog	Analog input 12.
INT0				I	ST	External interrupt 0.
FLT0				I	ST	Enhanced PWM Fault input (ECCP1 module).
SDI				I	ST	SPI data in.
SDA				I/O	ST	I^2C™ data I/O.
RB1/AN10/INT1/SCK/SCL	34	10	9			
RB1				I/O	TTL	Digital I/O.
AN10				I	Analog	Analog input 10.
INT1				I	ST	External interrupt 1.
SCK				I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL				I/O	ST	Synchronous serial clock input/output for I^2C mode.
RB2/AN8/INT2/VMO	35	11	10			
RB2				I/O	TTL	Digital I/O.
AN8				I	Analog	Analog input 8.
INT2				I	ST	External interrupt 2.
VMO				O	—	External USB transceiver VMO output.
RB3/AN9/CCP2/VPO	36	12	11			
RB3				I/O	TTL	Digital I/O.
AN9				I	Analog	Analog input 9.
CCP2(1)				I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
VPO				O	—	External USB transceiver VPO output.
RB4/AN11/KBI0/CSSPP	37	14	14			
RB4				I/O	TTL	Digital I/O.
AN11				I	Analog	Analog input 11.
KBI0				I	TTL	Interrupt-on-change pin.
CSSPP				O	—	SPP chip select control output.
RB5/KBI1/PGM	38	15	15			
RB5				I/O	TTL	Digital I/O.
KBI1				I	TTL	Interrupt-on-change pin.
PGM				I/O	ST	Low-Voltage ICSP™ Programming enable pin.
RB6/KBI2/PGC	39	16	16			
RB6				I/O	TTL	Digital I/O.
KBI2				I	TTL	Interrupt-on-change pin.
PGC				I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/KBI3/PGD	40	17	17			
RB7				I/O	TTL	Digital I/O.
KBI3				I	TTL	Interrupt-on-change pin.
PGD				I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

3: These pins are No Connect unless the ICPRT Configuration bit is set. For NC/ICPORTS, the pin is No Connect unless ICPRT is set and the DEBUG Configuration bit is cleared.

TABLE 1-3: PIC18F4455/4550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTC is a bidirectional I/O port.
RC0/T1OSO/T13CKI	15	34	32			
RC0				I/O	ST	Digital I/O.
T1OSO				O	—	Timer1 oscillator output.
T13CKI				I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2/UOE	16	35	35			
RC1				I/O	ST	Digital I/O.
T1OSI				I	CMOS	Timer1 oscillator input.
CCP2(2)				I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
UOE				O	—	External USB transceiver OE output.
RC2/CCP1/P1A	17	36	36			
RC2				I/O	ST	Digital I/O.
CCP1				I/O	ST	Capture 1 input/Compare 1 output/PWM 1 output.
P1A				O	TTL	Enhanced CCP1 PWM output, channel A.
RC4/D-/VM	23	42	42			
RC4				I	TTL	Digital input.
D-				I/O	—	USB differential minus line (input/output).
VM				I	TTL	External USB transceiver VM input.
RC5/D+/VP	24	43	43			
RC5				I	TTL	Digital input.
D+				I/O	—	USB differential plus line (input/output).
VP				I	TTL	External USB transceiver VP input.
RC6/TX/CK	25	44	44			
RC6				I/O	ST	Digital I/O.
TX				O	—	EUSART asynchronous transmit.
CK				I/O	ST	EUSART synchronous clock (see RX/DT).
RC7/RX/DT/SDO	26	1	1			
RC7				I/O	ST	Digital I/O.
RX				I	ST	EUSART asynchronous receive.
DT				I/O	ST	EUSART synchronous data (see TX/CK).
SDO				O	—	SPI data out.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

3: These pins are No Connect unless the ICPRT Configuration bit is set. For NC/ICPORTS, the pin is No Connect unless ICPRT is set and the DEBUG Configuration bit is cleared.

TABLE 1-3: PIC18F4455/4550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTD is a bidirectional I/O port or a Streaming Parallel Port (SPP). These pins have TTL input buffers when the SPP module is enabled.
RD0/SPP0	19	38	38			
RD0				I/O	ST	Digital I/O.
SPP0				I/O	TTL	Streaming Parallel Port data.
RD1/SPP1	20	39	39			
RD1				I/O	ST	Digital I/O.
SPP1				I/O	TTL	Streaming Parallel Port data.
RD2/SPP2	21	40	40			
RD2				I/O	ST	Digital I/O.
SPP2				I/O	TTL	Streaming Parallel Port data.
RD3/SPP3	22	41	41			
RD3				I/O	ST	Digital I/O.
SPP3				I/O	TTL	Streaming Parallel Port data.
RD4/SPP4	27	2	2			
RD4				I/O	ST	Digital I/O.
SPP4				I/O	TTL	Streaming Parallel Port data.
RD5/SPP5/P1B	28	3	3			
RD5				I/O	ST	Digital I/O.
SPP5				I/O	TTL	Streaming Parallel Port data.
P1B				O	—	Enhanced CCP1 PWM output, channel B.
RD6/SPP6/P1C	29	4	4			
RD6				I/O	ST	Digital I/O.
SPP6				I/O	TTL	Streaming Parallel Port data.
P1C				O	—	Enhanced CCP1 PWM output, channel C.
RD7/SPP7/P1D	30	5	5			
RD7				I/O	ST	Digital I/O.
SPP7				I/O	TTL	Streaming Parallel Port data.
P1D				O	—	Enhanced CCP1 PWM output, channel D.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels; I = Input
O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

3: These pins are No Connect unless the ICPRT Configuration bit is set. For NC/ICPORTS, the pin is No Connect unless ICPRT is set and the DEBUG Configuration bit is cleared.

TABLE 1-3: PIC18F4455/4550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTE is a bidirectional I/O port.
RE0/AN5/CK1SPP	8	25	25			
RE0				I/O	ST	Digital I/O.
AN5				I	Analog	Analog input 5.
CK1SPP				O	—	SPP clock 1 output.
RE1/AN6/CK2SPP	9	26	26			
RE1				I/O	ST	Digital I/O.
AN6				I	Analog	Analog input 6.
CK2SPP				O	—	SPP clock 2 output.
RE2/AN7/OESPP	10	27	27			
RE2				I/O	ST	Digital I/O.
AN7				I	Analog	Analog input 7.
OESPP				O	—	SPP output enable output.
RE3	—	—	—	—	—	See $\overline{\text{MCLR}}$/VPP/RE3 pin.
VSS	12, 31	6, 30, 31	6, 29	P	—	Ground reference for logic and I/O pins.
VDD	11, 32	7, 8, 28, 29	7, 28	P	—	Positive supply for logic and I/O pins.
VUSB	18	37	37	O	—	Internal USB 3.3V voltage regulator output.
NC/ICCK/ICPGC(3)	—	—	12			No Connect or dedicated ICD/ICSP™ port clock.
ICCK				I/O	ST	In-Circuit Debugger clock.
ICPGC				I/O	ST	ICSP programming clock.
NC/ICDT/ICPGD(3)	—	—	13			No Connect or dedicated ICD/ICSP port clock.
ICDT				I/O	ST	In-Circuit Debugger data.
ICPGD				I/O	ST	ICSP programming data.
NC/$\overline{\text{ICRST}}$/ICVPP(3)	—	—	33			No Connect or dedicated ICD/ICSP port Reset.
$\overline{\text{ICRST}}$				I	—	Master Clear (Reset) input.
ICVPP				P	—	Programming voltage input.
NC/ICPORTS(3)	—	—	34	P	—	No Connect or 28-pin device emulation.
ICPORTS						Enable 28-pin device emulation when connected to Vss.
NC	—	13	—	—	—	No Connect.

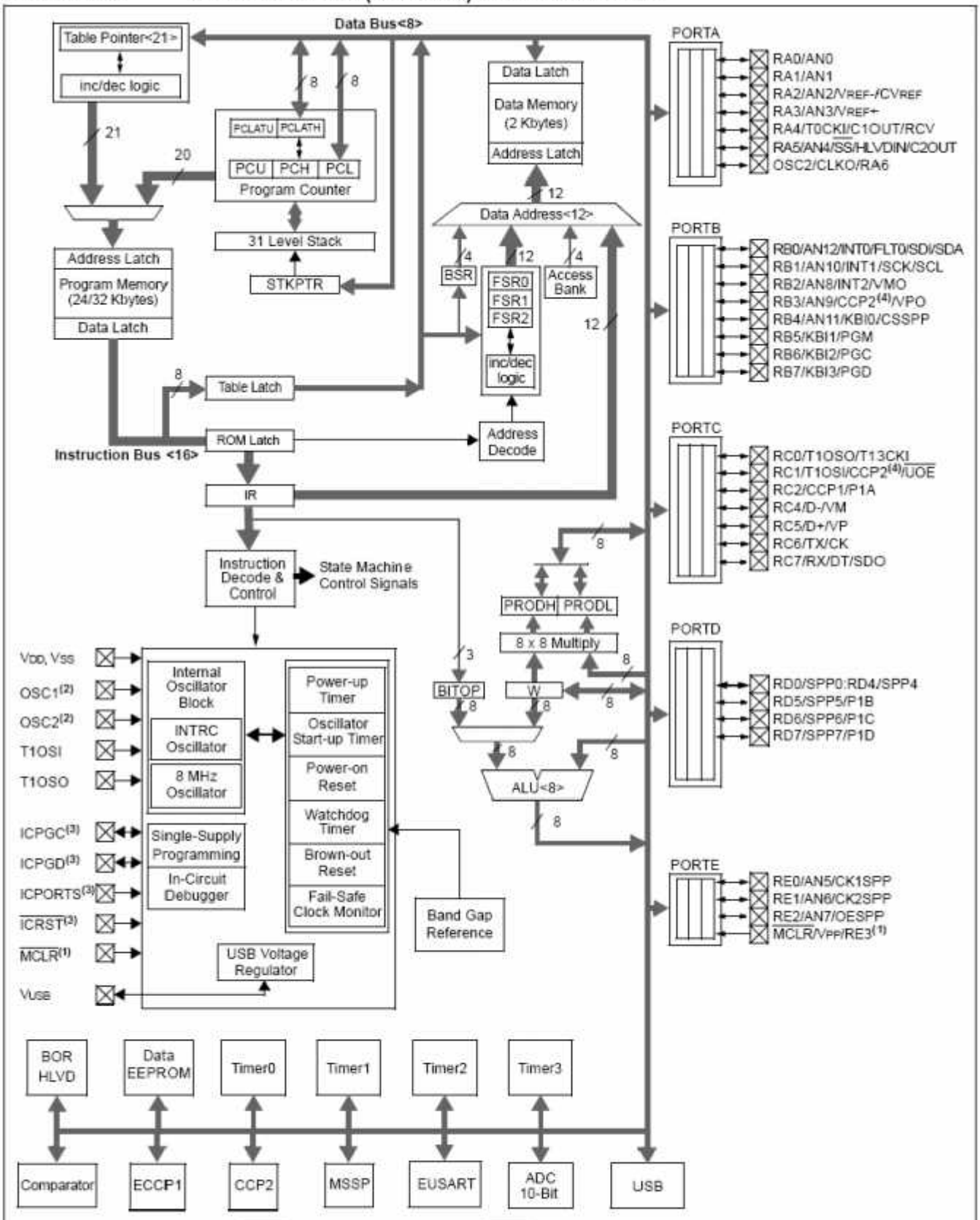
Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

3: These pins are No Connect unless the $\overline{\text{ICPRT}}$ Configuration bit is set. For NC/ICPORTS, the pin is No Connect unless ICPRT is set and the $\overline{\text{DEBUG}}$ Configuration bit is cleared.

FIGURE 1-2: PIC18F4455/4550 (40/44-PIN) BLOCK DIAGRAM



Note 1: RE3 is multiplexed with $\overline{MCLR}$ and is only available when the $\overline{MCLR}$ Resets are disabled.

2: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.

3: These pins are only available on 44-pin TQFP packages under certain conditions. Refer to **Section 25.9 "Special ICPORT Features (Designated Packages Only)"** for additional information.

4: RB3 is the alternate pin for CCP2 multiplexing.

28.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings(†)

Ambient temperature under bias ..... -40°C to +85°C

Storage temperature ..... -65°C to +150°C

Voltage on any pin with respect to Vss (except V_{DD}, $\overline{MCLR}$ and RA4) ..... -0.3V to (V_{DD} + 0.3V)

Voltage on V_{DD} with respect to Vss ..... -0.3V to +7.5V

Voltage on $\overline{MCLR}$ with respect to Vss **(Note 2)** ..... 0V to +13.25V

Total power dissipation **(Note 1)** 1.0W

Maximum current out of Vss pin ..... 300 mA

Maximum current into V_{DD} pin ..... 250 mA

Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD}) ..... ±20 mA

Output clamp current, I_{OK} (V_O < 0 or V_O > V_{DD}) ..... ±20 mA

Maximum output current sunk by any I/O pin ..... 25 mA

Maximum output current sourced by any I/O pin ..... 25 mA

Maximum current sunk by all ports ..... 200 mA

Maximum current sourced by all ports ..... 200 mA

Note 1: Power dissipation is calculated as follows:
$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$

2: Voltage spikes below Vss at the $\overline{MCLR}$/V_{PP}/RE3 pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{MCLR}$/V_{PP}/RE3 pin, rather than pulling this pin directly to Vss.

† **NOTICE:** Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

FIGURE 28-1: PIC18F2455/2550/4455/4550 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)

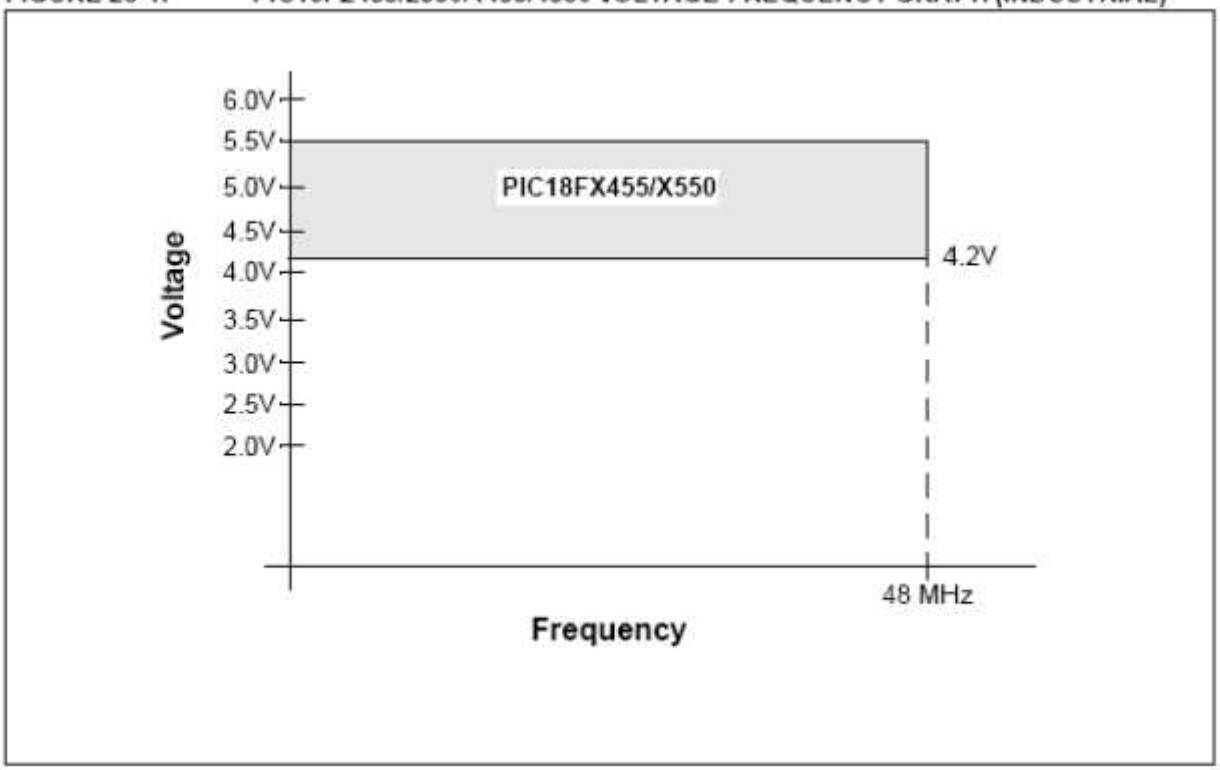
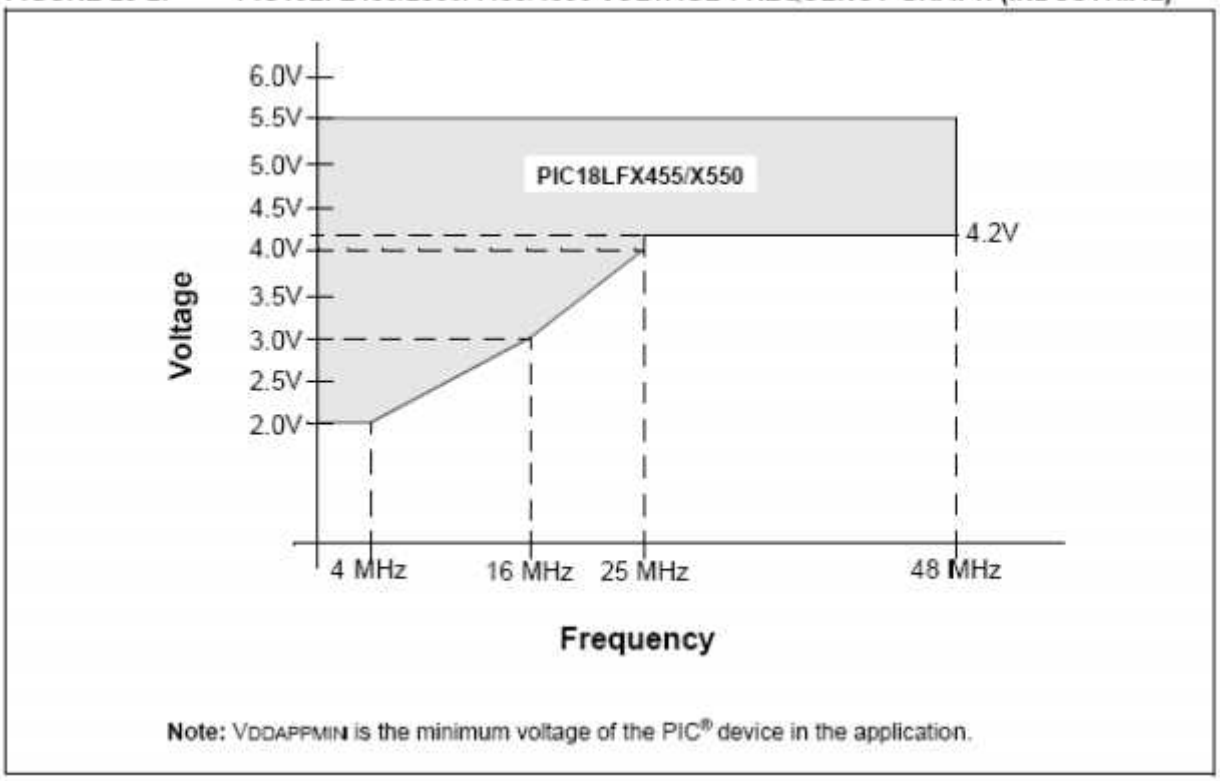


FIGURE 28-2: PIC18LF2455/2550/4455/4550 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)



Note: VDDAPPMIN is the minimum voltage of the PIC® device in the application.

Control PIC Code Developed In C Programming Language

```
#include<p18f4550.h>
#include<delays.h>
#include<adc.h>
#include<PWM.h>
#include<timers.h>
#include<usart.h>
#include <math.h>
#pragma config FOSC = INTOSC_HS
#pragma config WDT = OFF

float pi=3.14;
int FromRCW=0,FromRCCW=0,FromMF=0,FromMB=0,StopMotors=0,interupt=0,nX=0,nY=0,oX=0,oY=0,i=0;
void checksensors(char side);
void RotateCW(int degree);
void RotateCCW(int degree);
void MoveF(int distance);
void CalculateXY(int X, int Y)
{
                int distance=0, degree=0;
                distance = sqrt((X-oX)*(X-oX) + (Y-oY)*(Y-oY) );
                Delay10KTCYx(10);
                degree = (atan ( (Y-oY) / (X-oX) ))*(180/pi);
                Delay10KTCYx(10);
                RotateCW(degree);
                Delay10KTCYx(10);
                MoveF(distance);
                oX=X;
                oY=Y;
}
void stepper(char clock)
{
int i=0;
switch(clock)
{
case 'c':{
        while(i<=20)
        {
                        //Counter Clocl Wise
                        PORTDbits.RD4=0;
                        PORTDbits.RD5=0;
                        PORTDbits.RD6=0;
                        PORTDbits.RD7=1;
                        Delay1KTCYx(400);
                        PORTDbits.RD4=1;
                        PORTDbits.RD5=1;
                        PORTDbits.RD6=0;
                        PORTDbits.RD7=1;
                        Delay1KTCYx(400);
                        PORTDbits.RD4=1;
                        PORTDbits.RD5=1;
                        PORTDbits.RD6=1;
                        PORTDbits.RD7=0;
                        Delay1KTCYx(400);
                        PORTDbits.RD4=0;
                        PORTDbits.RD5=0;
                        PORTDbits.RD6=1;
```

```
                        PORTDbits.RD7=0;
                        Delay1KTCYx(400);
                        i++;
                        }
                        i=0;
                        break;
}
case 'a':{
                        while(i<=20)
                        {
                        //Clock wise

                        PORTDbits.RD4=1;
                        PORTDbits.RD5=0;
                        PORTDbits.RD6=0;
                        PORTDbits.RD7=0;
                        Delay1KTCYx(400);
                        PORTDbits.RD4=1;
                        PORTDbits.RD5=0;
                        PORTDbits.RD6=1;
                        PORTDbits.RD7=1;
                        Delay1KTCYx(400);
                        PORTDbits.RD4=0;
                        PORTDbits.RD5=1;
                        PORTDbits.RD6=1;
                        PORTDbits.RD7=1;
                        Delay1KTCYx(400);
                        PORTDbits.RD4=0;
                        PORTDbits.RD5=1;
                        PORTDbits.RD6=0;
                        PORTDbits.RD7=0;
                        Delay1KTCYx(400);
                        i++;
                        }
                        i=0;
                        break;
                }
                        PORTDbits.RD4=0;
                        PORTDbits.RD5=0;
                        PORTDbits.RD6=0;
                        PORTDbits.RD7=0;
}
}

int ToDecimal(char number)
{
        int dnumber=0;
        if(number == '0')
                {dnumber=0;}
        else if(number == '1')
                {dnumber=1;}
        else if(number == '2')
                {dnumber=2;}
        else if(number == '3')
                {dnumber=3;}
        else if(number == '4')
                {dnumber=4;}
        else if(number == '5')
                {dnumber=5;}
        else if(number == '6')
                {dnumber=6;}
        else if(number == '7')
```

```
		{dnumber=7;}
	else if(number == '8')
		{dnumber=8;}
	else if(number == '9')
		{dnumber=9;}
	else
		{dnumber=-1;}
	return dnumber;
}

void MoveF(int countcm)
{
	unsigned int count=0,Counter1=0,Counter2=0,Rwheel=0,Lwheel=0,Rspeed=0,Lspeed=0;
	int Sens1=0,Sens2=0;
	char side;
	count=countcm*21;
	Rwheel=count;
	Lwheel=count;
	Rspeed=396;
	Lspeed=396;
	PORTBbits.RB4=1; //left
	PORTBbits.RB5=1; //right
	side='F';

	SetChanADC(  ADC_CH0 );
	ConvertADC();
	while(BusyADC());
	Sens1=ReadADC();
	SetChanADC(  ADC_CH1 );
	ConvertADC();
	while(BusyADC());
	Sens2=ReadADC();
	if(Sens1<=16 || Sens2<=16)
		{
			SetDCPWM2(0);
			SetDCPWM1(0);
		}
	else
		{
		WriteTimer1(0);
		WriteTimer0(0);
		SetDCPWM2(Lspeed);
		OpenPWM2(99);
		SetDCPWM1(Rspeed);
		OpenPWM1(99);
		if(countcm<=8)
			{
			while(Counter1<=Rwheel && Counter2<=Lwheel && StopMotors==0 && interupt==0)
				{
					Counter1=ReadTimer1();
			Counter2=ReadTimer0();
						if(Counter1< Counter2)
							{
								Counter1=Counter1+(Counter2-Counter1);
							}
						else if(Counter1> Counter2)
							{
								Counter2=Counter2+(Counter1-Counter2);
							}
						checksensors(side);
						SetDCPWM2(Lspeed);
						SetDCPWM1(Rspeed);
```

```
                                        }
                                }
                        else
                                {
                                        Delay10KTCYx(80);
                                        while(Counter1<=Rwheel && Counter2<=Lwheel && StopMotors==0 && PORTDbits.RD0==0 &&
                                        PORTDbits.RD3==0 && interupt==0)
                                                {
                                                        Counter1=ReadTimer1();
                                                        Counter2=ReadTimer0();
                                                if(Counter1< Counter2)
                                                                {
                                                                        Counter1=Counter1+(Counter2-Counter1);
                                                                }
                                                        else if(Counter1> Counter2)
                                                                {
                                                                        Counter2=Counter2+(Counter1-Counter2);
                                                                }
                                                        checksensors(side);
                                                        SetDCPWM2(Lspeed);
                                                        SetDCPWM1(Rspeed);
                                                }
                                }
                        SetDCPWM1(0);
                        SetDCPWM2(0);
                        WriteTimer1(0);
                        WriteTimer0(0);
                        Counter1=0;
                        Counter2=0;
                        interupt=0;
                        StopMotors=0;
        }
}

void MoveB(int countcm)
{
        unsigned int count=0,Counter1=0,Counter2=0,Rwheel=0,Lwheel=0,Rspeed=0,Lspeed=0;
        int Sens3=0,Sens4=0;
        char side;
        count=countcm*21;
        Rwheel=count;
        Lwheel=count;
        Rspeed=396;
        Lspeed=396;
        PORTBbits.RB4=0; //left
        PORTBbits.RB5=0; //right
        side='B';

        SetChanADC(  ADC_CH2 );
        ConvertADC();
        while(BusyADC());
        Sens3=ReadADC();
        SetChanADC(  ADC_CH3 );
        ConvertADC();
        while(BusyADC());
        Sens4=ReadADC();
        if(Sens3<=16 || Sens4<=16)
                {
                        SetDCPWM2(0);
                        SetDCPWM1(0);
                }
        else
```

```
		{
		WriteTimer1(0);
		WriteTimer0(0);
		SetDCPWM2(Lspeed);
		OpenPWM2(99);
		SetDCPWM1(Rspeed);
		OpenPWM1(99);
		if(countcm<=8)
			{
			while(Counter1<=Rwheel && Counter2<=Lwheel && StopMotors==0 && interupt==0)
				{
					Counter1=ReadTimer1();
				Counter2=ReadTimer0();
					if(Counter1< Counter2)
						{
							Counter1=Counter1+(Counter2-Counter1);
						}
					else if(Counter1> Counter2)
						{
							Counter2=Counter2+(Counter1-Counter2);
						}
					checksensors(side);
					SetDCPWM2(Lspeed);
					SetDCPWM1(Rspeed);
				}
			}
		else
			{
				Delay10KTCYx(80);
				while(Counter1<=Rwheel && Counter2<=Lwheel && StopMotors==0 && PORTDbits.RD0==1 &&
				PORTDbits.RD3==1 && interupt==0)
				{
					Counter1=ReadTimer1();
					Counter2=ReadTimer0();
					if(Counter1< Counter2)
						{
							Counter1=Counter1+(Counter2-Counter1);
						}
					else if(Counter1> Counter2)
						{
							Counter2=Counter2+(Counter1-Counter2);
						}
					checksensors(side);
					SetDCPWM2(Lspeed);
					SetDCPWM1(Rspeed);
				}
			}
		SetDCPWM1(0);
		SetDCPWM2(0);
		WriteTimer1(0);
		WriteTimer0(0);
		Counter1=0;
		Counter2=0;
		interupt=0;
		StopMotors=0;
	}
}


void RotateCW(int degree)
{
		unsigned int count=0,Counter1=0,Counter2=0,Rwheel=0,Lwheel=0,Rspeed=0,Lspeed=0;
```

```
		count=degree*7.25;
		Rwheel=count;
		Lwheel=count;
		Rspeed=320;
		Lspeed=320;
		PORTBbits.RB4=1;
		PORTBbits.RB5=0;
		WriteTimer1(0);
		WriteTimer0(0);
		Counter1=0;
		Counter2=0;
		SetDCPWM2(Lspeed);
		OpenPWM2(99);
		SetDCPWM1(Rspeed);
		OpenPWM1(99);
		while(Counter1<=Rwheel && Counter2<=Lwheel)
			{
				Counter1=ReadTimer1();
		Counter2=ReadTimer0();
				if(Counter1< Counter2)
					{
						Counter1=Counter1+(Counter2-Counter1);
					}
				else if(Counter1> Counter2)
					{
						Counter2=Counter2+(Counter1-Counter2);
					}
				SetDCPWM2(Lspeed);
				SetDCPWM1(Rspeed);
			}
		SetDCPWM1(0);
		SetDCPWM2(0);
		WriteTimer1(0);
		WriteTimer0(0);
		Counter1=0;
		Counter2=0;
}

void RotateCCW(int degree)
{
		unsigned int count=0,Counter1=0,Counter2=0,Rwheel=0,Lwheel=0,Rspeed=0,Lspeed=0;
		count=degree*7.25;
		Rwheel=count;
		Lwheel=count;
		Rspeed=320;
		Lspeed=320;
		PORTBbits.RB4=0;
		PORTBbits.RB5=1;
		WriteTimer1(0);
		WriteTimer0(0);
		Counter1=0;
		Counter2=0;
		SetDCPWM2(Lspeed);
		OpenPWM2(99);
		SetDCPWM1(Rspeed);
		OpenPWM1(99);
		while(Counter1<=Rwheel && Counter2<=Lwheel)
			{
				Counter1=ReadTimer1();
		Counter2=ReadTimer0();
				if(Counter1< Counter2)
					{
```

```
                                        Counter1=Counter1+(Counter2-Counter1);
                                }
                        else if(Counter1> Counter2)
                                {
                                        Counter2=Counter2+(Counter1-Counter2);
                                }
                        SetDCPWM2(Lspeed);
                        SetDCPWM1(Rspeed);
                }
        SetDCPWM1(0);
        SetDCPWM2(0);
        WriteTimer1(0);
        WriteTimer0(0);
        Counter1=0;
        Counter2=0;
}

void avoidfront(void)
        {
        int Sens1=0,Sens2=0,Sens3=0,Sens4=0,Sens5=0,Sens6=0,Sens7=0,Sens8=0,count=0 ,Counter1=0,Counter2=0;
                SetDCPWM1(0);
                SetDCPWM2(0);
                Delay10KTCYx(200);
                SetChanADC(  ADC_CH0 );
                ConvertADC();
                while(BusyADC());
                Sens1=ReadADC();
                SetChanADC(  ADC_CH1 );
                ConvertADC();
                while(BusyADC());
                Sens2=ReadADC();
                if(Sens1<=16 || Sens2<=16)
                        {
                                SetChanADC(  ADC_CH4 );
                                ConvertADC();
                                while(BusyADC());
                                Sens5=ReadADC();
                                SetChanADC(  ADC_CH5 );
                                ConvertADC();
                                while(BusyADC());
                                Sens6=ReadADC();
                                SetChanADC(  ADC_CH6 );
                                ConvertADC();
                                while(BusyADC());
                                Sens7=ReadADC();
                                SetChanADC(  ADC_CH7 );
                                ConvertADC();
                                while(BusyADC());
                                Sens8=ReadADC();
                                if(Sens5<=12 || Sens6<=12)
                                        {
                                                SetChanADC(  ADC_CH6 );
                                                ConvertADC();
                                                while(BusyADC());
                                                Sens7=ReadADC();
                                                SetChanADC(  ADC_CH7 );
                                                ConvertADC();
                                                while(BusyADC());
                                                Sens8=ReadADC();
                                                if(Sens7<=12 || Sens8<=12)
                                                        {
                                                                while(Sens7<=12 || Sens8<=12)
```

```
                                        {
                                                SetChanADC(  ADC_CH6 );
                                                ConvertADC();
                                                while(BusyADC());
                                                Sens7=ReadADC();
                                                SetChanADC(  ADC_CH7 );
                                                ConvertADC();
                                                while(BusyADC());
                                                Sens8=ReadADC();
                                                MoveB(30);
                                        }
                                RotateCCW(85);
                                interupt=1;
                        }
                else
                        {
                                RotateCCW(85);
                                MoveF(20);
                                RotateCCW(85);
                                interupt=1;

                        }
        }
else if(Sens7<=17 || Sens8<=17)
        {
                SetChanADC(  ADC_CH4 );
                ConvertADC();
                while(BusyADC());
                Sens5=ReadADC();
                SetChanADC(  ADC_CH5 );
                ConvertADC();
                while(BusyADC());
                Sens6=ReadADC();
                if(Sens5<=17 || Sens6<=17)
                        {
                                while(Sens7<=17 || Sens8<=17)
                                        {
                                                SetChanADC(  ADC_CH6 );
                                                ConvertADC();
                                                while(BusyADC());
                                                Sens7=ReadADC();
                                                SetChanADC(  ADC_CH7 );
                                                ConvertADC();
                                                while(BusyADC());
                                                Sens8=ReadADC();
                                                MoveB(30);
                                        }
                                RotateCCW(85);
                                interupt=1;
                        }
                else
                        {
                                RotateCW(85);
                                MoveF(20);
                                RotateCW(85);
                                interupt=1;

                        }
        }
else
        {
                RotateCCW(170);
```

```
                                        MoveF(20);
                                        interupt=1;
                                }
                }
        }

void avoidback(void)
        {
        int Sens1=0,Sens2=0,Sens3=0,Sens4=0;
                SetDCPWM1(0);
                SetDCPWM2(0);
                Delay10KTCYx(200);
                        SetChanADC(  ADC_CH2 );
                        ConvertADC();
                        while(BusyADC());
                        Sens3=ReadADC();
                        SetChanADC(  ADC_CH3 );
                        ConvertADC();
                        while(BusyADC());
                        Sens4=ReadADC();
                        if(Sens3<=16 || Sens4<=16)
                                {
                                        MoveF(30);
                                        interupt=1;
                                }
        }

void checksensors(char side)
        {
                int Sens1=0,Sens2=0,Sens3=0,Sens4=0,Sens5=0,Sens6=0,Sens7=0,Sens8=0;
                switch(side)
                        {
                                case 'F': {
                                                SetChanADC(  ADC_CH0 );
                                                ConvertADC();
                                                while(BusyADC());
                                                Sens1=ReadADC();
                                                SetChanADC(  ADC_CH1 );
                                                ConvertADC();
                                                while(BusyADC());
                                                Sens2=ReadADC();
                                                if(Sens1<=16 || Sens2<=16)
                                                        {
                                                                SetChanADC(  ADC_CH2 );
                                                                ConvertADC();
                                                                while(BusyADC());
                                                                Sens3=ReadADC();
                                                                SetChanADC(  ADC_CH3 );
                                                                ConvertADC();
                                                                while(BusyADC());
                                                                Sens4=ReadADC();
                                                                if(Sens3<=16 || Sens4<=16)
                                                                        {
                                                                                SetDCPWM2(0);
                                                                                SetDCPWM1(0);
                                                                        Delay10KTCYx(200);
                                                                                RotateCCW(90);
                                                                                interupt=1;

                                                                        }
                                                                else
                                                                        {
```

```
                                                                avoidfront();

                                                                }
                                                        }
                                                break;
                                        }
                                case 'B': {
                                                SetChanADC(  ADC_CH2 );
                                                ConvertADC();
                                                while(BusyADC());
                                                Sens3=ReadADC();
                                                SetChanADC(  ADC_CH3 );
                                                ConvertADC();
                                                while(BusyADC());
                                                Sens4=ReadADC();
                                                if(Sens3<=13 || Sens4<=13)
                                                        {
                                                                SetChanADC(  ADC_CH0 );
                                                                ConvertADC();
                                                                while(BusyADC());
                                                                Sens1=ReadADC();
                                                                SetChanADC(  ADC_CH1 );
                                                                ConvertADC();
                                                                while(BusyADC());
                                                                Sens2=ReadADC();
                                                                if(Sens1<=16 || Sens2<=16)
                                                                        {
                                                                                SetDCPWM2(0);
                                                                                SetDCPWM1(0);
                                                                        Delay10KTCYx(200);
                                                                                RotateCCW(90);
                                                                                interupt=1;
                                                                        }
                                                                else
                                                                        {
                                                                                avoidback();
                                                                        }
                                                        }
                                                break;
                                        }
                        }
        }

#pragma interrupt aa
void aa(void)
{
        unsigned char c;
        char  u ;
        int i=0, distance=0, degree=0, f1=0,f2=0,f3=0;
        char data[8], status;
        if(PIR1bits.RCIF)
         {
                for(i=0;i<=8;i++)
                  {
                                while(!DataRdyUSART());
                        data[i] = ReadUSART();
                  }
                if(data[0]=='M')
                  {
                        if(data[1]=='F')
                        {
                          f1=ToDecimal(data[2])*100;
```

```
                f2=ToDecimal(data[3])*10;
                f3=ToDecimal(data[4]);
                distance=f1+f2+f3;
                        MoveF(distance);
                        for(i=0;i<=8;i++)
                                {
                                        while(BusyUSART()==1);
                                        u = data[i];
                                        putcUSART(u);
                                }
                }
        else if(data[1]=='B')                           {
                f1=ToDecimal(data[2])*100;
                f2=ToDecimal(data[3])*10;
                f3=ToDecimal(data[4]);
                distance=f1+f2+f3;
                        MoveB(distance);
                        for(i=0;i<=8;i++)
                                {
                                        while(BusyUSART()==1);
                                        u = data[i];
                                        putcUSART(u);
                                }
                }
        else if(data[1]=='T')                           {
                f1=ToDecimal(data[2])*100;
                f2=ToDecimal(data[3])*10;
                f3=ToDecimal(data[4]);
                nX=f1+f2+f3;
                f1=ToDecimal(data[5])*100;
                f2=ToDecimal(data[6])*10;
                f3=ToDecimal(data[7]);
                nY=f1+f2+f3;
                        CalculateXY(nX,nY);
                        for(i=0;i<=8;i++)
                                {
                                        while(BusyUSART()==1);
                                        u = data[i];
                                        putcUSART(u);
                                }
                }
                }
      else if(data[0]=='R')
        {
                if(data[1]=='T')
                {
                        if(data[2]=='1')
                         {
                f1=ToDecimal(data[3])*100;
                f2=ToDecimal(data[4])*10;
                f3=ToDecimal(data[5]);
                degree=f1+f2+f3;
                        RotateCW(degree);
                        for(i=0;i<=8;i++)
                                {
                                        while(BusyUSART()==1);
                                        u = data[i];
                                        putcUSART(u);
                                }
                   }
else if(data[2]=='0')                                                   {
                f1=ToDecimal(data[3])*100;
```

```
                f2=ToDecimal(data[4])*10;
                f3=ToDecimal(data[5]);
                degree=f1+f2+f3;
                        RotateCCW(degree);
                        for(i=0;i<=8;i++)
                                {
                                        while(BusyUSART()==1);
                                        u = data[i];
                                        putcUSART(u);
                                }
                        }
                }
else if(data[1]=='C')
                {
                if(data[2]=='1')
                {
                 f1=ToDecimal(data[3])*100;
                 f2=ToDecimal(data[4])*10;
                 f3=ToDecimal(data[5]);
                 degree=f1+f2+f3;
                        stepper('a');
                        for(i=0;i<=8;i++)
                                {
                                        while(BusyUSART()==1);
                                        u = data[i];
                                        putcUSART(u);
                                }
                   }
    else if(data[2]=='0')
                        {
                 f1=ToDecimal(data[3])*100;
                 f2=ToDecimal(data[4])*10;
                 f3=ToDecimal(data[5]);
                 degree=f1+f2+f3;
                        stepper('c');
                                for(i=0;i<=8;i++)
                                {
                                        while(BusyUSART()==1);
                                        u = data[i];
                                        putcUSART(u);
                                }
                        }
                }
        else if(data[1]=='M')
                {
                if(data[2]=='1')
                         {
                 f1=ToDecimal(data[3])*100;
                 f2=ToDecimal(data[4])*10;
                 f3=ToDecimal(data[5]);
                 degree=f1+f2+f3;
                 f1=ToDecimal(data[6])*100;
                 f2=ToDecimal(data[7])*10;
                 f3=ToDecimal(data[8]);
                 distance=f1+f2+f3;
                        RotateCW(degree);
                        MoveF(distance);
                        for(i=0;i<=8;i++)
                                {
                                        while(BusyUSART()==1);
                                        u = data[i];
                                        putcUSART(u);
```

```
                                        }
                                }
                else if(data[2]=='0')
                                        {
                                         f1=ToDecimal(data[3])*100;
                                         f2=ToDecimal(data[4])*10;
                                         f3=ToDecimal(data[5]);
                                         degree=f1+f2+f3;
                                         f1=ToDecimal(data[3])*100;
                                         f2=ToDecimal(data[4])*10;
                                        f3=ToDecimal(data[5]);
                                        distance=f1+f2+f3;
                                       RotateCCW(degree);
                                       MoveF(distance);
                                       for(i=0;i<=8;i++)
                                                {
                                                        while(BusyUSART()==1);
                                                        u = data[i];
                                                        putcUSART(u);
                                                }
                                       }
                        }
                }
        else if(data[0]=='S' && data[1]=='T' && data[2]=='O' && data[3]=='P' && data[4]=='M')
                {
                                                SetDCPWM1(0);
                                                SetDCPWM2(0);
                                                StopMotors=1;
                }
                                PIR1bits.RCIF = 0;
        }
}




#pragma code high_vector=0x08
void high_vector (void)
{ _asm goto aa _endasm }
#pragma code

void main(void)
{
        unsigned int spbrg;
        RCONbits.IPEN=1; // IPEN=1
        INTCONbits.GIEH = 1;
        PIR1bits.RCIF = 0;
        OSCCON=126;
        spbrg = 12 ;
        ADCON1=7;
        TRISB=0;
        PORTB=0;
        TRISD=15;
        PORTD=0;
        SetDCPWM1(0);
        ClosePWM1();
        SetDCPWM2(0);
        ClosePWM2();

OpenUSART(USART_TX_INT_OFF & USART_RX_INT_ON & USART_ASYNCH_MODE & USART_EIGHT_BIT &
USART_CONT_RX & USART_BRGH_HIGH,spbrg);
```

```
OpenADC( ADC_FOSC_32&ADC_RIGHT_JUST& ADC_12_TAD,ADC_CH0 &ADC_REF_VDD_VSS &
ADC_INT_OFF,ADC_8ANA);
OpenTimer3(TIMER_INT_OFF & T3_16BIT_RW & T3_SOURCE_INT &  T3_PS_1_1 &  T1_CCP1_T3_CCP2);
OpenTimer2(TIMER_INT_OFF & T2_PS_1_4 & T2_POST_1_1 &    T1_CCP1_T3_CCP2);
OpenTimer1(TIMER_INT_OFF & T1_16BIT_RW &  T1_PS_1_1 & T1_OSC1EN_OFF &  T1_SYNC_EXT_ON & T1_SOURCE_EXT
&  T1_CCP1_T3_CCP2 );
OpenTimer0( TIMER_INT_OFF & T0_16BIT &  T0_SOURCE_EXT & T0_EDGE_RISE & T0_PS_1_1 );

while(1)
        {
                SetDCPWM1(0);
                SetDCPWM2(0);
        }
}
```

Appendix E

- **CE Series Incremental Rotary Encoder**

LINE SEIKI

CE SERIES

INCREMENTAL ROTARY ENCODER

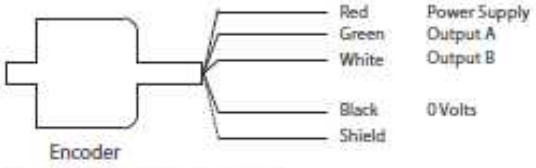
Outside Diameter 30mm (Shaft Φ4) PCD 22
Economical
Voltage output type
Suitable for General or Industrial Applications

STANDARD MODELS

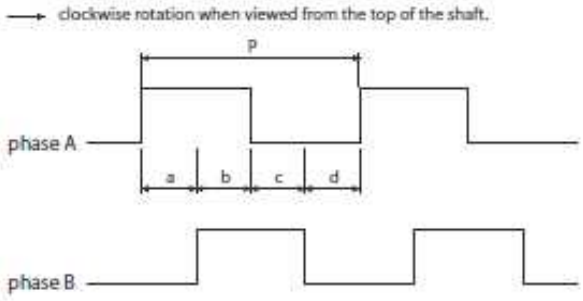
PPR \ Serial Number	CE (30ϕ)
60	CE – 60
100	CE – 100
200	CE – 200
250	CE – 250
300	CE – 300
360	CE – 360
400	CE – 400
500	CE – 500

Add " S " at the end of Model No. for 1 signal output (only signal A).
ex. CE–100S

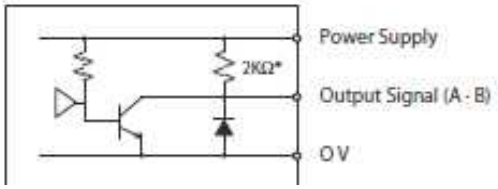
CONNECTIONS



OUTPUT WAVEFORM



OUTPUT WIRING



SPECIFICATIONS

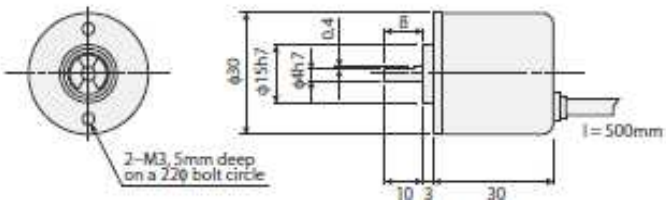
		CE
Electrical	Power Supply	5 ~ 12V DC ±5%
	Output Signal	Voltage / 90° Quadrature x 2 Signals
	Output Voltage	Logic 1 = 4 to 11 VDC (Reduced - 1V from Power Supply) Logic 0 = 0.5 VDC or less
	Maximum Response	60 KHz
	Current Consumption	40 mA maximum (20 mA for S Type)
	Sink Current	20 mA maximum
	Output Impedance	2 KΩ
	Operating Temperature	-10°C ~ +70°C
Mechanical	Maximum Speed of Shaft Input	6,000 rpm
	Maximum of Inertia of Shaft	2g-cm^2 Maximum
	Starting Torque	10gf-cm Maximum
	Angular Speed	1×10^3 rad / sec^2
	Maximum Radial Loading	1 Kg
	Maximum Axial Loading	0.5 Kg
	Vibration	10Hz to 50Hz • 1.5mm•2h
	Shock	50 G / 11 ms
	Weight (Approximately)	100 grams

- Signal A and Signal B are 90° quadrature
- $P = \frac{1}{PPR}$
- Accuracy

$$a \cdot b \cdot c \cdot d = \frac{P}{4} \pm \frac{P}{8}$$

$$h = P \pm \frac{P}{2}$$

DIMENSIONS



Appendix F

Testing Codes

F.1 Open Loop Response

```
#include<p18f4550.h>
#include<delays.h>
#include<PWM.h>
#include<timers.h>
#pragma config FOSC = INTOSC_HS
#pragma config WDT = OFF
void main (void)
{
TRISD=4;
 PORTD=0;
TRISC=0;
PORTC=0;
TRISB=0;
PORTB=0;
OpenTimer3(TIMER_INT_OFF & T3_16BIT_RW & T3_SOURCE_INT & T3_PS_1_1 & T3_SOURCE_CCP);
  OpenTimer2(TIMER_INT_OFF & T2_PS_1_4 & T2_POST_1_1 &    T3_SOURCE_CCP);
while(1)
{
                                /*Forward*/
                                PORTBbits.RB4=1;
                                PORTBbits.RB5=1;
                                SetDCPWM1(396);
                                OpenPWM1(99);
                                SetDCPWM2(396);
                                OpenPWM2(99);
                                Delay10KTCYx(100);
                                /*Left*/
                                PORTBbits.RB4=1;
                                PORTBbits.RB5=1;
                                SetDCPWM1(396);
                                OpenPWM1(99);
                                SetDCPWM2(198);
                                OpenPWM2(99);
                                Delay10KTCYx(100);
                                /*Backward*/
                                PORTBbits.RB5=0;
                                PORTBbits.RB4=0;
                                SetDCPWM1(396);
                                OpenPWM1(99);
                                SetDCPWM2(396);
                                OpenPWM2(99);
                                Delay10KTCYx(100);
                                /*Right*/
                                PORTBbits.RB4=1;
                                PORTBbits.RB5=1;
                                SetDCPWM1(198);
                                OpenPWM1(99);
                                SetDCPWM2(396);
                                OpenPWM2(99);
                                Delay10KTCYx(100);
}
}
```

F.2 Closed Loop Response

```
#include<p18f4550.h>
#include<delays.h>
```

```
#include<PWM.h>
#include<timers.h>
#pragma config FOSC = INTOSC_HS
#pragma config WDT = OFF

void main(void)
{
unsigned int Counter1=0,Counter2=0;
ADCON1=6;
TRISB=0;
PORTB=0;

OpenTimer3(TIMER_INT_OFF & T3_16BIT_RW & T3_SOURCE_INT &  T3_PS_1_1 &  T1_CCP1_T3_CCP2);
OpenTimer2(TIMER_INT_OFF & T2_PS_1_4 & T2_POST_1_1 &    T1_CCP1_T3_CCP2);
OpenTimer1(TIMER_INT_OFF & T1_16BIT_RW &  T1_PS_1_1 & T1_OSC1EN_OFF &  T1_SYNC_EXT_ON & T1_SOURCE_EXT
&  T1_CCP1_T3_CCP2 );
WriteTimer1(0);
OpenTimer0( TIMER_INT_OFF & T0_16BIT &  T0_SOURCE_EXT & T0_EDGE_RISE & T0_PS_1_1 );
WriteTimer0(0);
        while(1)
                {
                PORTBbits.RB4=1;
                PORTBbits.RB5=1;
                WriteTimer1(0);
                WriteTimer0(0);
                SetDCPWM2(396);
                SetDCPWM1(396);
                while(Counter1<=1910 || Counter2<=1910)
                        {
                        Counter1=ReadTimer1();  //  Right Wheel Steps  1cm = 9.55 Counts
                        Counter2=ReadTimer0();  //  Left Wheel Steps   1cm = 9.55 Counts
                        }
                WriteTimer1(0);
                 WriteTimer0(0);
                Counter1=0;
                Counter2=0;
                        }
{
                PORTBbits.RB4=1;
                PORTBbits.RB5=1;
                WriteTimer1(0);
                WriteTimer0(0);
                SetDCPWM2(396);
                SetDCPWM1(396);
                while(Counter1<=200 || Counter2<=200)
                        {
                        Counter1=ReadTimer1();  //  Right Wheel Steps  1cm = 9.55 Counts
                        Counter2=ReadTimer0();  //  Left Wheel Steps   1cm = 9.55 Counts
                        }
                WriteTimer1(0);
                 WriteTimer0(0);
                Counter1=0;
                Counter2=0;
                        }
{
                PORTBbits.RB4=1;
                PORTBbits.RB5=1;
                WriteTimer1(0);
                WriteTimer0(0);
                SetDCPWM2(396);
                SetDCPWM1(396);
                while(Counter1<=1433 || Counter2<=1433)
```

```
                    {
                    Counter1=ReadTimer1();  //  Right Wheel Steps  1cm = 9.55 Counts
                    Counter2=ReadTimer0();  //  Left Wheel Steps   1cm = 9.55 Counts
                    }
                WriteTimer1(0);
                 WriteTimer0(0);
                Counter1=0;
                Counter2=0;
                    }
{
                PORTBbits.RB4=1;
                PORTBbits.RB5=1;
                WriteTimer1(0);
                WriteTimer0(0);
                SetDCPWM2(396);
                SetDCPWM1(396);
                while(Counter1<=200 || Counter2<=200)
                    {
                    Counter1=ReadTimer1();  //  Right Wheel Steps  1cm = 9.55 Counts
                    Counter2=ReadTimer0();  //  Left Wheel Steps   1cm = 9.55 Counts
                    }
                WriteTimer1(0);
                 WriteTimer0(0);
                Counter1=0;
                Counter2=0;
                    }
{
                PORTBbits.RB4=1;
                PORTBbits.RB5=1;
                WriteTimer1(0);
                WriteTimer0(0);
                SetDCPWM2(396);
                SetDCPWM1(396);
                while(Counter1<=955 || Counter2<=955)
                    {
                    Counter1=ReadTimer1();  //  Right Wheel Steps  1cm = 9.55 Counts
                    Counter2=ReadTimer0();  //  Left Wheel Steps   1cm = 9.55 Counts
                    }
                WriteTimer1(0);
                 WriteTimer0(0);
                Counter1=0;
                Counter2=0;        }
```

F.3 Checking for Obstacle

```
#include<p18f4550.h>
#include<delays.h>
#include<adc.h>
#include<PWM.h>
#include<timers.h>
#include<compare.h>
#pragma config FOSC = INTOSC_HS
#pragma config WDT = OFF
void main(void)
{
                    unsigned int Counter1=0,Counter2=0;
                    ADCON1=6;
                    TRISB=0;
                    PORTB=0;
OpenADC( ADC_FOSC_32&ADC_RIGHT_JUST& ADC_12_TAD,ADC_CH0 &ADC_REF_VDD_VSS &
ADC_INT_OFF,ADC_8ANA);
```

```
                        while(1)
                        {
                        int Sens1=0,Sens2=0,Sens3=0,Sens4=0;
                        SetChanADC(  ADC_CH0 );
                        ConvertADC();
                        while(BusyADC());
                        Sens1=ReadADC();
                        if(Sens1<=16)
                                {
                                        SetDCPWM1(0);
                                        SetDCPWM2(0);
                                        Delay10KTCYx(50);
                                }
                        SetChanADC(  ADC_CH1 );
                        ConvertADC();
                        while(BusyADC());
                        Sens2=ReadADC();
                        if(Sens2<=16)
                                {
                                        SetDCPWM1(0);
                                        SetDCPWM2(0);
                                        Delay10KTCYx(50);
                                }
                        SetChanADC(  ADC_CH2 );
                        ConvertADC();
                        while(BusyADC());
                        Sens3=ReadADC();
                        if(Sens3<=16)
                                {
                                        SetDCPWM1(0);
                                        SetDCPWM2(0);
                                        Delay10KTCYx(50);
                                }
                        SetChanADC(  ADC_CH3 );
                        ConvertADC();
                        while(BusyADC());
                        Sens4=ReadADC();
                        if(Sens4<=16)
                                {
                                        SetDCPWM1(0);
                                        SetDCPWM2(0);
                                        Delay10KTCYx(50);
                                }
        }}}
```

F.4 Combining the Encoders with Ultra Sonic

```
#include<p18f4550.h>
#include<delays.h>
#include<adc.h>
#include<PWM.h>
#include<timers.h>
#include<compare.h>
#pragma config FOSC = INTOSC_HS
#pragma config WDT = OFF
void checksensors(char side);
void Move(char dir, int countcm)
{
        unsigned int count,Counter1=0,Counter2=0,Rwheel=0,Lwheel=0,Rspeed=0,Lspeed=0;
        char side;
```

```
	count=countcm*9.55;
	switch(dir)
		{
			case 'F': {
					Rwheel=count;
					Lwheel=count;
					Rspeed=396;	//	5V
					Lspeed=396;	//	5V
					PORTBbits.RB4=1;
					PORTBbits.RB5=1;
					side='F';
					break;
				}
			case 'B': {
					Rwheel=count;
					Lwheel=count;
					Rspeed=396;	//	5V
					Lspeed=396;	//	5V
					PORTBbits.RB4=0;
					PORTBbits.RB5=0;
					side='B';
					break;
				}
			}
		SetDCPWM2(Lspeed);
		OpenPWM2(99);
		SetDCPWM1(Rspeed);
		OpenPWM1(99);
		WriteTimer1(0);
		WriteTimer0(0);
	while(Counter1<=Rwheel || Counter2<=Lwheel)
		{
			Counter1=ReadTimer1();  // Right Wheel Steps  1cm = 9.55 Counts
			Counter2=ReadTimer0();  // Left Wheel Steps   1cm = 9.55 Counts
			checksensors(side);
		}
	WriteTimer1(0);
	WriteTimer0(0);
	Counter1=0;
	Counter2=0;
}
void Rotate(char clock, int degree)
{
//Rotation parameter: clock:c=clockwise,a=anticlockwise, degree=0-360;
unsigned int count=0,Counter1=0,Counter2=0,Rwheel=0,Lwheel=0,Rspeed=0,Lspeed=0;
		switch(clock)
			{
				case 'c':
					{
						count=degree*1.5;
				Rwheel=count;
						Lwheel=count;
						Rspeed=300;
						Lspeed=300;
						PORTBbits.RB4=1;
						PORTBbits.RB5=0;
						break;
					}
				case 'a':
					{
						count=degree*1.5;
						Rwheel=count;
```

```
                                                Lwheel=count;
                                                Rspeed=300;
                                                Lspeed=300;
                                                PORTBbits.RB4=0;
                                                PORTBbits.RB5=1;
                                                break;
                                        }
                        }
                SetDCPWM2(Lspeed);
                OpenPWM2(99);
                SetDCPWM1(Rspeed);
                OpenPWM1(99);
                WriteTimer1(0);
                WriteTimer0(0);
                while(Counter1<=count || Counter2<=count)
                        {
                        Counter1=ReadTimer1();  // Right Wheel Steps  1cm = 9.55 Counts
                        Counter2=ReadTimer0();  // Left Wheel Steps   1cm = 9.55 Counts
                        }
                WriteTimer1(0);
                WriteTimer0(0);
                Counter1=0;
                Counter2=0;
}
void avoidfront(void)
        {
                Move('B',50);
                Rotate('a',180);
        }
void avoidback(void)
        {
                Move('F',50);
        }
void checksensors(char side)
        {
        int Sens1=0,Sens2=0,Sens3=0,Sens4=0;
        switch(side)
                {
                        case 'F': {
                        SetChanADC(  ADC_CH0 );
                        ConvertADC();
                        while(BusyADC());
                        Sens1=ReadADC();
                        SetChanADC(  ADC_CH1 );
                        ConvertADC();
                        while(BusyADC());
                        Sens2=ReadADC();
                        if(Sens1<=16 || Sens2<=16)
                                {
                                        avoidfront();
                                }
                        break;
                                        }
                        case 'B': {
                        SetChanADC(  ADC_CH2 );
                        ConvertADC();
                        while(BusyADC());
                        Sens3=ReadADC();
                        SetChanADC(  ADC_CH3 );
                        ConvertADC();
                        while(BusyADC());
                        Sens4=ReadADC();
```

```
                        if(Sens3<=16 || Sens4<=16)
                                {
                                        avoidback();
                                }
                        break;
                                        }
                        }
        }
void main(void)
{
unsigned int Counter1=0,Counter2=0;
ADCON1=6;
TRISB=0;
PORTB=0;
OpenADC( ADC_FOSC_32&ADC_RIGHT_JUST& ADC_12_TAD,ADC_CH0
&ADC_REF_VDD_VSS&ADC_INT_OFF,ADC_8ANA);
OpenTimer3(TIMER_INT_OFF & T3_16BIT_RW & T3_SOURCE_INT &  T3_PS_1_1 &  T1_CCP1_T3_CCP2);
OpenTimer2(TIMER_INT_OFF & T2_PS_1_4 & T2_POST_1_1 &    T1_CCP1_T3_CCP2);
OpenTimer1(TIMER_INT_OFF & T1_16BIT_RW &  T1_PS_1_1 & T1_OSC1EN_OFF &  T1_SYNC_EXT_ON & T1_SOURCE_EXT
&  T1_CCP1_T3_CCP2 );
WriteTimer1(0);
OpenTimer0( TIMER_INT_OFF & T0_16BIT &  T0_SOURCE_EXT & T0_EDGE_RISE & T0_PS_1_1 );
WriteTimer0(0);
while(1)
        {Move('F',200);}}
```

Appendix G

- **Datasheets For WiFly GSX Transceiver & ITM-C-328 Serial Camera**
- **& EZ0 MaxSonar Ultrasonic Sensor**

ROVING networks
WIRELESS FOR LESS

RN-131G

www.rovingnetworks.com

m-131-ds v2.3 5/28/2009

"WiFly GSX" 802.11G Module

Features

- Qualified 2.4GHz IEEE 802.11b/g transceiver
- High throughput, 1Mbps sustained data rate with TCP/IP and WPA2
- Ultra-low power - 4uA sleep, 40mA Rx, 210mA Tx (max)
- Small, compact surface mount module
- On board ceramic chip antenna and U.FL connector for external antenna
- 8 Mbit flash memory and 128 KB RAM
- UART hardware interface
- 10 general purpose digital I/O
- 8 analog sensor interfaces
- Real-time clock for wakeup and time stamping
- Accepts 3.3V regulated or 2-3V battery
- Supports Adhoc connections
- On board ECOS -OS, TCP/IP stacks
- Wi-Fi Alliance certified for WPA2-PSK
- FCC / CE/ ICS certified and RoHS compliant.

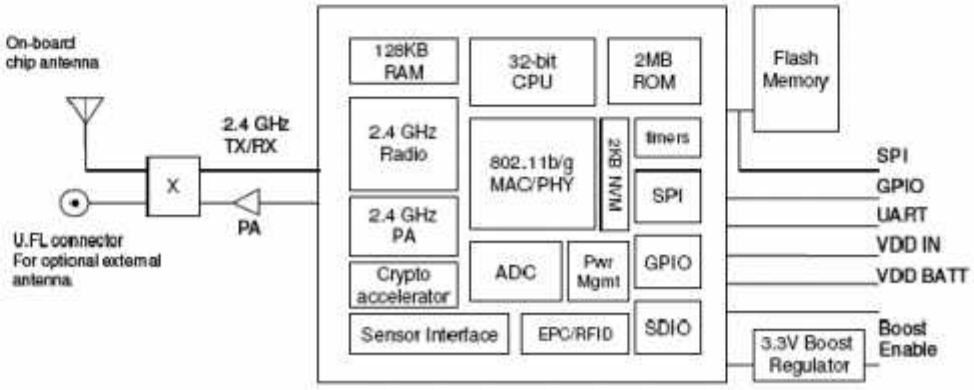
Applications

- Wireless audio
- Remote equipment monitoring
- Telemetry
- Security
- Industrial sensors and controls
- Home Automation
- Medical devices

Description

The WiFly GSX module is a stand alone, embedded wireless 802.11 networking module. Because of its small form factor and extremely low power consumption, the RN-131G is perfect for mobile wireless applications such as asset monitoring, GPS tracking and battery sensors. The WiFly GSX module incorporates a 2.4GHz radio, processor, TCP/IP stack, real-time clock, crypto accelerator, power management and analog sensor interfaces. This complete solution is preloaded with software to simplify integration and minimizes development of your application. In the simplest configuration the hardware only requires four connections (PWR, TX, RX, GND) to create a wireless data connection. Additionally, the sensor interface provides temperature, audio, motion, acceleration and other analog data without requiring additional hardware. The WiFly GSX module is programmed and controlled with a simple ASCII command language. Once the WiFly GSX is setup it can scan to find an access point, associate, authenticate and connect over any Wifl network.

Block Diagram



Overview

- Host Data Rate up to 1 Mbps for UART
- Intelligent, built-in power management with programmable wakeup
- Can be powered from regulated 3.3-3.7V source or 2.0-3.0V batteries
- Real time clock for time stamping, auto-sleep and auto-wakeup
- Configuration over UART using simple ASCII commands
- Web Server or Telnet configuration over WiFi
- Over the air firmware upgrade (FTP)
- Memory 128 KB RAM,2MB ROM, 2 KB battery-backed memory, 8 Mbit Flash.
- Secure WiFi authentication WEP-128, WPA-PSK (TKIP), WPA2-PSK, EAP-TLS for WPA1 & WPA2 Enterprise
- Built in networking applications DHCP, UDP, DNS, ARP, ICMP
- 802.11 power save and roaming functions

Environmental Conditions

Parameter	Value
Temperature Range (Operating)	-40 °C ~ 85 °C
Temperature Range (Storage)	-40°C ~ 85 °C
Relative Humidity (Operating)	≤90%
Relative Humidity (Storage)	≤90%

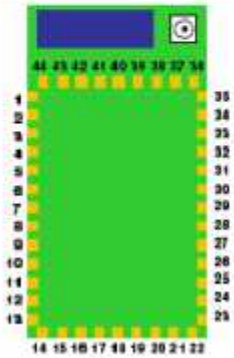
Electrical Characteristics

Supply Voltage	Min	Typ.	Max.	Unit
Supply Voltage VDD	3.0	3.3	3.7	VDC
Supply Voltage (VBATT option)	2.0	3.0	3.3	VDC
Power consumption				
Sleep		4		uA
Standby (doze)	-	15	-	mA
Connected (idle, RX)		40		mA
Connected (TX)		140	212	mA

Radio Characteristics

Parameter	Specifications
Frequency	2402 ~ 2480MHz
Modulation	DSSS(CCK-11, CCK-5.5, DQPSK-2, DBPSK-1)
Channel intervals	5MHz
Channels	1 - 14
Transmission rate (over the air)	1 – 11Mbps for 802.11b / 6 – 54Mbps for 802.11g
Receive sensitivity	-85dBm typ.
Output level (Class1)	+18dBm
Maximum RF input to U.FL connector	10 dBm

Pin Description



Topview
(pads not visible from top)

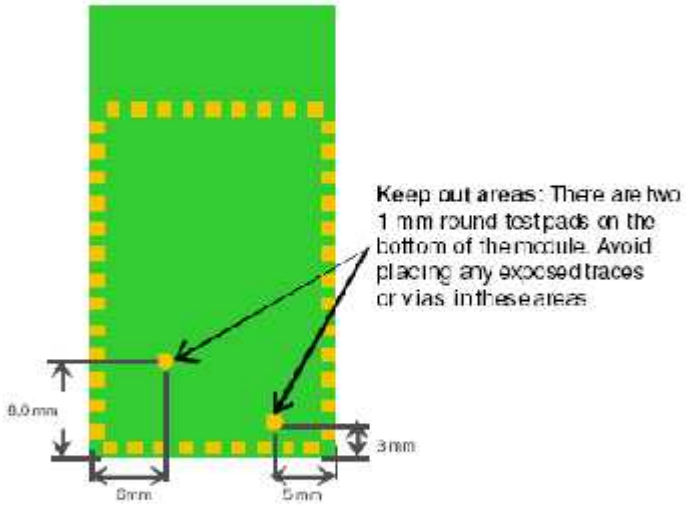
Pin	Name	Description	Default
1	SENSOR-6	Sensor interface, analog input to module, 1.2V	No connect
2	SENSOR-4	Sensor interface, Analog input to module, 1.2V	No connect
3	SENSOR-5	Sensor interface, Analog input to module, 1.2V	No connect
4	SENSOR-7	Analog input to module, 1.2V	No connect
5	RESET	Module reset, Active Low, reference to VDD-BATT, 160 min. pulse	Pull up
6	EPC-ANT-A	EPC port, RFID antenna A	No connect
7	EPC-ANT-B	EPC port, RFID antenna B	No connect
8	SUPERCAP	Balance center pin voltage on stacked super capacitors, Analog 3.3V	No connect
9	FORCE_AWAKE	Force the module to wakeup, input to module, 31us min. pulse	
10	GPIO-13	UART RTS flow control, 8mA drive, 3.3V tolerant	
11	GPIO-12	UART CTS flow control, 8mA drive, 3.3V tolerant	
12	UART-RX	RX to the module, 8mA drive, 3.3V tolerant	
13	UART-TX	TX from the module, 8mA drive, 3.3V tolerant	
14	SPI-MOSI	SPI master data out (Contact Roving Networks for details)	No connect
15	SPI-CLK	SPI clock, (Contact Roving Networks for details)	No connect
16	SPI-MISO	SPI master data in (Contact Roving Networks for details)	No connect
17	3.3V-REG-OUT	boost regulator control output, connect to 3.3V-REG-IN to enable	No connect
18	3.3V-REG-IN	boost regulator control input, connect to 3.3V-REG-OUT to enable	GND to disable
19	GND	Ground	
20	VDD-BATT	Battery input, 2.0-3.3V with boost regulator in use, 3.0-3.7V otherwise	
21	VDD-IN	3.3 to 3.7 voltage, do not connect when boost regulator is in use	
22	DMA-TX	Debug port	No connect
23	DMA-RX	Debug port	No connect
24	GPIO-9	Restore factory resets, 8mA drive, 3.3V tolerant	
25	GPIO-8	GPIO, 24mA drive, 3.3V tolerant	Weak pull down
26	GPIO-7	GPIO, 24mA drive, 3.3V tolerant	Weak pull down
27	GPIO-6	Association STATUS, 24mA drive, 3.3V tolerant	LED, Weak pull down
28	GPIO-5	Data transfer STATUS, 24mA drive, 3.3V tolerant	LED, Weak pull down
29	GPIO-4	Connection STATUS, 24mA drive, 3.3V tolerant	LED, Weak pull down
30	SENSOR-1	Sensor interface, analog input to module, 1.2V	
31	SENSOR-2	Sensor interface, analog input to module, 1.2V	
32	SENSOR-3	Sensor interface, analog input to module, 1.2V	
33	SENSE-PWR	Voltage output from module to power external sensors, 1.2-3.3V	
34	SENSOR-0	Wakeup from external condition	
35	NO CONNECT		No connect
36-44	GND	Must be connected for proper antenna performance	

5. ***Keep out areas.*** When designing your PCB avoid exposed trace and via beneath the module.

6. ***Powering the module.*** The WiFly module can be powered from either 3.0VDC batteries or 3.3VDC regulated power.

   3.0VDC battery power
   - Apply power to pin 20 (VDD-BATT)
   - Short pin 17 (3.3V-REG-OUT) to pin 18 (3.3V-REG-IN)

   3.3 VDC power
   - Apply power to pin 20(VDD-BATT) and pin 21 (VDD-IN)
   - Connect pin 18 (3.3V-REG-IN) to ground and leave pin 17 (3.3V-REG-OUT) unconnected.



Bottom view

7. ***Sensor Interfaces.*** Inputs must not exceed 1.2V. Sensitivity saturates at 400 mV.

8. ***Adhoc mode and Restoring Factory Settings.*** Adhoc mode is controlled through GPIO-9. It is a good idea to connect pin 24, GPIO-9 to a switch or jumper connected to a pull up. When GPIO-9 is driven high at power up the module will be in Adhoc mode. If GPIO-9 is then toggled low 5 times, the initial factory default configuration will be RESTORED. This is useful for cases where the module is mis-configured and is no long responding.

Compliance Information

- FCC Certified for us in the United States and CE approved for use in Europe and other countries.
- Environmentally friendly RoHS compliant

Ordering Information

Part Number	Description
RN-131G	With chip antenna
RN-131G-EVAL	Development Kit for the RN-131G (Includes the RN-131G module)
RN-UFL-SMA6	6 inch cable with U.FL connector on one end and SMA on the other
For other configurations, contact Roving Networks directly.	

Visit http://www.rovingnetworks.com for current pricing and a list of distributors carrying our products.

Copyright © 2009 Roving Networks. All rights reserved.

Roving Networks reserves the right to make corrections, modifications, and other changes to its products, documentation and services at any time. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

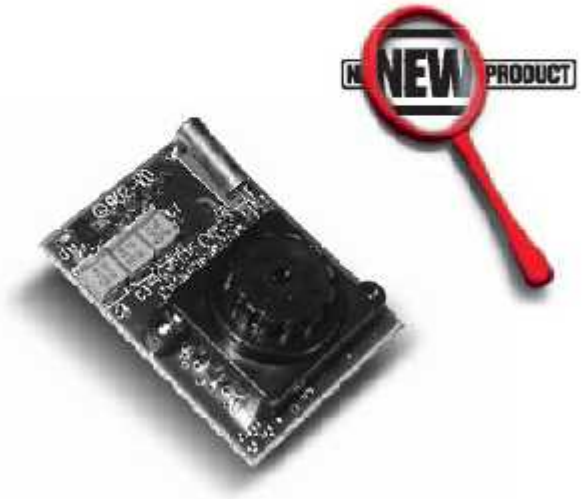
Visual Products

VISUAL

Intertec Components

ITM-C-328

Serial-Color-Cam-Module with JPEG-Compression



General Description

The ITM-C-328 JPEG compression module performs as a video camera or a JPEG compressed still camera and can be attached to a wireless or PDA host. Users can send out a snapshot command from the host in order to capture a full resolution single-frame still picture. The picture is then compressed by the JPEG engine and transferred via RS-232 to the host.

Specification:

Imager:	OV7640
Array Size:	640 x 480 pix
Pixel Size:	5,6 x 5,6 µm
Scanning :	Progressive
Image Area:	3,6 x 2,7 mm
S/N Ratio:	46 dB
Operation Voltage:	3,3 VDC
Lens:	f 4.0mm F 2.2
Dimension:	20 x 28 mm

Features:

- Low-cost, & low-powered solution for high resolution image capture
- Built-in down-sampling, clamping and windowing circuits for VGA/CIF/SIF/QCIF/160x128/80x64 image resolutions
- RS-232, 115.2K bps for transferring JPEG still pictures or 160x128 preview @8bpp with 0.75~6 fps
- JPEG CODEC with variable quality settings for different resolutions
- Built-in color conversion circuits for 4 gray/16 gray/256 gray/12-bit RGB/16-bit RGB/Pallet 256 RGB preview images

OV528 Serial Bridge

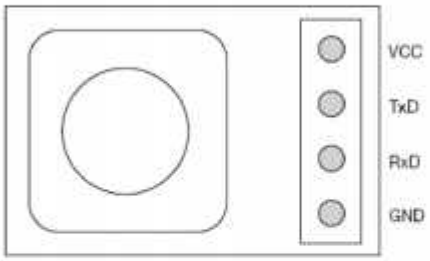
The OV528 Serial Bridge is a controller chip that can transfer image data from CameraChips to external device. The OV528 takes 8-bit YCbCr 4:2:2 progressive video data from an OV7640 CMOS-Chip.

The camera interface synchronizes with input video data and performs down-sampling, clamping and windowing functions with desired resolution, as well as color conversion that is requested by the user through serial bus host commands.

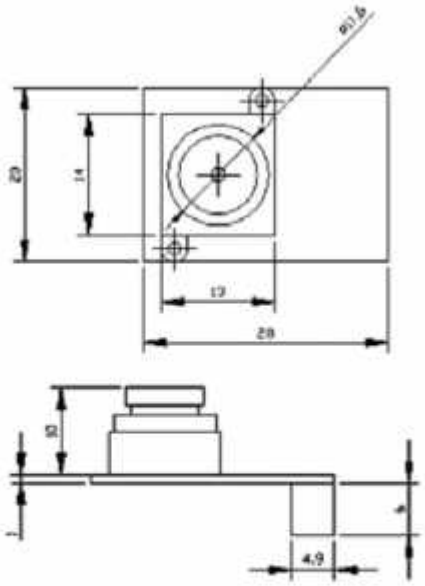
The JPEG CODEC with variable quality settings can achieve higher compression ratio & better image quality for various image resolutions.

Pin Discription:

VCC:..... Power 3,3VDC
TxD:..... Data Transmit (3,3V)
RxD:..... Data Receive (3,3V)
GND:..... Power Ground



Board Measurement



Electrical Specification

V_{DD} = 3.3V+10%, TA = 0 to 25°C

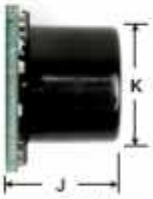
Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{DD}	DC supply voltage		3.0	3.3	3.6	V
Io	Normal Operation Current	Operating		60		mA
Is	Suspend Current	Suspend		100		uA
V_{IH}	High level input voltage	TTL	2.0			V
V_{IL}	Low level input voltage	TTL			0.8	V

Lens Specification

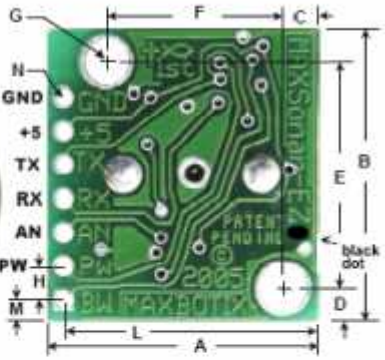
Description	Parameter
Imager Format	1/4"
F/#	2.8
Focal length (mm)	4.63
Field of View Diagonal (deg)	57
Horizontal (deg)	42
Vertical (deg)	16.5
Distortion	-3.3%
Relative Illumination	67%
Filter Option IR-cut filter	included

LV-MaxSonar®-EZ0™ High Performance Sonar Range Finder

With 2.5V - 5.5V power the LV-MaxSonar®-EZ0™ provides very short to long-range detection and ranging, in an incredibly small package. The LV-MaxSonar®-EZ0™ detects objects from 0-inches to 254-inches (6.45-meters) and provides sonar range information from 6-inches out to 254-inches with 1-inch resolution. Objects from 0-inches to 6-inches range as 6-inches. The interface output formats included are pulse width output, analog voltage output, and serial digital output.



approximately actual size



A	0.785"	19.9 mm	H	0.100"	2.54 mm
B	0.870"	22.1 mm	J	0.645"	16.4 mm
C	0.100"	2.54 mm	K	0.610"	15.5 mm
D	0.100"	2.54 mm	L	0.735"	18.7 mm
E	0.670"	17.0 mm	M	0.065"	1.7 mm
F	0.510"	12.6 mm	N	0.038" dia.	1.0 mm dia.
G	0.124" dia.	3.1 mm dia.	weight, 4.3 grams		

values are nominal

Features

- Continuously variable gain for beam control and side lobe suppression
- Object detection includes zero range objects
- 2.5V to 5.5V supply with 2mA typical current draw
- Readings can occur up to every 50mS, (20-Hz rate)
- Free run operation can continually measure and output range information
- Triggered operation provides the range reading as desired
- All interfaces are active simultaneously
  - Serial, 0 to Vcc
    - 9600Baud, 8N
  - Analog, (Vcc/512) / inch
  - Pulse width, (147uS/inch)
- Learns ringdown pattern when commanded to start ranging
- Designed for protected indoor environments
- Sensor operates at 42KHz
- High output square wave sensor drive (double Vcc)

Benefits

- Very low cost sonar ranger
- Reliable and stable range data
- Sensor dead zone virtually gone
- Lowest power ranger
- Quality beam characteristics
- Mounting holes provided on the circuit board
- Very low power ranger, excellent for multiple sensor or battery based systems
- Can be triggered externally or internally
- Sensor reports the range reading directly, frees up user processor
- Fast measurement cycle
- User can choose any of the three sensor outputs

Beam Characteristics

The LV-MaxSonar®-EZ0™ has the most sensitivity of the MaxSonar product line, yielding a controlled wide beam with high sensitivity. Sample results for measured beam patterns are shown below on a 12-inch grid. The detection pattern is shown for;

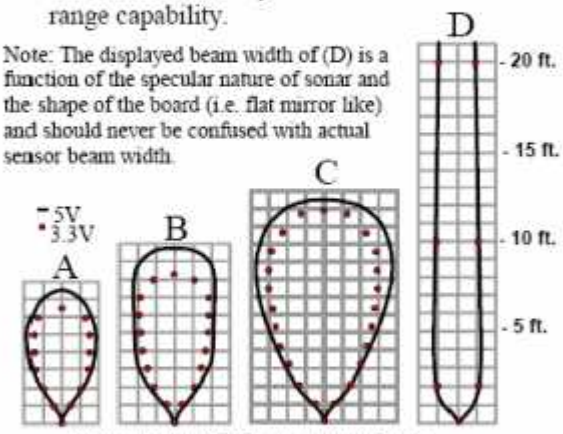
(A) 0.25-inch diameter dowel, note the narrow beam for close small objects,

(B) 1-inch diameter dowel, note the long narrow detection pattern,

(C) 3.25-inch diameter rod, note the long controlled detection pattern,

(D) 11-inch wide board moved left to right with the board parallel to the front sensor face and the sensor stationary. This shows the sensor's range capability.

Note: The displayed beam width of (D) is a function of the specular nature of sonar and the shape of the board (i.e. flat mirror like) and should never be confused with actual sensor beam width.



beam characteristics are approximate

MaxBotix® Inc.

MaxBotix, MaxSonar & EZ0 are trademarks of MaxBotix Inc.
LV-EZ0™ • v3.0c • 07/2007 • Copyright 2005 - 2007

Email: info@maxbotix.com Web: www.maxbotix.com

LV-MaxSonar®-EZ0™ Pin Out

GND – Return for the DC power supply. GND (& Vcc) must be ripple and noise free for best operation.

+5V –Vcc – Operates on 2.5V - 5.5V. Recommended current capability of 3mA for 5V, and 2mA for 3V.

TX – When the *BW is open or held low, the TX output delivers asynchronous serial with an RS232 format, except voltages are 0-Vcc. The output is an ASCII capital "R", followed by three ASCII character digits representing the range in inches up to a maximum of 255, followed by a carriage return (ASCII 13). The baud rate is 9600, 8 bits, no parity, with one stop bit. Although the voltage of 0-Vcc is outside the RS232 standard, most RS232 devices have sufficient margin to read 0-Vcc serial data. If standard voltage level RS232 is desired, invert, and connect an RS232 converter such as a MAX232. When BW pin is held high the TX output sends a single pulse, suitable for low noise chaining. (no serial data).

RX – This pin is internally pulled high. The EZ0™ will continually measure range and output if RX data is left unconnected or held high. If held low the EZ0™ will stop ranging. Bring high for 20uS or more to command a range reading.

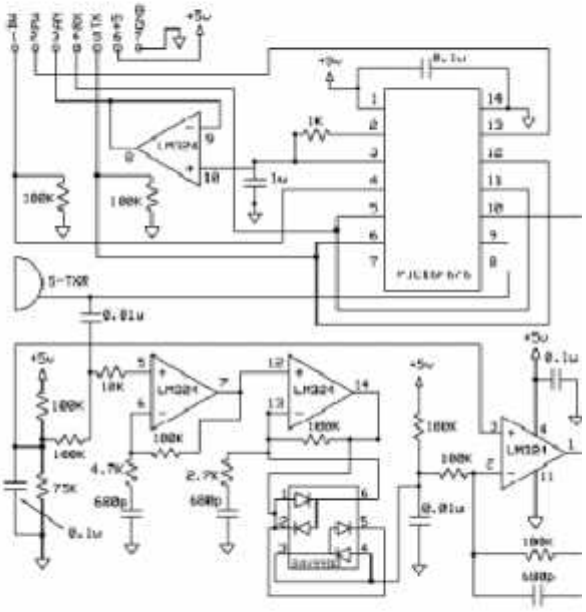
AN – Outputs analog voltage with a scaling factor of (Vcc/512) per inch. A supply of 5V yields ~9.8mV/in. and 3.3V yields ~6.4mV/in. The output is buffered and corresponds to the most recent range data.

PW – This pin outputs a pulse width representation of range. The distance can be calculated using the scale factor of 147uS per inch.

BW – *Leave open or hold low for serial output on the TX output. When BW pin is held high the TX output sends a pulse (instead of serial data), suitable for low noise chaining.

LV-MaxSonar®-EZ0™ Circuit

The LV-MaxSonar®-EZ0™ sensor functions using active components consisting of an LM324, a diode array, a PIC16F676, together with a variety of passive components.



LV-MaxSonar®-EZ0™ Timing Description

250mS after power-up, the LV-MaxSonar®-EZ0™ is ready to accept the RX command. If the RX pin is left open or held high, the sensor will first run a calibration cycle (49mS), and then it will take a range reading (49mS). Therefore, the first reading will take ~100mS. Subsequent readings will take 49mS. The LV-MaxSonar®-EZ0™ checks the RX pin at the end of every cycle. Range data can be acquired once every 49mS.

Each 49mS period starts by the RX being high or open, after which the LV-MaxSonar®-EZ0™ sends thirteen 42KHz waves, after which the pulse width pin (PW) is set high. When a target is detected the PW pin is pulled low. The PW pin is high for up to 37.5mS if no target is detected. The remainder of the 49mS time (less 4.7mS) is spent adjusting the analog voltage to the correct level. When a long distance is measured immediately after a short distance reading, the analog voltage may not reach the exact level within one read cycle. During the last 4.7mS, the serial data is sent. The LV-MaxSonar®-EZ0™ timing is factory calibrated to one percent at five volts, and in use is better than two percent. In addition, operation at 3.3V typically causes the objects range, to be reported, one to two percent further than actual.

LV-MaxSonar®-EZ0™ General Power-Up Instruction

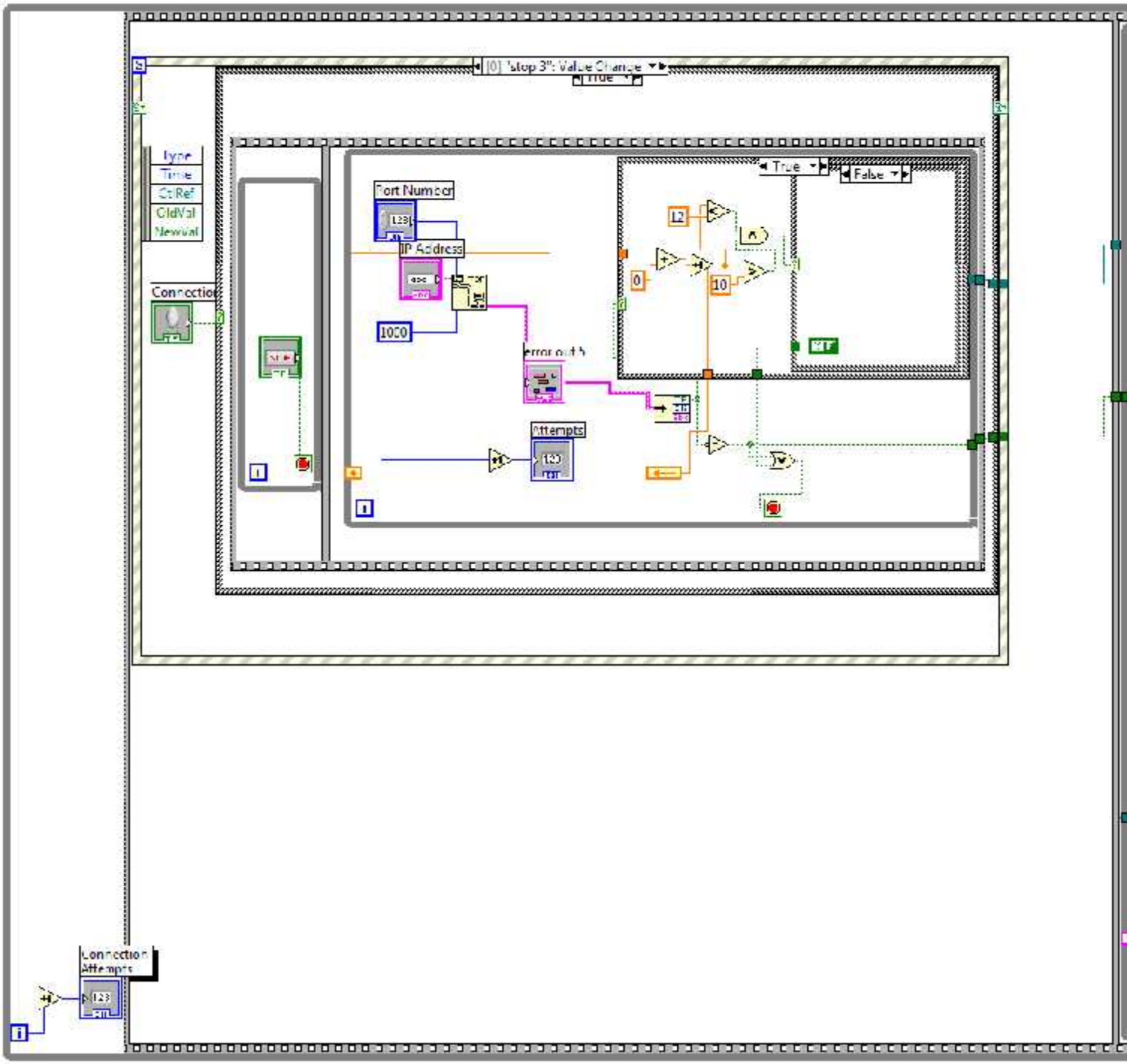
Each time after the LV-MaxSonar®-EZ0™ is powered up, it will calibrate during its first read cycle. The sensor uses this stored information to range a close object. It is important that objects not be close to the sensor during this calibration cycle. The best sensitivity is obtained when it is clear for fourteen inches, but good results are common when clear for at least seven inches. If an object is too close during the calibration cycle, the sensor may then ignore objects at that distance.

The LV-MaxSonar®-EZ0™ does not use the calibration data to temperature compensate for range, but instead to compensate for the sensor ringdown pattern. If the temperature, humidity, or applied voltage changes during operation, the sensor may require recalibration to reacquire the ringdown pattern. Unless recalibrated, if the temperature increases, the sensor is more likely to have false close readings. If the temperature decreases, the sensor is more likely to have reduced up close sensitivity. To recalibrate the LV-MaxSonar®-EZ0™, cycle power, then command a read cycle.

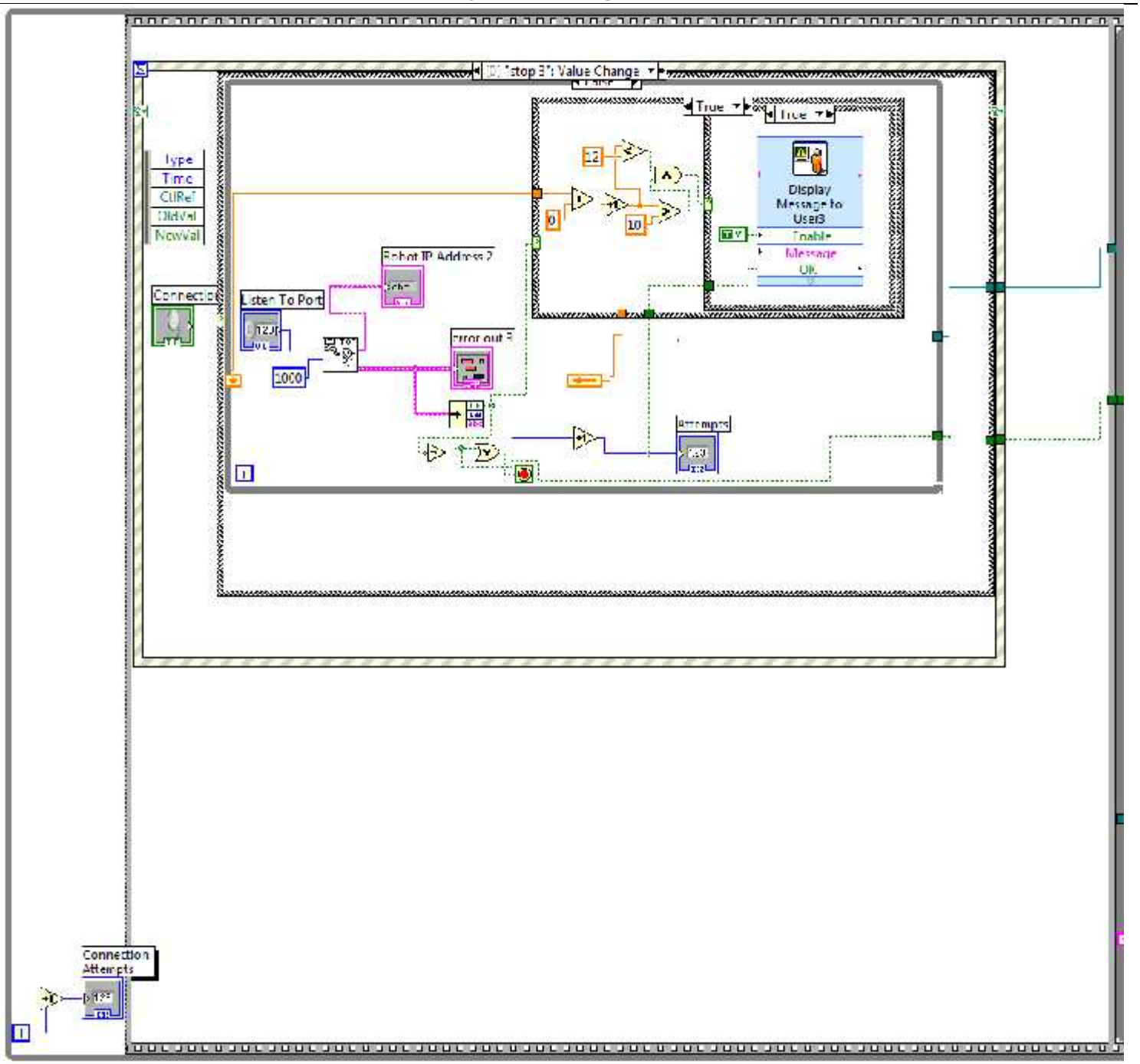
MaxBotix® Inc.
MaxBotix, MaxSonar & EZ0 are trademarks of MaxBotix Inc.

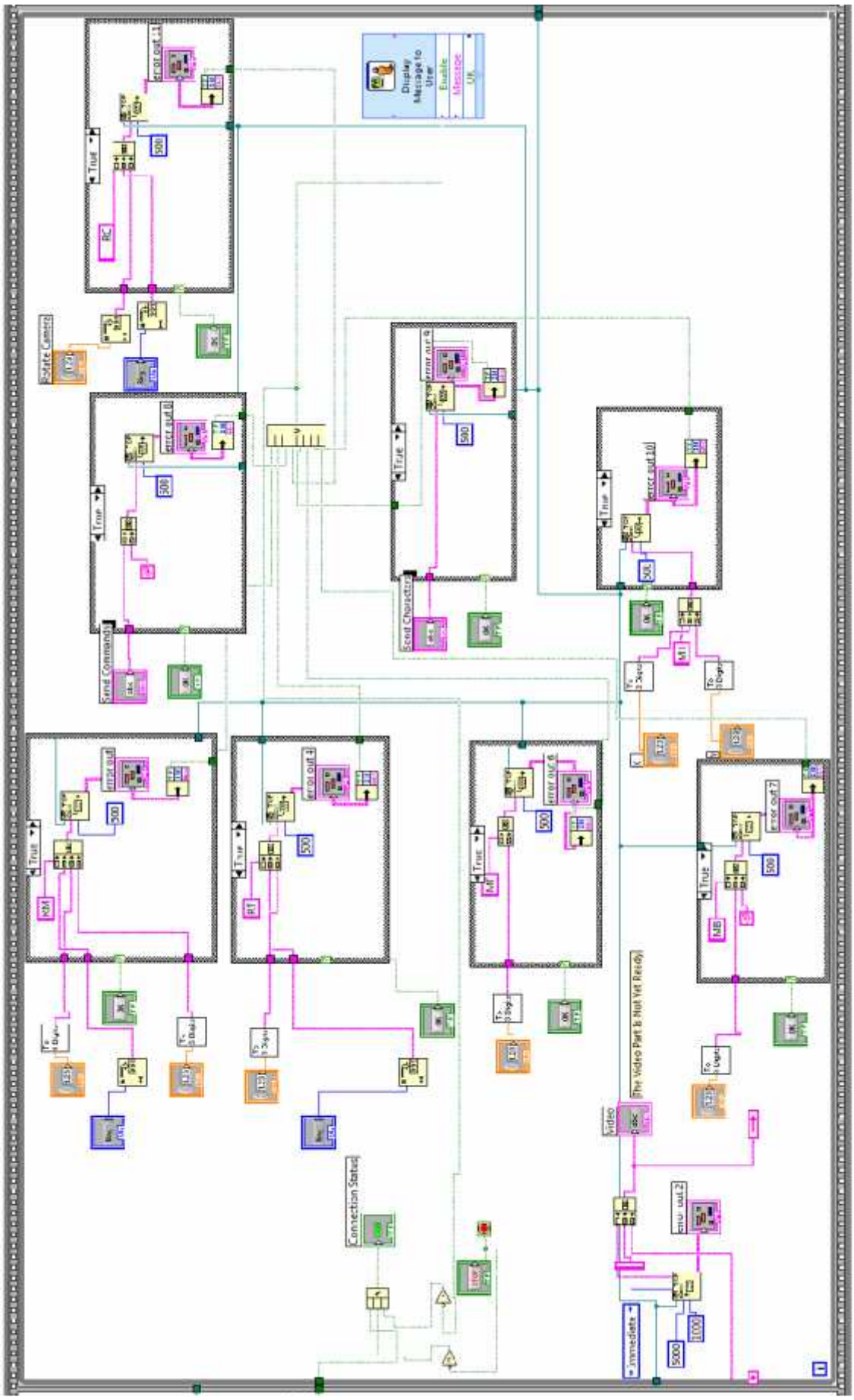
Appendix H

- **Robot Control Software Block Diagram (Code) LabVIEW 7.1 Software**


Type
Time
CtrRef
OldVal
NewVal
Connection
Port Number
IP Address
1000
Attempts
True
False
12
0
10
Connection Attempts

Automatic Connection by Listening Some Port Number (3000)




Display Message to User
Enable
Message
OK
True
RC
State Camera
Send Commands
Send Characters
Connection Status
The Video Part Is Not Yet Ready
Video
Immediate
5000
1000

Closing the Connection