IMPLEMENTATION OF AUTONOMOUS BALL FEEDER MOBILE ROBOT

By

AHMAD NIZAR B MUHAIDIN @ MAHYUDDIN

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

IMPLEMENTATION OF AUTONOMOUS
BALL FEEDER MOBILE ROBOT

by

Ahmad Nizar B Muhaidin @ Mahyuddin

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

Mr Mohd Haris Md Khir
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2005

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

Ahmad Nizar B Muhaidin @ Mahyuddin

ABSTRACT

The objective of the project is to design and implement autonomous ball feeder mobile robot. The robot will be able to feed the ball into the outer torch automatically. The purpose of designing the robot is to enter the ROBOCON competition organized by SIRIM. This is the first participation of University Technology PETRONAS in ROBOCON competition since it is an annually competition from 2002.

Without any past experience on building a robot, the Electrical & Electronic department has given the author challenges to build an autonomous robot base on certain constrains restricted by the rules stated by the organizer. The robot will be bigger in size and capable to carry heavier loads.

The scope of the study will be mainly on the design and implementation of the robot from scratch or little knowledge. The study will be handled part by part from researching on the whole part of the robot until the implementation of the workable robot. The robot implementation can be divided into two main sections which is hardware and controller part of the robot.

In the discussion part, all the robot implementation will be discussed in more detail as to make sure the objective of the project can be achieved successfully. The problem and the solution for the problem will also be discussed base on the student point of view.

Before ending the chapter, some recommendation has been suggested for further improvement for the next ROBOCON team members. The suggestions made are base on the current available technology and also the experience gain by the author through out this design project. To conclude the thesis paper, the conclusion will wrap up the whole findings in a general view.

ACKNOWLEDGEMENTS

First, the author would like to take this opportunity to express their appreciation to thank all parties involved in making the Electrical and Electronic Engineering Final Year Design Project success. This project would not have been possible without the assistance and guidance of certain individuals and organization whose contributions have helped in its completion.

The author's deepest appreciation goes to project supervisor, Mr Mohd Haris Md Khir for his support, guidance and patient throughout this design project. Greatest gratitude goes to all ROBOCON 2005 team members of University Technology PETRONAS (UTP) for their contribution in finishing the project within the period.

The author also would like to record the special thank to Ms Nasreen, the coodirnator of Final Year Project. Without their management and coordination, the flow of this project might not be as expected.

Special thanks also to Ms Siti Hawa who has given the author the numerous technical supports as a lab technician. She has been helpful to prepare the electronics components and parts for the project. Very special thanks to Azizan B Hashim the ROBOCON 2005 team leader, who has taught and give advice to the author in many areas which was not known to the author before. His help in giving ideas and programming skills is also one of the key factors of the success of this project.

Last but not least, the author would like to thanks the author's family and friends which have been supportive during the period of this project were held. Not to forget to all others who have help in the project directly and indirectly and might not be mentioned here.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Project Background

The 4th Asia-Pacific Robot Contest, or ABU ROBOCON, will be held in Beijing in the year 2005. It is an international competition organized by the Asia-Pacific Broadcasting Union. ABU ROBOCON is an annual robot contest starting from 2002, just for university, college and polytechnic students in the Asia-Pacific region. Under a common set of rules, participants will compete with their peers in other countries to create a robot using their creative and technological abilities in an open competition. The theme for the ABU ROBOCON 2005 will be ***"Climb on the Great Wall, Light the Holy Fire"***.

University Technology of PETRONAS become one of the participant for the ROBOCON 2005 selection and as a first participation, one team has represented the University Technology of PETRONAS for the national selection. The UTP ROBOCON team consists of Electrical and Mechanical students. Therefore, for the purpose, this project requires designing battle robots that capable to feed balls into five Torches and four Bonfires by collaboration between Manual and Automatic robot. The number of robots used are depend on the team strategies itself but restricted by a few rules like weight and height of the robots. All the robots should be made by hand from design to construction stages. Since the team consists of Electrical and Mechanical students, the scope of works will be divided into two. The scope of works for electrical students is more on controlling part like motors, sensors, programming and others while the mechanical students is more on the structure part of the robot.

1.2 Problem Statement

The robot must be an autonomous robot where it capable to feed the ball into the outer torch of the game field. All the structure part of the robot must be made by hand from designing to construction stages but must be restricted by the rules stated by the organizer. The rules stated that:

1. The total weight of one automatic robot must at least not exceed 10kg.
2. The maximum height of robot must not exceed 2m high.
3. The voltage used to power up the robot must less than 24V
4. The size of all the automatic robots must be fixed in 1m x 1m area of automatic starting zone.

Since the maximum weigh of the robot is 10kg, a high torque drive train motor needs to be used to drive the heavy structure. However the speed of the motor also need to be consider since the robot needs to move fast enough to reach the outer torch. For the controlling part, the robots will be controlled by using PIC microcontroller chip as a main controller for the robot. The robots will use pre-programmable method or line follower sensor as a guider for the robot to the outer torch.

1.3 Objectives

There will be a few objectives need to be achieved by the end of the project completed. The objective will be stated clearly as to make sure the successful of the project implemented. The objectives of this project are:

1.3.1 To design automatic robots that capable to feed balls into the Outer Torches

The robot must be able to move automatically to the Outer Torch without any human intervention. All the robot mechanism to feed the ball into the outer torch also will be controlled automatically base on the programming written.

1.3.2 To design a simple structure and mechanism of ball feeding robot

The design of the structure and mechanism part for the robot must be simple enough for easy troubleshooting purpose. A simple design ball feeding mechanism also can reduce the complexity in the programming part of the robot.

1.3.2 To equip the robot with the line follower sensor or pre-programmable method as a guider for the robot to the outer torch.

As to direct the robot to the outer torch, two methods can be used either line follower sensor or pre-programmable method. Line follower sensor will be equipped with the error correction algorithm as to make sure the straight movement of the robot while the pre-programmable will only use the pre determine path set in the program.

1.4 Scope of Works

Since the ROBOCON's team consists of Mechanical and Electrical students, the scope of works was divided into two. However, the scopes of works for Electrical part can be divided into three:

1.4.1 Controller part

PIC microcontroller will be used as a main controller for the robot and the C source code will be used to program the controller.

1.4.2 Sensor part

The suitable sensor is needed to be selected as a guider for the robot to the outer torch like line follower sensor.

1.4.3 Motor part

The author needs to select the best motor to be utilized as a drive train for the robot. Besides, the motor will also be used for the other purpose like lifting and feeding parts of the robot.

1.5 Gantt Chart

Please refer to **APPENDIX A** for the Gantt chart of the project

CHAPTER 2

LITERATURE REVIEW

Literature review is the research that has been done by collecting information from various sources such as from the internet and books. The research for the robot has been done part by part from the information on the robot competition until the components required to design the robot like the information on the previous ROBOCON, PIC microcontroller, PWM control, H-Bridge, servo motor, rotary encoder and finally on the line follower sensor. All the finding will guide the author to get the basic idea on designing the robot.

2.1 ROBOCON 2002

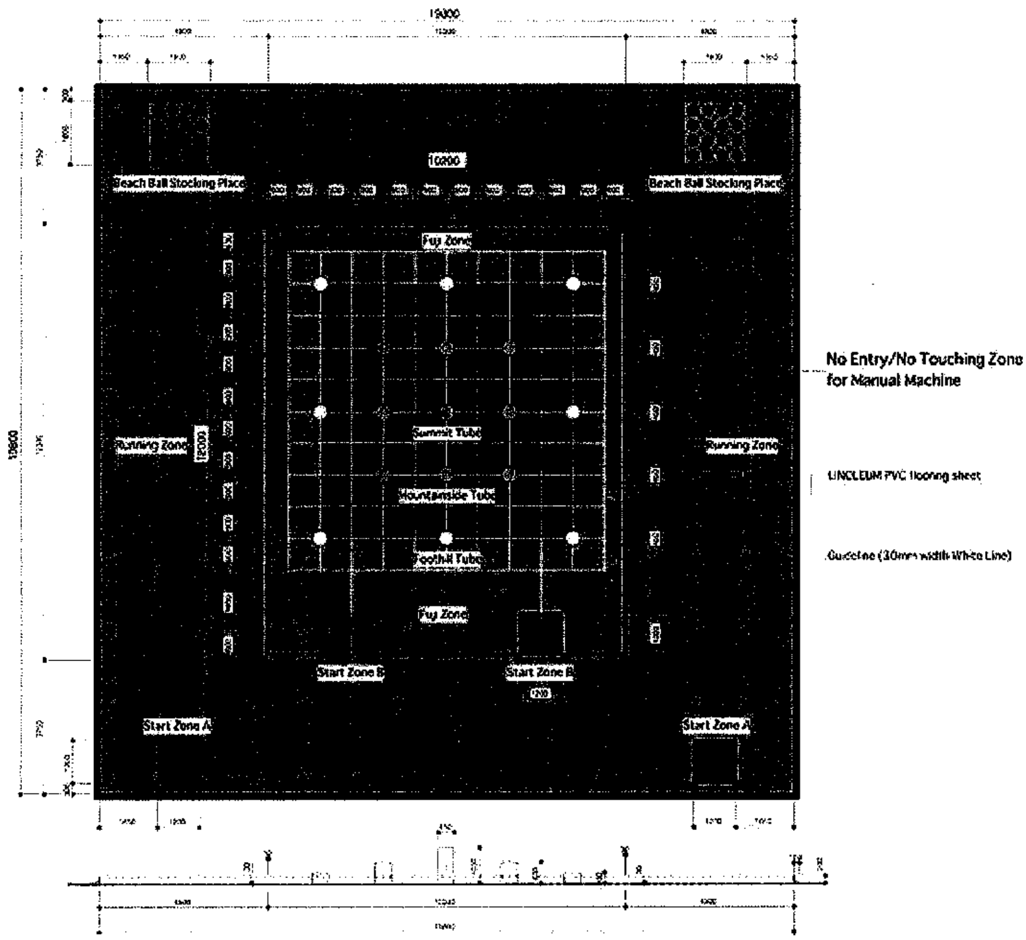


Figure 1 The game field layout for ROBOCON 2002

ROBOCON 2002 was held at Japan and the Theme for ROBOCON 2002 contest was **"Reach for the Top of Mt Fují",** was scheduled for August 31, 2002. It involves placing beach balls into cylinders of varying heights in a race to complete a diagonal line or, failing that, score points. To compete, at least one robot must play the game, navigate and manipulate the balls without intervention from human operators. The aim of this contest is to compete for points by placing beach balls into seventeen "Tubes" which they have "reach the summit", that is, when five consecutive "Tubes" are occupied in a diagonal line, which must also include the highest center "Summit" Tube(which represents the top of Mt. Fuji). The duration of each match is three minutes. The overall tournament involved preliminary and final rounds.

2.2 ROBOCON 2003

The aim of this Robot Contest is to handmade a machine from design to construction which will be most suitable to compete in the below contest theme. The aim of this contest is to shoot Takraw Balls into 9 baskets comprised of 3 nets in a triangular shape to compete for points. A team is considered the winner when the balls are shot into all baskets including 3 nets of centered basket or when one team scores more point than the opponent. The duration of each game is 3 minutes.

2.3 ROBOCON 2004

The theme of this contest is based on a love story in Asian legend. Couples called 'Gyeonwoo & Jiknyeo' are forced to be apart from each other with the Milky Way between them due to their laziness. Magpies and crows which feel sorry for the couple fly up to the sky and build a bridge with their bodies to get the couple together. It is called 'Ojak Bridge' (Bridge of Crow and Magpie). The couple gets together by crossing 'Ojak Bridge' once a year, on July 7th by lunar calendar. It always rains on this day and that is the tears of joy from Gyeonwoo and Jiknyeo for their reunion. The aim of this contest is to compete for accomplishing "Reunion" by completing the unfinished bridge and carrying Golden Gift by Automatic Machine from "Gyeonwoo Zone (Zone A)" to "Jiknyeo Zone (Zone B)". The duration of each match is three minutes.

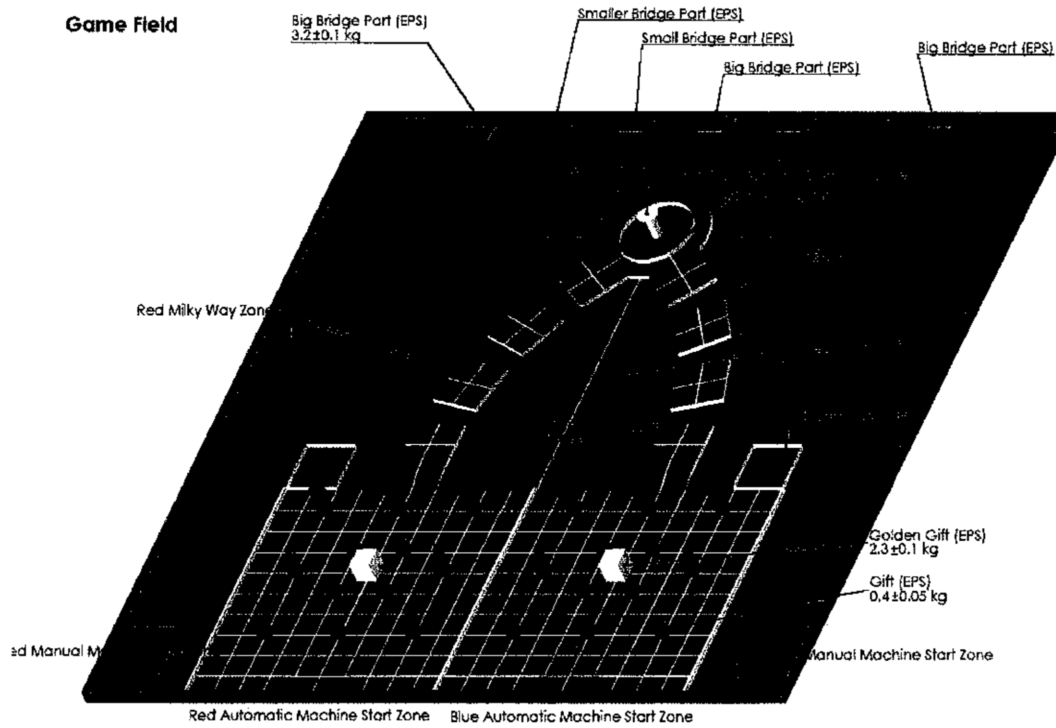


Figure 2 Game field layout for ROBOCON 2004

2.4 ROBOCON 2005

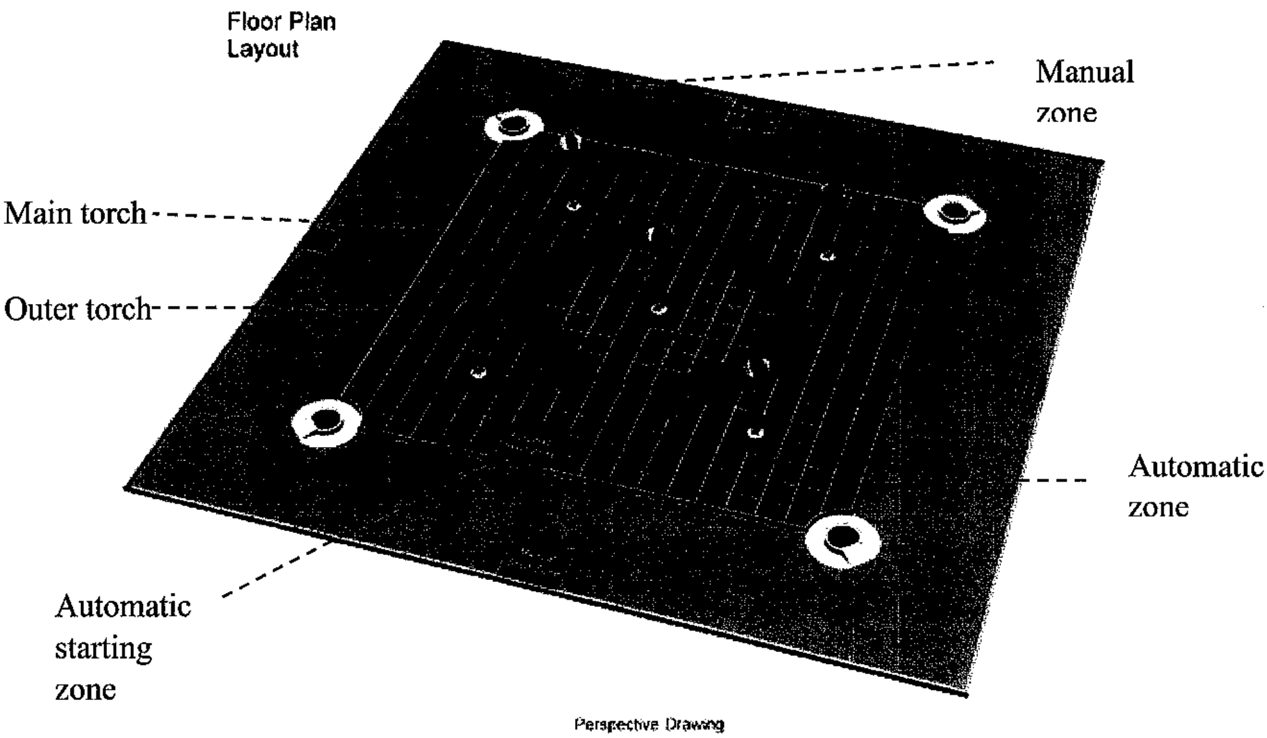


Figure 3 The outline of game field from the side view

The game field for ROBOCON 2005 is shown in the **Figure 3** above. Please refer to **APPENDIX B** for the other view of game field. The area of the game field is 14000mm x 14000mm in a square form and the floor is made of 2mm thick vinyl sheeting. The game field consists of Manual Zone, Bonfire Zone and Automatic Zone including the Beacon Tower Zone as shown above. The area of the automatic zone (Dark Green area) is 9000mm x 9000mm in a square form where the zone is surrounded by a wooden fence 100mm in high and 30mm in thickness. Automatic machine start zone for each team is 1000mm x 1000mm is located in the Automatic Zone and two Start Zones are opposite each other as shown in the layout. Only Automatic Machines may be operated in the Automatic Zone.

An octagon upland area, called the Beacon Tower Zone of 100mm in height, is located in the centre of the Automatic Zone. Five Torches are located in the Automatic Zone. The highest Main Torch of 1800mm in height is set in the centre of the beacon Tower Zone. The other four Outer Torches of 1500mm in height are distributed around it. The Main Torch is divided into red, blue and green portions called the red, blue and green Fuel Canister, equally by color clapboards. Each Outer Torches is divided into the same red and blue Fuel Canisters by colored clapboard but the Outer Torch can not be rotated. Please refer to **APPENDIX B** for the detail dimension of Outer Torch.

2.5 PIC MICROCONTROLLER

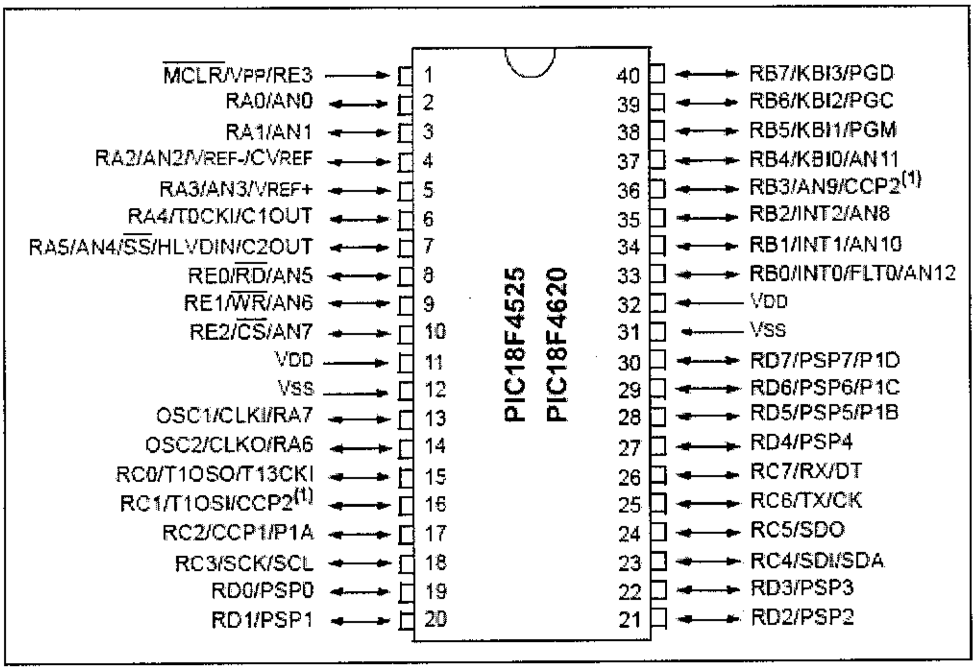


Figure 4 Pin layout of the PIC 18F4620

A PIC microcontroller is a single integrated circuit small enough to fit in the palm of a hand. 'Traditional' microprocessor circuits contain four or five separate integrated circuits - the microprocessor (CPU) itself, an EPROM program memory chip, some RAM memory and an input/output interface. With PIC microcontrollers all these functions are included within one single package, making them cost effective and easy to use. PIC microcontrollers can be used as the 'brain' to control a large variety of products. In order to control devices, it is necessary to interface (or 'connect') them to the PIC microcontroller.

Microchip PIC microcontroller model PIC18F4620 will be as a main controller part for the mobile robot. This PIC microcontroller is suitable to be used for this project since the size is small and low power consumption. For the feature of this microcontroller model, the microcontroller consists of:

Program memory (FLASH) - for storing a written program. Since memory made in FLASH technology can be programmed and cleared more than once, it makes this microcontroller suitable for device development.

EEPROM - data memory that needs to be saved when there is no supply. It is usually used for storing important data that must not be lost if power supply suddenly stops.

RAM-data memory used by a program during its execution. In RAM are stored all inter-results or temporary data during run-time.

PORT A, B, C and D are physical connections between the microcontroller and the outside world. All port consists of 8 I/O pins.

The other characteristic of the PIC18F4620 is described below:

Key features	PIC18F4620
Operating Frequency	DC-40 MHz
FLASH Program memory (14-bit words)	65536
Data memory (bytes)	3968
EEPROM Data Memory	1024
Interrupts	20
I/O ports	Ports A, B, C, and D
Timers	4
Capture/Compare/PWM module	1
Serial Communication	MSSP,USART
10-bit Analogue-to-digital Module	13 input channels

Table 1 PIC 18F4620 features table

PIC18F4620 is a flexible microcontroller since all the I/O ports can be used to any input or output to connect to the outside device. For the instruction part, C programming source code is used instead of using Assembly Language. This is because the C programming code is:

- Small and Powerful language
- Fewer keywords than Pascal
- Easy to learn & faster to code program

- Modular- via calling function by value

2.6 PWM Control

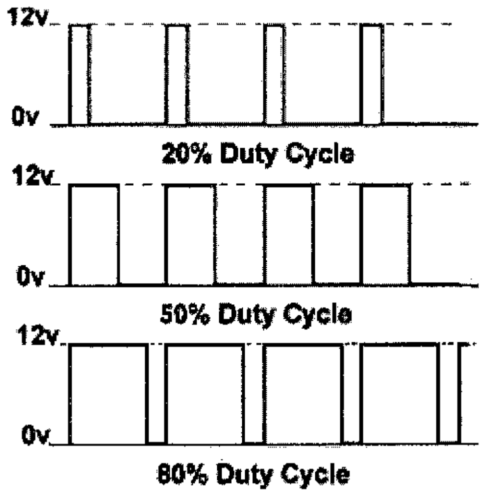


Figure 5 Duty cycle control by using PWM

Pulse-width modulation (PWM) control works by switching the power supplied to the motor on and off very rapidly. The DC voltage is converted to a square-wave signal, alternating between fully on (nearly 12v) and zero, giving the motor a series of power "kicks" as shown in **Figure 5** above. If the switching frequency is high enough, the motor runs at a steady speed. By adjusting the duty cycle of the signal or the width of the pulse of the time fraction it is "on", the average power can be varied, and hence the motor speed will also be varied. PWM is widely used to control the speed of a DC motor and the brightness of a bulb, in which case the PWM circuit is used to open/close a power line. If the line were opened for 1ms and closed for 1ms, and this were continuously repeated, the target would receive an average of 50% of the voltage and run at half speed or half brightness. If the line were opened for 1ms and closed for 3ms, the target would receive an average of 25%.

2.7 H Bridge

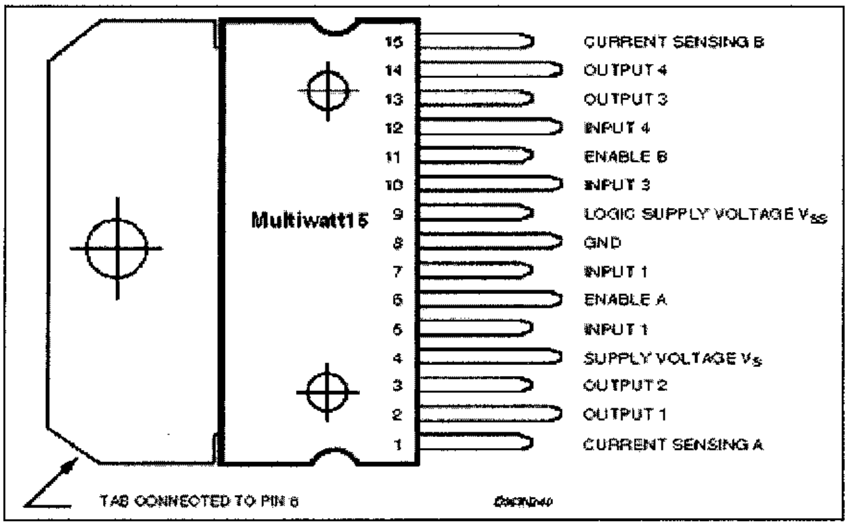


Figure 6 The L298N H-Bridge chip

The L298N consist of 8 motor control chip incorporates two H-bridge motor-driving circuits into a single 15-pin package. **Figure 6** shows a block diagram of this useful integrated circuit. Base on L298N datasheet, operating voltages is up to 46 V and the total DC current is up to 4 A only.

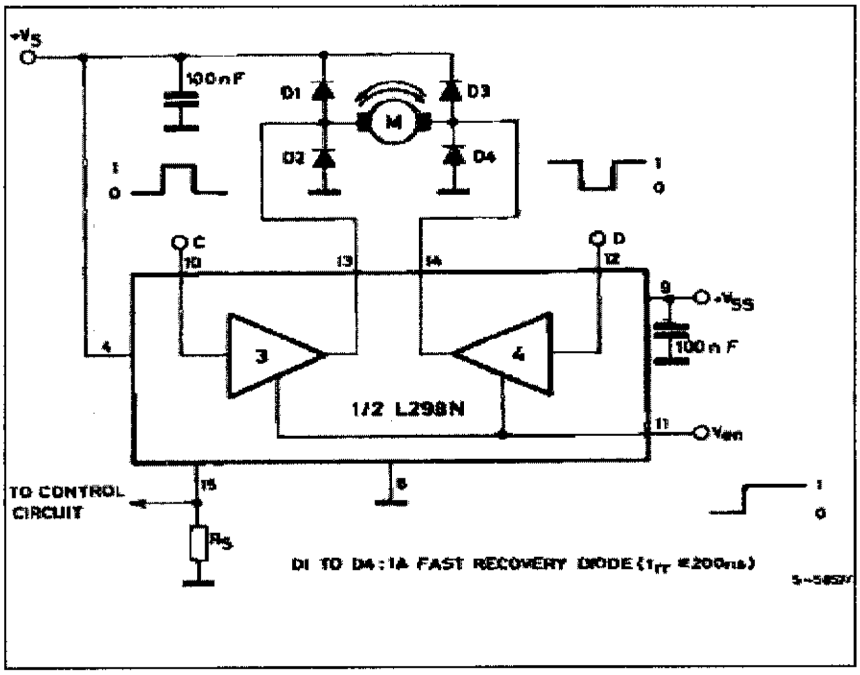


Figure 7 Schematic of half of L298N drive chip

Dual Full-Bridge driver L298N is used to control the direction of the drive train motor. The schematic of the motor circuit in **Figure 7** shows how the half of L298N controlling the movement of the wheel. Six bits are used to control two motors. Two of the bits (pin 10 and 12) determine the direction of the motors and one bit determine when the motors are on or off (pin 11). The direction of the motor will be determined base on the inputs apply in the **Table 2** below. The speed of a motor will be controlled by PWM the enable bit of its associated controller chip on and off.

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

Table 2 The inputs that determine the direction of DC motor

2.8 Servomotor

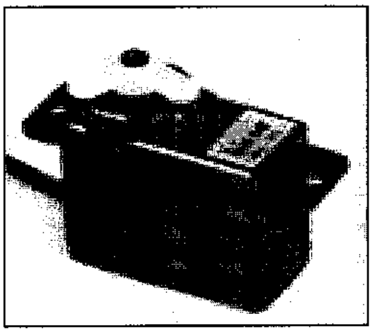


Figure 8 Futaba servomotor

Servo is a small device that has an output shaft as shown in **Figure 8** above. This shaft can be positioned to specific angular positions by sending the servo a PWM signal. As long as the PWM signal exists on the input line, the servo will maintain the angular position of the shaft. As the PWM signal changes, the angular position of the shaft changes. A typical servo has just three connection wires, normally red, black and white (or yellow). The red wire is the 6V supply, the black wire is the 0V supply, and the white (or yellow) wire is for the positioning signal.

In practice, servos are used in radio controlled cars, puppets, and robots. The control wire is used to communicate the angle. The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse Width Modulation. The servo expects to see a pulse every 20 milliseconds (.02 seconds). The length of the pulse will determine how far the motor turns. A 1.5 millisecond pulse, for example, will make the motor turn to the 90 degree position (often called the neutral position). If the pulse is shorter than 1.5 ms, then the motor will turn the shaft to closer to 0 degree. If the pulse is longer than 1.5ms, the shaft turns closer to 180 degree as illustrated in **Figure 9** below.

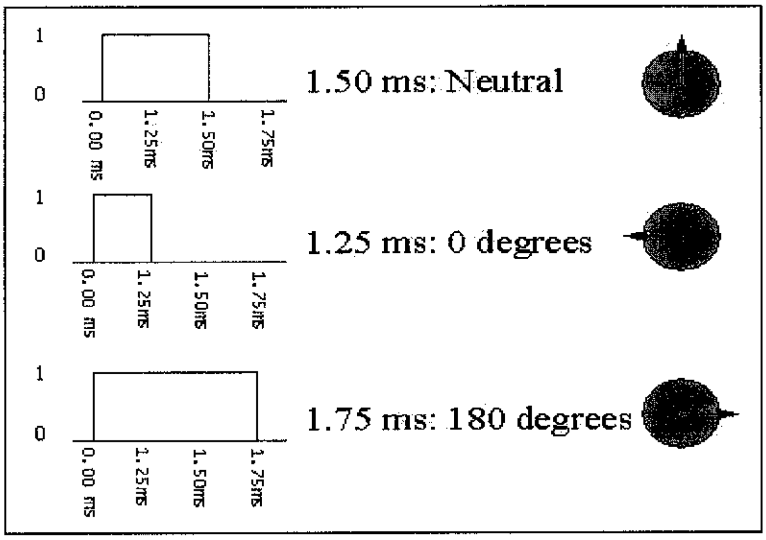


Figure 9 Example of duty cycle calculation

2.9 Photo reflector rotary encoder

The part chosen for encoder was the Hamatsu P5587 photo reflector as shown in **Figure 10** below. Encoder is a device that converts motion into a sequence of digital pulses and the pulse will use as an input to the main microcontroller to be processed. By counting a single bit, the pulses can be converted to relative position measurements to determine the distance travel. The photo reflector rotary encoder consists of IR transmitter and photo reflector pair. It is a 5 pin device with the following device.

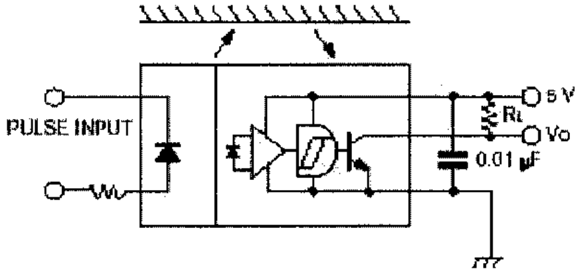


Figure 10 Photo reflector layout

Using a pull up resistor, the device will be at 5V if detecting black surface in front of it and 0V if detecting a white surface in front of it. The second part of the encoder is an encoder disk. There are a circle that are divided in equal slices of alternating black and white as shown in **Figure 11** below. The encoder disk will be mounted to the wheel with photo reflector fixed in place and facing a wheel creates encoder.

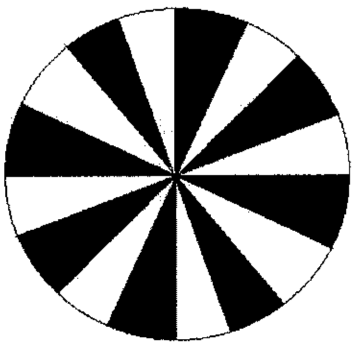


Figure 11 The encoder disk

2.10 Line follower sensor

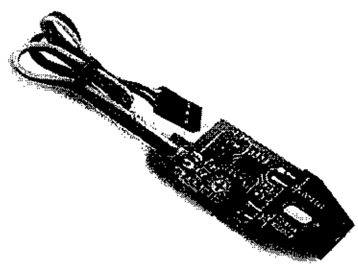


Figure 12 A single line follower sensor

The line follower sensor will be used to guide the robot to the target area by correcting the problem in path following so that the robot will not deviate from the path very far. More than a single line follower sensor will be mounted to the underside of a robot chassis toward the front of the vehicle. Position the sensor close to the floor with the red LEDs facing up. The sensor will operate in an extremely wide range from about 0.5" to 0.8" from the floor to almost touching the surface. But the most effective range is 0.8" base on the test conducted. The sensor appears to be immune to normal ambient lighting, although it may be necessary to shield the sensor from extremes.

A line sensor in its simplest form is a sensor capable of detecting a contrast between adjacent surfaces, such as difference in color. The simplest would be detecting a difference in color, for example black and white surfaces. Please refer **Figure 13** below for the surface detecting different between white and black surface. When the light shines on a white surface, most of the incoming light is reflected away from the surface. In contrast, most of the incoming light is absorbed if the surface is black. Therefore, by shining light on a surface and having a sensor to detect the amount of light that is reflected, a contrast between black and white surfaces can be detected.

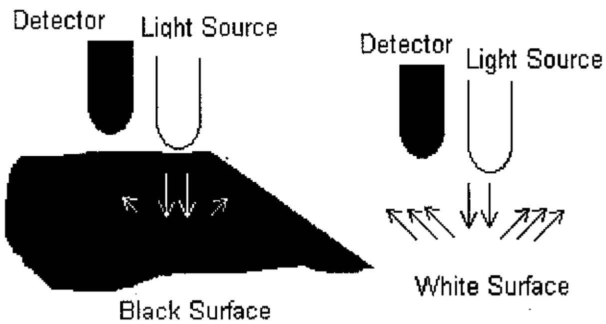


Figure 13 The different between the black and white surface

CHAPTER 3
METHODOLOGY

3.1 Procedure identification

Basically, there are 5 general steps of systematic approach for this project which is represented by general block diagram in **Figure 14** below. The general approaches are identifying the specification, doing a research, basic design, detail design and finally testing/troubleshooting. The detail of the block diagram will be explained below.

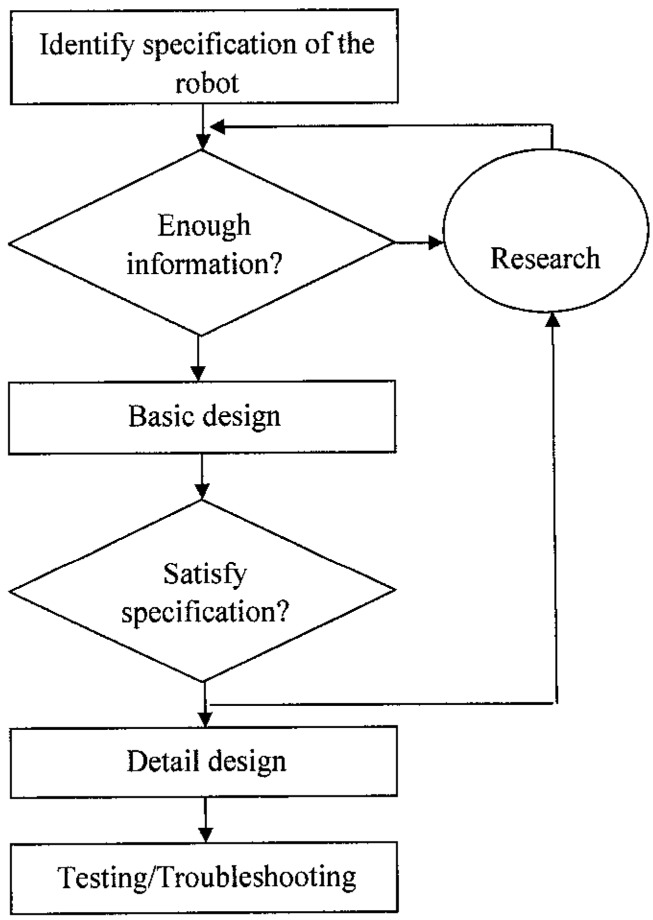


Figure 14 Methodology flow chart

3.1.1 Identify Specification

Since the ROBOCON is an annual international contest, the rules and regulation will be different for every year base on who is the organizer. Therefore, all the rules and regulations needs to be evaluated carefully as to make sure that the design of the robot can be acceptable during the competition. The rules stated will guide the author to get a basic idea on designing the robot.

3.1.2 Research

Research is very useful especially in assisting the successfully of the project. The researches have been done during early stage of the project. Literature from the internet is the main source in getting all the latest technology of robot especially on the previous robot contest. This is to get the idea and strategies on how the previous ROBOCON contester designed their robot with a different mechanism and technologies used. The author allocated about a month to find all the literature related to the battle robot. Basically the early stage of researches were more focus on the general components need to be implemented for the construction part of robot. This is for the budget preparation purpose for the whole ROBOCON project and also for basic design of the robot. However, research will continue until the author satisfies with the basic design of the robot.

3.1.3 Basic design

During the basic design, a few alternatives of the robot structure will be proposed. The structure of the robot is the most crucial part in designing the battle robot since all the designs must follow the rules stated by the organizer. Besides, the basic design also must be base on the strategies of the robots during the competition. The best alternative will be evaluated base on the lightest, robust and stable structure. The easies and fastest of purchasing the materials and equipments also will be consider as the best alternative because of the time constrain. Therefore, usually local material is more prefer than overseas material. During the basic design also, the arrived component will be assembled for conducting the test. The circuits and components will be assembled separately like drive train circuit, servo circuit and other before all the circuit will be integrated together. This is to make sure the workability of the separated circuit and once all the circuit have been tested successfully, the design will go to the detail design stage. However, if the test conducted does not meet the specification, the research need to be done again.

3.1.4 Detail design

The detail design is base on the best alternative selection during the basic design. The design will be more detail with the exact dimension of the robot structure. The equipments and tools need to be used for the robot will also be finalized during the detail design. All the successful tested circuit will be integrated together to get the complete set of the whole robot circuit.

3.1.5 Testing/Troubleshoot

The most crucial part in the methodology is testing and troubleshoots the robot. All the complete fabricated robot needs to be tested to identify any problem that might occur before the competition begin. All the problems need to be troubleshot during this period and the robot also need to be programmed for different mechanisms as to test the workability of the robot.

3.2 Tools Required

The tools can be divided into two parts; Equipments/Components and Software. The equipment/components will be used to construct the structure part of the robot while the software will be used to design the structure part of the robot and also for the programming purpose. Most of the materials to construct the robot are bought locally but there are also other materials that bought from the oversea. The list of equipments and software are shown below:

Equipments/Components:

1.	PIC 18F4620	1
2.	PIC 16F84A	6
3.	L298N H-Bridge	2
4.	7805 regulator	2
5.	7806 regulator	2
6.	20MHz Oscillator clock	1
7.	4MHz Oscillator clock	6
8.	Line follower sensor	5
9.	Wheel Encoder	2
10.	SPDT Relay	1
11.	Push button	1
12.	SPDT Switch	3
13.	DPDT Switch	1
14.	LED	10
15.	Diode 1N4001	20
16.	Heat Sink	6
17.	Electrical bicycle motors	2
18.	Electrical bicycle wheels	2
19.	Servomotors	3
20.	12V 7.5AH Yokohama Battery	1
21.	12V 1AH	1
22.	Aluminum L bar	
23.	Ball Custer	2

Software:

1. Multisim 6
2. Microsoft Office Visio 2003
3. Mach X Programmer
4. PCWH Compiler
5. Borland C++ 5.02

CHAPTER 4

RESULTS AND DISCUSSION

The results and discussions part will be divided into two sections which is hardware part and controlling/programming part. In the hardware part, the section will explain more on the structure part of the robot and the mechanism used to feed the ball into the outer torch. While for the controlling/programming part, the section will explain on the circuit construction and the algorithm used to control the robot. The flow of the movement of the robot will also be explained in this part base on the programming done by the author.

4.1 Hardware part

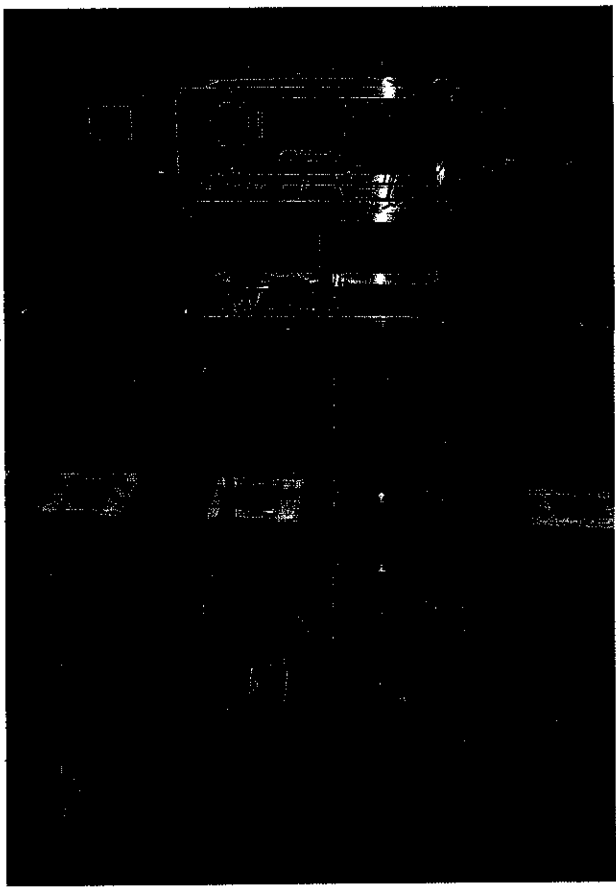


Figure 15 The structure part of the robot

The chassis of the robot was made from the L shape of Aluminum 6061 since it was the lightest and robust material that can be provided locally. The size of the robot is 55cm width x 35cm length x 150 cm high. Rivet was used as to mount the structure part rather than using the bolt and nut since it is more robust. Generally, the structure part of the robot can be divided into 3 parts:

1. Drive train
2. Lifter
3. Ball frame and railing

4.1.1 Drive train

Motor can be divided into two main parts; drive train and lifter part. Both of the mechanism will use different types of motor with a different characteristic. The drive train motor is desired to be greater torque with high rpm. High torque is required for the motor that capable to carry at least 10kg load since the minimum estimation weight for one robot is 10kg. The rpm of the motor is also important in the robot contest. This is because, during the contest the robot should be able to reach as fast as possible to the target area before the opponent robot can reach to make a score. Unfortunately, it is impossible to get both characteristics; greater torque and high speed in one motor. As to solve the problem, gear with a certain ratio will be used to increase a torque for high rpm motor. Therefore, the most suitable motor that can be used is electrical bicycle motor since it comes with a complete set of gear box. Thus, the test has been conducted on the complete set of electrical bicycle motor with a gear box. The purpose of the test is to get know the capability of the motor to drive the 10kg load and to identify the best duty ratio for the motor when carrying the 10kg load. **Figure 16** show the inner structure of the electrical bicycle gear box with the gear ratio for different steps of gears.

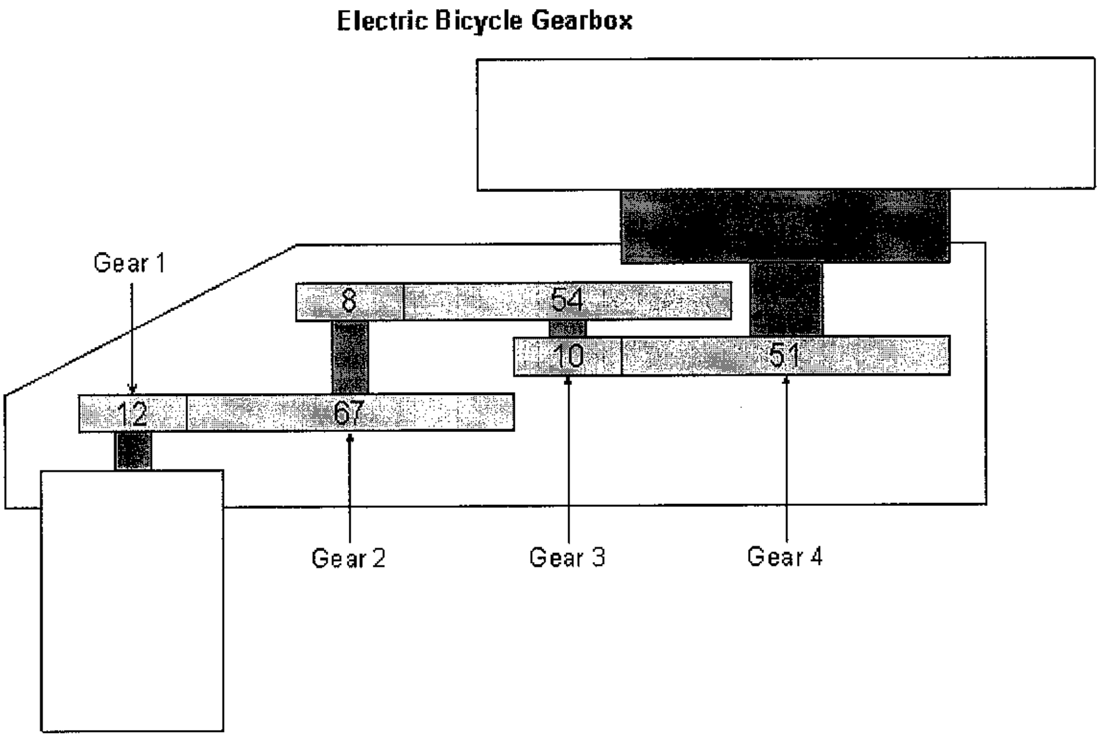


Figure 16 The bicycle motor gear box

4.1.2 Description of the experiment

The experiment has been conducted by placing 10kg load on the robot model with the electrical bicycle has been used as the drive train for the robot model.L298N was used as the H-bridge for load test. As to control the speed of the motor, PWM concept has been applied through the H-bridge circuit. The value for duty ratio of PWM was varied to find the different effect on the capability of the motor. For this experiment, two different duty ratios (0.75 and 0.90) were applied to get a different output results. Every experiment was conducted within 5 seconds by using 9cm radius of wheel.

Result of experiment:

Measured values				
Condition: Full gears (gear 1 to 4), 10kg load, battery 12V 7.5Ah, L298N h-bridge, tyre r=9 cm				
H-bridge duty ratio	**Test no.**	**Period**	**Distance travelled**	**Robot speed**
		(s)	**(m)**	**(m/s)**
0.75	1	5	3.30	0.66
0.90	1	5	4.76	0.95
	2	5	4.63	0.93
	3	5	4.59	0.92

Table 3 Results of the electrical bicycle performance

Base on the experiment result in **Table 3** above, the best performance of the motor with three steps of gears can be achieved by using 0.90 duty ratio. This is because the motor can achieve almost 1m/s speed with 10kg of load. Therefore, the motor is recommended to be used as a drive train for the robot.

Ball bearing

Coupler

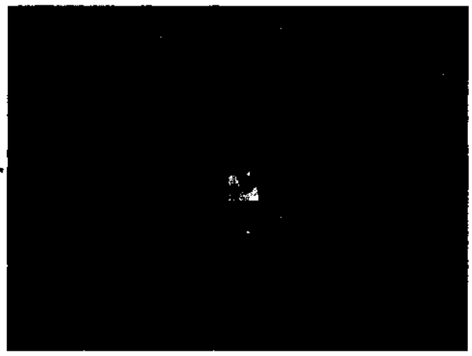


Figure 17 The coupler for drive train

Coupler was used to mount the wheels with the main frame and it was made from the soft solid aluminum. Ball bearing was mounted together with the coupler as to make sure the wheel will rotate smoothly.

4.1.3 Lifter motor

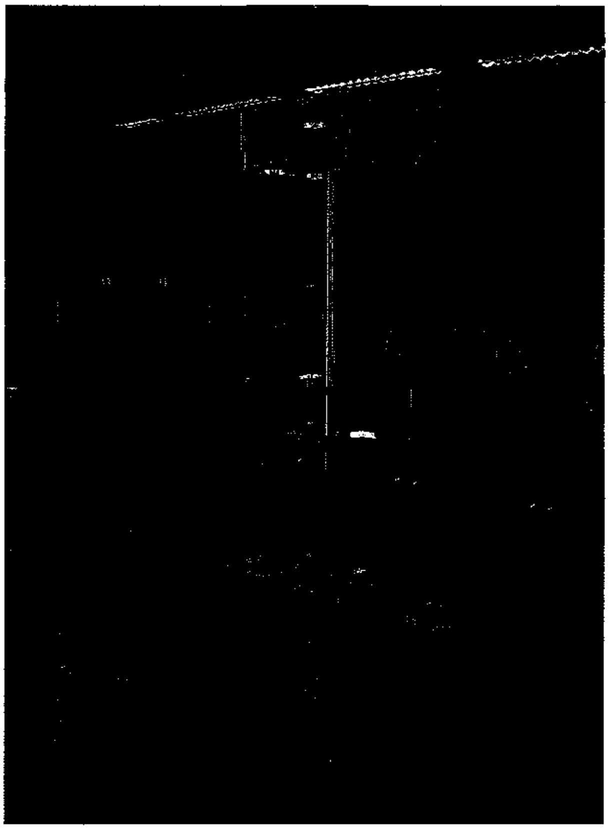


Figure 18 The ball frame elevate from 1.5m to 1.8m height

Lifter motor will be used to lift up a frame preloaded with balls from 1.5m to 1.8m as shown in **Figure 18** above. This is because the high of all automatic robots in the start zone must be less than 1.5m. Once the automatic robots leave the start zone, their form may change freely and the high must be limited to within 2m. This is because the maximum height of the outer torch is 1.8m and the ball frames need to lift up to make sure that the ball can be fed inside the outer torch. The lifting motor should be a high torque motor since it requires to lift up at least 2kg load. The rpm speed of the motor is not a critical issue since lifting part does not require a fast movement but the ability of motor to lift up the load.

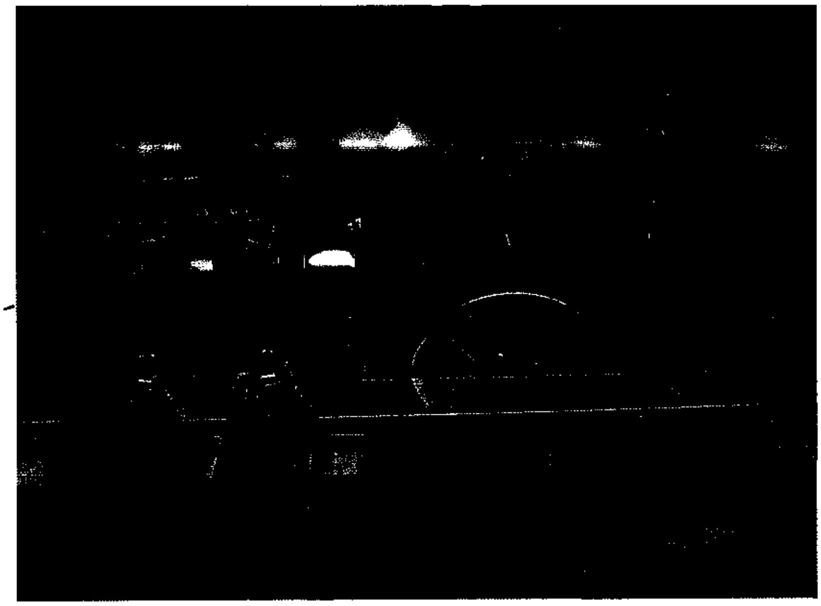


Figure 19 12V power window motor

12V power window motor has been chosen as the lifter motor since it has a higher torque to lift up load more than 2kg. The power window motor uses a concept of worm gear for the turning mechanism. For the lifting mechanism, bicycle sprocket and the chassis of power window have been welded together so that the sprocket can be used to turn together with the power window's gear. As shown in **Figure 19** above the sprocket was used to move the ball frame up and down guided by the holed aluminum pole. As to avoid slip, the contact of the sprocket must be close enough with the aluminum pole. Limit switch with the stopper as shown below were used to stop the lifter to the desire height which is from 1.5m to 1.8m.

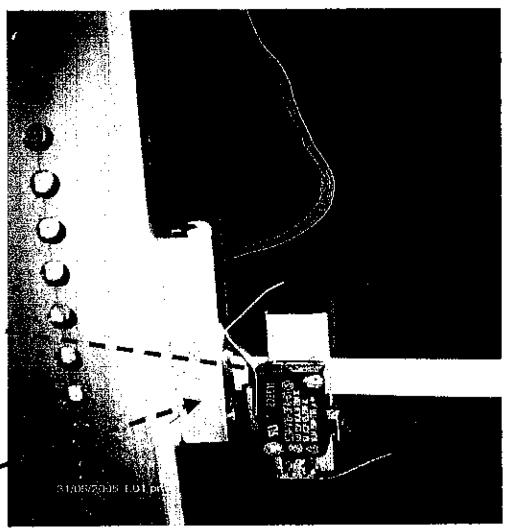
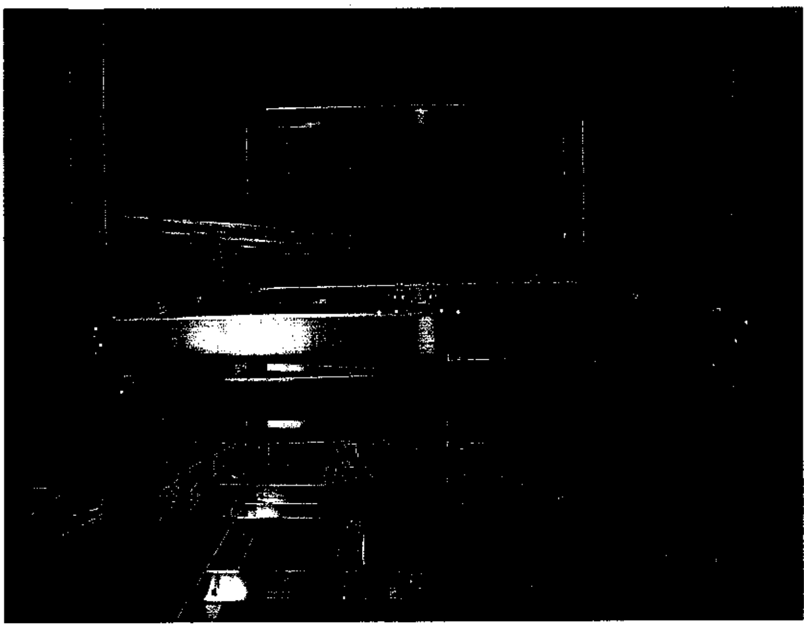


Figure 20 Limit switch

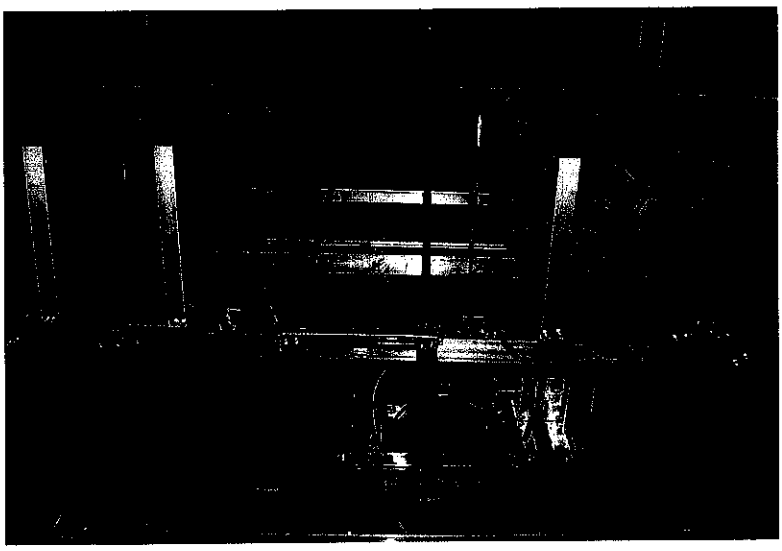
4.1.4 Ball frame and Railing



→ Incline rail

Figure 21 Railing on the ball frame

The ball frame as shown in **Figure 21** above is used to carry the balls into the target area which is the outer torch. The size of ball frame is 55cm width x 35cm length x 25 cm high and a total number of 6 balls can be loaded inside the frame with the size of the ball is 15cm. The base for the ball frame was designed with the concept of incline rail so that the ball will be in the falling position. The purpose of the incline rail is to simplify the feeding mechanism and does not need any other mechanism to push the ball out from the frame. Therefore, the gate has been designed to stop the ball from falling down before the robot reaches the outer torch.



→ Gate

Figure 22 The gate mechanism to stop the ball from falling down

The gate as shown in **Figure 22** above will move up and down by using the servo motor. Three servomotors were used to control the opening of the gate and every servomotor was mounted at the side of every gate. The function of the gate also can be used as a guider for the ball to fall into the torch. Once the gate was open, two ball will fall together to feed into the torch.

4.2 Controlling and Programming design part

All the circuits have been divided to section by section base on the different functionality of the circuit. The best way to construct the circuit is by using the Printed Circuit Board (PCB). However the chemical to construct the PCB is not available at the laboratory, thus the circuits have been constructed by using the veroboard only. The total numbers of 6 circuits have been constructed:

a. Main controller board
b. Power supply board
c. H-Bridge board
d. Photoreflector rotary encoder
e. DPST relay
f. Servomotor board
g. Line follower sensor

4.2.1 Main Controller Board

PIC18F4620 was used as a main controller for automatic ball feeder robot. The main controller will act as a brain for the robot where all the process and algorithm will be controlled by the controller. The microcontroller can be program either by using assembly language or a high level compiler in C language. Programming in assembly language will make the code more optimize in term of memory managements but as the codes gets into complex loop and subroutine, keeping track of the codes will be difficult. High level of programming skills and experience will be needed if the codes are in assembly language.

Since PIC 18F4620 consist of 40 pins with 4 sections of I/O ports; Port A, B, C and D, all the ports were dedicated with different types of I/O as stated in the **Table 4** below:

No.	Port	Input/Output
1	PIN A5= Buzzer	Output
2	PIN A4= Start Pushbutton	Input
3	PIN A3= Reserve	Input/Output
4	PIN A2= Reserve	Input/Output
5	PIN A1= Running Indicator	Output
6	PIN A0= Busy Indicator	Output
9	PIN B5= Line Sensor (Left)	Input
10	PIN B4= Line Sensor (Center)	Input
11	PIN B3= Line Sensor (Right)	Input
12	PIN B2= Door 3 Servo	Output
13	PIN B1= Door 2 Servo	Output
14	PIN B0= Door 1 Servo	Output
15	PIN C7= Serial Rx	Input
16	PIN C6= Serial Tx	Output
17	PIN C5= Lifter High Limit Switch	Input
20	PIN C2= Drive Train (Right) PWM	Output
21	PIN C1= Drive Train (Left) PWM	Output
22	PIN C0= Lifter Low Limit Switch	Input
23	PIN D7= Lifter Direction	Output
25	PIN D5= Encoder Pulse (Left)	Input
26	PIN D4= Encoder Pulse (Right)	Input
27	PIN D3= Drive Train (Left) Dir 1	Output
28	PIN D2= Drive Train (Left) Dir 0	Output
29	PIN D1= Drive Train (Right) Dir 1	Output
30	PIN D0= Drive Train (Right) Dir 0	Output

Table 4 The dedicated I/O ports for PIC18F4620

In hardware implementation as shown in **Figure 21** below, the microcontroller needs a regulated 5V voltage supply which is being regulated using LM7805 voltage regulator. The input voltage is from a separate 12V battery. The microcontroller is capable to run at 40 MHz clock but a 20MHz crystal clock is sufficient to the processing of the codes. Two normally open switches are used to control the operation of the microcontroller. The first one is used to give a low signal to microcontroller clear pin which will restart the whole process of the controller if pressed. The next button will be used to signal a start sequence which will start the operation of the robot.

The output pins from the microcontroller will be fed to the H Bridge circuit, lifter circuit, servo circuit and also a buzzer. The 6 pins that will be fed to the H bridge consist of 4 pins which will give the combination for the motor direction and 2 pins which will gives the PWM signal to the H bridge. The PWM signal is used to control the speed by varying the ON time and OFF time of the H bridge circuit. This will result in average power output to the motor depending on the duty ratio selected or generated by the PWM pins. The input pins to the microcontroller will be fed from encoder circuit, line follower circuit and switches like limit switch, reset and start button. The line follower circuit will give 3 inputs to the microcontroller which is input from left, center and right line. While the encoder circuit will give 2 inputs to the microcontroller which is coming from both of the wheels rotation reading. The limit switch is used to trigger the maximum high of the ball frame. Once the limit switch was triggered, the output from the limit switch will be fed into the main controller. The main controller will process the data and give the output to the relay to stop the movement of the lifter on the desire height.

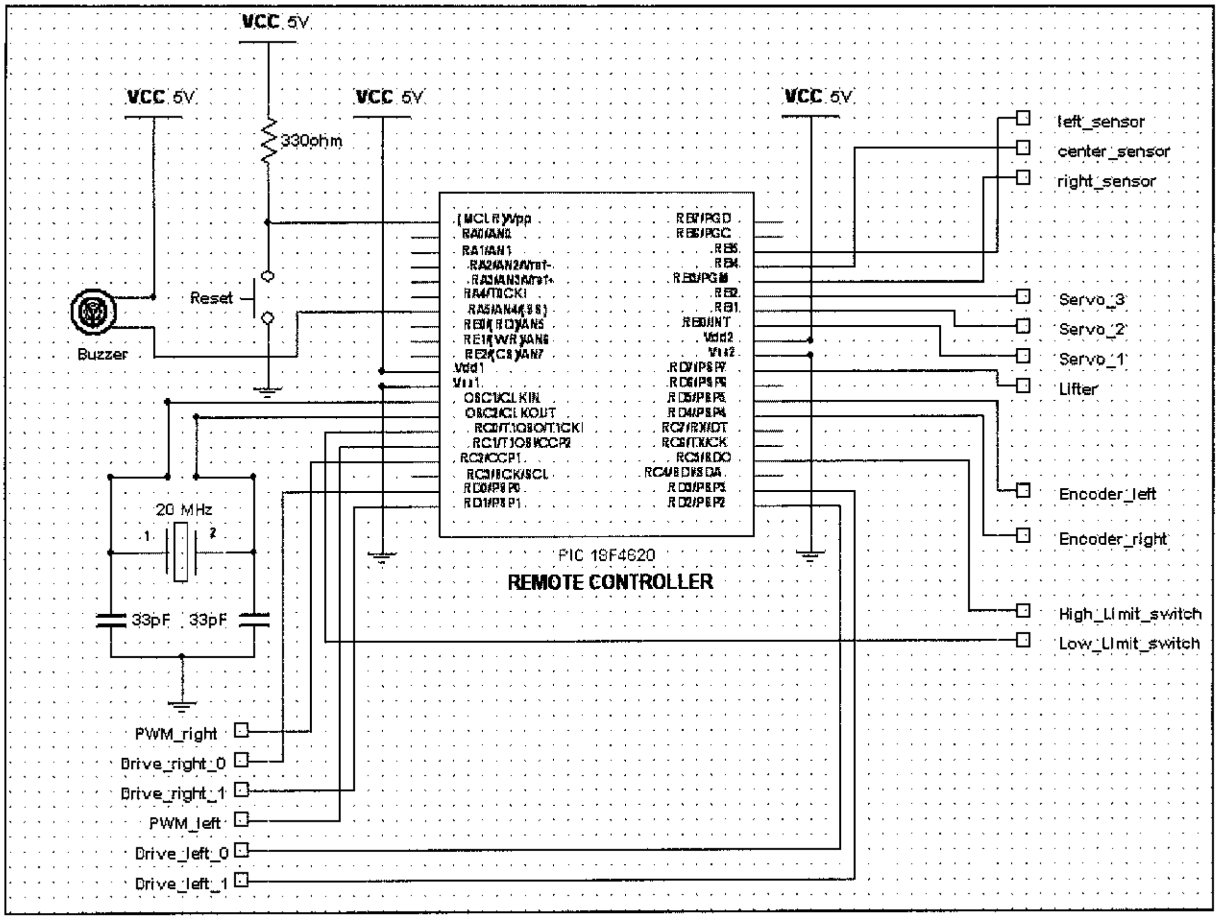


Figure 23 Layout of main controller circuit

Please refer to **Appendix C** for the Visio circuit of main controller board.

4.3 Power supply board

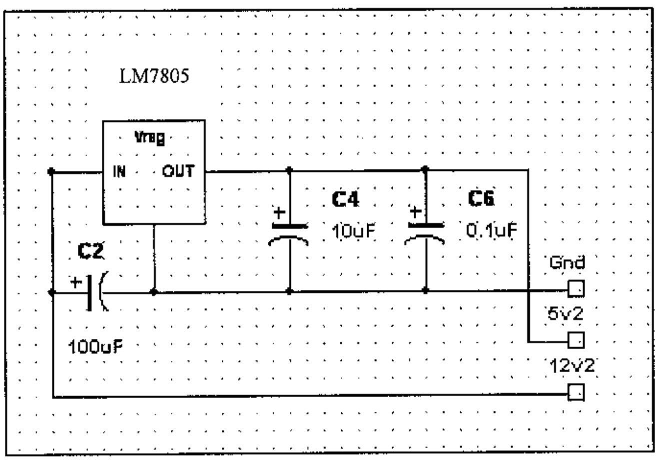


Figure 24 Layout of power supply circuit

12V 7.5AH seal lead acid and 12V 1AH batteries were used to power up the robot. Both of the batteries were used since it can be rechargeable when the power was degraded. 12V 7.5AH was used to power up all the drive train and lifter motor while 12V 1AH was used to power up the circuits part. Since most of the circuits require only 5V and 6V to be activated, so the regulator needs to be used to have a voltage drop from 12V to 5V or 6V. Therefore, the power supply board was designed so that the board can supply the circuit with the desire voltage. Regulators LM7805 and LM7806 have been used on the board to drop the voltage. Regulator LM7805 was used to the drop the 12V to 5V and regulator LM7806 was used to drop the 12V to 6V. As to stabilize the voltage output, capacitor will be placed in parallel with the output or input to the regulator with the ground as shown in **Figure 24** above. The purpose is to reduce the noise since one of the capacitor characteristic is to filter the noise. Usually 6V will only be used to power up the servomotor and 5V will be used to power up the IC like microcontroller and H-Bridge. Please refer to **Appendix C** for the Visio circuit of Power supply board.

4.4 H-Bridge board

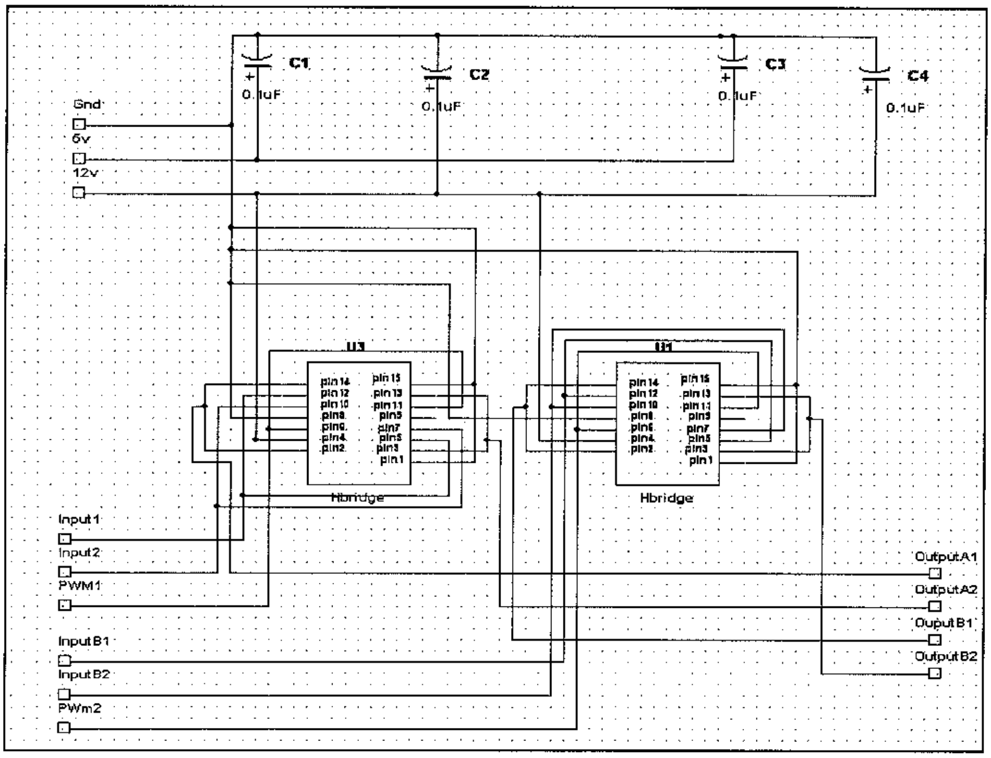


Figure 25 Layout of H-Bridge circuit

As described in the literature review above, the L298N H-Bridge is used to control the speed and direction of the motor. The speed will be controlled by varying the PWM signal and the direction will be controlled by varying the inputs base on the 2 inputs from the microcontroller as shown in **Figure 25** above. The output from the H-Bridge will connected to both of the left and right motors. The limitation of using the L298N H-Bridge compare to relay is the current limitation. Base on the datasheet, the maximum voltage is up to 46 V and the total DC current is up to 4 A only. Thus, the motor will not able to drive faster rather by using the relay. However, the advantage of using the L298N compare to relay is, the speed of the motor can be controlled by using the PWM compare to the relay which the speed is constant only. Since the L298N consist of 2 H-Bridge circuits, thus each H-Bridge circuit will be dedicated for each motor for left and right wheels. The steering of the robot will also be controlled by using the H-Bridge. For example, if the robot needs to turn to the right. Thus the PWM supply for the right motor will be disabled so that the right motor will stop. Meanwhile, the PWM supply for the left motor continues be enable so that the left motor will drive the robot to the right. The turning angle will be determined base on the try and error values of PWM duty ratio supplied on the left motor. Please refer to **Appendix C** for the Visio circuit of H-Bridge board.

4.5 Photo reflector rotary encoder

The module of photo reflector rotary encoder was used to determine the distance of the robot to move from one point to the other point. The encoder was mounted vertically on the drive train gear box and oppositely with the disk encoder. The disk encoder was mounted at the side of the wheel so that the encoder will read the wheel rotation base on the black and white of disk encoder color. The effective range to mount the encoder is 0.5cm from the disk encoder. A simple formula for encoder can be used to calculate the distance per pulse as shown below:

$$\text{Distance per pulse} = 2\pi r / 60$$

$$\text{Where } r = 9.75\text{cm}$$

$$\text{Distance per pulse} = 1.02\text{cm/pulse}$$

From the formula, the value of 60 comes from the number of black and white stripe at the disk encoder. Thus, from the calculated value, every time encoder read the slice of black or white, the distance will equal to 1.02cm. Since both of the wheels were mounted with the rotary encoder, the average of both distances per pulse for both rotary encoders will determine the distance of the robot moves.

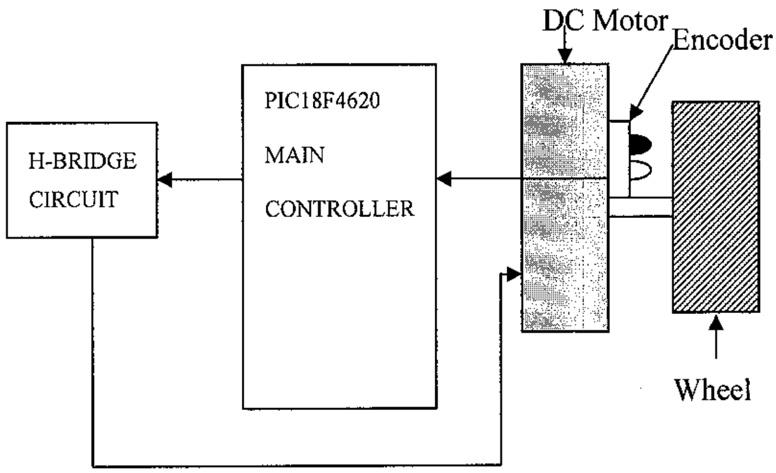


Figure 26 Block diagram for encoder loop

Base on **Figure 26** above, the encoder will read the number of black and white stripes on the encoder disk. By comparing the distance desired value programmed in the main controller, the main controller will send the signal to H-Bridge. The H-Bridge will control the rotation of motor base on the desire distance in the programming by sending the signal to the DC motor. For the clearance view, please refer **Figure 27** below for the mounting spot for the rotary encoder and disk encoder.

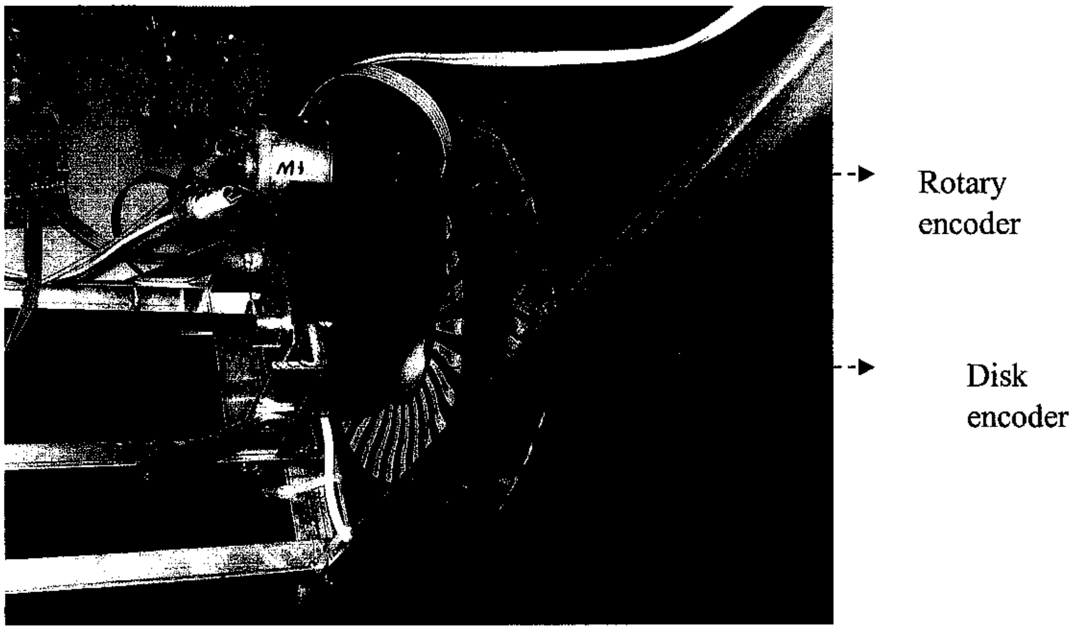


Figure 27 The mounting spot for the rotary encoder and disk encoder

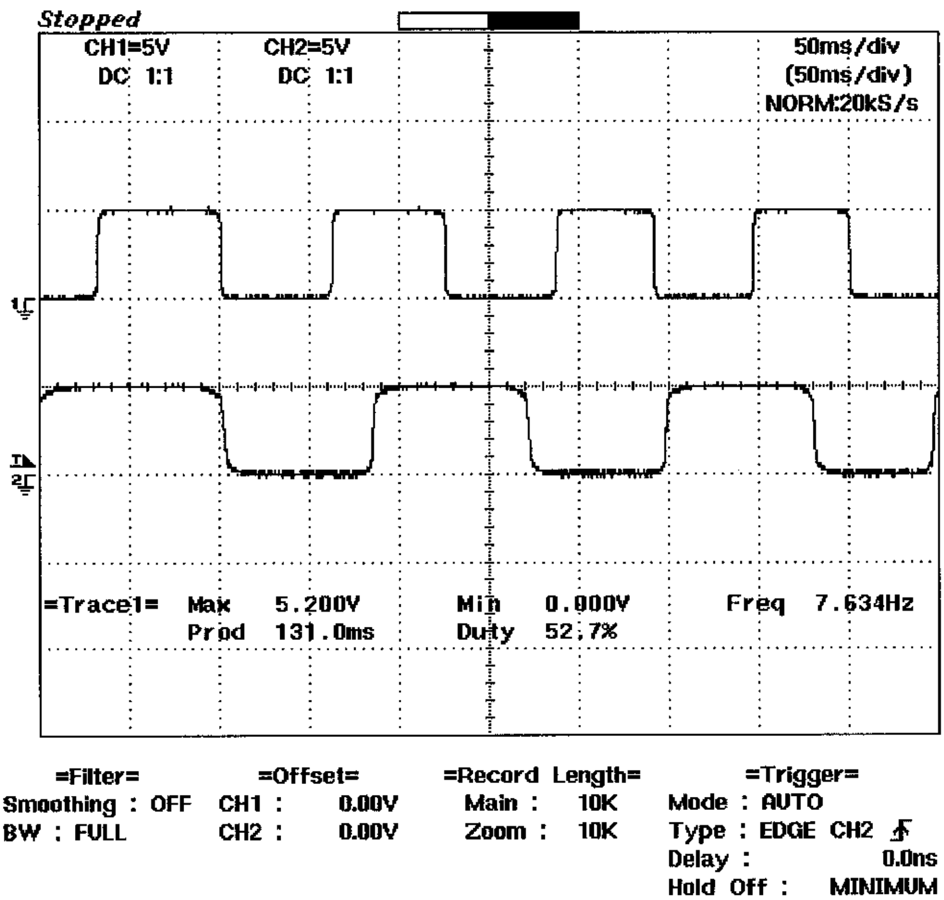


Figure 28 The output signal from encoders

The above signals in **Figure 28** show the output PWM signal captured from encodes reading fed into the microprocessors. From the encoder, we can see that with the PWM value for both motor is different because the speed on each motor is actually different. This digital signal obtained will be manipulated by microcontroller in order to get the speed and distance travel. The top signal refers to the left wheel encoder and the bottom signal refers to the right wheel encoders. The rotation of the wheel can now be represented in terms of pulse. This pulse is used to measure the distance travel by the robot. The distance value is also use in error correction codes to improve the accuracy of the robot.

4.6 SPDT relay

5V Single Pole Double Throw relay will be used to activate the 12V power window motor. The relay was used since the motor require direct drive to lift up more than 2kg load. For the SPDT Single Pole Double Throw Relays, the relay has three connections. Common, Normally Open, and Normally Closed. When the relay is off, the common is connected to the normally closed connection of the relay. When the relay coil is energized, the Common swings over to the Normally Open Connection of the relay.

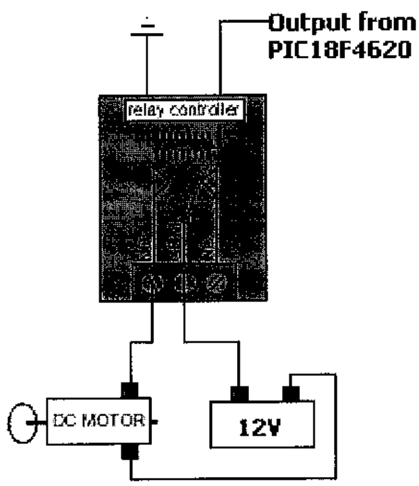


Figure 29 Relay connection to the power window motor

Therefore, by looking the relay connection in **Figure 29** above, when there is not input from the PIC18F4620, the relay coil will be de-energize and the common will be connected to the normally closed connection of the relay. When there is 5V input from the PIC18F4620, the relay coil is energized, the Common swings over to the Normally Open Connection of the relay. The motor will be activated and lifts up the ball frame until the lifter trigger the high limit switch. The output from the high limit switch will trigger the microcontroller to stop to energize the relay coil. Thus the ball frame will be stop at 1.8m height.

4.7 Servomotor board

low width (ms)	high Width (us)	Position of servo	Input from PIC 18F4620
18	2250	90 degrees left	001
18	1800	45 degrees left	010
18	1350	Straight (0 degrees)	011
18	810	45 degrees right	100
18	450	90 degrees right	101

Table 5 The signal received by servo motor from main controller

Servo controller will use to generate a precise PWM output to drive the servo motor. The servo controller will receive signal from the main microcontroller as referred on **Table 5** above. The servo controller will then translate the movement and prepare the needed pulse so that the gate attached to the servo is moved to the desired position.

Servomotor must use a separate PIC from main controller as to avoid any delay in the flow of programming process in the main microcontroller. This is because servomotor requires a continuous pulse from the PIC to maintain the shaft position of servo. Therefore, every servomotor will require a dedicated servo controller to control it. From the **Figure 30** below, the main controller will send a pulse to PIC16F84A and the servo controller will continuously generate pulses to maintain the servo on the desired position. PIC16F84A was used because servo does not require a large memory since it is solely used to generate a continuous pulse to the servomotor. As described above, the servomotors will be used to control the feeding mechanism of the robot and there will be 3 servos were used to control the opening of the gate.

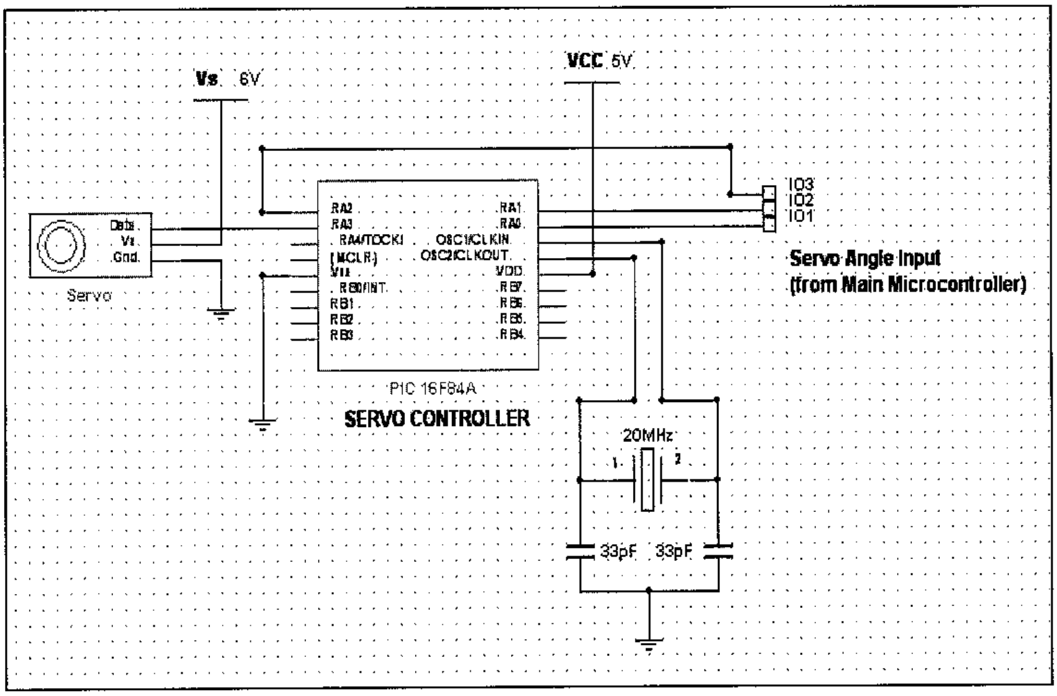


Figure 30 Layout of a servo controller

Servo motor requires a large current (up to 1A) and also will introduce a large amount of noise on to the power rail. Therefore in most cases the servo should be powered from a separate 6V power supply as shown in **Figure 30** above.

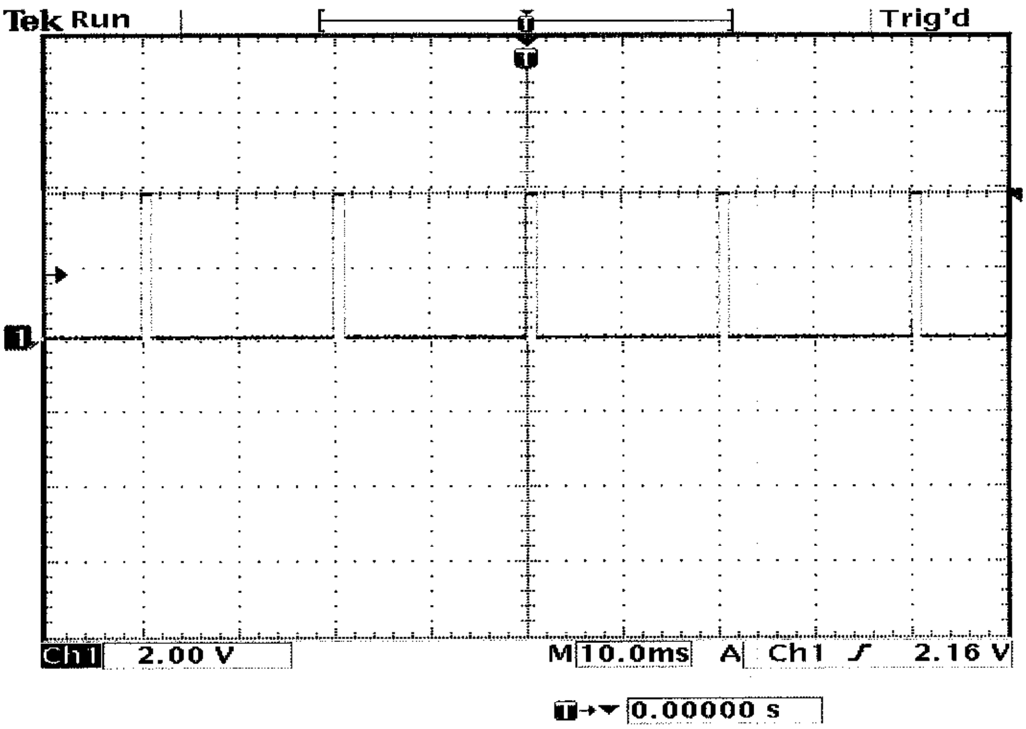


Figure 31 The input signal fed into the servo controller for 001 input

The above PWM signal in **Figure 31** shows 001 output signal from main controller fed into the servo controller board. The signal shows the high width duty ratio is 2250us and the width duty ratio is 20ms. This digital signal obtained will be manipulated by servo controller in order to get the angle desire. Please refer to **Appendix C** for the Visio circuit of Servomotor board.

4.8 Line follower

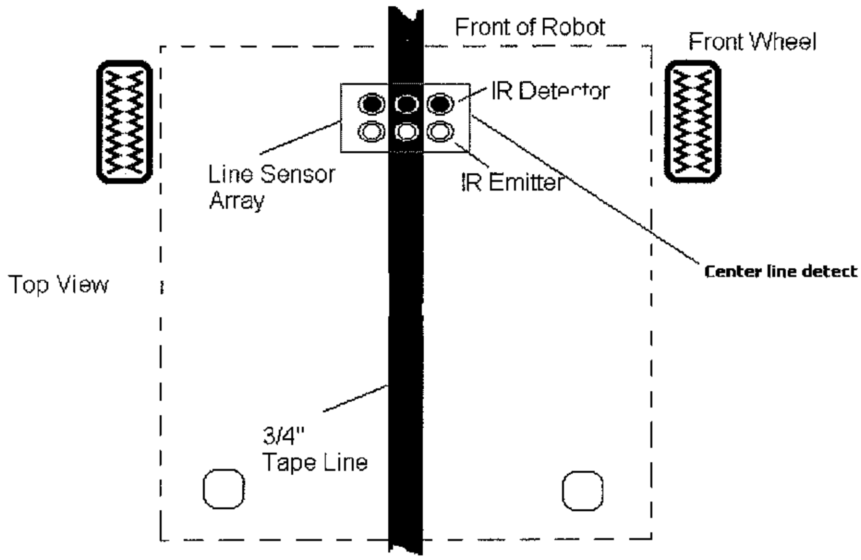


Figure 32 The placement of line follower on the robot

The module of line follower sensor described in the literature review has been used for this project. From **Figure 32**, the three pairs of sensors are used to keep the robot on the center line as it moves. Each center sensor output is monitored to determine the location of the tape relative to the robot. The main objective of the robot is to position itself such that the tape line falls between the two extreme sensors. If the tape line ever ventures past these two extreme sensors, then the robot will correct by turning in the appropriate direction to maintain tracking. Therefore, error correction algorithm need to be applied as to make sure that the robot does not deviate from the line. The error correction algorithm will be described base on the diagram in **Figure 33** below.

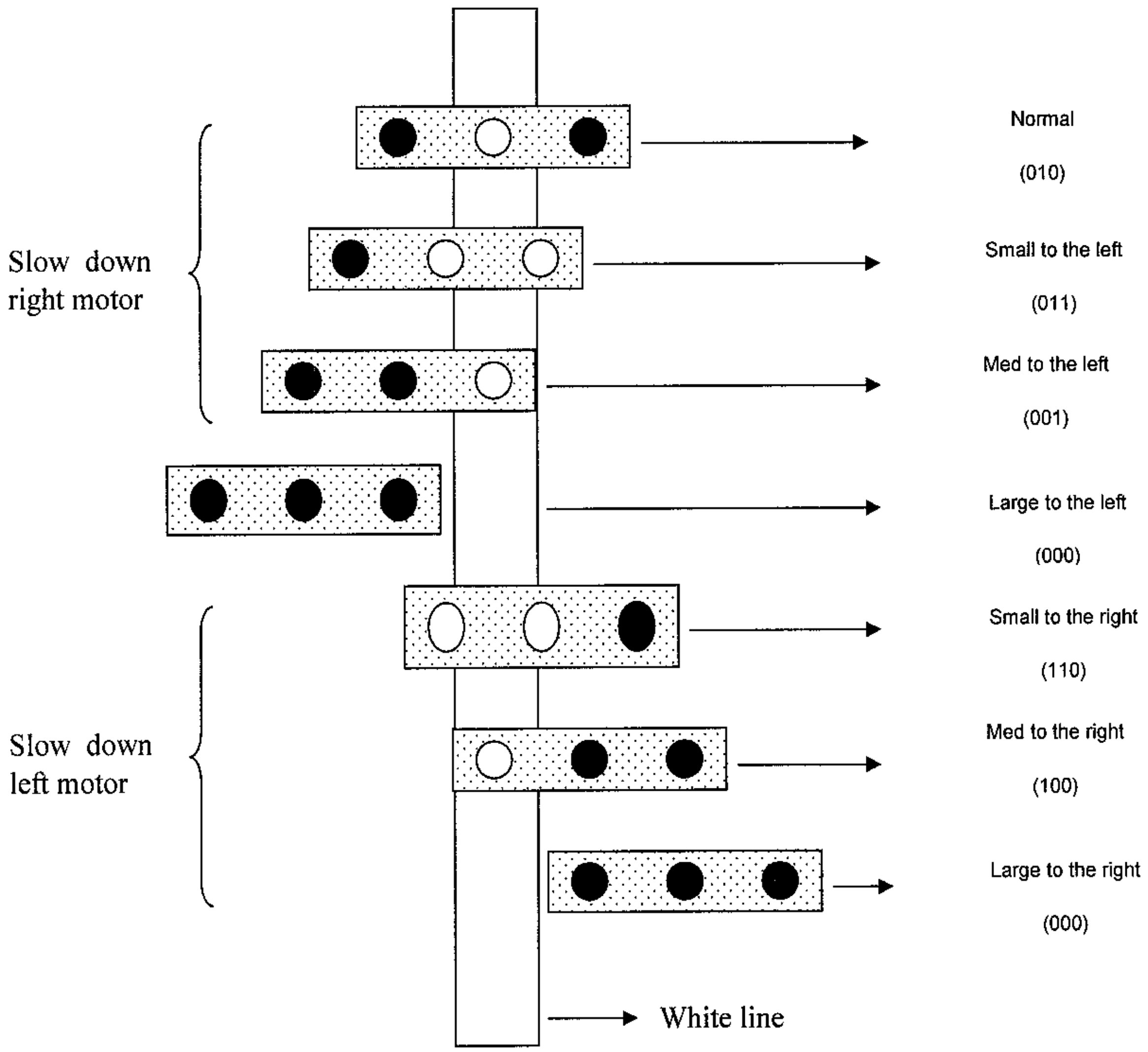


Figure 33 Error correction algorithm

For the line follower error correction algorithm, let if the sensor detects the white surface, the sensor will sense as bit 1 and if the black surface the sensor will sense as bit 0. Base on the **Figure 33,** there is a few condition need to be considered. For the first condition, the robot is in the normal condition where the center sensor detects white surface and the other detect the black surface. Thus, the line follower sensor will send the signal 010 to the main controller and the main controller will process the data. Base on the data received, the main controller will send the output signal to motor so that the motor will move in the normal duty ratio value as to go straight. For the second condition, the robot will deviate to the left and the condition is called 'small to the left'. For this condition, the center and the right sensor detect the white surface while the left sensor detects the black surface. The line follower will send 011 signals to the main controller and the main controller will process the data. Base on the data received, the main controller needs to make some error correction for this condition. For the error correction, the duty ratio for the right motor will be reduced so that the left motor will able to bring back the robot to the normal position. The error correction will continue for the other conditions stated above. Every single condition will have different percentage of duty ratio controlled since it depends how far the robot deviated from the normal condition. The correction duty ratio for the condition 'large to the left' is greater than the correction duty ratio for 'medium to the left'. Therefore, all the duty ratio values will be base on try and error correction and the programmer need to insert the values after observing the movement of the robot. For this algorithm, the conflict will occur for the condition 'large to the left' and 'large to the right'. This is because, both will send the same signal to main controller which is 000. As to solve this problem, the signal will depend on the previous condition for the robot. If the previous condition of the robot was 'medium to the left', so the main controller will now that the robot has deviated to the 'large to the left'.

4.9 Programming part

The movement of the robot will be base on the line follower sensor or pre-programmable movement. By using the line follower sensor, the robot will have the error correction movement where the microcontroller will make sure that the robot will move straight by using error correction algorithm. However by using the line follower sensor, the robot will move slowly since it takes time to correct the movement of the robot. While, the pre-programmable will use pre determine path to move the robot. The distance and direction of the robot will be determined in the programming and once the robot is moved, the robot will follow the movement base on the programmed in the microcontroller. The disadvantage of the pre-programmable is it might not move straight since there is no error correction algorithm. Thus, the possibility the robot will not move straight is high because it depends on the floor condition and the initial position of the wheels. However, the advantages of using the pre-programmable is it can move faster than using the line follower sensor. The whole programming part for the line follower sensor will be described in the flow chart in **Figure 34** below while the whole programming part for the pre-programmable movement will be showed in **Figure 35** below.

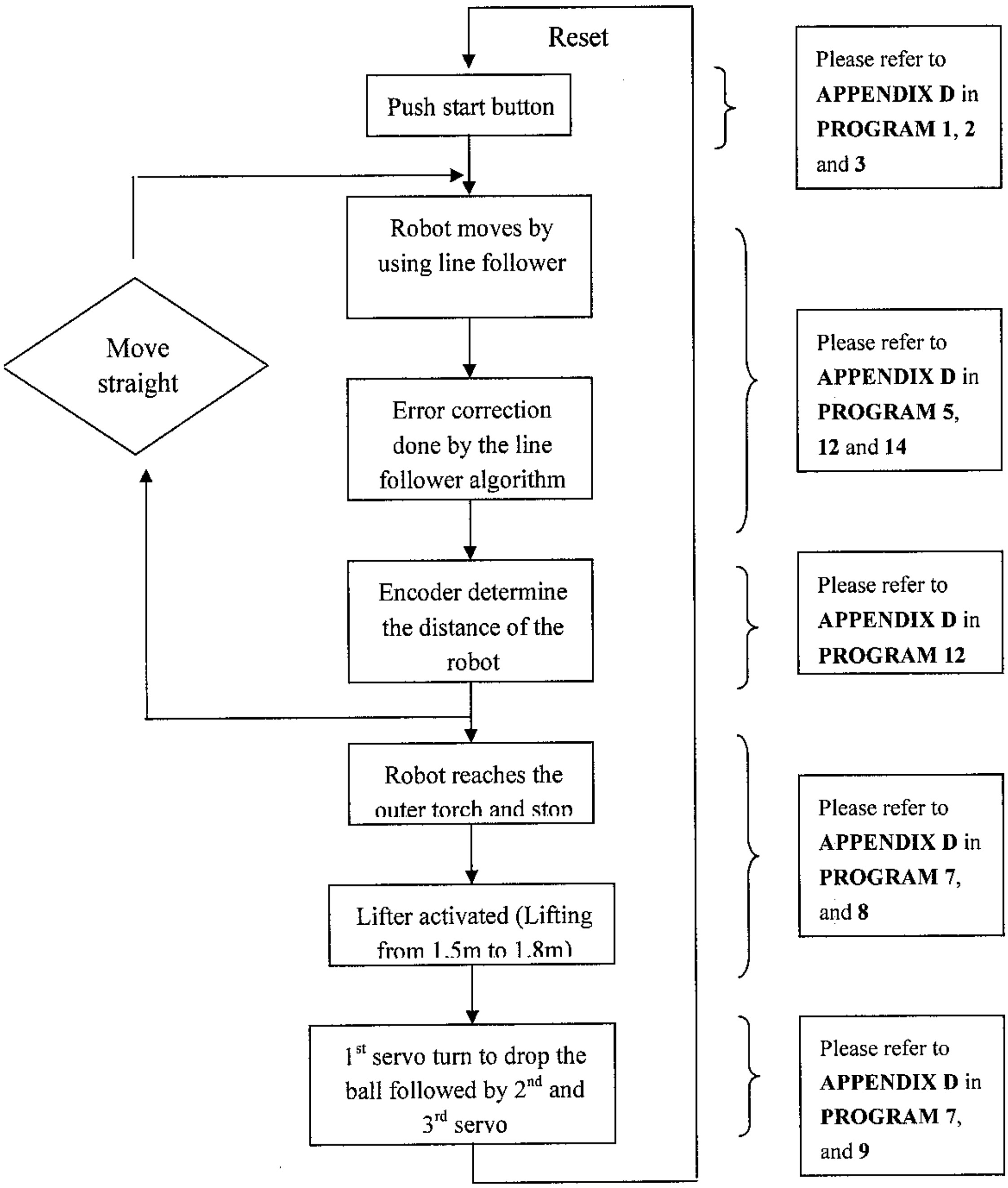


Figure 34 The flow of the robot movement by using the line follower sensor

4.9.1 Line follower robot operation

Initially, the robot will be placed on the white line of the starting area. The center line follower sensor must be put on the white line and the other two line follower sensor on the black floor as to make sure the robot will initially in normal movement condition. Once the start button was pressed, the robot will start to move forward. Meanwhile, the center line follower sensor will try to maintain the position on the white line of the floor. Once the sensor deviate from the white line, the error correction algorithm will be executed as to make sure the robot can move straight. The encoder reading will be used to determine the distance of the robot movement. The distance will be determined base on the programmed values. For example, if the robot needs to move 100cm forward, the value of 100cm will be inserted into the program. The encoder and the line follower will continuously execute the distance and error detection algorithm until the robot reaches to the outer torch. When the robot reaches the outer torch, the relay will be activated. Lifter motor will turn the sprocket to lift up the ball frame and the limit switch will be used to stop the ball frame to 1.8m high. Once the ball frame is elevated, the first servo motor will be activated. The first gate will be opened to drop the balls into the outer torch. After 20 seconds, the gate will be closed back. The robot will move 15cm forward to position the second gate at the outer torch. Then the servo motor for the second gate will open the gate to drop the balls into the outer torch. After 20 seconds, the gate will be closed back. The robot will move to the other outer torch to feed the ball for the gate number 3. When all the balls are fed, the robot will be stopped automatically.

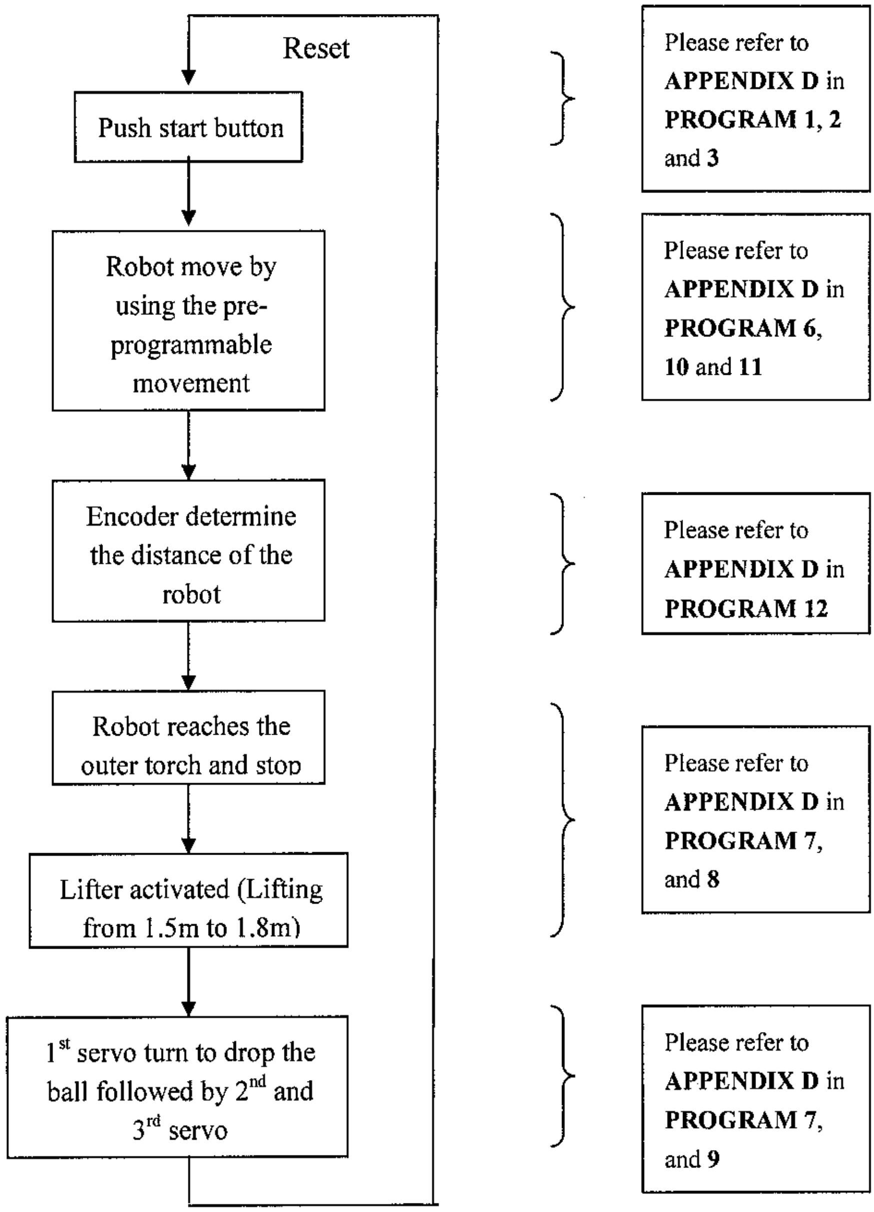


Figure 35 The flow of the robot movement by using the pre-programmable movement

4.9.2 Pre-programmable robot operation

The pre-programmable operation will look same with the line follower operation. The different is, the program will look simpler than the line follower sensor since there does not has an error correction algorithm by the line follower sensor. Initially, the robot will be placed in the starting area. The robot must be place straight enough as to make sure the robot can move straight forward. Once the start button was pressed, the robot will start to move forward. Meanwhile, the distance and the motor speed will be determined base on the programmed values. For example, if the robot needs to move 100cm forward with high speed, the value of 100cm and high duty ratio will be inserted into the program. The encoder will continuously execute the stripes reading until the robot reaches to the outer torch. When the robot reaches the outer torch, the relay will be activated. Lifter motor will turn the sprocket to lift up the ball frame and the limit switch will be used to stop the ball frame to 1.8m high. Once the ball frame is elevated, the first servo motor will be activated. The first gate will be opened to drop the balls into the outer torch. After 20 seconds, the gate will be closed back. The robot will move 15cm forward to position the second gate at the outer torch. Then the servo motor for the second gate will open the gate to drop the balls into the outer torch. After 20 seconds, the gate will be closed back. The robot will move to the other outer torch to feed the ball for the gate number 3. When all the balls are fed, the robot will be stopped automatically.

For the C programming part, the program will be explained in more detail in the attachment in **Appendix D**.

4.10 Final result

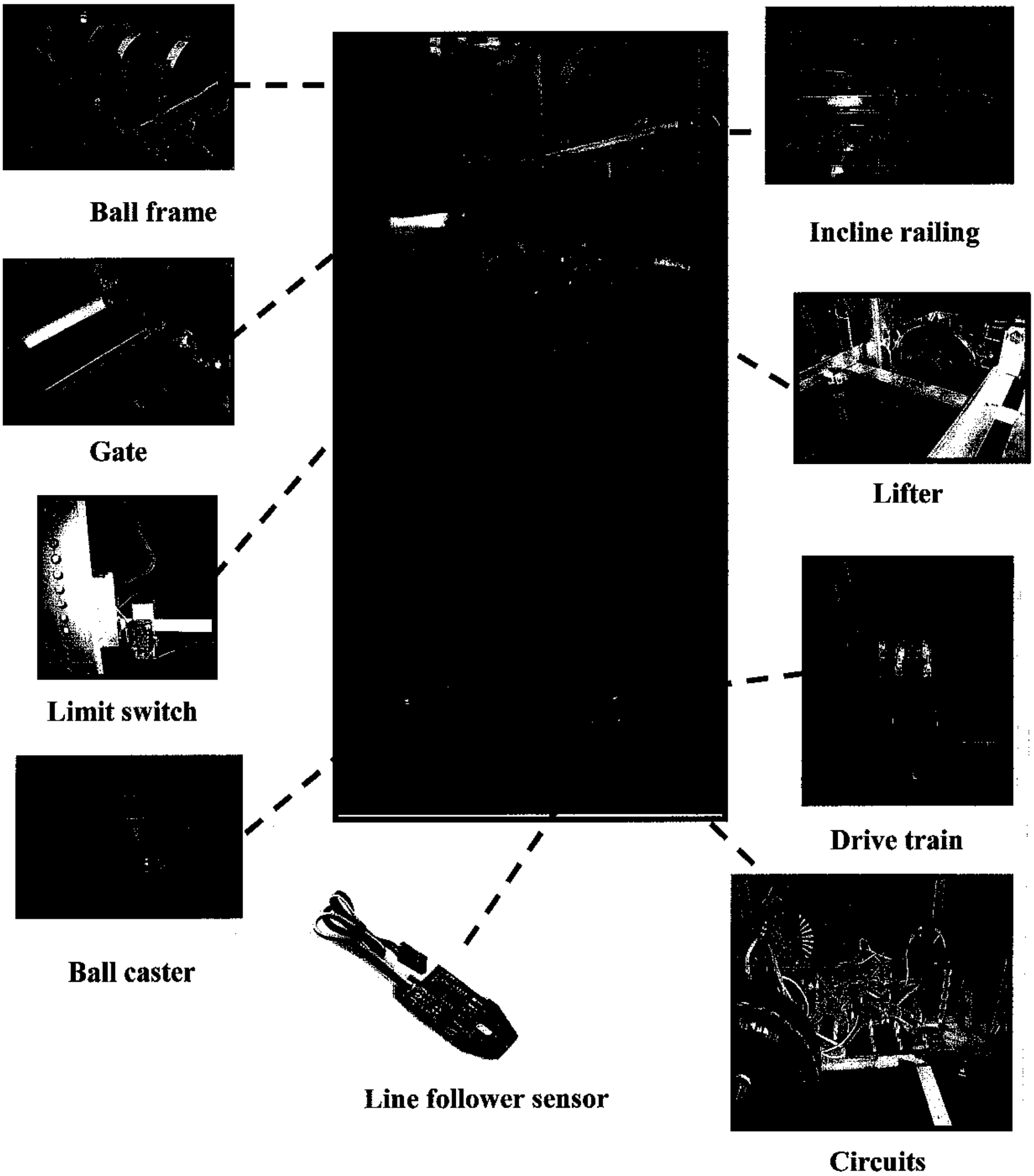


Figure 36 Final output of autonomous ball feeder mobile robot

CHAPTER 5

CONCLUSION AND RECOMMENDATION

As a conclusion, the author manage to achieve the objective of this project which is designing a simple ball feeder robot by using the PIC and C programming as a controller part for the robot. However the structure part of the robot still can be improved to make the design more rugged and lighter.

5.1 Drive train

The problem occurred when both of the wheels did not stop exactly on the desired distance. This is because almost 2 to 3cm overshoot will occur once both of the wheels tend to stop on the desired distance. As to solve the problem by mechanically, the couplers for both of the wheels have been tight more and the wheels also covered with the rubber layer. The rubber layer can help the wheels become more grip with the floor especially when the robot tend to stop. As to solve the problem by programming, the slow down and breaking system has been applied on the wheels. The slow down system has been applied by reducing the duty ratio of the motor 20cm before the robot stop. The additional of reverse command at the end of the robot movement can also help the robot stops on the spot.

5.2 Noise

The high current from the power source can create the electromagnetic field (EMF) effect. EMF can affect the functionality of the electronics since it creates noise around the wire. In this project, some cases has been seen when power source wire was put closely with the data wire. The data received by the main controller become intermittent and incorrect. The worse condition occurred when the power source wire was put across the main controller. The EMF can disturb the flow of the program inside the main controller. As to overcome the problem, all the data and power source wires were put far away to eliminate the EMF affect. Thus, a proper wiring is needed as to fix this problem.

5.3 Circuit

The fabrications of the circuit also need to be considered since the robot looks messy with the improper wiring on the veroboard circuit. A lot of problems occurred when the veroboard was used to construct the circuit. The circuits look messy with the wires and there was a short circuit problem when some of the circuits were not unconnected properly. The circuit can still be improved by using the PCB as to make the circuit design more systematic. The troubleshooting problem will be easier if the PCB were utilized for this project.

5.4 Line follower algorithm

The algorithm of line follower still can be improved by applying the PID concept in the error correction algorithm. By using the PID concept, the movement of the robot will become smoother than the existing algorithm. However, the number of sensor needs to be added more as to get the effectiveness of the algorithm used.

5.5 Wheel size

By using the smaller wheel, the effectiveness of the error correction algorithm can be improved. This is because it can reduce the overshoot of the wheels during the movement of the robot. The robot also manages to move back to the normal position faster than using the big wheel.

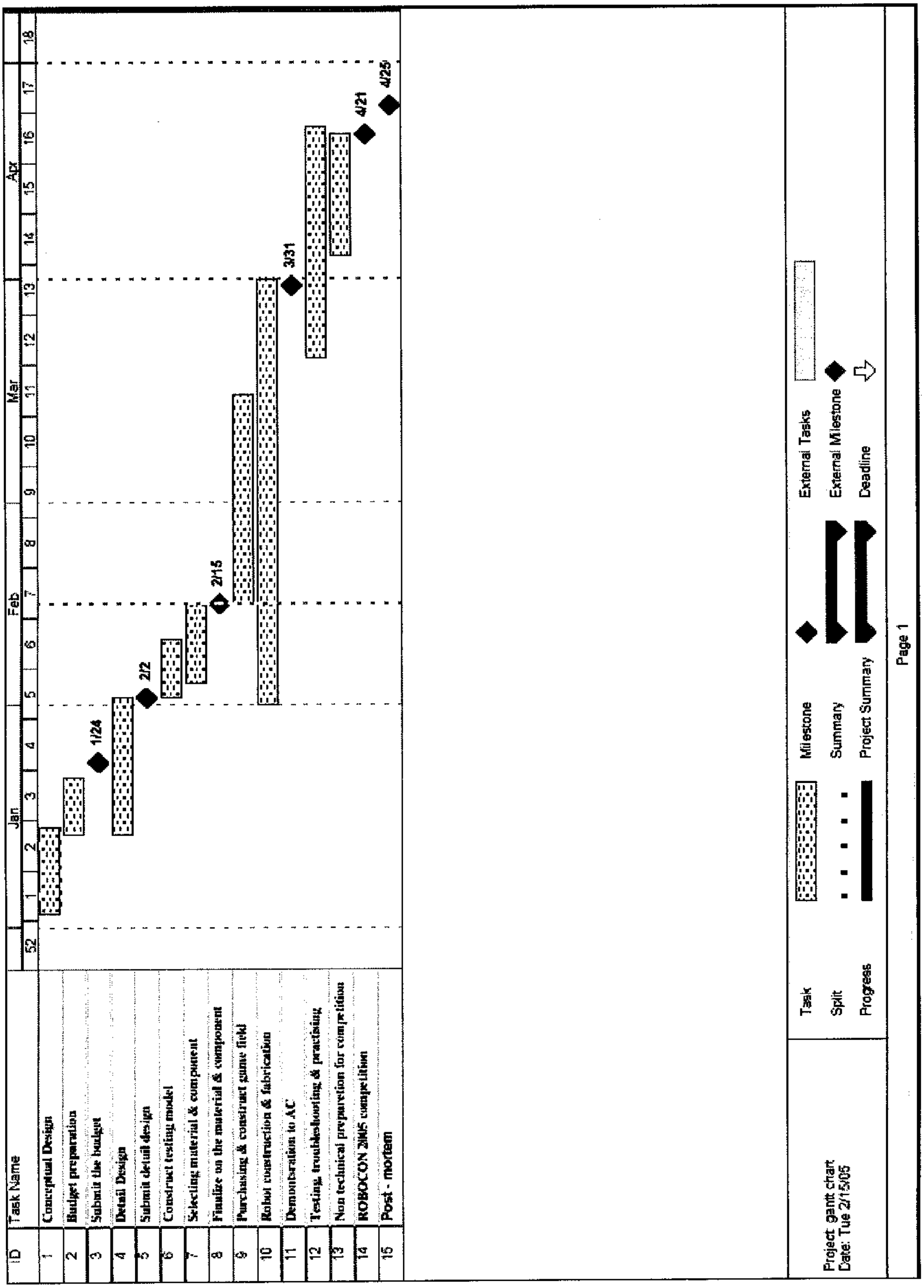
CHAPTER 6
REFERENCES

1. http://www.abu.org.my/public/compiled
2. http://www.robotroom.com/HBridge.html
3. http://www.dprg.org/tutorials/
4. http://www.robot-electronics.co.uk
5. http://www.lynxmotion.com
6. http://www.robotstore.com
7. http://my.farnell.com/jsp/home/homepage.jsp
8. http://www.rsmalaysia.com
9. http://www.parallax.com
10. http://www.seattlerobotics.org/encoder/index.html
11. http://www.abc.net.au/science/robocon/build.htm
12. http://www.kbs.co.kr/aburobocon2004/ABU2002_theme.html
13. Ulrich Nehmzow,1999,Mobile Robotics:A practical Introduction, Springer
14. Battle Bots the official guide, Clarkson, Mc Graw Hill
15. Build Your Own Robot1, Karl Lunt
16. Sensors for Mobile Robot ,Theory and Application, H.R Everett

APPENDICES

APPENDIX A

GANTT CHART



APPENDIX B

GAME FIELD OUTLINE AND THE DIMENSION OF OUTER TORCH

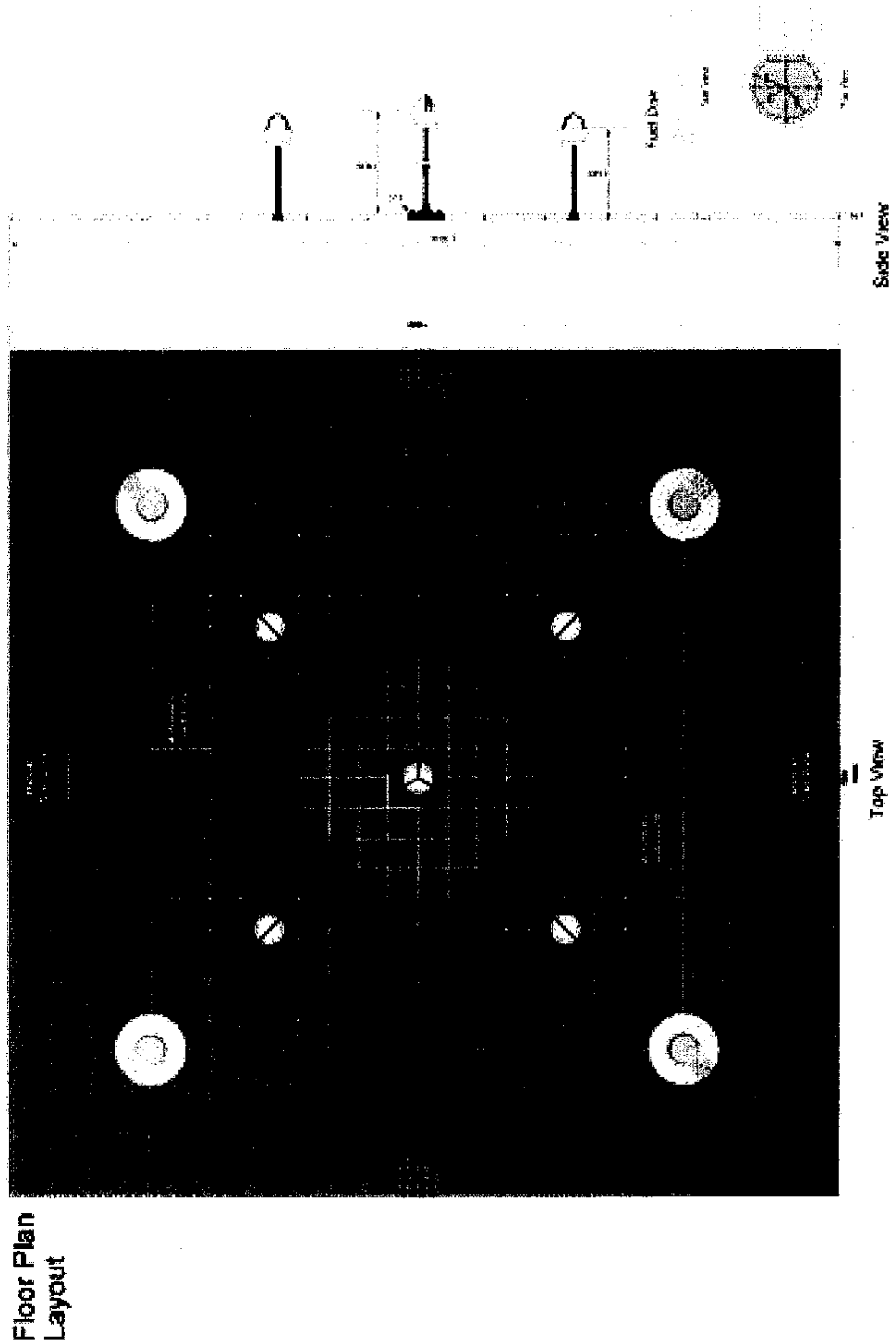


Figure 37 The other view of the game field

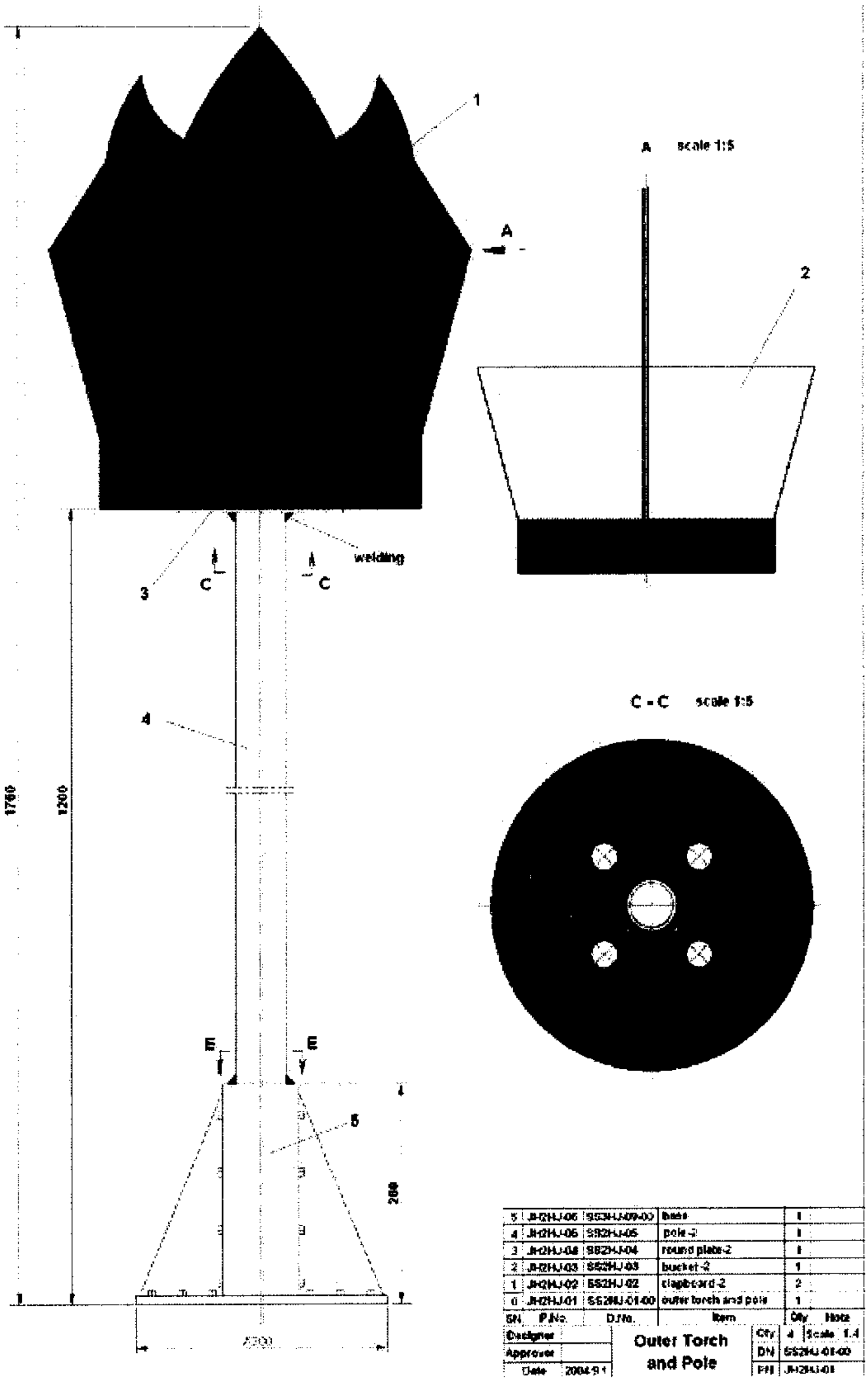


Figure 38 The detail dimension of the outer torch

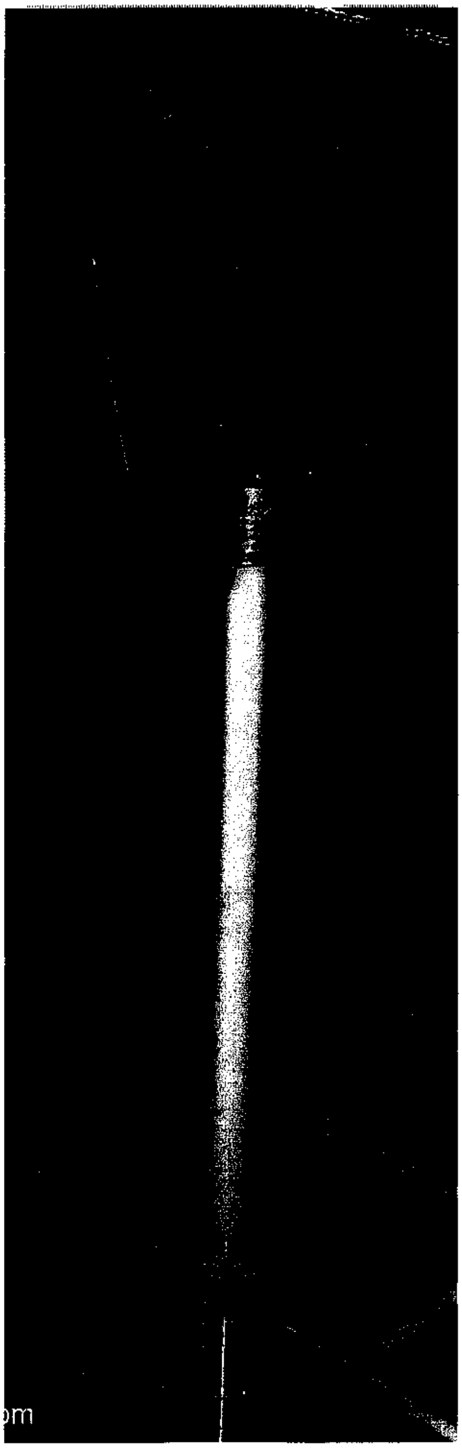


Figure 39 The Outer Torch

APPENDIX C

THE CIRCUITS OUTLINE

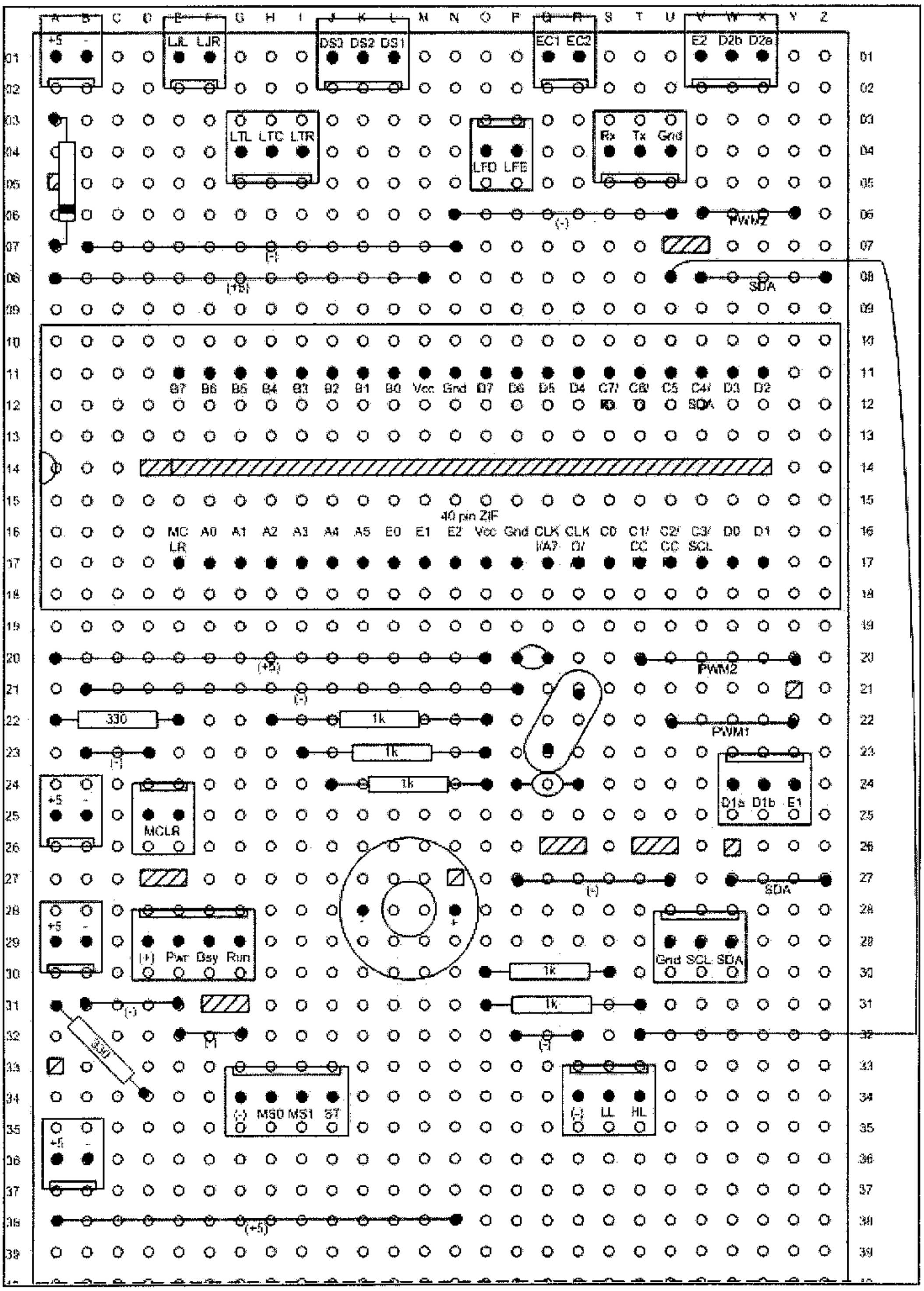


Figure 40 The main controller circuit layout

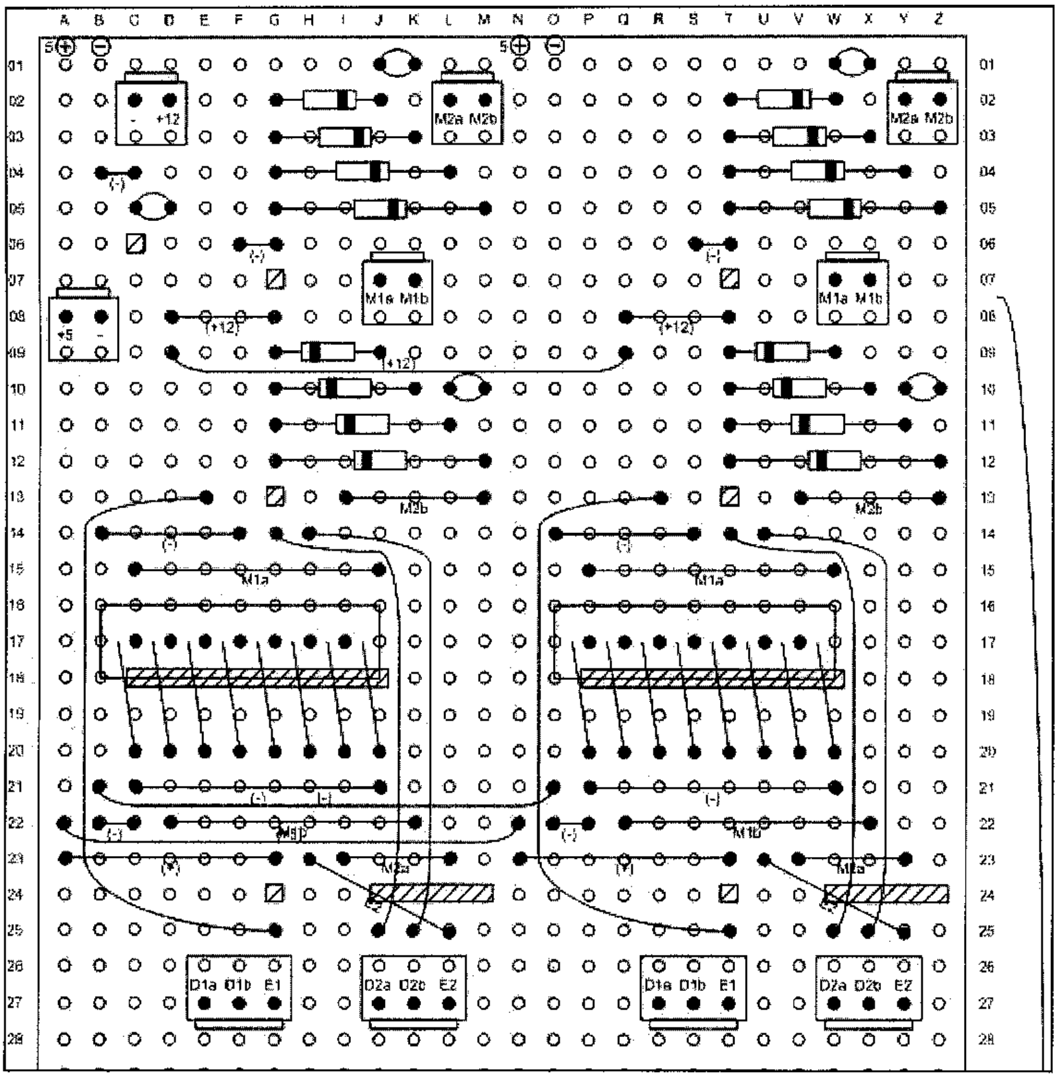


Figure 41 The L298N H-Bridge circuit layout

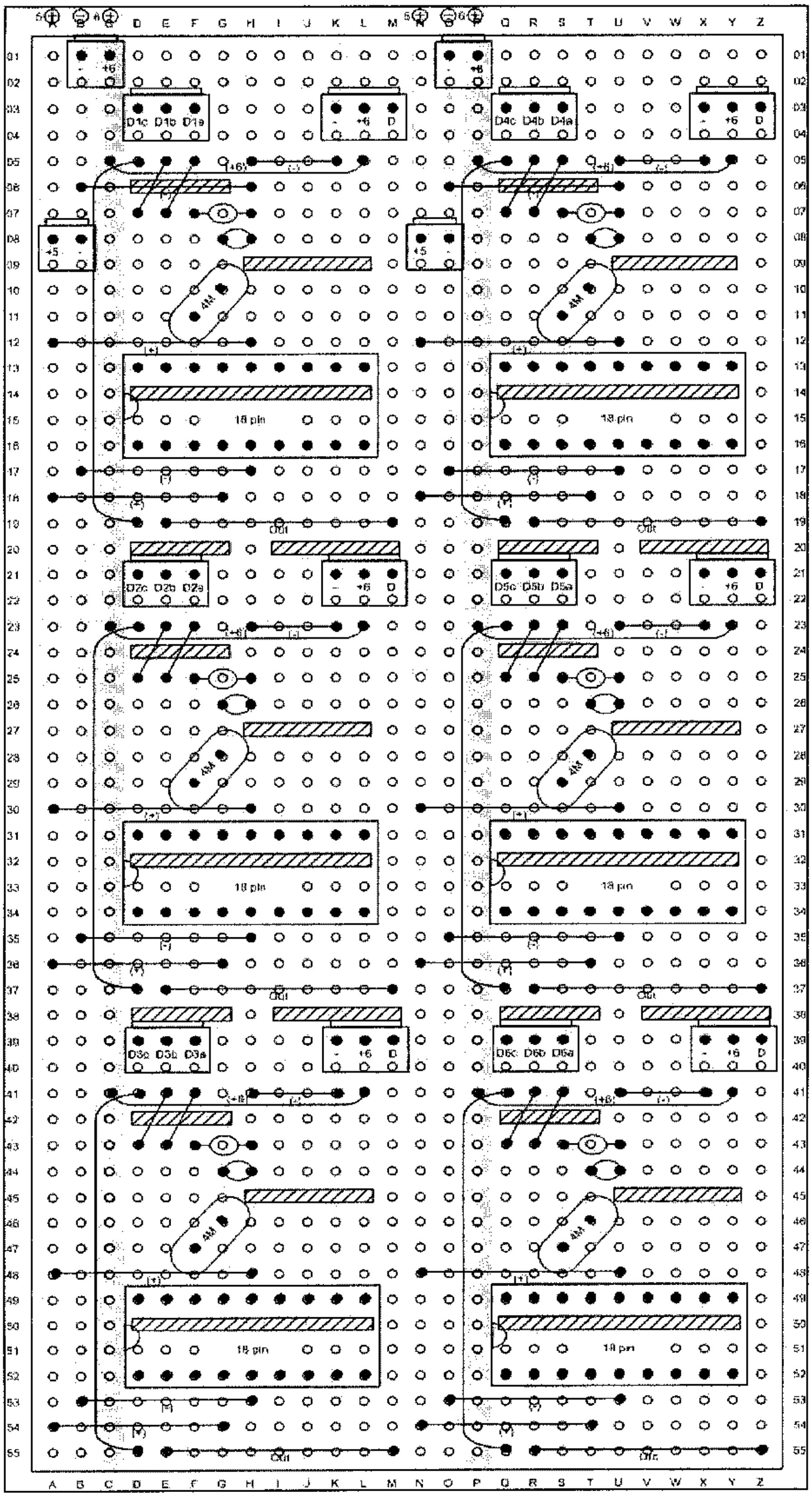


Figure 42 The servo motor controller circuit layout

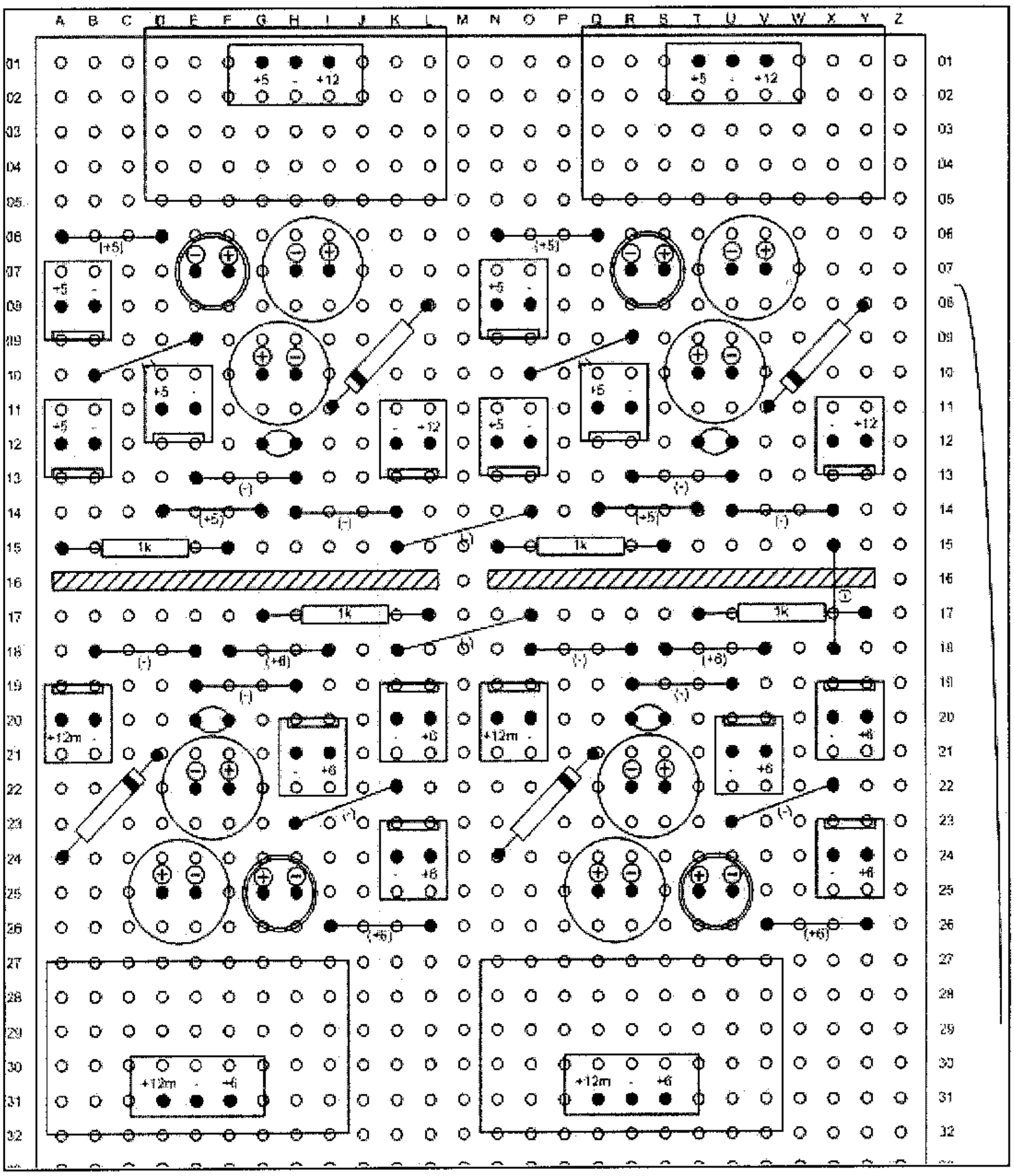


Figure 43 The power supply circuit layout

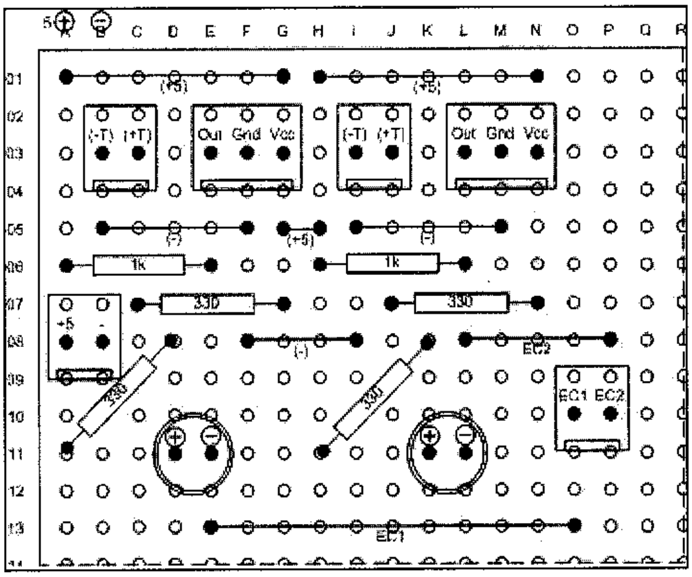


Figure 44 The rotary encoder circuit layout

APPENDIX D

PEOGRAMMING C CODING SOURCE

PROGRAM 1

Initialize the programming by declaring all the necessary variables required

```
#include <18F4620.H>
#use delay(clock=20000000)
#fuses  HS,NOPROTECT,NOWDT,NOLVP,CCP2C1
#use rs232(baud=56000,xmit=PIN_C6,rcv=PIN_C7)
```

PROGRAM 2

Declare the calling function (void) pre-programmable and line tracking for robot movement. The function for lifting, servomotor controller and buzzer also declared on this phase.

```
void PREPROGRAM_FORWARDREVERSE(int motor_dir,long distance_in_cm);
void PREPROGRAM_TURN(int motor_dir,long distance_in_cm);

void LINETRACKER_FOLLOW(int is_junction_detect, int sensor_set, int line_num, long distance_in_cm, int line_color);

void LINETRACKER_FORWARDREVERSE_DETECT(int motor_dir, int sensor_set, int line_num, int line_color);

void LINETRACKER_TURN_DETECT(int motor_dir, int sensor_set, int line_num, int line_color);

void MOTOR_MOVEMENT(int motor_left_dir,int motor_right_dir,long dutyratio_left,long dutyratio_right);

void LIFTER(int lifter_dir);
void DOOR(int servo_num,int servo_dir);

void BUZZER_BEEP(int maxcount);
```

PROGRAM 3

Insert the robot parameters for both pre-programmable and line tracking movement. The parameters include duty ratio for left and right wheels and the distance per pulse for the encoder calculation.

```
//--For general--
//distance_perpulse=2*pi*j/60
//j=9.75 cm
double distance_per_pulse=1.02;
```

Distance per pulse calculation

```
//--For pre-programmable movement--
long default_dutyratio_left=700;
long default_dutyratio_right=800;

long pp_low_dutyratio_left=600;
long pp_low_dutyratio_right=600;

long ppturn_low_dutyratio_left=500;
long ppturn_low_dutyratio_right=500;

long pp_slowdown_dutyratio_left=0;
long pp_slowdown_dutyratio_right=0;

long pp_delay_intermovement=1000;          //in ms
```

Duty ratios values of left and right wheels for forward and reverse movement, turning and breaking will be inserted through this portion.

```
//--For line tracking movement--
long lt_low_dutyratio_left=410;
long lt_low_dutyratio_right=440;

long ltturn_low_dutyratio_left=500;
long ltturn_low_dutyratio_right=500;

long ltfwdrev_low_dutyratio_left=400;
long ltfwdrev_low_dutyratio_right=480;

long lt_delay_intermovement=750; //in ms
```

Duty ratios on left and right wheels for forward and reverse movement will be inserted in this portion. The ratios also included the line tracking error correction for small, medium and large of left and right motor slow down.

```
long lt_stopdelay_short_left=100;  //in ms
long lt_stopdelay_short_right=115; //in ms

long lt_stopdelay_long_left=130;   //in ms
long lt_stopdelay_long_right=145;  //in ms
long large_overshoot_recursive_delay=500; //in ms

//---------------------
```

PROGRAM 4

Once the power supply is turned ON, the program below will be executed. All the mechanism used like drive train motor, lifter motor, servomotor and main controller board are in the initialized condition. The program will be looping while waiting the start push button is pressed.

```
void main()
{
  //Busy indicator
output_bit(PIN_A0,0);

//Initialize servo
        DOOR(0,0);
        DOOR(1,0);
        DOOR(2,0);

//Initialize lifter relay
        LIFTER(0);

//Initialize main controller board
  //Buzzer
        output_bit(PIN_A5,1);
  //Start indicator
        output_bit(PIN_A1,1);

//Initialize motor PWM
  //PWM=1250 Hertz
  setup_timer_2(T2_DIV_BY_16,249,1);
        setup_ccp1(CCP_OFF);
        setup_ccp2(CCP_OFF);
```

```
//Initialize motor
        MOTOR_MOVEMENT(0,0,0,0);

  //Busy indicator
        output_bit(PIN_A0,1);

  //Waiting for start pushbutton
  while(input(PIN_A4)==1)
  {
  }
  //Start indicator
        output_bit(PIN_A1,0);
        BUZZER_BEEP(1);
```

PROGRAM 5

If using the line tracking movement, this program will be executed. The values will be inserted base on the function declared in the initial phase of program.

Example:

LINETRACKER_FOLLOW (int is_junction_detect, int sensor_set, int line_num, long distance_in_cm, int line_color);

```
delay_ms(5000);
LINETRACKER_FOLLOW(0, 0, 0, 20, 0);
 LINETRACKER_FOLLOW(0, 0, 0, 20, 0);
 LINETRACKER_FOLLOW(0, 0, 0, 20, 0);
 LINETRACKER_FOLLOW(0, 0, 0, 20, 0);
 LINETRACKER_FOLLOW(0, 0, 0, 20, 0);
```

PROGRAM 6

If using the pre-programmable, this program will be executed. The values inserted base on the function declared in the initial phase of program.

Example:

PREPROGRAM_FORWARDREVERSE(*int motor_dir,long distance_in_cm*)

---

motor_dir	Movement pattern	Distance calculation
0	Forward	Average
1	Reverse	Average

PREPROGRAM_TURN(int motor_dir,long distance_in_cm)

motor_dir	Movement pattern	Distance calculation
0	Rotate right	Average
1	Rotate left	Average
2	Turn forward right	Left
3	Turn reverse right	Left
4	Turn forward left	Right
5	Turn reverse left	Right

```
PREPROGRAM_FORWARDREVERSE(0, 155);
PREPROGRAM_TURN(2, 50);
PREPROGRAM_FORWARDREVERSE(0, 100);
```

PROGRAM 7

After the robot reach the outer torch, the lifting and scoring mechanism program below will be executed.

```
//--Lifting--
delay_ms(5000);
LIFTER(1);
while(input(PIN_C5)==1)
{
}
    LIFTER(0);

delay_ms(1000);
```

Lifter motor will be lifted up the ball frame and stop once the limit switch is activated.

```
//--Lifting End--

//--Scoring--
DOOR(0,1);
delay_ms(5000);
DOOR(0,0);
delay_ms(1000);
```

The gate for 1st servomotor activated and feed the ball into the outer torch

```
PREPROGRAM_FORWARDREVERSE(0,6);

DOOR(1,1);
delay_ms(5000);
DOOR(1,0);
delay_ms(1000);
```

The gate for 2nd servomotor activated and feed the ball into the outer torch

```
PREPROGRAM_FORWARDREVERSE(0,6);

DOOR(2,1);
delay_ms(5000);
//--Scoring End--
```

The gate for 3rd servomotor activated and feed the ball into the outer torch

---

PROGRAM 8

The function of lifter will be called from the programming below

---

```
void LIFTER(int lifter_dir)
{
  //lifter_dir: 0=Stop, 1=Up
  switch(lifter_dir)
  {
//Stop
        case 0:
```

```
{
    //Switch: Off
                output_bit(PIN_D6,1);
    break;
}

//Up
        case 1:
{
//Switch: On
                output_bit(PIN_D6,0);
    break;
}

}
```

PROGRAM 9

The function of servo motor will be called from programming below.

```
void DOOR(int servo_num,int servo_dir)
{
//servo_num: 0=Door servo 1, 1=Door servo 2, 2=Door servo 3
//servo_dir: 0=Close, 1=Open

    switch(servo_num)
 {
    //Door servo 1
        case 0:
{
                output_bit(PIN_B0,servo_dir);

    break;
}
```

```
//Door servo 2
        case 1:
{
                output_bit(PIN_B1,servo_dir);

    break;
}

//Door servo 3
        case 2:
{
                output_bit(PIN_B2,servo_dir);

    break;
}
```

PROGRAM 10

The function of forward and reverse pre-programmable will be called from the programming below. The function also includes the encoder calculation distance.

```
void PREPROGRAM_FORWARDREVERSE(int motor_dir,long distance_in_cm)
{
  double average_distance;
  double total_distance_required;
  double distance_required_fwd;
  double distance_required_rv;

  long total_pulse_right;
  long total_pulse_left;

  int current_right_encoder;
  int previous_right_encoder;
  int current_left_encoder;
```

```
int previous_left_encoder;


long delay_counter;


//---------------------------------------------
//Read encoder and initialize total pulse
//---------------------------------------------


//Initialize total pulse
total_pulse_right=0;
total_pulse_left=0;
average_distance=0.0;
total_distance_required=(double)distance_in_cm;


//SLOW-DOWN ALGO
//Minus 20 cm (to implement slow-down algo)
if(distance_in_cm>=70)
{
  distance_required_fwd=total_distance_required-20.0;
}
else
{
  distance_required_fwd=total_distance_required;
}
//Read current encoder (assign it as previous encoder)
previous_right_encoder=input(PIN_D4);
previous_left_encoder=input(PIN_D5);


//-------------------------------------
//Change motor direction and run
//-------------------------------------
switch(motor_dir)
{
                //Forward
```

```
    case 0:
 {
          if(distance_in_cm>=70)
          {
          MOTOR_MOVEMENT(1,1,default_dutyratio_left,default_dutyratio_right);
  }
  else
          {
          MOTOR_MOVEMENT(1,1,pp_low_dutyratio_left,pp_low_dutyratio_right);
  }
                 break;
 }

          //Reverse
    case 1:
 {
          if(distance_in_cm>=70)
          {
          MOTOR_MOVEMENT(2,2,default_dutyratio_left,default_dutyratio_right);
  }
  else
          {
          MOTOR_MOVEMENT(2,2,pp_low_dutyratio_left,pp_low_dutyratio_right);
  }

                 break;
 }
}

//------------------------------
//*****Counting encoder pulse*****
    //------------------------------
while(average_distance<distance_required_fwd)
{
```

```
//Read both encoder output
  current_right_encoder=input(PIN_D4);
  current_left_encoder=input(PIN_D5);

  //If any of the encoder changes
  if (current_right_encoder!=previous_right_encoder || current_left_encoder!=previous_left_encoder)
  {

          //*****INCREASE ENCODER PULSE******
  //If right encoder changes
       if (current_right_encoder!=previous_right_encoder)
          {
 //Debounce algorithm
        previous_right_encoder=current_right_encoder;
        total_pulse_right=total_pulse_right+1;
          }

         //If left encoder changes
       if (current_left_encoder!=previous_left_encoder)
         {
          //Debounce algorithm
         previous_left_encoder=current_left_encoder;
       total_pulse_left=total_pulse_left+1;
          }

 //AVERAGE DISTANCE CALCULATION
 //Calculate average distance

average_distance=((distance_per_pulse*((double)total_pulse_right+(double)total_pulse_left))/2.0);

 }

 //-----------------------------------------------------------------
 //*****Start slow-down and reverse algo (ONLY if distance_in_cm>=30 cm) *****
```

```
//--------------------------------------------------------------------
//If distance_in_cm more than 70 cm
if(distance_in_cm>=70)
{

//****SLOW-DOWN ALGO****
//------------------------------------
  //*****Change motor direction and run*****
//------------------------------------
          switch(motor_dir)
        {
                        //Forward
                case 0:
            {

        MOTOR_MOVEMENT(1,1,pp_slowdown_dutyratio_left,pp_slowdown_dutyratio_right);
                                break;
        }

                        //Reverse
                case 1:
        {

        MOTOR_MOVEMENT(2,2,pp_slowdown_dutyratio_left,pp_slowdown_dutyratio_right);
                                break;
        }
        }

  //------------------------------
  //*****Counting encoder pulse*****
  //------------------------------
  while(average_distance<total_distance_required)
  {
        //Read both encoder output
```

```
current_right_encoder=input(PIN_D4);
current_left_encoder=input(PIN_D5);

//If any of the encoder changes
if (current_right_encoder!=previous_right_encoder ||
current_left_encoder!=previous_left_encoder)
{

//*****INCREASE ENCODER PULSE******
  //If right encoder changes
  if (current_right_encoder!=previous_right_encoder)
  {
   //Debounce algorithm
   previous_right_encoder=current_right_encoder;
   total_pulse_right=total_pulse_right+1;
  }

  //If left encoder changes
  if (current_left_encoder!=previous_left_encoder)
  {
   //Debounce algorithm
   previous_left_encoder=current_left_encoder;
   total_pulse_left=total_pulse_left+1;
  }

  //*****AVERAGE DISTANCE CALCULATION******
  //Calculate average distance
average_distance=((distance_per_pulse*((double)total_pulse_right+(double)total_pulse_left))/2.0);

}
}
   //****FINISH SLOW-DOWN ALGO****
```

```
  //*****REVERSE DIRECTION INSTANTANEOUSLY*****

//------------------------------------------------
  //*****Read encoder and initialize total pulse*****
//------------------------------------------------

  //Initialize total pulse
total_pulse_right=0;
  total_pulse_left=0;
average_distance=0.0;
  distance_required_rv=(double)(4);

//Read current encoder (assign it as previous encoder)
  previous_right_encoder=input(PIN_D4);
  previous_left_encoder=input(PIN_D5);

//------------------------------------------------
  //*****Change motor direction and run (in reverse)*****
//------------------------------------------------

switch(motor_dir)
  {
        //Forward
case 0:
    {
        MOTOR_MOVEMENT(2,2,default_dutyratio_left,default_dutyratio_right);
                    break;
    }

  //Reverse
case 1:
    {
MOTOR_MOVEMENT(1,1,default_dutyratio_left,default_dutyratio_right);
                    break;
```

```
        }
    }

    //-------------------------------
    //*****Counting encoder pulse*****
    //-------------------------------
    while(average_distance<distance_required_rv)
    {
        //Read both encoder output
        current_right_encoder=input(PIN_D4);
        current_left_encoder=input(PIN_D5);

    //If any of the encoder changes
if (current_right_encoder!=previous_right_encoder || current_left_encoder!=previous_left_encoder)
        {

            //*****INCREASE ENCODER PULSE******
    //If right encoder changes
            if (current_right_encoder!=previous_right_encoder)
            {
                //Debounce algorithm
        previous_right_encoder=current_right_encoder;
    total_pulse_right=total_pulse_right+1;
            }

        //If left encoder changes
        if (current_left_encoder!=previous_left_encoder)
    {
            //Debounce algorithm
            previous_left_encoder=current_left_encoder;
        total_pulse_left=total_pulse_left+1;
    }
```

```
//*****AVERAGE DISTANCE CALCULATION******

//Calculate average distance

average_distance=((distance_per_pulse*((double)total_pulse_right+(double)total_pulse_left))/2.0);

//*****FINISH REVERSE DIRECTION INSTANTANEOUSLY*****
}

//--------------
//*****Stop*****
//--------------
MOTOR_MOVEMENT(0,0,0,0);
//Delay
for (delay_counter=0;delay_counter<=pp_delay_intermovement;delay_counter++)
{
    delay_ms(1);
}
```

PROGRAM 11

The function for the line follower for the forward movement will be called from the program below.

```
void LINETRACKER_FOLLOW(int is_junction_detect, int sensor_set, int line_num, long distance_in_cm, int line_color)
{
/*
  is_junction_detect:
    0=Do not detect junction (use distance calculation to stop, based on distance_in_cm)
    1=Detect junction to stop (based on line_num)

  sensor_set:
        0=Junction sensor (left)
```

```
        1=Junction sensor (right)

line_color:
        0=Black
  1=White
*/

/*
SL    SC    SR         dir    line_check
 0    0     0          <-          3
 0    0     1                      2
 0    1     1                      1
 0    1     0                      0
 1    1     0                      4
 1    0     0                      5
 0    0     0          ->          6
*/

//For junction detection
int previous_detect;
int current_detect;
int total_line;

//For distance calculation
double average_distance;
double total_distance_required;

long total_pulse_right;
long total_pulse_left;

int current_right_encoder;
int previous_right_encoder;
int current_left_encoder;
int previous_left_encoder;
```

```
//For line tracking
int line_check;
int line_dir;

//For general
long delay_counter;

//--------------------------------
//*****DISTANCE CALCULATION ALGO*****
//--------------------------------
if (is_junction_detect==0)
{

  //---------------------------------------------
  //*****Read encoder and initialize total pulse*****
  //---------------------------------------------

  //Initialize total pulse
  total_pulse_right=0;
  total_pulse_left=0;
  average_distance=0.0;
  total_distance_required=(double)distance_in_cm;

  //Read current encoder (assign it as previous encoder)
  previous_right_encoder=input(PIN_D4);
  previous_left_encoder=input(PIN_D5);



//--------------------
//*****Run slowly*****
        //--------------------
        //printf("Line follow (distance)\r\n");
        line_check=0;
```

```
line_dir=0;
        MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

while(average_distance<total_distance_required)
{
        //----------------------------
        //*****LINE TRACKING ALGO*****
                //--------------------------
                //If center
if ((input(PIN_B5)!=line_color && input(PIN_B4)==line_color && input(PIN_B3)!=line_color) ||
(input(PIN_B5)==line_color && input(PIN_B4)==line_color && input(PIN_B3)==line_color))
 {
        if (line_check!=0)
{
//If previously straight
                if (line_dir==0)
        {
  //Left=Default,Right=Default

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=0;
}
else
//If previously towards right
                if (line_dir==1)
        {
  //Left=Stop short period,Right=Default

MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  for (delay_counter=0;delay_counter<=lt_stopdelay_short_left;delay_counter++)
  {
        delay_ms(1);
  }
```

```
MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

line_dir=0;
}
else
//If previously towards left
            if (line_dir==2)
    {
//Left=Default,Right=Stop short period

MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);

for (delay_counter=0;delay_counter<=lt_stopdelay_short_right;delay_counter++)
{
      delay_ms(1);
}


MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

line_dir=0;
}

                        line_check=0;
  }
}
  else
//If small overshoot to the left
if (input(PIN_B5)!=line_color && input(PIN_B4)==line_color && input(PIN_B3)==line_color)
  {
      if (line_check!=1)
{
                        //If previously towards left
```

```
                    if (line_dir==1)
            {

 //Left=Stop short period,Right=Default
//
        MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

//          for (delay_counter=0;delay_counter<=lt_stopdelay_short_left;delay_counter++)
//          {
//               delay_ms(1);
//          }

//          MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);



 //Left=Default,Right=Default

        MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

          line_dir=1;
          //line_dir=0;
        }
        else
        //Else
            {
          //Left=Default,Right=Stop short period

        MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);

          for (delay_counter=0;delay_counter<=lt_stopdelay_short_right;delay_counter++)
          {
               delay_ms(1);
          }
```

```
        MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

          line_dir=0;
        }
                                        line_check=1;
           }
        }
           else
                         //If small overshoot to the right
if (input(PIN_B5)==line_color && input(PIN_B4)==line_color &&   input(PIN_B3)!=line_color)
           {
              if (line_check!=4)
          {
                                        //If previously towards right
              if (line_dir==2)
              {

 //Left=Default,Right=Stop short period
//
        MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);

//          for (delay_counter=0;delay_counter<=lt_stopdelay_short_right;delay_counter++)
//          {
//              delay_ms(1);
//          }

//
        MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);


     //Left=Default,Right=Default

        MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);
          line_dir=2;
```

```
//line_dir=0;
}
else
//Else
        {
//Left=Stop short period,Right=Default

MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

for (delay_counter=0;delay_counter<=lt_stopdelay_short_left;delay_counter++)
{
        delay_ms(1);
}

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

line_dir=0;
}
                                line_check=4;
    }
}
else
                //If medium overshoot to the left
if (input(PIN_B5)!=line_color && input(PIN_B4)!=line_color && input(PIN_B3)==line_color)
{
        if (line_check!=2)
    {
                                //If previously towards left
                if (line_dir==1)
        {
//Left=Default,Right=Stop short period

MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);

for (delay_counter=0;delay_counter<=lt_stopdelay_short_right;delay_counter++)
```

```
        {
                delay_ms(1);
        }

      MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);


        line_dir=1;
      }
      else
      //Else
              {
       //Left=Default,Right=Stop long period

      MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);


        for (delay_counter=0;delay_counter<=lt_stopdelay_long_right;delay_counter++)
        {
                delay_ms(1);
        }

      MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);


        line_dir=0;
      }

                                            line_check=2;

            }
        }
      else
//If medium overshoot to the right
if (input(PIN_B5)==line_color && input(PIN_B4)!=line_color && input(PIN_B3)!=line_color)
      {
              if (line_check!=5)
          {
//If previously towards left
```

```
                    if (line_dir==2)
          {
  //Left=Stop short period,Right=Default

MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  for (delay_counter=0;delay_counter<=lt_stopdelay_short_left;delay_counter++)
  {
        delay_ms(1);
  }

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=2;
}
else
//Else
          {
//Left=Stop long period,Right=Default

MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  for (delay_counter=0;delay_counter<=lt_stopdelay_long_left;delay_counter++)
  {
        delay_ms(1);
  }

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=0;
}
                              line_check=5;
     }
  }
```

```
else
if (input(PIN_B5)!=line_color && input(PIN_B4)!=line_color && input(PIN_B3)!=line_color)
  {
                            //If large overshoot to the left
  if (line_check==1 || line_check==2)
     {
  //Left=Default,Right=Stop long period

  MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);

  for (delay_counter=0;delay_counter<=lt_stopdelay_long_right;delay_counter++)
  {
    delay_ms(1);
  }

  MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=1;
  line_check=3;
  }
     else
//If large overshoot to the right
        if (line_check==4 || line_check==5)
  {
  //Left=Stop long period,Right=Default

  MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  for (delay_counter=0;delay_counter<=lt_stopdelay_long_left;delay_counter++)
  {
    delay_ms(1);
  }

  MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);
```

```
line_dir=2;
                                    line_check=6;
  }


    }
                    //---LINE TRACKING ALGO FINISH---
```

PROGRAM 12

The distance algorithm for the robot will be executed by using the coding below. The program will read the input received from the encoder to be translated into pulse. The pulse will be send to the drive train motor to determine the desire distance

```
                //---DISTANCE CALCULATION ALGO---
               //Read both encoder output
   current_right_encoder=input(PIN_D4);
  current_left_encoder=input(PIN_D5);


      //If any of the encoder changes
if (current_right_encoder!=previous_right_encoder || current_left_encoder!=previous_left_encoder)
       {


            //*****INCREASE ENCODER PULSE******
              //If right encoder changes
            if (current_right_encoder!=previous_right_encoder)
              {
                    //Debounce algorithm
            previous_right_encoder=current_right_encoder;
            total_pulse_right=total_pulse_right+1;
                }


              //If left encoder changes
            if (current_left_encoder!=previous_left_encoder)
              {
                    //Debounce algorithm
             previous_left_encoder=current_left_encoder;
```

```
total_pulse_left=total_pulse_left+1;
    }



//*****AVERAGE DISTANCE CALCULATION******
        //Calculate average distance

average_distance=((distance_per_pulse*((double)total_pulse_right+(double)total_pulse_left))/2.0);
                        }
//---DISTANCE CALCULATION ALGO FINISH---


    }


}
    else
//-----------------------------------
```

PROGRAM 14

The error correction algorithm will be executed by using the algorithm below. The algorithm will be the same like have been described in the discussion part in page 41 above.

```
//*****LINE TRACKING ALGO*****
        //--------------------------

//If center
if ((input(PIN_B5)!=line_color && input(PIN_B4)==line_color && input(PIN_B3)!=line_color) ||
(input(PIN_B5)==line_color && input(PIN_B4)==line_color && input(PIN_B3)==line_color))
    {
        if (line_check!=0)
    {
                        //If previously straight
            if (line_dir==0)
        {
    //Left=Default,Right=Default
```

```
MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=0;
}
else
//If previously towards right
                if (line_dir==1)
        {
  //Left=Stop short period,Right=Default

MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  for (delay_counter=0;delay_counter<=lt_stopdelay_short_left;delay_counter++)
  {
        delay_ms(1);
  }

  MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=0;
}
else
                                //If previously towards left
                if (line_dir==2)
        {
  //Left=Default,Right=Stop short period

MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);

  for (delay_counter=0;delay_counter<=lt_stopdelay_short_right;delay_counter++)
  {
        delay_ms(1);
  }
```

```
MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=0;
 }

                                    line_check=0;

   }
 }
    else
                //If small overshoot to the left
if (input(PIN_B5)!=line_color && input(PIN_B4)==line_color && input(PIN_B3)==line_color)
    {
        if (line_check!=1)
  {
                                    //If previously towards left
                if (line_dir==1)
          {
  //Left=Default,Right=Default

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=1;
  //line_dir=0;
 }
 else
                if (line_dir==3)
          {
  //Left=Default,Right=Default

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=1;
 }
 else
```

```
//Else
        {
  //Left=Default,Right=Stop short period

MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);

  for (delay_counter=0;delay_counter<=lt_stopdelay_short_right;delay_counter++)
  {
        delay_ms(1);
  }

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=0;
 }

                              line_check=1;
   }
 }
    else
                //If small overshoot to the right
if (input(PIN_B5)==line_color && input(PIN_B4)==line_color && input(PIN_B3)!=line_color)
    {
        if (line_check!=4)
  {
                              //If previously towards right
                 if (line_dir==2)
        {
  //Left=Default,Right=Default

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=2;
  //line_dir=0;
```

```
}
else
                if (line_dir==4)
        {
  //Left=Default,Right=Default

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=2;
}
else
//Else
        {
  //Left=Stop short period,Right=Default

MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  for (delay_counter=0;delay_counter<=lt_stopdelay_short_left;delay_counter++)
  {
        delay_ms(1);
  }

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=0;
}

                                line_check=4;
    }
  }
else
                //If medium overshoot to the left
if (input(PIN_B5)!=line_color && input(PIN_B4)!=line_color && input(PIN_B3)==line_color)
{
        if (line_check!=2)
```

```
{
                                        //If previously towards left
        if (line_dir==1)
    {
  //Left=Default,Right=Stop short period

MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);

  for (delay_counter=0;delay_counter<=lt_stopdelay_short_right;delay_counter++)
  {
        delay_ms(1);
  }


MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=1;
}
else
//Else
    {
  //Left=Default,Right=Stop long period

MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);

  for (delay_counter=0;delay_counter<=lt_stopdelay_long_right;delay_counter++)
  {
        delay_ms(1);
  }


MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=3;
}
```

```
                              line_check=2;
      }
    }
  else
                    //If medium overshoot to the right
if (input(PIN_B5)==line_color && input(PIN_B4)!=line_color && input(PIN_B3)!=line_color)
  {
      if (line_check!=5)
    {
                              //If previously towards left
            if (line_dir==2)
        {
    //Left=Stop short period,Right=Default

  MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

    for (delay_counter=0;delay_counter<=lt_stopdelay_short_left;delay_counter++)
    {
          delay_ms(1);
    }

  MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

    line_dir=2;
  }
  else
  //Else
        {
    //Left=Stop long period,Right=Default

  MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

    for (delay_counter=0;delay_counter<=lt_stopdelay_long_left;delay_counter++)
    {
```

```
                delay_ms(1);
            }

        MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

          line_dir=4;
        }

                                                line_check=5;
              }
            }
        else
if (input(PIN_B5)!=line_color && input(PIN_B4)!=line_color && input(PIN_B3)!=line_color)
            {
                                        //If large overshoot to the left
                if (line_check==1 || line_check==2)
              {

        //Pre-assigned (to make sure run the error correction at the first place)
        large_overshoot_counter=large_overshoot_recursive_delay;

        //Start extreme recursive correction
        //-----------------
        //If still off track
    while(input(PIN_B5)!=line_color && input(PIN_B4)!=line_color && input(PIN_B3)!=line_color)
        {
          if (large_overshoot_counter==large_overshoot_recursive_delay)
          {
                        //Left=Default,Right=Stop long period

        MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);

                        for
(delay_counter=0;delay_counter<=lt_stopdelay_long_right;delay_counter++)
```

```
{
        delay_ms(1);
}

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

    large_overshoot_counter=0;
  }

//Discrete timer
        delay_ms(1);

  //Increase timer counter
  large_overshoot_counter=large_overshoot_counter+1;
}
//------------------

line_dir=1;
                        line_check=3;
}
      else
                  //If large overshoot to the right
        if (line_check==4 || line_check==5)
{

//Pre-assigned (to make sure run the error correction at the first place)
large_overshoot_counter=large_overshoot_recursive_delay;

//Start extreme recursive correction
//------------------
//If still off track
while(input(PIN_B5)!=line_color && input(PIN_B4)!=line_color && input(PIN_B3)!=line_color)
{
```

```
  if (large_overshoot_counter==large_overshoot_recursive_delay)
  {
                //Left=Stop long period,Right=Default

MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

        for (delay_counter=0;delay_counter<=lt_stopdelay_long_left;delay_counter++)
        {
        delay_ms(1);
                }

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

    large_overshoot_counter=0;
  }

//Discrete timer
        delay_ms(1);

  //Increase timer counter
  large_overshoot_counter=large_overshoot_counter+1;
}
//------------------

line_dir=2;
                                line_check=6;
  }

    }
                //---LINE TRACKING ALGO FINISH---

            //---DISTANCE CALCULATION ALGO---
        //Read both encoder output
 current_right_encoder=input(PIN_D4);
```

```
current_left_encoder=input(PIN_D5);

    //If any of the encoder changes
if (current_right_encoder!=previous_right_encoder || current_left_encoder!=previous_left_encoder)
    {

        //*****INCREASE ENCODER PULSE******
         //If right encoder changes
        if (current_right_encoder!=previous_right_encoder)
         {
              //Debounce algorithm
        previous_right_encoder=current_right_encoder;
        total_pulse_right=total_pulse_right+1;
          }

         //If left encoder changes
        if (current_left_encoder!=previous_left_encoder)
         {
              //Debounce algorithm
         previous_left_encoder=current_left_encoder;
        total_pulse_left=total_pulse_left+1;
          }


         //*****AVERAGE DISTANCE CALCULATION******
              //Calculate average distance

average_distance=((distance_per_pulse*((double)total_pulse_right+(double)total_pulse_left))/2.0);

          //~~Debug~~
                    }
         //---DISTANCE CALCULATION ALGO FINISH---

    }
```

```
}
        else
//-----------------------------------
//*****JUNCTION DETECTION ALGO*****
//-----------------------------------
if (is_junction_detect==1)
{

  //Initialize detection
  //KENA CHECK DULU SBLM JALAN!
                switch(sensor_set)
        {
                case 0:
            {
                current_detect=input(PIN_B7);
          break;
            }

          case 1:
        {
                current_detect=input(PIN_B6);
            break;
        }
  }

          previous_detect=current_detect;
          total_line=0;

          //--------------------
        //*****Run slowly*****
          //--------------------
          line_check=0;
  line_dir=0;
```

```
MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);



//-----------------
//*****Counting*****
//-----------------
while(total_line<line_num)
{

//--------------------------
//*****LINE TRACKING ALGO*****
//--------------------------
//If center
if ((input(PIN_B5)!=line_color && input(PIN_B4)==line_color && input(PIN_B3)!=line_color) ||
(input(PIN_B5)==line_color && input(PIN_B4)==line_color && input(PIN_B3)==line_color))
{
if (line_check!=0)
{
//If previously straight
if (line_dir==0)
{
//Left=Default,Right=Default

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

line_dir=0;
}
else
//If previously towards right
if (line_dir==1)
{
//Left=Stop short period,Right=Default

MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);
```

```
for (delay_counter=0;delay_counter<=lt_stopdelay_short_left;delay_counter++)
{
        delay_ms(1);
}

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

line_dir=0;
}
else
                                        //If previously towards left
                    if (line_dir==2)
        {
//Left=Default,Right=Stop short period

MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);

for (delay_counter=0;delay_counter<=lt_stopdelay_short_right;delay_counter++)
{
        delay_ms(1);
}

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

line_dir=0;
}

                                        line_check=0;
    }
}
    else
//If small overshoot to the left
if (input(PIN_B5)!=line_color && input(PIN_B4)==line_color && input(PIN_B3)==line_color)
```

```
{
    if (line_check!=1)
 {
                                //If previously towards left
            if (line_dir==1)
        {
  //Left=Default,Right=Default

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);


  line_dir=1;
  //line_dir=0;
}
else
            if (line_dir==3)
        {
  //Left=Default,Right=Default

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);


  line_dir=1;
}
else
//Else
        {
  //Left=Default,Right=Stop short period

MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);


  for (delay_counter=0;delay_counter<=lt_stopdelay_short_right;delay_counter++)
  {
        delay_ms(1);
  }

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);
```

```
line_dir=0;
}

line_check=1;

}
}
else
//If small overshoot to the right
if (input(PIN_B5)==line_color && input(PIN_B4)==line_color && input(PIN_B3)!=line_color)
{
if (line_check!=4)
{
//If previously towards right
if (line_dir==2)
{
//Left=Default,Right=Default

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

line_dir=2;
//line_dir=0;
}
else
if (line_dir==4)
{
//Left=Default,Right=Default

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

line_dir=2;
}
else
//Else
```

```
{
//Left=Stop short period,Right=Default

MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

for (delay_counter=0;delay_counter<=lt_stopdelay_short_left;delay_counter++)
{
delay_ms(1);
}

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

line_dir=0;
}

line_check=4;
}
}
else
//If medium overshoot to the left
if (input(PIN_B5)!=line_color && input(PIN_B4)!=line_color && input(PIN_B3)==line_color)
{
if (line_check!=2)
{
//If previously towards left
if (line_dir==1)
{
//Left=Default,Right=Stop short period

MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);

for (delay_counter=0;delay_counter<=lt_stopdelay_short_right;delay_counter++)
{
delay_ms(1);
```

```
}

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);


line_dir=1;
}
else
//Else
{
//Left=Default,Right=Stop long period

MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);


for (delay_counter=0;delay_counter<=lt_stopdelay_long_right;delay_counter++)
{
	delay_ms(1);
}

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);


line_dir=3;
}
				line_check=2;
	}
}
else
			//If medium overshoot to the right
if (input(PIN_B5)==line_color && input(PIN_B4)!=line_color && input(PIN_B3)!=line_color)
{
	if (line_check!=5)
	{
				//If previously towards left
		if (line_dir==2)
		{
//Left=Stop short period,Right=Default
```

```
MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  for (delay_counter=0;delay_counter<=lt_stopdelay_short_left;delay_counter++)
  {
          delay_ms(1);
  }

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=2;
}
else
//Else
          {
  //Left=Stop long period,Right=Default

MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  for (delay_counter=0;delay_counter<=lt_stopdelay_long_left;delay_counter++)
  {
          delay_ms(1);
  }

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

  line_dir=4;
}

                                        line_check=5;

      }
  }
else
```

```
if (input(PIN_B5)!=line_color && input(PIN_B4)!=line_color && input(PIN_B3)!=line_color)
        {
                                //If large overshoot to the left
              if (line_check==1 || line_check==2)
           {

        //Pre-assigned (to make sure run the error correction at the first place)
        large_overshoot_counter=large_overshoot_recursive_delay;

        //Start extreme recursive correction
        //-----------------
        //If still off track
    while(input(PIN_B5)!=line_color && input(PIN_B4)!=line_color && input(PIN_B3)!=line_color)
        {
          if (large_overshoot_counter==large_overshoot_recursive_delay)
          {
                        //Left=Default,Right=Stop long period

        MOTOR_MOVEMENT(1,0,lt_low_dutyratio_left,lt_low_dutyratio_right);

                        for
(delay_counter=0;delay_counter<=lt_stopdelay_long_right;delay_counter++)
                        {
                                delay_ms(1);
                        }


        MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

            large_overshoot_counter=0;
          }

                                        //Discrete timer
              delay_ms(1);
```

```
//Increase timer counter
large_overshoot_counter=large_overshoot_counter+1;
}
//-----------------

line_dir=1;
line_check=3;
}
else
//If large overshoot to the right
if (line_check==4 || line_check==5)
{

//Pre-assigned (to make sure run the error correction at the first place)
large_overshoot_counter=large_overshoot_recursive_delay;

//Start extreme recursive correction
//-----------------
//If still off track
while(input(PIN_B5)!=line_color && input(PIN_B4)!=line_color && input(PIN_B3)!=line_color)
{
if (large_overshoot_counter==large_overshoot_recursive_delay)
{
//Left=Stop long period,Right=Default

MOTOR_MOVEMENT(0,1,lt_low_dutyratio_left,lt_low_dutyratio_right);

for (delay_counter=0;delay_counter<=lt_stopdelay_long_left;delay_counter++)
{
delay_ms(1);
}

MOTOR_MOVEMENT(1,1,lt_low_dutyratio_left,lt_low_dutyratio_right);
```

```
  large_overshoot_counter=0;
 }

                                        //Discrete timer
     delay_ms(1);

 //Increase timer counter
 large_overshoot_counter=large_overshoot_counter+1;
}
//------------------

line_dir=2;
                                   line_check=6;
 }

  }
            //---LINE TRACKING ALGO FINISH---

     //---JUNCTION DETECTION ALGO---
            //Check line
            switch(sensor_set)
     {
            case 0:
     {
            current_detect=input(PIN_B7);
       break;
         }

       case 1:
     {
            current_detect=input(PIN_B6);
         break;
     }
```

```
}

                //If detected
 if (current_detect==line_color && previous_detect!=current_detect)
{
        previous_detect=current_detect;
  total_line=total_line+1;
}
  else
                //If not detected
if (current_detect!=line_color && previous_detect!=current_detect)
  {
        previous_detect=current_detect;
}
        //---JUNCTION DETECTION FINISH---

 }

}

//--------------
//*****Stop*****
//--------------
        MOTOR_MOVEMENT(0,0,0,0);
//Delay
for (delay_counter=0;delay_counter<=lt_delay_intermovement;delay_counter++)
{
        delay_ms(1);
}
}
```

```
void    MOTOR_MOVEMENT(int    motor_left_dir,int    motor_right_dir,long    dutyratio_left,long
dutyratio_right)

{

        //motor_left_dir: 0=Stop, 1=Forward, 2=Reverse

        //motor_right_dir: 0=Stop, 1=Forward, 2=Reverse


  /*

    Movement pattern            motor_left_dir            motor_right_dir

    -------------------------------------------------------------------------------


Stop                            0                                     0

Forward                         1                                     1

 Reverse                        2                                     2

 Rotate right                   1                                     2

 Rotate left                    2                                     1

Turn forward right              1                                     0

Turn reverse right              2                                     0

Turn forward left               0                                     1

 Turn reverse left              0                                     2

  */


  //Stop

        if (motor_left_dir==0 && motor_right_dir==0)

  {

          //Left Dir: Stop

        output_bit(PIN_D3,0);

   output_bit(PIN_D2,0);


          //Right Dir: Stop

   output_bit(PIN_D1,0);

          output_bit(PIN_D0,0);


                  //Left PWM: Off
```

```
setup_ccp2(CCP_OFF);

//Right PWM: Off
setup_ccp1(CCP_OFF);
}
else
//Forward
if (motor_left_dir==1 && motor_right_dir==1)
{
//Left Dir: Forward
output_bit(PIN_D3,0);
output_bit(PIN_D2,1);

//Right Dir: Forward
output_bit(PIN_D1,0);
output_bit(PIN_D0,1);

//Left PWM: On
setup_ccp2(CCP_PWM);
set_pwm2_duty(dutyratio_left);

//Right PWM: On
setup_ccp1(CCP_PWM);
set_pwm1_duty(dutyratio_right);
}
else
//Reverse
if (motor_left_dir==2 && motor_right_dir==2)
{
//Left Dir: Reverse
output_bit(PIN_D3,1);
output_bit(PIN_D2,0);

//Right Dir: Reverse
```

```
        output_bit(PIN_D1,1);
    output_bit(PIN_D0,0);

                //Left PWM: On
                setup_ccp2(CCP_PWM);
                set_pwm2_duty(dutyratio_left);

                //Right PWM: On
                setup_ccp1(CCP_PWM);
                set_pwm1_duty(dutyratio_right);
        }
else
        //Rotate right
        if (motor_left_dir==1 && motor_right_dir==2)
        {
        //Left Dir: Forward
    output_bit(PIN_D3,0);
        output_bit(PIN_D2,1);

        //Right Dir: Reverse
        output_bit(PIN_D1,1);
       output_bit(PIN_D0,0);

        //Left PWM: On
                setup_ccp2(CCP_PWM);
                set_pwm2_duty(dutyratio_left);

        //Right PWM: On
                setup_ccp1(CCP_PWM);
                set_pwm1_duty(dutyratio_right);
}
        else
//Rotate left
        if (motor_left_dir==2 && motor_right_dir==1)
```

```
{
        //Left Dir: Reverse
  output_bit(PIN_D3,1);
          output_bit(PIN_D2,0);

          //Right Dir: Forward
        output_bit(PIN_D1,0);
        output_bit(PIN_D0,1);

        //Left PWM: On
                  setup_ccp2(CCP_PWM);
                  set_pwm2_duty(dutyratio_left);

        //Right PWM: On
                  setup_ccp1(CCP_PWM);
                  set_pwm1_duty(dutyratio_right);
        }
        else
        //Turn forward right
        if (motor_left_dir==1 && motor_right_dir==0)
        {
        //Left Dir: Forward
  output_bit(PIN_D3,0);
          output_bit(PIN_D2,1);

          //Right Dir: Stop
          output_bit(PIN_D1,0);
        output_bit(PIN_D0,0);

        //Left PWM: On
                  setup_ccp2(CCP_PWM);
                  set_pwm2_duty(dutyratio_left);

        //Right PWM: Off
```

```
                    setup_ccp1(CCP_OFF);
}
        else
//Turn reverse right
        if (motor_left_dir==2 && motor_right_dir==0)
        {
        //Left Dir: Reverse
  output_bit(PIN_D3,1);
          output_bit(PIN_D2,0);

          //Right Dir: Stop
          output_bit(PIN_D1,0);
        output_bit(PIN_D0,0);

        //Left PWM: On
                    setup_ccp2(CCP_PWM);
                    set_pwm2_duty(dutyratio_left);

        //Right PWM: Off
                    setup_ccp1(CCP_OFF);
}
else
        //Turn forward left
        if (motor_left_dir==0 && motor_right_dir==1)
        {
        //Left Dir: Stop
  output_bit(PIN_D3,0);
          output_bit(PIN_D2,0);

          //Right Dir: Forward
          output_bit(PIN_D1,0);
        output_bit(PIN_D0,1);

        //Left PWM: Off
```

```
            setup_ccp2(CCP_OFF);


    //Right PWM: On
            setup_ccp1(CCP_PWM);
            set_pwm1_duty(dutyratio_right);
}
    else
//Turn reverse left
    if (motor_left_dir==0 && motor_right_dir==2)
{
    //Left Dir: Stop
  output_bit(PIN_D3,0);
      output_bit(PIN_D2,0);


      //Right Dir: Reverse
      output_bit(PIN_D1,1);
    output_bit(PIN_D0,0);
```

For the servo motor controller, PIC16F84A has been used separately from the main controller of PIC18F4620. Thus, the servo motor source is written as below.

```
#include <16F84A.H>
#use delay(clock=4000000)
#fuses  XT,NOPROTECT,NOWDT

#define delay_m90       2250    // - (-90)
#define delay_m45       1800    // \ (-45)
#define delay_0         1350 // | (0)
#define delay_p45       810     // / (+45)
#define delay_p90       450  //  - (+90)

#define delay_low 20

void main()
{
```

```
while(true)
{
  // release
  if (input(PIN_A2)==0 && input(PIN_A1)==0 && input(PIN_A0)==0)
  {
      output_low(PIN_A3);
  }
            else
  // m90
  if (input(PIN_A2)==0 && input(PIN_A1)==0 && input(PIN_A0)==1)
  {
      output_high(PIN_A3);
      delay_us(delay_m90);

      output_low(PIN_A3);
      delay_ms(delay_low);
  }
            else
  // m45
  if (input(PIN_A2)==0 && input(PIN_A1)==1 && input(PIN_A0)==0)
  {
      output_high(PIN_A3);
      delay_us(delay_m45);

            output_low(PIN_A3);
      delay_ms(delay_low);
  }
            else
  // 0
  if (input(PIN_A2)==0 && input(PIN_A1)==1 && input(PIN_A0)==1)
  {
      output_high(PIN_A3);
      delay_us(delay_0);

      output_low(PIN_A3);
      delay_ms(delay_low);
  }
            else
  // p45
  if (input(PIN_A2)==1 && input(PIN_A1)==0 && input(PIN_A0)==0)
```

```
    {
        output_high(PIN_A3);
        delay_us(delay_p45);

        output_low(PIN_A3);
        delay_ms(delay_low);
    }
            else
    // p90
    if (input(PIN_A2)==1 && input(PIN_A1)==0 && input(PIN_A0)==1)
    {
        output_high(PIN_A3);
        delay_us(delay_p90);

        output_low(PIN_A3);
        delay_ms(delay_low);
    }
  }
}
```

PIC18F2525/2620/4525/4620
Data Sheet

28/40/44-Pin
Enhanced Flash Microcontrollers
with 10-Bit A/D and nanoWatt Technology

MICROCHIP PIC18F2525/2620/4525/4620

28/40/44-Pin Enhanced Flash Microcontrollers with 10-Bit A/D and nanoWatt Technology

Power Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 2.5 μA typical
- Sleep mode current down to 100 nA typical
- Timer1 Oscillator: 1.8 μA, 32 kHz, 2V
- Watchdog Timer: 1.4 μA, 2V typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes, up to 40 MHz
- 4x Phase Lock Loop (PLL) – available for crystal and internal oscillators)
- Two External RC modes, up to 4 MHz
- Two External Clock modes, up to 40 MHz
- Internal oscillator block:
  - 8 user selectable frequencies, from 31 kHz to 8 MHz
  - Provides a complete range of clock speeds from 31 kHz to 32 MHz when used with PLL
  - User tunable to compensate for frequency drift
- Secondary oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor
  - Allows for safe shutdown if peripheral clock stops

Peripheral Highlights:

- High-current sink/source 25 mA/25 mA
- Three programmable external interrupts
- Four input change interrupts
- Up to 2 Capture/Compare/PWM (CCP) modules, one with Auto-Shutdown (28-pin devices)
- Enhanced Capture/Compare/PWM (ECCP) module (40/44-pin devices only):
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-Shutdown and Auto-Restart

Peripheral Highlights (Continued):

- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI™ (all 4 modes) and I²C™ Master and Slave modes
- Enhanced Addressable USART module:
  - Supports RS-485, RS-232 and LIN 1.2
  - RS-232 operation using internal oscillator block (no external crystal required)
  - Auto-Wake-up on Start bit
  - Auto-Baud Detect
- 10-bit, up to 13-channel Analog-to-Digital Converter module (A/D):
  - Auto-acquisition capability
  - Conversion available during Sleep
- Dual analog comparators with input multiplexing
- Programmable 16-level High/Low-Voltage Detection (HLVD) module:
  - Supports interrupt on High/Low-Voltage Detection

Special Microcontroller Features:

- C compiler optimized architecture:
  - Optional extended instruction set designed to optimize re-entrant code
- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Flash/Data EEPROM Retention: 100 years typical
- Self-programmable under software control
- Priority levels for interrupts
- 8 x 8 Single Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- Single-supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Wide operating voltage range: 2.0V to 5.5V
- Programmable Brown-out Reset (BOR) with software enable option

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ ECCP (PWM)	MSSP		EUSART	Comp.	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI™	Master I²C™			
PIC18F2525	48K	24576	3986	1024	25	10	2/0	Y	Y	1	2	1/3
PIC18F2620	64K	32768	3986	1024	25	10	2/0	Y	Y	1	2	1/3
PIC18F4525	48K	24576	3986	1024	36	13	1/1	Y	Y	1	2	1/3
PIC18F4620	64K	32768	3986	1024	36	13	1/1	Y	Y	1	2	1/3

! Other Special Features

- **lemory Endurance:** The Enhanced Flash cells ɔr both program memory and data EEPROM are ated to last for many thousands of erase/write ycles – up to 100,000 for program memory and ,000,000 for EEPROM. Data retention without efresh is conservatively estimated to be greater ıan 40 years.
- **ielf-programmability:** These devices can write ɔ their own program memory spaces under inter- al software control. By using a bootloader routine ɔcated in the protected Boot Block at the top of rogram memory, it becomes possible to create n application that can update itself in the field.
- **:xtended Instruction Set:** The PIC18F2525/ :620/4525/4620 family introduces an optional xtension to the PIC18 instruction set, which adds new instructions and an Indexed Addressing ıode. This extension, enabled as a device con- guration option, has been specifically designed ɔ optimize re-entrant application code originally leveloped in high-level languages, such as C.
- **:nhanced CCP module:** In PWM mode, this ıodule provides 1, 2 or 4 modulated outputs for ontrolling half-bridge and full-bridge drivers.)ther features include auto-shutdown, for lisabling PWM outputs on interrupt or other select onditions and auto-restart, to reactivate outputs nce the condition has cleared.
- **:nhanced Addressable USART:** This serial ommunication module is capable of standard ℝS-232 operation and provides support for the LIN ıus protocol. Other enhancements include ıutomatic baud rate detection and a 16-bit Baud ₹ate Generator for improved resolution. When the nicrocontroller is using the internal oscillator ılock, the USART provides stable operation for ıpplications that talk to the outside world without ısing an external crystal (or its accompanying ɔower requirement).
- **l0-bit A/D Converter:** This module incorporates ɔrogrammable acquisition time, allowing for a :hannel to be selected and a conversion to be nitiated without waiting for a sampling period and hus, reduce code overhead.
- **:xtended Watchdog Timer (WDT):** This :nhanced version incorporates a 16-bit prescaler, ıllowing an extended time-out range that is stable ıcross operating voltage and temperature. See **Section 26.0 "Electrical Characteristics"** for ime-out periods.

1.3 Details on Individual Family Members

Devices in the PIC18F2525/2620/4525/4620 family are available in 28-pin and 40/44-pin packages. Block diagrams for the two groups are shown in Figure 1-1 and Figure 1-2.

The devices are differentiated from each other in five ways:

1. Flash program memory (48 Kbytes for PIC18FX525 devices, 64 Kbytes for PIC18FX620).
2. A/D channels (10 for 28-pin devices, 13 for 40/44-pin devices).
3. I/O ports (3 bidirectional ports on 28-pin devices, 5 bidirectional ports on 40/44-pin devices).
4. CCP and Enhanced CCP implementation (28-pin devices have 2 standard CCP modules, 40/44-pin devices have one standard CCP module and one ECCP module).
5. Parallel Slave Port (present only on 40/44-pin devices).

All other features for devices in this family are identical. These are summarized in Table 1-1.

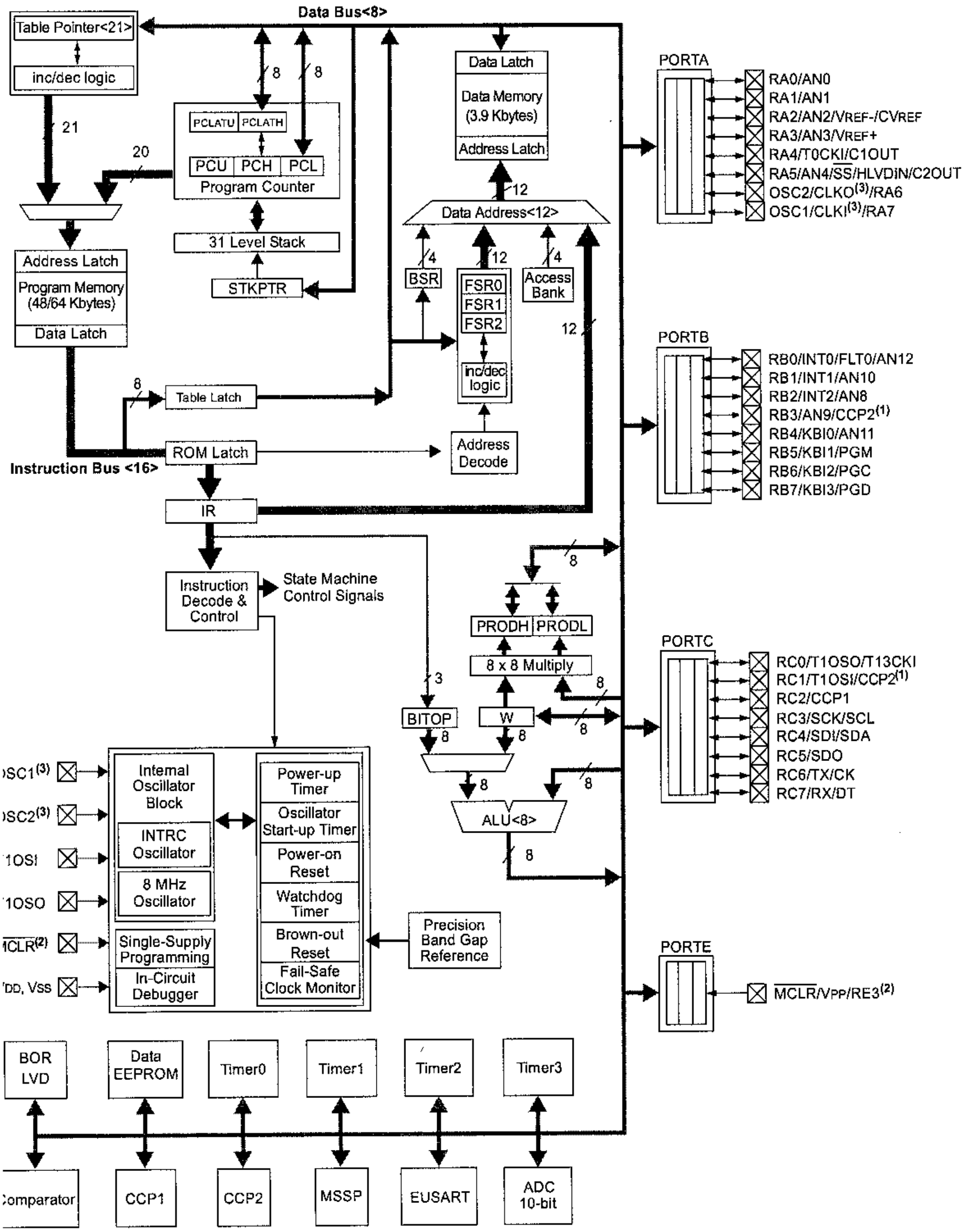
The pinouts for all devices are listed in Table 1-2 and Table 1-3.

Like all Microchip PIC18 devices, members of the PIC18F2525/2620/4525/4620 family are available as both standard and low-voltage devices. Standard devices with Enhanced Flash memory, designated with an "F" in the part number (such as PIC18**F**2620), accommodate an operating VDD range of 4.2V to 5.5V. Low-voltage parts, designated by "LF" (such as PIC18**LF**2620), function over an extended VDD range of 2.0V to 5.5V.

TABLE 1-1: DEVICE FEATURES

Features	PIC18F2525	PIC18F2620	PIC18F4525	PIC18F4620
Operating Frequency	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz
Program Memory (Bytes)	49152	65536	49152	65536
Program Memory (Instructions)	24576	32768	24576	32768
Data Memory (Bytes)	3968	3968	3968	3968
Data EEPROM Memory (Bytes)	1024	1024	1024	1024
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/Compare/ PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Parallel Communications (PSP)	No	No	Yes	Yes
10-bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{\text{MCLR}}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{\text{MCLR}}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{\text{MCLR}}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{\text{MCLR}}$ (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-pin SPDIP 28-pin SOIC	28-pin SPDIP 28-pin SOIC	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP

;URE 1-1: PIC18F2525/2620 (28-PIN) BLOCK DIAGRAM



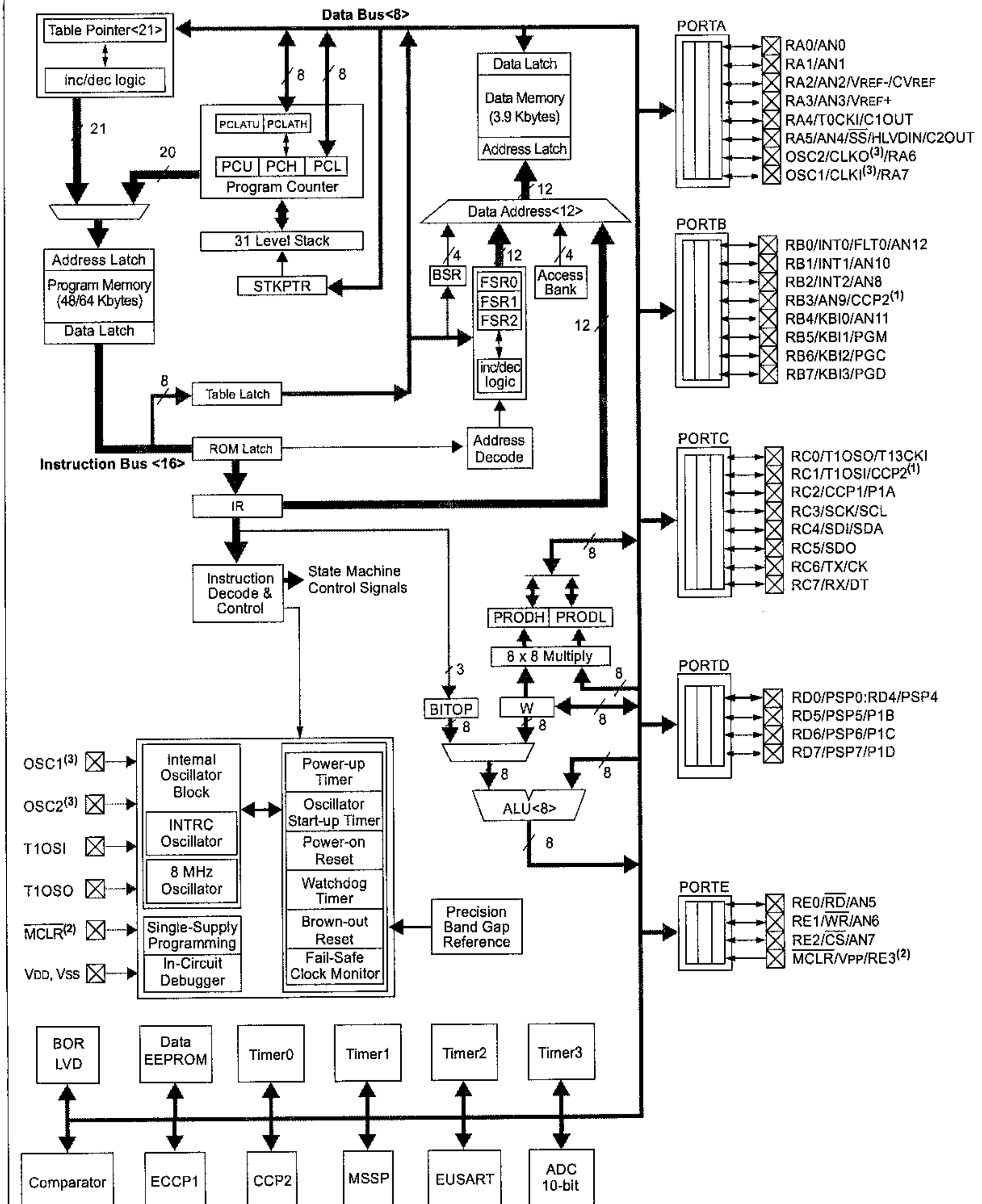
lote 1: CCP2 is multiplexed with RC1 when configuration bit CCP2MX is set, or RB3 when CCP2MX is not set.

2: RE3 is only available when $\overline{MCLR}$ functionality is disabled.

3: 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.

 Preliminary

FIGURE 1-2: PIC18F4525/4620 (40/44-PIN) BLOCK DIAGRAM



Note 1: CCP2 is multiplexed with RC1 when configuration bit CCP2MX is set, or RB3 when CCP2MX is not set.

2: RE3 is only available when $\overline{\text{MCLR}}$ functionality is disabled.

3: 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.

BLE 1-2: PIC18F2525/2620 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	SPDIP, SOIC	QFN			
:LR/VPP/RE3	1	26			Master Clear (input) or programming voltage (input).
$\overline{\text{MCLR}}$			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.
;C1/CLKI/RA7	9	6			Oscillator crystal or external clock input.
OSC1			I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; CMOS otherwise.
CLKI			I	CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
RA7			I/O	TTL	General purpose I/O pin.
;C2/CLKO/RA6	10	7			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.

gend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels I = Input
O = Output P = Power

te 1: Default assignment for CCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

TABLE 1-2: PIC18F2525/2620 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number SPDIP, SOIC	Pin Number QFN	Pin Type	Buffer Type	Description
					PORTA is a bidirectional I/O port.
RA0/AN0	2	27			
RA0			I/O	TTL	Digital I/O.
AN0			I	Analog	Analog input 0.
RA1/AN1	3	28			
RA1			I/O	TTL	Digital I/O.
AN1			I	Analog	Analog input 1.
RA2/AN2/VREF-/CVREF	4	1			
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	Comparator reference voltage output.
RA3/AN3/VREF+	5	2			
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	6	3			
RA4			I/O	ST	Digital I/O.
T0CKI			I	ST	Timer0 external clock input.
C1OUT			O	—	Comparator 1 output.
RA5/AN4/$\overline{SS}$/HLVDIN/C2OUT	7	4			
RA5			I/O	TTL	Digital I/O.
AN4			I	Analog	Analog input 4.
$\overline{SS}$			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.
RA7					See the OSC1/CLKI/RA7 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: Default assignment for CCP2 when configuration bit CCP2MX is set.

2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

3LE 1-2: PIC18F2525/2620 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	SPDIP, SOIC	QFN			
					PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
0/INT0/FLT0/AN12	21	18			
RB0			I/O	TTL	Digital I/O.
INT0			I	ST	External interrupt 0.
FLT0			I	ST	PWM Fault input for CCP1.
AN12			I	Analog	Analog input 12.
1/INT1/AN10	22	19			
RB1			I/O	TTL	Digital I/O.
INT1			I	ST	External interrupt 1.
AN10			I	Analog	Analog input 10.
2/INT2/AN8	23	20			
RB2			I/O	TTL	Digital I/O.
INT2			I	ST	External interrupt 2.
AN8			I	Analog	Analog input 8.
3/AN9/CCP2	24	21			
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.
4/KBI0/AN11	25	22			
RB4			I/O	TTL	Digital I/O.
KBI0			I	TTL	Interrupt-on-change pin.
AN11			I	Analog	Analog input 11.
5/KBI1/PGM	26	23			
RB5			I/O	TTL	Digital I/O.
KBI1			I	TTL	Interrupt-on-change pin.
PGM			I/O	ST	Low-Voltage ICSP™ Programming enable pin.
6/KBI2/PGC	27	24			
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.
7/KBI3/PGD	28	25			
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.

gend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels I = Input
O = Output P = Power

te 1: Default assignment for CCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

TABLE 1-2: PIC18F2525/2620 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	SPDIP, SOIC	QFN			
					PORTC is a bidirectional I/O port.
RC0/T1OSO/T13CKI	11	8			
RC0			I/O	ST	Digital I/O.
T1OSO			O	—	Timer1 oscillator output.
T13CKI			I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2	12	9			
RC1			I/O	ST	Digital I/O.
T1OSI			I	Analog	Timer1 oscillator input.
CCP2(2)			I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
RC2/CCP1	13	10			
RC2			I/O	ST	Digital I/O.
CCP1			I/O	ST	Capture 1 input/Compare 1 output/PWM 1 output.
RC3/SCK/SCL	14	11			
RC3			I/O	ST	Digital I/O.
SCK			I/O	ST	Synchronous serial clock input/output for SPI™ mode.
SCL			I/O	ST	Synchronous serial clock input/output for I^{2}C™ mode.
RC4/SDI/SDA	15	12			
RC4			I/O	ST	Digital I/O.
SDI			I	ST	SPI data in.
SDA			I/O	ST	I^{2}C data I/O.
RC5/SDO	16	13			
RC5			I/O	ST	Digital I/O.
SDO			O	—	SPI data out.
RC6/TX/CK	17	14			
RC6			I/O	ST	Digital I/O.
TX			O	—	EUSART asynchronous transmit.
CK			I/O	ST	EUSART synchronous clock (see related RX/DT).
RC7/RX/DT	18	15			
RC7			I/O	ST	Digital I/O.
RX			I	ST	EUSART asynchronous receive.
DT			I/O	ST	EUSART synchronous data (see related TX/CK).
RE3	—	—	—	—	See $\overline{MCLR}$/VPP/RE3 pin.
VSS	8, 19	5, 16	P	—	Ground reference for logic and I/O pins.
VDD	20	17	P	—	Positive supply for logic and I/O pins.

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: Default assignment for CCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

BLE 1-3: PIC18F4525/4620 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
:LR/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.
;C1/CLKI/RA7	13	32	30			Oscillator crystal or external clock input.
OSC1				I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; analog otherwise.
CLKI				I	CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
RA7				I/O	TTL	General purpose I/O pin.
;C2/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.

gend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels I = Input
O = Output P = Power

te 1: Default assignment for CCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

TABLE 1-3: PIC18F4525/4620 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number PDIP	QFN	TQFP	Pin Type	Buffer Type	Description
						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	Comparator reference voltage 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	6	23	23			
RA4				I/O	ST	Digital I/O.
T0CKI				I	ST	Timer0 external clock input.
C1OUT				O	—	Comparator 1 output.
RA5/AN4/$\overline{SS}$/HLVDIN/C2OUT	7	24	24			
RA5				I/O	TTL	Digital I/O.
AN4				I	Analog	Analog input 4.
$\overline{SS}$				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.
RA7						See the OSC1/CLKI/RA7 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: Default assignment for CCP2 when configuration bit CCP2MX is set.

2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

3LE 1-3: PIC18F4525/4620 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number PDIP	QFN	TQFP	Pin Type	Buffer Type	Description
						PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
0/INT0/FLT0/AN12	33	9	8			
RB0				I/O	TTL	Digital I/O.
INT0				I	ST	External interrupt 0.
FLT0				I	ST	PWM Fault input for Enhanced CCP1.
AN12				I	Analog	Analog input 12.
1/INT1/AN10	34	10	9			
RB1				I/O	TTL	Digital I/O.
INT1				I	ST	External interrupt 1.
AN10				I	Analog	Analog input 10.
2/INT2/AN8	35	11	10			
RB2				I/O	TTL	Digital I/O.
INT2				I	ST	External interrupt 2.
AN8				I	Analog	Analog input 8.
3/AN9/CCP2	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.
4/KBI0/AN11	37	14	14			
RB4				I/O	TTL	Digital I/O.
KBI0				I	TTL	Interrupt-on-change pin.
AN11				I	Analog	Analog input 11.
5/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.
6/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.
7/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.

jend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels I = Input
O = Output P = Power

te 1: Default assignment for CCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

TABLE 1-3: PIC18F4525/4620 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number PDIP	QFN	TQFP	Pin Type	Buffer Type	Description
						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	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.
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	—	Enhanced CCP1 output.
RC3/SCK/SCL	18	37	37			
RC3				I/O	ST	Digital I/O.
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.
RC4/SDI/SDA	23	42	42			
RC4				I/O	ST	Digital I/O.
SDI				I	ST	SPI data in.
SDA				I/O	ST	I^2C data I/O.
RC5/SDO	24	43	43			
RC5				I/O	ST	Digital I/O.
SDO				O	—	SPI data out.
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 related RX/DT).
RC7/RX/DT	26	1	1			
RC7				I/O	ST	Digital I/O.
RX				I	ST	EUSART asynchronous receive.
DT				I/O	ST	EUSART synchronous data (see related TX/CK).

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: Default assignment for CCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

3LE 1-3: PIC18F4525/4620 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 Parallel Slave Port (PSP) for interfacing to a microprocessor port. These pins have TTL input buffers when the PSP module is enabled.
0/PSP0	19	38	38			
RD0				I/O	ST	Digital I/O.
PSP0				I/O	TTL	Parallel Slave Port data.
1/PSP1	20	39	39			
RD1				I/O	ST	Digital I/O.
PSP1				I/O	TTL	Parallel Slave Port data.
2/PSP2	21	40	40			
RD2				I/O	ST	Digital I/O.
PSP2				I/O	TTL	Parallel Slave Port data.
3/PSP3	22	41	41			
RD3				I/O	ST	Digital I/O.
PSP3				I/O	TTL	Parallel Slave Port data.
4/PSP4	27	2	2			
RD4				I/O	ST	Digital I/O.
PSP4				I/O	TTL	Parallel Slave Port data.
5/PSP5/P1B	28	3	3			
RD5				I/O	ST	Digital I/O.
PSP5				I/O	TTL	Parallel Slave Port data.
P1B				O	—	Enhanced CCP1 output.
6/PSP6/P1C	29	4	4			
RD6				I/O	ST	Digital I/O.
PSP6				I/O	TTL	Parallel Slave Port data.
P1C				O	—	Enhanced CCP1 output.
7/PSP7/P1D	30	5	5			
RD7				I/O	ST	Digital I/O.
PSP7				I/O	TTL	Parallel Slave Port data.
P1D				O	—	Enhanced CCP1 output.

jend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels I = Input
O = Output P = Power

te 1: Default assignment for CCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

TABLE 1-3: PIC18F4525/4620 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/$\overline{RD}$/AN5	8	25	25			
RE0				I/O	ST	Digital I/O.
$\overline{RD}$				I	TTL	Read control for Parallel Slave Port (see also $\overline{WR}$ and $\overline{CS}$ pins).
AN5				I	Analog	Analog input 5.
RE1/$\overline{WR}$/AN6	9	26	26			
RE1				I/O	ST	Digital I/O.
$\overline{WR}$				I	TTL	Write control for Parallel Slave Port (see $\overline{CS}$ and $\overline{RD}$ pins).
AN6				I	Analog	Analog input 6.
RE2/$\overline{CS}$/AN7	10	27	27			
RE2				I/O	ST	Digital I/O.
$\overline{CS}$				I	TTL	Chip Select control for Parallel Slave Port (see related $\overline{RD}$ and $\overline{WR}$).
AN7				I	Analog	Analog input 7.
RE3	—	—	—	—	—	See $\overline{MCLR}$/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.
NC	—	13	12, 13, 33, 34	—	—	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: Default assignment for CCP2 when configuration bit CCP2MX is set.

2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

PARALLAX INC

599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003

General: info@parallaxinc.com
Technical: support@parallax.com
Web Site: www.parallax.com
Educational: http://www.parallax.com/html_pages/edu/

Standard Servo (#900-00005)

General Information

The Parallax standard servo is ideal for robotics and basic movement projects. These servos will allow a movement range of 0 to 180 degrees. The Parallax servo output gear shaft is a standard Futaba configuration. The servo is manufactured by Futaba specifically for Parallax.

Technical Specifications

> Power 6vdc max

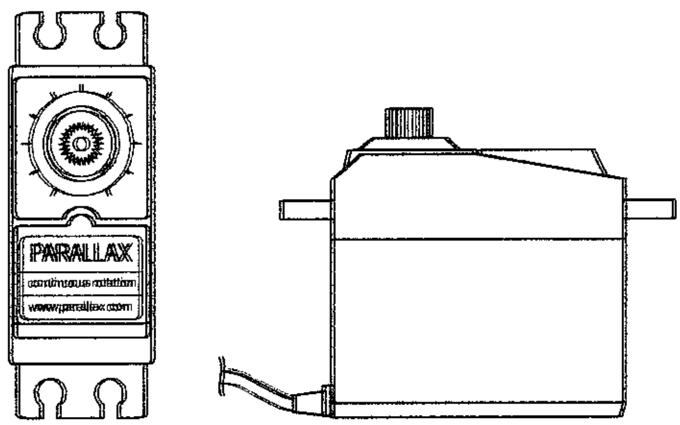
> Speed 0 deg to 180 deg in 1.5 seconds on average

> Weight 45.0 grams/1.59oz

> Torque 3.40 kg-cm/47oz-in

> Size mm (L x W x H) 40.5x20.0x38.0

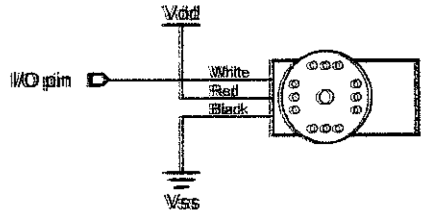
> Size in (L x W x H) 1.60x.79x1.50



Motor Control from a BASIC Stamp

Parallax (www.parallax.com) publishes many circuits and examples to control servos. Most of these examples are available for download from our web site. On www.parallaxinc.com type in "servo" and you'll find example codes below.

Wiring setup



The servo is controlled by pulsing of it's signal line. If you are using an Basic Stamp this is done with the pulsout command. Below is stamp code that will help you with basic control of a servo. The codes below may not move the servos from on extreme to another but is will give you a general demonstration on function.

Stamp1 code

```
SYMBOL Servo_pin  = 0      'I/O pin that is connected to servo
SYMBOL Temp = W0           'Work space for FOR NEXT
start:
   FOR temp =  70 TO 250
  PULSOUT Servo_pin,temp
  PAUSE 50
   NEXT
  FOR temp =  250 TO 70
  PULSOUT Servo_pin,temp
  PAUSE 50
   NEXT
GOTO start
```

'Stamp 2 ,2e,2pe

```
Servo_pin  CON          0     'I/O pin that is connected to servo
Temp        VAR         Word         'Work space for FOR NEXT
start:
  FOR temp =  200 TO 1200
    PULSOUT Servo_pin,temp
    PAUSE 50
  NEXT
  FOR temp =  1200 TO 200
    PULSOUT Servo_pin,temp
    PAUSE 50
  NEXT
GOTO start
```

'Stamp 2sx,2p24/40

```
Servo_pin  CON          0     'I/O pin that is connected to servo
Temp        VAR         Word         'Work space for FOR NEXT
start:
   FOR temp =  500 TO 3000
  PULSOUT Servo_pin,temp
  PAUSE 20
   NEXT
  FOR temp =  3000 TO 500
  PULSOUT Servo_pin,temp
  PAUSE 20
   NEXT
GOTO start
```

National Semiconductor

May 2000

LM78XX
Series Voltage Regulators

General Description

'he LM78XX series of three terminal regulators is available vith several fixed output voltages making them useful in a vide range of applications. One of these is local on card egulation, eliminating the distribution problems associated vith single point regulation. The voltages available allow hese regulators to be used in logic systems, instrumenta- ion, HiFi, and other solid state electronic equipment. Al- hough designed primarily as fixed voltage regulators these levices can be used with external components to obtain ad- ıstable voltages and currents.

'he LM78XX series is available in an aluminum TO-3 pack- ıge which will allow over 1.0A load current if adequate heat inking is provided. Current limiting is included to limit the ıeak output current to a safe value. Safe area protection for he output transistor is provided to limit internal power dissi- ıation. If internal power dissipation becomes too high for the ıeat sinking provided, the thermal shutdown circuit takes ıver preventing the IC from overheating.

Considerable effort was expanded to make the LM78XX se- ies of regulators easy to use and minimize the number of ıxternal components. It is not necessary to bypass the out- put, although this does improve transient response. Input by- passing is needed only if the regulator is located far from the filter capacitor of the power supply.

For output voltage other than 5V, 12V and 15V the LM117 series provides an output voltage range from 1.2V to 57V.

Features

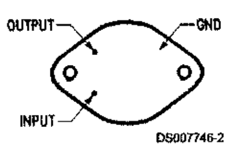
- Output current in excess of 1A
- Internal thermal overload protection
- No external components required
- Output transistor safe area protection
- Internal short circuit current limit
- Available in the aluminum TO-3 package

Voltage Range

LM7805C	5V
LM7812C	12V
LM7815C	15V

Connection Diagrams

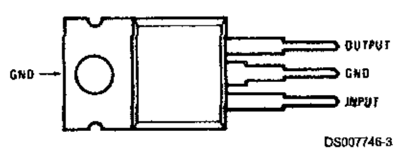
Metal Can Package
TO-3 (K)
Aluminum



DS007746-2

Bottom View
Order Number LM7805CK,
LM7812CK or LM7815CK
See NS Package Number KC02A

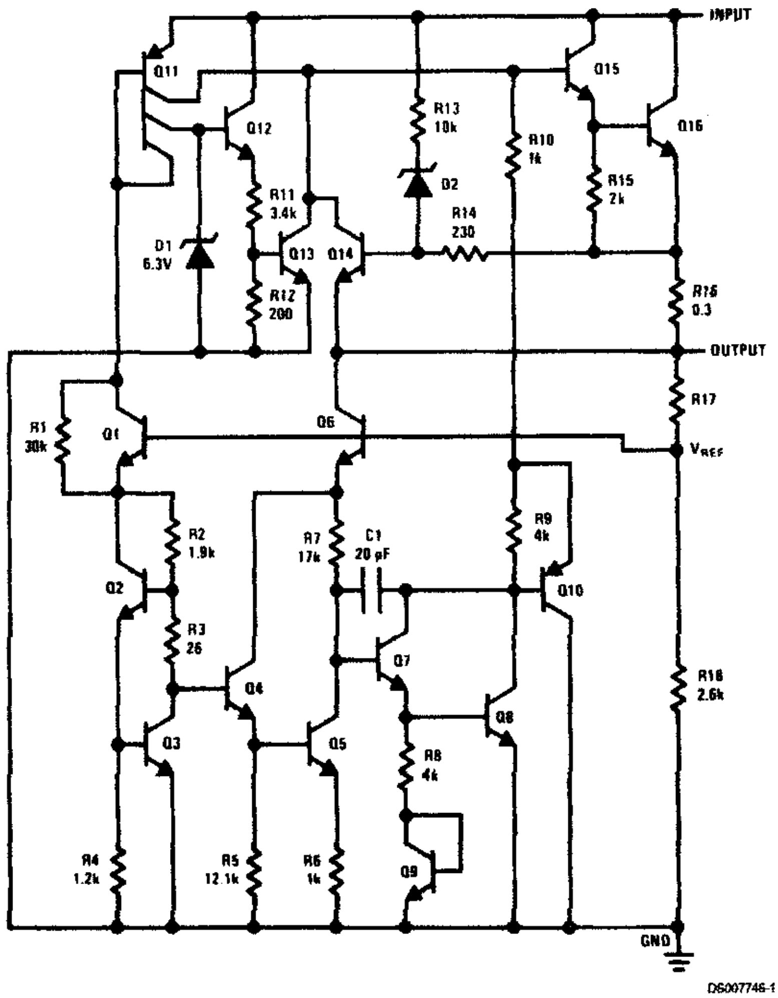
Plastic Package
TO-220 (T)



DS007746-3

Top View
Order Number LM7805CT,
LM7812CT or LM7815CT
See NS Package Number T03B

Schematic



INPUT
Q11
Q12
Q15
Q16
R13
10k
R10
1k
R11
3.4k
D2
R15
2k
R14
230
D1
6.3V
Q13
Q14
R12
200
R16
0.3
OUTPUT
R17
R1
30k
Q1
Q6
V_{REF}
R2
1.9k
R7
17k
C1
20 pF
R9
4k
Q2
Q10
R3
26
Q7
Q4
Q8
Q3
Q5
R8
4k
R18
2.6k
R4
1.2k
R5
12.1k
R6
1k
Q9
GND
DS007746-1

olute Maximum Ratings (Note 3)

tary/Aerospace specified devices are required,
contact the National Semiconductor Sales Office/
utors for availability and specifications.

/oltage
= 5V, 12V and 15V) 35V
il Power Dissipation (Note 1) Internally Limited
ting Temperature Range (T_A) 0°C to +70°C

Maximum Junction Temperature
(K Package) 150°C
(T Package) 150°C
Storage Temperature Range *−65°C to +150°C*
Lead Temperature (Soldering, 10 sec.)
TO-3 Package K 300°C
TO-220 Package T 230°C

trical Characteristics LM78XXC (Note 2)

$_J \leq 125°C$ unless otherwise noted.

Output Voltage			5V			12V			15V			Units
Input Voltage (unless otherwise noted)			**10V**			**19V**			**23V**			
Parameter	**Conditions**		**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	
Output Voltage	Tj = 25°C, 5 mA ≤ I_O ≤ 1A		4.8	5	5.2	11.5	12	12.5	14.4	15	15.6	V
	P_D ≤ 15W, 5 mA ≤ I_O ≤ 1A		4.75		5.25	11.4		12.6	14.25		15.75	V
	$V_{MIN} \leq V_{IN} \leq V_{MAX}$			(7.5 ≤ V_{IN} ≤ 20)			(14.5 ≤ V_{IN} ≤ 27)			(17.5 ≤ V_{IN} ≤ 30)		V
Line Regulation	I_O = 500 mA	Tj = 25°C		3	50		4	120		4	150	mV
		ΔV_{IN}		(7 ≤ V_{IN} ≤ 25)			14.5 ≤ V_{IN} ≤ 30)			(17.5 ≤ V_{IN} ≤ 30)		V
		0°C ≤ Tj ≤ +125°C			50			120			150	mV
		ΔV_{IN}		(8 ≤ V_{IN} ≤ 20)			(15 ≤ V_{IN} ≤ 27)			(18.5 ≤ V_{IN} ≤ *30)*		V
	I_O ≤ 1A	Tj = 25°C			50			120			150	mV
		ΔV_{IN}		(7.5 ≤ V_{IN} ≤ 20)			(14.6 ≤ V_{IN} ≤ 27)			(17.7 ≤ V_{IN} ≤ 30)		V
		0°C ≤ Tj ≤ +125°C			25			60			75	mV
		ΔV_{IN}		(8 ≤ V_{IN} ≤ 12)			(16 ≤ V_{IN} ≤ 22)			(20 ≤ V_{IN} ≤ 26)		V
Load Regulation	Tj = 25°C	5 mA ≤ I_O ≤ 1.5A		10	50		12	120		12	150	mV
		250 mA ≤ I_O ≤ 750 mA			25			60			75	mV
	5 mA ≤ I_O ≤ 1A, 0°C ≤ Tj ≤ +125°C				50			120			150	mV
Quiescent Current	I_O ≤ 1A	Tj = 25°C			8			8			8	mA
		0°C ≤ Tj ≤ +125°C			8.5			8.5			8.5	mA
Quiescent Current Change	5 mA ≤ I_O ≤ 1A				0.5			0.5			0.5	mA
	Tj = 25°C, I_O ≤ 1A				1.0			1.0			1.0	mA
	$V_{MIN} \leq V_{IN} \leq V_{MAX}$			(7.5 ≤ V_{IN} ≤ 20)			(14.8 ≤ V_{IN}≤ 27)			(17.9 ≤ V_{IN} ≤ 30)		V
	I_O ≤ 500 mA, 0°C ≤ Tj ≤ +125°C				1.0			1.0			1.0	mA
	$V_{MIN} \leq V_{IN} \leq V_{MAX}$			(7 ≤ V_{IN} ≤ 25)			(14.5 ≤ V_{IN}≤ 30)			(17.5 ≤ V_{IN} ≤ 30)		V
Output Noise *Voltage*	T_A =25°C, 10 Hz ≤ f ≤ 100 kHz			40			75			90		μV
Ripple Rejection	f = 120 Hz	I_O ≤ 1A, Tj = 25°C or	62	80		55	72		54	70		dB
		I_O ≤ 500 mA 0°C ≤ Tj ≤ +125°C	62			55			54			dB
	$V_{MIN} \leq V_{IN} \leq V_{MAX}$			(8 ≤ V_{IN} ≤ 18)			(15 ≤ V_{IN} ≤ 25)			(18.5 ≤ V_{IN} ≤ 28.5)		V
Dropout Voltage	Tj = 25°C, I_{OUT} = 1A			2.0			2.0			2.0		V
Output Resistance	f = 1 kHz			8			18			19		mΩ

Electrical Characteristics LM78XXC (Note 2) (Continued)

0°C ≤ T_J ≤ 125°C unless otherwise noted.

	Output Voltage		5V			12V			15V			Units
	Input Voltage (unless otherwise noted)		10V			19V			23V			
Symbol	Parameter	Conditions	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
	Short-Circuit Current	Tj = 25°C		2.1			1.5			1.2		A
	Peak Output Current	Tj = 25°C		2.4			2.4			2.4		A
	Average TC of V_{OUT}	0°C ≤ Tj ≤ +125°C, I_O = 5 mA		0.6			1.5			1.8		mV/°C
V_{IN}	*Input Voltage* Required to Maintain Line Regulation	Tj = 25°C, I_O ≤ 1A		7.5		14.6			17.7			V

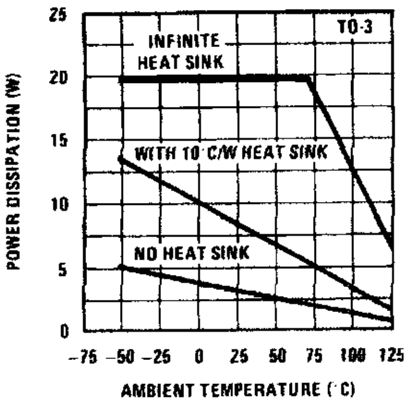
Note 1: *Thermal resistance of the TO-3 package (K, KC) is typically 4°C/W junction to case and 35°C/W case to ambient. Thermal resistance of the TO-220 package* (T) is typically 4°C/W junction to case and 50°C/W case to ambient.

Note 2: All characteristics are measured with capacitor across the input of 0.22 µF, and a capacitor across the output of 0.1µF. All characteristics except noise voltage and ripple rejection ratio *are measured using pulse techniques (t_w ≤ 10 ms, duty cycle ≤ 5%). Output voltage changes due to changes in internal temperature must* be taken into account separately.

Note 3: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. For guaranteed specifications and the test conditions, see *Electrical Characteristics.*

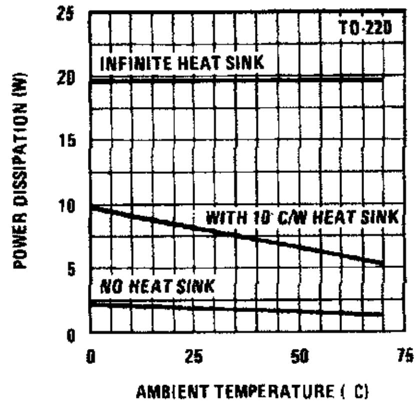
cal Performance Characteristics

um Average Power Dissipation



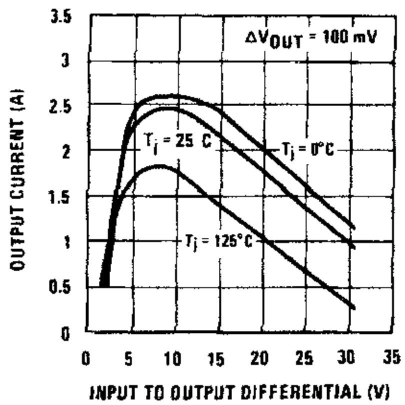
DS007746-5

Maximum Average Power Dissipation



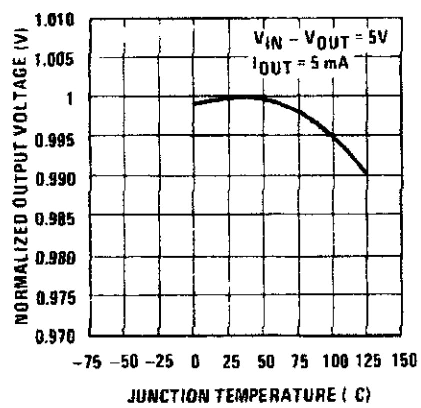
DS007746-6

utput Current



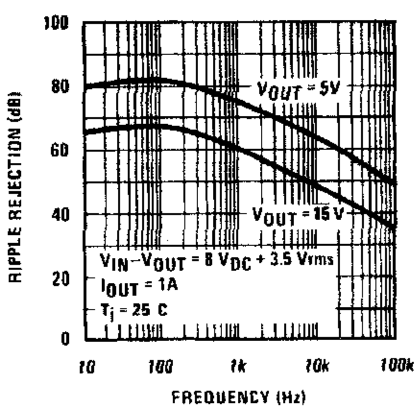
DS007746-7

Output Voltage (Normalized to 1V at T_J = 25°C)



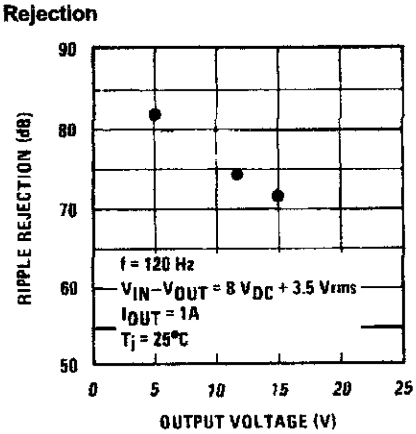
DS007746-8

Rejection



DS007746-9

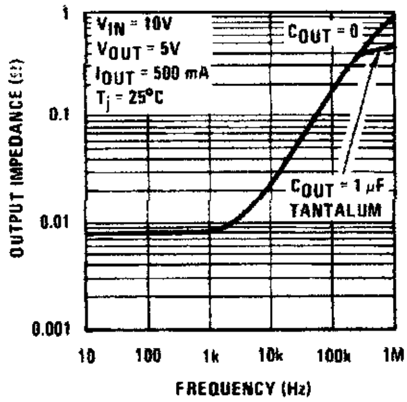
Ripple Rejection



DS007746-10

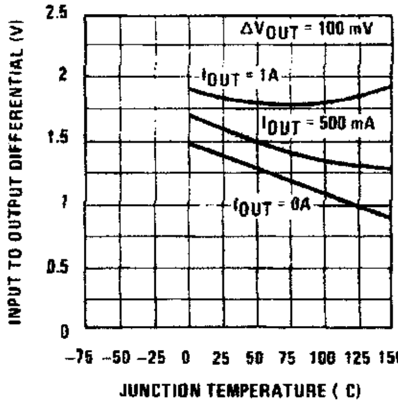
Typical Performance Characteristics (Continued)

Output Impedance



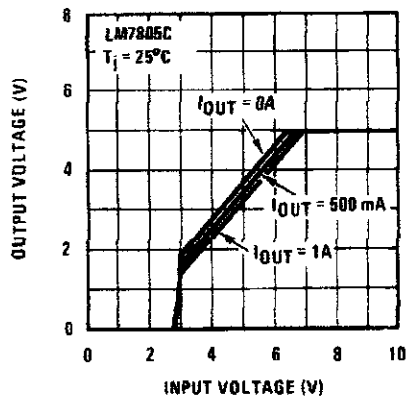
DS007746-11

Dropout Voltage



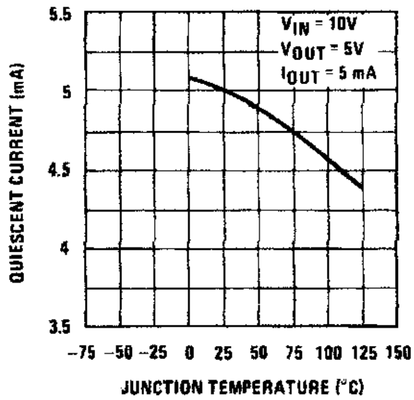
DS007746-12

Dropout Characteristics



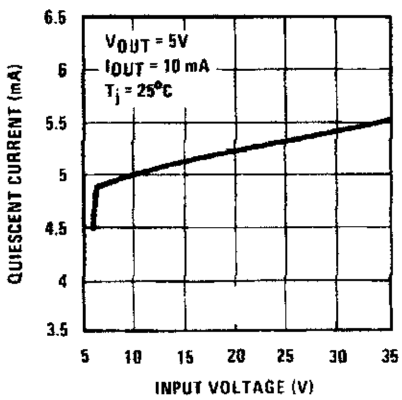
DS007746-13

Quiescent Current



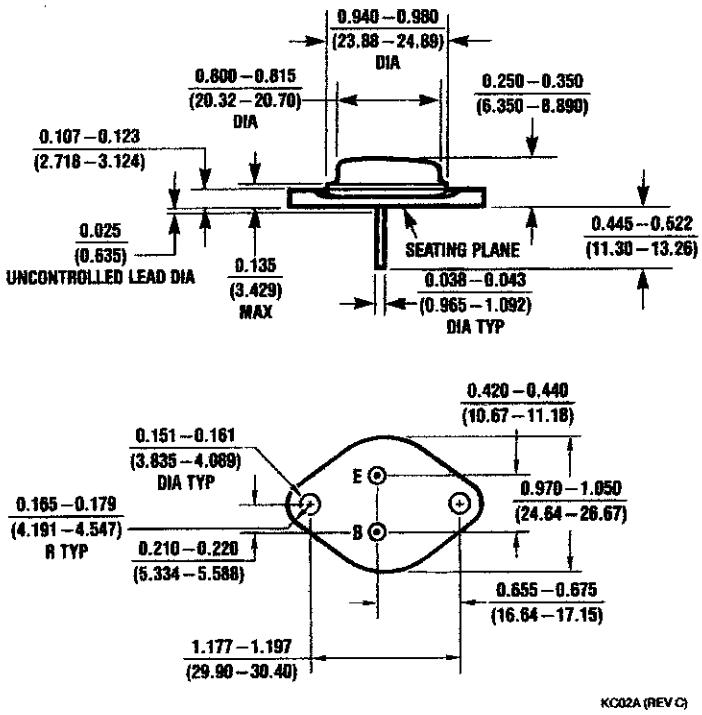
DS007746-14

Quiescent Current



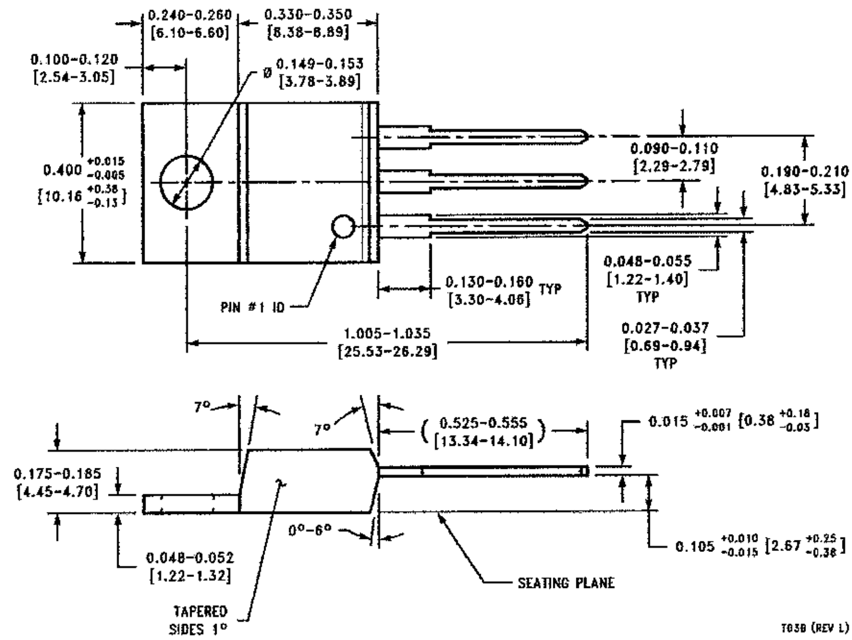
DS007746-15

sical Dimensions inches (millimeters) unless otherwise noted



Aluminum Metal Can Package (KC)
Order Number LM7805CK, LM7812CK or LM7815CK
NS Package Number KC02A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



TO-220 Package (T)
Order Number LM7805CT, LM7812CT or LM7815CT
NS Package Number T03B

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

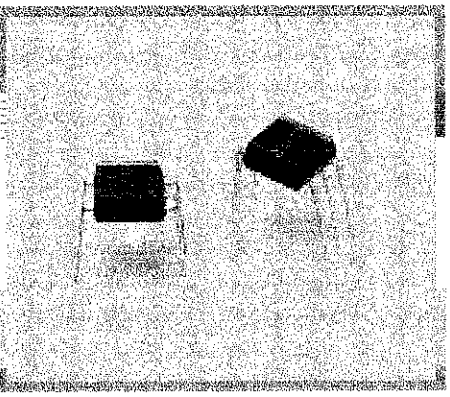
1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com
www.national.com

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 8790

National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: ap.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507



otoreflector
5587, P5588
²hoto IC output (digital) photoreflectors

and P5588 are photoreflectors combining a high power infrared LED and low voltage photo IC. The photo IC consists of a high sensitivity iode, amplifier, schmitt trigger circuit, and output phototransistor, etc. on a single chip.

Features

Miniature package
Low voltage operation
Photo IC, open collector output
P5587: "H" level output at light input
P5588: "L" level output at light input

Applications

- Paper detection in copiers and printers, etc.
- Tape end detection in VTRs, tape recorders, etc.

ıbsolute maximum ratings (Ta=25 °C)

Parameter		Symbol	Value	Unit
ıput .ED)	Forward current	I_F	50	mA
	Reverse voltage	V_R Max.	5	V
	Power dissipation	P	80	mW
utput oto IC)	Supply voltage	Vcc	-0.5 to +7	V
	Output voltage	Vo	-0.5 to +7	V
	Output current	Io	8	mA
	Power dissipation	P	80	mW
ɜrating temperature		Topr	-25 to +85	°C
rage temperature		Tstg	-30 to +85	°C
dering		-	260 °C, 3 s, refer to Dimensional outline	-

ːlectrical and optical characteristics (Ta=25 °C, Vcc=5 V, unless otherwise noted)

Parameter		Symbol	Condition	P5587 Min.	P5587 Typ.	P5587 Max.	P5588 Min.	P5588 Typ.	P5588 Max.	Unit
ıput .ED)	Forward voltage	V_F	I_F=20 mA	-	1.23	1.45	-	1.23	1.45	V
	Reverse current	I_R	V_R=5 V	-	-	10	-	-	10	μA
	Terminal capacitance	Ct	V=0 V, f=1 MHz	-	30	-	-	30	-	pF
utput oto IC)	Supply voltage	Vcc		2.2	-	7	2.2	-	7	V
	Low level output voltage	V_{OL}	I_{OL}=4 mA *1	-	0.1	0.4	-	0.1	0.4	V
	High level output current	I_{OH}	Vo=5 V *2	-	-	10	-	-	10	μA
	Current consumption	Icc		-	1.3	3.0	-	1.3	3.0	mA
ansfer acteristics	L→H Threshold input current	I_{FLH}	R_L=1.2 kΩ, d=3 mm Reflecting surface: white paper (reflectivity 90 % or more)	-	-	10	-	-	-	mA
	H→L Threshold input current	I_{FHL}		-	-	-	-	-	10	mA
	Hysteresis	-	*3	-	0.8	-	-	0.8	-	-
	L→H Propagation delay time	t_{PLH}	I_F=15 mA R_L=1.2 kΩ d=3 mm	-	-	20	-	-	30	μs
	H→L Propagation delay time	t_{PHL}		-	-	30	-	-	20	μs
	Rise time	tr		-	0.07	-	-	0.07	-	μs
	Fall time	tf		-	0.03	-	-	0.03	-	μs

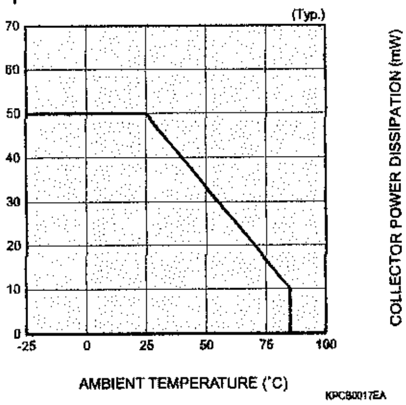
P5587: I_F=0 mA, P5588: I_F=15 mA
P5587: I_F=15 mA, P5588: I_F=0 mA
P5587: I_{FHL}/I_{FLH}, P5588: I_{FLH}/I_{FHL}

ːe) Connect a 0.01 μF capacitor or larger between Vcc and GND.

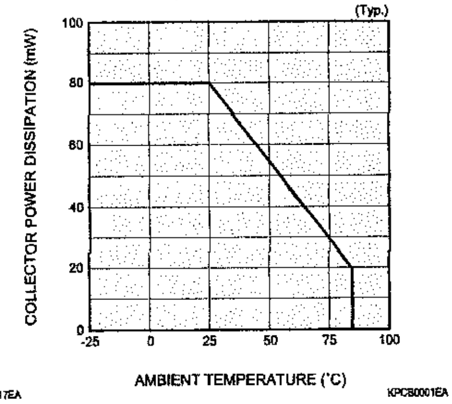
SOLID STATE DIVISION

HAMAMATSU

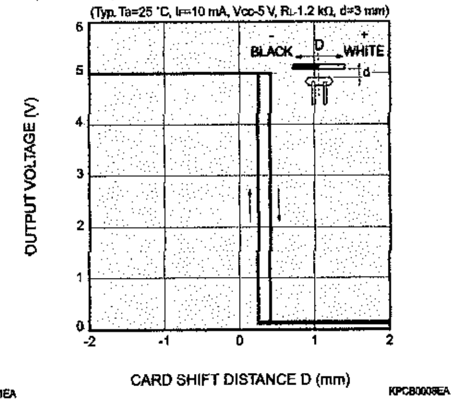
D forward current vs. ambient nperature



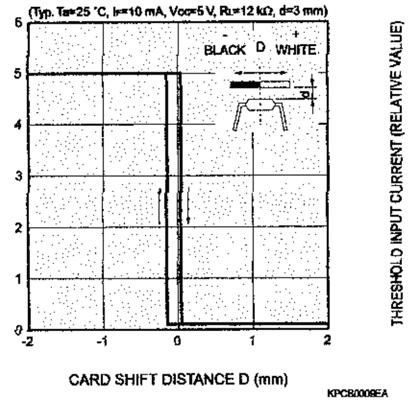
■ Photo IC power dissipation vs. ambient temperature



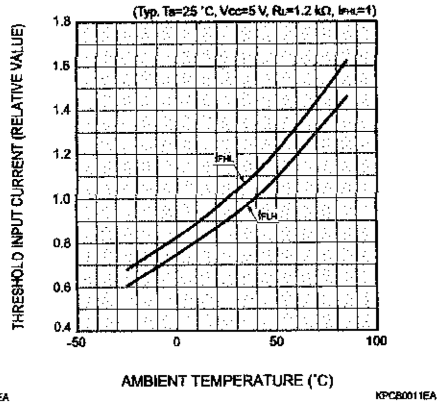
■ Position detection characteristic (P5587)



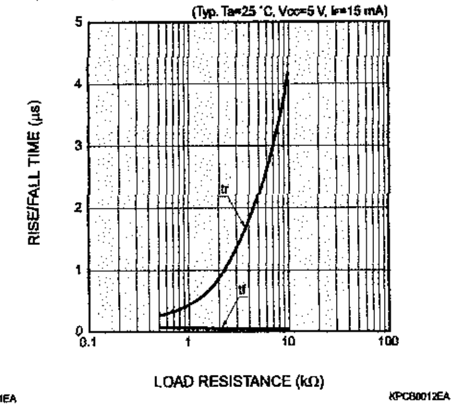
sition detection characteristic 5587)



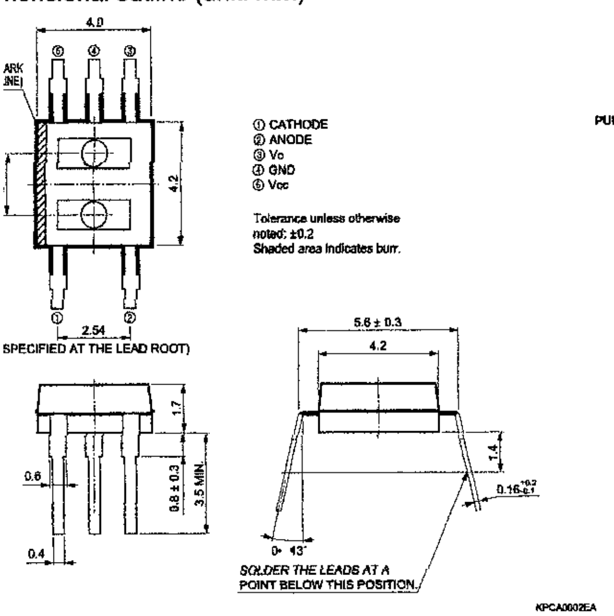
■ Threshold input current vs. ambient temperature (P5587)



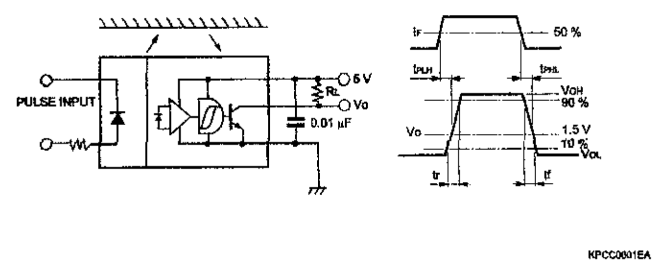
■ Rise/fall time vs. load resistance (P5587)



nensional outline (unit: mm)



■ Response time measurement circuit (P5587)



) PHOTONICS K.K., Solid State Division
-cho, Hamamatsu City, 435-8558 Japan, Telephone: (81) 053-434-3311, Fax: (81) 053-434-5184, http://www.hamamatsu.com
su Corporation: 360 Foothill Road, P.O.Box 6910, Bridgewater, N.J. 08807-0910, U.S.A., Telephone: (1) 908-231-0960, Fax: (1) 908-231-1218
natsu Photonics Deutschland GmbH: Arzbergerstr. 10, D-82211 Herrsching am Ammersee, Germany, Telephone: (49) 08152-3750, Fax: (49) 08152-2658
tsu Photonics France S.A.R.L.: 8, Rue du Saule Trapu, Parc du Moulin de Massy, 91882 Massy Cedex, France, Telephone: 33-(1) 69 53 71 00, Fax: 33-(1) 69 53 71 10
Hamamatsu Photonics UK Limited: 2 Howard Court, 10 Tewin Road, Welwyn Garden City, Hertfordshire AL7 1BW, United Kingdom, Telephone: (44) 1707-294888, Fax: (44) 1707-325777
mamatsu Photonics Norden AB: Smidesvägen 12, SE-171 41 Solna, Sweden, Telephone: (46) 8-509-031-00, Fax: (46) 8-509-031-01
Photonics Italia S.R.L.: Strada della Moia, 1/E, 20020 Arese, (Milano), Italy, Telephone: (39) 02-935-81-733, Fax: (39) 02-935-81-741

Cat. No. KPC1003E0
May 2001 DN