Design of A Three-Wheel Omni-Directional Mobile Robot Base Module

A thesis in the partial fulfillment of the requirements for the degree of

Masters of Engineering

in

Mechatronics

at Massey University, Turitea Campus

Palmerston North

New Zealand

Jinrui Deng

2013

Abstract

In this research, the aim is to develop a modular autonomous mobile robot base that has a certain degree of flexibility and cost effectiveness for some indoor mobile robot applications that may have limited maneuverable space. The structure of the mobile robot and the wheel design are the major investigation areas. A modular mobile robot construction that is able to quickly integrate with different wheels and add on sub-systems has been developed for this project. The experiments made on the test model are positive.

In this project, the mobile robot is built with omni-directional wheels. The omni-directional wheels make the mobile robot maneuverable in its motions. The shape of the mobile robot base and number of wheels that are mounted on the mobile robot were decided based on the structure of the omni-directional wheels.

Modular design makes the omni-directional mobile robot a very practical application. Most of the parts in the omni-directional mobile robot can be easily replaced and reused. The mobile robot itself is constrained to be one that is inexpensive and simple. This would allow others to replicate its concept and improve on it. The control system can be improved by simply replacing the control circuit board on the mobile robot. The new control system can be easily integrated with the peripheral devices, such as motors and sensors. Moreover, the size of the mobile robot base is adjustable, which makes this base design valuable for those who are interested in the development of omni-directional robot applications but are concerned about the size of the robot.

The omni-directional mobile robot consists of three parts: the mechanical system design, the electric and electronic system design, the control system design and programming. When designing mechanical systems, the software "Solidworks" was used. It is a powerful computer aided tool for mechanical drawings and machinery. The software "Altium designer" was selected to design the electric and electronic circuits. It is excellent in drawing schematics of the circuits and producing printed circuit board (PCB). The microcontroller "ATmega32" was selected as the center controller for the mobile robot. It connects every part of the mobile robot together to form the control system. The software "AVR studio" is the communication tool chosen between the microcontroller and the computer, and "C language" is the computer language that was used for programming.

The maneuverability and moving accuracy of the omni-directional mobile robot was examined and the results were analyzed with the computer software "Matlab". The three-wheel omni-directional mobile robot performs differently between its rotational motions and linear motions. The results show that the mobile robot has better moving accuracy when it is in the rotational motions. The maneuverability of the omni-directional mobile robot has been proven to be effective. The mobile robot is able to perform motions like sliding, self-rotating, curving, circling which other mobile robots are either not able to do so or do so with difficulty.

Acknowledgement

I would like to express my deepest appreciation to my supervisor Liqiong Tang , who has the attitude and the substance of a genius: she continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Without her guidance and persistent help this dissertation would not have been possible.

I would like to thank my friend Samuel Garratt, who provided much useful advice through the entire project; and to Bruce Collins and Anthony Wade, in the electronics workshop, for their useful advice, providing of electronics components, and producing of printable circuit boards.

I would also like to express my gratitude to all the staff in the mechanical workshop for their advice and help in making mechanical parts, and to staff in the administration seat for their cheerful and friendly support in all the administrative matters needed in my project.

Furthermore I would like to thank my parents for all the support that they have provided and encouragement that they have given me throughout the entire process.

List of notation

List of Symbols

DIR	Direction
EN	Enable
GND	Ground

List of Abbreviations

ADC	Analogue to Digital Convertors
ADCSRA	ADC Control and Status Register A
ADMUX	ADC Multiplexer Selection Register
AGV	Autonomous Guided Vehicle
ALU	Arithmetic Logic Unit
COM1A	Compare Output Mode for compare Unit A
COM1B	Compare Output Mode for compare Unit B
CS	Clock Select it
CTC	Clear on Timer Compare
DAC	Digital to Analogue Convertor
HDPE	High Density Polyethylene
ICR1	Input Capture Resister in timer 1
IO	Input and Output
OCR1A	Output Compare Resister1 A
OCR1B	Output Compare Resister1 B
P&F	Phase and Frequency correct PWM mode
PCB	Printed circuit board
PID	Proportional–Integral–Derivative
PWM	Pulse Width Modulation
RAM	Random Access Memory
RPM	Revolution per Minute
TCCR1A	Time /Counter1 Control Resister A
TCCR1B	Time /Counter1 Control Resister B
TCNT1	Timer/Counter1 in Timer1 mode
WGM	Waveform Generation Mode

Table of contents

Contents

List of Figures

List of tables

CHAPTER 1- INTRODUCTION

Electronic autonomous mobile robot designs have been widely developed in the last hundred years. From the first electronic autonomous mobile robots back in 1945 (Grey, 1945) to the variety of fanciful robots in recent years, mobile robot designs have become more practical and achievable. In the past, the mobile robots were controlled by heavy, large, and expensive computer systems that could not be carried and had to be linked via cables or wireless devices. Today, however, small mobile robots with numerous actuators and sensors that are controlled by inexpensive, small, and light embedded computer systems which are carried on board the robot have become very popular. There has been a tremendous increase in the interest of studying in mobile robotics. Not just as interesting toys or inspired by science fiction stories and movies, but as a perfect tool for learning engineering, and real life applications which benefit our human society (Vukosavljev, Kukolj, Papp, & Kovacevic, 2011). The prime foci of modern robotic systems are to achieve the designs with high flexibility, moving accuracy, robustness, and efficiency. This research mainly focuses on autonomous mobile robot investigation and study, especially the mobile robot with a special type of wheel design, to achieve better maneuverability. Throughout this research, the aim is to develop a modular autonomous mobile robot base that has a certain degree of flexibility and cost effectiveness for some indoor mobile robot applications that may have limited maneuverable space. The structure of the mobile robot and the wheel design are the major investigation areas. A proposed modular mobile robot construction that is able to quickly integrate with different wheels and add on sub-systems was developed. The experiments made on the test model were positive. With further improvement and the add-on of more intelligent functionalities, the mobile robot should be useful for some of the indoor applications such as service robots in healthcare, hospitality and warehouse management.

1.1 Introduction of autonomous mobile robotics

Autonomous mobile robotics is a fascinating research topic that a large number of researchers have put a lot of effort into, and still, there is great potential for people to study and develop more functional and stable robotic designs. In autonomous mobile robot designs, there are three popular research domains, which are "Mobile robot design principle and control methodologies", "Humanoids mobile robot", and "Service robot" (Ulrich Nehmzow, 2003).

1.1.1 Mobile robot design principle and control methodologies.

Ulrich Nehmzow (2003) stated that "To change a mobile robot from a computer on wheels that is merely able to sense some physical environmental properties through its sensors into an intelligent agent, able to identify features, to detect patterns or regularities, to learn from experiences, to localize, build maps, and to navigate requires the simultaneous applications of many research disciplines. In this sense, mobile robots reverse the trend towards more and more specialization, and demands lateral thinking and the combination of many disciplines. Engineering and computer science form the core of mobile robotics; obviously, when questions of intelligent behavior arise, artificial intelligence, cognitive science, psychology and philosophy offer hypotheses and answers. Analysis of system components, for example, through error calculations, statistical evaluations etc. are in the domain of mathematics, and regarding the analysis of the whole system, physics proposes explanations, for example through chaos theory." (p, 1).

1.1.2 Humanoids mobile robots.

For centuries, the interest that people have shown in building machines which mimic human beings has kept increasing. Developing human like machines that can help us achieve different tasks has drawn great attention from robot researchers. As the robotic technology develops, topics like "walking robots", "robotic arms", "robotic sensing systems" become more popular. People even try to make the robots to be able to anticipate their actions and correct their wrong moves. In other words, the robots are designed to be more and more human like with higher intelligence. As a result, autonomous mobile robots offer a uniquely suited research platform for investigations of intelligent behaviors.

One of the most common humanoid mobile robot applications is the walking robot. Walking robots are considered the most adaptable robot. They can generally move on various environments as their special leg structures enable them to do so.

Robots with six or more legs have the advantage of stability. When a six legged robot is moving, it keeps three legs on the ground at all times, while the other three legs are in the air. As a result, the robot's center of mass will always be within a triangle formed by its three legs on the ground, this gives the robot great stability. (Yanto, XiaoLei, & Bowling, 2004). Figure 1.1 shows an image of the six-legged mobile robot.

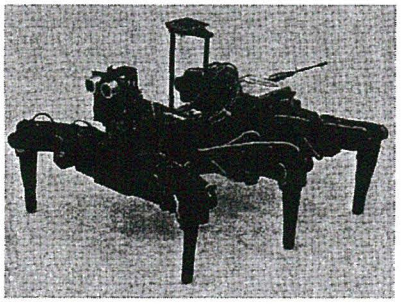


Figure 1.1: six-legged mobile robot.

Four-legged robots are considerably harder to balance, but are still fairly simple when compared to the dynamics of bipedal robots. Figure 1.2 shows an image of a four-legged robot (Kiriki, Kimuro, & Hasegawa, T, 1999).

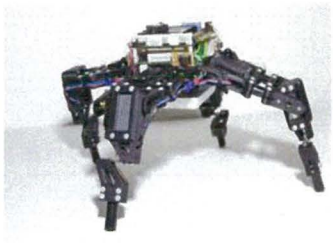


Figure 1.2: four-legged mobile robot.

The bipedal robots are the most difficult to balance, with only one leg on the ground and one leg in the air during walking. Static balance for bipedal robots can be achieved if the robot's feet are relatively large and the ground contact areas of both feet are overlapping (Ali, Amran, &Kawamura, 2010). Figure 1.3 and figure 1.4 show images of two bipedal robots.

Figure 1.3: bipedal mobile robot

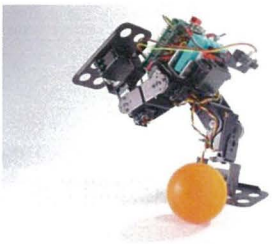


Figure 1.4: bipedal robot with larger feet

Aesthetic and artistic elements are also big considerations for mobile robotics. For example micro-robots, or miniature legged robots, match our sense of beauty. Figure 1.7 shows an image of an aesthetic mobile robot.

Figure 1.5: a good looking robot design.

There are some advantages and drawbacks of the walking robots:

- **The advantages of legged locomotion:**

The walking robots have great adaptability and maneuverability on rough terrain. They can move over rough surfaces or even across holes and chasms, as long as their legs' reach exceeds the width of the irregular ground terrain. In addition, they can move over an object if there is enough clearance between their legs and the ground.

- **The disadvantages of legged locomotion:**

The legged robots are generally hard to balance in their movements, especially for four legged and bipedal robots. The power required for the legged robots is quite high since the legs are able to move in several degrees of freedom and have to be capable of lifting and lowering the robot. Compared to the wheeled robots, the mechanical complexity of the legged robots is another disadvantage. The cost for building a walking robot is generally higher.

1.1.3 Service robots.

Ulrich Nehmzow (2003) pointed out that "There are also commercial applications for mobile robots. Transportation, surveillance, inspection, cleaning or household robots are just some examples. However, autonomous mobile robots have not yet made much impact upon

industrial and domestic applications, mainly due to a lack of robust, reliable and flexible navigation and mechanism behavior for autonomous mobile robots operating in unmodified, semi-constructed environments. Installing makers such as beacons, visual patterns or instruction loops (guiding wires buried in the ground) is one solution to this problem, but it is expensive, inflexible and sometimes outright impossible. The alternative navigation in unmodified environments requires sophisticated sensor signal processing techniques which are still in their experimental evaluation phases. So, to let mobile robots work in areas which are inaccessible to humans, or to perform repetitive, difficult or dangerous tasks, is yet another strong motivation for developing intelligent, autonomous robots." (p,2).

Wheeled mobile robots, being one of the most popular service mobile robot applications, has a unique advantage over the other kinds. First of all, with the great locomotion mechanism of the wheeled mobile robot, effective movement can be achieved with relatively simple mechanical structural design. Secondly, wheeled mobile robots have great performances when running on flat surfaces. They can accomplish the tasks with high efficiency and relatively low power consumption compared to the other types of mobile robots, i.e. legged mobile robots. Lastly, wheeled robots are more robust and easy to maintain. As a result, nearly all industrial applications of mobile robotics apply some form of wheeled locomotion (Bräunl, 2006). Figure 1.5 and figure 1.6 shows two images of a 3 wheeled and a 4 wheeled mobile robot.

Figure 1.6: three wheeled mobile robot

Figure 1.7: four wheeled mobile robot

In this research, the focus will be studying wheeled mobile robots rather than legged robots, due to their simplicity in mechanism, relatively low cost and less power consumption.

From the literature above, there is still high potential for the study of mobile robotics. One of the many directions of the study is to increase the maneuverability for mobile robots. Wheeled robots have limitations when they are running on rough terrain, on the other hand, when they are running in human environments (i.e. engineered, smooth surfaces, both indoors and outdoors), their running smoothness, robustness and maneuverability make them valuable to be studied. In addition, their comparatively low cost is another reason for their popularity. In summary, this research chooses to investigate and study wheeled mobile robots over legged mobile robots. A mobile robot base module with a special type of wheel mount will be developed to achieve better maneuverability. The robot base module should have a certain degree of flexibility and cost effectiveness for some indoor mobile robot applications that may have limited maneuverable space. The structure of the mobile robot and the wheel design are major investigation areas. A proposed mobile robot construction module could be developed to be able to quickly integrate with different wheels and add on sub-systems. The experiments made on the test model are positive. With further improvements and the add-on of more intelligent functionalities, the mobile robot should be useful for some of the indoor applications such as for service robots in healthcare, hospitality and warehouse management.

1.2 The research topic

The aim of this research is to develop a wheeled mobile robot base module for studying mobile robotics with different wheel configurations, and control methodologies. The structure of the base module will allow the components that are mounted on the mobile robot to be interchangeable, and the size of the robot base to be adjustable, which gives the robot base potential to be developed using different add on sub-systems to achieve different tasks.

Moreover, with the wheel structure design, the robot will have a certain degree of flexibility for some indoor mobile robot applications that may have limited maneuverable space. The objectives are:

1 Design a wheeled mobile robot base module. The base module should be adjustable in size and able to easily integrate with different add on sub-systems. The base module also needs to be reliable and robust.

2 Study the wheel structure of the mobile robot. Choosing a specific wheel design for this project which would suit the base well. The wheel design will gain the robot a certain degree of flexibility so that it would benefit some indoor mobile robot applications which may have limited maneuverable space.

3 Design a control system for this mobile robot. The control system needs to be reliable.

1.3 The scope of the research

The scope of this research is to develop a reliable, robust, and economic mobile robot base module. The base can be easily adjusted in size and integrated with different add on sub-systems. In addition, a wheel structural design which gains the mobile robot a high maneuverability will also be developed. The mobile robot should be able to benefit some indoor mobile robot applications that may have limited maneuverable space.

- ✧ Mobile robot design with modular design methodology.
- ✧ Design of a wheeled mobile robot mechanical system.
- ✧ Design of a mobile robot electric and electronic system.
- ✧ Mobile robot control system design and programming.
- ✧ Testing prototype building

1.4 The organization of the thesis:

The thesis contains eight chapters. Each chapter has a short introduction to give an overview of its content. Following is a summary of the chapters. Chapter 1 explains the topics of this research by stating its requirements, objectives and scope. Chapter 2 provides a background to the history behind mobile robots, the available approaches and techniques used in moving a mobile robot around an indoor environment, and the growing need for mobile robots in general. Chapter 3 introduces the history of omni-directional wheels, characteristics of different omni-directional wheels, and omni-wheel applications with different wheel combinations. Chapter 4 discusses the mechanical design of the mobile robot base module. It firstly introduces the concept of modular design, and then presents the mobile robot hardware selection. Lastly, the process of designing the mobile robot mechanical system is described. Chapter 5 explains the electric and electronic system design of the mobile robot. It discusses the selection of the electric and electronic components of the robot. Most importantly it describes the process of the electric and electronic system design. Chapter 6 presents the mobile robot's control system. It introduces the controller that is chosen for this project and the programming for the robot's control system. Chapter 7 demonstrates the testing prototype build for the mobile robot. Five tests were set for the mobile robot, so that its linear and rotational motions could be examined. The collection and analysis of the results are presented. Chapter 8 concludes with the research's future improvements and conclusions.

Chapter 2 – LITERATURE REVIEW

Autonomous wheeled mobile robot design is widely applied in automatic control systems. However, it is limited by operating in unmodified, semi-constructed environments, which is due to its lack of robust, flexible navigation, maneuverability, and mechanism behavior. As a result, there is potential to develop an autonomous mobile robot base module that has a good maneuverability and cost effectiveness for some indoor mobile robot applications that may have limited maneuverable space. The module needs to be easily integrated with different add on sub-systems and adjustable in size. An appropriate wheel design could be developed to assist the base module design, and a control system could also be developed to operate the robot to fulfill its tasks.

2.1 The Development of Mobile Robot

Niemueller(2003) figured that: "A robot is a mechanical or virtual intelligent agent that can perform tasks automatically or with guidance, typically by remote control. In practice, a robot is usually an electro-mechanical machine that is guided by computer and electronic programming. Robots can be autonomous, semi-autonomous or remotely controlled. Robots range from humanoids such as ASIMO and TOPIO to Nano robots, Swarm robots, Industrial robots, military robots, mobile and servicing robots. By a mimicking lifelike appearance or automating movement, a robot may have intent or agency of its own". (p, 1).

The development of mechanics had greatly cut down the human and animal labors back in time. As more requirements for mechanical applications had increased with the progress of human civilization, more original ideas of designing complex mechanisms were formed. This had a great impact for later on the generation of modern robots.

The complex mechanisms were improved with the development of the semi-conductors. The amazing electronic components had brought a tremendous innovation to the pure mechanisms. Therefore, the age of the modern robotics had started. It was introduced by Niemueller(2003) that: "The modern robots are based on two enabling technologies: Telemanipulators and the ability of numerical control of machines. Telemanipulators are remotely controlled machines which usually consist of an arm and a gripper. The movements of arm and gripper follow the instructions the human gives through his control device. First telemanipulators have been used to deal with radio-active material. Numeric control allows controlling machines very precisely in relation to a given coordinate system. It was first used in 1952 at the MIT and lead to the first programming language for machines. The combination of both of these techniques leads to the first programmable telemanipulator. The first industrial robot using these principles was installed in 1961. These are the robots one knows from industrial facilities like car construction plants." (p,1).

Niemueller(2003) also stated that: "The development of mobile robots was driven by the desire to automate transportation in production processes and autonomous transport systems. The former lead to driver-less transport systems used on factory floors to move objects to different points in the production process in the late seventies. New forms of mobile robots have been constructed lately like insectoid robots with many legs modeled after examples nature gave us or autonomous robots for underwater usage. Since a few years wheel-driven robots are commercially marketed and used for services like "Get and Bring" (for example in hospitals). Humanoid robots are being developed since 1975 when Wabot-I was presented in Japan. The current Wabot-III already has some minor cognitive capabilities. Another humanoid robot is "Cog", developed in the MIT-AI-Lab since 1994. Honda's humanoid robot

became well known in the public when presented 1back in 1999. Although it is remote controlled by humans it can walk autonomously (on the floor and stairs)." (p.1).

2.2 Mobile robot structure and drive

With mobile robot's fast development, the structure and drive mode of the robot designs have attracted a great focus; as they are the key factors that decide the mechanic configurations for the robot. Great robot structural designs and drive mode selections give the robot greater stability, moving accuracy and efficiency. Therefore, it is necessary to take these two factors into consideration in this research.

There are many different kinds of structural design and drive mode selections for mobile robots. I.e. using one DC motor drive two wheels is the easiest way of driving a mobile robot. This robot is only able to move forward and backward. In this section, several higher level drive designs such as differential drive, synchro-drive, and Ackermann steering will be introduced, and their advantages and drawbacks will be discussed.

2.2.1 Single wheel drive:

The single wheel drive is one of the simplest drive modes. Its two rear wheels connect to one motor which provides the driving force for the mobile robot. There is a front wheel (caster wheel) which directs the robot when it turns at the corner. Figure 2.1, 2.2 and 2.3 illustrate the 3 different motions of the single wheel drive.

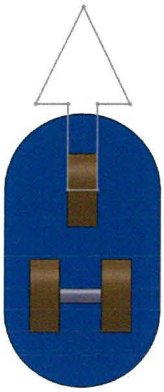


Figure 2.1: single drive "forward"

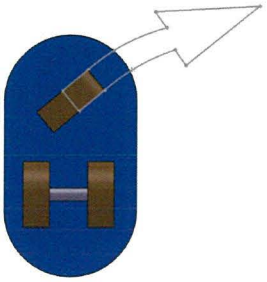


Figure 2.2: single drive "curve"

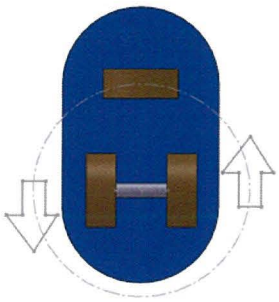


Figure 2.3: single drive turning on the spot

Figure 2.1 shows that, when the caster wheel points straight forward, if the two rear wheels rotate forward, the robot will move forward, on the contrary, if the two rear wheels rotate backward, the robot will reverse. From Figure 2.2, it can be seen that when the caster has an angle to its forward direction, the mobile robot will then be driven to move alone a curve track. The sharpness of the angle determines the radius of the curve. When the caster wheel forms an angle of 90 degree to the forward direction (shown in Figure 2.3), the force that is provided by the two rear wheels will be perpendicular to the caster wheel. Therefore, the robot will not be able to turn on the spot.

Advantages and drawbacks:

Single wheel drive is the simplest mechanical structure, so it is very easy to build. With only one motor to drive the robot, single wheel drive is an ideal low cost robot. However due to its simplicity of mechanical design, its maneuverability has been greatly limited. Moreover, it has a relatively low moving accuracy since the robot is only directed by the front caster wheel.

2.2.2 Differential Drive

Differential drive has two wheels connected with two motors which provide the force for the robot to move. Differently from the single drive, the two wheels are not connected, which means that they can rotate on different speeds. By changing the speeds of the two wheels, the robot will be able to perform different movements. A caster wheel is not essential for the differential drive, since the robot is directed by the two wheels only.

Figure 2.4: differential drive "forward"

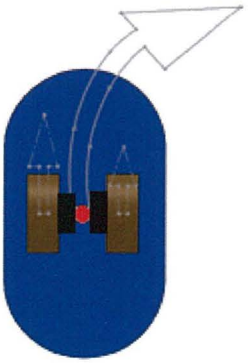


Figure2.5 differential drive "curve"

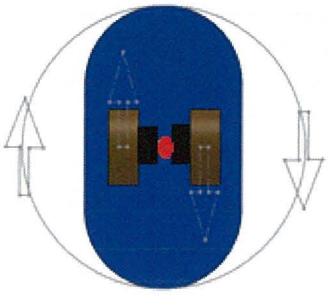


Figure 2.6: differential drive "turning on the spot"

It is shown in Figure 2.4 that, when the two wheels rotate at the same speed (forward or backward), the robot will move straight forward or backward. When the two wheels rotate at different speed, as it is shown in Figure 2.5, the robot will follow a curve track. By adjusting the speed difference between the two wheels, the radius of the curve can be set. When the two wheels rotate on the same speed but opposite directions, the robot will turn on the spot.

Advantages and drawbacks:

Differential drive has better maneuverability than single wheel drive. This is because with two wheels rotate on different speeds, the robot is able to perform straight, curving, and rotating movements. The relatively low cost is another factor which interests many robotics students. However, it is widely known that the motors do not perform perfectly as they are told to. As a result, even given the same speed, two motors in the same module will not rotate at the speed; one is always more or less, faster or slower than the other. This will directly generate a speed difference between the two wheels, which will affect the performance of the mobile robot. In addition, if the robot runs over uneven surface, contact lost between the wheels and ground would occur, this also affects the accuracy of the mobile robot's performance. Lastly, the differential drive does not gain the robot the ability to perform fancy movements, such as sliding and turning sharp angles.

2.2.3 Tracked Robot

A tracked mobile robot can be seen as a special case of a wheeled robot with differential drive. In fact, the only difference is the robot's better maneuverability in rough terrain and its higher friction in turns, due to its tracks and multiple points of contact with the surface. It has a better flexibility driving the robots through rough terrains. The drawback of the tracked drive is the complexity of its mechanical design; the tracked drive design is normally complex and expensive (Weidong, Lei, Zhijiang, & Lining, 2008).

Figure 2.7: Tracked wheel drive

2.2.4 Ackerman Steering

Before Ackerman steering first came out, when driving a vehicle and steering to one side (either left or right), the 4 wheels of the vehicle would rotate at the same angular speed. As a result, some of the wheels would slip. The slipping problem becomes even worse when the vehicle runs at high speeds. Ackerman steering, with the idea of have a central point that all the wheels will rotate around, solves the slipping issue. Figure 2.9 is an image of ackerman steering. It can be seen that when the vehicle is steering, two rear wheels (wheel 4 and wheel 2) have slower angular speeds than the two front wheels (wheel 3 and wheel 1). The paths of each wheel have different radius, which means the four wheels runs at different speeds. The wheel on the outside (path with greater radius) travels further distances than the wheel on the

inside (path with smaller radius). The angular velocities of the 4 wheels follow the descending order of 1, 2, 3 and 4. The left side wheels move faster than the right side wheels when the vehicle is turning right, and the front wheels move faster than the rear wheels when the vehicle is steering. Thus, the wheels do not have to slip when they turn around (Weinstein & Moore, 2010).

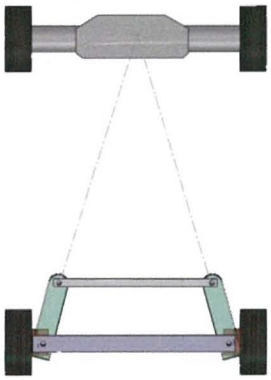


Figure 2.8: Ackermann steering "forward"

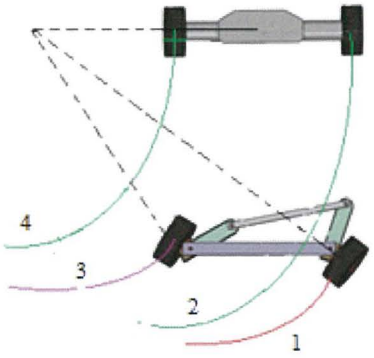


Figure 2.9: Ackermann steering "turning"

Ackerman steering gains the vehicle a smooth turning motion, makes the vehicle's driving controllability better than normal differential drive designs. This is the main reason that Ackerman steering is widely used in car industry. In addition, it reduces the tyre wear caused by any turning motions. While on the other hand, there are also some drawbacks. Ackerman steering cannot turn on the spot, neither can it slide. Moreover, its mechanical complexity is not necessary for this project.

2.2.5 Omni-directional drive

All the mobile robots and drive modes introduced above have the similar deficiency: they cannot drive in all possible directions. Therefore, a new type of drive "omni-directional drive" has come in sight for many robotics enthusiasts. The omni-directional robots, with the help of onmi-directional wheels, are able to drive in any directions as it is commanded, such as sliding and center rotating. The great maneuverable performance of their rotational motions is the most important factor for their popularity. Therefore, the omni-directional drive is selected for this project over the other drives. The omni-directional wheel and omni-directional drive are presented in chapter 3.

2.3 Mobile robot control methodologies

It can be seen from the previous section 2.2 that, after the robot's structure and drive have been selected, the robot is supposed to move as it was told to. However, in reality, the mobile robot will be never able to fully follow the commands it received. The motors on the mobile robot will always attempt to rotate faster or slower than their given values. This is because (using a simple differential drive with two motors as an example):

1. The two motors will never perform the same even they were given the exact same commands. Motors do not run at a constant speed from the beginning to the end of the motion, they accelerate and slow down through time. When the motors drive with different loads, they tend to draw more or less power to satisfy the output torque that required from the loads. Moreover, due of the mechanical insufficiency in robotic manufacture, the center of the robot's mass will not locate in the middle of the two motors. So that the weight of the robot is unevenly put on the two motors. In addition, when the robot is carrying other loads, the loads are most likely distributed unevenly on the two motors. The uneven loads will cause the two motors to draw different power from the power supply; therefore, even if the motors were set

to rotate at the same speed, they will still output different torques toward different loads and rotate at different speeds.

2. When the robot moves forward and encounters an unsmooth surface, there is a high possibility that its wheels are not fully and constantly contacting the ground. The wheels will then slip and cause errors in terms of a shifted displacement or angle in the robot's motion. The robot will tend to turn to one side instead of moving straight forward.

To be able to adjust the movements of the robots, control methodologies for robotics are essential. The simplest control methodology is the open loop control. In open loop control, the controllers of the robot only send commands to the actuators on the robot, such as motors or sensors. The motors or sensors will then carry out the commands. If any actuators generated any errors, the performances of the robot would be more or less affected. To be able to correct the errors, the closed loop control methodologies were developed. For the closed loop control methodologies, a feedback from the actuators is necessary. For instance, an encoder is a device that is able to collect the rotational speed of a motor. By mounting an encoder to a motor, the speed of the motor can be observed, and if any strange performances of the motor occurred, the information will be collected and sent back to the controllers. Controllers can then make new commands to adjust the motor's actions. The entire adjusting process is called closed loop control. There are many closed loop control methodologies; the following section will introduce some popular ones.

2.3.1 On-off control

The on-off control is the simplest closed loop control. The idea of on-off controller is to set two thresholds to constantly compare with the current value. If the current value drops down to the bottom threshold, the power will be turned on and the current value will be boosted up. If the current value reaches the top threshold, the power will be cut off, and the current value will decrease.

For instance, assuming that the current value is the speed of the motor, and the two thresholds are set to be the maximum and minimum speed that the motor can rotate. If the current speed is too slow, down to the minimum threshold, the controller will power on the motor, and boost its speed up, then power off the motor as soon as the maximum speed is reached. The motor will then be repeatedly powered on and off and rotate at speeds with variable values. If setting the two thresholds to a speed interval, such as 1000 revolutions per second (top) to 800 revolutions per second (bottom), and if the desired speed of the motor is 900 revolutions per second, the speed of the motor will be locked in this interval and stay around the desired speed.

The motor is not directly powered on and off (the thresholds have to be reached); therefore, there is a delay on the power actions.so that the on-off controller is also known as a hysteresis controller.

2.3.2 Proportional–Integral–Derivative (PID) control

The proportional–integral–derivative (PID) controller is simple, efficient and effective in a wide range of applications. In fact, it is the majority of controller types in industrial applications. Selvakumar (2013) stated that: "A PID controller calculates an "error" value as the difference between a measured process variable and a desired set point. The controller attempts to minimize the error by adjusting the process control inputs.

The PID controller calculation (algorithm) involves three separate constant parameters, and is accordingly sometimes called three-term control: the proportional, the integral and derivative values, denoted P, I, and D." (p,1).

The proportional controller creates a value that is proportional to the current error value. The proportional gain is a constant value. The proportional controller quickly boosts up the output value to reach the reference value and eliminate the error. As the error value gets smaller, the

proportional controller reduces its adjustment correspondingly. When the error is completely eliminated, the proportional controller stops working. If the proportional gain is too high, the system will become unstable, and if the proportional gain is too low, it will take long time for the controller to eliminate the error, which is not effective. Thus, setting a proper proportional gain is important in designing a proportional controller.

The derivative controller restraints the changing rate of the output value. Since proportional controller make the error elimination very fast, so that using a controller slows it down, to guarantee a certain output value can be achieved without having the proportional controller overshooting the error value. Proportional and derivative controller work together forms a PD controller, which is widely used in industrial automatic systems.

The contribution from the integral controller is proportional to both the magnitude of the error and the duration of the error. A PID controller contributes greatly to automatic systems; it cuts off the error values and stabilizes the system.

2.3.3 Other controller

There are some other controllers which are widely used in modern automatic control systems, such as fuzzy controller which is based on the theory of the fuzzy logic, and the neural network controller etc. Each controller has its great advantages and is suitable for different types of systems.

In conclusion, many control methods can be applied in this project. Since the main focus of this project are to study the structure of an omni-directional mobile robot base module, and test the maneuverability and moving accuracy of the designed mobile robot base, the advanced control method is not the primary. Therefore, the "on-off" control which is the basic control method is selected for the mobile robot's control system, due to its simplicity. The other control methods such as PID control can be added for the future development.

2.4 Modular design and applications

2.4.1 Modular Design and its Benefit

Modular design is an approach that subdivides a system into smaller parts (modules) that can be independently created and then used in different systems to drive multiple functionalities. Modular design generally has many benefits, such as it reduces the cost in design. Modular design divide the system into smaller parts, then each subdivided smaller part can be mass-produced. When building the system, what is left to do is to simply assemble the smaller parts together and the system will be complete. The smaller parts are detachable, replaceable and reusable. People do not need to spend much time on learning the basic information of the system; instead, they only need to directly apply and operate the modules, which make the learning process much more effective.

2.4.2 Modularization in robotic Design and Applications

Jantapremjit. P & Austin. D. (2001) stated that: "A modular robot can be defined as a robotic system constructed from a set of standardized components (or building blocks). Modular robots are of interest because they permit construction of a wide variety specialized robots from the set of the standard components. Over the past 20 years, research efforts in the field of modular robotics have been directed towards robotic manipulations with the goal of versatility and adaptability. Less effort has been made in the field of self-reconfigurable modular robots, which are modular robots that can autonomously change their configuration." (p,1).

There are many advantages of using modular design with robots:

Reusability: Budgeting and efficiency are very important considerations when working on a project. The first advantage of using modular design is resources reuse. Some electronic and mechanical parts in a system are very expensive, so reusing parts is a desirable property.

Upgrade, Replace and Repair: Modules should be designed to be upgradable, replaceable and repairable. Being upgradable is important for a system, since the fast pace of the development in technology requires good systems to be able to keep up with the times. Then, being replaceable gives the system a great tolerance for mistakes; if one part or module in the system is broken, it can be easily solved by replacing the broken part. If the broken part can be easily fixed, which means that replacing the part is not necessary, repairing will just do the job.

Design and building: Modularizing a system might increase its complexity; but it may actually help the other way. It is often easier to deal with a number of well-defined connected systems, than one large bulk of a system. Designing a small power board, then a motor driver board, and a sensor board would be easier than designing a power/motor/sensor board. As well as simpler design, modularization also works for simpler assembly; this means that every section of the system can be built as a complete module. Each of them can be tested as a separate part, and if there are any problems, changing one module would get them solved as opposed to changing many.

There are also some drawbacks of using modular design. One big disadvantage is that applying modular design to a system normally requires a great deal of forethoughts, researches and designs. However, a design with good quality and economic efficiency is

always the priority. In this research, apply modular design to the mobile robot system is one of the main goals.

Chapter 3: Omni-wheel design and the characteristics

3.1 General introduction of omnidirectional wheels

Omni-directional wheels have been widely used in robotics and mechatronic systems for many years. There are two popular types of omnidirectional wheels which are popular in mobile robot design, which are the omni-wheels and the mecanum wheels. Omni-wheels and mecanum wheels are very similar in that they both have rollers built into the wheels. The difference between them is that, omni-wheels have the rollers at a 90-degree angle to the direction of the wheel; the structure of mecanum wheels has rollers at a 45-degree angle. Figure 3.1 and 3.2 show the images of the two types' omnidirectional wheels.

Figure 3.1: Omni-wheel

Figure 3.2: Mecanums

Nowadays, many advanced mechanical applications with omni-wheels have been developed for industrial or domestic usage, such as omnidirectional conveyor systems, wheel chairs, service vehicles etc. One of the most popular applications is the omni-directional Autonomous Guided Vehicle (AGV). Omni-directional AGV helps the normal autonomous guided vehicles to be able to achieve omni-directional motions. Vehicles with normal wheels have more or less, some limitations in moving maneuverability; however, with the assist of the omni-directional wheels, their maneuverability can be greatly improved, they can perform

advanced motions which the normal vehicles cannot or hardly can, such as center rotating and sliding movements.

Omni-directional AGV applications have been utilized in industrial manufacturing field. With the ability of omni-directional moving, the omni-directional AGV operates in the places which the normal AGV would hardly do. Places such as wards of the hospital, or offices which have many narrow path and blocking obstacles (beds, tables, chairs) require a very high degree of flexibility in locomotion for an AGV application and the omni-directional AGVs just meet the qualification. In this research, an omni-directional AGV is developed to have the potential servicing in these places with high requirements of the vehicle's maneuverability. To start with, the study of omni-directional wheel is necessary.

3.2 The Development of Omni-directional wheels

Rojas (2004) metioned that: "The first omnidirectional wheel was designed by J. Grabowiecki in the US, which consists of a main wheel and several transversal rollers. It was considered to be functional as a conveyor system. Figure 3.3 shows the original image of the first omnidirectional design.

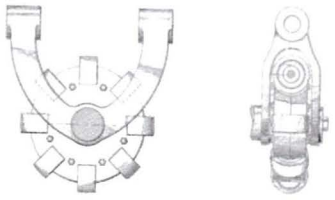


Figure 3.3: the original omnidirectional wheel

Not long after this design, the inventors committed themselves to modifying the wheels; they developed the vehicle with omnidirectional wheels to be capable with moving forward or sideways without steering the wheels.

After many attempts, one of the first modern omnidirectional wheels was designed by Bengt in 1973; the profile of the wheel is almost circular. Figure3.4 is an image of this design.

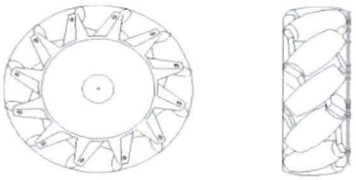


Figure 3.4: the first modern omnidirectional wheel

After that, a new wheel shape was designed: a polygon shaped wheel with small transversal rollers at the polygon's corners. " (p,1-3). Figure 3.5 shows an image of this kind of wheels.

Figure3.5: polygon shaped wheels

There was also a disadvantage with this type of wheels, which is that the gaps between the rollers would cause the wheels to rotate roughly, and the vehicle keeps shaking while moving.

Later on, the idea of making the profile of the wheel none-gap contact between the rollers came out. In order to make the wheel run smoothly, the profile of the polygon wheels are manufactured round by using a stack of two polygon shaped wheels. The gaps are eliminated since the contacting points to the ground change from one polygon wheel to the other. This new type of wheel which is later on called the omni-wheel was developed and improved to be more stable and precise in its motions. The omni-wheel is now one of the most popular wheels that are applied in the modern omni-directional vehicle designs. Figure 3.6 is an image of a two stack omni-wheel.

Figure 3.6: Two Stack omni-wheel

The other possible way to design the omni-directional wheels is manufacturing the wheels to have shapes as spheres or quasi-spheres, but the complexity of the wheel control is a drawback.

3.3 The structure of Omni-directional wheeled mobile robots

To understand the omni-directional wheels and equip them on mobile robots, a good starting point is to study the structure of omni-directional mobile robots with different wheel combinations. There are two popular types of omni-directional wheels which are generally applied in mobile robot manufacture, which are the mecanum wheel and omni-wheel (see 3.1).

In the following section, the two different kinds of omni-directional wheels will be discussed separately.

3.3.1 The structure of Mecanum wheeled mobile robots

Applying four mecanum wheels on a mobile robot is generally considered as the most effective way of building a mecanum wheeled mobile robot. Two front wheels and two rear wheels are all parallel to each other (Figure 3.7). The force motion analysis of the four mecanum wheeled mobile robot will be presented in the following section.

a. Force analysis

Figure 3.7 presents the structure of a 4 mecanum wheeled mobile robot. The 4 wheels which are presented as W1, W2, W3 and W4 are all parallel to each other. Two front mecanum wheels have the same shape except the rollers which are arranged on the hubs of the mecanum wheels point to opposite directions. The rollers on the rear wheels point to opposite directions as well. The angle between the rollers and the hub of a mecanum wheel is 45 degrees. As a result, when the wheels rotate, the driving force splits into 2 parts, half of the force is still driving the mobile robot to move forward, while the other half is dispersed to a perpendicular direction to its forward direction. The perpendicular force will drag the mobile robot away from its forward moving track, so it has to be eliminated. By pairing the two wheels which have the rollers on the hubs of the mecanum wheels arranged opposite to each other, the perpendicular forces generated by the two motors can be cancelled.

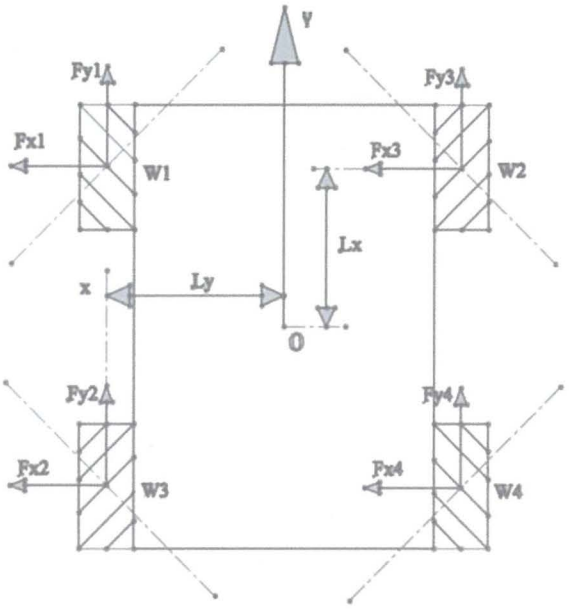


Figure 3.7: Analysis of the force vectors of the Mecanum wheel

a.1 Translational motion

In Figure 3.7, "O" represents the origin of the XY coordinate system, and X and Y are the axis. The force from 4 wheels can be split into X and Y components and then summed to get the following calculations (1) and (2) to define the total force applied on the vehicle:

Equation 3.1: calculation of total force on the X axis for 4 mecanum-wheeled mobile robot

$$F_x = \sum_{w=1}^{4} F_x w \qquad (3.1)$$

Whereas in Equation 3.1, Fx represents the total force on X axis and "W" represents the wheel number.

Equation 3.2 calculation of total force on the Y axis for 4 mecanum-wheeled mobile robot

$$F_y = \sum_{w=1}^{4} F_y w \qquad (3.2)$$

Whereas in Equation 3.2, Fy represents the total force on Y axis and "W" represents the wheel number.

If there is pure translation, then the direction of the movement makes an angle α (shown in Figure 3.8) with the X axis. This angle can be presented as Equation 3.3:

Equation 3.3 calculation of angle of direction for 4 mecanum-wheeled mobile robot

$$\alpha = arctan\frac{F_y}{F_x} \qquad (3.3)$$

Whereas in Equation 3.3, Fx and Fy represent the total force on X and Y axis and "α" represents angle of direction.

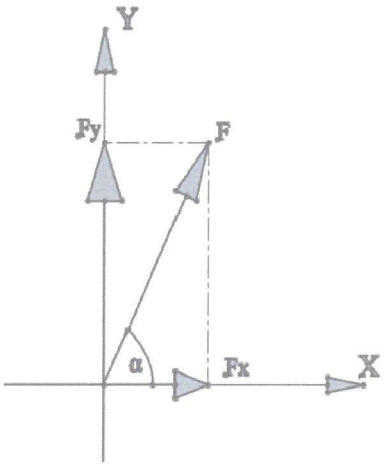


Figure3.8: Angle α between the Total force and the X axis

a.2 Rotational motion

If there is any rotational motion, the total torque τ can then be calculated as Equation 3.4. Defining positive rotation as counter-clockwise around the center of the mobile robot "O", the resulting torque can then be presented with the following equation.

Equation 3.4 calculation of total torque for 4 mecanum-wheeled mobile robot

$$\tau = (-F_x1 - F_x2 + F_x3 + F_x4) * L_y + (F_y1 - F_y2 - F_y3 + F_y4) * L_x \quad (3.4)$$

Whereas in Equation 3.4, τ represents the total torque that the mobile robot gains from its 4 wheels, Fx represents the force split into the X axis, and Fy represents the force split into the Y axis. Lx and Ly represents the distances from the origin to the split force Fx and Fy.

b. Motion simulation of the 4 mecanum wheeled mobile robot

The motion simulation of a 4 mecanum wheeled mobile robot is demonstrated in Figure 3.9.

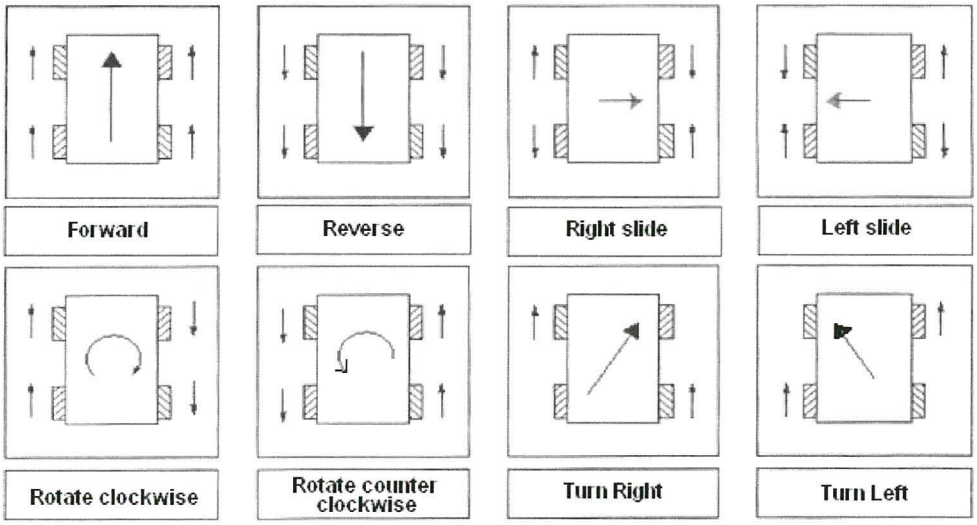


Figure 3.9 motion simulation of the 4 mecanum wheeled mobile robot

When the mobile robot moves forward or reverse, the 4 wheels rotate into the same direction (either forward or backward). Different from the forward and reverse motion, sliding motion requires the wheels to rotate into different directions. When the robot slides to its right, the two wheels on the left hand side rotate away from each other, and the two wheels on the right hand side rotate toward each other. When the robot slides to its left, the 4 wheels rotate into the opposite directions to the right sliding motion. With the wheels on the left hand side of the mobile robot rotating forward, and the right hand ones rotating backward, the mobile robot is able to perform clockwise rotation, and when the wheels rotate into the opposite directions the mobile robot is able to accomplish counter clockwise rotation. With only the front left wheel and rear right wheel running forward, the mobile robot is able to turn to its right, and with front right wheel and rear left wheel running forward, the mobile robot is able to turn to its left. (Salih, Rizon, Yaacob, Adom, & Mamat, 2006).

3.3.2 The structure of omni-wheeled mobile robots

The number of wheels is an important consideration in designing an omni-wheeled mobile robot. The most popular shape designs of omni-wheeled mobile robot base are the three-omni-wheeled triangle shape and the four-omni-wheeled square shape.

a. Three-omni-wheeld mobile robot

Three omni-wheel driving system is very commonly used due to its good maneuverability and the simple control. Figure 3.10 is an image of a three-omni-wheeld mobile robot base. The shape of the base is an equilateral triangle. Three omni-wheels are mounted on the 3 tips of the triangle which form 120 degrees angles between each other. The axis of the wheels intersects at the center of mobile robot.

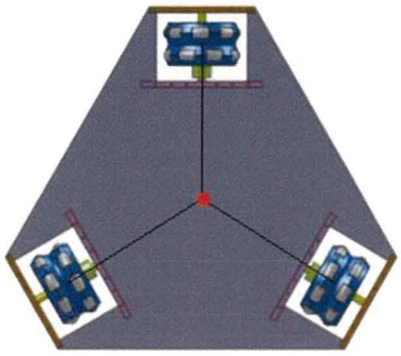
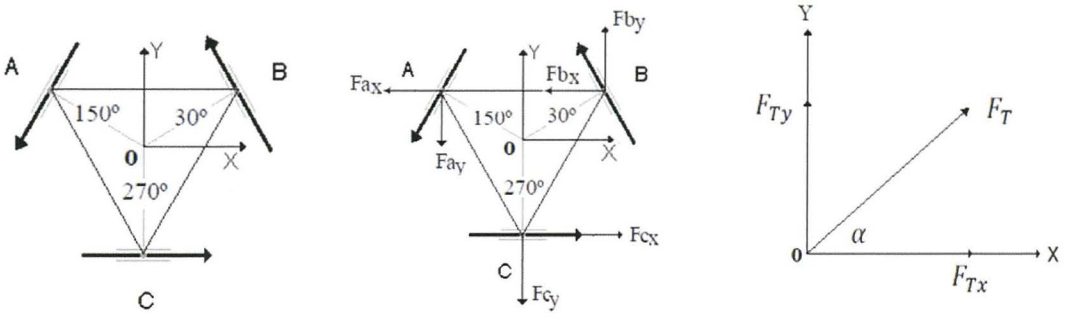


Figure 3.10: an image of a Three-omni-wheeled mobile robot base

a.1 Force analysis

a.1.1 Linear movement

Figure 3.11 shows the force analysis of the 3 mecanum wheeled mobile robot, as it can be seen from Figure 3.11 (left) that, the center of the mobile robot is the point "O". A coordinate system was created. The origin of the coordinate is the center of the mobile robot "O", X and Y axis are marked by light arrows. The force vectors generated by each wheel are represented by arrows A, B and C, and their directions relative to the coordinate axis X, are 150° (A), 30° (B) and 270° (C) respectively.



Left Middle Right

Figure 3.11: graph of force analysis in linear motion

In Figure 3.11(middle), force provided by the wheels A, B and C are presented as FA, FB and FC. Each force is split into two components, which are Fax, Fay; Fbx, Fby; and Fcx and Fcy respectively. The total force that is applied on the X axis can then be calculated as:

Equation 3.5 calculation of total force on X axis for the 3 omni-wheeled mobile robot

$$F_x = Fax + Fbx + Fcx \quad (3.5)$$

Whereas Fx represents the vector of the total force on X axis.

Equation 3.6 calculation of total force on Y axis for the 3 omni-wheeled mobile robot

$$F_y = Fay + Fby + Fcy \quad (3.6)$$

Whereas Fy represents the vector of the total force on Y axis.

Figure 3.11(right) demonstrates the direction that the mobile robot is running in. Due to the vectors of the total force on X and Y axis, there is an angle between the mobile robot's heading direction and the X axis. By calculating this angle, the direction of the mobile robot can be known.

Equation 3.7 calculation of angle of direction for the 3 omni-wheeled mobile robot

$$\alpha = arctan\frac{F_y}{F_x} \quad (3.7)$$

Whereas α represents the angle between the X axis and the direction of the mobile robot. Fx and Fy are the total force on X and Y axis.

a.1.2 Rotational movement

The pure rotational motion of the mobile robot should make the robot rotate around its center. In Figure 3.12, the circle represents the direction that the mobile robot rotates in. "*l*" is the distance between the center of robot mobile and the center line of the wheel.

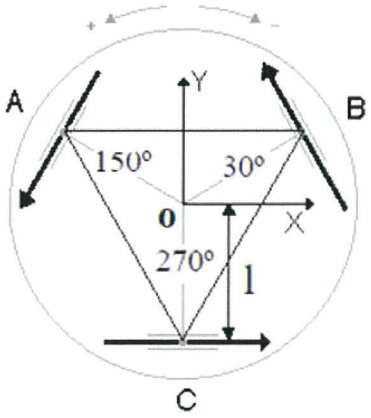


Figure3.12: inversed graph of rotating analysis

The total torque Tτ then results in the equation:

Equation 3.8 calculation of total torque for the 3 omni-wheeled mobile robot

$$\boldsymbol{T\tau = \tau A + \tau B + \tau C = (FA + FB + FC) * l} \qquad \textbf{(3.8)}$$

Whereas Tτ represents the total torque that is applied on the mobile robot, and τA, τB and τC represent the torque provided by the wheel A, B and C. FA, FB and FC represent the force that is provided the wheel A, B and C, and *l* the center to center distance.

b. Four omni-wheeled mobile robot

Figure 3.13 is an image of a four omni-wheeled mobile robot. It can be seen that A, B, C and D represent the four wheels. "O" is the center of the chassis. The chassis is a square. The distance between the center "O" and the center line of the wheel is represented as "*l*".

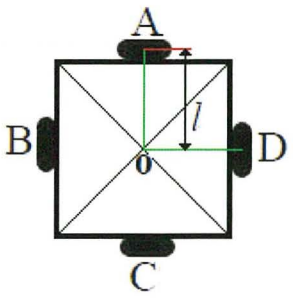


Figure 3.13 four omni-wheeled mobile robot structure

b.1 Force analysis

b.1.1 Linear movement

Force analysis for the linear movement of a 4 mecanum wheeled mobile robot is quite simple. Figure 3.14 demonstrates the force that is provided by the four wheels. FA, FB, FC and FD represent the force from the wheel A, B, C and D respectively.

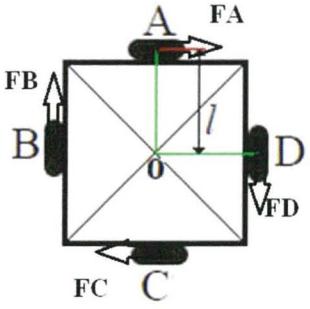


Figure 3.14 four omni-wheel robot structure

The total force of the 4 wheels is represented as F. Assuming there is a coordinate system which has the origin as the center of the mobile robot "O", and horizontal direction as X axis, vertical direction as Y axis, so that the total force will result in:

Equation 3.9 calculation of total force in X axis for 4 omni-wheeled mobile robot

$$Fx = FB + FD \quad (3.9)$$

Whereas Fx represents the total force on the X axis, FB and FD are the force that is provided by the wheel B and D.

Equation 3.10 calculation of total force in Y axis for 4 omni-wheeled mobile robot

$$Fy = FA + FC \quad (3.10)$$

Whereas Fy represents the total force on the Y axis, FA and FC are the force that is provided by the wheel A and C.

While the angle α is the direction of the resultant displacement vector, which can be presented in Equation 3.11.

Equation 3.11 calculation of total force in Y axis for 4 omni-wheeled mobile robot

$$\alpha = arctan\frac{F_y}{F_x} \quad (3.11)$$

Whereas in Equation 3.11, Fx and Fy represent the vector of the total force on X and Y axis. "α" represents the angle of direction.

b.1.2 Rotational movement

In the rotational movement, the total torque of the mobile robot is equal to the sum of the 4 torques from the 4 wheels. It can be seen from Figure 3.14 that the torque "τ" from each

wheel is equals to force multiply the center to center distance "*l*", which will result in the total torque "Tτ" equations:

Equation 3.12 calculation of total torque for the 4 omni-wheeled mobile robot.

$$T\tau = \tau A + \tau B + \tau C + \tau D = FA * l + FB * l + FC * l + FD * l \quad (3.12)$$

Whereas Tτ represents the total torque, τA, τB, τC and τD represent the torques that are provided by the 4 wheel A, B, C, and D. FA, FB, FC and FD represent the force provided by the 4 wheel A, B, C, and D; *l* represents the center to center distance.

3.4 conclusion

The mecanum wheel and omni-wheel all have very great maneuverability and suitable for this project. Because the omni-wheels are economic applicable, this project chooses omni-wheel over mecanum. Three-omni-wheeled and four-omni-wheeled mobile robots are actually quite similar in their structures, it can be seen from the force and torque analysis that both mobile robot structures use the same method for their calculations. It is only the matter of fact that four-omni-wheeled mobile robot has one more wheel, so it provides more force and torque for the mobile robot to perform. However, considering the cost of an extra motor and wheel, the four-omni-wheeled mobile robot structure is not the first choice for this project. In conclusion, this project will select the three-omni-wheeled base structure for the mobile robot, since it has great maneuverability, and it is economically practical. More of the three-omni-wheeled drive control will be discussed in Chapter 7.

CHAPTER 4 Mobile Robot Mechanical System Design

In this chapter, the mechanical system design of the mobile robot is introduced. The three-omni-wheeled mobile robot is designed to be able to achieve the goal of having a good maneuverability in its running motions. The mobile robot should be flexibly running in a place where space is limited, i.e. a narrow path, or in between the tables or chairs. Modular design is applied to the mobile robot system which makes the robot base to be easily adjusted in size, and have the ability to integrate with add-on subsystems. The mechanical design process of the mobile robot is presented, which includes the mechanical components selection and drawing, the mechanical drawings are done with using the software Solidworks.

4.1 Mobile robot system configuration

It is generally known that the human body structure has three main parts: the skeleton, which supports the body's daily activities; the organs and blood vessels, which produce and consume energy and transfer it to different parts of the body; and most importantly, the brain, which creates thoughts, i.e. making commands, process the information collected from the sensors (eyes, nose, mouth etc.), and then makes responses. The concept of robotic systems is quite similar, since the concept of robot design is originally from observing the behavior of the human body. Therefore, mobile robot systems normally consist of three main parts as well, which are the mechanical part, the electronic part and the central control part.

As different parts of the human body have different functions, the three parts of mobile robots are very much alike in terms of their typical functions. The mechanical part of the robot is just like a skeleton in the human body, it supports the full weight of the robot, carries the robot

when running and stopping. An excellent mechanical design makes the robot robust, i.e. when the robot contacts any obstacles, its body will handle the impact force, so that it will be safe from the crash. Moreover, the joints between the mechanical components connect the body of the robot as a whole; they provide the robot with good flexibility and buffer the forces that attempt to damage it.

Electronic parts are like the organs and blood vessels in the human body, i.e. the battery is like a human heart, it provides the energy to the whole body, and the wires are like the blood vessels, they transfer the energy to different places of the robot, making sure that every component can gain enough energy to function probably. The motors are like hands and feet, they consume energy to complete tasks.

The central control part is like the brain system, the central controller of the robot is like the brain of a human, it sends commands to its parts, tells the robot to complete tasks. It also collects data with sensors (e.g. distance sensors, cameras, compass, encoders, etc.) from the outer environment, and then responds to the obstacles.

A typical mobile robot system configuration can be demonstrated in Figure 4.1

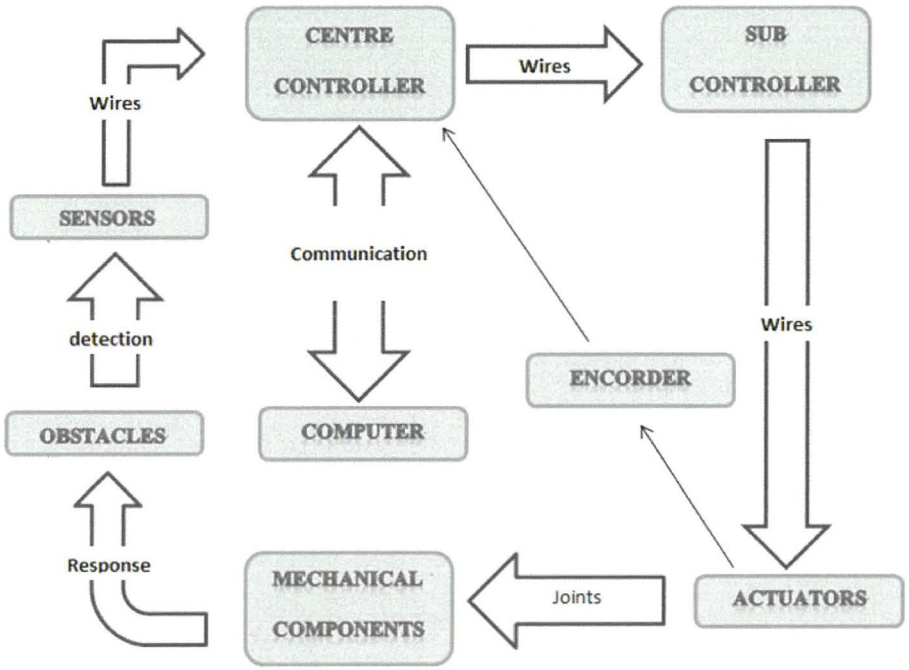


Figure 4.1 Mobile system configuration

As it can be seen from Figure 4.1, the central controller controls the behaviors of the robot; it makes commands for the whole system, and can also communicate with the computer, receiving the commands from computer by programming. Then the sub-controllers control the some specific components, and cooperate with the central controller. I.e. the motor driver controller controls the speed and rotating directions of the motors.

The actuators, in another word, motors are the executers. The motors carry out the commands and provide driving force to the mobile robot. Sometimes, encoders are used to feedback the information from the motors to the central controller, so that the central controller will be able to observe the motor's motion, and adjust the movements of the motors by sending new commands. Sensors are the devices that the robot uses to observe the environment; they collect data from the environment and send it to the central controller to process. Once they detected any obstacles, central controller will commands the motors to respond. All of the signal and data transformations are sent through the wires which are inside the body of the

robot (except the wireless transformation between the computer and the robot), so great wire connections are significant for the robot. The design of wire connections, circuits, and the sub controllers will be discussed in Chapter 5. The central controller and programming will be discussed in Chapter 6.

4.2 Mobile robot Mechanical system design

4.2.1 Overall view of the mobile robot's mechanical structure

To start the mechanical design, the first step is to have an overall view of the mobile robot's structure. Module design is applied in this research, and three omni-wheels are chosen for driving the robot, and the shape of the mobile robot's chassis is set to be an equilateral triangle. Figure 4.2 demonstrates the structure of the mobile robot.

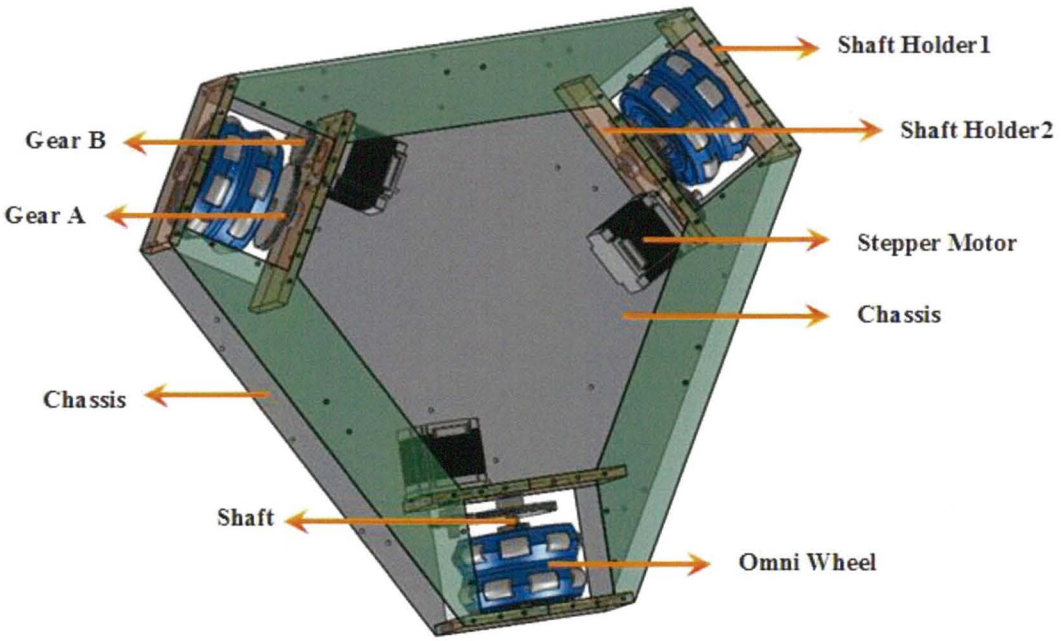


Figure 4.2: the overall view of the draft

As it can be seen from Figure 4.2 that, there are three shafts go through three omni-wheels, locking the wheels at their spots. Each shaft is fixed by two shaft holders (shaft holder 1 and 2) which are mounted on the chassis. Three stepper motors lay on the chassis as well and are fixed buy the shaft holder 2. The connection between the motor shaft and the wheel shaft is established by a gear mesh, whereas gear A in the mesh is mounted on the wheel shaft and gear B is on the motor shaft. The transformation of torque can be achieved through the gear mesh from the motors to the wheels. Modular design makes every part of the mobile robot base replaceable, the parts are in geometric symmetry. The three wheel shafts are made in the same module, so are the shaft holder 1 and 2, gear A and B, omni-wheel and stepper motor. The benefits of this design are obvious; if there is any part of the robot's base that is badly broken, it will be easy to replace the bad part. In addition, the size of the robot can be simply adjusted by changing the dimensions of the chassis in the drawings; this gives the robot a great potential becoming a test platform for different projects which require different robot base sizes. Last but not least, the mechanical design reserves the central place for the circuit boards which contain the central controllers, sub-controllers and other electronic components; if there is any project that would like to use this robot base as a test material but applying a different control theory with different controllers, the circuit boards can then be easily replaced by new modules with new circuit boards and controllers.

4.2.2 The mechanic selections

a. Material selection

The first thing that needs to be decided is the material of the mobile robot. The common materials used for robot design are: Aluminium, HDPE, Carbon Fibre, and Styrofoam (society of robots, 2005-2012).

Aluminium – It is widely accepted by people for general robot design, the reasons for its popularity is that it is strong, light, resistant to corrosion and affordable; very importantly, it is very easy to cut, drill, bend, and shape.

(High Density Polyethylene) HDPE – It is a type of plastic. It is very suitable for many robot designs since it is light, cheap, very easy to cut, drill, shape, resistance to corrosion; in addition, it has a very low thermal conductivity and a higher strength to weight ratios than metals.

Carbon Fibre – This is a relatively new material for robots. It has a very high force and weight ratio, in other words, when compared to other metals, it is likely to have more strength than metal if given the same weight. However, the drawback of this material is that it is very strong in longitudinal directions, but is much weaker in bending directions. This holds it back from many robot applications which require side force handling.

Styrofoam – It has a great degree of lightness, but it has a very low strength and high wear. This material is normally used for shaping rapid prototypes that make use of its great shaping flexibility. However, it is not preferable in building the robot base that will be used for mobile robot applications.

In this research, the mobile robot is likely to make contact with obstacles like walls, tables and chairs during the testing process. Thus the material that is selected for this project has to be strong enough to take the impacts. A lighter robot will definitely save more power from its battery, so heavy metals, such as steel, are not preferable. Another consideration is the economic feature; this feature is extremely important for good designs. Finally, aluminum is the material that was selected for the base of the robot, because of its strong strength, lightness, economic affordability, and availability. The alternative would be the HDPE, since it has great strength to weight ratios than metals, and is very economic.

b. Dimensions and Mass Calculations

After the shape and material of the mobile robot have been selected, the size of the robot and the dimensions of each component can now be set. When the dimensions have been set, the sum of the components' volume can be calculated, and since the density of aluminum is known as $\mathbf{2.7g * cm^{-3}}$, the total weight of the mobile robot can then be calculated. This will help the motor selection which will be discussed in the later section.

There are 6 parts that contribute to most of the weight on the mobile robot base. They are the chassis, the three pairs of shaft holder 1 and 2, three pieces of top cover plates which are used to hold the sensors, the omni-wheels, the batteries and the stepper motors.

Chassis – Figure 4.2 is an image of the mobile robot base created with the software "Solidworks". The shape of the chassis is an equilateral triangle with the three triangle tips removed, and it also has three rectangular sections next to each tip area cut out. Calculating the volume of the chassis can be done in a way that uses the volume of the big triangle with three little triangle shaped tips and three rectangular shaped slots subtracted from the total.

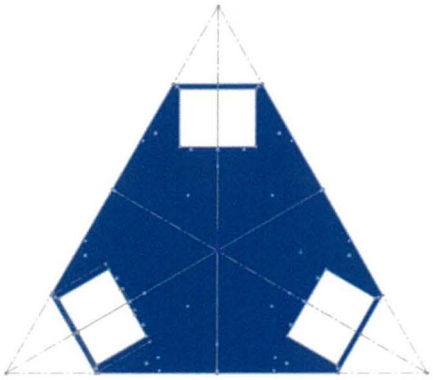


Figure 4.3: chassis of the robot

The dimensions of the chassis are:

The length of the side of the big triangle is 660mm

The length of the side of the little triangle is 140mm.

The length and width of the rectangle are 120 and 90mm.

The thickness of the chassis is 2mm.

Equation 4.1 calculation of the area of an equilateral triangle

$$A = \frac{\sqrt{3}}{4} a^2 \qquad (4.1)$$

Whereas in Equation 4.1, "A" represents the area and "a" represents the length of the side of an equilateral triangle.

Thus, the area of the big triangle equals $\frac{\sqrt{3}}{4} * 660^2 = 188620mm^2$.

The area of the little triangle equals $\frac{\sqrt{3}}{4} * 140^2 = 8487mm^2$.

Equation 4.2 calculation of the area of a rectangle

$$A = L * W \qquad (4.2)$$

Whereas in Equation 4.2, "A" represents the area, "L" and "W" are the Length and width of the rectangle.

Thus, the area of the rectangle equals $\mathbf{120 * 90 = 10800mm^2}$.

The total area of the chassis equals

$$A_{big\ triangle} - (A_{little\ triangle} * 3) - (A_{square} * 3)$$

$$= 188620 - (8487 * 3) - (10800 * 3)$$

$$= 130759mm^2$$

Equation 4.3 calculation of the volume of a cylinder

$$V = A * H \qquad (4.3)$$

Whereas in Equation 4.3, "V" represents the volume of a cylinder, "A" represents the area, and "H" the thickness.

Thus the volume of the chassis equals $\mathbf{130759 * 2 = 261518mm^3 = 261.5cm^3}$

Shaft holder – Figure 4.4 is an image of the two cuboid shaft holders.

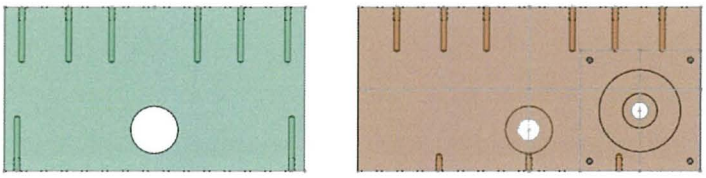


Figure4.4: shaft holder1 (left) &2(right)

The dimensions of the shaft holders:

Shaft holder 1: Length - 140mm; Width - 76mm and Thickness – 8mm.

Shaft holder 2: Length – 140mm; Width - 76mm and Thickness – 8mm.

According to the equation of the volume of a cube, the volume of the two shaft holders can be calculated.

Equation 4.4 calculation of the volume of a cube:

$$\boldsymbol{V = L * W * H} \qquad \textbf{(4.4)}$$

Whereas in Equation 4.4, "V" represents the volume of a cube, "L" the length, "W" the width and "H" the thickness.

Thus the volume of the shaft holder 1 equals $\mathbf{140 * 76 * 8 = 85120mm^3 = 85.1cm^3.}$

Thus the volume of the shaft holder 2 equals $\mathbf{140 * 76 * 8 = 85120mm^3 = 85.1cm^3.}$

Top cover plate – Figure 4.5 is an image of the top cover plate on the mobile robot.

Figure 4.5: top cover plate

In Figure 4.5, it can be seen that the main part of the top cover plate is a trapezium shape. The 2 extruded parts on both sides of the trapezium can be neglected since they contribute little to the volume. Therefore, the volume of the trapezium part is assumed to be volume of the cover plate.

The dimension of the top cover plate:

Length of the top side - 210mm; Length of the bottom side – 380mm; height – 65mm

Thickness – 2mm

Equation 4.5 calculation of the volume of a trapezium:

$$\boldsymbol{A = (a + b)/2 * h} \qquad \textbf{(4.5)}$$

Whereas in Equation 4.5, "A" represents the area of the trapezium, "a" and "b" are the two parallel sides of the trapezium and "h" is the height.

Thus the area of the top covers plate equals $\boldsymbol{(210 + 380)/2 * 65 = 38350mm^2}$.

According to Equation 4.4, the volume of the top cover plate equals

$$\boldsymbol{38350 * 2 = 76700mm^3 = 76.7cm^3}$$

Calculation of the total mass of the mobile robot base

The volume of the mobile robot base is the sum every component's volume. There are 1 piece of chassis; 3 pieces of each shaft holder 1 and 2 and 3 pieces of the top cover plate to form the base structure of the mobile robot. Therefore, the volume of the mobile robot base equals

$$V_{chassis} + (Vs_{haft\ holder1} * 3) + (V_{shaft\ holder2} * 3) + (V_{cover\ plate} * 3)$$

$$= 261.5 + (86.1 * 3) + (86.1 * 3) + (76.7 * 3)$$

$$= 1008.2cm^3$$

Whereas $V_{chassis}$ represents the volume of the chassis, $V_{shaft\ holder\ 1}$ represents the volume of the shaft holder 1, and $V_{shaft\ holder\ 2}$ represents the volume of the shaft holder 2, same as Vtop cover, it represents the volume of the top cover.

Equation 4.6 calculation of mass

$$M = V * D \qquad (4.6)$$

Whereas in Equation 4.6, "M" represents mass, "D" represents density and "V" represents volume.

Since the density of Aluminum is known as $2.7g * cm^{-3}$.

Thus the mass of the mobile robot base equals $1008.2 * 2.7 = 2722.14g = 2.72kg$.

Omni-wheel – Figure 4.6 is an image of an omni-wheel that is selected for this project.

Figure 4.6 image of an Omni-wheel

Data of the omni-wheel: Outer diameter - **101.6mm** ;Weight- **0.31kg**

Battery – Figure 4.7 is an image of a lead-acid rechargeable battery that is selected for this project.

Figure 4.7: image of a lead-acid battery (RS Company)

The battery weighs 0.99kg. Two batteries of the same kind are chosen for this project.

To sum up, the weight of the robot is the sum of the weight of the mobile robot base, 3 omni-wheels and 2 batteries, which equals $(\mathbf{2.72}) + (\mathbf{0.31} * \mathbf{3}) + (\mathbf{0.99} * \mathbf{2}) = \mathbf{5.63}\boldsymbol{kg}$.

(Some other parts which contribute little to the mass are neglected)

4.3 Motor selection

Once the weight of the robot has been calculated, the torque that is required from the motor to drive the mobile robot can also be calculated.

In this project, three pairs of gear that are in mesh with a ratio 1: 2 (50 teeth: 100 teeth) are chosen to transmit the torque from the motor shaft to the wheel shaft. The smaller Gear (50 teeth) is mounted on the motor shaft, and bigger Gear (100 teeth) is mounted on the wheel shaft. Figure 4.8 presents the two gears and their mesh.

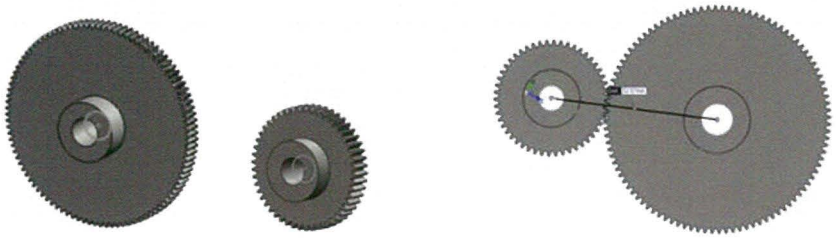


Figure4.8: gear mesh

In Figure 4.8 (left), the bigger gear is named Gear A and the smaller Gear B. The outer diameter of Gear A is 71mm and the outer diameter of Gear B is 36mm.

In Figure 4.7(right), the gear mesh is 50:100 = 1:2, and the center to center distance is 52.57mm.

Speed of the robot

Stepper motors output different torques at different speeds. Therefore, the speed of the mobile robot' motion needs to be decided first. Because this mobile robot is a test module, rather than a racing vehicle, the normal speed of the robot is equal to the rotational speed of the wheel, which is set to be one revolution per second.

Since the diameter of the omni-wheel is 101.6mm; every full rotation of the wheel will make a displacement of $\boldsymbol{\pi * 101.6 = 319mm = 31.9cm}$, so the linear speed of the mobile robot is 31.9 cm/s.

Torque calculation

The weight of the mobile robot is equal to the force that the motor is required to provide. The stepper motors first generate torque and transmit it to the omni-wheels, and then the wheels

drive the mobile robot to move. Knowing the speed of the wheel and the weight of the mobile robot, the torque that is required from the wheel can be calculated.

Equation 4.7 calculation of torque

$$\boldsymbol{T = F * D} \qquad \textbf{(4.7)}$$

Whereas in Equation 4.7, "T" represents torque, "F" represents force and "D" represents the distance between the center of the wheel and contact point where the force occurs.

The force is equal to the weight of the robot, which is **5.63kg**, since there are other components and the motors' weight added on later, the weight of the robot can be roughly estimated as **8kg** in total. The distance (D) is the radius of the wheel, which is $\boldsymbol{5.08cm}$.

Thus the torque which is required from the 3 wheels equals $\boldsymbol{8kg * 5.08cm = 40.6kg.cm}$. There are 3 motors on the robot chassis that generate torque to drive the wheels, and there is a 1:2 ratio gear mesh between each wheel shaft and motor shaft, so that the torque that is required from each motor equals $\boldsymbol{40.6kg.cm * 1/3(3\ motors) * 1/2(ratio) = 6.77kg.cm}$. The stepper motor stepper motor TNZ00335 is selected for this project, Figure 4.9 is an image of the chosen stepper motor.

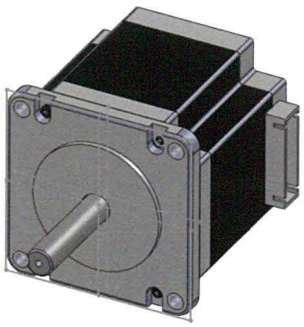


Figure 4.9: image of stepper motor TNZ00335

This stepper motor is able to output a 7.2kg.cm torque at a speed of 120 revolutions per minutes (RPM). It is a two phase stepper motor. Every step rotates 1.8° and its step angle accuracy is 5%. Each phase of the stepper motor can take 3A current, and phase resistance is 0.74omh. This project chooses stepper motor over DC motors and servo motors, because that stepper motors are more accurate in their rotational motions. Every pulse signal that is given to the stepper motor rotates it by one step, thus by measuring the number of the pulses, the number of the stepper motor's revolution can be calculated. On the other hand, a DC motor requires an encoder to obtain the speed of the motor, which makes the design a bit complicated.

4.4 The top layer of the robot

Sharp sensor - There will be 3 sharp sensors on the robot for detecting obstacles and sending signals to the central controller. Figure 4.10 is an image of the selected sharp sensor.

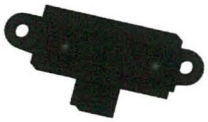


Figure 4.10: sharp sensor

There will be a sensor mounted on the head of the mobile robot which is a servo motor construction. The sensor then will be able to rotate scan in a certain range for obstacles. The other sensors are put on both sides of the robot, so that they can help to scan the obstacles which are out of the head sensor's reach. The sensors will be discussed specifically in Chapter 5.

Servo motor – a servo motor is used to rotate a sharp sensor to scan obstacles a certain range. This servo motor construction is assumed as the head of the robot. Figure of the servo motor is shown in Figure 4.11 is an image of the selected motor.

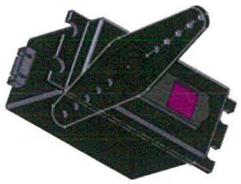


Figure 4.11: servo motor

Servo holders – the servo construction includes three holders that hold the servo motor on the mobile robot. Figure 4.12 is an image of the three holders.

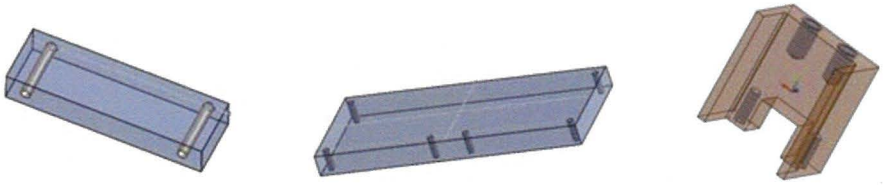


Figure 4.12: Three servo holders

The side protection of the robot – There are 3 pieces of Aluminum sheet on each side of the triangle shaped base. They protect the mobile robot when it hits obstacles. There are two square spots cut off from the protection so that there will be air flow inside the mobile robot, preventing the motors or the batteries from getting too hot. Figure 4.13 is an image of the protection.

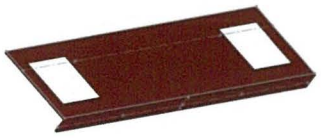


Figure 4.13: The side protection of the robot

Fan holder – A fan is placed on top of the mobile robot to cool the motors and batteries down when they are getting too hot. A fan holder is designed to stabilize the fan. Figure 4.14 is an image of the fan holder.

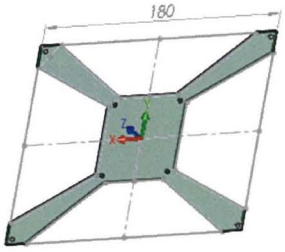


Figure 4.14: Fan holder

Final View of the robot – Figure 4.15 is an image of the final version of the mobile robot.

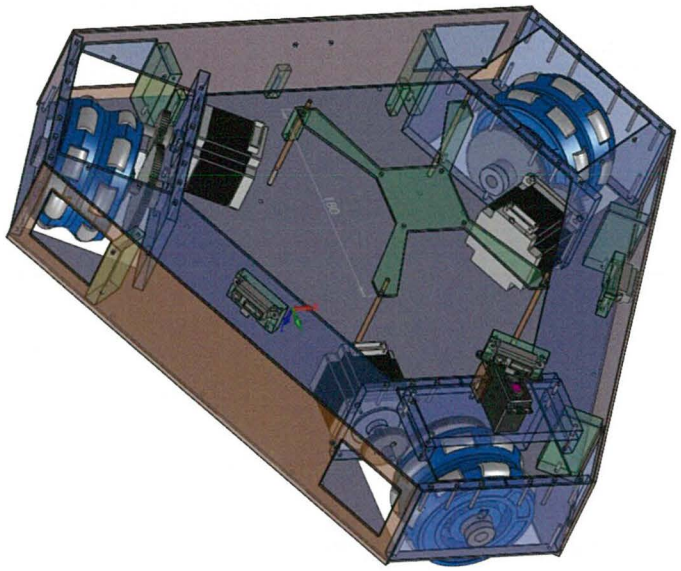


Figure 4.15: the final view of the robot

In Figure 4.15, every mechanical component is assembled on the chassis of the mobile robot. It can be seen that, the central area of the chassis is reserved for the circuit board. The stepper motors and sharp sensors can be easily connected to any circuit boards with wires. When machining these components, due to the great figure of modular design, once a part is designed, its design applies to all the parts which are in the same module.

CHAPTER 5 Mobile Robot Electric and Electronic System Design

This chapter discusses the electric and electronic design of the robot. Three sections will be mentioned which are the power regulation, the Microcontroller, and the sensors and actuators. The power regulation circuit gives the Microcontroller a stable constant voltage to operate, so that the microcontroller is able to give commands to the actuators. The microcontroller also receives signals from the sensors, and then modifies the commands which were given to the actuators.

5.1 Electric and electronic system configuration

The electronic design of the mobile robot can be diveded into 4 sections, the power regulation, the mocrocontroller, and the sensors and actuators. Each of them has their individual functons and communicate with other sections. The mobile robot then can gain stable power from the batteries, accomplish tasks, receive signals from the sensors and response to the obstacles.

In the electric and electronic design configeration, the power supply module contains the batteries which supply power to the mobile robot. One battery supplies the 3 stepper motors, the other one supplies the microcontroller and other electronic components. The microcontroller requires stable votage to function, thus a power regulation circuit is designed to stablize the output votage from the batteries. The regulated votage is 5V for microcontroller and other electronic components. Then the microcontroller sends commands to the three motor drivers which are named as motor driver A, B and C. The microcontrollor communicates with each motor driver through 3 control lines which are the Pulse-width modulation (PWM) control line, Directional (DIR) control line and Enable (EN) control line.

There are 4 output lines on each motor driver, the 4 outputs go into the each stepper motor, power on the different phases of the motors' coils in a certain order to control the motor' motion. The Sharp sensors will collect the data from the obstacles and send signals back to the microcontroller. The microcontroller will respond to the obstacles by modifying the commands which are sent to the motor drivers. The microcontroller is programmed by a computer. Figure 5.1 shows the configeration of the electric and electronic design.

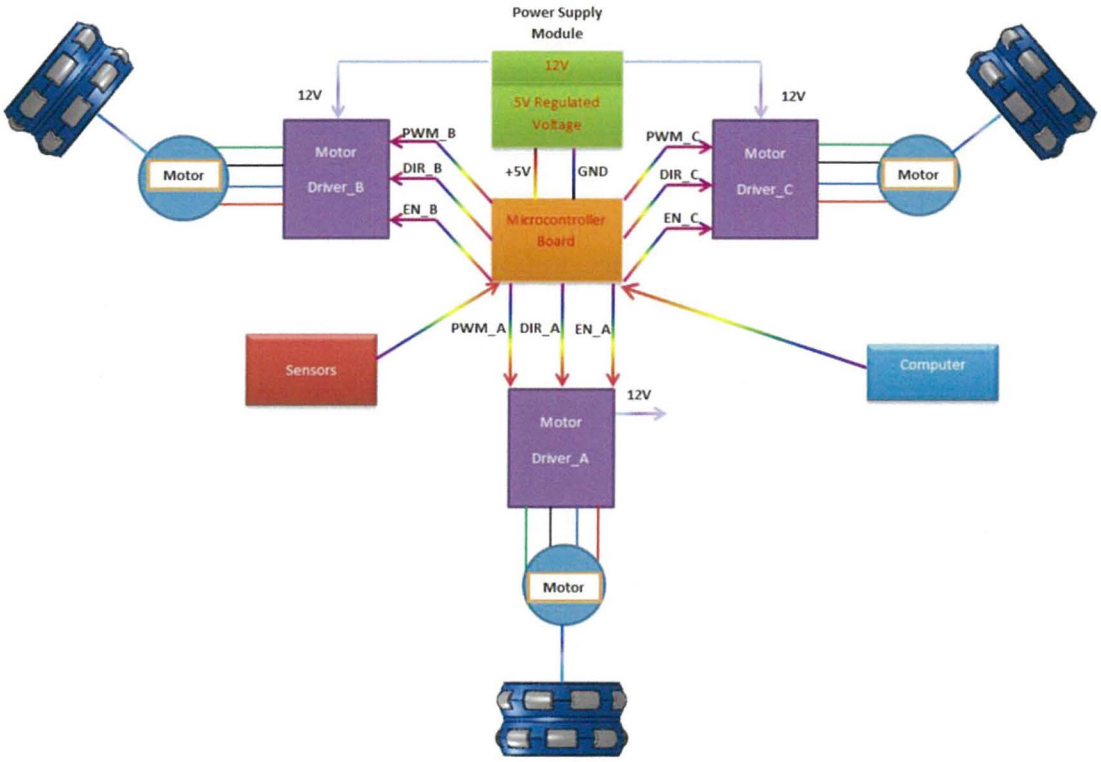


Figure 5.1 Electronic design configuration of the mobile robot

5.2 power supply circuits

The power supply module supports all the behavior of the mobile robot. Without the power supply, the mobile robot will not be able to move, compute and respond. The power supply

module consists of two parts, which are the positive 12V output for the stepper motors and the positive 5V output for the microcontroller, the servo motor and the sensors.

5.2.1 +5V voltage regulation circuit

Microcontroller operations require a constant voltage, if the voltage is not stable, the microcontroller will not be able to function properly. The timing sequences, data transmission and register statuses are all affected by the supply voltage. As a result, stabilizing the input voltage at a constant value for the microcontroller is necessary.

The microcontroller requires a positive 5V input, While the battery output positive DC 12V. Therefore a voltage regulator is required to regulate the positive 12 V to 5V. The voltage regulator LM340 is selected in this project. Figure 5.2 is an image of the regulator LM340.

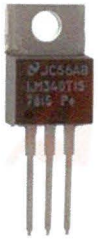


Figure 5.2 Voltage regulator LM340

According to the datasheet of the LM340 (available in Appendix A.1), the voltage regulation circuit can be built as shown in Figure 5.3.

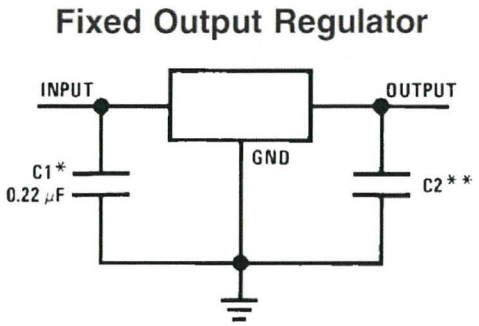


Figure 5.3 Fixed output regulation circuit

LM340 has 3 legs, one is for input, one is for output and the last one is for ground (GND). The input leg connects to the battery, takes 12V into the regulator, and then the output leg gives out a constant 5V to the microcontroller. The two capacitors are decoupling capacitors.

A schematic of the regulation circuit was drawn with the software Altium Designer. Figure 5.4 presents the voltage regulation circuit.

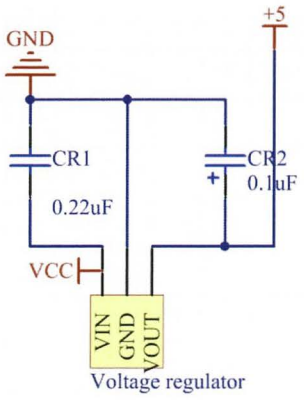


Figure 5.4 Schematic of the voltage regulation circuit

5.2.2 +12V supply

The +12V is directly pulled from the batteries. One consideration of the power circuit is that if there was a short circuit on the circuit board, some chips might be easily fried; therefore, a

protection fuse is required to cut off the power when the short circuit occurs. However, when the mobile robot encounters a rough surface, the motor will draw more power from the batteries, in the form of current. This unexpected big current can sometimes be twice as much as the normal motor operating current. Therefore, the protection fuse should only cut off the short circuit current but the current drawn by the motors. Figure 5.5 shows the schematic of the fuse protection circuit.

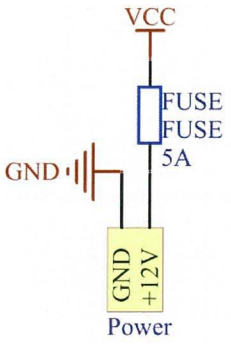


Figure 5.5 Fuse protection circuit

Two capacitors are needed for decoupling for the power circuit. The noise that is mixed in the power line is shunted through the two capacitors. The typical capacitor values for the decoupling circuit are 0.1 μF and 470μF, one is relatively high, which filters out the low frequency noise and the other is relatively low, which filters out the high frequency noise. Figure 5.6 demonstrates the schematic of the decoupling circuit.

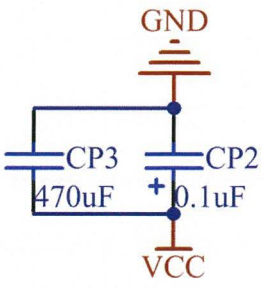


Figure 5.6 decoupling circuit for the power circuit

5.3 Microcontroller

A microcontroller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory is also often included on chips, as well as a typically small amount of random access memory (RAM). Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications. Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, remote controls, office machines, power tools, toys and other embedded systems. They reduce the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices. Therefore, microcontroller is preferable in this project.

The microcontroller ATmega32 is chosen in this project. Because according to its datasheet (partial datasheet is presented in Appendix A.2), ATmega32 is a high-performance and low-power 8 bit microcontroller. Its CPU frequency can be up to 16MHz. ATmega32 has one 16-bit and two 8-bit timers, four Pulse width modulation (PWM) channels, 8 channels 10-bit Analogue to Digital Convertors (ADC). Moreover, ATmega32 has great interrupt and Input and Output (IO) sources, which are needed for this project. Figure 5.7 is an image of ATmega32.

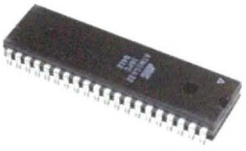


Figure 5.7 image of the microcontroller Atmega32

5.3.1 Pulse Width Modulation (PWM) generation

PWM stands for Pulse Width Modulation. It generates a set of pulses whose width (duration) can be altered. In motor control system, the speed of a motor can be regulated by changing the supply voltage from relatively low to high. At the high voltage, the motor might run at full speed; and at the low voltage, the motor rotates slowly. However, in many systems, the motors are only given a constant voltage. So the alternative is to power on and off the motor with a set of pulses. Then the motors will receive small kicks and run smoothly because of the inertia of the rotors. PWM can just do the job. The frequency of the pulses can be adjusted through the PWM; this will change the motor's speed. The higher the frequency is set to be, the faster the motor rotates. The width of the pulses controls the motor's output torque. The longer the width, the greater the torque is. The frequency of the pulses is called the frequency of the PWM, and the percentage of the width of the pulses on high is called the duty cycle.

Microcontroller ATmega 32 can generate PWM signals with simple approach. An advantage of using the microcontroller to generate the PWM signal for the motors is that once it has been set up correctly, the PWM signal will continue to be generated automatically in the background. The microcontroller can change the frequency and duty cycle of the PWM easily, in other words, the microcontroller is able to controls the performance of the motor with simple approach.

The ATmega32 uses various timers for producing PWM. Different timers have different numbers of channels to generate PWM. In ATmega32, timer0 and timer2 are 8-bit timers, and timer1 is a 16-bit timer. Using a greater timer can generate a PWM with higher resolutions. Therefore, the timer1 is selected for producing the PWM for the motors. There are different

ways to create a PWM with timer1, the details will be presented in the PWM generating section in Chapter 6.

5.3.2 ADC

ATmega32 needs to collect information from the sharp sensors, so it can detect the obstacles and make responses. However, for most sensors, when they detect an obstacle, they normally send an analogue signal (analogue current or voltage) to the microcontroller. It is known that Microcontrollers are great in dealing with the digital values, but it cannot process the analogue signal directly. In this case, a conversion from the analogue value to digital value is required. Atmega32 has 8-channel 10-bit ADC, it is able to receive 8 different analogue inputs at the same time, and convert them to digital for processing. The details of ADC functions are presented in the ADC section in Chapter 6.

5.3.3 Crystal Oscillator

On ATmega32, the two pins XTAL1 and XTAL2 are input and output pins for the clock system. According to the datasheet of ATmega32 (partial datasheet is presented in Appendix A.3), an inverting amplifier circuit is shown in Figure 5.8.

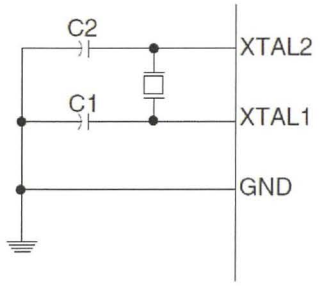


Figure 5.8 External Clock circuit

A 16MHz crystal is chosen for this project to achieve the maximum data processing speed. Two capacitors with the value of 10 pF are selected to form the oscillator clock circuit. This external clock circuit will provide a constant 16MHz clock signal for the microcontroller, it is the base clock signal for the timers, counters and the Arithmetic Logic Unit (ALU). Figure 5.9 shows the Crystal Oscillator Operating Modes and capacitor selections.

CKOPT	CKSEL3..1	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
1	101[1]	0.4 - 0.9	–
1	110	0.9 - 3.0	12 - 22
1	111	3.0 - 8.0	12 - 22
0	101, 110, 111	1.0 ≤	12 - 22

Figure 5.9 Crystal Oscillator Operating Modes and capacitor selections

Figure 5.10 presents the schematic drawing of the external clock circuit.

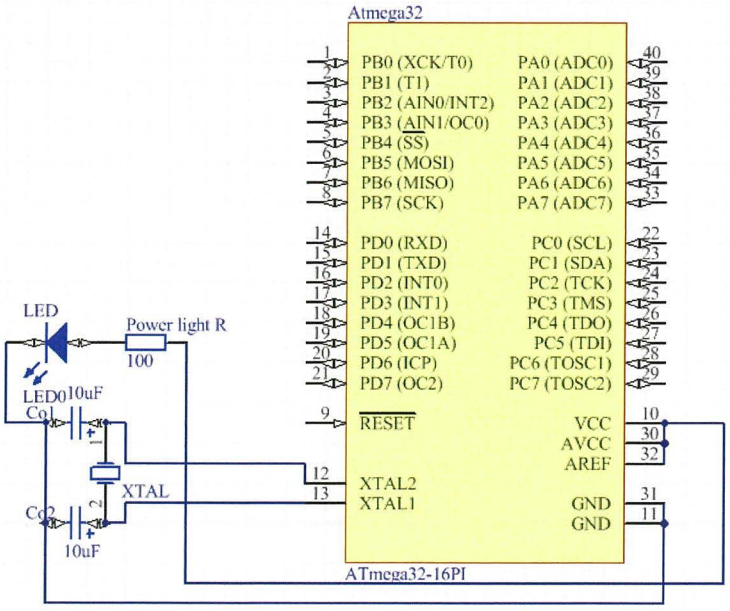


Figure 5.10 External clock circuit

5.4 Sensors and motor drivers

5.4.1 Sensors

Three sharp sensors "GP2Y0A21YK0F" are used to scan the obstacles for the mobile robot. Two of the sensors are mounted on both sides of the robot's head. There is a 120 degree angle in between these two sensors. The last sensor is mounted on the head of the mobile robot, which is a servo motor construction. The servo motor will rotate in a range of 90 degree, thus the sensor on top of the servo motor is able to scan a 90 degree fan-shaped region in front of the mobile robot. The two side sensors detect the obstacles which are out of the head sensor's reach. Figure 5.11 is an image of the sharp sensor.

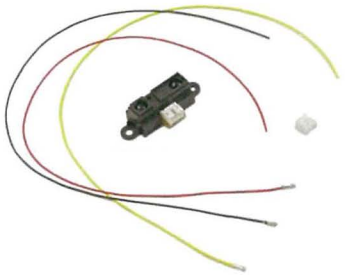


Figure 5.11 image of the Sharp sensor GP2Y0A21YK0F

The configuration of the "sharp GP2Y0A21YK0F" is provided in its data sheet. Figure 5.12 demonstrates partial configuration of the sharp sensor.

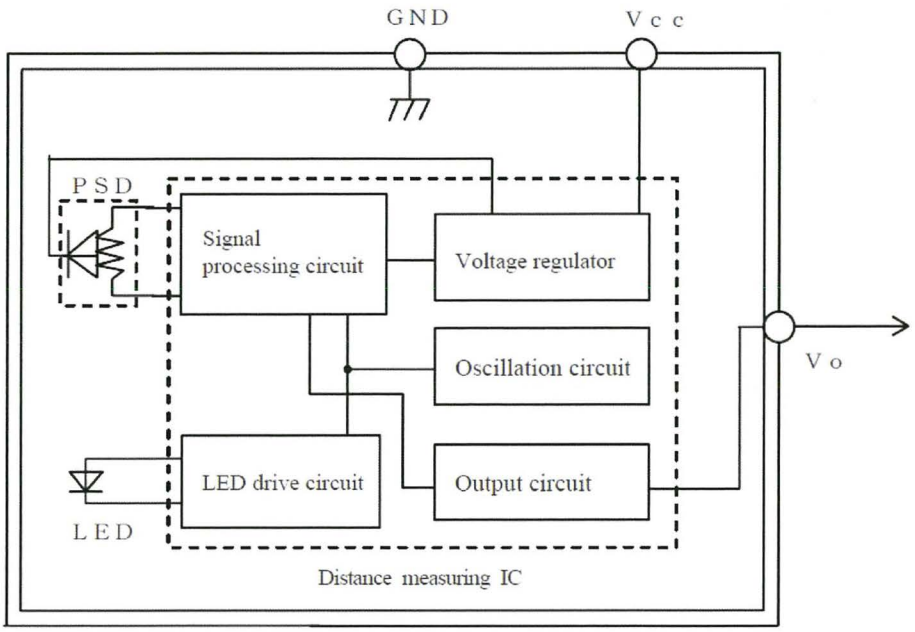


Figure 5.12 Sharp GP2Y0A21YK0F configurations

This sharp sensor has an emitter and a receiver. When it is operating, the emitter shoots infrared light straight forward, and if there are any obstacles that blocked the light, the light will reflect back. The receiver will then pick up the light, feedback various signals to the sensor according the light intensity. The sensor will output analogue voltage values to represent the distances between the obstacles and the sensor. Higher voltage values represent shorter distances (stronger light intensity); in contrast, lower voltages mean longer distances (weaker light intensity). Figure 5.13 shows the inverse output voltage versus distance.

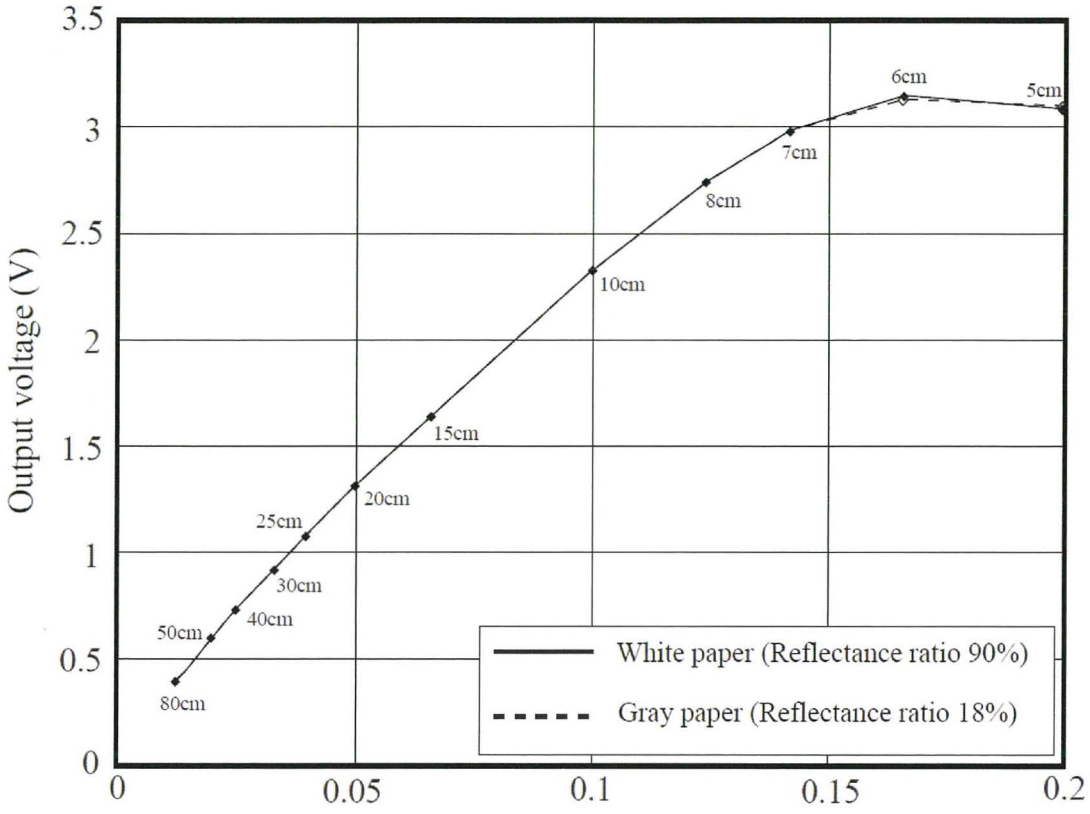


Figure 5.13 Inverse output value vs Distance

Figure 5.13 demonstrates that the inversed output voltage has good features. Especially from the range 8cm to 30cm, the figure is almost linear. Partial datasheet of the Sharp sensor GP2Y0A21YK0F is presented in appendix A.4.

5.4.2 Stepper motor drivers

Motor drivers are widely used in modern motor control systems. The motor drivers reduce the complexity of the motor control, protect the motor from drawing hazardous current, and cooperate with the microcontroller to optimize the motor's performance. One popular stepper motor driver circuit is formed by the combination of the stepper motor driver chip L298 and the stepper motor controller chip L297. Figure 5.14 is an image of the two chips.

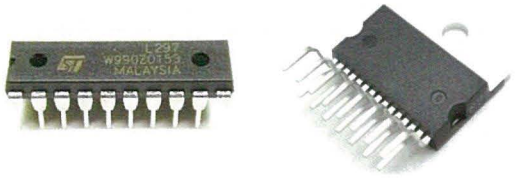


Figure 5.14 L297 (left) & L298 (right)

The L297 (stepper motor controller chip) receives control signals from the microcontroller, and provides all the necessary drive signals for L298 (stepper motor driver chip), such as motor's rotating directions, enable and reset and mode selection. The mode selection will be discussed in the motor driving section in Chapter 6. The driver circuit is shown in Figure 5.15.

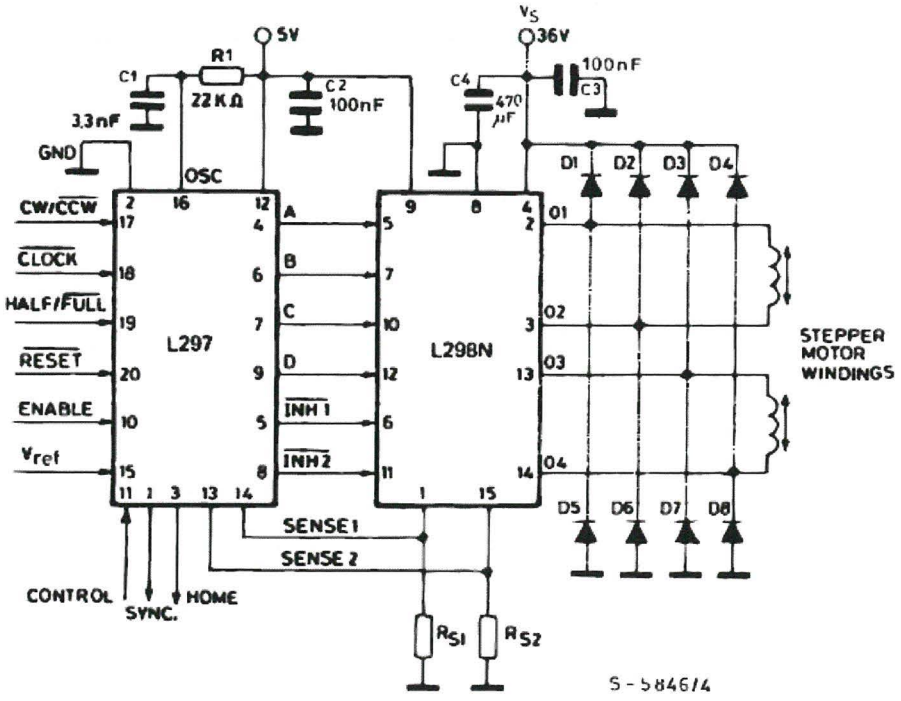


Figure 5.15 Stepper motor driver circuit L297&L298.

In Figure 5.15, L297 is the chip connected with the microcontroller. It receives the signals from the ATmega32 and transfers them into L298. L298 outputs 4 signals to the 4 phases of the motor, which switches on and off the 4 phases in sequence. R1 the 22kΩ resistor and C1 the 3.3 nF capacitor form a RC Resonance circuit, which provides the clock signal for the driver circuit to be its chopper rate. C2 and C3 are all 100 nF capacitors and C4 is a 470 μF capacitor, these three capacitors help the circuit to filter out the unexpected noise signals. Sense resistor 1 and 2 are only used for the chopper circuit, they sense the current that is drawn into the motors and L298, and cuts off the circuit if the current is over the preset value. In this project, the driver circuit works on the driving mode, not the chopping mode, so the sense resistors will not be used. Thus pin 1 and 15 on L298 can be connected directly to ground. The datasheet and pin configurations of the stepper driver L297& L298 are presented in appendix A.5.

L297 & L298A connections to the microcontroller

There are three pairs of stepper motor controller and driver chips L297 and L298 that are connected to the microcontroller. The three pairs have similar connections. Therefore, choosing L297 and L298 A as an example, their pin connections to the microcontroller are:

Pin 2(GND) – Ground connection.

This pin connects to the GND pin on the microcontroller.

Pin 10(Enable) – Chip enable input.

This pin connects to the microcontroller pin 16. When pin 16 send a signal high to the enable pin, the driver chip is activated.

Pin 12(Vs) – 5V supply input.

This pin connects to the VCC pin on the microcontroller.

Pin 15(Vref) – Reference voltage for the chopper circuit. A voltage applied to this pin determines the peak load current.

This pin needs to be connected even the chopper mode is not selected. L297 will not function properly if pin 15 is not connected. Pin 15 requires an input analogue voltage value as the driver chip's voltage reference. Since the microcontroller does not generate analogue signals, a digital to analogue convertor (DAC) is needed. The DAC chip pcf8591 is chosen to convert the digital signals from the microcontroller output to analogue signals, and transmit the signals to pin 15 on L297.

Pin 16(OSC) –oscillation pin.

The three OSC pin on each L297 is connected to each other, so that the base clock of the 3 stepper motor drivers are synchronized.

Pin 17(CW/_CCW) –Clockwise/counter-clockwise direction control input.

This pin is connected to the microcontroller pin 17. A high signal from the microcontroller will rotate the motor in clockwise. Vice versa, a signal low from the microcontroller will rotate the motor in counter-clockwise.

Pin 18(CLOCK) – Step clock. An active low pulse on this input advances the motor one increment. The step occurs on the rising edge of this signal.

The clock pin receives the PWM from the microcontroller pin 18.

Pin 19(HALF/FULL) – Half/full step select input.

This pin connects to the microcontroller pin 1. When high selects half step operation, when low selects full step operation. The mode selection will be discussed in the motor driving mode selection section in Chapter 6.

Pin 20(RESET) –Reset input.

Reset pin is connected to the Enable pin, so every time when the chip is enabled, it automatically reset.

The schematic of the motor driver A is presented in Figure 5.16.

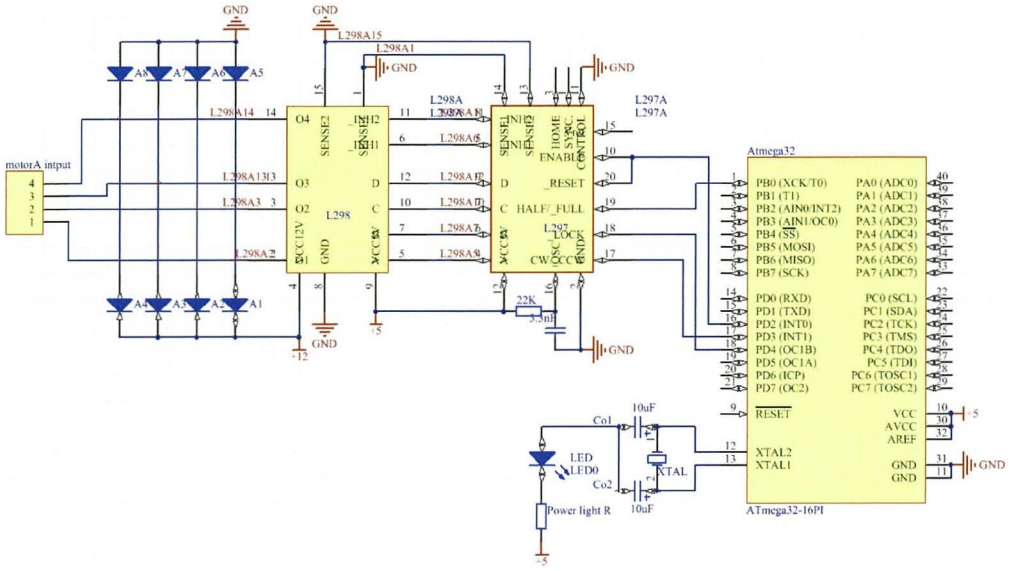


Figure 5.16 Schematic of motor driver A

5.5 Schematic and PCB of the mobile robot electronic system

The schematic of the mobile robot electronic system design is presented in Figure 5.17.

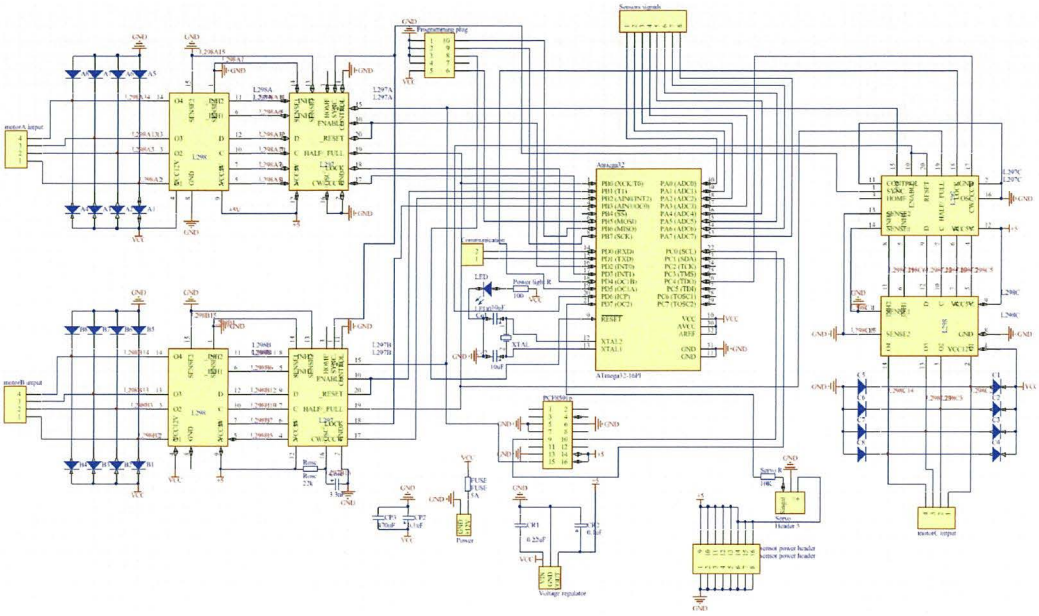


Figure 5.17 Schematic of the mobile robot electronic system design

In Figure 5.17, three stepper motor drivers are connected with the ATmega32. They receive the signals from the microcontroller, to control the motors' performances. The oscillator circuit generates the clock signal for the ATmega32. The positive 12 V power is connected to the motors' leads, providing power for the motors. The 12V also connect to the voltage regulator to regulate the voltage as a constant +5V, providing power for the ATmega32. The clearer schematic image will be presented in Appendix B.1.

The software Altium Designer can create a PCB drawing based on the schematic. The PCB of the mobile robot electronic system design is shown in Figure 5.18.

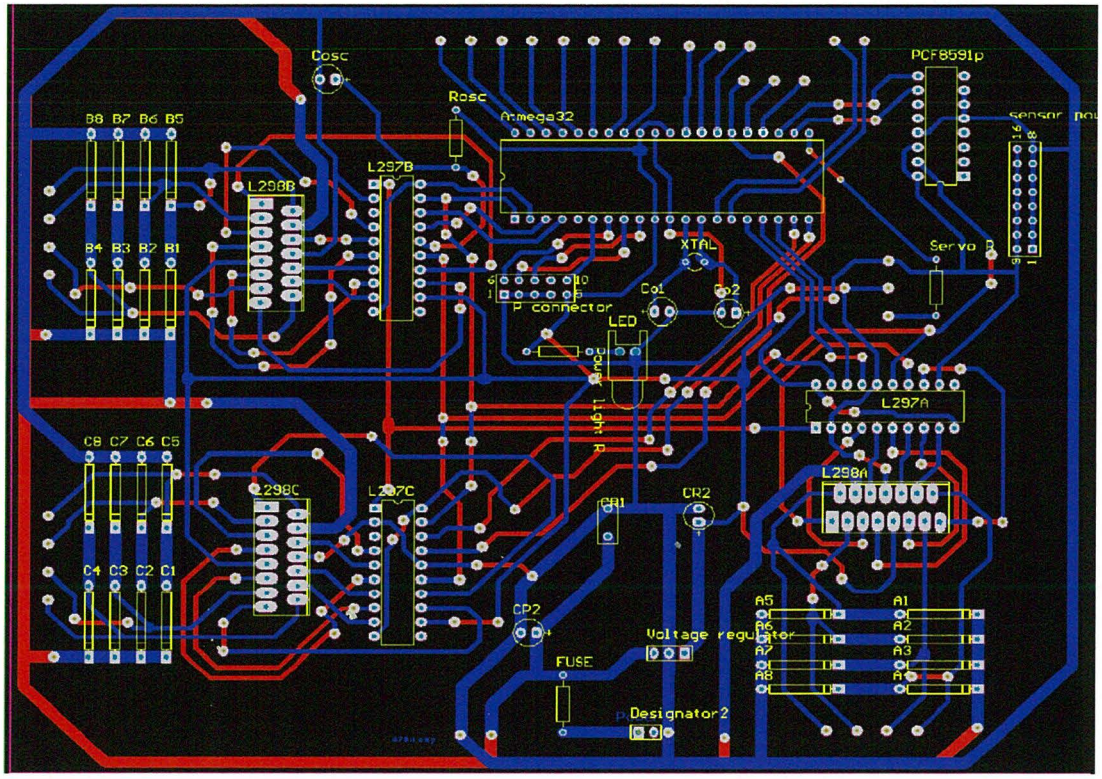


Figure 5.18 PCB of the mobile robot electronic system design

The PCB has two layers, which are the top layer and bottom layer. The connections on the top layer are colored as blue, and the ones on the bottom layer are colored as red.

CHAPTER 6 Microcontroller and Programming for Robot Control

This chapter discusses the microcontroller functions and C programming for the mobile robot control system. A good programming helps the mobile robot to function stably. In order to design an effective and reliable programming system, the study of the functions in microcontroller Atmega32 is necessary. There are 4 important basic functions in microcontroller Atmega32 that are used for this project, which are the Timer/Counter, PWM, I2C and ADC. The Timers is mainly used to generate PWM and delay functions. The PWM signal is for the stepper motor control. I2C helps the microcontroller to communicate with its peripheral devices. ADC function converts the analogue signals to digital for the microcontroller. In this chapter, the introduction and applications of these functions are presented.

6.1 Robot microcontroller based Control System

The microcontroller based control system consists of 5 parts; they are the timing sequence, Motor control, Sensors Data Receiving, Serial communication and the Priority. With the timers in the Atmega32, the programming system is able to flow in a time sequence. The timers help with the PWM generations and the interrupt floats. The PWM function generates PWM signals to control the stepper motors with help of the stepper motor drivers. The ADC functions help the microcontroller to read the data from the sensors. It is generally known that most sensors transfer analogue signals to the microcontroller and microcontroller can only process digital signals. Thus, ADC function converts the analogue signals to digital which makes the data readable to the microcontroller. I2C is the communication tool between the microcontroller and its peripheral devices. The interrupts work with the timers, ADC to

decide the priorities of different functions in programming. Figure 6.1 presents the configuration of the microcontroller programming system.

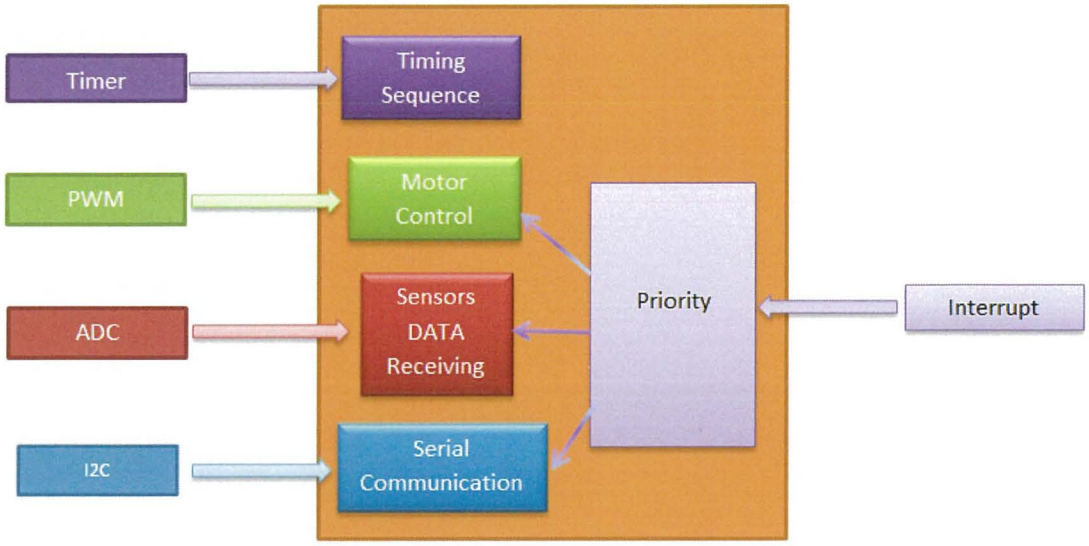


Figure 6-1 Robot software control system

6.2 Timers and timing sequence

Timers in Atmega32

There are three timers in Atmega32, which are the Timer0, Timer1, and the Timer2. Timer0 and Timer2 are 8-bit timers and Timer1 is a 16-bit timer. Since Timer1 has a higher resolution than the other two timers, in the following sections, Timer1 and its test program will be chosen as examples to interpret the timer functions.

Resisters in Timer1

The timers are used everywhere in the microcontroller programming. From the basic clock signal to advanced PWM generation functions, the timers contribute greatly. Timer1, a 16-bit timer in microcontroller ATmega 32, is selected to be the PWM generation tool in this project. Firstly there are some resisters in timer 1 that need to be set to active timer 1's functions.

The first two resisters are: Time /Counter1 Control Resister A (TCCR1A) and Time /Counter1 Control Resister B (TCCR1B). These two resisters are 8-bit resisters. In TCCR1A the Bit 7 and Bit 6 are the Compare Output Mode 1 and 0 for Compare Unit A. (COM1A1 and COM1A0), and Bit 7: Bit 6 are the Compare Out Mode1 and 0 for Compare Unit B (COM1B1 and COM1B0). The 4 bits COM1A1, COM1A0, COM1B1 and COM1B0 work together to set the compare mode of the timer1. The other two bits in TCCR1A that are used in this project are the Bit 1 and Bit 0. They are the Waveform Generation Mode (WGM), Bit 1 is WGM11 and Bit 0 is WGM10. In TCCR1B, the Bit 4 and Bit 3 are the WGM as well, Bit 1 is WGM13 and Bit 2 is WGM12. The four bits WGM13, WGM12, WGM11 and WGM10 work together to set the timer 1 PWM generation mode. PWM generation mode will be presented in the PWM generation section.

The Bit 2, Bit 1 and Bit0 in TCCR1B are the Clock Select (CS) Bits. Bit2, Bit1 and Bit0 are the CS12, CS11 and CS10. These three bits set the prescaler for the timer 1. Table 6.1 shows the prescaler value.

Table 48. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Table 6.1 Clock Select Bit Description

There are three more resisters that are used in this project. These are the Timer/Counter 1 (TCNT1), the Output compare resister1A (OCR1A) and the Output compare resister1B (OCR1B). TCNT1 is used to store the current value of the timer1; it has two parts, which are the TCNT1 High and TCNT1 Low. OCR1A and OCR1B can be set as the compare value. When timer1 start time counting, TCNT1 will increase. The OCR1A and OCR1B keep comparing their values to TCNT1. Once TCNT1 is equal to OCR1A or OCR1B, the Compare Output Mode bits COM1A1, COM1A0, COM1B1 and COM1B0 will decide the actions of the timer1. If the 4 Compare Output Mode bits are set to clear the timer1 when TCNT1 equals OCR1A or OCR1B, the timer1 will then be cleared.

Design of a simple delay function with Timer1

In this design, the timer1 will create a 1 second delay to switch on and off a LED light. Set microcontroller PortB 0 as the output to LED. Since the microcontroller has a 16Mhz frequency, set the prescaler as 1024 so that when the TCNT1 value reaches 15624, 1second delay is created. Switch on the LED and clear TCNT1, then start counting to another 1second and switch off the LED and clear TCNT1. As this action continuously running, the LED will be toggled every second. The C programming code can be presented as:

```
# include <avr /io.h>
int main ( void )
{
      DDRB |= (1 << 0); // Set PortB0 as output to the LED
      TCCR1B |= (1 << CS10)& (1<<CS12); // Set the prescaler as 1024
    for (;;)
   {
    // Check the TCNT1 value, once 1 second was reached, execute the if statement
      if ( TCNT1 >= 15624)
    {
      PORTB ^= (1 << 0); // Toggle PortB0 which toggles the LED
      TCNT1 = 0; // Clear TCNT1
      }
   }
}
```

Timers contribute in every part of the programming, in the following sections, more specific usage of the timer1 will be presented with PWM generation, I2C communication, and ADC etc.

6.3 Stepper motor control

a. Stepper motor selection

Stepper motor moves only 1 step when receiving a pulse signal. The stepper motor that is chosen for this project has a 1.8 degree step angle. It will take 200 steps for the stepper motor to complete a full revolution, which means giving a PWM with frequency of 200Hz, the stepper motor will be running at the speed of 1 revolution per second (RPS).

There are two main types of stepper motors, which are the bipolar stepper motor and unipolar stepper motors. The main difference between these two types is that the bipolar stepper motor

only has two coils around its rotor, while the unipolar stepper motor always has one to two extra leads coming out from the motor coils; the leads are connected with the center tap of each coil. Figure 6.2 is an image of a bipolar stepper motor, and Figure 6.3 is an image of a unipolar stepper motors.

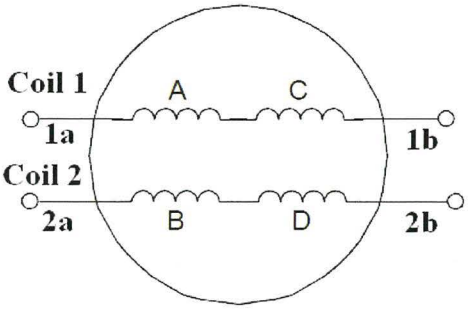


Figure 6.2 image of a bipolar stepper motor

In Figure 6.2 the bipolar stepper motor has 4 coils with 4 leads. The two coils are connected into two pairs. Coil A and C are connected, so are the coils B and D. Coils A and C are named pair 1, and coils B and D are named pair 2. The four leads are named after the pairs, which are 1a, 1b, 2a, and 2b respectively. Energize the leads in a certain order will activate the stepper motor.

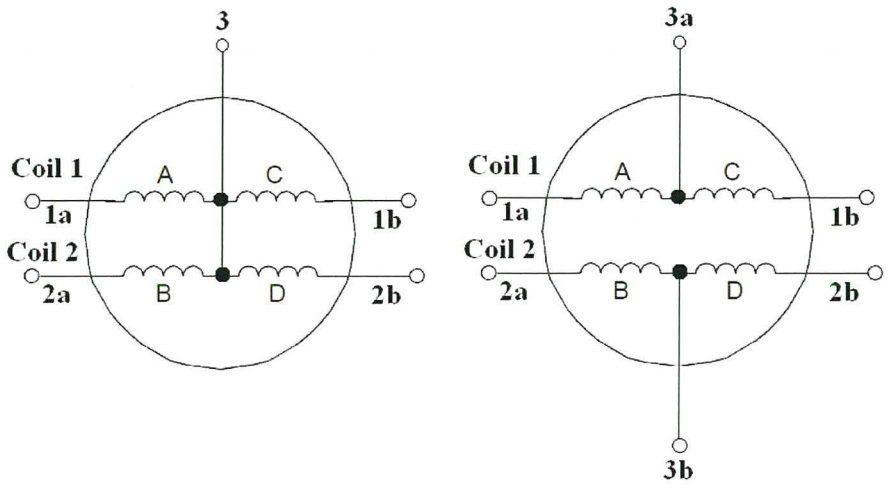


Figure 6.3 image of unipolar stepper motors

In Figure 6.3, the stepper motor on the left is a 5 leads unipolar stepper motor, and on the right side is a 6 leads unipolar stepper motor.

Bipolar stepper motors do not have the center tap which makes the motor construction easier and the bipolar stepper motor generally provides more torque than the unipolar motor, which becomes the reason that bipolar stepper motor was chosen for this project.

b. Stepper motor drive mode selection

There are 4 popular drive modes for the bipolar stepper motor, which are the wave drive mode, the full step drive mode, the half step drive mode, and the micro-step drive mode.

Wave drive mode – This is the simplest drive mode. In this driving method, the least power is consumed, since only one phase of the motor is energized at a time. Table 6.2 demonstrates the order that 4 phases of the motor were energized.

Phase	T	T+1	T+2	T+3
1a	High	Low	Low	Low
2a	Low	High	Low	Low
1b	Low	Low	High	Low
2b	Low	Low	Low	High

Table 6.2 Wave drive mode

The advantage of this driving method is that it guarantees the accuracy of the motors rotational motion. However, the drawback is that only one phase of the motor coils is energized at one time, which is very inefficient and it produces less torque than other methods. As a result, this mode is not preferable for this project. Figure 6-4 shows the signals on each coil in wave drive mode.

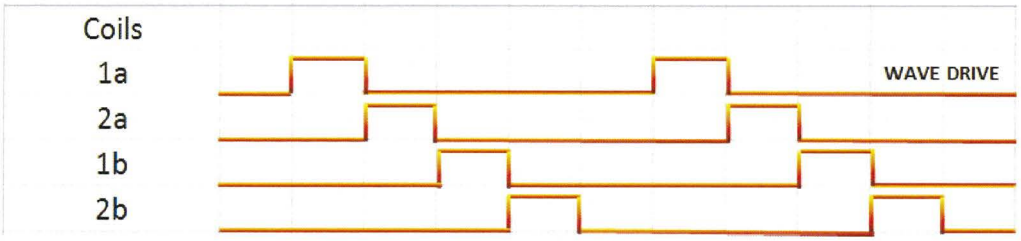


Figure 6.4 signals in wave drive mode

Full step drive mode – In this method, both pairs of the coils of the stepper motor are always energized. Full step drive energizes the coils so that the coils can change their polarities to keep the rotor rotate, instead of switching on and off the coils in sequence. This method gives much more torque than the wave drive mode. One drawback is that it has a natural resonant frequency which increases the stepper motor's vibration. Table 6.3 shows the sequence of the energized coils in full step drive mode.

Phase	T	T+1	T+2	T+3
1a	High	High	Low	Low
2a	Low	High	High	Low
1b	Low	Low	High	High
2b	High	Low	Low	High

Table 6.3 sequence of the energized coils in full step drive mode

Figure 6.5 shows the signals on each coil in full step drive mode.

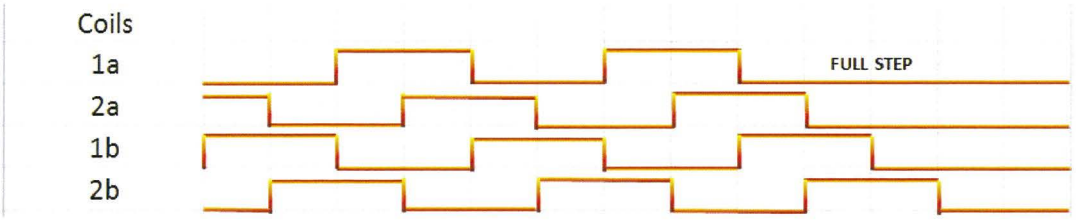


Figure 6.5 signals in full step drive mode

Half step drive mode – In the half step drive mode, the energized coils changes from one to two repeatedly. Alternating between energizing a single phase and both phases together gives the motor a higher resolution in its performances. In this project, the chosen stepper motor is Nema23 T57H56. Each step of the motor is 1.8 degree, which means that it will take 200 steps to complete a revolution in the full step drive mode. However, when it is operating in half step drive mode, it will take 400 steps to complete a revolution, which also means that the half step drive mode has twice the resolution as the full step drive mode. However, the torque will vary depending on the step position. The torque is less when a single phase is energized and greater when both phases are energized. Because the half step drive mode achieves high resolution control of the motor's performances, it is selected for this project.

Table 6.4 shows the coils energizing states in half step drive mode.

Phase	T	T+1	T+2	T+3	T+4	T+5	T+6	T+7
1a	High	Low	Low	Low	Low	Low	High	High
2a	High	High	High	Low	Low	Low	Low	Low
1b	Low	Low	High	High	High	Low	Low	Low
2b	Low	Low	Low	Low	High	High	High	Low

Table 6.3 Half step drive mode

Figure 6.6 shows the signals on each coil of the stepper motor in the half step drive mode.

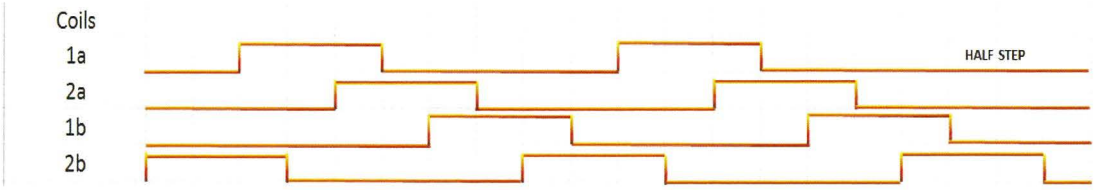


Figure 6.6 signals in half step mode

Micro-step drive mode – when operating the motor in micro-step drive, the steps that the motor rotates to complete a full revolution will be a fraction of a full step (i.e. 1/4, 1/8, 1/16 or 1/32), then the step-rate has to be increased by a corresponding factor (4, 8, 16 or 32) for the same rpm. This drive mode requires very high processing power and the magnitude of the current varies in the coils. The mobile robot in this project is not capable of constantly providing enough power for this drive mode, the power will be drained fast to the motor and consumed rapidly. Therefore, the micro-step drive mode is not affordable to this project. Figure 6.7 shows the signals on each coil in micro-step drive mode.

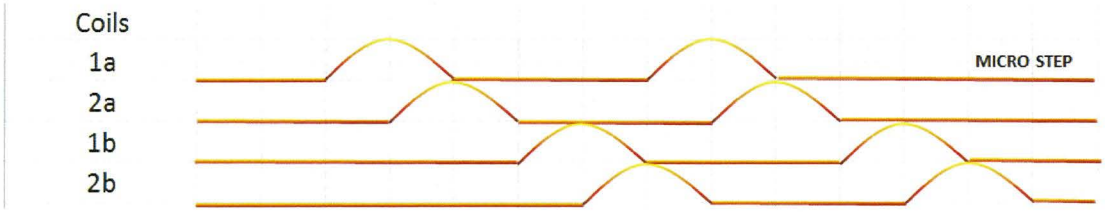


Figure 6.7 signals in micro-step mode

In conclusion – This project selected the half step drive mode to drive the motors, since it has a higher resolution and is effective in power consuming compared to the full step drive mode. On the other hand, compared to the wave drive mode, the half step drive mode has the capability of generating a higher torque, which suits this project very well.

6.3 PWM generation and motor control programming

This project focuses on the study of the maneuverability and moving accuracy of the mobile robot, rather than speed. So the robot will run on a constant speed which increases the stability of its performances.

The three stepper motor drivers are designed in the same configurations in this project. Therefore, the control and programming on motor driver A will be similar to the other two drivers, except the pin connection differences.

The motor driver L297&L298 has 6 input pins, which are the pin17 – CW/CCW, pin18 – CLOCK, pin19 – Half/Full, pin20 – RESET, pin10 – ENABLE, and pin 15 – Vref.

This project selected half step drive mode for the stepper motors, so that Pin19 will receive a constantly high signal from the PB0 on the microcontroller Atmega32. The RESET pin20 and ENABLE pin10 are connected together. Every time the driver chips are enabled, it also reset. These two pins connect to PD2 on Atmega32. A high signal from PD2 will activate the driver chips.

a. Hardware PWM mode selection

The CLOCK pin18 on the stepper motor driver L297 connects to PWM signals from the microcontroller. There are many ways of creating a PWM signal in a microcontroller. The simplest way is creating a PWM with a timer. The timer toggles the signals when the time reaches the set value. However, the timer has to be manually reset every time when it reaches the set value. This puts a burden on the Microcontroller and complicates the programming code. In addition, this method makes it difficult to change the duty cycle and frequency of the PWM.

Atmega32 provides 3 powerful methods for generating PWM with its hardware. There are certain pins on the microcontroller that have alternative functions which can automatically generate PWM without manually resetting any timers. The hardware itself clears the counter once it reaches the set value. Moreover, the duty circle and frequency of the PWM can be easily adjusted. The three PWM generating modes are the "clear on timer compare mode (CTC)", the "fast PWM mode", and the "phase and frequency correct mode", and the phase and frequency correct mode which is the most powerful method, is selected for this project.

Phase and Frequency correct (P&F) PWM – Figure 6.8 demonstrates the principle of the phase and frequency correct PWM mode. In the Figure, ICR1 is the Input Capture Resister in

timer1. It is set to be the top of the timer1, so every time when the timer counts reach this top value, actions like signal toggle or timer clear could occur depending on the bits set in TCCR1A and TCCR1B. ICR1 has very fast response to changed values. If it was reset to another value as top, it could quickly capture the reset value and adjust itself to the new value. OCR1A is the Output Compare resister1. When the timer1 TCNT1 counts up and reaches OCR1A, actions like signal toggle or timer clear could occur depending on the bits set in TCCR1A and TCCR1B as well. OC1A is the Output Compare A Match Output; it is the alternative function of the pin PD5 on the microcontroller. The OC1A will be toggled to form a PWM signal, and it will continuously output the PWM signal to the stepper motor driver A.

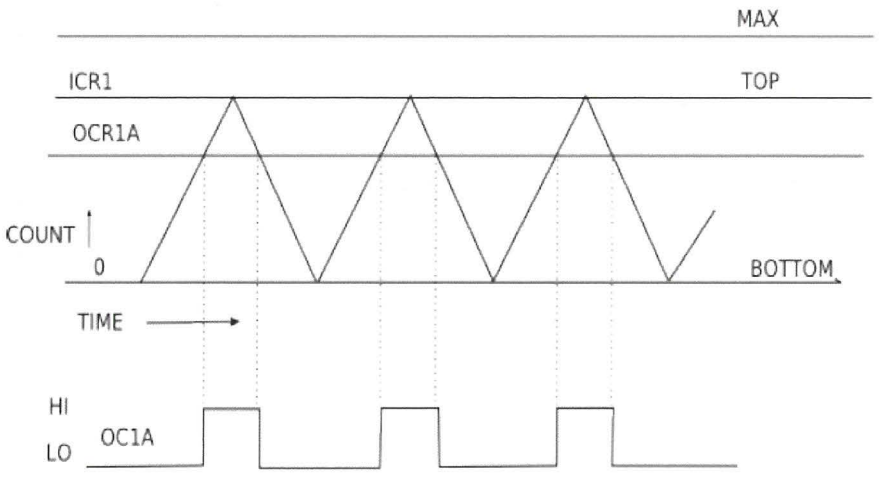


Figure 6.8 Phase and frequency correct PWM mode

It can be seen that when the timer1 starts to count up, the OC1A is kept low until the counter value reaches OCR1A, the OC1A is toggled to high. The timer keeps counting up, arrive at the top ICR1, and it starts to count down. When the counter value down to OCR1A again, the OC1A is toggle to low and will be kept low until the timer counts up again and reaches OCR1A. The timer starts to count up again when its value dropped down touches the bottom line. Therefore the OC1A is toggled to form a PWM. The frequency of the PWM is depend on the top value of ICR1, the higher the ICR1 value is set to be, the lower the frequency

becomes, vice versa, the lower the ICR1 value is set to be, the higher the PWM frequency becomes. The OCR1A value decides the duty circle of the PWM. The higher the OCR1A value is, the smaller the duty circle is, and lower the value of OCR1A will increase the duty circle.

In this project, the speed of the mobile robot's wheel was set to be 1rps, since the gear mesh ratio between the wheel shaft and motor shaft is 2:1, the speed that is required from the motor is 2rps. After testing the stepper motors, it was found that when the motors are running at the speed of 2.5 rps, their performances are very smooth, thus, in the programming C code, the stepper motors are set to run at the speed of 2.5 rps.

Since half drive mode was selected for the stepper motors, the stepper motors will complete a full revolution after they are energized by 400 pulses. With the known 400 pulses per revolution and 2.5 revolutions per second, the frequency of the required PWM can be gained, which is $400 * 2.5 = 1000\text{Hz}$.

b. The design of C code of the PWM generation for the stepper motor driver

Firstly, the timer1 needs to be initiated. The microcontroller cpu runs at 16MHz. To create a 1000Hz frequency PWM, the prescaler of the timer1 is set to 8. Thus the top value ICR1 equals$(16000000\text{Hz}/8(\text{prescaler}))/1000\text{Hz} = 2000$. The ICR1 value is then set to be 1000, because timer1 counts up and down in one pulse period, so with the top value set at 1000 counts, a 2000 counts will appear in that pulse period.

Timer1 initialization

1. Set timer1 top value.

```
ICR1 = 0x3E8; // set top value as 1000
```

2. Set duty circle of the PWM to 50%.

```
OCR1B = 500; // set compare match to 250 for OC1B (50% duty cycle)
```

3. WGM selection

The four WGM bits in TCCR1A and TCCR1B decide the generation mode of the PWM waveform. Set WGM13 to high, WGM12, WGM11, and WGM10 to low, the phase and frequency correct mode is then selected for this project, and ICR1 is set as the top value of the timer1. The datasheet of the WGM selection table is presented in Appendix A.6. ("_BV" in the code functions as setting high to the bit.)

```
TCCR1A |= 0x00;
TCCR1B |= _BV(WGM13); //select the waveform mode as phase and frequency correct mode
```

4. COM selection

The four COM pins in TCCR1A and TCCR1B decide the output compare mode. Set COM1A1, COM1A0, COM1B1 and COM1B0 to high, the OCR1A and OCR1B are then set (set OCR1A to high) on compare match when timer1 is up counting. OCR1A and OCR1B are then cleared (set OCR1B to low) on compare match when the timer1 is down counting. The datasheet of the COM selection table is presented in Appendix A.7.

```
TCCR1A |= _BV(COM1A1)|_BV(COM1A0)|_BV(COM1B1)|_BV(COM1B0);
//switch on the compare output mode for compare units A & B, which enables the OC1A and OC1B.
//set OCR1A&B on compare match while up counting, and clear them when counting down
```

5. Prescaler selection

The timer1 prescaler is set to 8 for creating the 1000Hz frequency for the PWM, so the CS10, CS12 are set to low, and the CS11 is set to high. The prescaler setting also starts the timer.

TCCR1B |= _BV(CS11); // start the timer1 and select the prescaler to 8

// this set and the ICR1 set create a 1000Hz PWM to the motor ,make it run at 2.5rps

The final C code is presented as:

```
/*Half step mode selection*/

PORTB |= 0x01; //Set PORTB0=1 for DriverA,B&C all chose HALF mode;


/*initiate timer1*/

ICR1 = 0x7D0; // set top as 1200 (the speed of the motors will be 2.5res/sec)

OCR1B = 1000; // set compare match as 250 for OC1B (duty cycle is set to be 50%)


TCCR1A |= 0x00;

TCCR1B |= _BV(WGM13); //select the waveform mode as phase and frequency correct
mode

TCCR1A |= _BV(COM1A1)|_BV(COM1A0)|_BV(COM1B1)|_BV(COM1B0);

//switch on the compare output mode for compare units A & B,which enables the
OC1A&B

//and this will set OCR1A&B on compare match while upcounting, and clear them when
counting down

TCCR1B |= _BV(CS11); // start the timer1 and select the prescaler to 8

                    // this gives out a 1000Hz PWM to the motor ,make it run at 2.5rps

}
```

6.4 I2C for Vref

The stepper motor drivers need to receive control signals from the microcontroller to control the motors. CLOCK (PWM), Half/_FULL, CW/_CCW, Enable and _RESET are the control signals that the stepper motors drivers need. There is another pin on the stepper motor driver that requires signals from the microcontroller, which is the pin 15 Vref. The Vref pin on the L297 is the voltage reference for the chopper circuit. Even if the chopper mode is not activated, the voltage reference pin 15 still shall not be left empty, or the stepper motor will not work at all. A voltage applied on this pin determines the peak load current of the motor. This Vref pin requires an analogue signal, from 1V to 3V. It is well known that microcontroller does not generate analogue signals, so a DAC converter has to be added into the electronic system of the mobile robot. The I2C function is chosen for the communication between the microcontroller and the analogue converter chip.

However, the pin18 on L297 requires an input analogue signal, from 0v to 3v. In this project, the stepper motor will be drawing a maximum power from the battery, so that the current limit voltage reference is expected to be maximum, which is 3v.

The DAC convertor chip that is selected for this project is the PCF8591. Three address pins A0, A1 and A2 are used for programming the hardware address, allowing the use of up to eight devices connected to the I2C-bus with the master chip the microcontroller ATmega32. The two-line bidirectional I2C-bus allows the 8 peripheral devices exchange address, control and data witht the microcontroller. The datasheet of the PCF8591 is presented in Appendix A.

The I2C (Inter-IC) bus is a bi-directional two-wire serial bus that provides a communication link between integrated circuits (ICs). In an I2C bus communication system, there is a master

chip, and a certain number of slave chips. The address of the slave chips makes them identical and it is what the master chip uses to track its slaves.

The DAC chip PCF8591 receives a digital signal from the microcontroller and converts it to a 2.8V voltage reference for the stepper motor drivers L297. The I2C configuration is available in Appendix E.1 and the C code is demonstrated in Appendix E.2.

6.5 ADC function

Many sensors collect information and transmit the signals in terms of analogue. For instance the sharp sensors which are used in this project collect the distance information from obstacles and output a set of voltage values to represent the different distances. However, the microcontroller is not able to deal with the analogue signals, thus a conversion between analogues to digitals needs to happen. For this an ADC which stands for analogue to digital converter is needed. ATmega32, just like other MCUs, has ADCs on chip. An ADC converts an input voltage into a number; it has a resolution, which the higher resolution the ADC has, the more accurate the conversion is. A 10 Bit ADC has a resolution in the range of 0-1023. (2^10=1024). The ARef pin is the voltage reference. When input voltage is GND the output is 0 and when input voltage is equal to ARef the output is 1023. So if the input range is 0-ARef then the digital output is 0-1023.

- **ADC Prescaler** – The ADC requires a clock pulse to perform its conversion, just as any other electronic chips do. The MCUs base clock frequency is normally higher than the clock frequencies for other functions. Therefore, for timers/counters and ADC conversions, the clock frequency can be gained by a division of the base clock frequency. The ADC requires a frequency between 50 KHz to 200 KHz. At a higher frequency the conversion is faster while at a lower frequency the conversion is more accurate. The base clock frequency can be set to any value by the user (In this project a 16MHz crystal is used which means the base clock frequency for the MUC is set to 16MHz). The base clock frequency can be divided by 2, 4, 16, 32, 64, and 128 by setting the Prescaler, thus smaller frequencies for the ADC can be obtained.

- **ADC channels** – The ADC in Atmega32 has 8 channels. It means that there are 8 different terminals available for taking samples from the sensors. The channel selection is done by setting a certain bits in the ADC registers.

- **ADC registers**

The ADC has only four registers.

1. ADC Multiplexer Selection Register – ADMUX : For selecting the reference voltage and the input channel.

ADMUX is a 8-bit register. Bit 7 and Bit 6 are the reference selection bits, named as REFS1 and REFS0. The reference selection table is presented in appendix A.9.

In this project, the internal 2.56 voltage reference was selected, thus the Bit 7 and Bit 6 are all set to high.

```
ADMUX = (1<< REFS1) | (1<< REFS0);
```

Bit 5 is ADC Left Adjust Result (ADLAR), setting it to high or low selects the mode of the result adjustment. In this project, the left adjust is chosen, thus:

```
ADMUX = (1<<ADLAR);
```

Bits 4:0 are the Analog Channel and Gain Selection Bits. Appendix A.10 presents the channel and gain selection table. Since the ADC0 and none gain are chosen for the project, so all 5 bits set to 0.

2. ADC Control and Status Register A – ADCSRA: For enabling and starting the ADC conversions, and also showing the status of the conversions. There are seven bits used for this project in this register, which are:

Bit 7 enables the ADC conversion.

Bit 6 starts the ADC conversion.

Bit 4 is the ADC interrupt flag. When the ADC is complete, the interrupt flat is executed by being set to 1.

Bit 3 is the ADC interrupt enable bit. Setting this bit to 1 enables the ADC interrupt.

Bits 2-0 are the ADC prescaler bits. The prescaler selection table is presented in Appendix A.11. The prescaler is set to 2 for this project, thus the bits 2-0 are all set to zero.

3. The ADC Data Register – ADCL (lower bits) and ADCH (higher bits): The final result of conversion is here.

In this project, the sharp sensors are selected to collect the values of distances between the obstacles and the mobile robot. The microcontroller Atmega32 needs to read the data from the sensors and then process them to response to the obstacles. The first step is coveting the analogue values into digital. Here is a test of the microcontroller ADC functions. The Atmega32 reads values from a sharp sensor, and switches on 4 LEDs in sequence according to the distances change between the sensor and the obstacle. The schematic of this test is demonstrated in Figure 6.9:

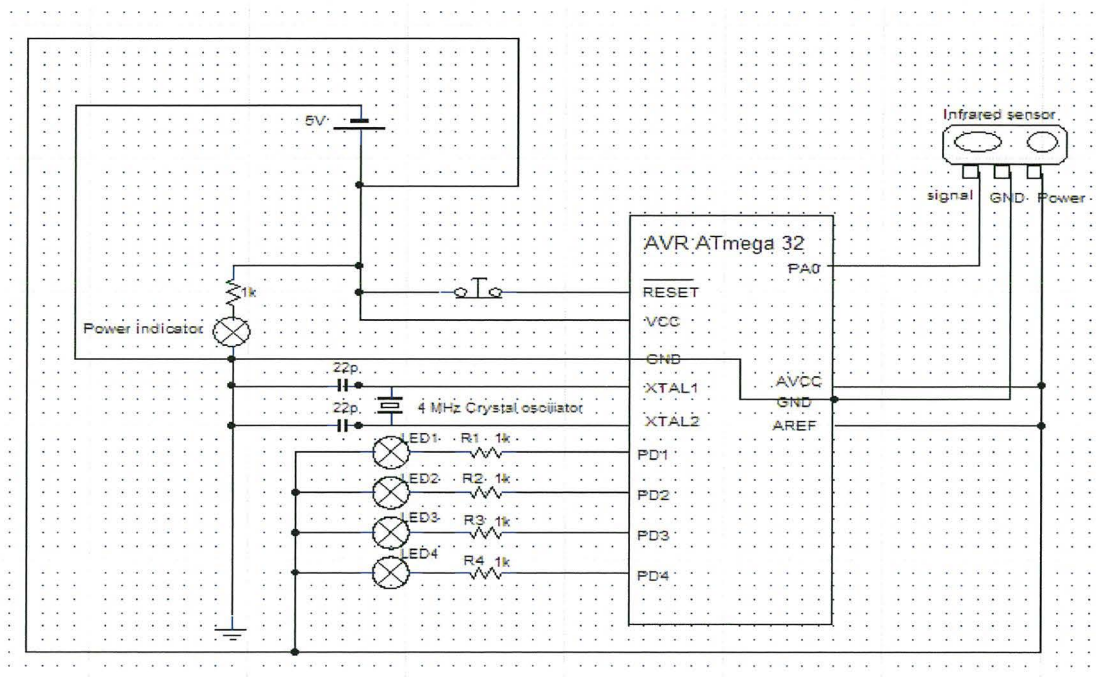


Figure 6.9 ADC test

The reading signal connects to PortA0 as the input to the microcontroller, and PD1-PD4 are selected as outputs and connected to 4 LEDs in different colors. The LEDs are switched on when the sensor detects an obstacle. If the red LED is on, the sensor has 6cm to the obstacle, and yellow matches 10cm, green matches 25cm, and the last one blue matches 30cm. In this project, the 10cm is selected to be the clearance that the robot should keep from any obstacles. Once the sensors detect an obstacle which is in closer than 10meters, the microcontroller will stop the mobile robot. The program code is demonstrated in Appendix E. 3.

6.6 conclusions

The control system is basically formed with the microcontroller, the stepper motor drivers, and the sharp sensors. The microcontroller sends signals to the stepper motor drivers, and the driver chips can control the stepper motors with the given commands. The "PWM" defines

the speed and torque of the stepper motors. And the "Enable" starts the motors, and the "CW/CCW" defines the rotational directions of the motors.

The bipolar stepper was selected for this project and it performs in its half step drive mode, which gives the stepper motor a better accuracy of resolution in each revolution, and reduces the power consumptions compared to the Full step drive mode.

PWM was created by timer1, the phase and frequency correct drive mode was the method. P&F mode gives the PWM a high resolution, and by setting the top and compare values, the frequency and duty circle of the PWM can be easily adjusted.

The I2C was used to be the communication tool between the microcontroller and the DAC converter PCF8591; a 2.8 V analogue reference was created.

The Sensors formed the feedback system, when the sensor detects any obstacles which is in a closer than 10meters to the mobile robot, it will send signals to the microcontroller, and the microcontroller will tell the mobile robot to stop. The signals that were sent to the microcontroller are analogue ones; the ADC function converts them to digitals, so that the microcontroller is able to read them.

Chapter 7 Testing

7.1 Basic movements of the three-omni-wheeld mobile robot

To start with the test design, the study of the basic movements of the robot is necessary. The omni-directional mobile robot has different control methods compared to the traditional robots. With the three omni-wheels, the robot can achieve some difficult tasks that the normal robots are difficult to do so. For instance the omnidirectional robot can slide to left and right without turning its head in the sliding directions in advance, and it can rotate around its center without shifting.

The Figure 7.1 presents a simplified construction of the three-omni-wheeld mobile robot. In Figure 7.1, the wheels on the mobile robot are named as wheel 1, 2 and 3. Every wheel is connected to a motor. The motors are then named as motor 1, 2 and 3 as well.

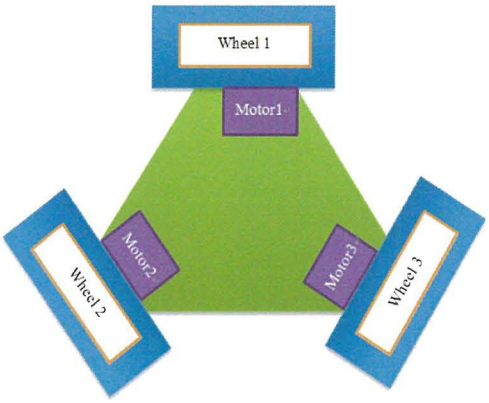


Figure 7.1 Wheel numbers of the robot

The basic movements of the mobile robot are "straight forward", "straight backward", "turning left and right", and "rotating clockwise and counter-clockwise". Figure 7.2 demonstrates the basic movements of the mobile robot.

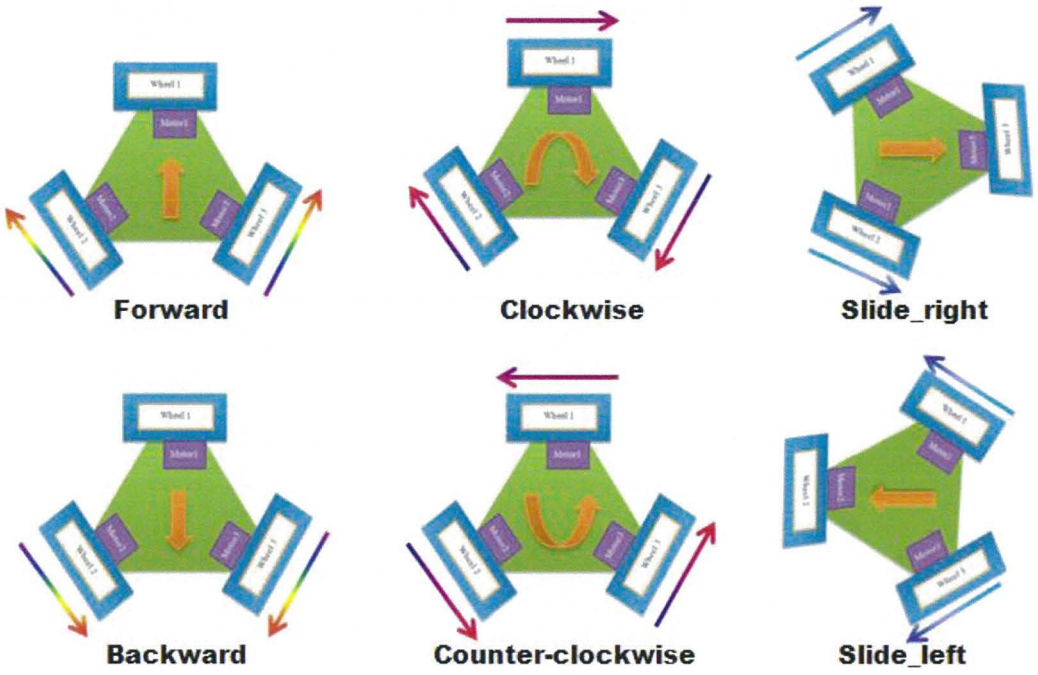


Figure 7.2 Omnidirectional mobile robot basic movements

In Figure 7.2, the yellow arrow in the middle of the robot shows the moving direction of the mobile robot. The colored arrows beside the wheels show the wheels' rotating directions. The basic movements of the robot in accordance with Figure 7.2 are listed in Table 7.1.

movement \ Wheel	Wheel 1	Wheel 2	Wheel 3
Straight forward	Hold still	Clockwise	Counter-clockwise
Straight backward	Hold still	Counter-clockwise	Clockwise
Slide_Left	Counter-clockwise	Hold still	Clockwise
Slide_Right	Clockwise	Counter-clockwise	Hold still
Rotate CW	Clockwise	Clockwise	Clockwise
Rotate CCW	Counter-clockwise	Counter-clockwise	Counter-clockwise

Table 7.1 Rotational directions of the omni-wheels in basic movements

Table 7.1 lists the wheels rotational directions for the basic motions. The arrows shows next to the wheels represent the directions that the motors rotate in. The clockwise and counter-clockwise wheel rotational directions are defined according to the directions that the arrows point to toward the center of the mobile robot.

When the mobile robot moves straight forward or backward, only two wheels which are the wheel 2 and 3 will rotate. The two wheels rotate in the opposite directions. Switching rotational directions of the two wheels switches the motions of the mobile robot between forward move and reverse.

When the mobile robot slides to the left, wheel 1 and wheel 3 rotate in opposite directions, wheel two holds still; and when it slides to the right, wheel 1 and wheel 2 rotate in opposite directions, wheel 3 holds still.

When the mobile robot rotates around its center, three wheels rotate together. The wheels rotate in the same rotational direction, and switch the rotational directions switches the motions of the mobile robot between clockwise rotate and counter-clockwise rotate.

The wheels are connected and driven by the motors. As a result, the motors rotate in the same directions as the wheels do. The motors are driven by the motor drivers L297 & L298 which receive control signals from the microcontroller ATmega32. The signals are the Enable/Reset which turns on and off the driver chips; and the PWM and CW/CCW which controls the motors' speed and torque, and rotational directions. Table 7.2 shows the pin connections between Atmega32 and stepper motor drivers.

Motor	Out pin	Data
	16	Enable/Reset(High=Enable, Low=Reset)
Motor1	18	PWM
	17	Direction input for Motor 1 (High=Clockwise, Low=Counter-clockwise)
	2	Enable/Reset(High=Enable, Low=Reset)
Motor2	4	PWM
	3	Direction input for Motor 2 (High=Clockwise, Low=Counter-clockwise)
	20	Enable/Reset(High=Enable, Low=Reset)
Motor3	19	PWM
	26	Direction input for Motor 3 (High=Clockwise, Low=Counter-clockwise)

Table 7.2 Pin connections between Atmega32 and the stepper motor drivers

In Table 7.2, the out pin column shows the numbers of the output pins on the microcontroller ATmega32.

The control signals that the 3 motor drivers receive from the microcontroller are listed in table 7.3, along with the basic motions.

Basic Motion	Motor 1		Motor 2		Motor 3	
	Enable/Reset	DIR_1	Enable/Reset	DIR_2	Enable/Reset	DIR_3
Forward	Low	Low	High	High	High	Low
Backward	Low	Low	High	Low	High	High
Left slide	High	Low	Low	Low	High	High
Right slide	High	High	High	Low	Low	Low
Rotate CW	High	High	High	High	High	High
Rotate CCW	High	Low	High	Low	High	Low

Table 7.3 control signals for the basic motions of the mobile robot

It can be seen from Table 7.3 that the Enable/Reset and DIR pins on the stepper motor drivers toggle the motors to achieve the basic motions of the mobile robot.

7.2 Maneuverability test design

The three-omni-wheeld mobile robot, different from robots with normal wheels, has a special triangle shaped mechanical structure. This special structure gives the mobile robot abilities to achieve the performances of some special motions which other mobile robots would not be able to or hardly do so. The special motions like sliding and center rotating will be tested, meanwhile, the normal motions such as forward and backward moving need to be tested as well to observe the overall performance of the 3omni-wheeled mobile robot in any motions.

The maneuverability and accuracy test design of three-omni-wheeld mobile robot consists of 5 parts. The 5 tests are linear forward moving test, rotational moving test, square path moving test, sharp angle turning test and a real life in room simulating test. Both the basic motions and the complex motions of the mobile robot will be tested. The basic motions include the pure linear motions (forward/backward) and the pure rotational motions (rotating CW/CCW). The complex motions include the combination of the pure linear and rotational motions, such as curving.

7.2.1 Linear movement test

Purpose of the test – Linear movement is the most basic motion that many modern mobile robots can achieve. The three-omni-wheeld mobile robot; however; due to its special physical construction, is considered less accurate in its linear motions than the normal mobile robot. This test will focus on examining the accuracy of this three-omni-wheeld mobile robot design in completing the linear motion tasks.

In the linear motion, there are only two wheels rotate. One of the simplest linear motions is moving straight forward. In the straight forward motion, wheel 2 and 3 rotate clockwise and counter-clockwise respectively, as shown in Figure 7.3; each wheel's rotational movement contributes a speed vector into the forward direction, and create a horizontal speed vector which cancels each other. Meanwhile, wheel 1 holds still making sure that the robot moves straight. Wheel 1 does not contribute speed in forward motion. The side rollers on wheel 1 contact the ground acting as caster wheels. The side rollers are not good at maintaining the straight motions. Thus a little error of lead angle that is created by the rollers will gradually lead the mobile robot astray from the perfect straight path. In addition, motor 2 and motor 3 are never able to provide the exact same amount of torque to rotate wheel 2 and wheel 3; therefore, the expecting result of this test is that the mobile robot will move astray from the given straight route, the off distance varies every time when the mobile is tested.

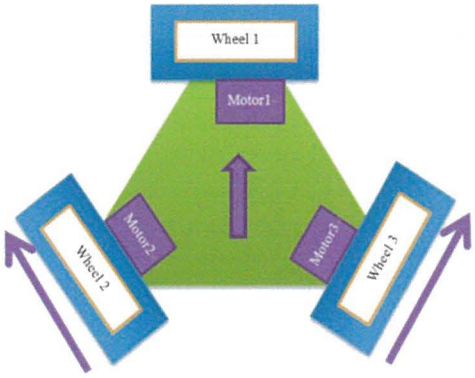


Figure 7-3 forward moving

Test design – One of the simplest linear movements "moving straight forward" is selected to test the accuracy of the mobile robot's linear motion. The mobile robot is commanded to move along a 10 meters straight route which is marked with plastic tapes on the ground. The test is done in the Aghort Hall, Massey University. The test design is shown in Figure 7.4.

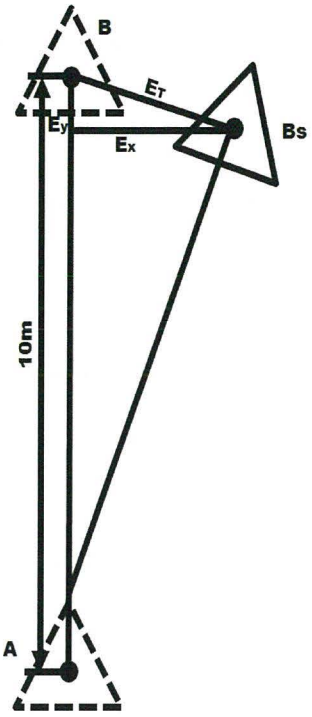


Figure 7-4 forward motion test

In Figure 7.4, it can be seen that the mobile robot starts running at position A. It then moves straight forward for 10 meters, reaches the end position B. If the robot follows the straight route, it will stop at position B perfectly. Due to the inaccuracy of the mechanical design of the omni-wheels, the unbalanced torque outputs from the motors, and the unevenness of the ground, the mobile robot would run away from the perfect route. The robot would stop at position Bs which stands for the "shifted position B". The displacement between B and Bs is the error of inaccuracy. This error displacement is named as ET (Total Error) and consists of two vectors, which are Ex (error vector on horizontal axis) and Ey (error vector on vertical axis). By measuring and analyzing the error values the accuracy of the robot's linear motion can be obtained. The microcontroller programming for the test is presented in Appendix C.1.

The robot forward motion test was carried out 30 times and the data that was captured is presented in Table 7.4. Matlab is then employed for the analysis of the test data. The Mean value, standard deviation and confidence interval of Error Ex and Ey are calculated. A photo of the robot's forward motion test is shown in Figure 7.5.

Figure 7.5 a photo of the mobile robot's linear motion test

Test result – The result of the linear motion test is listed in Table 7.4

Result	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ErrorX(cm)	15.8	34.2	21.2	-5.3	12.2	-15.8	23.6	-26.3	29.3	35.4	10.8	20.9	-9.4	26.9	18.2
ErrorY(cm)	5.6	9.8	8.2	-6.4	-8.3	-2.5	-8.6	3.6	15.8	10.9	-12.5	-6.9	9.2	5.8	5.4
Result	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
ErrorX(cm)	22.8	-10.2	-8.4	-6.3	26.5	30.4	13.4	16.2	-6.4	13.7	12.9	10.9	22.6	28.7	19.7
ErrorY(cm)	-13.6	8.5	12.1	15.9	-8.5	9.6	9.4	-2.8	-6.3	5.9	15.4	-3.8	-11.1	5.6	2.2

Table 7.4 Result of linear motion test

Following is the Matlab program that calculates the Mean value, standard deviation and confidence interval of Error X and Error Y. The equation of confidence interval is shown in Figure 7.1

Equation 7.1 95% confidence interval

$$\mathbf{CI} = [\mathbf{M} - \frac{\mathbf{Z*SD}}{\sqrt{N}}, M + \frac{\mathbf{Z*SD}}{\sqrt{N}}] \quad \textbf{(7.1)}$$

Whereas in Equation 7.1, CI represents confidence interval; M represents the Mean value; Z is a constant, which is given as 1.96 for 95% confidence interval; SD represents the standard deviation; and N represents the number of test that has been carried out. The Matlab programming is attached in Appendix C1

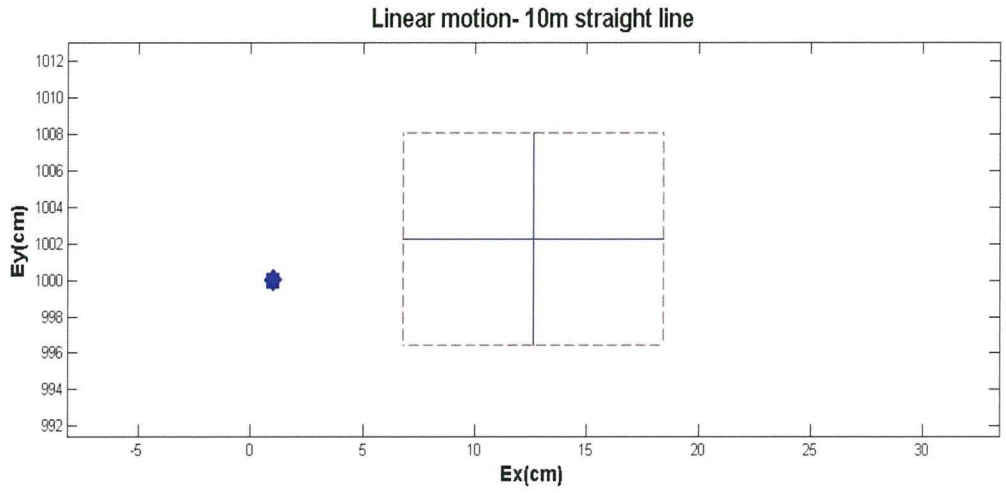


Figure 7.6 95% confidence interval of the shifted displacement in linear motion

Figure 7.6 presents the outcome of the mobile robot's linear motion test. The 95% confidence interval of the shifted displacement is shown in the red dotted area. The blue dot is the average value of the positions that the mobile robot stops at. Therefore, it can be 95% confidently stated that the robot would not stop at the position B perfectly. This proves that the linear motion of the three-omni-wheeld robot is not very accurate. The main causes of the errors are the mechanical inaccuracy of the omni-wheels, unbalanced torque provision from the motors, and the rough surface of the ground.

Using Matlab, the mean of the error comprises in the percentage of 10 meters distance can be presented.

Equation 7.2 mean of the error comprises in the percentage of 10 meters distance

$$Error\% = \frac{(MX^2+MY^2)^{0.5}}{10meters} * 100 \qquad (7.3)$$

Whereas in equation 7.2, Error% represents the percentage that the error is taken place in the 10meters test distance. MX represents the mean value of error X, and MY represents the mean value of error Y.

The total error occupies 1.28% of the 10 meters test distance. Since the error is less than 1.5%, it can be considered that the accuracy of the mobile robot's linear motion is acceptable.

7.2.2 Rotational motion test

Purpose of the test – Similar to the linear motion, rotational motion is one of the basic motions that the mobile robot can perform. Compare with the linear motions, the three-omni-wheeld mobile robot is much better at its rotational motions. With wheel 1, 2, and 3 rotate to the same direction, the mobile robot is able to rotate around its center. Reversing the rotational direction of the motors reverses the rotational direction of the robot. The rotational accuracy of the three-omni-wheeld mobile robot is designed to be tested and compared with the accuracy of the linear motion. Two rotational motion tests are designed, which are the center rotating test and the circling test.

a. Center rotating test

In the center rotating test, all three omni-wheels drive the mobile robot. If the three wheels rotate in one direction, the mobile robot rotates in the same direction; reversing the motor's

rotational direction reverses the mobile robot's direction. The test design is shown in Figure 7.7.

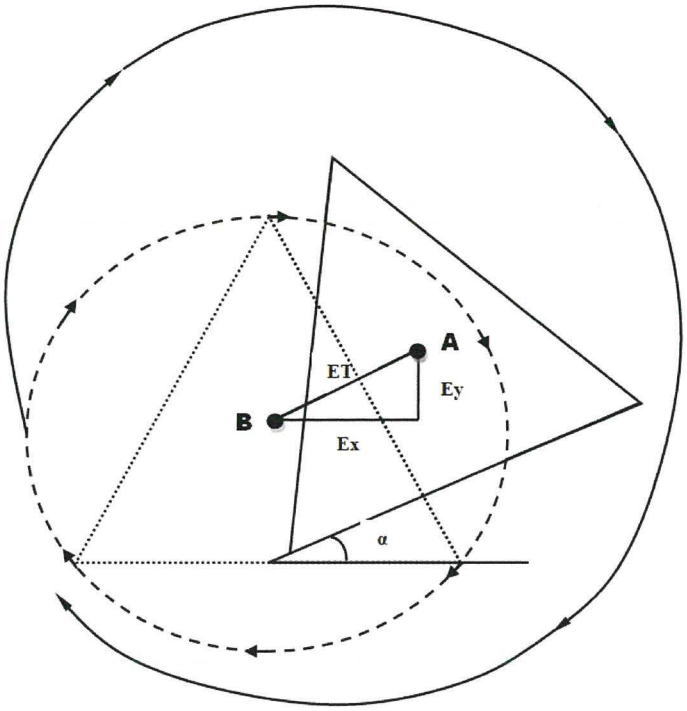


Figure 7.7 Center rotation test

In Figure 7.7, the dotted triangle represents the motor's initial position. Point B is the center of the mobile robot. The mobile robot is designed to rotate around its center for 8 circles. The ideal rotational motion is that the mobile robot only rotates in the dotted circle area, any shift displacement in any directions will be considered as errors. Because of the inaccuracy of the omni-wheel' mechanical structure and the unbalanced torque provisions from the three motors, the mobile robot is expected to shift away from its initial position B. The point A is the assumed ending position that the robot would stop at. The circle in solid line is the assumed area that the mobile robot could shift into. The displacement that the mobile robot shifts from the dashed circle is called ET, which stands for the total error. There are two vectors that contribute to ET, one is the error generated along the horizontal axis named as

"Ex" and the vertical error "Ey". Ex and Ey will be measured for presenting the inaccuracy of the center rotating motion. There is also a shifted angle between the mobile robot's initial and ending positions, which is named "α". Measuring and analyzing this angle also contribute to the accuracy test of the mobile robot's rotating motion. The programming for the test is presented in Appendix C.2. Figure 7.8 is a photo of the mobile robot's center rotating test. Figure 7.8 is an image of the center rotating test.

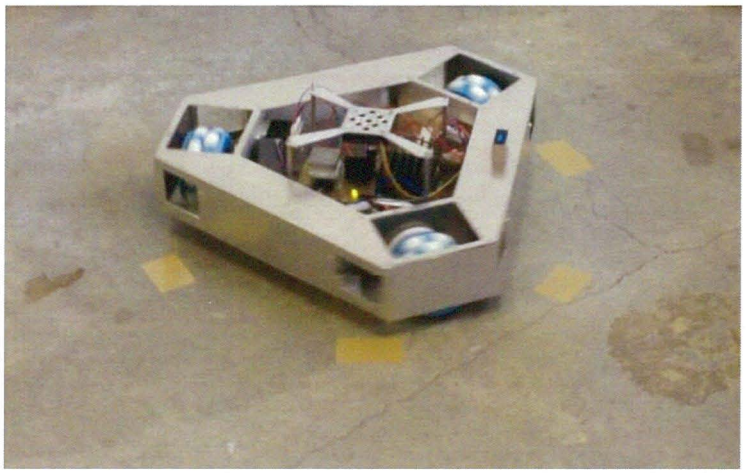


Figure 7.8 center rotating test

Test result – the test was carried out 30 times, and the data of errors is collected and presented in Table 7.5

Result	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ErrorX(cm)	10.2	13.5	8.4	-6.9	-9.2	-9.1	2.8	-8.5	-1.6	2.8	6.4	-8.2	-7.6	6.6	-5.3
ErrorY(cm)	5.6	-9.8	8.2	-6.4	-8.3	-2.5	-8.6	3.6	15.8	10.9	-12.5	-6.9	9.2	-5.8	5.4
Result	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
ErrorX(cm)	3.4	-7.6	5.8	8.9	3.5	-4.9	5.6	5.8	-6.6	-5.9	-8.9	7.1	6.2	-9.3	12.5
ErrorY(cm)	-13.6	8.5	12.1	-15.9	-8.5	9.6	9.4	-2.8	-6.3	5.9	15.4	-3.8	-11.1	5.6	2.2

Table 7.5 Error of displacement in center rotation

Analysing the data in Table 7.5 with Matlab, the 95% confidence interval of the error's distribution can be obtained. The Matlab programming code is presented in Appendix D.2.Figure 7.9 demonstrates the result.

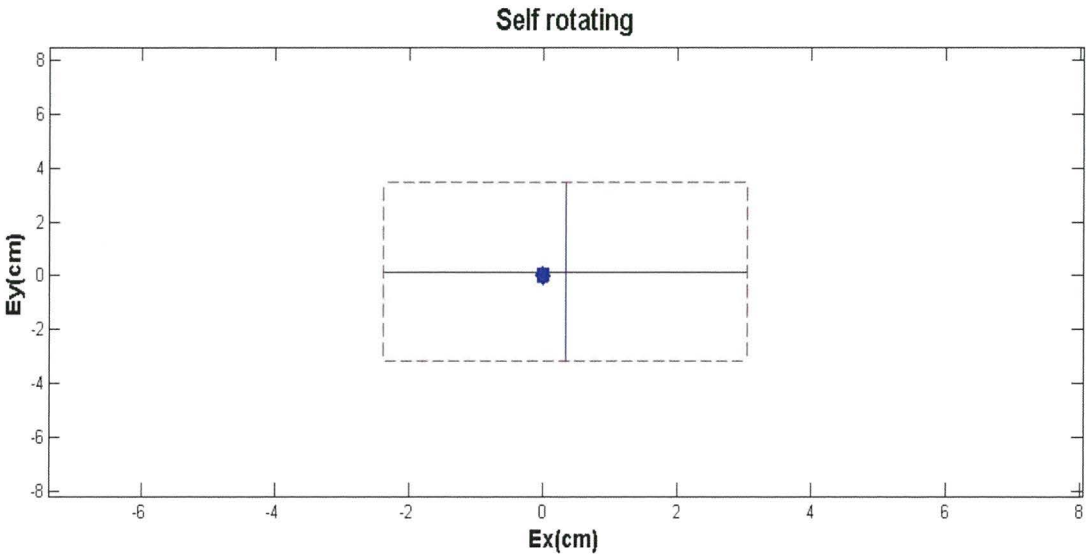


Figure 7.9 displacement shifted in center rotational motion

In Figure 7.9, it is obvious that the average mobile robot's stop position (the blue dot) is included in the red dotted block (95% confidence interval that the mobile robot could possibly stop at its initial position). The rotational moving accuracy of the mobile robot is proved to be great.

The test was carried out 30 times, and the data of shifted angle α is collected and listed in the Table 7.6.

Result	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
α	8.2	13.5	8.4	-6.9	-9.2	-9.1	2.8	-8.5	-11.6	12.8	6.4	-8.2	-17.6	6.6	-5.3
Result	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
α	13.4	-7.6	5.8	8.9	13.5	-4.9	5.6	5.8	-16.6	-5.9	-8.9	7.1	6.2	-9.3	12.5

Table 7.6 Error of angle shift in center rotating test

Process the data in Table 7-6 with Matlab. The result is presented in Figure 7.10. The Matlab programming code is attached in Appendix D.3.

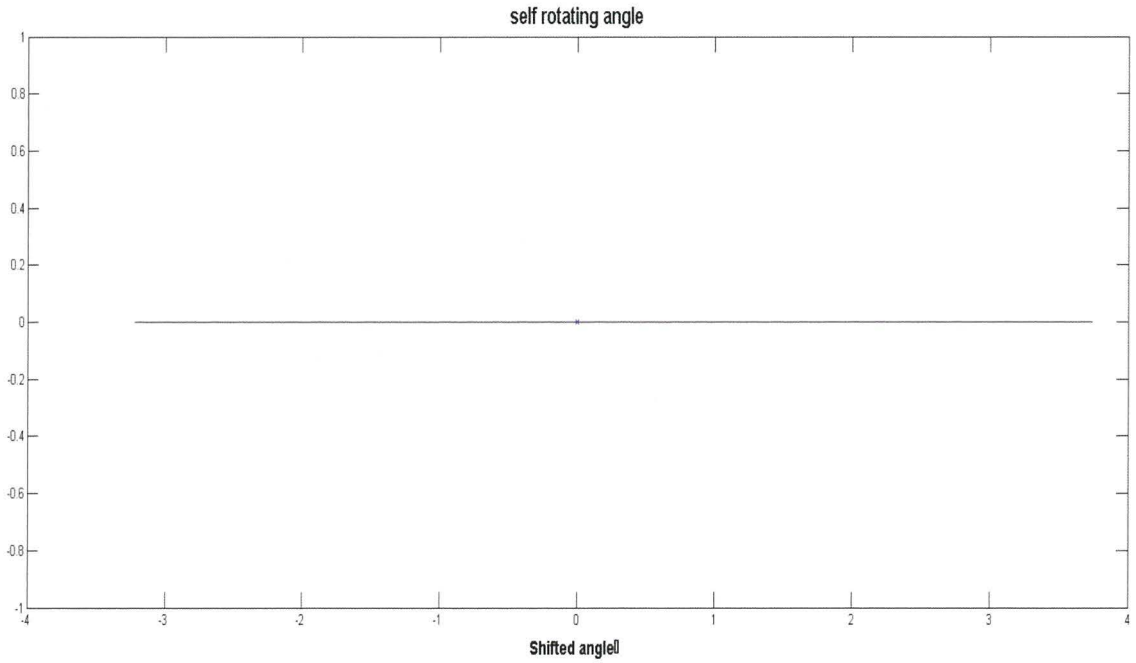


Figure 7.10 Angle shifted in center rotating motion

In Figure 7-10, it is confident to state that, 95% of time the shifted angle would locate on the blue line, which means 95% of time the shifted angel could be possibly equal to zero degree, in other words, 95% of time the mobile robot is possible to stop right at its initial position. This proves that the mobile robot has a great accuracy in the center rotating motion.

The distance between the center of the mobile robot and any wheels is 21cm. Every time the robot rotates around its center for one circle, each wheel actually travels 2 * π * radius of the circle (mobile robot's center to wheel distance). When the mobile robot completes 8 cycles' rotation, each wheel on the chassis has travelled approximately 10 meters linearly. With the same travel distance (10 meters), the accuracy of the mobile robot's rotational and linear motions can be compared. In the linear motion, 95% time the mobile robot will not be possible to stop at its initial position, while in the rotational motion, 95% time the mobile robot will be possible to stop at its initial position, which proved that the rotational motion has much greater accuracy over the linear motion.

b. circling test

Purpose of the test – The rotational motion includes not only center rotating, but also circling or curving. In real life applications, many mobile robots are required to move along curve paths. The accuracy of the curving motion is an important feature for a mobile robot. Thus, the test of the mobile robot's circling or curving motion becomes significant. When the mobile robot follows curve paths, two wheels rotate in the same direction, driving the robot moving forward; the other wheel rotates in the opposite direction, directing the robot to follow the curves. Chang the speed of the 3 wheels will change the radius of the curve.

Test design – This test is set in Massey Aghort hall. The mobile robot is told to curve moving to complete a full circle. The diameter of the circle is 2 meters. Figure 7.11 demonstrates the test design. The dotted circle is the perfect circle track that is designed for the mobile robot to follow, the robot is told to depart from position A (the dotted triangle represent the mobile robot's initial position), move along the dashed circle, and finally stop at the position A again to complete the full circle. However, due to the lack of accuracy of the omni-wheels' mechanical structure and the unbalanced torque provisions from the motors, the robot will

move astray from the perfect circle, run on the solid line instead, and it will finally stop at position B. The displacement between the position A and position B is called total error and is E_T. The error has two vectors, Ex and Ey, which are located in horizontal and vertical axis. The error will be measured and analyzed to examine the accuracy of the mobile robot's circling motion. The programming for this test is presented in Appendix

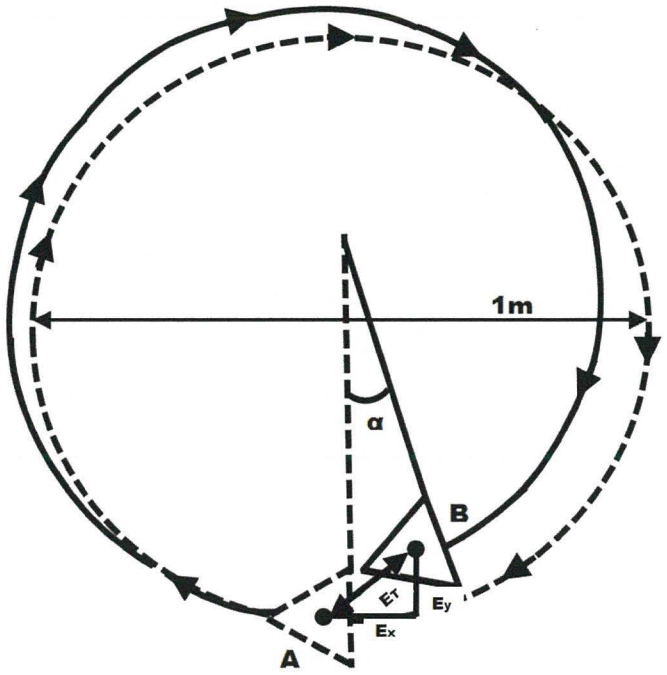


Figure 7.11 Circling angle shift

Figure 7.12 is an image of the mobile robot in the circling test

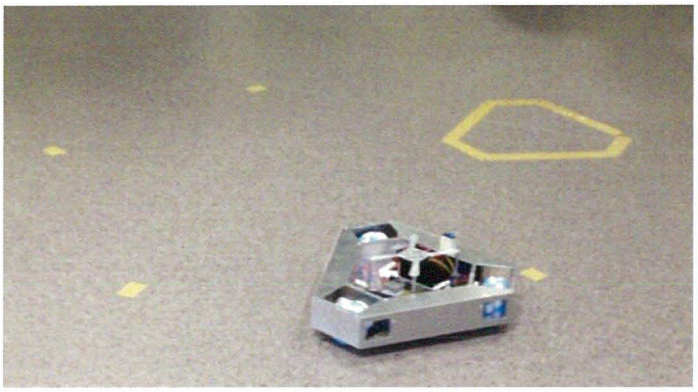


Figure 7.12 image of the mobile robot in the circling test

Test result – The result of the test is presented in Table 7.7.

Result	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ErrorX(cm)	-13.6	8.5	12.1	-15.9	-8.5	9.6	9.4	-2.8	-6.3	5.9	15.4	-3.8	-11.1	5.6	2.2
ErrorY(cm)	13.4	-7.6	5.8	8.9	13.5	-4.9	5.6	5.8	-16.6	-5.9	-8.9	7.1	6.2	-9.3	12.5
Result	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
ErrorX(cm)	11.2	-8.6	-6.2	9.5	6.8	-5.4	-6.5	8.2	6.1	-9.1	-6.7	7.2	4.9	-6.8	-9.6
ErrorY(cm)	8.2	6.1	-9.1	-6.7	7.2	4.9	-6.8	-9.6	11.2	-8.6	-6.2	9.5	6.8	-5.4	-6.5

Table 7.7 shifted displacement in the circling test

Analyzing the data in Table 7.7with Matlab, the accuracy of the mobile robot's circling motion can be obtained. The Matlab programming is presented in AppendixD.4

Figure 7.13 demonstrates the result data analysis for the circling motion.

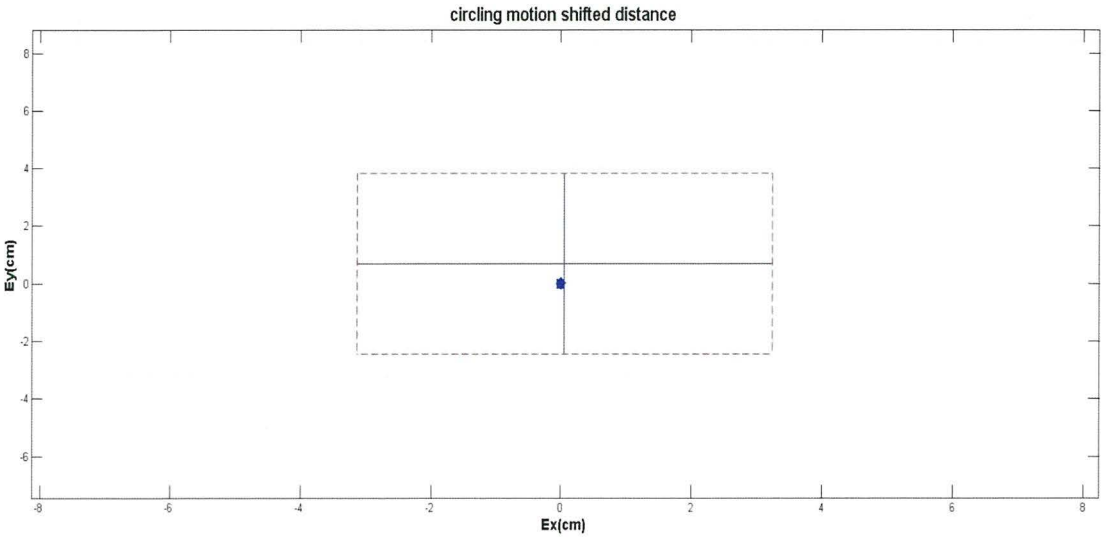


Figure 7.13 displacement shifted in the circling motion test

In Figure 7.13, it can be seen that the mobile robot's stop position B (the blue dot) is included in the red block area (95% confidence interval). Therefore, it can be stated that 95% of time that the mobile robot is possible to stop at its initial position, finish the circling movement perfectly. Thus, the accuracy of the mobile robot's rotational motion is proved to be great once again.

7.2.3 Test of the mobile robot following a square-shaped track

Purpose of the test – This test is designed to examine the combinational motions of the mobile robot. The motions include the pure linear motion (moving forward) and rotational motion (turning a 90° angle). In industrial fields, there are many mobile robot applications that require the robot to be able to perform both linear and rotational motions. Therefore, the test of the combinational motion of the mobile robot is significant.

Test design – The test design is presented in Figure 7.14. The robot is told to follow the square-shaped track. The length of the square is 3 meters. The mobile robot will depart from its initial position A (the dotted triangle A), move along the square-shaped track which is marked with the dashed line. The mobile robot will turn a 90 degree angle at position B, C, and D and finally travel back to Position A. In reality, the mobile does not follow the tracks perfectly; it is quite likely to move astray from the track. The solid lines in Figure 7.14 are the expected tracks that the mobile robot would actually move on. The shifted displacement (error) at position A, B, C and D are named Ea, Eb, Ec and Ed. The errors will be measured and analyzed.

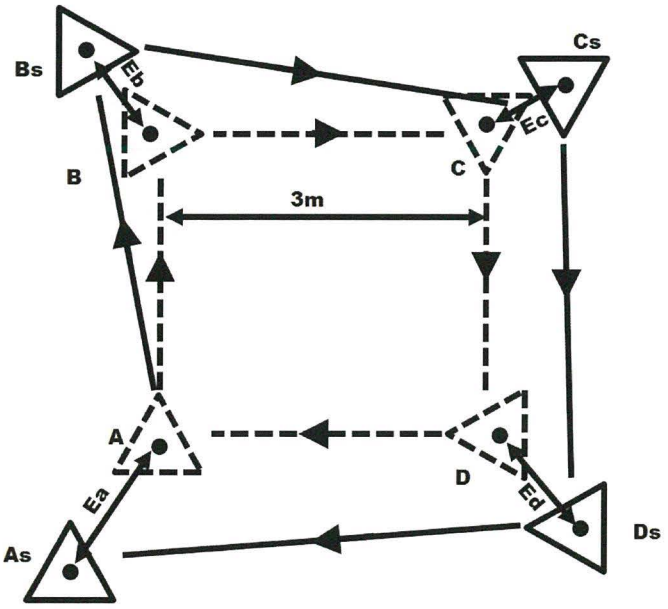


Figure 7.14 test of the mobile robot following a square-shaped track

The test is carried out 5 times. Each time the shifted distances are recorded and presented in Table 7.8. The shifted displacement at each position of A, B, C and D can be divided into two vectors, one vector is located at horizontal axis and the other at vertical axis. The vectors of

the errors are named Eax, Eay; Ebx, Eby; Ecx, Ecy; and Edx, Edy respectively. The c programming of the test is presented in Appendix

	Test 1	Test 2	Test 3	Test 4	Test 5
Eax(cm)	-5.8	-6.2	5.1	4.8	-4.7
Eay(cm)	4.6	4.8	-3.1	6.4	4.5
Ebx(cm)	2.2	3.3	1.7	-2.8	-1.3
Eby(cm)	1.5	1.8	-2.6	2.9	3.6
Ecx(cm)	1.8	2.6	2.7	-3.4	1.1
Ecy(cm)	-2.6	2.8	-1.9	-1.1	3.2
Edx(cm)	6.6	5.6	-5.1	4.6	4.8
Edy(cm)	4.9	4.2	-3.7	-6.8	-3.1

Table 7.8 Error vectors at position A, B, C, & D

Figure 7.15 demonstrates the shifted displacements at position A, B, C and D.

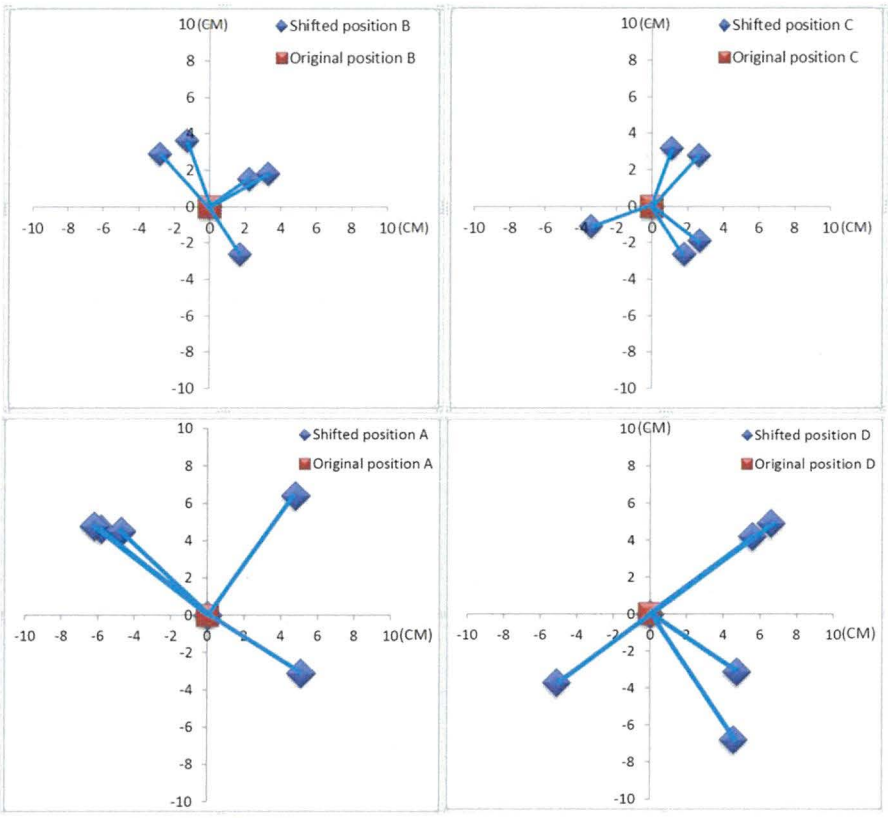


Figure 7.15 demonstration of the shifted displacements at position A, B, C and D

In Figure 7.15, red dot (origin) in each image is the perfect position that the mobile robot is told to stop at. The blue dots are the positions which the mobile robot actually stopped at. The blue lines represent the shifted displacements.

The test is done in the Aghort Hall in Massey Turitea campus. Figure 7.16 shows four images of the test.

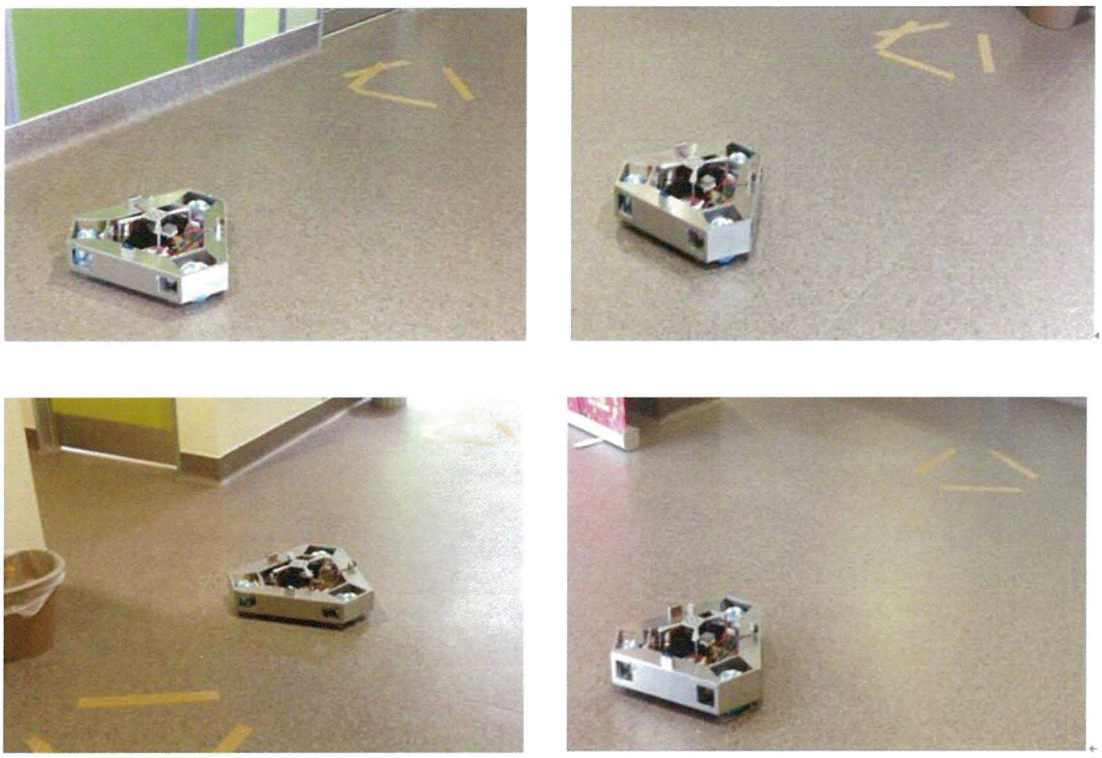


Figure 7.16 square-shaped track test

The surface of the testing ground has different degree of roughness at position A, B, C and D. The unevenness of the ground surface at position A and D are worse than that of position B and C. In addition, the errors slightly accumulate at each position from B to C to D then back to A. Therefore, it can be seen in Figure 7.15 that the displacements shifted at position A and D are noticeably greater than that of B and C. The displacement difference shows that the roughness of the testing surface affects the motion of mobile robot in a noticeable way, which is mainly due to the mechanical structure of the omni-wheels. However, the mobile robot is able to reach all positions in the test, which proves that the mobile robot design is capable of

completing its combinational motions. The errors of shifted displacement can be reduced by applying better omni-wheels for the mobile robot.

7.2.4 Test of the mobile robot turning a sharp angle

a. 60 degree sharp angle turning

Purpose of the test – Some mobile robot applications are required to turn sharp angles, however, with normal wheels, the mobile robot is not able or hardly to do so. The great maneuverability of the omni-wheels makes the omni-wheeled mobile robot capable of achieving sharp angle turnings.

Test design –Figure 7.17 demonstrates the test design. The mobile robot will depart from position A (the dashed triangle A represents the initial position of the mobile robot), move forward for 1.5 meter, turn a 60 degree angle and stop at position B (the dashed triangle B) after travelling another 1.5 meters. The reason 1.5 meters linear movement is added in this test is that it makes the measurement of the turned angle easier. The mobile robot is expected to generate an error in terms of angle which shifts from the designed track (the dashed lines) and stop at the shifted position Bs. This shifted angle is named "α". The shifted angle will be measured to examine the accuracy of the mobile robot's sharp turning movement.

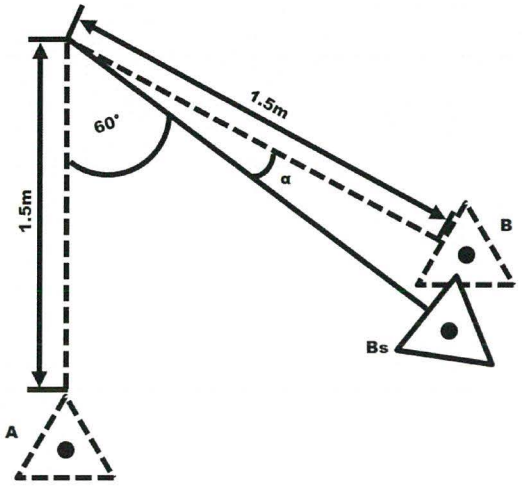


Figure 7.17 test of the mobile robot turning a sharp angle (60 degree)

The c programming for this test is presented in Appendix C.5. Figure 7.18 is an image of the mobile robot in the sharp angle turning test.

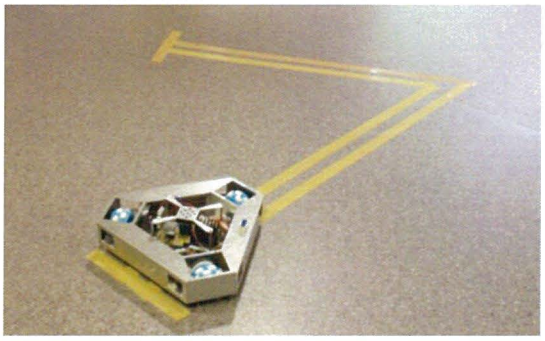


Figure 7.18 60° sharp angle turning test

b. 450 degree sharp angle turning

Test design – In this test, the angle changes from 60 degree to 45 degree. Figure 7.19 demonstrates the test design. The mobile robot departs from the same position A, moves forward and turns a 45 degree angle, and finally stops at position B. The mobile robot is expected to run astray from the designed track (the dashed lines) to the solid lines, and stop at the shifted position Bs. The shifted angle "α" will be measured and presented.

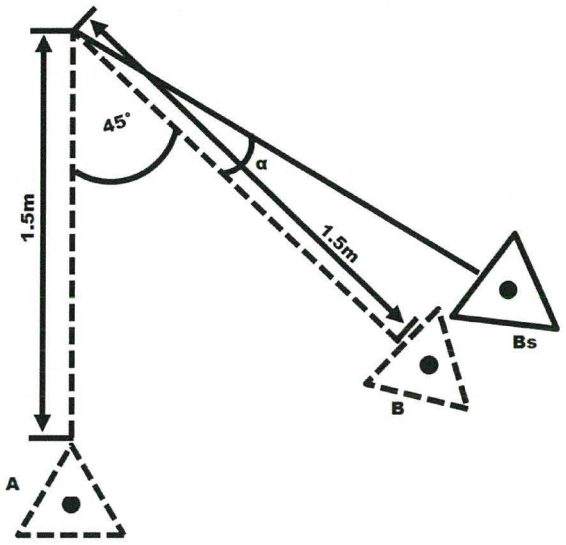


Figure 7.19 test of the mobile robot turning a sharp angle (45 degree)

The c programming for this test is presented in Appendix C.6. The test is done in Aghort Hall in Massey Turitea campus. Figure 7.20 is an image of the mobile robot in the 45 degree angle turning test.

Figure 7.20 45° sharp angle turning test

Result of the two tests – The error angles in both tests are ranged in 2 degree to 6 degree. The percentages of the error occupies in the turning angles are presented in Figure 7.21.

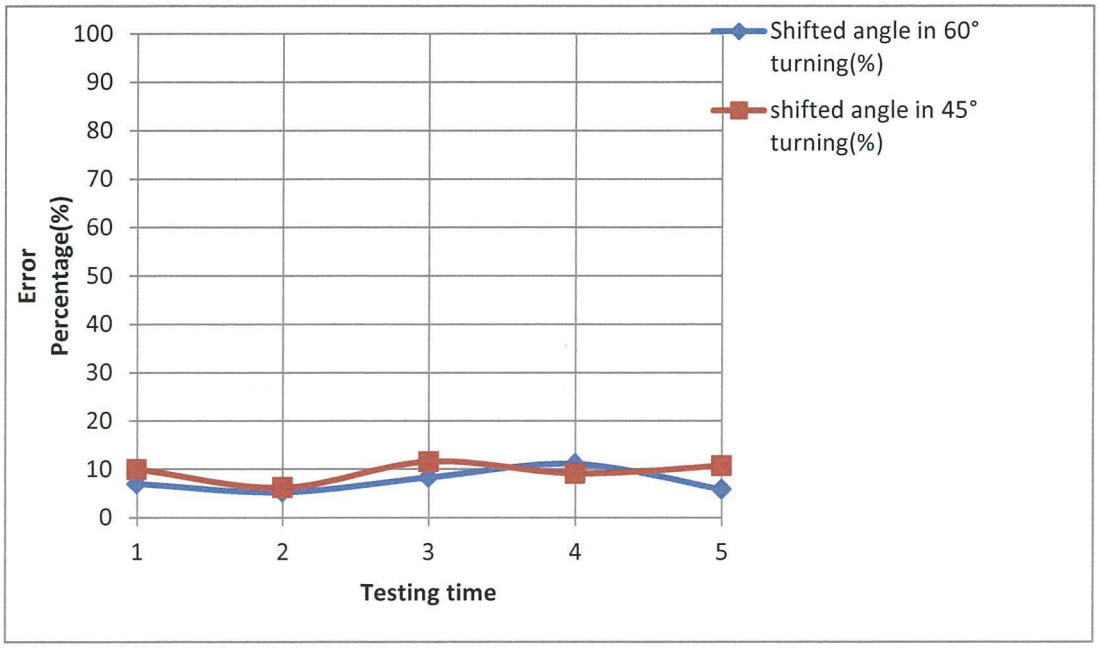


Figure 7.21 results of the sharp angle turning test

In Figure 7.21, the red and blue dots represent the percentages of error in 45 and 60 degree angle turning test respectively. It can be seen that the error percentages are quite similar between these two tests. The errors are in the range from 2 degree to 6 degree. From the previous linear motion test, it is known that the mobile robot generate errors in its linear motion. Thus, in this test, it can be assumed that most of the errors occurred in the mobile robot's 1.5 meters linear motion. This test, once again presents the great maneuverability of the omni-wheeled mobile robot, and a relatively good moving accuracy in its rotational motions.

7.2.5 Test of a real life simulation

Purpose of the test – In the previous tests, the basic functions of the three-omni-wheeled mobile robot have been examined. This test then simulates a real life mobile robot application which travels in a room with a given track. The test requires the mobile robot to perform most

of its movements, which are straight running, rotating, sliding, curving, and angle turning. This test design is mainly to examine the maneuverability of the three-omni-wheeled mobile robot, errors of shifted displacements and angles in the test are expected. The anticipation for the mobile robot is that it will complete the given track and reach the given ending position with minor displacement and angle shifts.

Test design – The test is done in the Mechatronics lab in Massey turitea campus. Figure 7.22 demonstrates the test design.

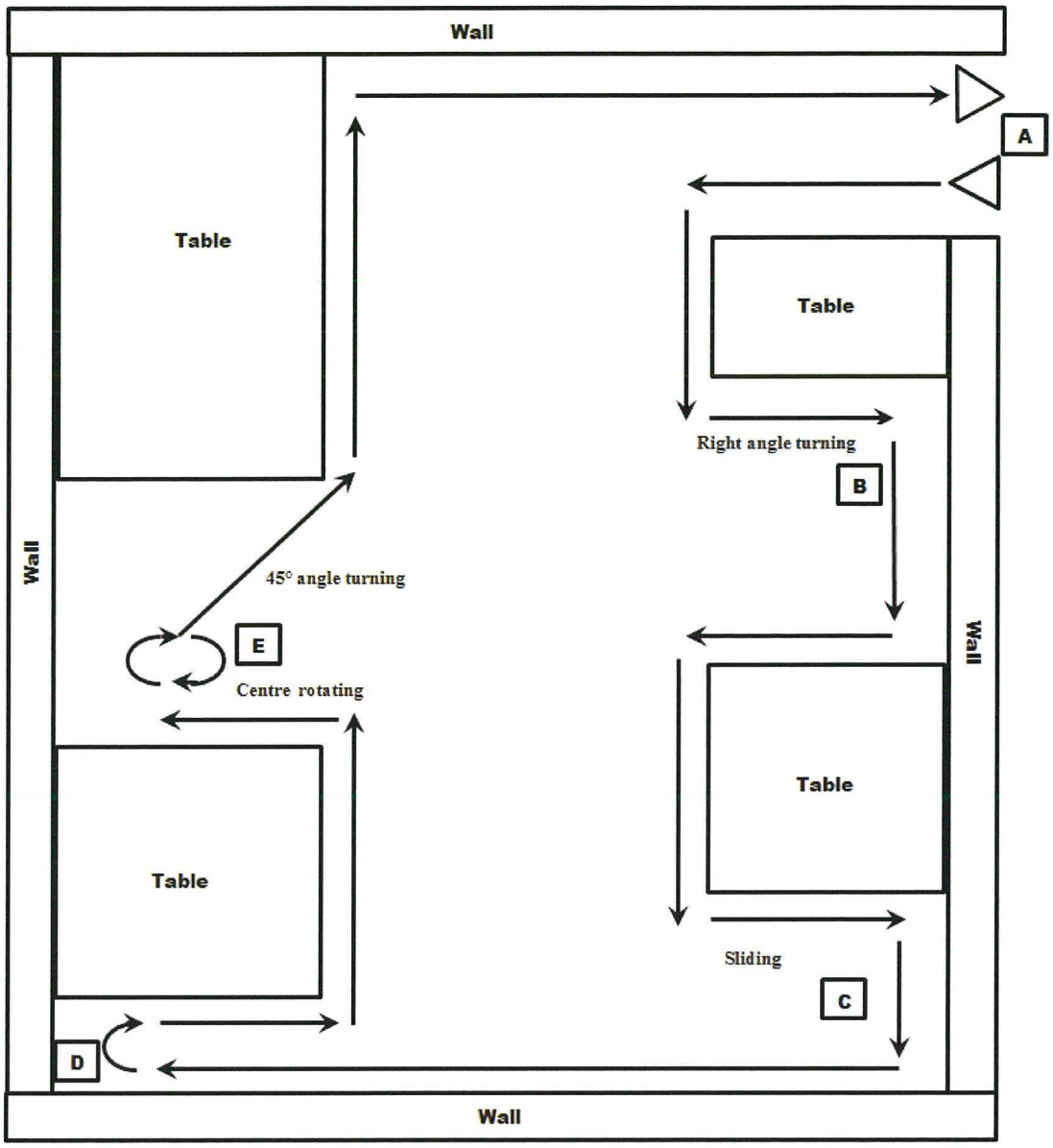


Figure 7.22 in room test

In Figure 7.22, the solid lines represent the tracks that the mobile robot is told to follow. The mobile robot firstly departs from position A and turns a couple of right angles to the section B. Section B is designed to test the right angle turning motion of the mobile robot. The mobile robot needs to turn a few right angles to travel out of section B. Secondly when the mobile robot reaches section C, it will slide into the section C and move forward to the bottom of the map. After that the mobile robot slides out of section C and turn 90 degree, pointing its head

toward section D. The mobile robot will move into Section D and meet a dead end. There is limited space in section D for the mobile robot to perform a U turn, thus the mobile robot needs to perform a 180 degree angle rotation, turn its head back and come out from section D. Then the robot keeps travelling to section E. It will perform a 360 degree center rotation and then come out from section E with a 45 degree angle. After the mobile robot left section E, it will turn a 45 degree angle, pointing its head straight up toward the top line. Finally the mobile robot will reach the top line and follow this solid line back to its initial position A as the completion of the test. Figure 7.23 is an image of the mobile robot's different performances in the test. The c programming for this test is presented in Appendix

Right angle turning

Sliding

180 degree turning

Center rotating

Figure 7.23 performance of the mobile robot in the simulation test

Test result

The test was done for 10 times. For most of the times the mobile robot generated different errors at different sections. In some sections, very little errors were generated, and some other sections, relatively big errors occurred. When the mobile robot was on its linear motions, it generated greater errors than its rotational motions do. The rough ground surface caused the errors as well. However, when separately tested the mobile robot at different sections, the mobile robot was able to complete its tasks. This proved that the mobile robot is capable of completing all of the motions that are required in this simulation test. Replacing better omni-wheels and find a smooth testing ground for the mobile robot would help it reduce the errors greatly.

The mobile was separately tested 5 times at each section. When the mobile robot travelled through different sections, it had different performances.

Section B – the robot completed the tasks of moving in and out of section B. Due to the rough surface of the testing ground, when the mobile robot was performing right angle turning, relatively big errors were generated. The mobile robot went out the section B with a shifted angle and displacement every time of the test.

Section C – the mobile robot performed a left sliding motion into the section C, and a forward moving down to the bottom of the map, and then slid of section C. The testing ground at section C has much smoother surface than section B, the result of this section was better. Smaller error was generated in this section.

Section D – the mobile robot moved into section D and met the dead end. Then it turned 180 degree and came out of section D. The 180 degree turning motion of the mobile robot was very accurate, and the surface of the ground was in a good condition. Thus the performance of the mobile robot in section D was excellent. There was only an average tiny error generated at section D.

Section E – the mobile robot reached section E and performed a 360 degree center rotation. The rotational movement was very accurate. The ground surface at section E was smooth. After the rotation was completed, the mobile robot turned 45 degree and moved out of section E. The 45 degree sharp turn was accurate. Thus the performance of the mobile robot was great at section E. After the mobile robot left section E, it was able to move back to its initial position A. however, some noticeable errors were generated since the mobile robot was on its linear motion for a long distance from section E to position A.

7.3 conclusion

The 5 tests were successfully carried out. The three-omni-wheeld mobile robot was proved to be highly maneuverable in completing most of the tasks in these test designs. The mobile robot's rotational motions have greater accuracy over its linear motions. The mobile robot is able to perform sliding, right angle turning, sharp angle turning and center rotating motions, which the mobile robots with normal wheels are either not able to do so or do so with difficulty. These motions give the robot a high freedom of flexibility in accomplishing complex tasks.

The main factors that affected the moving accuracy of the mobile robot are the unevenness of the testing ground, the lack of accuracy of the omni-wheel's mechanical structure, and the unbalanced torque provision from the motors.

Chapter 8 Conclusion and Future Development

8.1 Conclusion

In summary, this research has shown the process of building an omni-directional mobile robot and the test on its maneuverability and moving accuracy. The steps in the mobile robot design process are listed here:

Firstly, this research has shown a brief history of the development of mobile robots. The mobile robot structure and different drives were introduced. These were:

- Single drive
- Differential drive
- Tracked robot
- Ackerman steering
- Omni-directional drive

The advantages and drawbacks were discussed, and omni-directional drive was selected for this project over the other drives.

Then the mobile robotic control methodology was introduced. These were:

- On-off control
- PID control
- Other control methods such as fuzzy control and neutral network control

The On-off control was chosen for the mobile robot control system, due to its simplicity.

Modular design was discussed as it was applied to the mobile robot base build. The modular design in the mobile robot base gives the mobile robot a great ability to integrate into other applications. I.e. adding a wireless communication device in its control system and a camera on the top of its head will make it an exploring robot, it can then travel into a confined space, and draw a map of the space or transport items to certain destinations. The modular design makes the size of the mobile robot base to be easily changed, so that the base module can be used for other projects that have the limitation of size required for the robot design. The control system can be simply replaced by changing the control circuit board. When adding a new control system to this mobile robot base module, the mobile robot can achieve many complicated tasks with great performances in maneuverability. The above features make the mobile robot base module a great value to people who are interested in the study of omni-directional mobile robotic systems.

Secondly, the omni-directional wheels were presented. Two main types of omni-directional wheels were discussed, which are:

- Mecanum wheel
- Omni-wheel

The omni-wheel was selected for this project due to it is economic applicable, and also its simplicity in mechanical structure. The force analysis of three-omni-wheeld and four omni-wheeled mobile robots were presented. The three-omni-wheeld structure was chosen for this project because it relatively cost saving.

Thirdly the mechanical system of the mobile robot base was designed. The material selection was introduced and aluminum was chosen for this project since it has good density and is cost

effective. The calculation of the mobile robot's weight and speed were shown, and thus the torque that is required from the motors was calculated. The motor selection was done.

After the selections of the mobile robot's mechanical components were complete, the mobile robot's electric and electronic components were selected, these were:

- Power regulator
- Microcontroller
- Sharp sensor
- Stepper motor driver

The power regulator formed the power circuit and provides power to the motors and microcontroller. The microcontroller controls the motors and receives data from the sensors. The sensors collect information and transmit it to the microcontroller. The stepper motor driver works with the microcontroller, controls the performance of the stepper motors. After the component selections were done, the electric and electronic system was designed. The final schematic and PCB of the mobile robot's electronic system were shown.

Then the microcontroller functions and C programming were presented. These were:

- Timers and counters
- PWM
- I2C
- ADC

Timers and counters are the basic functions in microcontrollers; timers can be used to generate PWM. I2C is the microcontroller's communication function. It helps the microcontroller to communicate with its peripheral devices. ADC function converts the

analogue signals from the sensors to digital, so that the microcontroller can process the data which is collected from the sensors. PWM is used for the stepper motor control. There are different ways to generate PWM with the timers in the microcontroller ATmega32, which were:

- CTC mode
- Fast PWM mode
- Phase and frequency correct mode

Phase and frequency correct mode was used to generate PWM in this project, since it allows the frequency and duty circle of the PWM to be changed during the motor's motions. In addition, it is more accurate than the other modes.

The stepper motor drive mode was introduced, which were:

- Wave drive mode
- Full step drive mode
- Half step drive mode
- Micro-step drive mode

Each stepper motor drive mode has its advantages and drawbacks. The half step drive mode was selected, mainly because of its driving effectiveness and power saving feature.

Lastly, five tests were designed to examine the maneuverability and moving accuracy of the three-omni-wheeled mobile robot. These were:

- Linear motion test –10 meters straight track
- Rotational motion test – 1. Center rotating test; 2. Circling test

- Square-shaped track test
- Sharp angle turning test – 1. 60 degree sharp angle turning; 2. 45 degree sharp angle turnning
- In room simulation test

The tests have proved that the three-omni-wheeld mobile robot has great maneuverability. The mobile robot is able to perform motions that other mobile robots with normal wheels in build are either not able to do so or do so with difficulty. Motions like sliding, center rotating, curving, sharp angle turning are what gain the mobile robot a great favor in the design of maneuverable robotics system. The maneuverability tests were succeeded and the results were collected and analyzed. The results show that the mobile robot has different degree of accuracy when performing different motions. When the robot is in its linear motion, i.e. moving straight forward or backward, it is likely to generate errors in terms of displacement or angle shift from the track that the mobile robot was told to follow. On the other hand, the performances of the mobile robot's rotational motions are great, only minor errors were generated. The errors are mainly due to the lack of accuracy of the omni-wheel's mechanical structure and the unbalanced torque provision from the motors. The smoothness of the testing surface affects the mobile robot's performances as well. The omni-wheels generate a lot of errors when they are running on rough surfaces.

8.2 Improvement and future development

8.2.1 Robot mechanic components

a. Materials selection for the mobile robot base design

In this project, aluminum was chosen for building the mobile robot base design. Aluminum is relatively heavy compared to other materials. The heavy body weight of the robot puts on a great burden for the motors and the batteries. Selecting lighter materials can reduce the torque provided by the motors and the current drawn from the batteries. The HDPE material is generally lighter than aluminum and metal materials. Applying HDPE for the mobile robot can reduce its body weight greatly; accordingly, the size of the motors and batteries can be reduced as well, in other words, smaller and cheaper motors and batteries can be selected. Therefore, applying HDPE for the mobile robot could reduce the cost largely.

b. Omni-wheel selection

The omni-wheel that was chosen for this project is an affordable module; the lack of accuracy in its mechanical structure is a big drawback. Because of its cheap price, this module is mainly used for educational uses. It can help students to get start with omni-directional mobile robotic system design. Therefore, selecting a module of omni-wheel which has a better mechanical structure that provides a greater accuracy in its linear motions can improve the performance of the mobile robot in a noticeable way. The drawback is that the omni-wheels with more accurate mechanical structure are generally quite expensive.

c. Microcontroller

The microcontrollers that is used for this project is ATmega32. It has three timers which are timer 0, timer 1, and timer 2. Timer 0 and timer 2 are just 8-bit timers; only timer 1 is a 16-bit

timer. It is known that 16-bit timers have greater resolutions than 8-bit timers. Thus timer 1 is much more accurate in generating the PWM signals with the "Phase and Frequency Correct" mode selected. In addition, the 16-bit timer is capable of changing the speed of the motor in the "Phase and Frequency Correct" mode, while timer 0 and timer 2 couldn't. In this project, since the mobile robot has low speed requirement (torque is the primary demand for the motors), the speed of the motors are kept constantly through the test, so that using timer 0 and timer 2 to control the motors are just fine. However, in real life applications, the mobile robots are likely to be told for a change in speed, in this case, selecting a better microcontroller which has 3 timers with 16-bit resolutions or using three ATmega32 chips can achieve the speed control.

8.2.2 Mobile robot control system

a. Control method

In this project, the simplest control method "on-off" control was selected. When the sensor detects an obstacle that is close to the mobile robot, the microcontroller will stop the mobile robot from hitting it. The problem of using an "on-off" control is that, there is a delay between the sensor detecting an obstacle and the mobile robot responding to it. To eliminate this delay, a higher control method is required. PID control can be added on the robot. It will eliminate the delays and increase the moving accuracy of the mobile robot. Some other control systems can also be added on to the mobile robot's control system if necessary.

b. Wireless communication

A wireless communication can be established between the microcontroller and a computer. The robot can then be controlled with a computer keyboard, a console or a cell phone with the

wireless connections. The wireless device "XBee" is affordable and powerful, which can be added on the mobile robot as a communication tool.

8.2.3 Electronic system

a. PCB modular design

In this project, the PCB board was made as the whole module. It can be subdivided into a few smaller modules, such as the power supply module, motor controller module, microcontroller module. With the modular design added in, the mobile robot's electronic system would become more stable. If any problems on the circuit occurred, each module can be taken off and examined, so that the problems would be easily found solved. Moreover, replacing or upgrading a circuit module would be much easier.

8.2.4 The appearance of the robot

A cover can be designed for the mobile robot so that it will look more adorable to people. A platform can be built on the top of the mobile robot to carry items, such as a camera or a laptop.

Reference

Ali, F., Amran, A. C., & Kawamura, A. (7-10 Dec, 2010). *Bipedal robot walking strategy on inclined surfaces using position and orientation based inverse kinematics algorithm.* Paper presented at the Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference; *pp. 181 -186.*

Bräunl, T. (2006). Mobile robot design and applications with embedded systems. *Embedded Robotics.* Springer-Verlag Berlin Heidelberg 2003, 2006, Germany; *P.1.*

Da Costa, A. L., Ceonceicao, A. G. S., Cerqueira, R. G., & Ribeiro, T. T. (18-20 Feb, 2013). *Omnidirectional mobile robots navigation: A joint approach combining reinforcement learning and knowledge-based systems.* Paper presented at the Biosignals and Biorobotics Conference (BRC), 2013 ISSNIP; *pp. 1-6.*

Do Nascimento, T. P., Da Costa, A. L., & Paim, C. C. (22-25 Sept, 2009). *AxeBot Robot the Mechanical Design for an Autonomous Omnidirectional Mobile Robot.* Paper presented at the Electronics, Robotics and Automotive Mechanics Conference, CERMA '09; *pp. 187-192.*

Hongjun, Kim, & Byung Kook, Kim. (7-12 Oct, 2012). *Minimum-energy trajectory planning and control on a straight line with rotation for three-wheeled omni-directional mobile robots.* Paper presented at the Intelligent Robots and Systems (IROS); *pp. 3119-3124.*

Jantapremjit, P., & Austin, D. (2001). *Design of a Modular Self-Reconfigurable Robot.* Paper presented at the Australian Conference on Robotics and Automation, Sydney; *pp. 38 – 43.*

Jung Won, K., Bong Sung, K., & Myung-Jin, C. (14-17 Oct, 2008). *Development of omni-directional mobile robots with mecanum wheels assisting the disabled in a factory environment.* Paper presented at the Control, Automation and Systems, 2008. ICCAS 2008. International Conference; *pp.2070 – 2075.*

Kanjanawanishkul, K., & Zell, A. (12-17 May, 2009). *Path following for an omnidirectional mobile robot based on model predictive control.* Paper presented at the Robotics and Automation, 2009. ICRA '09. IEEE International Conference; *pp. 3341-3346.*

Kiriki, T., Kimuro, Y., & Hasegawa, T. (1999). *A 4-legged mobile robot control to observe a human behavior.* Paper presented at the Robot and Human Interaction, 1999. RO-MAN '99. 8th IEEE International Workshop; *pp. 195 – 200.*

Nehmzow, U. (2003). *Mobile Robotics: A Practical Introduction* (2nd ed.). Springer-varlag, London; *pp. 1 - 2.*

Niemueller, T., & Widyadharma, S. (2003). An Introduction to Robotics. *Artificial Intelligence.* Chair of Computer Science V, Knowledge-based Systems Group. RWTH Aachen; *P.1.*

Rojas, R. (2004). *A short history of omnidirectional wheels.* Retrieved Jan, 11, 2012 from: http://robocup.mi.fu-berlin.de/buch/shortomni.pdf.

Salih. J, Rizon. M, Yaacob. S, Adom.A.H, & Mamat.M.R. (2006). Designing Omni-Directional Mobile Robot with Mecanum Wheel. *American Journal of Applied Sciences, 3*(5); *pp 1-4.*

Selvakumar. M, Selvaraj. P, Sivaprasath. V, Vishnu. R, & Subramanian. R. (May, 2013). An efficient control of BLDC motor using PID controller and genetic algorithm. *discovery Engineering, 1*(2); *pp. 2-13.*

Vukosavljev, S. A., Kukolj, D., Papp, I., & Kovacevic, V. (5-6 Sept, 2011). *A Generic Embedded Robot Platform for Real Time Navigation and Telepresence Abilities.* Paper presented at the Engineering of Computer Based Systems (ECBS-EERC), 2011 2nd Eastern European Regional Conference; *pp. 54 – 60.*

Walter, G. (1950). *An Electromechanical Animal*, Dialectica (online publication). *Vol. (4) , Issue (3); pp. 206 - 213.*

Weidong, Wang, Lei, Zhou, Zhijiang, Du, & Lining, Sun. (2008, 2-5 July 2008). *Track-terrain interaction analysis for tracked mobile robot.* Paper presented at the Advanced Intelligent Mechatronics, 2008. AIM 2008. IEEE/ASME International Conference; *pp. 126-131.*

Weinstein, A. J., & Moore, K. L. (2010, 14-17 March 2010). *Pose estimation of Ackerman steering vehicles for outdoors autonomous navigation.* Paper presented at the Industrial Technology (ICIT), 2010 IEEE International Conference; *pp. 579-584.*

Yanto, G. XiaoLei, Y., & Bowling, A. (April 26, 2004 - May 1, 2004). *A navigable six-legged robot platform.* Paper presented at the Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference. *Volume: 5; P.5106.*

Appendix

Appendix A Datasheet

Appendix A.1 Datasheet of the voltage regulator LM340

Order this document by LM340/D

MOTOROLA

LM340, A Series

Three-Terminal Positive Fixed Voltage Regulators

This family of fixed voltage regulators are monolithic integrated circuits capable of driving loads in excess of 1.0 A. These three–terminal regulators employ internal current limiting, thermal shutdown, and safe–area compensation. Devices are available with improved specifications, including a 2% output voltage tolerance, on A–suffix 5.0, 12 and 15 V device types.

Although designed primarily as a fixed voltage regulator, these devices can be used with external components to obtain adjustable voltages and currents. This series of devices can be used with a series–pass transistor to boost output current capability at the nominal output voltage.

- Output Current in Excess of 1.0 A
- No External Components Required
- Output Voltage Offered in 2% and 4% Tolerance*
- Internal Thermal Overload Protection
- Internal Short Circuit Current Limiting
- Output Transistor Safe–Area Compensation

THREE–TERMINAL POSITIVE FIXED VOLTAGE REGULATORS

SEMICONDUCTOR TECHNICAL DATA

T SUFFIX
PLASTIC PACKAGE
CASE 221A

Pin 1. Input
2. Ground
3. Output

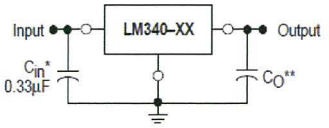
Heatsink surface is connected to Pin 2.

ORDERING INFORMATION

Device	Output Voltage and Tolerance	Operating Temperature Range	Package
LM340T–5.0	5.0 V ± 4%	T_J = 0° to +125°C	Plastic Power
LM340AT–5.0	5.0 V ± 2%		
LM340T–6.0	6.0 V ± 4%		
LM340T–8.0	8.0 V ± 4%		
LM340T–12	12 V ± 4%		
LM340AT–12	12 V ± 2%		
LM340T–15	15 V ± 4%		
LM340AT–15	15 V ± 2%		
LM340T–18	18 V ± 4%		
LM340T–24	24 V ± 4%		

* 2% regulators are available in 5, 12 and 15 V devices.

Simplified Application



A common ground is required between the input and the output voltages. The input voltage must remain typically 1.7 V above the output voltage even during the low point on the input ripple voltage.

XX these two digits of the type number indicate voltage.
* C_{in} is required if regulator is located an appreciable distance from power supply filter.
** C_O is not needed for stability; however, it does improve transient response. If needed, use a 0.1 µF ceramic disc.

Appendix A.2 Microcontroller ATmega 32 datasheet

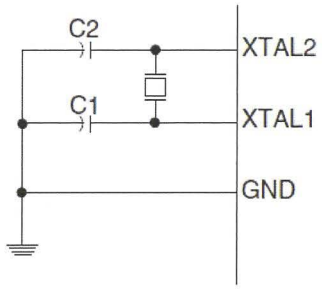
Features

- High-performance, Low-power Atmel®AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 32Kbytes of In-System Self-programmable Flash program memory
  - 1024Bytes EEPROM
  - 2Kbytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    - 8 Single-ended Channels
    - 7 Differential Channels in TQFP Package Only
    - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V for ATmega32L
  - 4.5V - 5.5V for ATmega32
- Speed Grades
  - 0 - 8MHz for ATmega32L
  - 0 - 16MHz for ATmega32
- Power Consumption at 1MHz, 3V, 25°C
  - Active: 1.1mA
  - Idle Mode: 0.35mA
  - Power-down Mode: < 1µA

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in Figure 12. Either a quartz crystal or a ceramic resonator may be used. The CKOPT Fuse selects between two different Oscillator amplifier modes. When CKOPT is programmed, the Oscillator output will oscillate will a full rail-to-rail swing on the output. This mode is suitable when operating in a very noisy environment or when the output from XTAL2 drives a second clock buffer. This mode has a wide frequency range. When CKOPT is unprogrammed, the Oscillator has a smaller output swing. This reduces power consumption considerably. This mode has a limited frequency range and it can not be used to drive other clock buffers.

For resonators, the maximum frequency is 8MHz with CKOPT unprogrammed and 16 MHz with CKOPT programmed. C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for choosing capacitors for use with crystals are given in Table 4. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 12. Crystal Oscillator Connections



The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1 as shown in Table 4.

Table 4. Crystal Oscillator Operating Modes

CKOPT	CKSEL3..1	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
1	101[1]	0.4 - 0.9	–
1	110	0.9 - 3.0	12 - 22
1	111	3.0 - 8.0	12 - 22
0	101, 110, 111	1.0 ≤	12 - 22

Note: 1. This option should not be used with crystals, only with ceramic resonators.

GP2Y0A21YK0F

Distance Measuring Sensor Unit
Measuring distance: 10 to 80 cm
Analog output type

■Description

GP2Y0A21YK0F is a distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector) , IRED (infrared emitting diode) and signal processing circuit.
The variety of the reflectivity of the object, the environmental temperature and the operating duration are not influenced easily to the distance detection because of adopting the triangulation method.
This device outputs the voltage corresponding to the detection distance. So this sensor can also be used as a proximity sensor.

■Features

1. Distance measuring range : 10 to 80 cm
2. Analog output type
3. Package size : 29.5×13×13.5 mm
4. Consumption current : Typ. 30 mA
5. Supply voltage : 4.5 to 5.5 V

■Agency approvals/Compliance

1. Compliant with RoHS directive (2002/95/EC)

■Applications

1. Touch-less switch
   (Sanitary equipment, Control of illumination, etc.)
2. Robot cleaner
3. Sensor for energy saving
   (ATM, Copier, Vending machine)
4. Amusement equipment
   (Robot, Arcade game machine)

■Block diagram

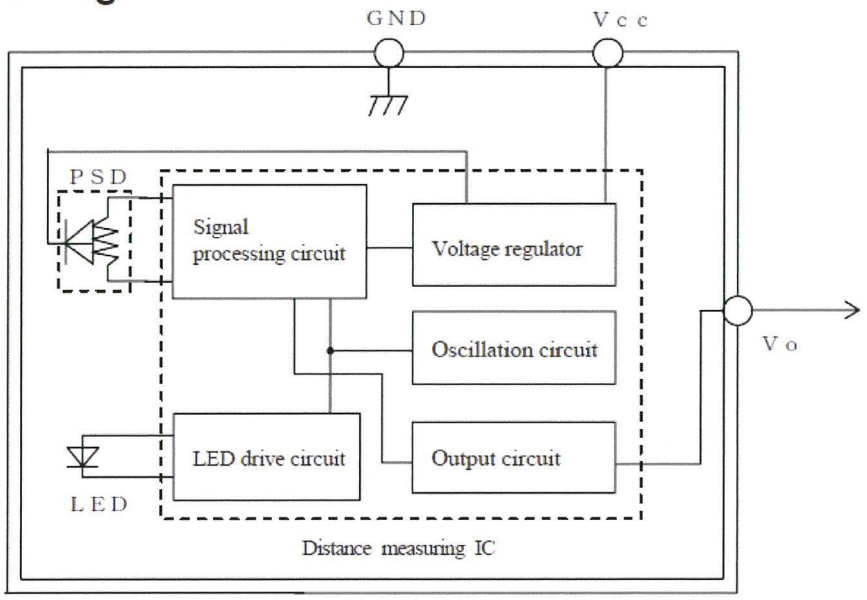
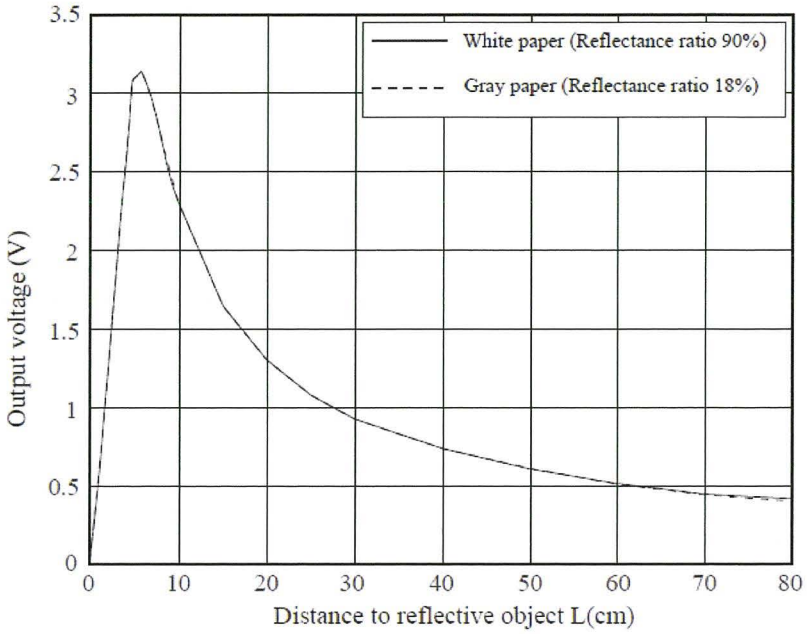
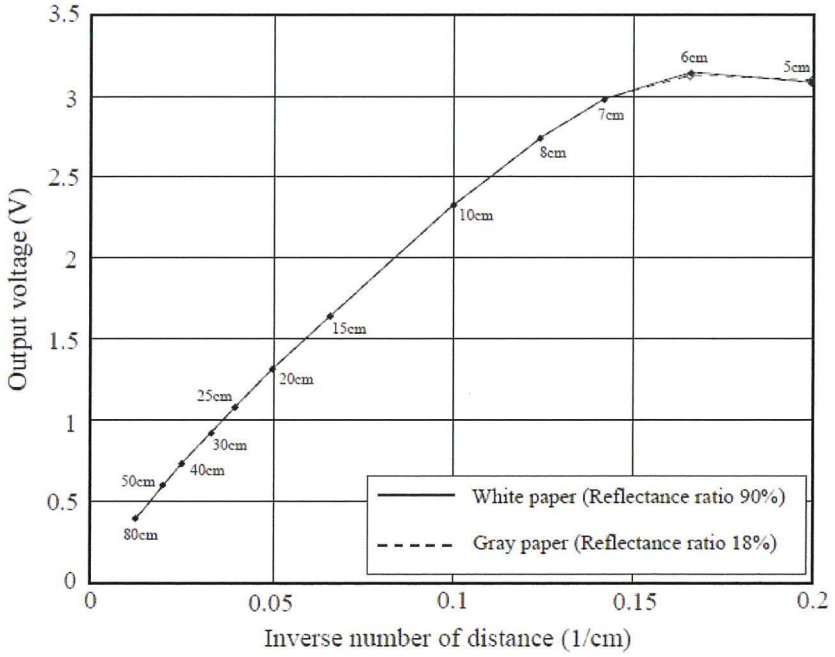


Fig. 2 Example of distance measuring characteristics(output)





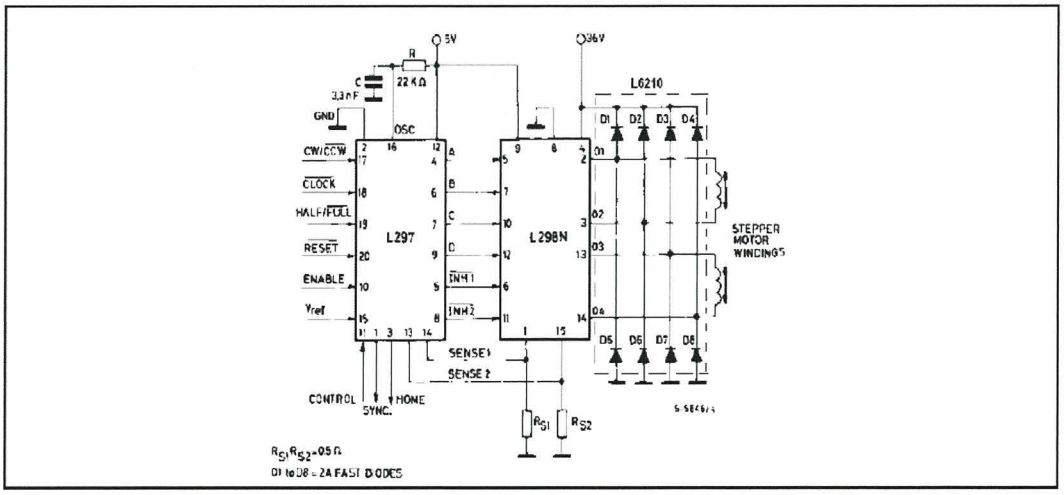
DESCRIPTION

The L297/A/D Stepper Motor Controller IC generates four phase drive signals for two phase bipolar and four phase unipolar step motors in microcomputer-controlled applications. The motor can be driven in half step, normal and wawe drive modes and on-chip PWM chopper circuits permit switch-mode control of the current in the windings. A feature of this device is that it requires only clock, direction and mode input signals. Since the phase are generated internally the burden on the microprocessor, and the programmer, is greatly reduced. Mounted in DIP20 and SO20 packages, the L297 can be used with monolithic bridge drives such as the L298N or L293E, or with discrete transistors and darlingtons.

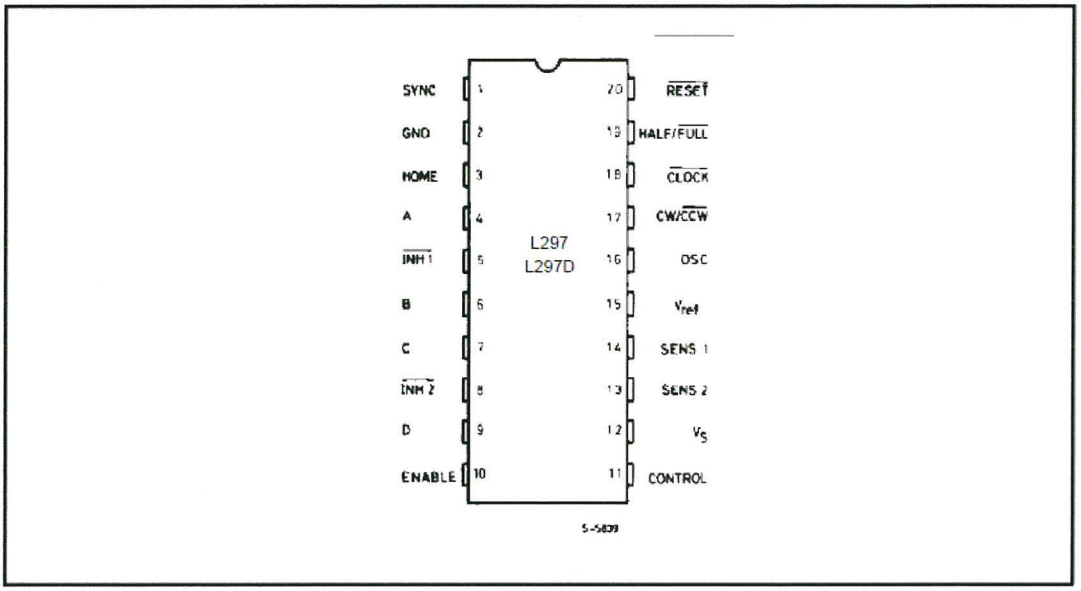
ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_s	Supply voltage	10	V
V_i	Input signals	7	V
P_{tot}	Total power dissipation (T_{amb} = 70°C)	1	W
T_{stg}, T_j	Storage and junction temperature	-40 to + 150	°C

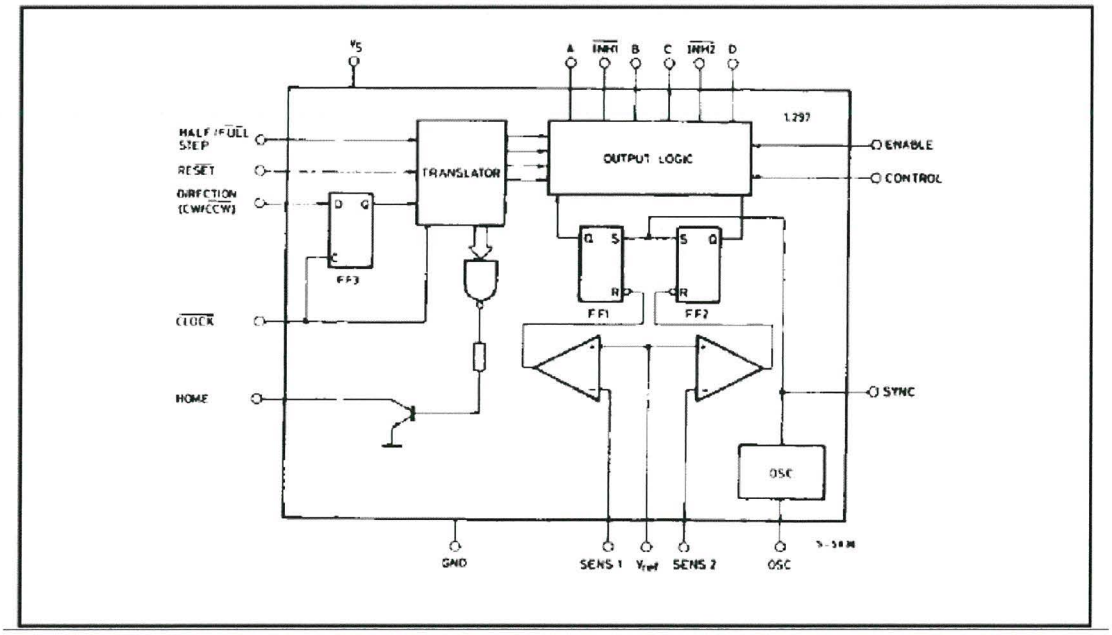
TWO PHASE BIPOLAR STEPPER MOTOR CONTROL CIRCUIT



PIN CONNECTION (Top view)



BLOCK DIAGRAM (L297/L297D)



PIN FUNCTIONS - L297/L297D

N°	NAME	FUNCTION
1	SYNC	Output of the on-chip chopper oscillator. The SYNC connections The SYNC connections of all L297s to be synchronized are connected together and the oscillator components are omitted on all but one. If an external clock source is used it is injected at this terminal.
2	GND	Ground connection.
3	HOME	Open collector output that indicates when the L297 is in its initial state (ABCD = 0101). The transistor is open when this signal is active.
4	A	Motor phase A drive signal for power stage.
5	$\overline{INH1}$	Active low inhibit control for driver stage of A and B phases. When a bipolar bridge is used this signal can be used to ensure fast decay of load current when a winding is de-energized. Also used by chopper to regulate load current if CONTROL input is low.
6	B	Motor phase B drive signal for power stage.
7	C	Motor phase C drive signal for power stage.
8	$\overline{INH2}$	Active low inhibit control for drive stages of C and D phases. Same functions as INH1.
9	D	Motor phase D drive signal for power stage.
10	ENABLE	Chip enable input. When low (inactive) INH1, INH2, A, B, C and D are brought low.
11	CONTROL	Control input that defines action of chopper. When low chopper acts on INH1 and INH2; when high chopper acts on phase lines ABCD.
12	V_s	5V supply input.
13	$SENS_2$	Input for load current sense voltage from power stages of phases C and D.
14	$SENS_1$	Input for load current sense voltage from power stages of phases A and B.
15	V_{ref}	Reference voltage for chopper circuit. A voltage applied to this pin determines the peak load current.
16	OSC	An RC network (R to V_{CC}, C to ground) connected to this terminal determines the chopper rate. This terminal is connected to ground on all but one device in synchronized multi - L297 configurations. f ≅ 1/0.69 RC
17	CW/$\overline{CCW}$	Clockwise/counterclockwise direction control input. Physical direction of motor rotation also depends on connection of windings. Synchronized internally therefore direction can be changed at any time.
18	$\overline{CLOCK}$	Step clock. An active low pulse on this input advances the motor one increment. The step occurs on the rising edge of this signal.

PIN FUNCTIONS - L297/L297D (continued)

N°	NAME	FUNCTION
19	HALF/$\overline{\text{FULL}}$	Half/full step select input. When high selects half step operation, when low selects full step operation. One-phase-on full step mode is obtained by selecting FULL when the L297's translator is at an even-numbered state. Two-phase-on full step mode is set by selecting FULL when the translator is at an odd numbered position. (The home position is designate state 1).
20	$\overline{\text{RESET}}$	Reset input. An active low pulse on this input restores the translator to the home position (state 1, ABCD = 0101).

THERMAL DATA

Symbol	Parameter		DIP20	SO20	Unit
$R_{th\text{-}j\text{-}amb}$	Thermal resistance junction-ambient	max	80	100	°C/W

Appendix A.6 microcontroller datasheet – WGM selection table

Table 47. Waveform Generation Mode Bit Description[1]

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

Appendix A.7 microcontroller datasheet – COM selection table

Table 46. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM [1]

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 14: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match when up-counting. Set OC1A/OC1B on compare match when downcounting.
1	1	Set OC1A/OC1B on compare match when up-counting. Clear OC1A/OC1B on compare match when downcounting.

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. See "Phase Correct PWM Mode" on page 101. for more details.

Appendix A.8 Datasheet of the ADC chip PCF8591

1. General description

The PCF8591 is a single-chip, single-supply low-power 8-bit CMOS data acquisition device with four analog inputs, one analog output and a serial I^2C-bus interface. Three address pins A0, A1 and A2 are used for programming the hardware address, allowing the use of up to eight devices connected to the I^2C-bus without additional hardware. Address, control and data to and from the device are transferred serially via the two-line bidirectional I^2C-bus.

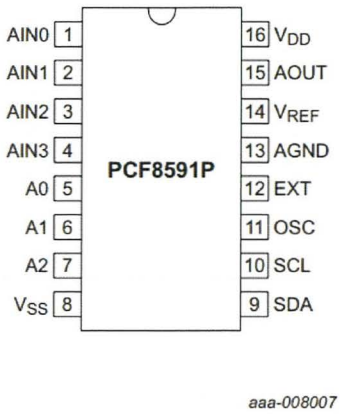
The functions of the device include analog input multiplexing, on-chip track and hold function, 8-bit analog-to-digital conversion and an 8-bit digital-to-analog conversion. The maximum conversion rate is given by the maximum speed of the I^2C-bus.

2. Features and benefits

- Single power supply
- Operating supply voltage 2.5 V to 6.0 V
- Low standby current
- Serial input and output via I^2C-bus
- I^2C address selection by 3 hardware address pins
- Max sampling rate given by I^2C-bus speed
- 4 analog inputs configurable as single ended or differential inputs
- Auto-incremented channel selection
- Analog voltage range from V_{SS} to V_{DD}
- On-chip track and hold circuit
- 8-bit successive approximation A/D conversion
- Multiplying DAC with one analog output.

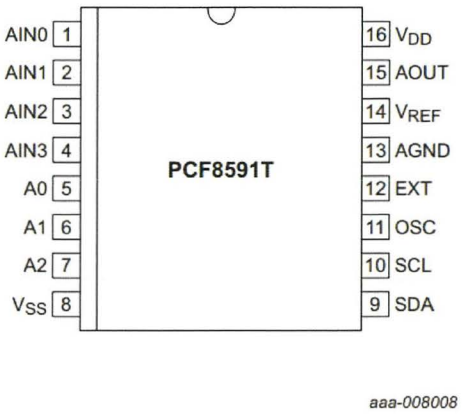
3. Applications

- Supply monitoring
- Reference setting
- Analog control loops



Top view. For mechanical details, see Figure 22 on page 21.

Fig 2. Pin configuration for PCF8591P (DIP16)



Top view. For mechanical details, see Figure 23 on page 22.

Fig 3. Pin configuration for PCF8591T (SO16)

7.2 Pin description

Table 4. Pin description

Symbol	Pin	Description
AIN0	1	analog inputs (A/D converter)
AIN1	2	
AIN2	3	
AIN3	4	
A0	5	hardware slave address
A1	6	
A2	7	
V_{SS}	8	ground supply voltage
SDA	9	I^2C-bus serial data input and output
SCL	10	I^2C-bus serial clock input
OSC	11	oscillator input/output
EXT	12	external/internal switch for oscillator input
AGND	13	analog ground supply
V_{REF}	14	voltage reference input
AOUT	15	analog output (D/A converter)
V_{DD}	16	supply voltage

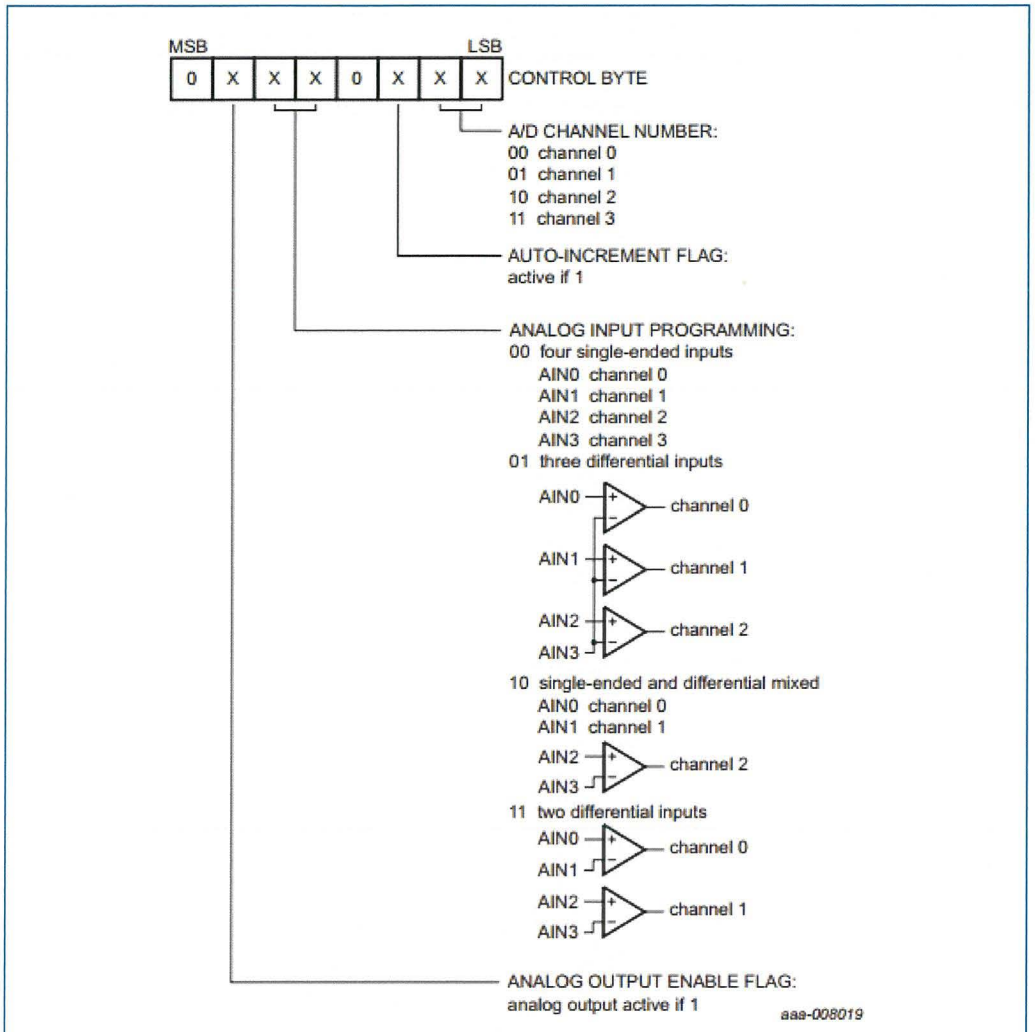
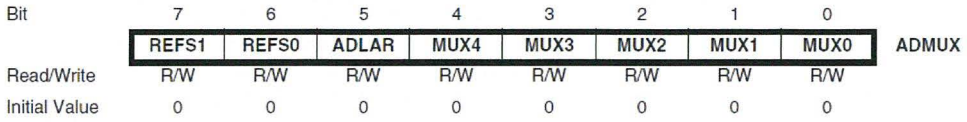


Fig 4. Control byte

Appendix A.9 ADC – reference selection table



Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 83. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 83. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Appendix A.10 ADC – channel and gain selection table

Table 84. Input Channel and Gain Selections

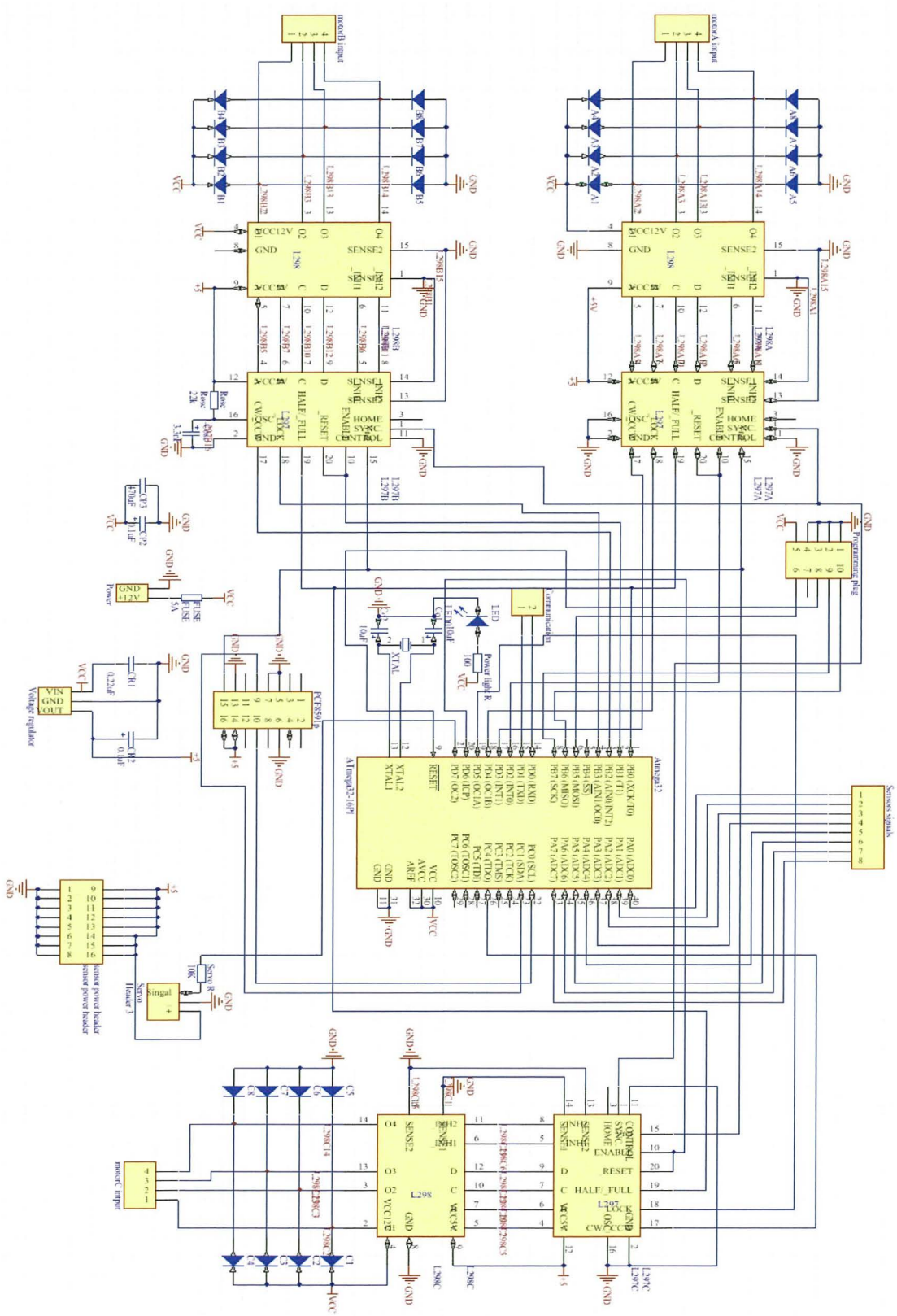
MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000		ADC0	ADC0	10x
01001		ADC1	ADC0	
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	
10000	N/A	ADC0	ADC1	1x
10001		ADC1	ADC1	
10010		ADC2	ADC1	
10011		ADC3	ADC1	
10100		ADC4	ADC1	
10101		ADC5	ADC1	
10110		ADC6	ADC1	
10111		ADC7	ADC1	
11000		ADC0	ADC2	
11001		ADC1	ADC2	
11010		ADC2	ADC2	
11011		ADC3	ADC2	
11100		ADC4	ADC2	

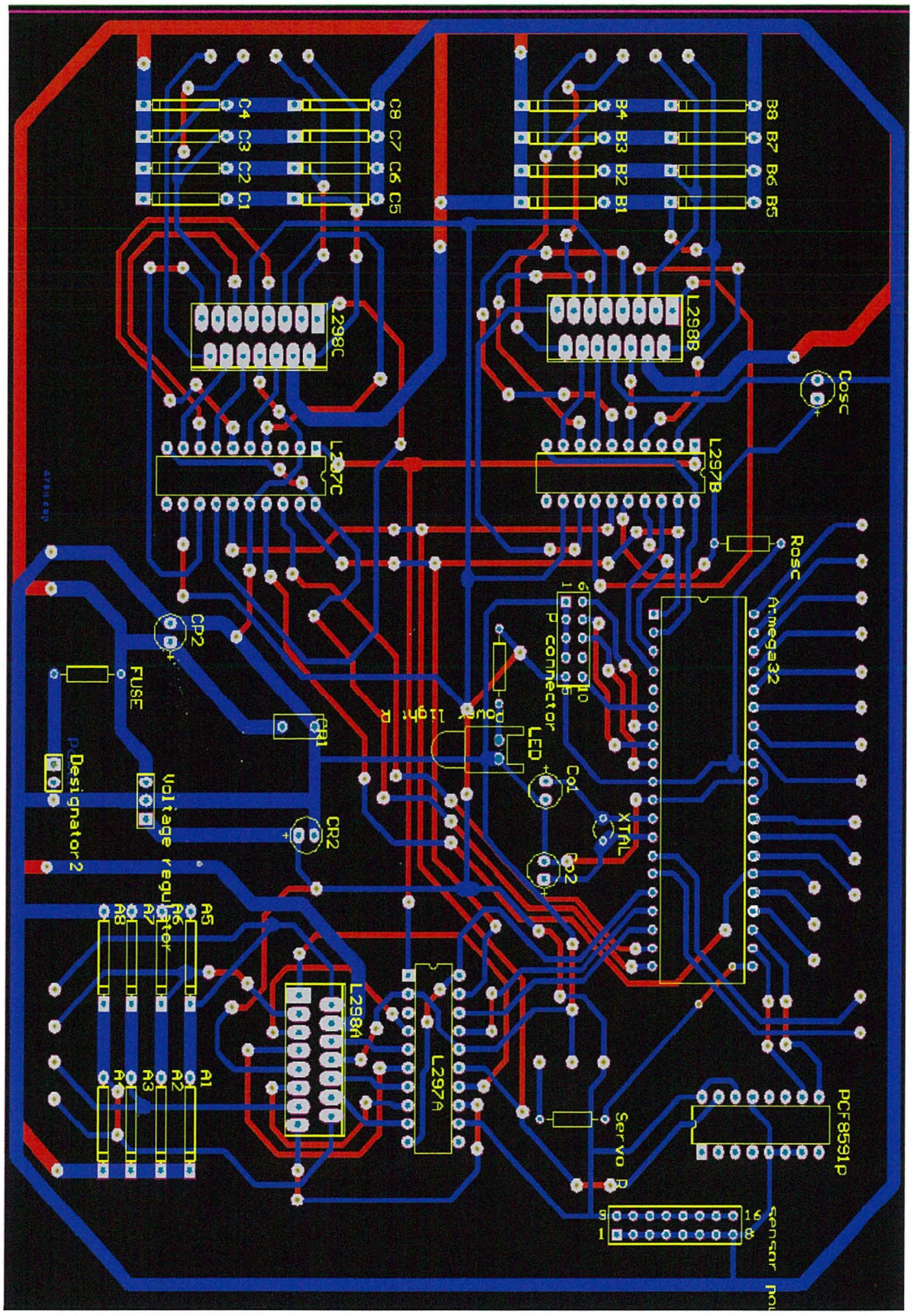
Appendix A.11 ADC – prescaler selection table

Table 85. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Appendix B schematic and PCB




C8 C7 C6 C5
C4 C3 C2 C1
B8 B7 B6 B5
B4 B3 B2 B1
L298C
L298B
L297C
L297B
Cosc
Rosc
Atmega32
P connector
LED
Co1
XTAL
CO2
CR2
CR1
FUSE
Designator2
Voltage regulator
L298A
L297A
PCF8591p
Servo
A5
A6
A7
A8
A1
A2
A3
A4

Appendix C Microcontroller programming

Appendix C.1 programming for linear motion test.

Initialize the speed of the robot

```
/*Half step mode selection*/
PORTB |= 0x01; //Set PORTB0=1 for DriverA,B&C all chose HALF mode;

/*initiate timer1*/
ICR1 = 0x3E8; // set top as 1000 (the speed of the robot will be 2.5res/sec)
OCR1B = 500; // set compare match as 500 for OC1B (duty cycle will be always set to
50%)

TCCR1A |= 0x00;
TCCR1B |= _BV(WGM13); //select the waveform mode as phase and frequency correct
mode
TCCR1A |= _BV(COM1A1)|_BV(COM1A0)|_BV(COM1B1)|_BV(COM1B0);
//switch on the compare output mode for compare units A & B,which enables the
OC1A&B
//and this will set OCR1A&B on compare match while upcounting, and clear them when
counting down
TCCR1B |= _BV(CS11); // start the timer1 and select the prescaler to 8
                        // this gives out a 1000Hz PWM to the motor ,make it run at 2.5rps
}
```

This part of the programming is for initializing the timer 1 in the microcontroller Atmega23. It selected the timer 1 to run in the Phase and Frequency Correct mode. By setting the register ICR1 value to 0x7D0 which is 1200 in decimal, the speed of each motor on the robot can then be set as 2.5 revolusions per second (rps).

The speed setting for the motors are kept the same for all tests, since under this speed, the output torque from the motors run the mobile robot smoothly.

Motion control

```
/*-------------------------------------moving forward----------------------------------*/
void moving_forward()
{
//Select rotating directions
PORTD |= 0x08; //Set PORTD2=0 MotorA Counter-clockwise
PORTB &= ~0x04;    //Set PORTD2=1 MotorB clockwise
//PORTC &= ~0x10; //Set PORTC4=1 MotorC Counter-clockwise
//Enable MotordriverB&C
PORTD |= 0x04; //Enable driver A (PD2)
PORTB |= 0x02; //Enable driver B (PB1)
//PORTD |= 0x40; //Enable driver C (PD6)
return;
}
void stop_forward()
{
PORTB &= ~0x02; //Disable driver B (PB1)
PORTD &= ~0x40; //Disable driver C (PD6)
return;
}
_delay_ms(5000);
_delay_ms(5000);
stop_forward();
return;}
```

```
/*-------------------------------------------------------------------------------------------
----------------------------------Motor movement testing-------------------------------
-------------------------------------------------------------------------------------------*/
void movement_test()
{
//First step: moving forward
//Select rotating directions
moving_forward();
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(5000);
stop_forward();
return;}
```

Appendix C.2 programming for the center rotating test

```
/*---------------------------------rotating clockwise---------------------------------*/
void rotating_clockwise()
{
//Select rotating directions
PORTD |= 0x08; //Set PORTD3=1 MotorA clockwise
PORTB |= 0x04; //Set PORTB2=1 MotorB clockwise
PORTC |= 0x10; //Set PORTC4=1 MotorC clockwise
//Enable MotordriverA,B&C
PORTD |= 0x04; //Enable driver A (PD2)
PORTB |= 0x02; //Enable driver B (PB1)
PORTD |= 0x40; //Enable driver C (PD6)
return;
}
void stop_rotatingCW()
{
PORTD &= ~0x04; //Disable driver A (PD2)
PORTB &= ~0x02; //Disable driver B (PB1)
PORTD &= ~0x40; //Disable driver C (PD6)
return; }

/*-----------------------------------------------------------------------------------------
----------------------------------Motor movement testing---------------------------------
-----------------------------------------------------------------------------------------*/
void movement_test()
{/*----------------------rotating clockwise----------------------*/
moving_forward();
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(5000);
stop_rotatingCW();
return;
}
```

AppendixC.3 programming for the circling test

```
/*----------------------------------circling clockwise---------------------------------*/
void circling_clockwise()
{
//Select rotating directions
PORTD |= 0x08;    //Set PORTD3=1 MotorA clockwise
PORTB |= 0x04;    //Set PORTB2=1 MotorB clockwise
PORTC &= ~0x10; //Set PORTC4=1 MotorC counter-clockwise
//Enable MotordriverA,B&C
PORTD |= 0x04; //Enable driver A (PD2)
PORTB |= 0x02; //Enable driver B (PB1)
PORTD |= 0x40; //Enable driver C (PD6)
return;
}
void stop_circlingCW()
{
PORTD &= ~0x04; //Disable driver A (PD2)
PORTB &= ~0x02; //Disable driver B (PB1)
PORTD &= ~0x40; //Disable driver C (PD6)
return;
}

/*-------------------------------------------------------------------------------------------
-----------------------------------Motor movement testing-----------------------------------
-------------------------------------------------------------------------------------------*/
void movement_test()
{
/*----------------------circling test----------------------*/
Circling_clockwise();
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(3500);
stop_circlingCW();
return;
}
```

Appendix C.4 programming for the square-shaped track following test

```
/*--------------------------------turnning to the right---------------------------------*/
void turnning_right()
{
//Select rotating directions(Step1:turnning 90degree to the right)
PORTD |= 0x08; //Set PORTD3=1 MotorA clockwise
PORTB |= 0x04; //Set PORTD2=1 MotorB clockwise
PORTC |= 0x10; //Set PORTC4=1 MotorC clockwise
//Enable MotordriverA,B&C
PORTD |= 0x04; //Enable driver A (PD2)
PORTB |= 0x02; //Enable driver B (PB1)
PORTD |= 0x40; //Enable driver C (PD6)
_delay_ms(1600);//Turn 90 degree to the right
PORTD &= ~0x04; //Disable driver A (PD2)
PORTB &= ~0x02; //Disable driver B (PB1)
PORTD &= ~0x40; //Disable driver C (PD6)
_delay_ms(500);

//Select rotating directions(Step2:moving forward)
moving_forward();
return;
}

void stop_turnningR()
{
stop_forward();
return;
}
```

```
/*-------------------------------------------------------------------------------------------
-----------------------------------Motor movement testing-----------------------------------
-------------------------------------------------------------------------------------------*/
void movement_test()
{
/*----------------------square shape----------------------*/
//First step: moving forward
//Select rotating directions
moving_forward();
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(3500);
stop_forward();
_delay_ms(500);

//second step: turnning to the right
turnning_right();
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(3500);
stop_turnningR();
_delay_ms(500);

//Third step: turning to the right
turnning_right();
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(3500);
stop_turnningR();
_delay_ms(500);

//Fourth step: turning to the right
turnning_right();
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(3500);
stop_turnningR();
_delay_ms(500);

//Fifth step: turnning to the right
turnning_right();
stop_turnningR();
_delay_ms(5000);
return;
}
```

Appendix C.5 programming for the mobile robot turning sharp angle (60 degree)

```
/*--------------------------------------angle turnning----------------------------------*/
void angle_selection(unsigned char angle)
{
switch (angle)
   {
    case 0:    //turnning 0 degree
    _delay_ms(0);
    break;
    case 30: //turnning 30 degree
    _delay_ms(533);
    break;
    case 45: //turnning 45 degree
    _delay_ms(800);
    break;
    case 60: //turnning 60 degree
    _delay_ms(1067);
    break;
    case 90: //turnning 90 degree
    _delay_ms(1600);
    break;
        case 120: //turnning 120 degree
    _delay_ms(2133);
    break;
    case 135: //turnning 135 degree
    _delay_ms(2400);
    break;
    case 180: //turnning 180 degree
    _delay_ms(3200);
   }
return;
}
```

```
void turnning_angleCCW()
{
//Select rotating directions
PORTD &= ~0x08; //Set PORTD3=1 MotorA counter-clockwise
PORTB &= ~0x04; //Set PORTB2=1 MotorB counter-clockwise
PORTC &= ~0x10; //Set PORTC4=1 MotorC counter-clockwise
//Enable MotordriverA,B&C
PORTD |= 0x04; //Enable driver A (PD2)
PORTB |= 0x02; //Enable driver B (PB1)
PORTD |= 0x40; //Enable driver C (PD6)

angle_selection(angle);

PORTD &= ~0x04; //Disable driver A (PD2)
PORTB &= ~0x02; //Disable driver B (PB1)
PORTD &= ~0x40; //Disable driver C (PD6)
return;
}

/*----------------------60 degree sharp turn---------------
void movement_test()
{
//----------------------turning to the left-----------------------
//Step 1: moving forward
moving_forward();
_delay_ms(5000);
_delay_ms(5000);
stop_forward();
_delay_ms(1000);
//Step 2: turnning 60 degree
angle = 60;
turnning_angleCCW();
_delay_ms(1000);
//Step 3: moving forward
moving_forward();
_delay_ms(5000);
_delay_ms(5000);
stop_forward();
return;
}
```

Appendix C.6 programming for the mobile robot turning sharp angle (45 degree)

```
void turnning_angleCCW()
{
//Select rotating directions
PORTD &= ~0x08; //Set PORTD3=1 MotorA counter-clockwise
PORTB &= ~0x04; //Set PORTB2=1 MotorB counter-clockwise
PORTC &= ~0x10; //Set PORTC4=1 MotorC counter-clockwise
//Enable MotordriverA,B&C
PORTD |= 0x04; //Enable driver A (PD2)
PORTB |= 0x02; //Enable driver B (PB1)
PORTD |= 0x40; //Enable driver C (PD6)

angle_selection(angle);

PORTD &= ~0x04; //Disable driver A (PD2)
PORTB &= ~0x02; //Disable driver B (PB1)
PORTD &= ~0x40; //Disable driver C (PD6)
return;
```

```
/*-----------------------45 degree sharp turn----------------
void movement_test()
{
//-----------------------turning to the left-----------------------
//Step 1: moving forward
moving_forward();
_delay_ms(5000);
_delay_ms(5000);
stop_forward();
_delay_ms(1000);
//Step 2: turnning 45 degree
angle = 45;
turnning_angleCCW();
_delay_ms(1000);
//Step 3: moving forward
moving_forward();
_delay_ms(5000);
_delay_ms(5000);
stop_forward();
return;
}
```

Appendix C.7 programming for the in room simulation test

```
/*-------------------------------------------------------------------------------
----------------------------------Motor movement testing-------------------------
-------------------------------------------------------------------------------*/
/*--------------------------------Position A-------------------------------------*/
void movement_test()
{
moving_forward();
_delay_ms(5000);
stop_forward();
_delay_ms(1000);


turnning_left();
_delay_ms(5000);
_delay_ms(1000);
stop_turnningL();
_delay_ms(1000);
```

```
//-------------------------------position B-----------------------------------//
turnning_left();
_delay_ms(4000);
stop_turnningL();
_delay_ms(1000);

turnning_right();
_delay_ms(5000);
_delay_ms(1400);
stop_turnningR();
_delay_ms(1000);



turnning_right();
_delay_ms(4500);
stop_turnningR();
_delay_ms(1000);

turnning_left();
_delay_ms(5000);
_delay_ms(5000);
stop_turnningL();
_delay_ms(1000);
```

```
//--------------------------------position C-------------------------------------//
sliding_left();
_delay_ms(4000);
stop_slidingL();
_delay_ms(1000);

moving_forward();
_delay_ms(3000);
stop_forward();
_delay_ms(1000);

sliding_right();
_delay_ms(5000);
stop_slidingR();
_delay_ms(1000);

turnning_right();
_delay_ms(5000);


stop_turnningR();
_delay_ms(1000);
```

```
//------------------------------position D-----------------------------------//
angle = 180;
turnning_angleCW();
_delay_ms(1000);

moving_forward();
_delay_ms(3000);
stop_forward();
_delay_ms(1000);

turnning_left();
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(5000);
stop_turnningL();
_delay_ms(1000);


turnning_left();
_delay_ms(3000);
stop_turnningL();
_delay_ms(1000);

rotating_clockwise();
_delay_ms(3200);
_delay_ms(3200);
stop_rotatingCW();
_delay_ms(1000);

angle = 135;
turnning_angleCW();
_delay_ms(1000);

moving_forward();
_delay_ms(5000);
stop_forward();
_delay_ms(1000);
```

```
angle = 45;
turnning_angleCCW();
_delay_ms(1000);

moving_forward();
_delay_ms(5000);
_delay_ms(3000);
stop_forward();
_delay_ms(1000);

turnning_right();
_delay_ms(5000);
_delay_ms(5000);
_delay_ms(1000);
stop_turnningR();
----------*/
return;
}
```

Appendix C.8 code for ADC

```
#include <avr/io.h>
#include <util/delay.h>
#include <inttypes.h>

void portonoff(uint8_t portnum);
void initADC0(void); // prescalar of 2 and enable ADC
void LEDon(uint8_t lednum);
void turnalloff(void);
uint8_t ReadAnalog(void); //returns analog voltage read

int count[10];
uint8_t lowsign;
uint8_t analogV[30]; // stores the analor voltage read
int analogfilt; // stores the analog voltage read
int lowfilt; // stores the analog voltage read
uint8_t ledon;
```

```
int main(void)
{
    DDRD |= 0xFF; // PD0-7 will be output
    DDRC |= 0x03; // PC0,1 will be output
    DDRA &= ~0x01; // PA0 will be input
    PORTD |= 0xFF; // PD0-7 will be high
    PoRTC |= 0x03; // PC0,1 will be high

    analogfilt = 0x00;
    lowfilt = 0x00;
    int i;
    int j;
    int numofaverages;// collect a certain number of data and average the value
    initADC0();
    numofaverages = 6;// average 6 values
    while (1)
    {   analogfilt = 0;
        lowfilt = 0;
        for (i = 0; i < 10;i++) { //reset counts to 0
            count[i] = 0;
        }
        for (i = 0; i < numofaverages; i++) {
            analogV[i] = ReadAnalog();
        }
        for (j = 0; j < numofaverages; j++) {

            for (int ImpBit=0; ImpBit<4;ImpBit++) {

                if (analogV[j] & (1 << (ImpBit + 4) ) )
                        { // If that bit is high
                    count[ImpBit]++;    // Increment counter
                }
            }
        }

        }
```

```
    if (count[0] > 3) { // flashing enough
            PORTD &= ~0x01;
        }
        else {
            PORTD |= 0x01;
        }

        if (count[1] > 3) { // flashing enough
            PORTD &= ~0x02;
        }
        else {
            PORTD |= 0x02;
        }

        if (count[2] > 3) { // flashing enough
            PORTD &= ~0x04;
else {
            PORTD |= 0x04;
        }

        if (count[3] > 1) { // flashing enough
            PORTD &= ~0x08;
        }
        else {
            PORTD |= 0x08; //turn off LED
        }

    }
    return 0;
}
```

```
void turnalloff(void) {
	PORTD |= 0x4E;
}

void portonoff(uint8_t portnum) {
	_delay_ms(500);
	PORTD &= ~portnum; //turn on LED
	_delay_ms(500);
	PORTD |= portnum; // turn off LED
}

void LEDon(uint8_t lednum) {
	PORTD &= ~lednum;
}

void initADC0(void) {
	ADCSRA=0X85; // Prescalar Division factor of 2 and enable ADC
}

uint8_t ReadAnalog(void)
{
	ADMUX = 0xE0;		//Left Adjust
	ADCSRA |= 0x40; // start conversion
	while (ADCSRA & (0x40));// loop until conversion end
	lowsign = ADCL>>6;
	return ADCH;

}
```

Appendix D Matlab programming

Appendix D.1 analysis for the result of the linear motion test

```
X=[15.8 34.2 21.2 -5.3 12.2 -15.8 23.6 -26.3 29.3 35.4 10.8 20.9 -9.4 26.9 18.2 22.8
-10.2 -8.4 -6.3 26.5 30.4 13.4 16.2 -6.4 13.7 12.9 10.9 22.6 28.7 19.7]'
Y = [5.6 9.8   8.2 -6.4 -8.3 -2.5 -8.6 3.6 15.8 10.9 -12.5 -6.9 9.2 5.8 5.4 -13.6 8.5 12.1
15.9 -8.5 9.6 9.4 -2.8 -6.3 5.9 15.4 -3.8 -11.1 5.6 2.2]'
MX=mean(X)
MY=mean(Y)
SDX=std(X)
SDY=std(Y)
CIX=[MX-(1.96*SDX/sqrt(30)) MX+(1.96*SDX/sqrt(30))]
CIY=[MY-(1.96*SDY/sqrt(30)) MY+(1.96*SDY/sqrt(30))]
%when z equals to 1.96, the confidence level is 95%
%plotting the perfect position B in this test, since the straight route is 10meters long, and
in the plotting unit of this chart is cm, so the position B will be:
plot(0,1000,'x')
hold on

%plotting the error bars
plot([CIX(1) CIX(2)],[MY+1000 MY+1000],'-') %X-error bars
plot([MX MX],[CIY(1)+1000 CIY(2)+1000],'-') %Y-error bars

%plotting the error area
plot([CIX(1) CIX(2)],[CIY(2)+1000 CIY(2)+1000],'r--') %top red line
plot([CIX(2) CIX(2)],[CIY(1)+1000 CIY(2)+1000],'r--') %R Red line
plot([CIX(2) CIX(1)],[CIY(1)+1000 CIY(1)+1000],'r--') %B red line
plot([CIX(1) CIX(1)],[CIY(1)+1000 CIY(2)+1000],'r--') %L red line

%changing the scalars of the plot for a better observation
axis([CIX(1)-15 CIX(2)+15 CIY(1)+10-5 CIY(2)+10+5])
```

Appendix D.2 displacement shift analysis of the center rotating test

```
X= [10.2  13.5  8.4  -6.9  -9.2  -9.1  2.8  -8.5  -1.6  2.8  6.4  -8.2  7.6  6.6
-5.3  3.4  -7.6  5.8  8.9  3.5  -4.9  5.6  5.8  -6.6  -5.9  8.9  7.1  6.2  -9.3
12.5]'
Y= [5.6  -9.8  8.2  -6.4  -8.3  -2.5  -8.6  3.6  15.8  10.9  -12.5  -6.9  9.2
-5.8  5.4  -13.6  8.5  12.1  -15.9  -8.5  9.6  9.4  -2.8  -6.3  5.9  15.4  -3.8
-11.1  5.6  2.2] '
MX=mean(X)
MY=mean(Y)
SDX=std(X)
SDY=std(Y)
%Use equation 7-1
CIX=[MX-(1.96*SDX/sqrt(30)) MX+(1.96*SDX/sqrt(30))]
CIY=[MY-(1.96*SDY/sqrt(30)) MY+(1.96*SDY/sqrt(30))]
%plotting the stop position B, which should be (0,0)
plot(0, 0,'x')
hold on
plot([CIX(1) CIX(2)],[MY MY],'-') %X-error bars
plot([MX MX],[CIY(1) CIY(2)],'-') %Y-error bars
plot([CIX(1) CIX(2)],[CIY(2) CIY(2)],'r--') %top red line
plot([CIX(2) CIX(2)],[CIY(1) CIY(2)],'r--') %Right Red line
plot([CIX(2) CIX(1)],[CIY(1) CIY(1)],'r--') %Bottom red line
plot([CIX(1) CIX(1)],[CIY(1) CIY(2)],'r--') %Left red line

%Changing the scalars of the axis
axis([CIX(1)-15 CIX(2)+15 CIY(1)+10-5 CIY(2)+10+5])
```

Appendix D.3 angle shift analysis of the center rotating test

```
a = [8.2  13.5  8.4  -6.9  -9.2  -9.1  2.8  -8.5  -11.6  12.8  6.4  -8.2  -17.6
6.6  -5.3  13.4  -7.6  5.8  8.9  13.5  -4.9  5.6  5.8  -16.6  -5.9  -8.9  7.1  6.2
-9.3  12.5]% Using "a" represents the shifted angles
Ma=mean(a)
SDa=std(a)
%Use equation 7-1
CIa=[Ma-(1.96*SDa/sqrt(30)) Ma+(1.96*SDa/sqrt(30))]

%plotting the stop position B, which should be (0,0)
plot(0, 0,'x')
hold on
plot([CIa(1) CIa(2)],[0 0],'-') %a-error bars
```

Appendix D.4 shifted displacement analysis of the circling test

```
X= [-13.6  8.5  12.1  -15.9  -8.5  9.6  9.4  -2.8  -6.3  5.9  15.4  -3.8  -11.1
5.6  2.2  11.2  -8.6  -6.2  9.5  6.8  -5.4  -6.5  8.2  6.1  -9.1  -6.7  7.2  4.9
-6.8  -9.6]'
Y= [13.4  -7.6  5.8  8.9  13.5  -4.9  5.6  5.8  -16.6  -5.9  -8.9  7.1  6.2  -9.3
12.5  8.2  6.1  -9.1  -6.7  7.2  4.9  -6.8  -9.6  11.2  -8.6  -6.2  9.5  6.8  -5.4
-6.5] '
MX=mean(X)
MY=mean(Y)
SDX=std(X)
SDY=std(Y)
%Use equation 7-1
CIX=[MX-(1.96*SDX/sqrt(30)) MX+(1.96*SDX/sqrt(30))]
CIY=[MY-(1.96*SDY/sqrt(30)) MY+(1.96*SDY/sqrt(30))]
%plotting the stop position B, which should be (0,0)
plot(0, 0,'x')
hold on
plot([CIX(1) CIX(2)],[MY MY],'-') %X-error bars
plot([MX MX],[CIY(1) CIY(2)],'-') %Y-error bars
plot([CIX(1) CIX(2)],[CIY(2) CIY(2)],'r--') %top red line
plot([CIX(2) CIX(2)],[CIY(1) CIY(2)],'r--') %Right Red line
plot([CIX(2) CIX(1)],[CIY(1) CIY(1)],'r--') %Bottom red line
plot([CIX(1) CIX(1)],[CIY(1) CIY(2)],'r--') %Left red line

%Changing the scalars of the axis
axis([CIX(1)-5 CIX(2)+5 CIY(1) -5 CIY(2) +5])
```

Appendix E I2C configuration and code

Appendix E.1 I2C configuration

The Ermicroblog (Microcontrollers and Electronics project blog) presents the information of I2C: "

I2C (read as I Squared C) bus first introduced by Philips in 1980, because of its simplicity and flexibility the I2C bus has become one of the most important microcontroller bus system used for interfacing various IC-devices with the microcontroller. The I2C bus use only 2 bidirectional data lines for communicating with the microcontroller and the I2C protocol specification can support up to 128 devices attached to the same bus. Today many I2C IC-devices available on the market such as Serial EEPROM, I/O Expander, Real-Time Clock, Digital to Analogue Converter, Analogue to Digital Converter, Temperature Sensor and many more. The I2C protocol use master and slave method, the master which is usually the microcontroller while the slave can be any I2C devices such as Serial EEPROM, I/O Expander or even another microcontroller. All of these devices connected to the I2C bus; one for the serial data called SDA (serial data) and the other for synchronize clock called SCL (serial clock); each of these slave devices has their own individual 7 bits of the address length. Figure E.1shows a prototype of a typical I2C communication.

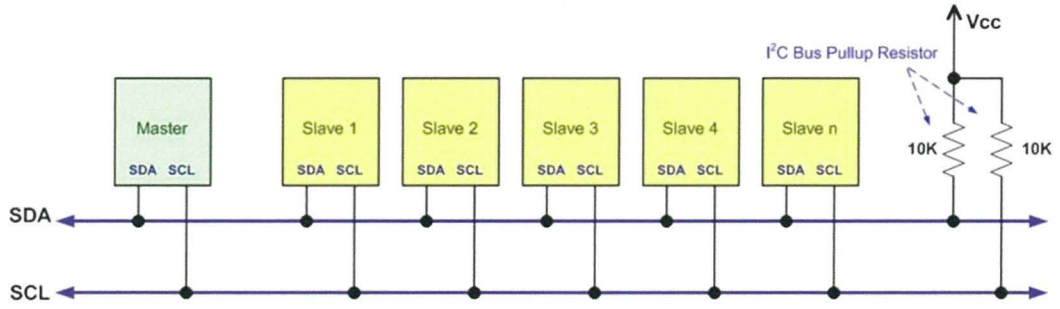


Figure E.1 A typical I2C communication prototype

In a typical I2C, to be able to communicate between masters and slaves, the address finding is the first step to be set. The 7 bits address consists of 4 bits device identification and 3 bits

device physical address. For example the Microchip PCF8591 is selected for this project, the first 4 bits for this device identification is "**1001**" and the last 3 bits could be selected by setting the appropriate address at pins A0, A1 and A2 on the serial EEPROM. Therefore by using these 3 bits we could attach up to 8 Microchip PCF8591 serial EEPROM on the same I2C bus.

To program the I2C, and write the C code, there are certain steps need to be set. The program starts by initializing the Atmega32 ports used and continue with the ADC and the TWI (two wire interfaces) peripherals initiation. This program basically works by checking the identification marks on the serial EEPROM (0b00001111 and 0b1111000). Below is the description of all functions used in this code:

- ✓ i2c_writebyte() function is used to write or store 8-bits data on the I2C devices on specified I2C device address or register
- ✓ i2c_readbyte() function is used to read the 8-bits data on the I2C devices on specified I2C device address or register
- ✓ i2c_transmit() function is called by the two function above to transmit the data to the I2C device.

I2C Bus Clock (SCL) –The I2C-bus used only 2 lines for communicating among the I2C devices; because it use the serial data transfer method therefore the I2C protocol use a clock pulse (SCL) together with the data bits (SDA) for synchronization, each of these data bits is accompanied by the pulse clock on the bus. Setting the **TWPS1 = 0** and **TWPS0 = 0** in the **TWSR** register (prescaler value 1); setting the **TWBR** register to 48 (30 hexadecimal) and with the AVRJazz Mega168 board frequency of 11059200 Hz, we could calculate the SCL frequency as follow:

SCL Frequency = CPU Clock / ((16 + 2(TWBR) x (Prescaler Value)).

SCL Frequency = 11059200 / (16 + 2 x 48) = 98.743 kHz.

The C code below is used to set the SCL frequency:

```
* Initial TWI Peripheral */
WSR = 0x00;    // Select Prescaler of 1
SCL frequency = 11059200 / (16 + 2 * 48 * 1) = 98.743 Khz
WBR = 0x30;    // 48 Decimal
```

I2C Bus Data (SDA) Write Operation – The only thing needs to do is to instruct and read the status of this TWI peripheral; all of the complex arbitration or handshaking between master and slave will be done by TWI peripheral. The following is time diagram Figure E.2 show the explanation of how to store the data to the I2C serial EEPROM:

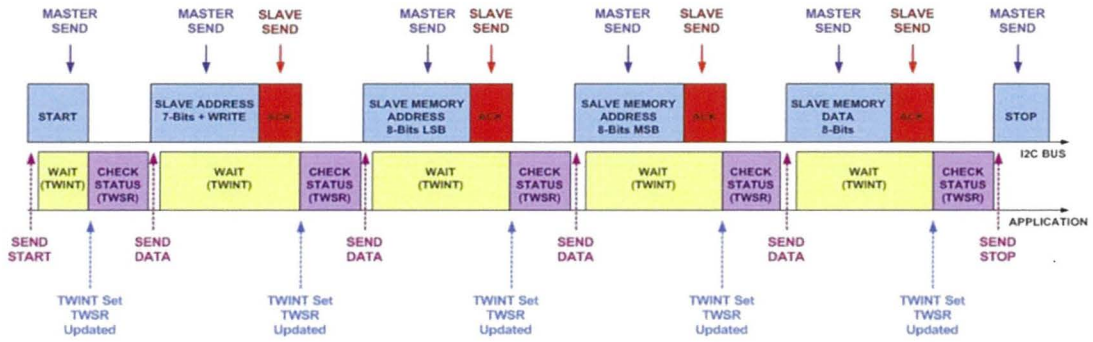


Figure E.2 Typical I2C Serial EEPROM Data Write

From the above Figure we begin the connection by sending the START condition. This START condition is automatically generated by the TWI peripheral inside the AVR microcontroller. By setting **TWINT=1** (interrupt flag bit), **TWSTA=1** (start condition bit) and **TWEN=1** (enable TWI bit), in the **TWCR** register; the TWI will send the START condition to the I2C bus when it available. Then we just wait until the TWI peripheral confirm that it has sent the START signal to the bus; this can be done by waiting for the **TWINT** bit in register **TWCR** to be set (logical "**1**") by the TWI peripheral.

```
// Transmit Start Condition
twi_status=i2c_transmit(I2C_START);
```

The C code in the i2c_transmit() function with I2C_START argument:

```
TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);
// Wait for TWINT=1 in the TWCR Register
while (!(TWCR & (1 << TWINT)));
// Return TWI Status Register, mask the prescaler bits (TWPS1,TWPS0)
return (TWSR & 0xF8);
```

This TWI setting firstly activated the TWI peripheral by setting the **TWINT=1** (logical "**1**") and then waited this interrupt flag to become logical "**1**"; As soon as the TWI peripheral is actived, it will automatically reset this interrupt flag bit to logical "**0**". When it's done the TWI peripheral will set this interrupt flag to logical "**1**". Therefore by using the C while statement and masking the **TWINT** bit in **TWCR** register, it will make the program wait until the **TWINT** bit being set by the TWI peripheral.

The **i2c_transmit()** function is use to encapsulate the repetition process of SEND and WAIT, this function will return the **TWSR** register status with **TWPS1** and **TWPS0** bits being masked. The **TWSR** register status macro has been defined in the **compat/twi.h** include file, therefore using this predefine macro; we could decide whether to continue, retry the process or to quit for error (for the complete explanation of the TWI status register, please refer to the AVR ATMega32).

```
// Check the TWI Status
if (twi_status == TW_MT_ARB_LOST) goto i2c_retry;
if ((twi_status != TW_START) && (twi_status != TW_REP_START)) goto i2c_quit;
```

The next process is to select which I2C device that Atmega32 wants to talk to; this can be done by sending the slave address (4 bits device ID and 3 bits physical address) and the write bit (TW_WRITE = 0, defined in the **compat/twi.h** include file) and wait for the slave response.

```
// Send slave address (SLA_W)
TWDR = (dev_id & 0xF0) | (dev_addr & 0x07) | TW_WRITE;
// Transmit I2C Data
twi_status=i2c_transmit(I2C_DATA);
// Check the TWSR status
if ((twi_status == TW_MT_SLA_NACK) || (twi_status == TW_MT_ARB_LOST)) goto i2c_retry;
if (twi_status != TW_MT_SLA_ACK) goto i2c_quit;
```

Putting the slave address and the write bit to the **TWDR** register and again using the **i2c_transmit()** function with I2C_DATA argument to transmit the data and wait for the status returned by this function. When the selected I2C device response with the acknowledge signal; which mean the I2C slave device acknowledge of the address that sent then the data can be continually sent; first step is to select the memory address and next the data that needs to store in the serial EEPROM device.

After successful sending the data finally the master/slave connection can be closed by sending the STOP condition to the I2C-bus.

```
// Send Stop Condition
twi_status=i2c_transmit(I2C_STOP);
```

The whole I2C-bus write operation C program is implemented in the **i2c_writebyte()** function. " (web page,1)

Appendix E.2 The I2C code for the Vref

The I2C code for Vref is presented as:

```
//*****************************************************
// File Name   : TwoWireControl.c
// Version          : 1.0
// Description : I2c control of ADC/DAC device and USART communication with
middle layer,
//                    and analog voltage reading with driving motors in between
transmission to
//                    middle layer.
//                    Keep driving function changed (23 Aug)
//                    Separate step size for turning and moving forward
//                    On 26 October, accomodation for the initial acceleration was
incorporated
//                    into the stepping forward and stepping turning
// Author      : Samuel Garratt
// Target      : AVR Mega32 Microcontroller
// Compiler         : AVR-GCC 4
// Programmer      : AVRISP
// Last updated : 28 November, 2011

#ifndef F_CPU
#define F_CPU 16000000
#endif

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <compat/twi.h>
```

```
// ************** I2C functions **************
unsigned char i2c_transmit(unsigned char type) {
    switch (type) {
        case I2C_START:     //Send start condition
            TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);
            break;
        case I2C_DATA:      //Send data
            TWCR = (1 << TWINT) | (1 << TWEN);
            break;
        case I2C_STOP:      // Send stop condition
            TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);
            return 0;
    }
    // It is odd how you set TWINT high to start and then wait for it to go high but
    // but actually it goes low as soon as the TWI peripheral becomes active
    // it will automatically set it to low.

    // Wait for TWINT flag set in TWCR register
    while (!(TWCR & (1 << TWINT) ));
    // Return TWI Status Register, mask the prescalar bits (TWPS1, TWPS0)
    return (TWSR & 0xF8);
}
```

```
int i2c_writebyte(unsigned int dev_addr, unsigned int dev_id,
    unsigned int controlbyte, char data) {
    // dev_addr is the number of the particular device
    // dev_id is the unique code for the particular device
    // control byte determines input or output
    // data is the analog voltage to be outputted
    unsigned char n = 0; // number of tries to transmit

    unsigned char twi_status;
    char r_val = -1; // Initialise a failure return value unless successful

i2c_retry:
    if (n++ >= MAX_TRIES) return r_val;

    // Transmit Start Condition
    twi_status = i2c_transmit(I2C_START);

    // Check the TWI Status
    if (twi_status == TW_MT_ARB_LOST) goto i2c_retry;
    // If the status is not start or repeated start then quit
    if ((twi_status != TW_START) && (twi_status != TW_REP_START)) goto
    i2c_quit;
    // Send slave address (SLA_W)
    TWDR = SLA_W; // 0 for write (1 for read)
```

```
    // Transmit I2C data
    twi_status = i2c_transmit(I2C_DATA);
    // Check the TWSR status
    if ((twi_status == TW_MT_SLA_NACK) || (twi_status == TW_MT_ARB_LOST))
goto i2c_retry; // retry if no acknowledgement
    if (twi_status != TW_MT_SLA_ACK) goto i2c_quit; // Quit if still no ack

    // Load the control byte into the data register
    TWDR = controlbyte;
    //Transmit the control byte
    twi_status = i2c_transmit(I2C_DATA);
    if (twi_status != TW_MT_DATA_ACK) goto i2c_quit;
// Load the digital data to be transmitted
    TWDR = data;
    // Transmit Digital data to be converted to analog
    twi_status = i2c_transmit(I2C_DATA);
    if (twi_status != TW_MT_DATA_ACK) goto i2c_quit;

    // TWI Transmit Ok
    r_val = 1;

i2c_quit:
     // Transmit I2C data
     twi_status = i2c_transmit(I2C_STOP);
     return r_val;
}
```

```
void i2c_init(void) {
        TWSR = 0x00; // Select Prescalar as 1 (4^0 = 1)
    // SCL frequency = 16MHz / (16 + 2 * (TWBR) * 4(Prescalar) ) = 0.1MHz.
    TWBR = 0x48;     // 72 Decimal
    // This should give 0.1MHz as DAC can operate at 100kHz
}
int main(void)
{
    int error = 0;
    /* Initialise TWI Peripheral */
    DDRC |= 0x03; // set porc SDA and SCL as onput
    PORTC |= 0x03; // Enable pull ups on SDA and SCL (C0 and C1)

    i2c_init();
    sei(); // enable global interrupt
    error = i2c_writebyte(PCFADC_FIXED, PCFADC_ID, WRITECONTROL,
MotorRefVoltage);

    if (error == -1)
    {
        //error!
    }
    else { // no error
        cli();
          }
    while (1);
return 0;
}
```