I-ROOMBA

By

MOHD AZWAN BIN MOHAMED RASELI

PROJECT DISSERTATION

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

MAY 2011

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

I-ROOMBA

by

Mohd Azwan Bin Mohamed Raseli

A Project Dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

[illegible]

(Dr. Noohul Basheer Zain Ali)
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
May 2011

CERTIFICATION OF ORIGINALITY

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

[illegible]

(Mohd Azwan Bin Mohamed Raseli)

ACKNOWLEDEMENT

First and foremost, praise Allah the Almighty, who has helped and gave me courage and strength in completing the Final Year Project (FYP). I would not be able to make it through if it was not for His blessings.

Besides, the credit also must be given to my supervisor of FYP, Dr. Noohul Basheer Zain Ali because assisting me all the way to this point. Without his kindness and knowledge in supervising me, I would not get what I have now in final report of FYP.

I also want to express my grateful to my entire lovable family members. To my parent, Mohamed Raseli bin Jusoh and Che Khaidah Muhammad, who have always stood by my side and always remembered me in their prayers. They also have provided me so much support in form of material and moral support.

I would also like to take this opportunity to express my deepest gratitude to all my friends who have shared their knowledge and gave comments through the completion of this project. They have colored the most memorable moment in my life, as it would always be in my heart forever.

Last but not least, many thanks to the others whose name was not mentioned in this page but has in one way or another contributed to the accomplishment of this project. Thank you very much.

ABSTRACT

As technology advances, robots will become more prevalent in society. In order for the robots to be effective, considerable research in the mechanisms of the robot needs to be done to discover how to integrate robots into our society and to ease the human life. This paper describes a robot, the I-Roomba vacuum cleaning robot which has been around us since 2002. The robot was invented by i-Robot Company on 2002 and since then, the development of the robot did not stop where after two years, on 2004 i-Robot has stood with two model of Roomba. The different with the two models are cost and energy consumption. The objective of this study is to develop new liken product with Roomba that been produced by i-Robot. To achieve the objective, there many steps involve such as, the hardware part, whereby the movement of the robot, the vacuum ability and the sensors that control the robot need to take into consideration. Meanwhile, the important part of it was to program the robot in such way that it can cover the whole area, with or without obstacle. The needs to find the correct algorithm to cover whole surface is very important. There are a few algorithms that have been applied to the existing Roomba, and the most successful one is by the iRobot itself. Since the software part of the robot is very important, the research has been focused on developing the software, while at the same time the prototype has been developed accordingly.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Background of Study

Domestic robots are robots which are used in home. This group of robots is intended to do tasks such as housekeeping also to serve as educational and entertainment robots. With reference to human kind history, people have always been looking for simplifications and improvements that would make their life easier. Housekeeping devices especially domestic robots produce wide range of possibilities to achieve this goal. The issue is relatively up to date matter, because robotics has become so popular and the cost to build robot become more affordable than before. The status that housekeeping robots can obtain in the future, can be compared to the status of all others domestic devices (e.g. oven, fridge, hairdryer) which are nowadays present at our homes, and create basics of each home's equipment. That can be achieved only if their performance, utility and prices meet the satisfaction of the market. As a matter of fact, the project is focused on this area of robotics namely, I-Roomba, the vacuum cleaning robot. One of the project's aims was to design and implement an algorithm of movement of autonomous (or semiautonomous) vacuum cleaner that would be able to work in a place, with or without obstacles. According to this, objectives of the project have been focused on the surface covering and the time consumption.

The first vacuum prototype sold on market was called trilobite. It was manufactured by the Swedish corporation Electrolux and was available for purchase in 2001. One year later, in 2002 the first generation of the most widely famous, the *Roomba* [1] robotic floor cleaners have been on sale. Nevertheless, till now second and third generations of that robot have been produced. Each generation has obtained new improvement. Begin with better brushes and larger dustbins finish to the new algorithms of cleaning.

1.2 Problem Statement

Housekeeping, one of the main things the people do when they are at home. Conventional method that they use to clean their home is using the manual vacuum cleaner that needs the person who uses the vacuum to go around seek for the dirty place to be cleaned. But, people always seek and looking for simplification to make their life easier. So, the idea to build a prototype for an autonomous vacuum cleaner that has same function with the current I-Roomba comes out. In addition, robotic have become so popular nowadays where everyone want the experience to build their own robot and this kind of activities quiet affordable too.

1.3 Objective

The main objective for this study is to make a research and build an I-Roomba liken prototype, the vacuum cleaning robot that produce by i-Robot several years past. Many things need to put into considerations while constructing the robot, especially the algorithm for the robot to cover all places when it is placed in a closed room. The vacuum part also need to be sort out, where the robot only use the brushes and it will rotate all the time as it discover the whole surface in a places like a room. Finally, the prototype of the I-Roomba is expected to be produced by developing the surface covers algorithm in the beginning, and continue with the other parts that related to the robot like the dirt sensors, bump, drop off sensor and the auto-charging mechanism.

The main objectives are as follows:

i) Selection of the project's suitable test robot to test the algorithm
ii) Evaluation of several algorithms that would implement different approaches
iii) Comparison between selected algorithms, considering their efficiency in surface covering and its time consumption.
iv) Final outcome is the other version of I-Roomba liken prototype.

1.4 Scope of Study

The development of the I-Roomba requires knowledge and research regarding microcontroller or similar area of study. The multifunction of Programmable Interface Controller (PIC) needs to be utilized as it can be used as the brain of the robot, the place where the program will be executed in the robot. As we know, it is automatic vacuum cleaning robot, so the behavior of the robot does not need us to control it. The MPLAB software will be used to test and demonstrate the algorithm that has been decided. The study also will be focusing on C programming language because we will interpret the algorithm to the C programming language so that the PIC can be programmed accordingly.

CHAPTER 2
LITERATURE REVIEW

This section will further discuss about the technology of the robot. Besides, this section also highlights some theory about the robot from the previous research done by others.

2.1 I-Roomba by I-Robot

I-Roomba is the intelligent vacuum cleaner robot that uses many concept of engineering. The current version of the robot uses many sensors such as infrared sensor, cliff sensor and self-charger contact in order to make the robot runs well [2]. Roomba's shape is round to maximize the robot's ability to escape from obstacles; a round robot can always turn in place. The power system uses standard rechargeable batteries. Because of Roomba's energy-efficient cleaning mechanism, battery capacity is sufficient for 1–2 hours of running time [3]. A behavior-based programming scheme directs Roomba. The program contains many highly tuned strategies for avoiding and escaping tight spots, stuck situations that don't trigger the bump sensor, and other hazards. Roomba's cleaning strategy, combining random bounce and wall following, is designed to maximally cover the floor, even in the presence of clutter [3]. There are three sections of Roomba that allow the robot to do the job of cleaning the floor well, the Drive system, the Hoover section and the Brains or sensor system.

2.1.1 The Drive System

The Drive system: it comprises of a 3 wheel chassis with two main wheels to drive and a third directional wheel at the front. All the three wheels are rubberized and generate good traction. The wheels are on a movable chassis, which allows them to compensate for change in floor heights for various obstacles. This makes the unit very versatile for going over door jams and small height differences in the floor. The drive motors drive the Roomba at a good speed similar to manual vacuuming that done by human. With the unit being driven by 2 wheels it allows the unit to turn on the spot. This allows the unit both to get into and out of small spaces easily [4].

2.1.2 The Hoover System

The Hoover system: A slightly different vacuum system to normal vacuums. The first element is a rubberized paddle that collects the heavy material on the surface. Stones and surface dirt is scrapped up and into the collection bin. The second element is a deep brush that brushes deep into the fiber of your carpet. It will collect dust and hairs and ground in dirt. The third element is a rotating brush. This brush pushes any dirt at the edge of the wall or around or under chair legs and other furniture. This is an extra benefit as it does something most traditional cleaners don't which is really clean the edges of the home. All of this debris is deposited into a collection bin at the back of the unit, either through a filter system to remove dust or in the pet models a straight bin. Due to the size of the unit the vacuum holds about 3 vacuums before it needs emptying [4].

2.1.3 The Brain

The Brain of the Roomba: The design team could have gone with two methods of brains, one that efficiently remembers the room and then cleans it with one pass. This provides the issue that when you move the sofa to the ottoman it gets confused. I-Robot team went with a more simple system. The I-Robot Roomba finds the edges of the room and crosses at angles. It also does a circular clean if it detects dirt. This thought process is based on the various sensors it has. This thing is loaded with sensors from, distance sensors it will slow down when it gets near the wall, allowing for fast speed over the open carpet. Bump sensors for the edge allowing the unit to get up close to the edge of the wall. Height sensors, in the house with many edges must like this feature that it can detect the top of the stairs. It also has a homing beacon that allows it to detect the home base for charging but also I-Robot Roomba barrier towers or room divides [4].

From all the features, the most critical part should be the navigation of the robot itself. It is because the robot able to cover the whole place in a room, even though there is some obstacle in that room. So, to create the right and accurate algorithm and applies to the C programming language is the only way to make the robot behave like it does currently. This is because the brain of the Roomba is based on the microcontroller that it has to control it behavior.

2.2 The Robot Navigation - Mapping

2.2.1 Odometry for Motor Control and Navigation

The task of adding sensors to a robot's drivetrain that enable the robot's controller to measure the movement of the wheels is known as odometry. The data from the movement of the wheels can be used to quite accurately calculate the position of the robot from a known starting point and can be used to control the operation of the motors and to keep their speed and response to changing control values within a set range of parameters. This section introduces to the topic of odometry and how it can be used to navigate a robot. This hardware can be also to control the operation the robot's motors, keeping them within a set acceleration and speed profile [9].

In order to build an odometry system, you must have a way of measuring the angular velocity of the motor. This can be accomplished in a number of ways [10]:

i. Mono Phase Encoders: A sensor is placed on the motor shaft or wheel such that when the shaft rotates, the circuitry generates alternating 1's and 0's. One way of implementing a simple encoder is by attaching a disc with holes to the shaft, and using a break beam sensor to detect when a hole passes by. A mono phase encoder cannot determine the direction of motion. A serious failure mode occurs when the motor is essentially stationary with a hole half-way in front of the sensor.

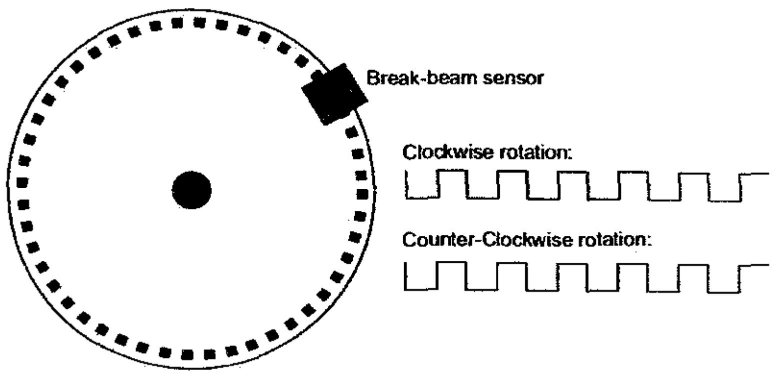


Figure 1(a): Simple Encoder. As a perforated disk rotates, a break beam sensor alternates between on and off.

ii. Quadrature Phase Encoders. An improvement upon mono phase encoders, quadrature phase encoders use two simple encoders arranged so that they produce waveforms 90 degrees out-of-phase. When the shaft rotates in one direction, signal A will "lead" signal B; when rotating in the opposite direction, signal "B" will "lead" signal A. Quadrature phase encoders are generally considered the most precise form of motor feedback, and can be quite expensive. A simple state machine can decode the quadrature phase signals into "up" and "down" pulses. The two out-of-phase signals provide immunity from spurious transitions due to a hole being half-way in front of one of the sensors: only one signal can be transitioning at any given time, and if this signal jumps back and forth, it maps to alternating (and cancelling) forward and backward motion.

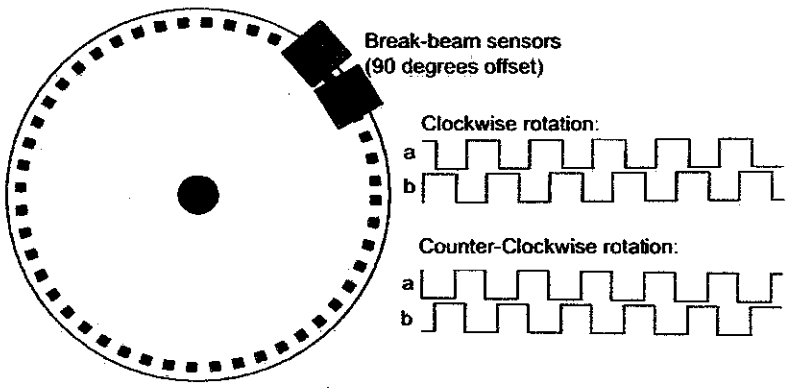


Figure 1(b): Quadrature Phase Encoder. Two simple encoders, arranged so that they pick up 90 degrees out of phase with each other so that can determine the direction.

2.3 Theoretical Result

Two more interesting and tested algorithms, which were dismissed, are presented in a table, together with reasoning of their failure. Nevertheless, more advanced tests were provided for further described algorithms which are namely, random motion and spiral motion algorithms. They have been tested if they are operating properly, what means that they fulfilled aim and all assumed objectives [5].

The efficiency was seen as area coverage in time. Taking into account sensors and motors constraints it had been very important to keep illumination levels and initial conditions as similar as possible for each type of test. That assured the most applicable results from each simulation. Owing the fact that both algorithms involved random movement, their outputs were highly unpredictable. Nevertheless, based on numerous tests, following efficiencies for operation time equals 1 minute have been achieved (table 1).

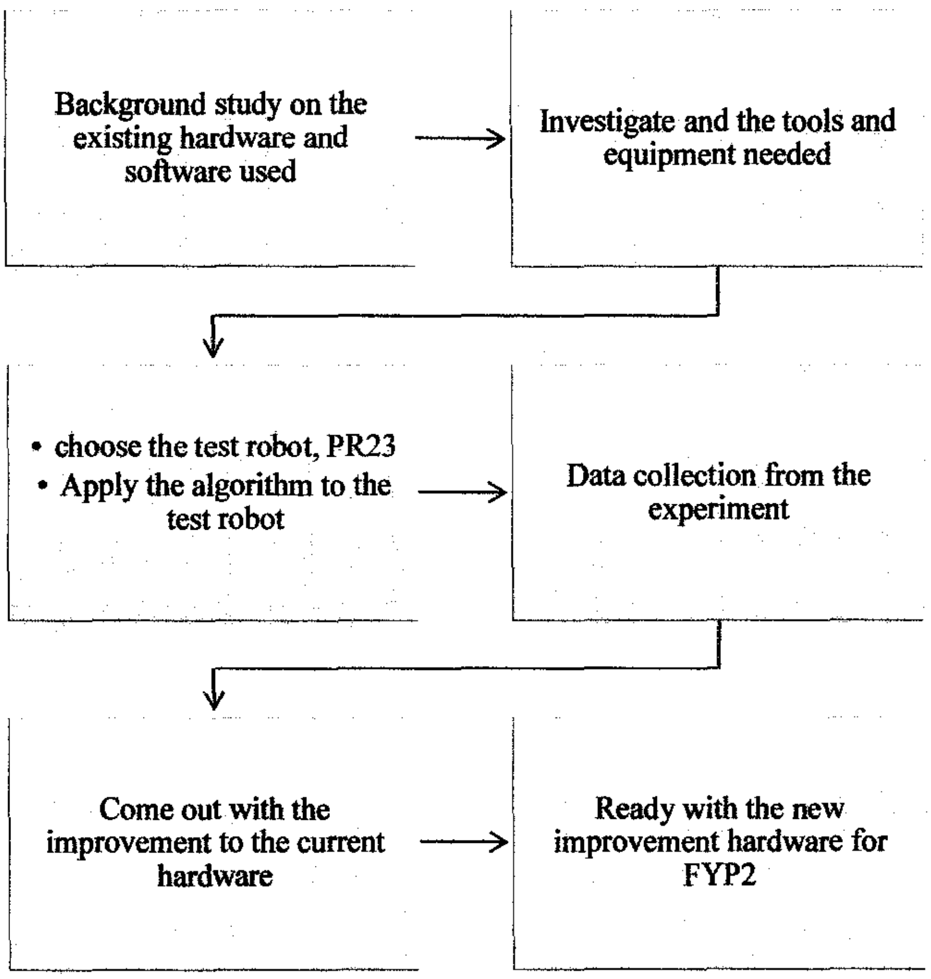
Table 1: Comparison of the efficiency of the two presented surface covering algorithms

Workspace dimensions: 400x500 mm; no obstacles	
Random Motion	Spiral Motion
1114 cm^2	1430 cm^2
Workspace dimensions: 800x500 mm; no obstacles	
Random Motion	Spiral Motion
1850 cm^2	2034 cm^2
Workspace dimensions: 800x500x100 mm; one obstacle: 100 cm^2	
Random Motion	Spiral Motion
1532 cm^2	1296 cm^2
Workspace dimensions: 800x500x100 mm; five obstacles sum: 150 cm^2	
Random Motion	Spiral Motion
870 cm^2	540 cm^2

CHAPTER 3
METHODOLOGY

3.1 Procedure Identification

This project has been divided into two main phase, that is FYP 1 and FYP 2. Below the chart shown the project flow for FYP1:



Background study on the existing hardware and software used
Investigate and the tools and equipment needed
• choose the test robot, PR23
• Apply the algorithm to the test robot
Data collection from the experiment
Come out with the improvement to the current hardware
Ready with the new improvement hardware for FYP2

Figure 2: Plan for FYP 1 project development

This is the FYP2 project flow:

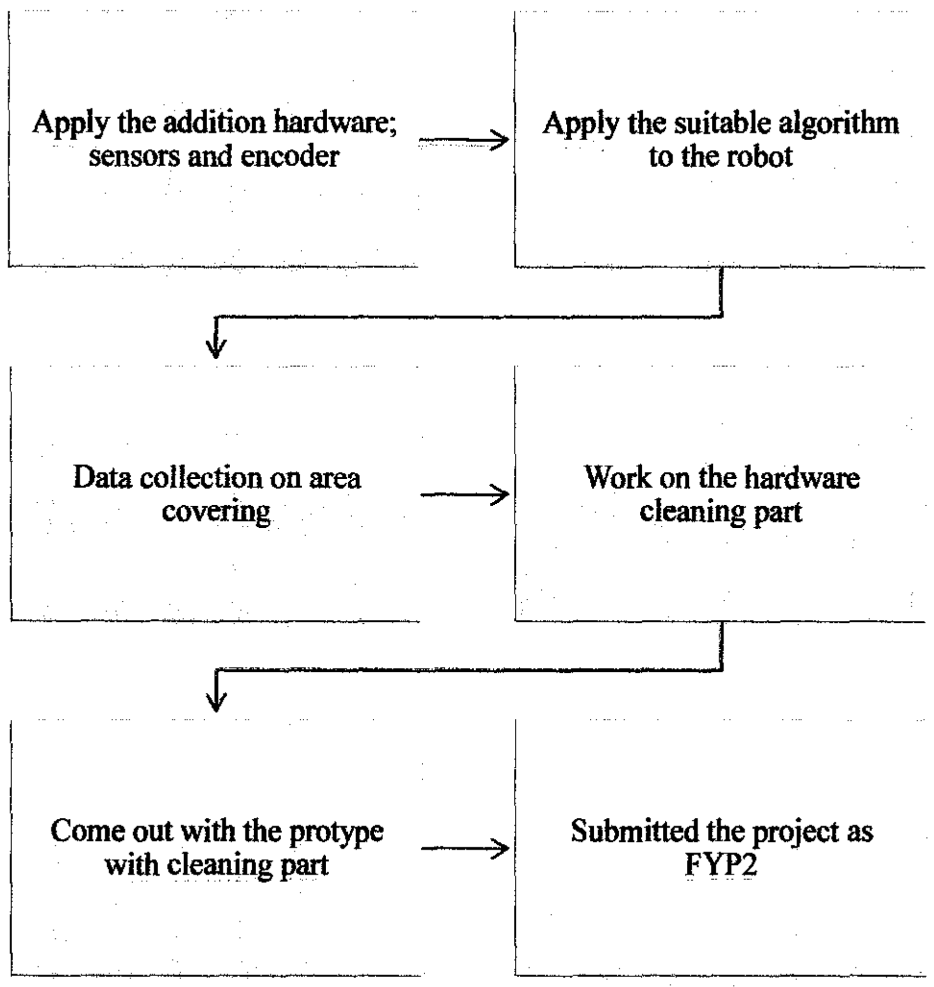

Apply the addition hardware; sensors and encoder
Apply the suitable algorithm to the robot
Data collection on area covering
Work on the hardware cleaning part
Come out with the protype with cleaning part
Submitted the project as FYP2

Figure 3: Plan for FYP 2 project development

Figure 2 and 3 describe the flow for each semester of FYP, starting from the background study until the final prototype has been submitted.

3.2 Tools and Equipment Used

3.2.1 Hardware

The development of this I-Roomba robot will begin with the research of the suitable algorithm to be applied to the robot so that it can cover whole surface area in a room. Initially, I will be use the PR23 robot by Cytron to apply and test the algorithm. The testing should consist of several possible algorithms and from that I will choose the best to be applied to the prototype robot. After the suitable algorithm been applied, the further step of the robot development will continue, the adding sensor, and the charging method need to be work out. The working prototype will be expected to be built when all the programming part is done. This is because I have to make sure the programming is running accordingly before started building the prototype otherwise the prototype will not works as desired.

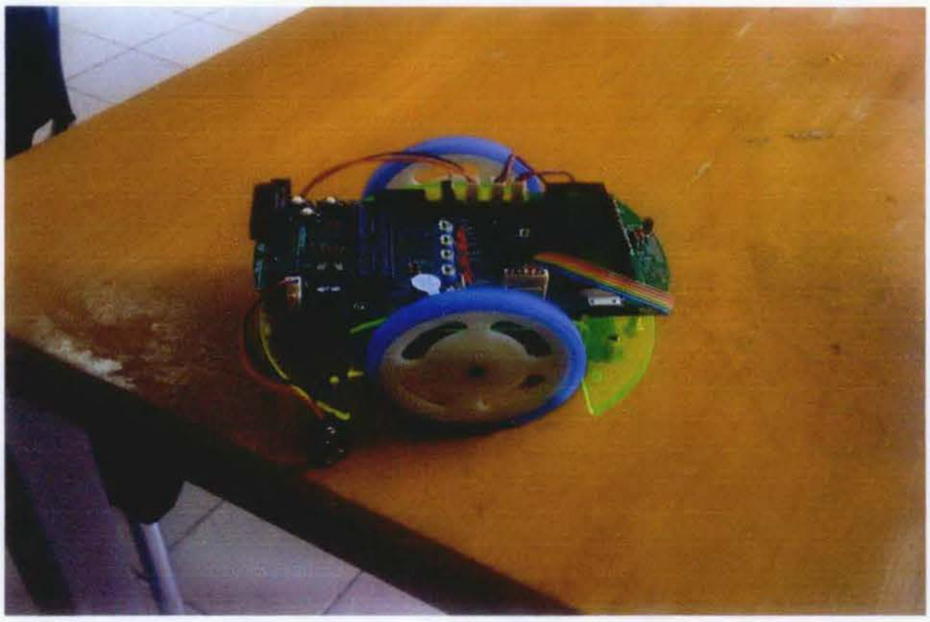


Figure 4: PR23 robot

3.3 The Limitation of Current Hardware

In order to achieve the objective to cover as much area in certain time, the hardware that is used need to be suitable to the certain algorithm that will be used. So, looking at the current hardware, there are some limitations that limit the robot to cover the whole area as desired. The main limitation to the current hardware is the sensors part. The sensor part seems to be so critical because that is the only input to the robot to control the movement of the robot. Looking at the current hardware, there is only a sensor that attached at the top front of the robot. With the single sensor, it can detect the front obstacles only and make a decision to turn to a side whether left or right every time it meets the obstacles.

The other limitation in current hardware is the controller to navigate the robot in an area. As the robot do now is the random movement, ignoring the area or the mapping of the area that it needs to cover. One of the controllers that widely used in navigation of a robot is called odometry or encoders' technique.

The solution to this limitation problem is to add more sensors to the test robot so that it can detect the obstacles either in front, in left or in right side of it. The common sensors that is used for this purpose is the ultrasonic sensor. By having this additional sensor, the performance in sense of area coverage of the robot is expected to be positive.

Figure 5: One type of ultrasonic sensor

Another part should be further improved is for the navigation of the robot. In this case the part that needs to be added for this purpose is the encoders. The installation for this encoder is at the wheel of the robot. The encoder can be self-designed, or rather than that, can also be purchased at the shop.

Figure 6: One type of encoder

Instead of using ultrasonic sensor, analogue sensor also can be used as both sensors are used to measure distance. In this project, Roomba will use the analogue sensor since the robot only need simple distance detector to navigate the robot accordingly. It has simple 3 wire connection that is positive, negative and signal. These are some of the features for this sensor:

i) 4.5V to 5.5V operating voltage.
ii) Working distance from 10cm to 80cm.
iii) Output voltage change over distance (2.45V – 0.45V)
iv) External control circuit is unnecessary.

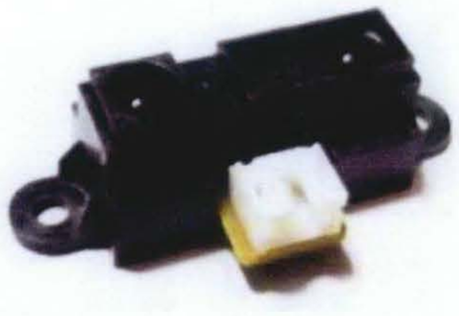


Figure 6(a): Analogue distance sensor

CHAPTER 4
RESULT AND DISCUSSION

4.1 The Algorithm

In this section details on two designed surface covering algorithms are presented end discussed. The first is the simple random-walk based one. The second one is more complicated hybrid concept that merges two different approaches – random movement and spiral motion.

4.1.1 Random Walk Based Algorithm

The first algorithm has been based on the idea of random movement (fig. 7). The robot is moving in forward direction till an obstacle is sensed and then it stops. Next, by comparing sensor readings decides in which direction to turn – left or right. Finally, by generating a random number decides how much to turn. The overriding aspect of this algorithm is a distance between the robot and obstacles. It has been done, by setting into its source code, a distance threshold value. Simply, when the distance is lower than the given threshold value, the robot takes an action in order to avoid obstacles. As a result of this, the manner in which robot is moving depends strongly on its surrounding. Despite the fact, that the algorithm assumes ongoing turns from the obstacle (i.e. if obstacle is sensed on the left, it turns right) what may cause stacking in corners. The algorithm fulfills the aim of the work, allowing the robot to move around a room avoiding obstacles.

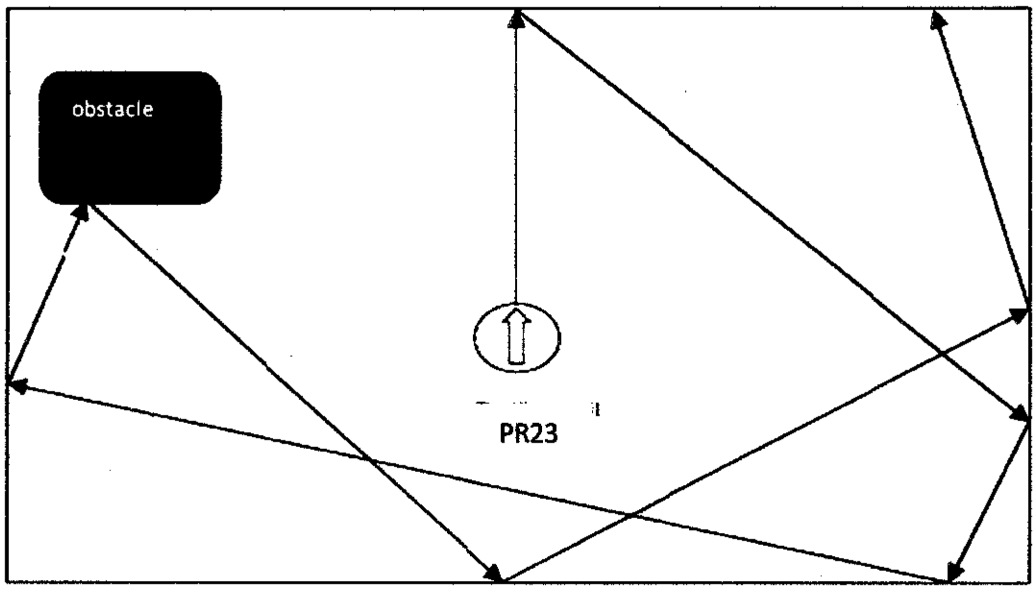


Figure 7: The 8 iterations of movement according to random algorithm.

This working mode is presented below in a form of a flow chart (fig. 8).

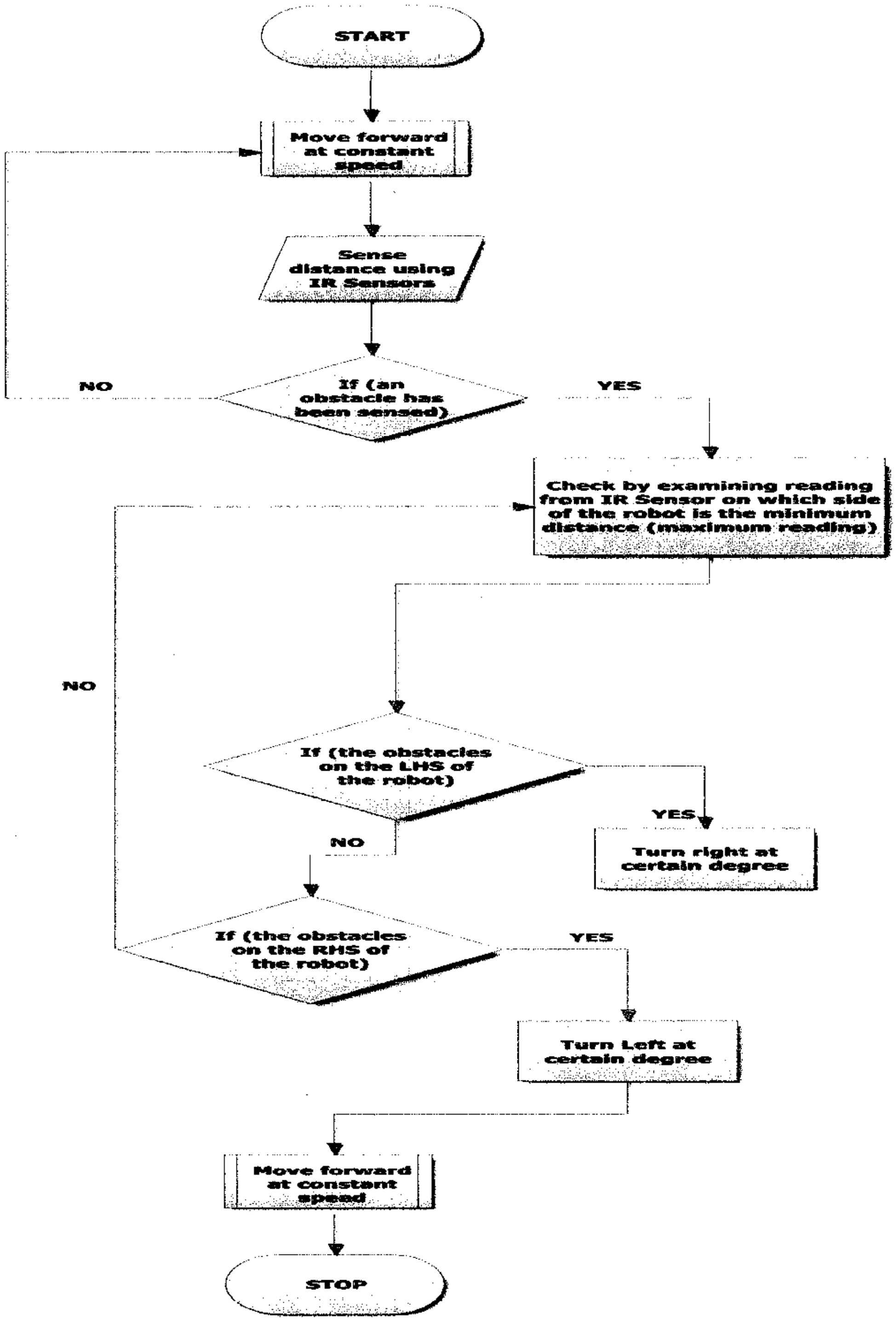

START
Move forward at constant speed
Sense distance using IR Sensors
If (an obstacle has been sensed)
NO
YES
Check by examining reading from IR Sensor on which side of the robot is the minimum distance (maximum reading)
NO
If (the obstacles on the LHS of the robot)
YES
Turn right at certain degree
NO
If (the obstacles on the RHS of the robot)
YES
Turn Left at certain degree
Move forward at constant speed
STOP

Figure 8: The flow chart for the random walk based algorithm.

4.1.2 Hybrid Algorithm

The second algorithm has been based on the idea of a spiral motion coupled with random turning (fig. 9). First of all, the robot checks if there is enough place to start moving spirally. If yes, the robot convolutes in a RHS direction, increasing a radius from center point until an obstacle is sensed. When the obstacle is sensed, robot stops and compares sensor readings to decide in which direction to turn, left or right. After the decision is made, the robot turns a given direction for a previously allotted angle. Finally, in order to allow itself to begin convoluting, it moves away from the obstacle for a given range. It has been done likewise to algorithm I, by setting into its source code, a distance threshold value. As a result of this, the manner in which robot is moving depends strongly on its surrounding characteristics.

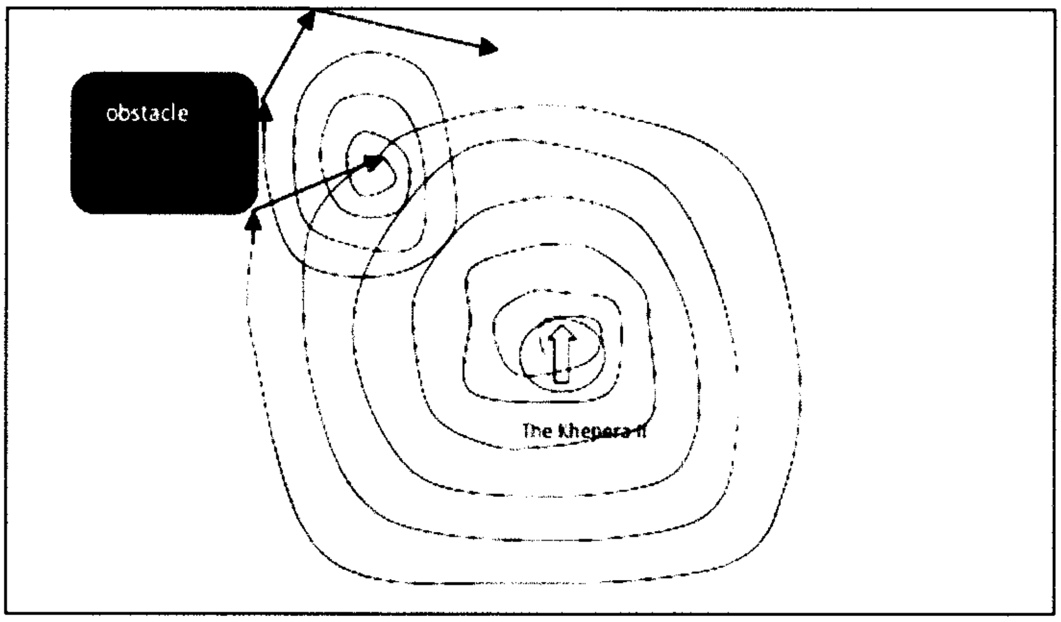
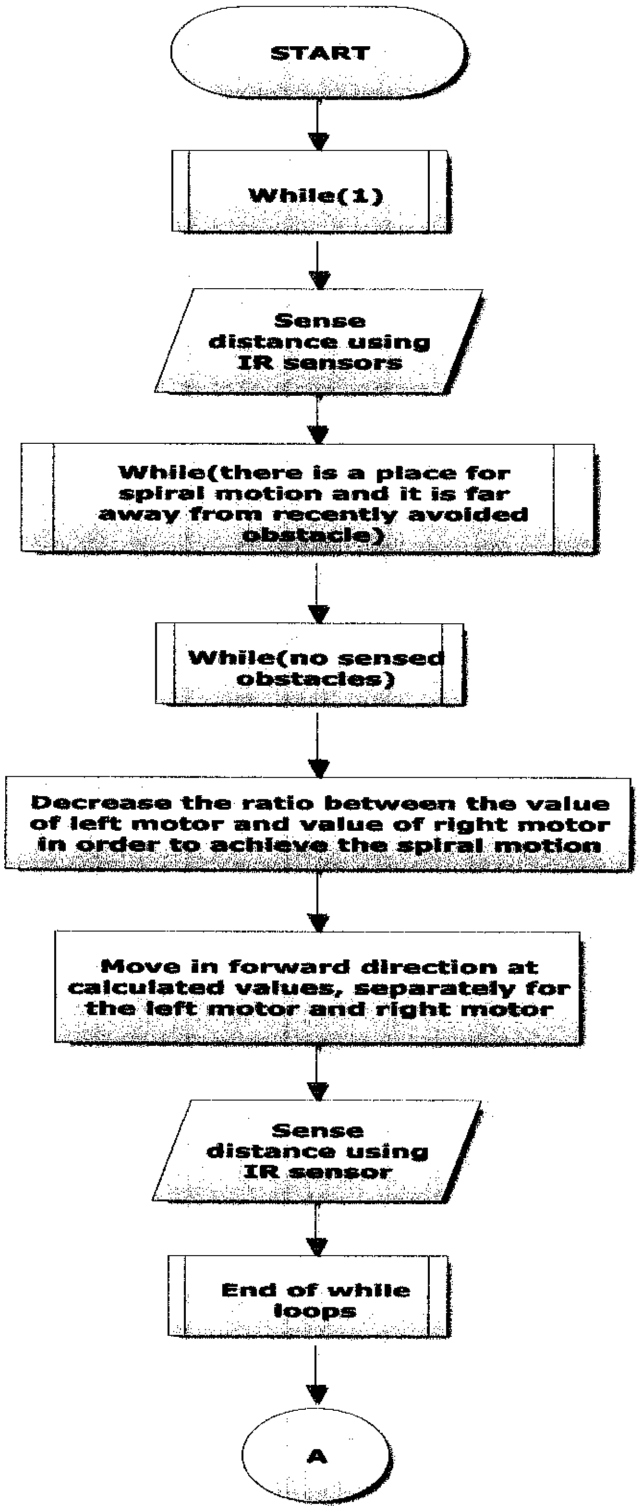


Figure 9: The 3 iterations of movement according to algorithm II

The principle how the algorithm works is shown in the form of a simplified flow chart (fig. 10) that gives a full and clear view on properties of the algorithm.


START
While(1)
Sense distance using IR sensors
While(there is a place for spiral motion and it is far away from recently avoided obstacle)
While(no sensed obstacles)
Decrease the ratio between the value of left motor and value of right motor in order to achieve the spiral motion
Move in forward direction at calculated values, separately for the left motor and right motor
Sense distance using IR sensor
End of while loops
A

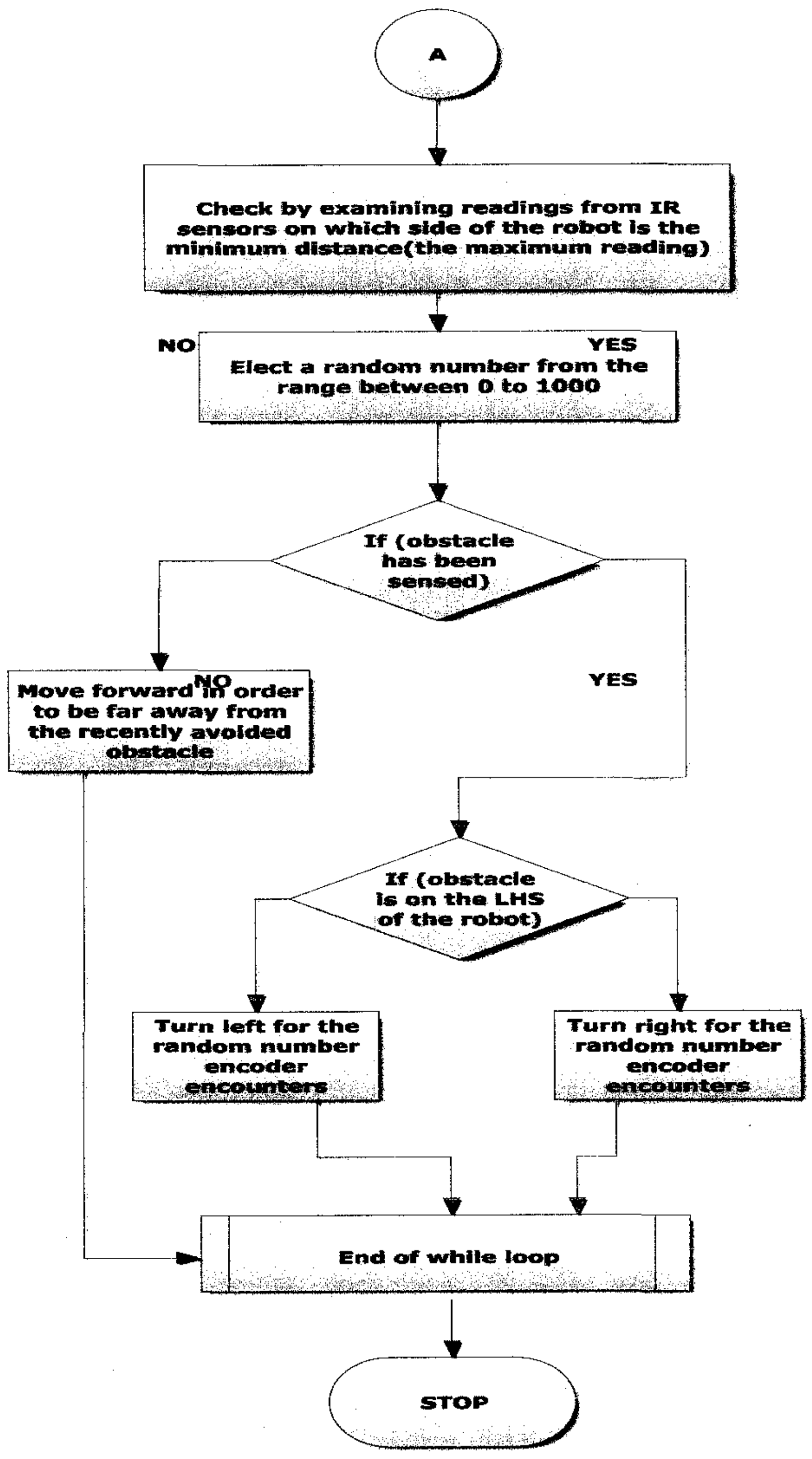

A
Check by examining readings from IR sensors on which side of the robot is the minimum distance(the maximum reading)
NO
YES
Elect a random number from the range between 0 to 1000
If (obstacle has been sensed)
NO
YES
Move forward in order to be far away from the recently avoided obstacle
If (obstacle is on the LHS of the robot)
Turn left for the random number encoder encounters
Turn right for the random number encoder encounters
End of while loop
STOP

Figure 10: The flow chart for the hybrid algorithm.

4.2 Experimental Result

4.2.1 Random Motion

The result shown is the tested algorithm with the current hardware of the PR23. The experiment is conducted in an area of 200cm x 200cm dimension with random motion.

Case 1: No Obstacles

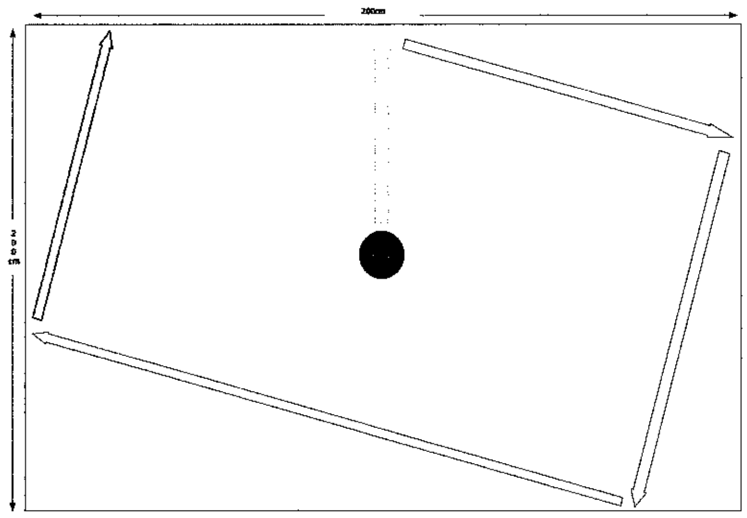


Figure 11a: The 4 iterations of random motion

Case 2: With an Obstacle

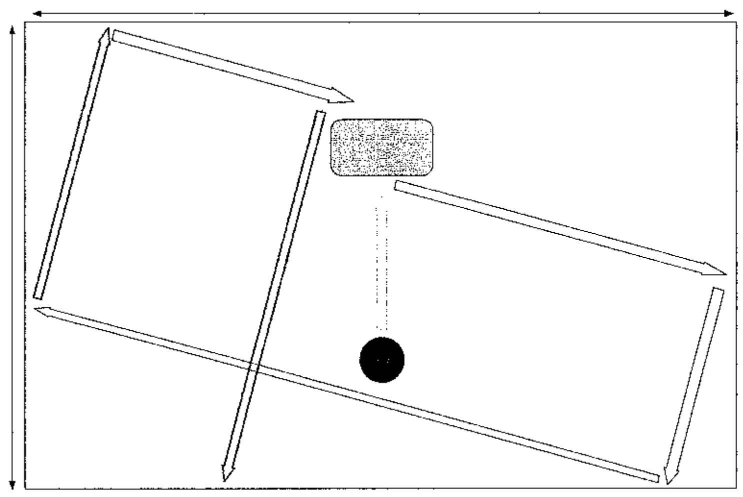


Figure 11b: The 7 iterations of random motion with an obstacle

4.3.2 Spiral Motion

The result shown is the tested algorithm with the current hardware of the PR23. The experiment is conducted in an area of 200cm x 200cm dimension with spiral motion.

Case 1: No Obstacles

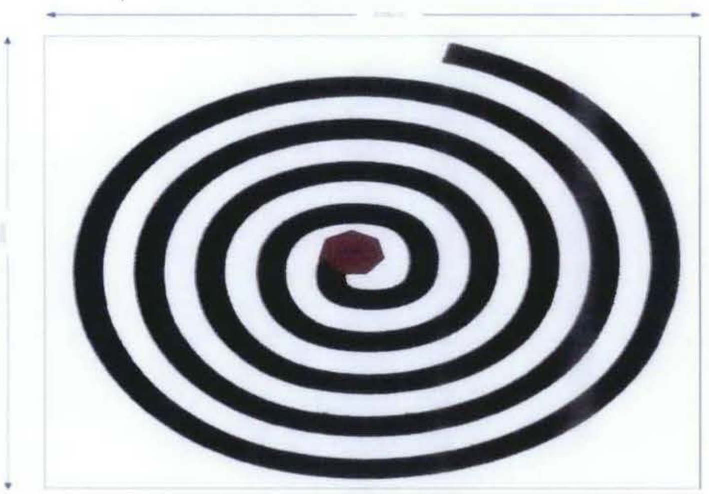


Figure 12a: The iteration of spiral motion

Case 2: With an Obstacle

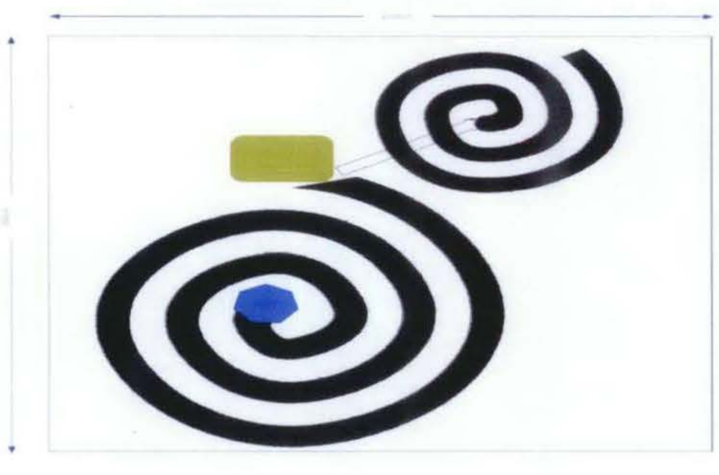


Figure 12b: The 2 iterations of spiral motion with an obstacle

4.3 Design Prototype

The design for this robot consist both mechanical and electrical design. Before take further step of doing the prototype, firstly the base must be strong and suitable enough to support the cleaning mechanism that later will be attached to the main base. The electrical part of this project is the board and its program. Many factors must be taking into consideration to build this prototype, for instance, the size of the base, power consumption and also able to move freely.

4.3.1 First Model

The initial model for this project is using the PR23 robot. This robot have round base, powered with two servo motor left and right. It also attached with an analogue sensor for navigation purpose. The sensor is located middle front of the robot.

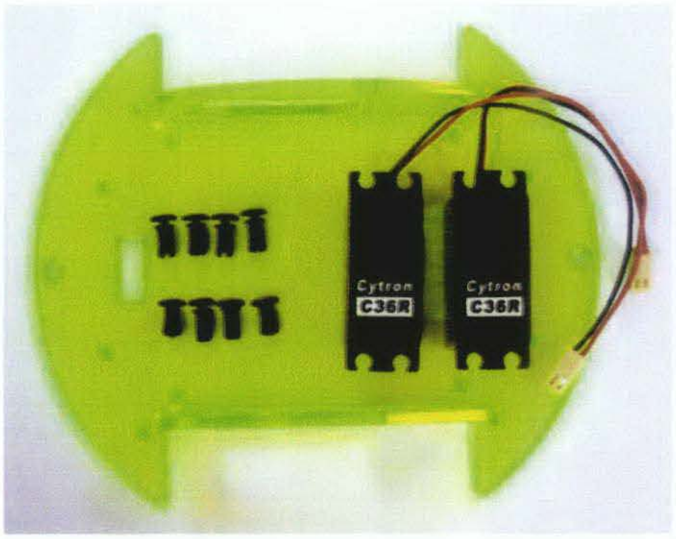


Figure 13: Robot base for first prototype

The base is 16cm in diameter, and has two wheels to drive the prototype around. The movement with two wheel quiet smooth, but the only problem is just the size of the base. Since the diameter only 16cm, it cannot occupy a space to attach additional hardware for cleaning purpose.

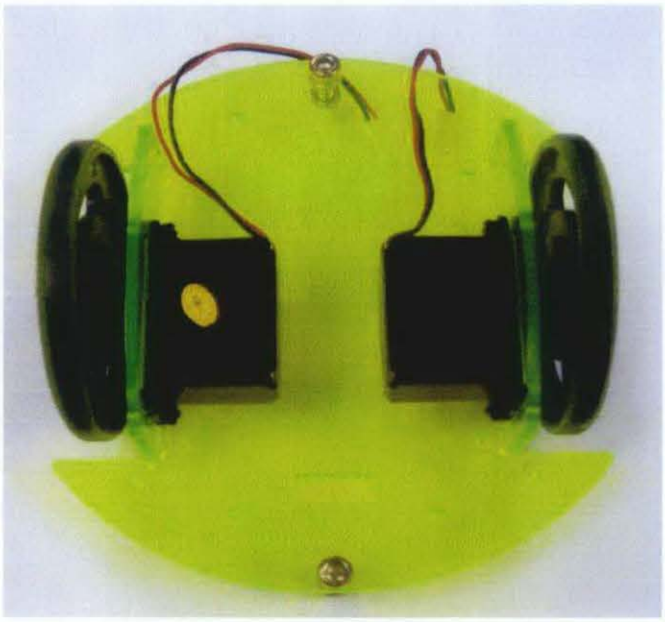


Figure 14: The base with the wheel

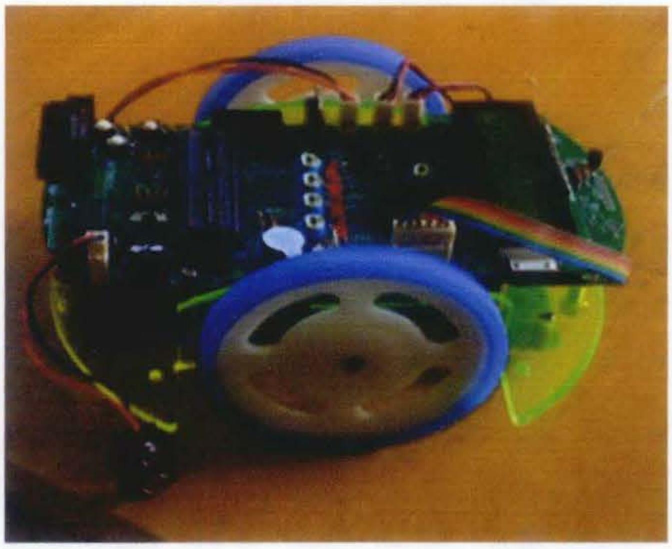


Figure 15: First prototype with a sensor

4.3.2 Second Model

Since the initial model has limitation, some modifications have been done to improve the model, with larger space and more sensors. The space need to be large enough to occupy cleaning system. The second model has more sensors and also has been uplifted to give more space between the base and main board.

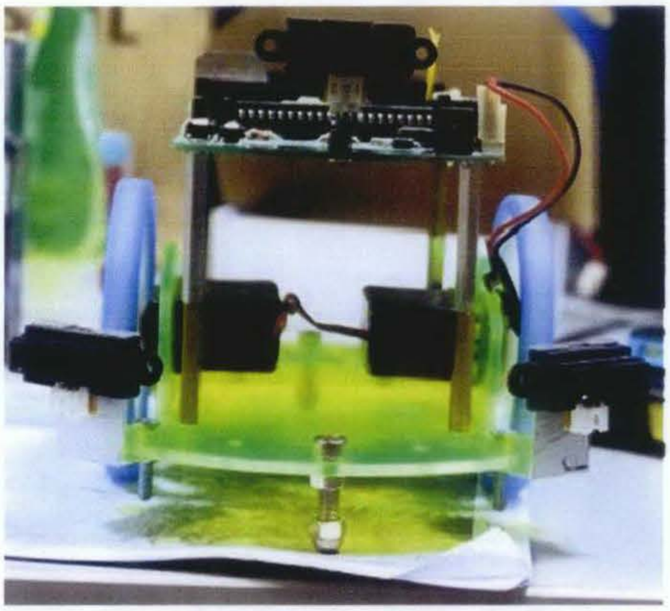


Figure 16: The second model with larger space and 3 sensors

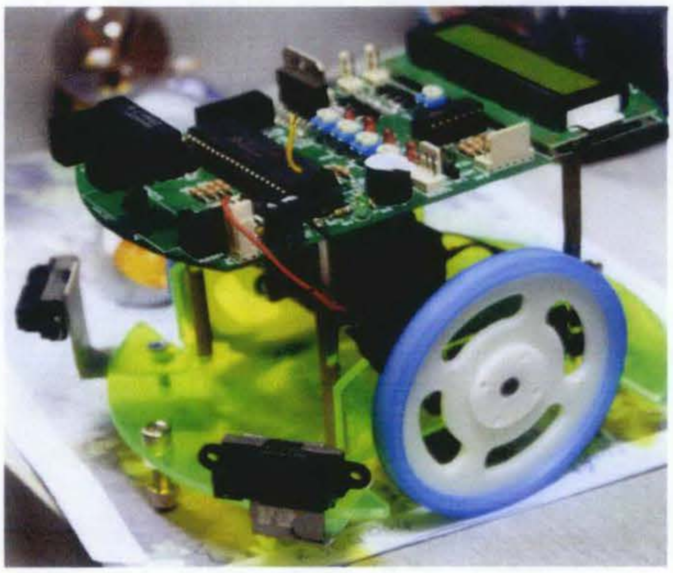


Figure 17: Side view of second model

4.3.3 The Latest Prototype

From the first and second model, the major problem for previous prototype is the size of the base. The size is too small to occupy the cleaning part of this robot. So, the robot base has been changed to a larger and suitable base. It is rectangular base with rubber chain wheel. The size also quiet big, with 8cm (W) x 16.5cm (L) x 5cm (H) in dimension and it is expandable to certain size that is suitable to mount the cleaning part.

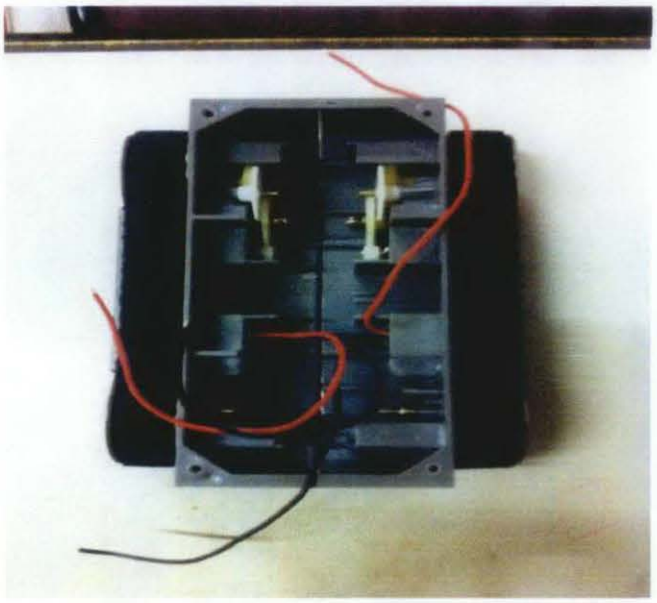


Figure 18: New base for the prototype

The new base is quiet big and seems sufficiently enough to mount additional hardware in the middle of the base.

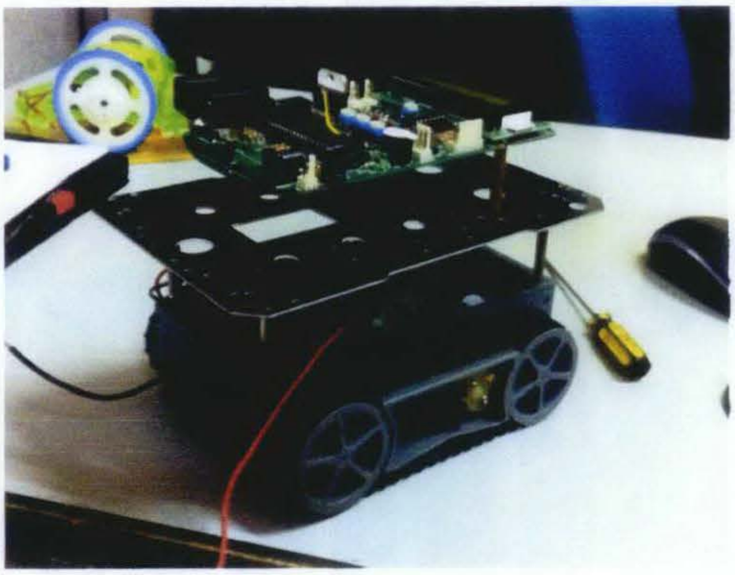


Figure 19: The looks with attached board

4.4 Main Base Modification

The new base of the latest prototype has sufficient enough space to suit the vacuum part. However, some modification needs to be done to attach the vacuum to the robot. The design for the base has been done so that the vacuum can be attached properly to the base. Here is the drawing for the base modification:

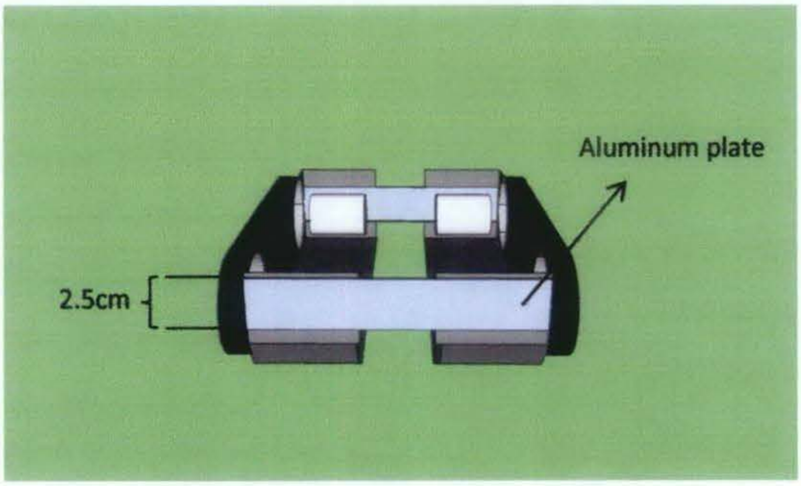


Figure 20: Front view of the base

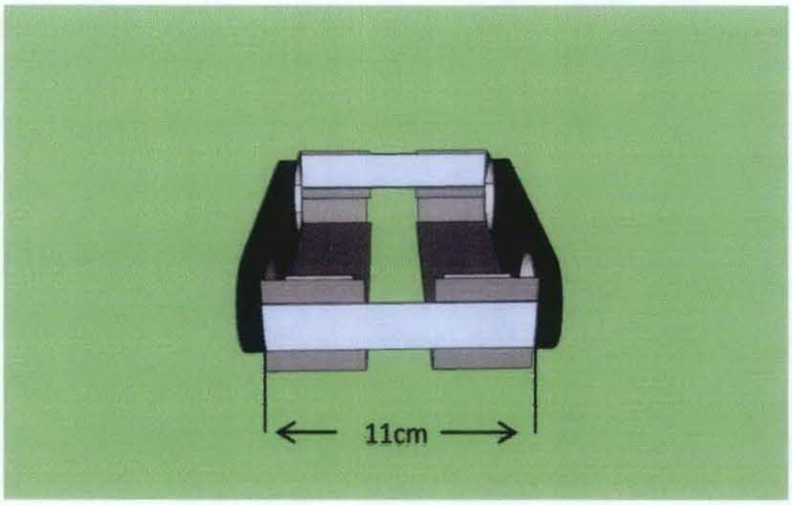


Figure 21: Back view of the base

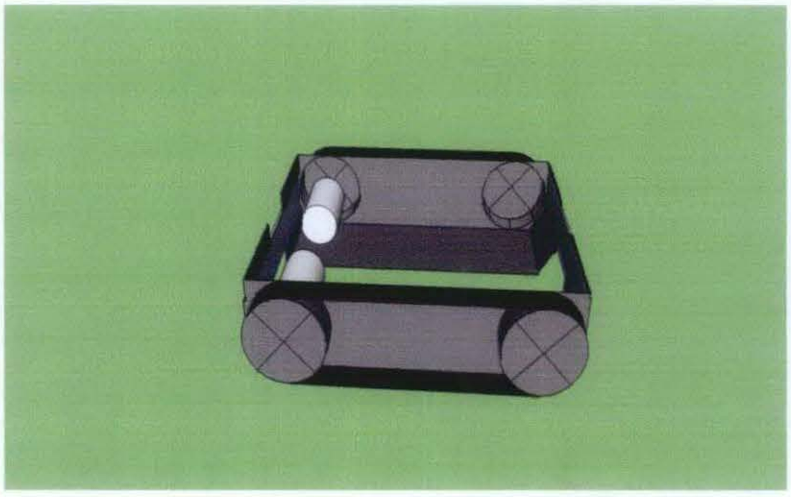


Figure 22: Side view of the base

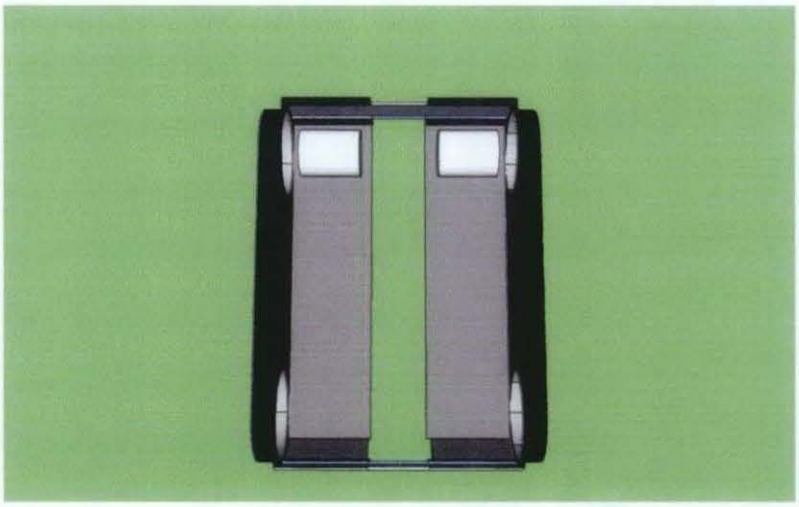


Figure 23: Top view of the base

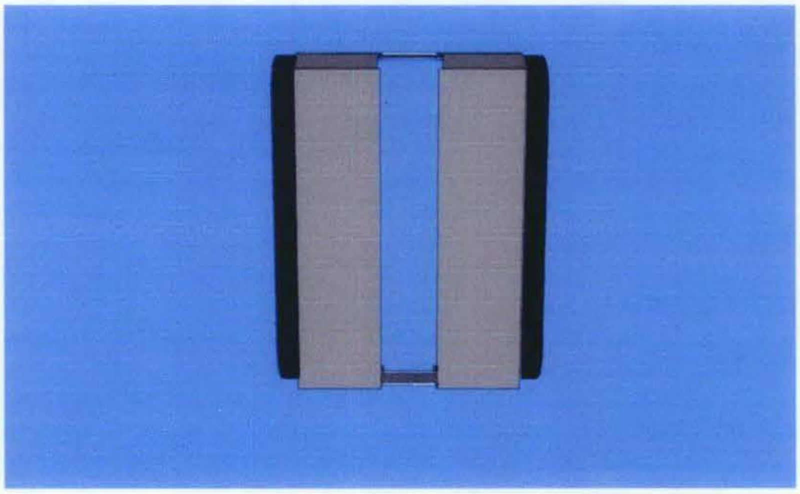


Figure 24: Bottom view of the design

Based on the drawing, the base has been separated and aluminum is used to create the gap in the middle of the base. The idea is to attach the vacuum part within the gap created. So, the figure on the next page shows the real picture after the modification been done.

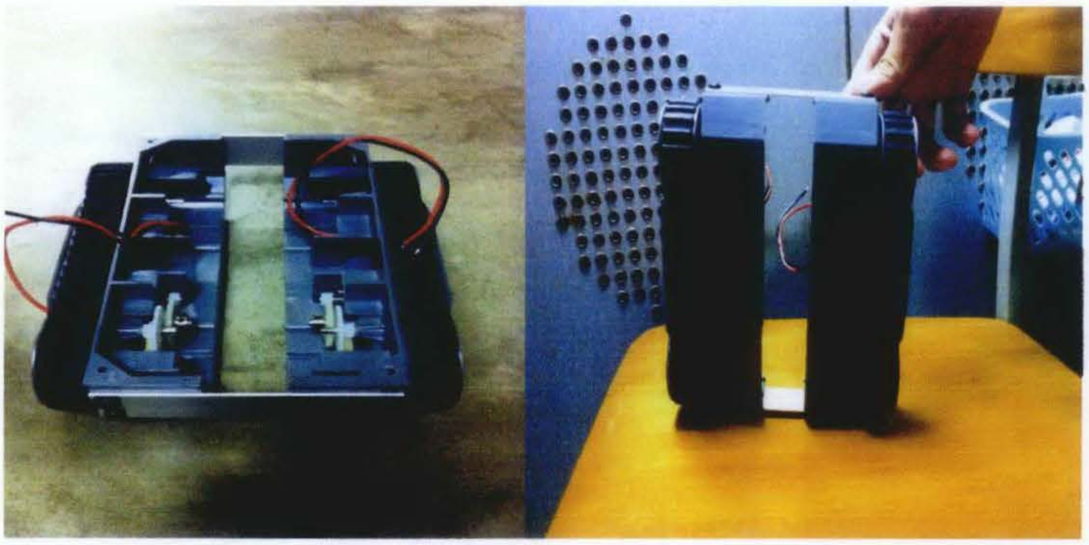


Figure 25: The base after the modification has been done

The modification has created 3cm gap in between to make the width become 11cm, while the length of the body is 18cm. Based on the new layout, the additional part which is the vacuum should be placed in the middle of the base.

4.5 Vacuum Design

The design for the vacuum further been done after the modification of the base is successful. Based on the basic principle of vacuum cleaner, four units of fan is to be used as the suction blower of the vacuum. Three of the fans are 30mm x 30mm x 10mm of dimension, and the other one is 40mm x 40mm x 20mm.

Figure 26: Fans for the vacuum

Then, the drawing of the vacuum design is prepared:

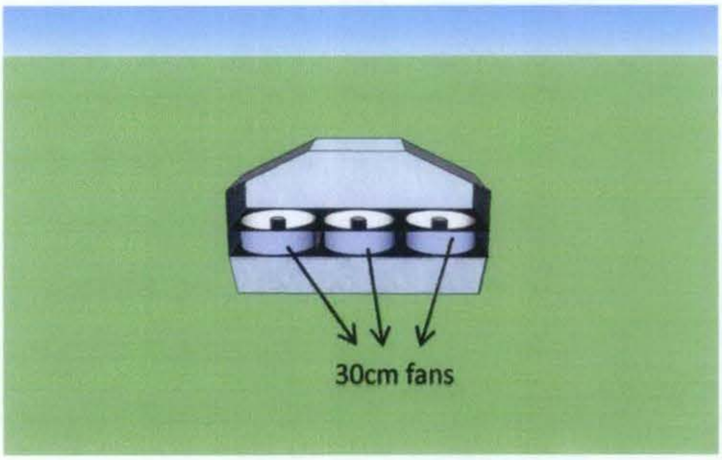


Figure 27: Front view of the vacuum

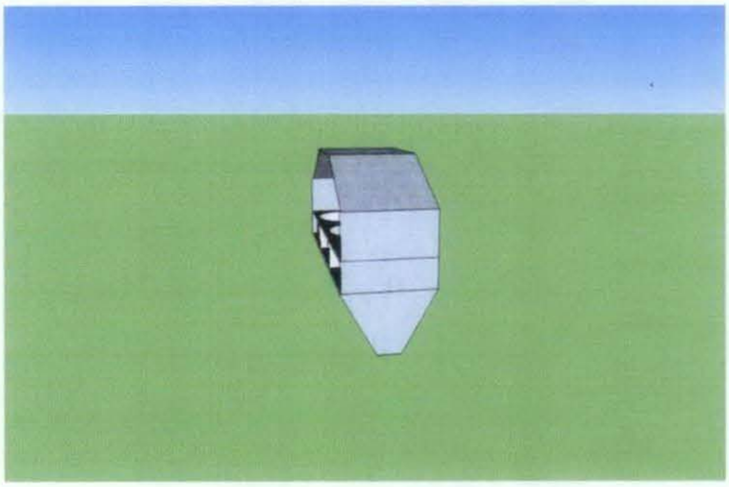


Figure 28: Side view of the vacuum

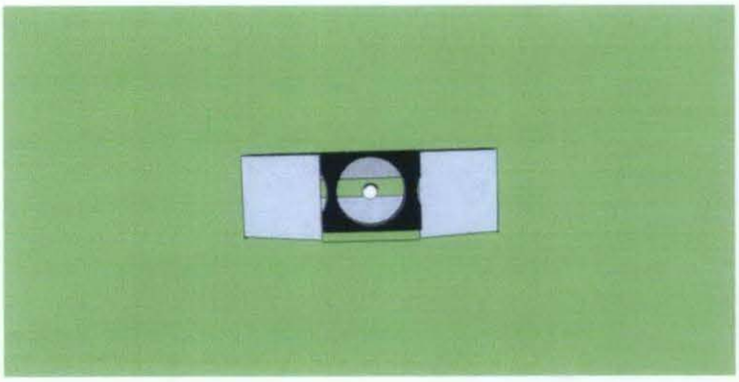


Figure 29: Top view of the vacuum

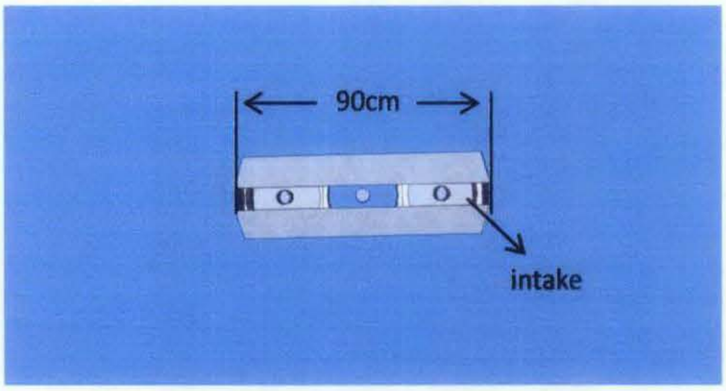


Figure 30: Bottom view of the vacuum

The main part of this vacuum is the fans. It is used as the blower to the vacuum. Based on the drawing, the fans are located in the middle of the vacuum. The intake of the vacuum is design to be smaller than the upper body to create lower pressure. When the fans on, it will create the much lower pressure at the upper part of the fans because of the high velocity air created from the fans. So, at the lower part, the pressure is higher, thus can create the suction at the intake of the vacuum. The vacuum is then attached to the other part as in the figure below:

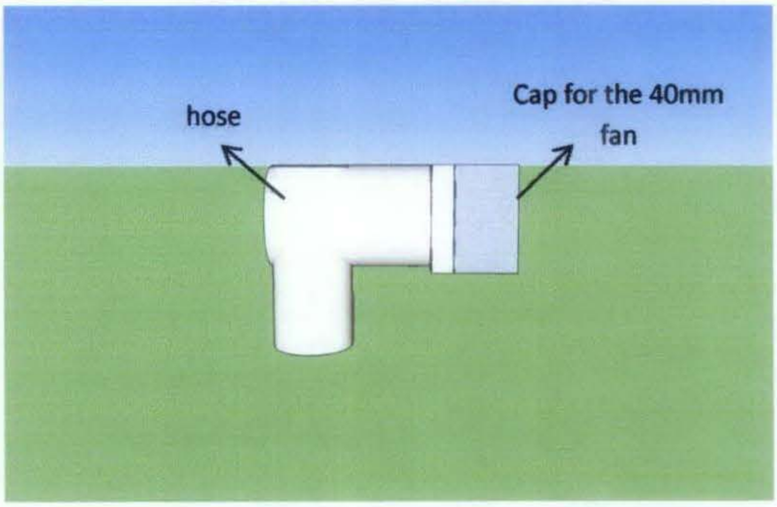


Figure 31: The upper part of the vacuum

The upper part consists of a hose and a fan. A 40mm fan is attached at the end of the hose. Then, there is cover for the fan, and the cover also acts as the dust and trash container of the vacuum. The function of the 40mm fan is to bring the thrash from the vacuum to the container.

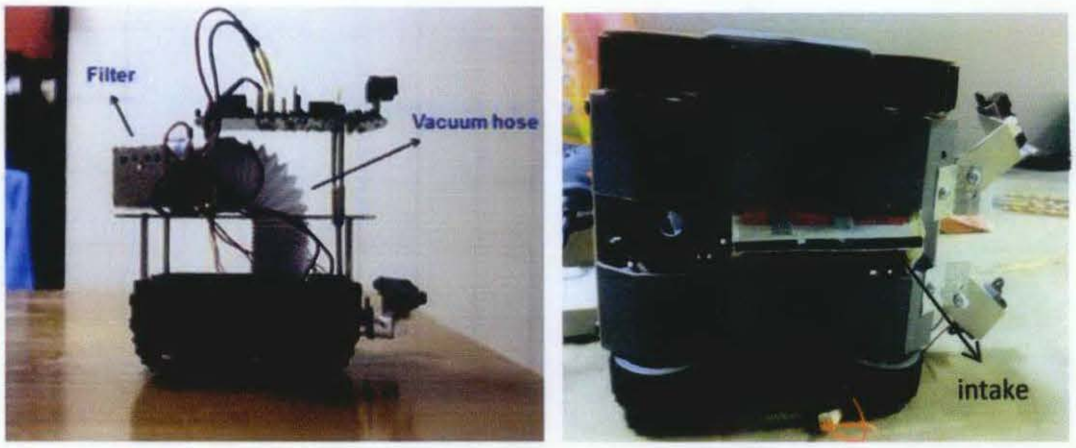


Figure 32: The vacuum after been attached to the base

4.6 Area Coverage

One of the objectives of this project is to get the maximum area coverage in the closed compound area with the applied algorithm. In order to achieve this objective, the test area has been identified to test how much can be covered by the robot in certain time, in this case in 5 minutes time. The test area is one close compound with 1m x 0.5m in dimension. The robot is place in the starting point and the place that it has go will be identified and mark until 5 minutes is finish. From that marking, the area coverage can be calculated and efficiency of the robot can be computed. The testing is also been done with an obstacles to see how much can the robot cover the surface area.

The graphical representation of the coverage area that has been covered by the robot is as below:

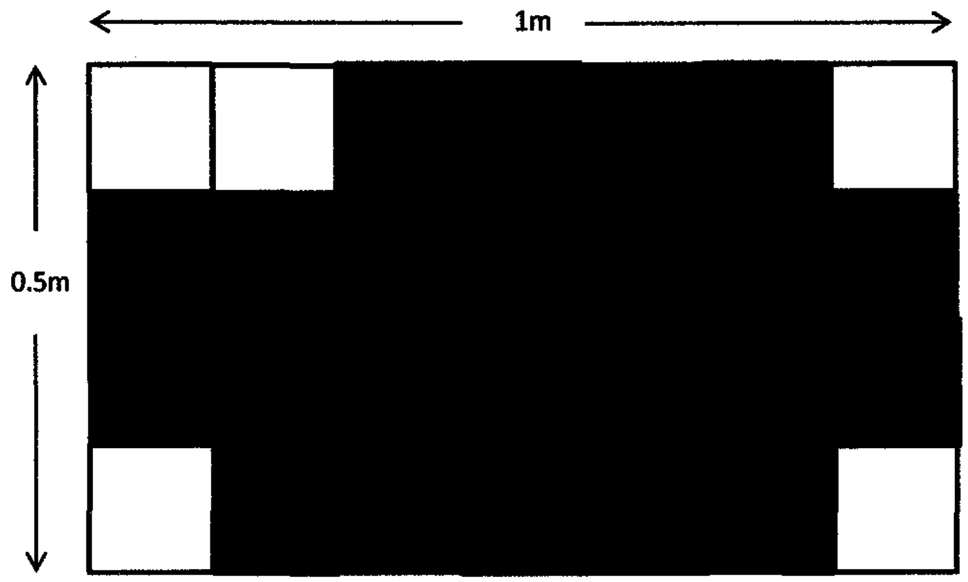


Figure 33: Area coverage without obstacle

The testing area is made using grid so that the movement to certain grid can easily been recorded for the analysis. Based on the Figure 33, the surface that can be cover by the robot in 5 minutes is:

0.017875 m^2 is the area for 1 small square, so the coverage area is:

0.017875 m^2 x 23 box = 0.411125 m^2

So, the efficiency of the robot is:

Efficiency = $\frac{0.411125}{0.5}$ x100 = 82.25%

Then, the testing is continued by adding an obstacle in the test area:

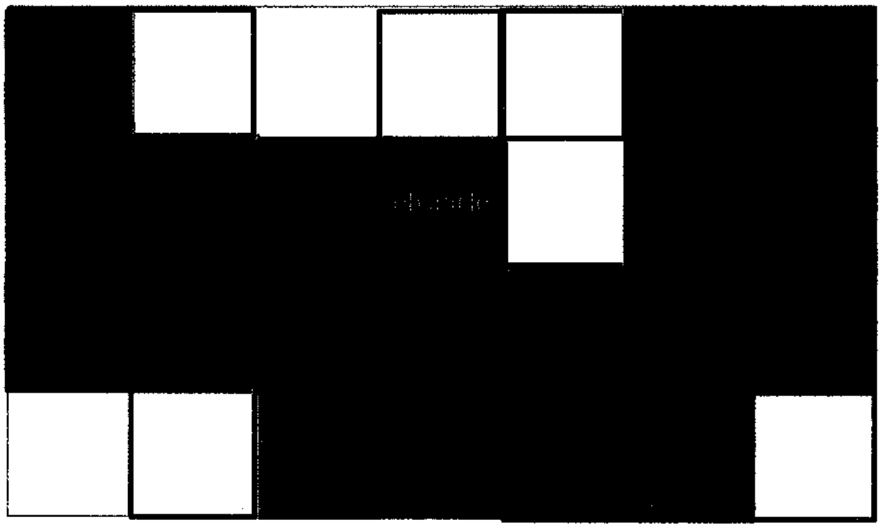


Figure 34: Area coverage with an obstacle

With the same testing area, an obstacle is placed in the middle of the compound. With the same method to determine the coverage area, the result is:

$0.017875 \text{ m}^2 \times 20 \text{ boxes} = 0.3575 \text{ m}^2$

So, the coverage efficiency is:

$\text{Efficiency} = \frac{0.3575}{0.5} \times 100 = 71.5\%$

From the result for both situation, the maximum coverage efficiency can be obtained is around 83% without an obstacle.

CHAPTER 5
CONCLUSIONS AND RECOMMENDATION

5.1 CONCLUSION

The aim of this work was to design and implement algorithms of movement of autonomous (or semiautonomous) vacuum cleaner that would be able to work in a place, with or without obstacles. The algorithm seems very important because the further part development of I-Roomba needs the movement of the surface covers first. So, the developed algorithm that is random motion algorithm has fulfilled this goal. What is more, the main objectives for the project have been achieved successfully based on analysis of developed algorithms, efficiencies and their limitations. Apart from that, the I-Roomba liken prototype also has been produced with basic concept and ability of the I-Robot's Roomba. The design of the prototype has been developed successfully with the vacuum part has been attached to the prototype that makes it has the functionality of an autonomous vacuum cleaner.

5.2 RECOMMENDATION

Although the prototype has been produced, with the basic ability and function of the I-Robot's Roomba, further research and improvement can be done to enhance the ability and efficiency of the prototype. In terms of power consumption, the research can be done to improve the power consumption and energy so that the prototype can runs continuously with minimum power required.

Other than that, the prototype can be further enhanced by adding the special features of such as auto recharge ability, virtual wall and scheduler function. To obtain that, the research in terms of programming, sensors and integrated circuit need to be done.

REFERENCES

[1] I-Robot official website, irobot.com [Online]. Available: http://www.irobot.com/uk/home_robots_roomba_tech.cfm [Accessed: August 20, 2010]

[2] Julia Layton, "Homepage – How Stuff Works", [Online]. Available: http://electronics.howstuffworks.com/gadgets/home/robotic-vacuum1.htm [Accessed: August 20, 2010]

[3] Joseph L.Jones, "Robot at Tipping Point, The Road to Irobot Roomba", IEEE Robotics & Automation Magazine, March 2006. [E-book] Available: IEEE Xplore

[4] "Irobot Roomba Information", November 9, 2009. [Online]. Available: http://www.irobotroomba-info.com/what-is-the-irobot-roomba [Accessed: August 18,2010]

[5] Krzysztof Skrzypczyk "Surface Covering Algorithm for Semiautonomous Vacuum cleaner", Proceedings of the 12th WSEAS International Conference on Automatic Control, Modeling & Simulation 2009

[6] John Iovine, *PIC Microcontroller Project Book,* McGraw-Hill, pp 1-9.

[7] Howie Choset, Kevin Lynch. Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia Kavraki, and Sebastian Thrun, *Principles of Robot Motion-Theory, Algorithms, and Implementations,* Prentice-Hall of India, 2005

[8] Joseph L. Jones, *Robot Programming-A Practical Guide to Behavior-Based Robotics*, McGraw-Hill, 2004

[9] Myke Predko, *Programming Robot Controllers,* McGraw-Hill, 2003

[10] Edwin Olson "A primer an Odometry and Motor Control", January 2009 Available: eolson@mit.edu

APPENDICES

APPENDIX A: GANNT CHART FYP 1

Activities / Week	1	2	3	4	5	6		8	9	10	11	12	13	14	15	16	17	18	19
Topic selection	▨																		
Research of Project Title		▨	▨	▨															
Submission of Preliminary Report				■															
Algorithm Study					▨	▨		▨	▨										
Applied the algorithm to C language						▨		▨	▨	▨									
Preparation for Progress Report						▨		▨											
Submission of Progress Report							Mid-semester break	■											
Work the algorithm with the PR23 test robot									▨	▨	▨	▨							
Submission of Draft Report												■							
Data collection (The movement, how much it covers the surface)											▨	▨	▨	▨	▨				
Improvement and additional features (sensors, vacuum part)												▨	▨	▨	▨	▨	▨		
Data recording (Surface covers)													▨	▨	▨	▨			
Submission of Interim Report															■				
Preparation for Oral presentation																▨	▨	▨	▨
Oral presentation																			■

Legend :

■ Milestone

■ Work Progress

▨ Research

APPENDIX B: GANNT CHART FYP 2

Activities / Week	1	2	3	4	5	6	7		9	10	11	12	13	14
Setup the additional sensors to the robot	■													
Programming and cofiguring the new hardware (both sensor)		■	■											
Adding a motor for sweeping and vacuuming purpose (makes its look like cleaning robot)			■	■										
Data Collection (make the testing to the robot, make sure been programmed accordingly)				■	■			Mid Sem Break						
Test and improvement of the cleaning mechanism						■	■							
Submission of progress report									■					
Pre EDX preparation										■				
EDX											■			
Submission of Technical Report													■	
Oral presentation													■	■

```
//==========================================================================================
===========
// [redacted]              [redacted]
//  Project                :PR23
//  Project description :Simple line follow
//  Version                :v1.4
//==========================================================================================
===========

//  include
//==========================================================================================
===========
#include <pic.h>

//  configuration
//==========================================================================================
===========
__CONFIG ( 0x3F32 );

//  define
//==========================================================================================
===========
#define sw1          RE0
#define sw2          RE1
#define motor_ra     RC0
#define motor_rb     RC3
#define motor_la     RC4
#define motor_lb     RC5

#define s_left       RB0
#define s_mleft      RB1
#define s_mright     RB2
#define s_right      RB3

#define buzzer       RE2

#define rs           RB7
#define e            RB6
#define lcd_data     PORTD
#define b_light      RB5
#define SPEEDL       CCPR1L
#define SPEEDR       CCPR2L

#define CHANNEL0     0b10000001   // AN0 ( Ultrasonic )
#define CHANNEL1     0b10001001   // AN1 ( Distance sensor )

//skps protocol
#define p_select         0
#define p_joyl           1
#define p_joyr           2
#define p_start          3
#define p_up             4
#define p_right          5
#define p_down           6
#define p_left           7
#define p_l2             8
#define p_r2             9
#define p_l1             10
#define p_r1             11
#define p_triangle       12
#define p_circle         13
#define p_cross          14
#define p_square         15
#define p_joy_lx         16
#define p_joy_ly         17
#define p_joy_rx         18
#define p_joy_ry         19
#define p_joy_lu         20
#define p_joy_ld         21
#define p_joy_ll         22
#define p_joy_lr         23
#define p_joy_ru         24
#define p_joy_rd         25
#define p_joy_rl         26
#define p_joy_rr         27

#define p_con_status     28
```

```
#define p_motor1        29
#define p_motor2        30

#define RX_PIN      RC7
#define TX_PIN      RA2
#define BOT_ADD     100



//  global variable
//==============================================================================================
============
unsigned char data[6] = {0};
const unsigned char line [] = {"1.LINE FOLLOW"};
const unsigned char US[] = {"2.Ultrasonic"};
const unsigned char AS[] = {"3.Analog Sensor"};
const unsigned char XB[] = {"4.SKXBEE"};
const unsigned char SKPS[] = {"5.SKPS"};
const unsigned char *mode [5] = {&line[0],&US[0],&AS[0],&XB[0],&SKPS[0]};

unsigned int result;
unsigned int To=0,T=0,TH=0;
unsigned char REC;
unsigned char i=0,raw;

unsigned int us_value (unsigned char mode);

//  function prototype
//==============================================================================================
============
void init(void);
void delay(unsigned long data);
void send_config(unsigned char data);
void send_char(unsigned char data);
void e_pulse(void);
void lcd_goto(unsigned char data);
void lcd_clr(void);
void send_string(const char *s);
void dis_num(unsigned long data);

void line_follow(void);
void ultrasonic(void);
void wireless_xbee(void);
void analog_sen(void);
void SKPS_PScon(void);

void forward(void);
void stop (void);
void backward (void);
void reverse (void);
void left(void);
void right(void);

void uart_send(unsigned char data);
unsigned char uart_rec(void);
unsigned char skps(unsigned char data);
void skps_vibrate(unsigned char motor, unsigned char value);
void read_adc(char config);


//  interrupt prototype
//==============================================================================================
============
static void interrupt isr(void)
{
    if (TMR0IF)                         // TMR0 is overflow
    {
        TMR0IF = 0;                     // clear flag bit
        To +=0x100;                     // count number of TMR0 overflow ( make it to 16bit
TMR)
    }

    if(RBIF)                            // there is change bit on RB4-RB7
    {
        RBIF = 0;                       //                       ____
        if (RB4)                        // Rb4 is 1 mean is rising form 0  __|
        {
            TMR0 = 0;                       // clear all counter involved, start new count for
period of RB4 high
            To = 0;
        }
                                            //                                   ___
        else TH = TMR0 + To;                // RB4 is 0 mean is falling form 1      |_____  //
save TH, RB4 high period
    }

    if(RCIF)
    {
        RCIF = 0;                           // clear flag bit

        if (RCREG == 'R') data[i=0]= RCREG;             // check if start byte 'R' is met
        else if (RCREG == 100) data[i=0]= RCREG;        // check if start byte 'd'(decimal 100)
is met
        if ((data[0] == 'R'))data [i++] = RCREG;        // save the data in data array
        if (i>4) i = 4;                                 // if the data array reached max, set
the index to 4
    }
}


//  main function
//==================================================================================================
============
void main(void)
{

    unsigned char m=0,i =0;
    delay(20000);
    init();                                     // initiate cnfiguration and initial
condition
    buzzer = 1;                                 // inditcate the circuit is on with beep
    lcd_clr();                                  // clear the LCD screen
    send_string("Select mode");                 // display "select mode"
    lcd_goto(20);                               // move to 2nd line
    send_string(mode[m]);                       // display string according to the mode
    buzzer = 0;                                 // stop beep

    while(1)                                    // loop
    {
        if( !sw1)                               // if button SW1 is pressed
        {
            while(!sw1);                        // wait ubtul button is released
            m++;
            if ( m > 4) m = 0;                  // if mode is added more than three, set to
zero
            lcd_goto(20);                       // start display at 20
            send_string(mode[m]);               // display string depend on mode
            send_string("       ");             // space to overwrite long words
        }

        if (!sw2)                               // if button SW2is pressed
        {
            while(!sw2);                        // wait until button is released
            switch(m)                           // check what is the current mode, execute
the mode
            {

                case 0 :    line_follow();      // mode 1 : line follow
                            break;
                case 1 :    ultrasonic();       // mode 2 : ultrasonic mode
                            break;
                case 2 :    analog_sen();       // mode 3 : analog sensor mode
                            break;
                case 3 :    wireless_xbee();    // mode 4 : wireless xbee mode
                            break;
                case 4 :    SKPS_PScon();       // mode 5 : PS2 Controller Mode
                            break;
                default :   ;
            }
        }
    }
}

//==================================================================================================
==========
// Initailization
// Description : Initialize the microcontroller
//==============================================================================
===========
void init()
{
    // ADC configuration
    ADCON1 = 0b10000100;                //set RA0 and RA1 as Analog Input, left justified

    // setup for capture pwm
    RBIE = 1;                           // enable interrupt on change of port B

    // motor PWM configuration
    PR2 = 255;                      // set period register
    T2CON =     0b00000100;         //
    CCP1CON =   0b00001100;         // config for RC1 to generate PWM   ( for more detail refer
datasheet section 'capture/compare/pwm')
    CCP2CON =   0b00001100;         // config for RC2 to generate PWM

    // Tris configuration (input or output)
    TRISA = 0b00000011;             //set RA0 and RA2 pin as input,other as output
    TRISB = 0b00011111;             //set RB0-RB4 pin as input, other as output
    TRISC = 0b10000000;             //set PORTC pin as output
    TRISD = 0b00000000;             //set all PORTD pin as output
    TRISE = 0b00000011;

    // TMR 0 configuation
    T0CS = 0;
    PSA = 0;
    PS2 = 1;                        // prescale 1:32
    PS1 = 1;                        //
    PS0 = 1;                        //
    TMR0IE = 1;                     // TMR0 Interrupt
    TMR0 = 0;

    //setup UART
    SPBRG = 0x81;                       //set baud rate to 9600 for 20Mhz
    BRGH = 1;                           //baud rate high speed option
    TXEN = 1;                           //enable transmission
    TX9 = 0;
    CREN = 1;                           //enable reception
    SPEN = 1;                           //enable serial port
    RX9 = 0;
    RCIE = 1;                           //enable interrupt on eachdata received

    // enable all unmasked interrupt
    GIE = 1;
    PEIE = 1;

    // LCD configuration
    send_config(0b00000001);        //clear display at lcd
    send_config(0b00000010);        //Lcd Return to home
    send_config(0b00000110);        //entry mode-cursor increase 1
    send_config(0b00001100);        //diplay on, cursor off and cursor blink off
    send_config(0b00111000);        //function

    TX_PIN = 1;
    b_light = 0;
    buzzer = 0;
    stop();
}

//==============================================================================
===========
// Mode subroutine
//==============================================================================
===========
// Mode 1 : line follow subroutine
// Description: Program for the robot to follow line
// For more detail about line follow concept please refer PR5
//==============================================================================
===========
void line_follow()
{
    unsigned char memory;
    lcd_clr();                              // clear lcd screen
    send_string("Position");                // display "position" string


    while(1)
    {
        if ((s_left==1)&&(s_mleft==0)&&(s_mright==0)&&(s_right==0))                 // if only
sensor left detected black line
        {   forward();                                                              // motor
forward
            SPEEDL = 0;                                                             // left
motor speed is 0
            SPEEDR = 255;                                                           // right
motor speed is 255(full speed)
            memory = PORTB&0b00001111;                                              // save
current sensor position
            lcd_goto(20);                                                           // lcd go to
2nd line 1st character
            send_string ("right  ");                                                // display
"right"mean the robot's position is on the right side of the line
        }
        else if ((s_left==1)&&(s_mleft==1)&&(s_mright==0)&&(s_right==0))            // if only
sensor left detected black line
        {   forward();                                                              // motor
forward
            SPEEDL = 180;                                                           // left
motor speed is 180
            SPEEDR = 255;                                                           // right
motor speed is 255(full speed)
            memory = PORTB&0b00001111;
            lcd_goto(20);
            send_string ("m_right2");
        }
        else if ((s_left==0)&&(s_mleft==1)&&(s_mright==0)&&(s_right==0))            // if only
sensor middle left detected black line
        {   forward();                                                              // motor
forward
            SPEEDL = 200;                                                           // left
motor speed is 200
            SPEEDR = 255;                                                           // right
motor speed is 255(full speed)
            memory = PORTB&0b00001111;
            lcd_goto(20);
            send_string ("m_right1  ");
        }
        else if ((s_left==1)&&(s_mleft==1)&&(s_mright==1)&&(s_right==0))            // if sensor
middle left and sensor left detected black line
        {   forward();                                                              // motor
forward
            SPEEDL = 200;                                                           // left
motor speed is 200
            SPEEDR = 255;                                                           // right
motor speed is 255(full speed)
            memory = PORTB&0b00001111;
            lcd_goto(20);
            send_string ("m_right1  ");
        }
        else if ((s_left==0)&&(s_mleft==1)&&(s_mright==1)&&(s_right==0))            // if sensor
middle left and sensor middle right detected black line
        {   forward();                                                              // motor
forward
            SPEEDL = 255;                                                           // left
motor speed is 255(full speed)
            SPEEDR = 255;                                                           // right
motor speed is 255(full speed)
            memory = PORTB&0b00001111;
            lcd_goto(20);
            send_string ("middle ");
        }
        else if ((s_left==0)&&(s_mleft==0)&&(s_mright==1)&&(s_right==0))            // if only
sensor middle right detected black line
        {   forward();                                                              // motor
forward
            SPEEDL = 255;                                                           // left
motor speed is 255(full speed)
            SPEEDR = 200;                                                           // right
motor speed is 200
```

```
            memory = PORTB&0b00001111;
            lcd_goto(20);
            send_string ("m_left1   ");
        }
        else if ((s_left==0)&&(s_mleft==1)&&(s_mright==1)&&(s_right==1))          // if sensor
middle left, sensor middle right and sensor right detected black line
        {   forward();                                                            // motor
forward
            SPEEDL = 255;                                                         // left
motor speed is 255(full speed)
            SPEEDR = 200;                                                         // right
motor speed is 200
            memory = PORTB&0b00001111;
            lcd_goto(20);
            send_string ("m_left1   ");
        }
        else if ((s_left==0)&&(s_mleft==0)&&(s_mright==1)&&(s_right==1))          // if sensor
right and sensor middle right detected black line
        {   forward();                                                            // motor
forward
            SPEEDL = 255;                                                         // left
motor speed is 255(full speed)
            SPEEDR = 180;                                                         // right
motor speed is 180
            memory = PORTB&0b00001111;
            lcd_goto(20);
            send_string ("m_left2 ");
        }
        else if ((s_left==0)&&(s_mleft==0)&&(s_mright==0)&&(s_right==1))          // if only
sensor right detected black line
        {   forward();                                                            // motor
forward
            SPEEDL = 255;                                                         // left
motor speed is 255(full speed)
            SPEEDR = 0;                                                           // right
motor speed is 0
            memory = PORTB&0b00001111;
            lcd_goto(20);
            send_string ("left   ");
        }
        else if ((s_left==0)&&(s_mleft==0)&&(s_mright==0)&&(s_right==0))          // if all
sensor coult not detected black line
        {   forward();                                                            // motor
forward
            if ((memory == 0b00000001)||(memory == 0b00000011)||(memory == 0b0000010)||(memory
== 0b0000111))
            {
                SPEEDL = 0;                                                       // left
motor speed is 0
                SPEEDR = 255;                                                     // right
motor speed is 255(full speed)
            }
            else if ((memory == 0b00001000)||(memory == 0b0000100)||(memory ==
0b00001100)||(memory == 0b0001110))
            {
                SPEEDL = 255;                                                     // left
motor speed is 255(full speed)
                SPEEDR = 0;                                                       // right
motor speed is 0
            }
        }
        else if ((s_left==1)&&(s_mleft==1)&&(s_mright==1)&&(s_right==1))          // if all
sensor detected black line
        {
            forward();                                                            // motor
forward
        }
    }
}
//=====================================================================================
============
// Mode 2 : Ultrasonic
// Description: Maintain distance measure using ultrasonic between obsacle and robot
// Can choose data acquiring methode between ADC, PWM and UART
//=====================================================================================
============
void ultrasonic(void)
```

```
{
    int distance;                                                          // variable for
distance measuring
    char n=1;                                                              // index for
indicat mode
    lcd_clr();                                                             // clear lcd
    send_string("Measure mode");                                           // display
string
    lcd_goto(20);                                                          // lcd goto 2nd
line
    send_string("ADC");                                                    // Display
string "ADC"

    while(1)                                                               // loop forever
unless sw2 is pressed
    {
        if( !sw1)                                                          // if button s1
is pressed
        {
            while(!sw1);                                                   // wait until
sw1 is release
            n++;                                                           // increment n
            lcd_goto(20);                                                  // goto 2nd line
            switch (n)                                                     // check current
value of n
            {
                case 2 :    send_string("PWM");
                            break;                                         // break out
from switch
                case 3 :    send_string("UART");
                            break;                                         // break out
from switch
                default:    send_string("ADC");                            // if not 2 or
3, set ti back to 1 and display "ADC"
                            n =1;
            }

        }

        if (!sw2)                                                          // if button sw2
is pressed
        {
            while(!sw2);                                                   // wait until
sw2 is release
            break;                                                         // break out
form looping
        }
    }


    lcd_clr();                                                             // clear the lcd
    send_string("Distance");                                               // display
string "Distance"

    while(1)                                                               // loop forever
    {
        lcd_goto(20);                                                      // lcd goto 2nd
line
        distance = us_value(n);                                            // disance
variable are equal to value return from subroutine us_value
        dis_num(distance);                                                 // display the
value of distance


        if (distance> 40)                                                  // check if
distance more than 40
        {
            forward();                                                     // then forward
with full speed
            SPEEDL = 255;
            SPEEDR = 255;
            buzzer = 0;
        }
        else if (distance> 30)                                             // check if
distance more than 40
        {
            forward();                                                     // forward with
```

```
medium speed
            SPEEDL = 230;
            SPEEDR = 230;
            buzzer = 0;
        }
        else if( distance >20)                                                  // check if
distance more than 40
        {
            stop();                                                             // then stop
            buzzer = 0;
        }
        else                                                                    // else,
distance less than 20
        {
            backward();                                                         // then backward
with medium speed and on the buzzer
            SPEEDL = 230;
            SPEEDR = 230;
            buzzer = 1;
        }
    }

}

//==================================================================================================
============
// Mode 3 : Analog Distance Sensor
// Description : Maintain distance between robot and obstacle
//==================================================================================================
============
void analog_sen(void)
{
    int distance;                                                               // variable for
distance measuring
    lcd_clr();                                                                  // clear lcd
    send_string("Distance");                                                    // display
string

    while(1)
    {
        lcd_goto(20);                                                       // lcd goto 2nd line
        read_adc(CHANNEL1);                                                 // read adc channel
1 ( analog distance sensor input)
        distance = result;                                                  // assign distance
as the result oft he reading
        dis_num(result);                                                    // display the
value

        /*if (distance< 200)                                                    // check if
distance less than 200
        {
            backward();                                                     // backward with
full speed
            SPEEDL = 255;
            SPEEDR = 255;
            buzzer = 0;
        }*/
        if (distance<450 )                                              // check if distance
less than 250
        {
            backward();                                                     // backward with
medium speed
            SPEEDL = 230;
            SPEEDR = 230;
            buzzer = 0;
        }
        /*else if( distance < 300)                                          // check if distance
less than 300
        {
            stop();                                                         // stop
            buzzer = 0;
        }*/
        else if (distance>450)                                                          // else,
distance more than 300
        {
            right();                                                        //forward with
medium speed and on buzzer
```

```
            SPEEDL = 200;
            SPEEDR = 100;
            delay(100000);
            right();
            SPEEDL = 200;
            SPEEDR = 100;
        }
    }
}

//=====================================================================================
============
//Mode 4 :  Xbee
// Description : Control the robot using UART ( XBEE or an UART wireless module.
//=====================================================================================
============
void wireless_xbee (void)
{
    lcd_clr();                                                              // clear the lcd
    while(1)                                                                // looping forever
    {
        lcd_goto (0);
        if (data[0] == 100)                                                 // check if UART
start byte is met
        {
            send_string("  XBEE CONTROL  ");                                // display string
            SPEEDL = 200;                                                   // set the motor
speed
            SPEEDR = 200;
            while(1)
            {
                lcd_goto(20);
                if (RCREG == '8')                                               // if character
'8' is detected, the robot move forward
                {
                    forward();
                    send_string("FORWARD        ");
                }

                else if (RCREG == '2')                                          // if character
'2' is detected, the robot move backward
                {
                    backward();
                    send_string("BACKWARD        ");
                }
                else if (RCREG == '6')                                          // if character
'6' is detected, the robot turn right
                {
                    right();
                    send_string("TURN RIGHT      ");
                }

                else if (RCREG == '4')                                          // if character
'4' is detected, the robot turn left
                {
                    left();
                    send_string("TURN LEFT       ");
                }

                else if (RCREG == '5')                                          // if character
'5' is detected, then stop the robot
                {
                    stop();
                    send_string("INVALID COMMAND  ");
                }

                else                                                            // else then
stop the robot
                {
                    stop();
                    send_string("INVALID COMMAND  ");
                }
            }
        }
        else    send_string("COMMAND");
    }
}
```

```
//==============================================================================================
==
// Mode 5: SKPS
// Description: Control the robot using PS Controller
//==============================================================================================
==
void SKPS_PScon()
{
    unsigned char up_v, down_v, left_v, right_v;

    //Disabled Intterupt
    RCIE = 0;
    GIE = 0;
    PEIE = 0;
    TMR0IE = 0;

    stop();         //stop all motor

    lcd_clr();
    lcd_goto(0);
    send_string("Controlling PR23");
    lcd_goto(20);
    send_string("   Using SKPS");

    while(1)
    {
        //test button for horn
        if(skps(p_l1)==0)buzzer=1;          //if button L1 pressed, beep the buzzer
        else buzzer=0;

        //read joy stick value process
        up_v=skps(p_joy_ru);
        down_v=skps(p_joy_rd);
        left_v=skps(p_joy_ll);
        right_v=skps(p_joy_lr);

        //button control for mobilility
        if(skps(p_up)==0)                   //check "up" button
        {
            forward();                      //move forward
            SPEEDL=230;
            SPEEDR=230;
        }
        else if(skps(p_down)==0)            //check "down" button
        {
            backward();                     //move backward
            SPEEDL=230;
            SPEEDR=230;
        }
        else if(skps(p_left)==0)            //check "left" button
        {
            left();                         //rotate left
            SPEEDL=230;
            SPEEDR=230;
        }
        else if(skps(p_right)==0)           //check "right" button
        {
            right();                        //rotate right
            SPEEDL=230;
            SPEEDR=230;
        }

        //analog control for mobility
        else if(up_v>0)
        {
            forward();
            if(left_v>0)
            {
                if(up_v>left_v)SPEEDL=up_v-left_v+140;
                else SPEEDL=140;
                SPEEDR=up_v+140;
            }
            else if(right_v>0)
            {
                if(up_v>right_v)SPEEDR=up_v-right_v+140;
                else SPEEDR=140;
                SPEEDL=up_v+140;
            }
            else
            {
                SPEEDL=up_v+140;
                SPEEDR=up_v+140;
            }
        }
        else if(down_v>0)
        {
            backward();
            if(left_v>0)
            {
                if(down_v>left_v)SPEEDR=down_v-left_v+140;
                else SPEEDR=140;
                SPEEDL=down_v+140;
            }
            else if(right_v>0)
            {
                if(down_v>right_v)SPEEDL=down_v-right_v+140;
                else SPEEDL=140;
                SPEEDR=down_v+140;
            }
            else
            {
                SPEEDL=down_v+140;
                SPEEDR=down_v+140;
            }
        }
        else if(left_v>0)
        {
            left();
            SPEEDL=left_v+120;
            SPEEDR=left_v+120;
        }
        else if(right_v>0)
        {
            right();
            SPEEDL=right_v+120;
            SPEEDR=right_v+120;
        }
        else
        {
            stop();
            SPEEDL=255;
            SPEEDR=255;
        }
    }
}

//==========================================================================================
====
// Ultrasonic value
// Description : Retrive data from Ultrsonic. Can choose methode between ADC, PWM and UART
// Parameter : mode     1) using analog
//                      2) using pwm
//                      3) using uart
//==========================================================================================
====
unsigned int us_value (unsigned char mode)                                          //
subroutine for ultrasonic measurement
{
    unsigned int value;
    switch (mode)                                                                   // retrive
value of measured distane based on the methode selected
    {
        case 1: read_adc(CHANNEL0);
                value = result;                                                     // max vslue
2.55v = 2.55/5 *1024 - 1 =  522, resolution = 10mV/ inch, 10m/5*1024 =~ 2
                break;
        case 2: value = TH;                                                         // each
value = 256*4/20mhz = 51.2us, i inch = 147us
                break;                                                              // can
change using smaller timer prescale, but resulation fixed 147us / inch
        case 3: if ( data [0]=='R') value = (data[1] - 0x30)*100+ (data[2] - 0x30)*10+ (data[3]
- 0x30); // 1 = 1 inch
                else
                {
                    lcd_goto(20);                                                       // if stater
byte is not 'R', Display 'not connected'
                    send_string("not connected");
                    while(1);                                                           // loop
forever
                }
          default: ;
    }
    return value;
}


//=================================================================================================
===========
// read adc
// Description: subroutine for converting analog value to digital with average 200 samples
// Parameter : config  ( select the channel )
//=================================================================================================
===========
void read_adc(char config)
{
    unsigned short i;
    unsigned long result_temp=0;

    ADCON0 = config;
    delay(10000);                   // delay after changing configuration
    for(i=200;i>0;i-=1)             //looping 200 times for getting average value
    {
        ADGO = 1;                   //ADGO is the bit 2 of the ADCON0 register
        while(ADGO==1);             //ADC start, ADGO=0 after finish ADC progress
        result=ADRESH;
        result=result<<8;           //shift to left for 8 bit
        result=result|ADRESL;       //10 bit result from ADC

        result_temp+=result;
    }
    result = result_temp/200;       //getting average value
    ADON = 0;                       //adc module is shut off
}


//=================================================================================================
===========
// Motor control function
// Description : subroutine to set the robot moving direction
//=================================================================================================
===========
void forward ()
{
    motor_ra = 0;
    motor_rb = 1;
    motor_la = 0;
    motor_lb = 1;
}

void backward ()
{
    motor_ra = 1;
    motor_rb = 0;
    motor_la = 1;
    motor_lb = 0;
}

void left()
{
    motor_la = 1;
    motor_lb = 0;
    motor_ra = 0;
    motor_rb = 1;
}
void right()
{
    motor_la = 0;
    motor_lb = 1;
    motor_ra = 1;
    motor_rb = 0;
```

```
}

void stop()
{
    motor_la = 1;
    motor_lb = 1;
    motor_ra = 1;
    motor_rb = 1;
}

//=================================================================================================
===========
//  LCD functions
//=================================================================================================
============
void delay(unsigned long data)                  //delay function, the delay time
{
    for( ;data>0;data-=1);                      //depend on the given value
}

void send_config(unsigned char data)            //send lcd configuration
{
    rs=0;                                       //set lcd to config mode
    lcd_data=data;                              //lcd data port = data
    delay(400);
    e_pulse();                                  //pulse e to confirm the data
}

void send_char(unsigned char data)              //send lcd character
{
    rs=1;                                       //set lcd to display mode
    lcd_data=data;                              //lcd data port = data
    delay(400);
    e_pulse();                                  //pulse e to confirm the data
}

void e_pulse(void)                              //pulse e to confirm the data
{
    e=1;
    delay(300);
    e=0;
    delay(300);
}

void lcd_goto(unsigned char data)               //set the location of the lcd cursor
{
    if(data<16)                                 //if the given value is (0-15) the
    {                                           //cursor will be at the upper line
        send_config(0x80+data);
    }
    else                                        //if the given value is (20-35) the
    {                                           //cursor will be at the lower line
        data=data-20;                           //location of the lcd cursor(2X16):
        send_config(0xc0+data);                 //
-----------------------------------------------------
    }                                           // |
|00|01|02|03|04|05|06|07|08|09|10|11|12|13|14|15| |
}                                               // |
|20|21|22|23|24|25|26|27|28|29|30|31|32|33|34|35| |
                                                //
-----------------------------------------------------

void lcd_clr(void)                              //clear the lcd
{
    send_config(0x01);
    delay(350);
}

void send_string(const char *s)                 //send a string to display in the lcd
{
    while (s && *s)send_char (*s++);
}

void dis_num(unsigned long data)
{
    unsigned char hundred_thousand;
    unsigned char ten_thousand;
    unsigned char thousand;
    unsigned char hundred;
    unsigned char tenth;

    hundred_thousand = data/100000;                     // devide to get the numerator      eg:
5234/1000 = 5
    data = data % 100000;                               // modulas to get the remainder     eg:
5234%1000 = 234
    ten_thousand = data/10000;
    data = data % 10000;
    thousand = data / 1000;
    data = data % 1000;
    hundred = data / 100;
    data = data % 100;
    tenth = data / 10;
    data = data % 10;

    send_char(hundred_thousand + 0x30);                 //0x30 added to become ASCII code char
'0' to '9'
    send_char(ten_thousand + 0x30);
    send_char(thousand + 0x30);
    send_char(hundred + 0x30);
    send_char(tenth + 0x30);
    send_char(data + 0x30);
}

//================================================================================================
====
// uart function
//================================================================================================
====
void uart_send(unsigned char data)  //function to send out a byte via uart
{
    while(TXIF==0);                 //wait for previous data to finish send out
    TXREG=data;                     //send new data
}

unsigned char uart_rec(void)        //function to wait for a byte receive from uart
{
    unsigned char temp;
    while(RCIF==0);                 //wait for data to received
    temp=RCREG;
    return temp;                    //return the received data
}

//================================================================================================
==
// skps function
//================================================================================================
==
unsigned char skps(unsigned char data)          //function to read button and joystick
{                                               //information on ps controller
    uart_send(data);
    return uart_rec();
}

void skps_vibrate(unsigned char motor, unsigned char value)
{                                               //function to control the vibrator motor
    uart_send(motor);                           //on ps controller
    uart_send(value);
}
```

GP2Y0A21YK/ GP2Y0D21YK

General Purpose Type Distance Measuring Sensors

Features

- Less influence on the color of reflective objects, reflectivity
- Line-up of distance output/distance judgement type
  Distance output type (analog voltage) : **GP2Y0A21YK**
  Detecting distance : 10 to 80cm
  Distance judgement type : **GP2Y0D21YK**
  Judgement distance : 24cm
  (Adjustable within the range of 10 to 80cm [Optionally available])
- External control circuit is unnecessary
- Low cost

Applications

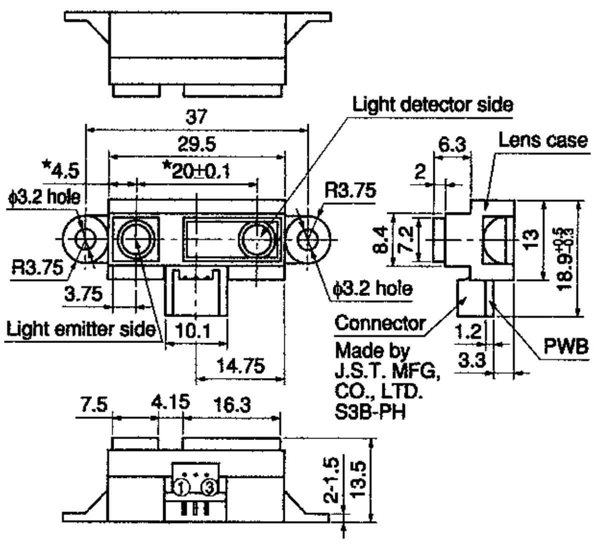
- TVs
- Personal computers
- Cars
- Copiers

Absolute Maximum Ratings (T_a=25°C, V_{CC}=5V)

Parameter	Symbol	Rating	Unit
upply voltage	V_{CC}	–0.3 to +7	V
utput terminal voltage	V_O	-0.3 to V_{CC} +0.3	V
perating temperature	T_{opr}	–10 to +60	°C
torage temperature	T_{stg}	–40 to +70	°C

Outline Dimensions

(Unit : mm)



Terminal connection
① V_O
② GND
③ V_{CC}

✻ The dimensions marked * are described the dimensions of lens center position.
✻ Unspecified tolerance : ±0.3mm

Recommended Operating Conditions

Parameter	Symbol	Rating	Unit
)perating supply voltage	V_{CC}	4.5 to +5.5	V

Electro-optical Characteristics

(T_a=25°C, V_{CC}=5V)

Parameter		Symbol	Conditions	MIN.	TYP.	MAX.	Unit
)istance measuring range		ΔL	[1] [3]	10	–	80	cm
)utput terminal voltage	GP2Y0A21YK	V_O	L=80cm [1]	0.25	0.4	0.55	V
	GP2Y0D21YK	V_{OH}	Output voltage at High [1]	V_{CC} −0.3	–	–	V
		V_{OL}	Output voltage at Low[1]	–	–	0.6	V
)ifference of output voltage	GP2Y0A21YK	ΔV_O	Output change at L=80cm to 10cm[1]	1.65	1.9	2.15	V
)istance characteristics ıf output	GP2Y0D21YK	V_O	[1] [4] [2]	21	24	27	cm
∖verage Dissipation current		I_{CC}	L=80cm [1]	–	30	40	mA

ɔte) L : Distance to reflective object
Using reflective object : White paper (Made by Kodak Co. Ltd. gray cards R-27 · white face, reflective ratio ; 90%)
We ship the device after the following adjustment : Output switching distance L=24cm±3cm must be measured by the sensor
Distance measuring range of the optical sensor system
Output switching has a hysteresis width. The distance specified by Vo should be the one with which the output L switches to the output H

ig.1 Internal Block Diagram

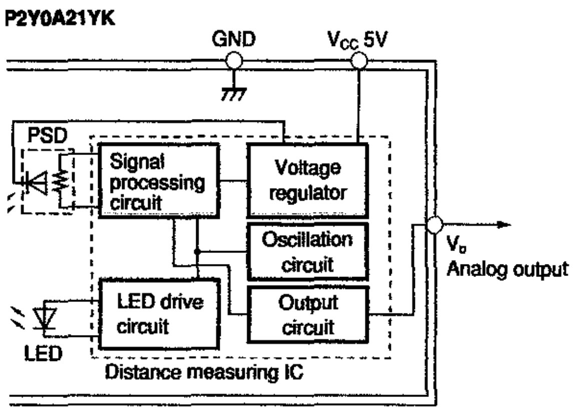
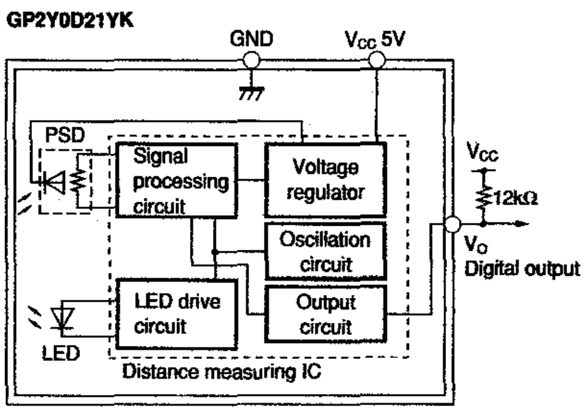
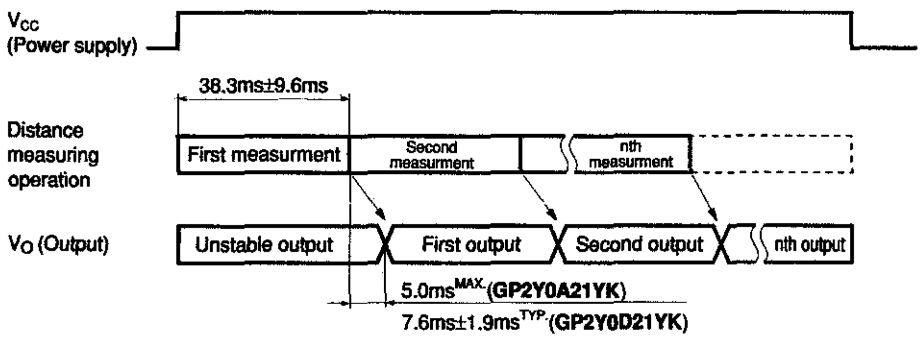


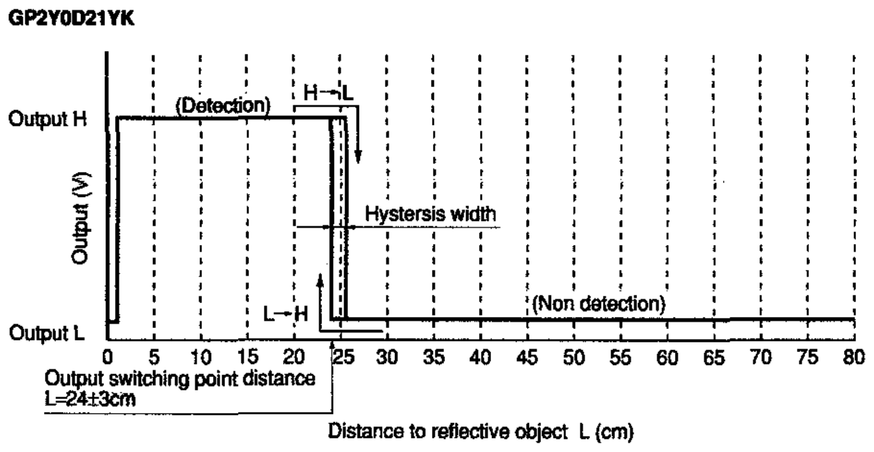
Fig.2 Internal Block Diagram



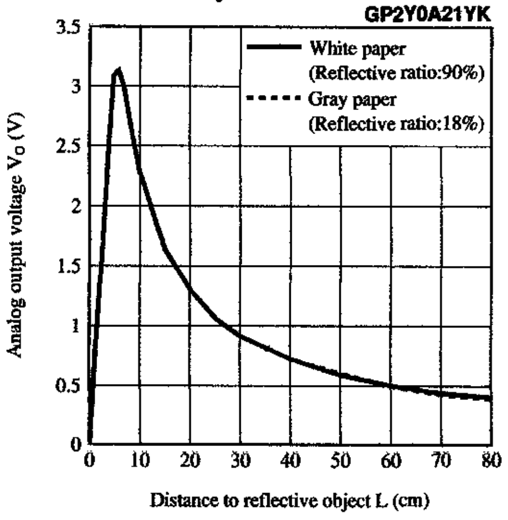
ɪg.3 Timing Chart



ig.4 Distance Characteristics



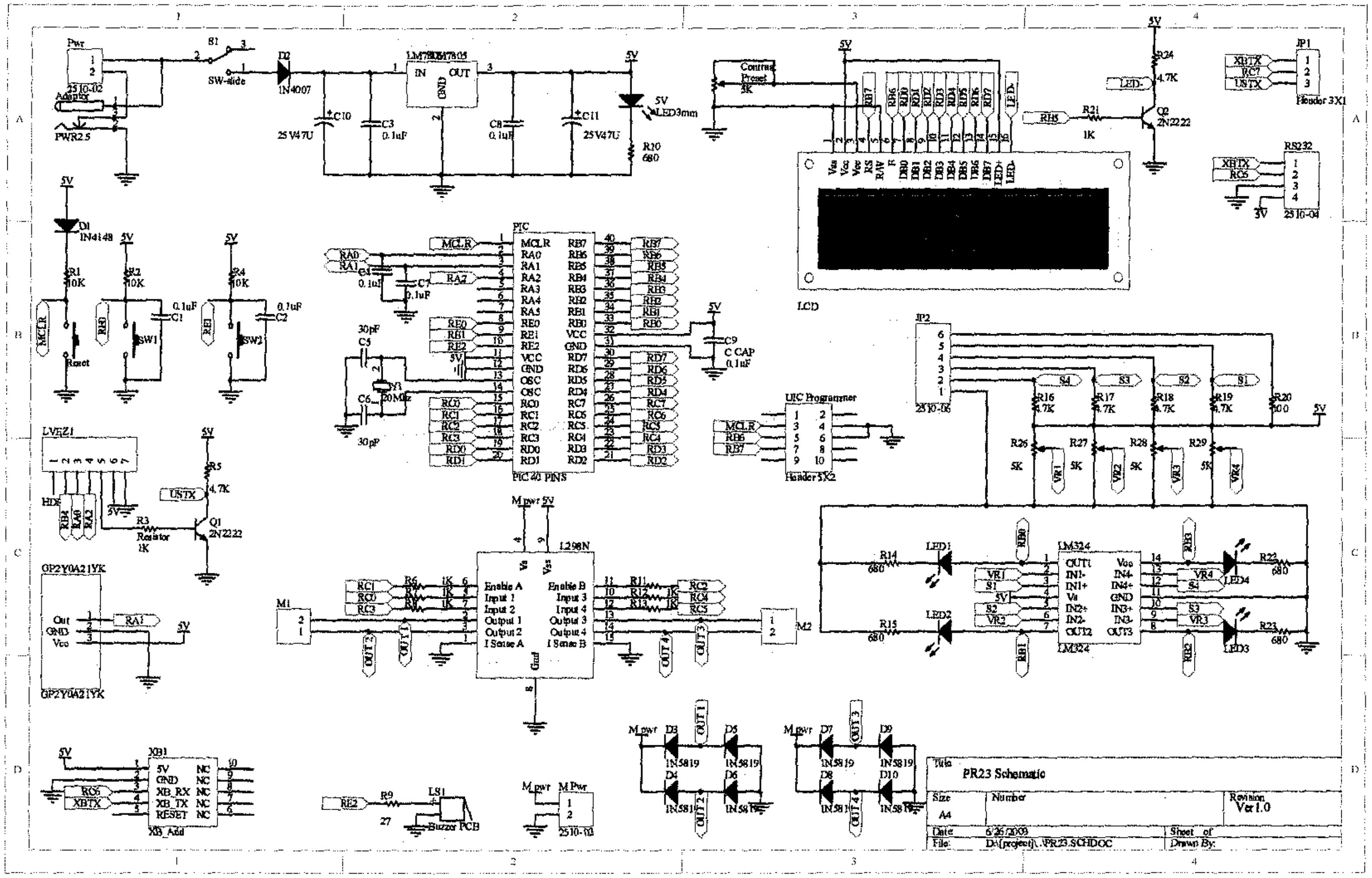
ig.5 Analog Output Voltage vs. Distance to Reflective Object



PR23 Hardware List

No.	Component	Product Code	Unit
1	IC PIC16F877A	IC-PIC-16F877A	1
2	IC Socket-40 pin	IS-40PIN	1
3	Crystal H49S (Low Profile) 20MHz	CR-H49S-20M	1
4	Voltage Regulator +5V	VR-7805	1
5	Diode 1N4007	DI-1N4007	1
6	Diode 1N4148	DI-1N4148	1
7	Diode 1N5819	DI-1N5819	8
8	Electrolytic Capacitor 25V 47uF	CP-EC-25-47	2
9	Multilayer Capacitor 0.1uF	CP-MC-0.1UF	7
10	Ceramic Capacitor 30pF	CP-CC-30PF	2
11	Slide Switch 3 Pins Black medium	SW-SL-3N-061206	1
12	Buzzer PCB Mount	SO-BUZZ-PCB	1
13	LED 3mm Red	DS-LED-3NR	4
14	LED 3mm Green	DS-LED-3NG	1
15	IC L298	IC-L-298	1
16	DC Plug (Adaptor Socket)	CN-HL-2527-B	1
17	6x6x1 Push Button 4 Pins	SW-PBM-4N-060601	3
18	Mini Jumper	CN-PH-MJ	1
19	IC Socket-14 pin	IS-14PIN	1
20	IC LM324	IC-LM-324	1
21	2510 PCB Connector 6 Ways	CN-06-2510	1
22	2510 PCB Connector 4 Ways	CN-04-2510	1
23	2510 PCB Connector 2 Ways	CN-02-2510	4
24	Right Angle Pin Header (Male) 7 Ways	CN-PH-M140R	1
25	LCD (16x2)	DS-LCD-JHD162A	1
26	Straight Pin Header(Male) 16 Ways	CN-PH-M140S	1
27	Transistor 2N2222	TR-2N-2222	2
28	Straight 2mm Female Header 3 Ways	CN-PH-F110S	1
29	Straight Pin Header(Male) 3 Ways	CN-PH-M140S	1
30	10 Ways Straight Box Header	CN-IDC-BOX-10	1
31	PCB for PR23	PCB-PR-023	1
32	IR Sensor Set (4 Sensor)	SN-IRS-04	1
	- IR LED x 4		
	- 2510-06 (R/A)		
	- IR Snesor PCB Board		
33	PCB Stand (screw & screw)8mm	SD-SS-08	2
34	Rainbow Cable 6 Ways (20cm Length)	WR-RM10	1
35	9V Battery Snap	CN-BA-9S	1
36	PCB Stand (screw & screw)30mm	SD-SS-30	4
37	Battery Holder UM3x4 (NX817A)	CN-BA-UM3-04-A	1
38	Mobile Robot Base Set	HD-BSC	1
39	Resistor 1/4W 27R	RS-025W-27R	1
40	Resistor 1/4W 100R	RS-025W-100R	1
41	Resistor 1/4W 680R	RS-025W-680R	5
42	Resistor 1/4W 1K	RS-025W-1K	8
43	Resistor 1/4W 4K7	RS-025W-4K7	6
44	Resistor 1/4W 10K	RS-025W-10K	3
45	Female Header 5 ways	CN-PH-F110S	2

46	Preset	RS-RM-065-5K-H	5
	Optional		
46	UIC00A Programmer	UIC00A Programmer	1
47	Analog Distance sensor (SHARP)	SN-GP2Y0A21	1
48	Ultrasonic Range Finder	SN-LV-EZ1	1
49	Xbee Starter Kit(MaxStream)	SKXBEE	1


Pwr
2510-02
Adaptor
PWR2.5
S1
SW-slide
D2
1N4007
LM7805
IN
OUT
GND
C10
25V47U
C3
0.1uF
C8
0.1uF
C11
25V47U
5V
LED3mm
R10
680
Contrast Preset
5K
Vss Vcc Vee RS R/W E DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7 LED+ LED-
LCD
R24
4.7K
LED-
RB5
R21
1K
Q2
2N2222
JP1
XBTX
RC7
USTX
Header 3X1
RS232
XBTX
RC6
2510-04
D1
1N4148
R1
10K
R2
10K
R4
10K
C1
0.1uF
C2
0.1uF
MCLR
RB0
RE1
Reset
SW1
SW2
PIC
MCLR RA0 RA1 RA2 RA3 RA4 RA5 RE0 RE1 RE2 VCC GND OSC OSC RC0 RC1 RC2 RC3 RD0 RD1
RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0 VCC GND RD7 RD6 RD5 RD4 RC7 RC6 RC5 RC4 RD3 RD2
PIC 40 PINS
C4
0.1uF
C7
0.1uF
30pF
C5
C6
Y1
20MHz
30pF
C9
C CAP
0.1uF
JP2
2510-06
S4 S3 S2 S1
R16 4.7K
R17 4.7K
R18 4.7K
R19 4.7K
R20 100
R26 5K
R27 5K
R28 5K
R29 5K
VR1 VR2 VR3 VR4
UIC Programmer
Header 5X2
LVEZ1
HDF
RB4 RA0 RA2
R5
4.7K
USTX
R3
Resistor
1K
Q1
2N2222
GP2Y0A21YK
Out
GND
Vcc
RA1
M pwr
L298N
Enable A
Input 1
Input 2
Output 1
Output 2
I Sense A
Enable B
Input 3
Input 4
Output 3
Output 4
I Sense B
Vs
Vss
Gnd
R6 R7 R8 1K
R11 R12 R13 1K
RC1 RC0 RC3
RC2 RC4 RC5
M1
M2
OUT 1 OUT 2 OUT 3 OUT 4
LM324
OUT1 IN1- IN1+ Vs IN2+ IN2- OUT2
Vcc IN4- IN4+ GND IN3+ IN3- OUT3
R14 680
R15 680
R22 680
R23 680
LED1 LED2 LED3 LED4
RB0 RB1 RB2 RB3
D3 D4 D5 D6 D7 D8 D9 D10
1N5819
XB1
5V GND XB_RX XB_TX RESET NC
XB_Add
RC6
XBTX
RE2
R9
27
LS1
Buzzer PCB
M Pwr
2510-02
Title
PR23 Schematic
Size
A4
Number
Revision
Ver1.0
Date
6/26/2009
File
D:\[project]\..\PR23.SCHDOC
Sheet of
Drawn By: