Behaviour- Based Control of Mobile Robot.

Mohammed Shazil
Student Id No.: S11030422

Edwin Avinesh Kumar
Student Id No.: S11031073

School of Engineering and Physics
Faculty of Science and Technology
University of the South Pacific

November 2008.

Supervisor
Mr. Praneel Chand
School of Engineering and Physics
Faculty of Science and Technology
University of the South Pacific

A report submitted in fulfillment of the requirements for the degree of Bachelor of Engineering Technology.

Declaration of Originality

We, Mohammed Shazil and Edwin Avinesh Kumar, hereby declare that this project report, is to the best of our knowledge original and it has not been copied from anywhere. All the cited materials have been referenced in the relevant section.

_____ _____

Mohammed Shazil

(Student Id No. S11030422)

Edwin Avinesh Kumar

(Student Id No. S11031073)

12th November, 2008.

Acknowledgments

We would like to take this opportunity to thank all those who have helped us in some way or the other and without whom this project may not have been successful.

Firstly, we would like to thank our supervisor Mr. Praneel Chand for allocating this project to us and also for his continual support and guidance throughout the project. We would also like to thank the EN300 Coordinators Mr. Ravinesh Singh and Dr. M. Daniel K. Wood for their help in arranging for the project parts.

We are also sincerely thankful to Mr. Roneel Sharan and Mr. Imran Jannif for their interest in our project and for helping us in the program development.

We are really grateful to Mr. Jeff La'ava and Mr. Sanjay Singh for their support in the labs and for working extra hours so that the students could complete their projects.

Finally, we would like to thank all our friends and colleagues who may have helped us in any way.

List of figures

List of tables

Abstract

This project report looks at behaviour-based methods used for the control of an autonomous mobile robot. There are several distinct behaviours that can be used for the completion of the task required for the robot. The main behaviours of the robot can be said to be object detection, object handling and transportation, obstacle avoidance and goal detection. The behaviour-based scheme attempts to fulfill simple missions in which several behaviours compete for the vehicle's control. A small-assembled robot model (Squarebot) is programmed to autonomously perform the task following the behavior–based scheme. The model is also modified to an extent so that it can easily and successfully perform all the operations. The model is evaluated in its ability to perform the assigned tasks, such as object detection and its manipulation. Furthermore, discussion about the proposed approach is given, as well as an overall description of the tasks to be performed by the robot.

Table of Contents

Chapter 1

Introduction and Literature Review

1.1 Overview

Behavior-based robotics is the dominant approach to robot design today. Based on the Behaviorist line of research in psychology, a behavior-based robotic system divides control of a robot into small, independent behavioral modules, whose combined efforts produce the final actions of the robot [1]. Behavior- based control is an interesting but difficult approach of controlling mobile robots. The behavior-based approach has produced intelligent systems for use in a wide variety of areas, including military applications, mining, space exploration, agriculture, factory automation, service industries, waste management, health care, disaster intervention and the home [2]. Behavior- based approaches are an extension of reactive architectures and fall between purely reactive and planner- based extremes [3]. The main idea of this approach is to identify different controllers and responses to the different sensor inputs, with the desired robot behaviors. This report shows us a solution to the problem of object detection, handling and its transportation using behaviour- based approach. A general overview of behavior-based robotics is presented in the next chapter.

This chapter provides a background on some of the successful past projects related to this project. Furthermore, a brief discussion about VEX Robotics has been presented. The relevant control architectures for mobile autonomous robots has also been discussed. Finally, a detailed overview of the objectives of this project has been given.

1.2 VEX Robotics

The Vex Robotics Design System is a Carnegie Mellon University designed robotic kit. It seeks to enter the arena of robotics education already dominated by such offerings as the LEGO Mindstorms Robotics Invention system and the outgoing Robovation Kit. Despite this stiff competition, the VEX Kit certainly has many

sterling qualities that will allow it to work with and not necessarily against other robotics education systems [4].

One of the best parts of the VEX system is that it pays particular attention to the interactions between the different subsystems that compose the robot (structure, power, logic, etc). This feature sets the VEX system apart from other robotics kits, and it very closely approximates the experience of building real robots [4].

The Vex Robotics Design System is quickly becoming the robotics platform of choice in middle school, high school, technical schools and university classrooms. Versatile, inexpensive and accessible, it is also an ideal platform for student robotic competitions. Beginning builders can design, assemble and quickly iterate their robots through trial and error. Advanced builders can utilize sophisticated programming capabilities to power highly intelligent bots [5].

Vex has provided the most fruitful engineering design platform in all the years from when it has first been developed. The technology is from FIRST (For Inspiration and Recognition of Science and Technology), an ideal program for enthusing students to apply theory and practice real problem solving. By relaxing the open-endedness of the hardware fabrication, Vex increases the ability to do engineering exploration [6].

The VEX Starter Kit includes an Inventor's Guide which guides the user through the construction of what Innovation First refers to as the Squarebot. This is a basic four-wheeled robot, which seemed like a sound way to start our experiment in robot building.

The Vex Robotics Starter Kit is composed of more than 500 pieces. While Vex robots use plastic for wheels and gears, they are constructed mostly from metal struts, axles, and plates, making them much stronger than many other robotic kits. Motive force is provided by three motors that can rotate continuously clockwise or counterclockwise and one servo-motor that moves forward and back through a 120-degree arc, it also has sensors included with it which can be used to autonomously control the robot and they are flexible enough to enable a wide range of robot designs.

1.2.1 SquareBot

The SquareBot is basically a square of girders with four wheels, two motors, batteries, the microcontroller and RF receiver. It is easy but satisfying to assemble from the VEX Design System. One pleasant surprise was that the robot needs absolutely no tinkering after completion. As long as the connectors are attached to the right ports and the settings of the microcontroller defaulted to this robot, it can easily be controlled using the Radio Control that comes with the design kit.

With the use of some sensors, the robot can be made completely autonomous or both autonomous and partially radio controlled. The SquareBot design has been used for our project with significant modifications made to it, so that it is capable of achieving the set objectives successfully. A variety of sensors has also been attached to it for the purpose of the project.

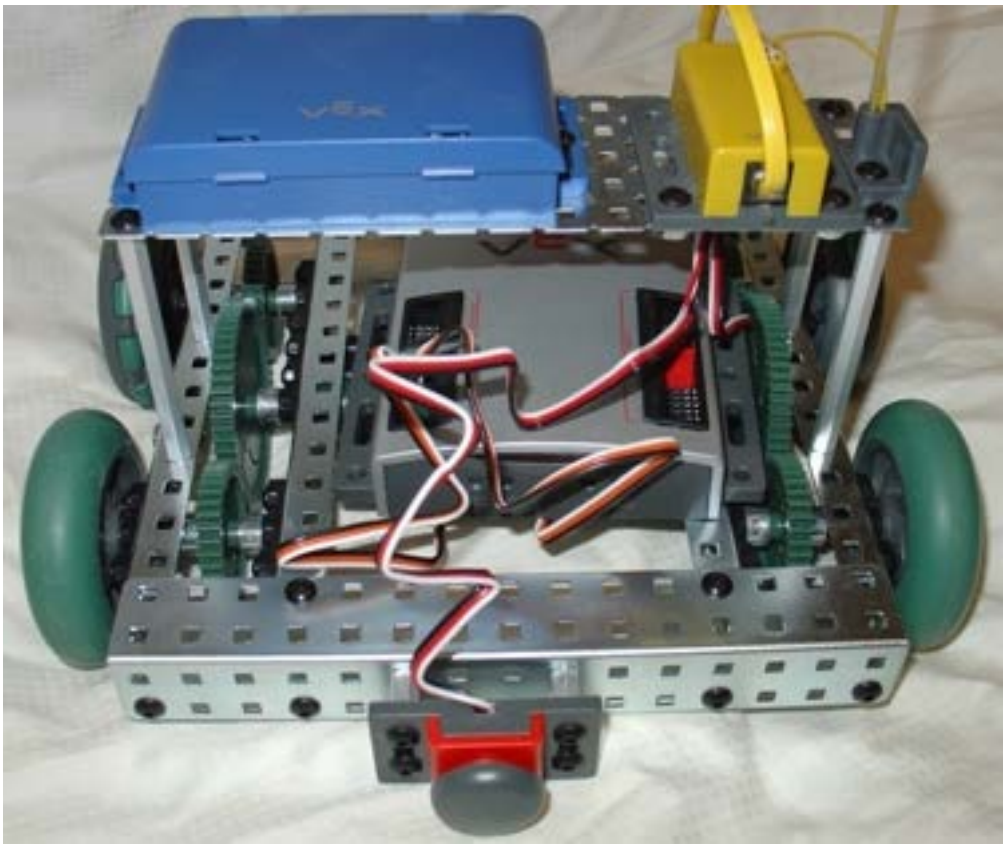


Figure 1.1 Squarebot.

1.2.2 VEXplorer

VEXplorer is the next generation of an important DIY (Do-it-yourself) robotics design system. It has over 300 parts with many pre-assembled modules which

includes: a spycam that can broadcast to your TV over distances of up to 150 ft.; an enhanced robot arm with a powerful gripper that can pick up a filled soda can; and all-terrain tires to tackle most surfaces. VEXplorer also includes 4 motor, 24 gears, various coated heavy metal parts, cables, tools, screws, nuts and more.

The VEXplorer robot kit is a direct descendent of the VEX robot kit that is the basis for student robot design competitions. It is designed and marketed to "bring robotics creation outside the classroom taking the consumer robotics category to new heights."

VEXplorer includes all the parts, motors and wheels to make a remote controlled rolling robot with a webcam and a claw arm strong enough to pick up a can of soda. The kit, of course, can be used to build any kind of robot imaginable.

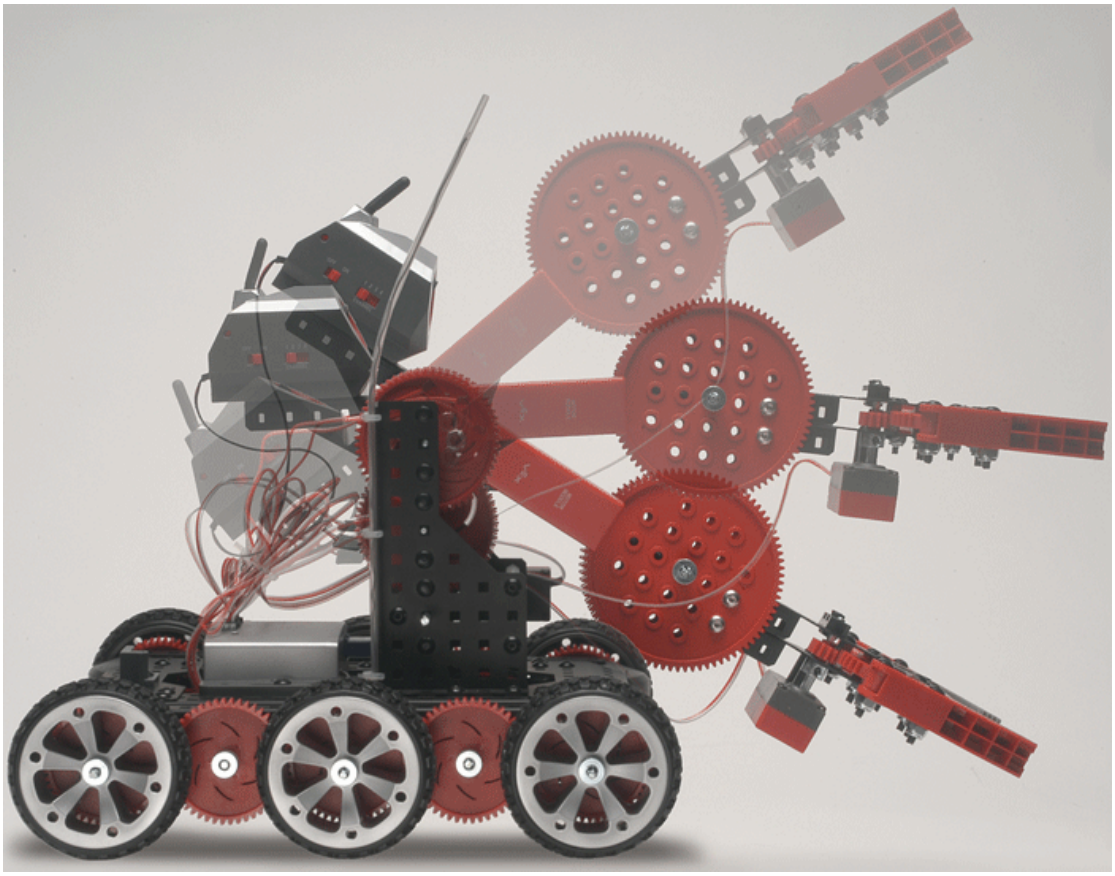


Figure 1.2 VEXplorer.

1.2.2 Protobot Robotic Kit

The Protobot Robot Kit provides the perfect foundation for any Robotics experience. The instructions on the manual can be followed to build the VEX Protobot or VEX Tumbler, or use the 300+ included parts to design various other VEX Robotic creations.

Figure 1.3 VEX Protobot Kit.

1.2.3 VEX HEXBUGS

1.2.3.1 Original HEXBUG

HEXBUG is a crawling bug that feels its way around, sensing objects in its path and avoiding them. It is of small size that can fit in the hands and it also has the ability to hear. It can be controlled by sound, where it scurries through a handclap or a loud noise. The robotic bug will travel forward until it hits an obstacle or hears a loud noise. It will then backup in a half circle and then move forward in a different direction. HEXBUG comes in five distinct shapes and colors [7].

Figure 1.4 Original HEXBUG.

1.2.3.2 HEXBUG Inchworm

HEXBUG Inchworm is also a crawling bug that features 2 IR Channels. It can change directions with the 7-way steering. It is also a similar size as the Original HEXBUG.

Figure 1.5 HEXBUG Inchworm.

1.2.3.3 HEXBUG Crab

The HEXBUG Crab is another crawling bug that seeks out dark places to hide. It is of small size, which can fit in the hands and it also has the ability to hear. A handclap or a loud noise can change its direction of movement.

Figure 1.6 HEXBUG Crab

1.2.4 Other VEX Robotics Models

Similarly, there are many other design that can be created for specific purposes from the many VEX Robotics Design Kits available. A sample robot from VEX (The Tomahawk) is shown below.

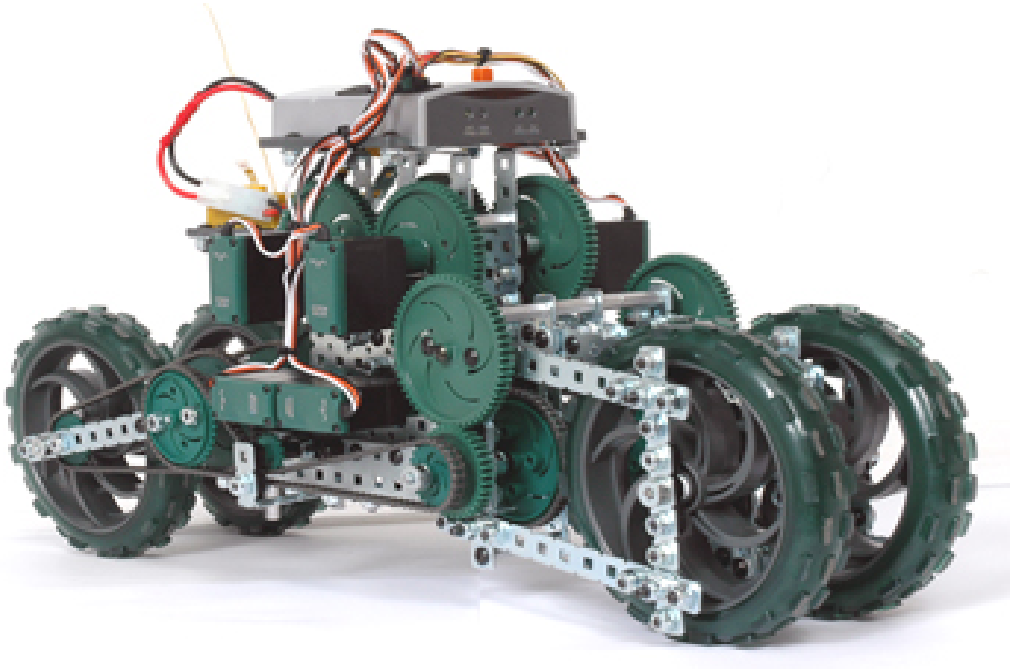


Figure 1.7 VEX Tomahawk

1.3 Microcontroller Overview

Microcontrollers are functional computer systems embedded in a chip. They contain a processor core, memory (a small amount of RAM, program memory, or both), and programmable input/output peripherals. Microcontrollers are usually "embedded" inside some other device (often a consumer product) so that they can control the features or actions of the product. They are used in automatically controlled products and devices, such as automobile engine control systems, remote controls, office machines, appliances, power tools, and toys. Microcontrollers are dedicated to one task and run one specific program. The program is stored in ROM (read-only memory) and generally does not change. By reducing the size, cost, and power consumption compared to a design using a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to electronically control many more processes [8].

1.3.1 VEX Microcontroller

The controller module contains two PIC18F8520 microprocessors and bristles with input/output ports. One microprocessor, the "Master Processor", manages signals from the RC receiver and generates PWM for motor and servo control, it sends precisely timed electrical signals required to drive the motors. The second processor, the "User Processor", is available for software development i.e. it can be programmed by users to control the robot. Sixteen ports are shared among analog and digital input/output functions, and eight more are dedicated to driving motors. They share a 10 MHz crystal oscillator and each processor can perform 10 million instructions per second. The communications role is particularly important, because the controller is normally hooked up to an FM radio receiver that takes commands from a human-operated six-channel remote control. The maximum operating range is about 35 to 40 meters.

The controller has jacks for serial port and 2 receiver ports so you can have 2 separate controllers (12 Channel R/C). Among other components is a LM294 5V Regulator and a smaller 2931A 8pin Soic (probably a 3.3V regulator). There are 5 LED’s (one glows through a red slit over the other 4 but it is not an IR port). A ULN2003A (Seven Darlington Drivers) chip is found near the Servo Motor outputs on the microcontroller [9].

The 16 Analog/Digital 3 pin ports read 0 = ground, 1 = +5v and 2 = input/output. Pin 0 is the one closest to the edge of the case and the other pins go inwards in the ascending order. The jumpers have 3 pins but the center pin is no contact. So they act as switches to ground the weak pull-up inputs.

The motor outputs appear to be: Pin 0 (outer edge pin) to the black wire = ground Pin, 1 (center) to the orange wire = 7 - 8volt from the battery. Pin 2 (closest inward) to the white wire = PWM Signal.

Figure 1.8 VEX Microcontroller

1.3.2 PIC 18F8520 Overview

The PIC is made using Harvard Architecture i.e. it has separate storage and signal pathways for instructions and data. It has advantage over the Von Neumann Architecture where the instruction and data are fetched from the same memory using the same data bus. This family offers the same advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of High Endurance Enhanced Flash Program Memory [10]. The PIC18FXX20 family also provides an enhanced range of program memory options and versatile analog features that make it ideal for complex, high-performance applications.

A comparison of the features of the PIC18FXX20 family is shown in the table below, in terms of the amount of memory, input and output pins, No. of CCP pins and its processing speed.

Table 1.1 Comparison of the features of the 18FXX20 family of Microcontrollers

Device	Program Memory		Data Memory		I/O	10-bit A/D (Ch)	CCP (PWM)	Timers 8-bit/16-bit	Ext Bus	Max Fosc (MHz)
	Bytes	# Single-Word Instructions	SRAM (Bytes)	EEPROM (Bytes)						
PIC18F6520	32K	16384	2048	1024	52	12	5	2/3	No	40
PIC18F6620	64K	32768	3840	1024	52	12	5	2/3	No	25
PIC18F6720	128K	65536	3840	1024	52	12	5	2/3	No	25
PIC18F8520	32K	16384	2048	1024	68	16	5	2/3	Yes	40
PIC18F8620	64K	32768	3840	1024	68	16	5	2/3	Yes	25
PIC18F8720	128K	65536	3840	1024	68	16	5	2/3	Yes	25

The PIC18F8520 has 32K bytes of flash program memory, it only has 2K bytes of RAM including the memory mapped I/0 which controls the PIC's on-chip devices. There is 1K byte of EEPROM data memory, and memory access at the instruction set level is via 256 word pages. It has data retention capability of over 40 years [10].

The dual microprocessor architecture drives what the programmer sees at the 'C' level. VEX programs are architected around a loop which polls the SPI to exchange RC and PWM data. However, this project would not be going deep in the capabilities of the dedicated processor, focusing instead, on the resources directly available on the User Processor to program it to perform the tasks. These include serial ports, PWM outputs, A/D ports, Interrupt Capabilities and Digital I/O.

1.4 Project Objectives

The major part of this project is the assembling of the VEX Design Kit into a suitable robot and the programming of its microcontroller to perform the required operations (behaviours) autonomously as outlined below.

- Firstly, the robot needs to move freely on the task platform i.e. it will explore the environment. This will be done with the help of several Bump Sensors and Limit switches.
- Secondly, the robot will be detecting objects (pucks) on the platform. This will be done using Line Tracking Sensors.
- Thirdly, when the robot has detected an object, it should stop and pick it up. Object handling will be achieved using a claw kit that will be mounted in front of the robot. The Claw Kit will be operated using a servomotor.
- After the robot has successfully picked up an object, it will be doing goal detection. The goal will be a light source, which will be detected using a light sensor.
- Once the robot has successfully detected the goal location, it will transport the object that it had picked up earlier to that location. The object will be dropped off at the goal location. An ultrasonic range finder will be used to detect the distance to the goal and when to drop off the object.
- Finally, obstacle avoidance is also a part of the project. When in the process of transporting the object to the goal location, the robot will be avoiding any other object that may fall in its path.

The robot will be made fully autonomous, i.e. it will perform the above operations with the program that will be loaded to the microcontroller, even though a radio control has been supplied with the kit. The variety of sensors will be used to study the behaviour of the robot in the described environment.

1.5 Report Outline

A general introduction to behaviour based robotics followed by a discussion about the various robotic kits available from VEX Robotics has been presented in this chapter. The overview of microcontrollers and that, which is used in this project, has also been given. The primary and secondary objectives of this project have also been clearly highlighted.

Chapter 2 gives a detailed discussion about behaviour- based robotics and its history. The influence of artificial intelligence has been further looked at. The different robot control approaches available, has also been looked at together with the behaviours to be performed by the robot in this project.

Chapter 3 looks at the mechanical design of the robot. The assembly of the robot and the components used in the structure has been discussed about. The mounting of object handling equipment has also been shown together with the modifications made in placing the sensors. The chapter concludes by presenting pictures of the final robot to be used for the project.

The electrical system of the robot is highlighted in chapter 4. The sensors used for the project have been looked at. This chapter further looks at the processing and system integration. The power distribution and the connection diagram of all the components with the VEX microcontroller has been shown.

The software system being one of the major parts of the project, is dedicated to chapter 5. A discussion of the software used for the programming of the robot can be found in this chapter. The control system and a thorough discussion of the program flowchart has been presented in this chapter. Finally the sensor subsystem and its interaction with the other subsystems in the robot, has been highlighted.

Chapter 6 will be giving an overview of the different types of robot fields and environments for robots of different applications. A detailed discussion of the different types of robots has been given together with the different types of platforms or environments they work in. Finally, the field (environment) used for the robot's operation in this project is shown.

The performance estimations of the modified VEX Robot for this project is discussed in chapter 7. Firstly the speeds of the robot has been shown for different combinations of gears and wheels from the VEX Kit. The performance of the sensors is then looked

at for the different applications they are used in. Finally, a discussion is presented on the accuracy of the arrival of the robot at the goal location.

Experimentations are carried out in Chapter 8, where the time in which the task was completed was measured. The results were taken for different number of objects. Objects were placed starting from 1 and going up to 5 and for each, the time for ten trials were taken. The positions of the robot and the objects were also noted.

Chapter 9 concludes by summarizing all the work done and presented in this project. Suggestions for future work in the development of the software and the structure has also been given.

Chapter 2

Behaviour- Based Robotics

2.1 Overview

For the last ten or fifteen years, research in robotics has been dominated by diverse applications of the behavior-based approach. This approach emphasizes the distributed, cooperative efforts of individual behaviors to cause the robot to act correctly within a specified environment. A behavior-based robot lacks central control, creating a need to manage which behaviors have control of the robot at a given time. This management is called arbitration and it is the key aspect of any behavior-based approach. Development of a behavior-based system is characterized by dividing the task up into simpler tasks, which when accomplished together accomplish the main task [1].

Historically, the field of robotics has drawn a dichotomy between deliberative and reactive control in robotics. Thus, to fully understand the present approach to robotics, one must understand it in the context of the history of the field as a whole. A brief history of robotics research together with the reactive versus deliberative control of mobile robots has been presented in this chapter.

2.2 Early Efforts in Robotics

Although behaviour-based robotics is a relatively new field as academic fields go, it is possible to find historical successful researchers. Surprisingly, early efforts in robotics reflect similar ideas to the behaviour-based approach, with the focus relying on simple reflexes to produce correct action within the environment. It is generally agreed that Grey Walter's Machina Speculatrix (1953), a small robot made from vacuum tubes, was the first behaviour-based robot. It was later implemented as Grey Walter's Tortoise. It had no high-level knowledge and could not translate its actions into symbolic meaning. This

robot explored its environment, and was capable of returning to a recharging station, designated by a light bulb, to recharge its batteries when they were low.

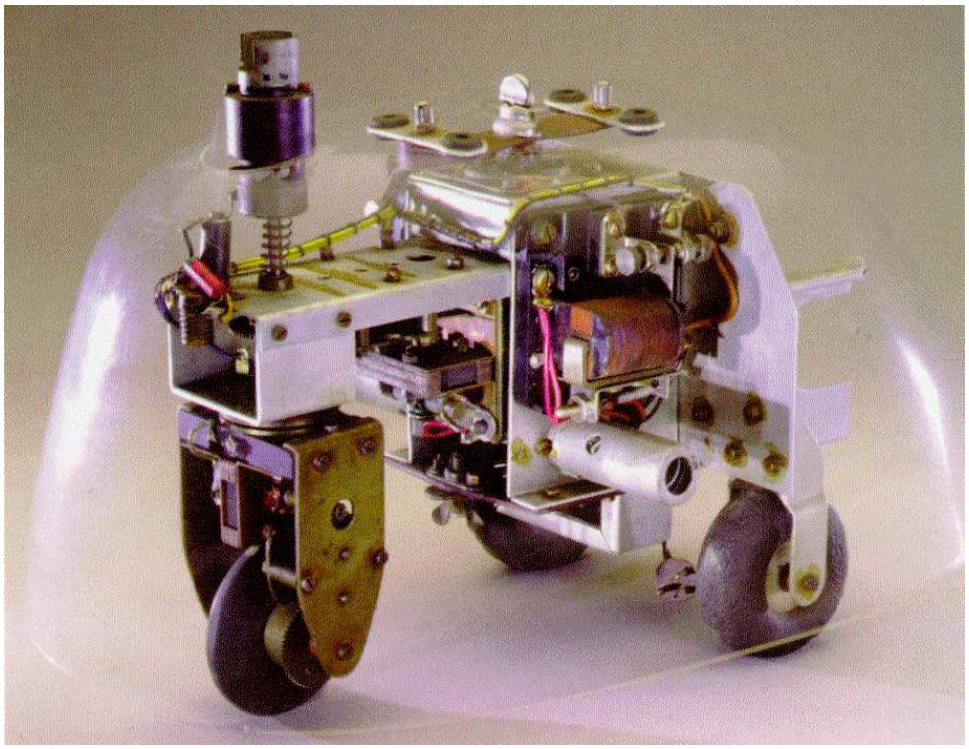


Figure 2.1 Grey Walter's Machina Speculatrix

Control of the Tortoise was divided into simple reactive behaviours that were organized by priority. These behaviours were, "head toward weak light," "back away from bright light," and "turn and push." The "head toward weak light" behavior caused the robot to move towards a low intensity light source. "Back away from bright light" was an aversive behaviour, which caused the robot to move away from a high intensity light source. Finally, the "Turn and push behaviour" was an object avoidance behaviour. Turn and push had priority over the other two behaviours [1].

The combination of these simple behaviours, along with an interesting aspect of the hardware used to construct the robot, causes the execution of the fairly complex behaviour of exploring and recharging. This is because when the battery is fully charged, the robot perceives the light bulb as a high intensity source of light, causing the tortoise

to be repelled from it and move outwards to explore the environment. However, as time passes and the battery power becomes low, the light bulb appears to the robot as a low intensity light source, causing the tortoise to be attracted to it. These characteristics create an oscillating behavior between light attraction and light repulsion based on the status of the battery. That is, complex behavior produced through simple reactions.

2.3 Influence of Artificial Intelligence

Artificial intelligence is the study of ideas to bring into being machines that respond to stimulation consistent with traditional responses from humans, given the human capacity for consideration, judgment and intention. Each such machine should engage in critical appraisal and selection of the different options available within itself. Produced by human skill and labor, these machines should conduct themselves in agreement with life, spirit and sensitivity, though in reality, they are imitations [11].

The serious study of Robotic fields and Artificial Intelligence first began, following the invention of modern computers during World War II [12]. Thus, approaches to robotics tended to coincide with approaches to artificial intelligence. In particular, methods to produce intelligence in robots were considered to be merely extensions of the methods used to produce intelligence in computers [1].

The early emphasis in artificial intelligence on symbolic representation resulted in most early robotics attempts being modeled around representing the robot's environment through a world model. This approach can be categorized as the deliberative approach. That is, robots perceive the environment using sensors; produce a symbolic representation of those perceptions in the form of a world model, and then that representation to reason about what actions to take next.

A good example of the deliberative approach is the Stanford Research Institute's Shakey robot (Nilsson 1969). One of the first mobile robots, Shakey used the STRIPS planning

system (Fikes and Nilsson 1971) to develop a model of the world, deliberate on which actions to take, and then create a plan to execute that action. It was early experimental efforts like this that led to the advent of the Hierarchical Architecture, the most prominent control architecture within the deliberative approach [1].

2.4 Robot Control Approaches

Robot control is the means by which the sensing and action of a robot are coordinated. The infinitely many possible robot control programs all fall along a well-defined control spectrum. Some of the common approaches of controlling mobile robots have been discussed below.

2.4.1 Deliberative (Planner-based) Control

This type of control requires a lot of correct information about the robot performance and as well as the environment in which it needs to operate. It involves a lot of planning i.e. looking into the future. This means that there may be quite a lot of tasks, which the robot may have to give prior consideration to. A robot needs to have some sort of thinking ability to differentiate between the presented tasks and accomplish them according to their significance.

The planner- based control relies heavily on symbolic representations and real life models. Usually they are hierarchical in structure, which are typically rigid. A robot controlled by this structure should be capable of learning and prediction. This type of control can be regarded as too slow for real- time response because its performance speed decreases with complexity. It can be used for communication and control in predetermined and predictable ways based on internal models.

2.4.1.1 Hierarchical Structure

The Hierarchical Architecture is the traditional architecture for robotics. It was the most popular approach in the 1980s, to the extent that the U.S. government developed a

standard robot architecture, called NASREM, which reflected this model. The hierarchical model is particularly well suited to static environments, which has made it the primary approach in areas of robotics with this constraint, such as manufacturing [1].

The hierarchical model consists of four components: sensory processing, world modeling, task decomposition, and value judgment. Each of these components is divided into a hierarchy, with higher levels being more abstract than the lower levels. Thus, the lowest level of the hierarchy consists of the basic control of the robots sensors and actuators, while the highest level has an abstract idea of the task that needs to be accomplished. The intermediary layers help convert this high level plan into primitive controls for the actual robot [2].

The four main components work in a linear fashion. First, the robot obtains raw data through its sensors; then it proceeds to construct a world model based on this raw data; finally the robot decides what actions to perform based purely on the world model.
Thus, this approach does not allow a direct connection between sensation and action. Instead this approach places great importance on the accuracy of the world model [3].

2.4.2 Reactive Control

Reactive control is a system for tightly coupling perception and action, typically in the context of motor behaviors, to produce a timely robotic response in dynamic and unstructured worlds. It is different from deliberative control in the sense that no world models, persisting state, history, or look ahead/ search/planning are used. These systems are collections of reactive rules and can be quite powerful. Robots using this type of control don't usually think; they act according to what instructions have been fed to the robot. Although the system has limited flexibility for increasing complexity, it is still fast, regardless of task complexity.

This control is highly effective for dynamic domains where fast reaction can sometimes be difficult to work with. The system can be learned where most of reinforcement

learning is aimed at learning reactive policies. When designed, the system takes enumeration of relevant situations and conditions. This control unlike the deliberative control does not look into the past or the future. It is an excellent and ever-present substitute for both hybrid and behavior-based systems [13].

2.4.3 Hybrid Control

Hybrid Control is a combination of reactive and deliberative control. This means that a robot with this control can think and act separately & concurrently. They are usually called “three-layer systems” where the major challenge is the middle layer, which must coordinate the other two, and which operate on very different time-scales and representations (signals vs. symbols). This control is currently one of the two dominant control strategies in robotics because it can take the best of the reactive and deliberative control properties. But the flaw is that it may also suffer from the worst of both.

Designing the middle layer of this three- layered system can be extremely difficult. The layer coordination is an important question, just like behavior coordination in Behaviour-Based Control. This is the reason why it is not best suited for all problems and domains (e.g., multi-robot control).

2.4.4 Behavior-Based Control

Behaviour- based control systems are networks of behaviors using uniform representation and similar time-scale. Robots using this type of control respond in real-time i.e. they are reactive but not stateless i.e. not purely reactive. This control utilizes distributed representations allowing for a variety of behavior coordination mechanisms.

This control system is an alternative to hybrid control. It has the same articulateness properties as hybrid control. Although it historically grew out of reactive systems, it is not constrained. It can contain reactive components, just like hybrid systems. The key difference is in the “deliberative” component.

2.4.4.1 Hybrid vs. Behavior-based Control

Hybrid and Deliberative planners rely heavily on world models and can readily integrate world knowledge. They have broader perception and scope since a lot of information processing and storage needs to be done about the environment.

Behavior-based systems can afford modular development with real-time robust performance in dynamic world. The system is tightly coupled with incoming sensory data and can provide for incremental growth.

2.4.4.2 What are Behaviours?

Behaviors are processes, dynamical systems and building blocks for control, representation, and learning in Behaviour- Based Control. They are observable, time-extended robot-environment interactions, coupling, sensing & action. They follow a set control laws/processes that exploit system dynamics to achieve specific goals. They can take inputs from sensors or other behaviours and send outputs to effectors or other behaviours [14].

2.4.4.3 Behaviour coordination

Behaviour coordination is the general action selection problem with two options. One is arbitration and the other is fusion. Arbitration means selecting among a set of behaviours whereas fusion is the combining of several behaviours. Arbitration is simpler and much more prevalent and also lends itself to learning mechanisms. Various control architectures use a mixture of the two options at different levels [14].

2.5 Behaviours of the Robot in this Project

As any behaviour based robot has a collection of various smaller behaviours (operations), which add up to the final behaviour of the robot, the robot in this project is also controlled by some of the operations as, outlined below.

2.5.1 Wander Around (Explore)

When the robot first starts to move, it will be freely going straight unless there is some signal received from any of the sensors. The robot will be changing directions whenever it will bump into any of the walls, i.e. whenever any of the bump sensors or the limit switches will be pressed against the wall, the program is designed to guide the robot away from the wall. The robot will keep doing this until it has detected an object.

2.5.2 Object Detection

While the robot is on free run the line tracking sensors will be active and sensing for any object that may come across its path. There are three line tracking sensors mounted on the robot. One will be mounted in the centre of the robot ahead of the front wheels and the other two sensors will be placed just in front of the right and left front wheels. All the line tracking sensors will be detecting the objects but the sensor in the centre is the one that will be used to help the robot pick up the object. When the object is detected by either of the right or the left line tracking sensor, the robot is supposed to turn a little so that the centre sensor can detect the object and then the program would send the command to pick up the object.

2.5.3 Object Handling and Transportation

Once the sensor detects an object, the robot is supposed to stop and pick it up. Since the sensor is a little behind the position of the claw, the robot will be backing up a little so that the claw can properly hold the object. Instead of just grabbing and pushing the object as stated in the objective, the claw would now be picking the object and then transporting it to a goal location. A servo motor will be used to open and close the claw and another motor will be raising the claw with the help of a wrist kit.

2.5.4 Goal Detection

The robot would need to transport the object to a goal location. The goal location will be a light source that will be coming from a 240V AC Fluorescent Lamp. The lamp will be focused on the robot field. A Light Sensor will be used to detect the light source. The

robot will be rotating when trying to detect the light source. When the light source has been detected, the robot will stop rotating and start moving towards the light source. Since the lamp will be mounted at the walls, the robot would need to stop in front of it and place the object. To detect the wall, an Ultrasonic Range Finder is used. This will measure the distance between the robot and the wall and will stop the robot from bumping the wall.

2.5.5 Obstacle Avoidance

When the robot is trying to transport the object to the goal location, it should be avoiding any other object that would be coming in front of the wheels. The left and right Line Tracking Sensors would do the sensing of these objects. When an object is detected in front of the wheels, the robot will backup and follow a different path to the goal location.

Chapter 3

Actuation and Mechanics

3.1 Chassis

The Chassis Design for the robot used in this project is based on the Squarebot design. It has been designed and built to satisfy the following criteria:

- Predictable and easily measurable turning mechanism.
- Allowing good integration to other components and acquiescent to future changes in design.
- Easy to repair in case of damage.

This design criteria had been taken into account by VEX when designing the Robotic kits. The chassis is made using the 4 chassis rails and the 2 chassis bumpers supplied in the kit with the angles bolted together with screws and keps nuts. The overall dimensions of the robot are contained within a 20cm x 20cm x 20cm box.

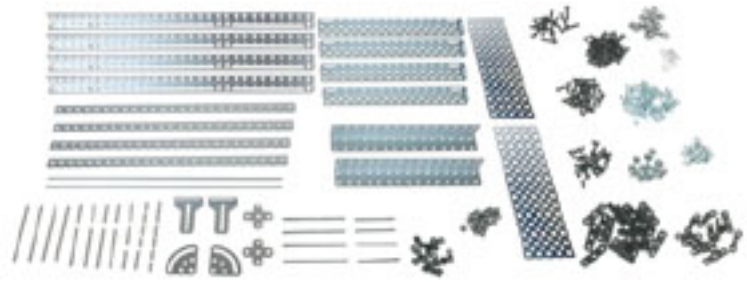


Figure 3.1 The Mechanical Components of the VEX Design Kit

The chassis is a 3 level design, where the lowest level is for the wheels that facilitate the contact between the robot and the work plane. The motors and the gears for the wheels are placed within the chassis. The next stratum is where the microcontroller is placed. The third level is where the battery pack and the radio receiver are mounted. Most of the other components are on top of the chassis. The chassis has evenly distributed holes on it for easy mounting of the components.

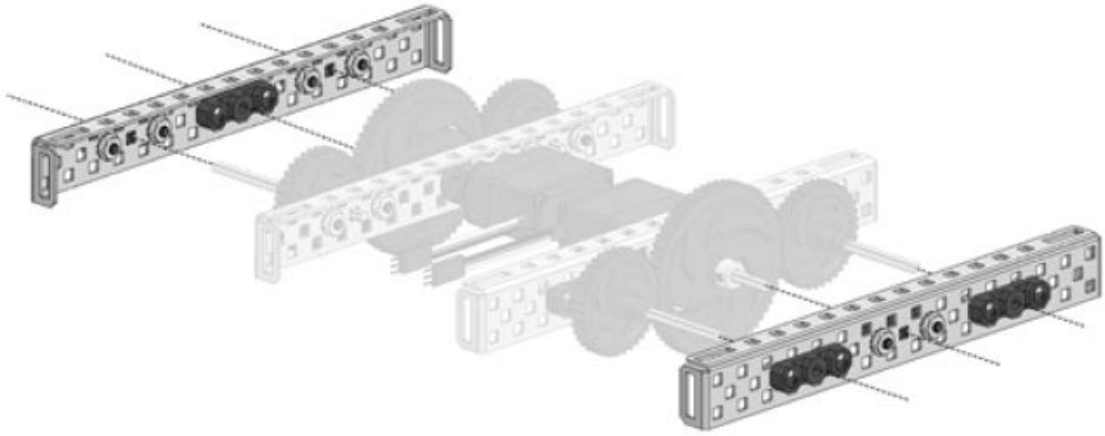


Figure 3.2 The Base Assembly of the Robot

3.2 Locomotion

3.2.1 Gears, Speed, and Strength

It can be observed that the Squarebot is reasonably fast and can also navigate in fairly uneven environment. The robot can be made faster and stronger but not both at the same time. Taking a look at the gear train on the Squarebot; the motor is connected to a 60-tooth gear, while the wheel axles are connected to 36-tooth gears. Each time the motor makes one complete rotation; it moves the large gear through 60 teeth, which in turn, also moves the smaller gear through 60 teeth. However, 60 teeth on the 36-tooth gear represent nearly 2 rotations (1.67 rotations, to be precise), which means that the wheels have a faster turning speed, or angular velocity, than the motor. This process of powering a larger gear to make a smaller gear go faster is called gearing up.

Figure 3.3 The Gear Kit for VEX Robot

While increasing speed can be a good thing, the swap is a reduction in strength, or torque, and at times, it is desirable to reduce speed and increase torque by gearing down as in the application of a tank. This is accomplished by powering a small gear and using it to drive a larger gear.

The linear velocity can also be altered by changing the wheel size. When the motors have a fixed number of rotations per minute, using wheels with a larger circumference will enable the robot to cover more distance with each rotation.

Figure 3.4 The Wheel Kit for VEX Robot

Calculating the gear ratio (drive gear: driven gear) can give a precise calculation of how much speed (or torque) is gained. In the case of the Squarebot, the ratio is 60:36, that is 5:3. A ratio of 5:1 is obtained by using the 60-tooth gear to drive the 12- tooth gear, and even higher ratios can be obtained by using more than 2 gears in a gear train. One concern is that some torque is always lost to friction with each conversion; so minimizing the number of gears in a gear train is generally desirable.

3.2.2 Driving Base Considerations

The driving base is the basis of all other aspects of Robot Design. One needs to find the right balance of speed, strength, and maneuverability to accomplish the desired task. Obviously, the optimal base needed for driving at maximum speed on a smooth, flat surface will differ from one needed to climb up uneven terrain. As described above, the balance between speed and strength is accomplished by adjusting the gear ratios between the input (motors) and the output (wheels).

Maneuverability can be defined as the speed and precision with which the robot can change direction. While a high degree of maneuverability can be desirable, it can actually

be a disadvantage at times, as a robot which changes direction too easily can be difficult to control.

3.2.3 Four-Wheel Drive

A good type of drive train is the four-wheel-drive system; the Squarebot is an example of this. A four-wheel-drive robot has more traction and control than a two-wheel-drive robot. The trade-off is the increased wheel base i.e. the distance from front to back wheels, which can cause turning problems, especially when the wheel base exceeds the wheel width i.e. the distance from left-side to right-side wheels.

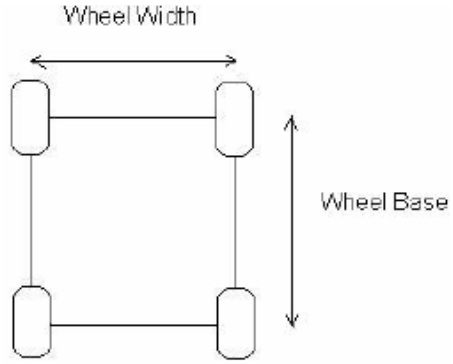


Figure 3.5 Schematic showing Wheel Width and Wheel Base

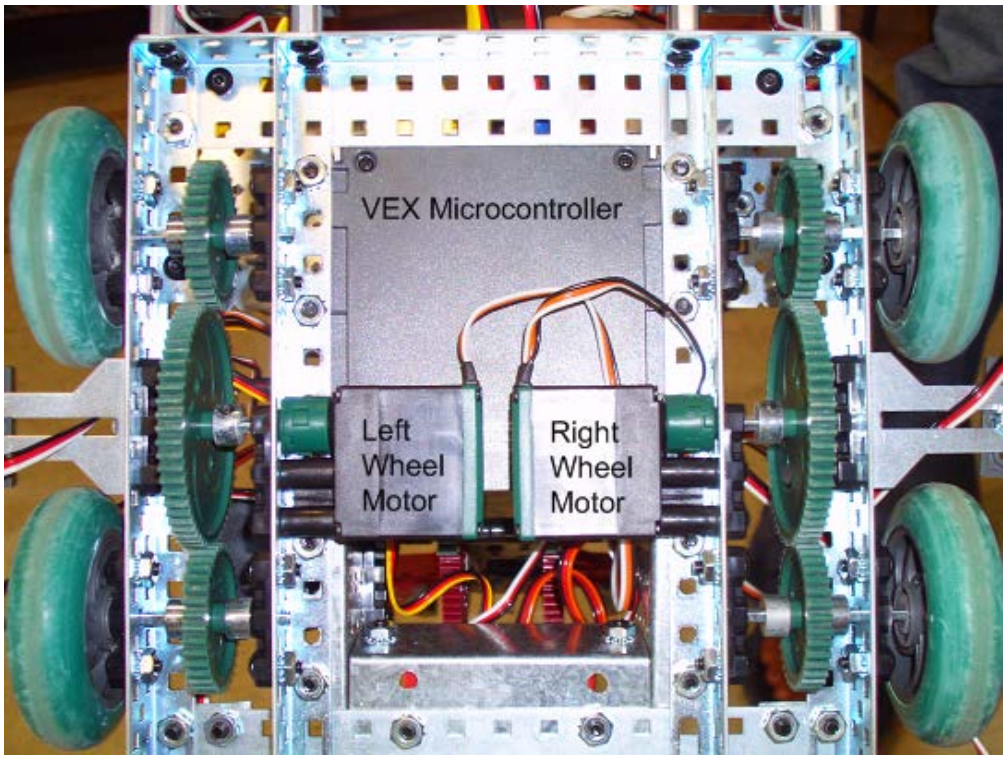


Figure 3.6 Bottom view of the four-wheel-drive system of the Robot.

3.2.4 Drive Train

The drive train for the robot consists of two Vex Continuous Rotation motors with the following specifications as given in Table 1.

Table 3.1 Specifications of Motors used for the robot.

Motor	Voltage		No Load		Stall	
	Operating Range	Nominal	Speed (RPM)	Current (A)	Torque	Current (A)
Continuous Rotation	4.4-15V	7.5V	100	0.14	6.5in*lbs	1.6

3.3 Claw Kit

The claw kit from VEX Robotics has also been fixed to the robot to help it easily grab and manipulate objects. The claw is capable of picking up objects of size of a can, or grasping objects as delicate as a feather, without much difficulty. This claw is compatible with all VEX robots and mechanical parts.

Figure 3.7 VEXplorer Claw Kit

A motor is required to operate the claw kit. Either the servo motor or the VEX continuous rotation motor can be used to for the opening and closing of the claws.

The dimensions of the claw are: Length 20.32 cm x Width 7.94 cm x Height 3.81cm. with a weight of 0.4 lbs. Hence this does not make much difference to the operation of the robot as a whole. The claw can open a maximum of 8.57 cm with the assistance of a gear mechanism powered by the motor.

3.4 Wrist Kit

An increased control over object manipulation can be achieved by adding the VEXplorer Wrist Kit to a robot. By integrating the wrist kit with the Vex claw kit, the robot can gain significant dexterity.

Figure 3.8 VEXplorer Wrist Kit

The wrist kit is also operated using a motor. The motor that is supplied with the wrist kit is not compatible with the VEX microcontroller being used in this project. This is because the kit was originally designed for the VEXplorer robot. But the good thing is that other motors can be used with the kit, so a motor was ordered for this kit separately.

Instead of pushing the objects to the goal location, it was decided to improve the design further by having the robot to lift and then transport the objects. This is where the wrist kit became handy. But to have the arms move up and down, it was needed to be mounted at a higher position. Hence to achieve this, brackets were designed to mount the wrist kit to which the claw was attached. This idea was extracted from the VEXplorer Robot Design. Together with these brackets, an arm was also made which would be joining the claw kit to the wrist and hence increasing the length. The length had to be increased so that the claws would go to the right position in front of the robot when lowered to pick up the objects. The brackets were made using thin sheet metal to minimize weight.

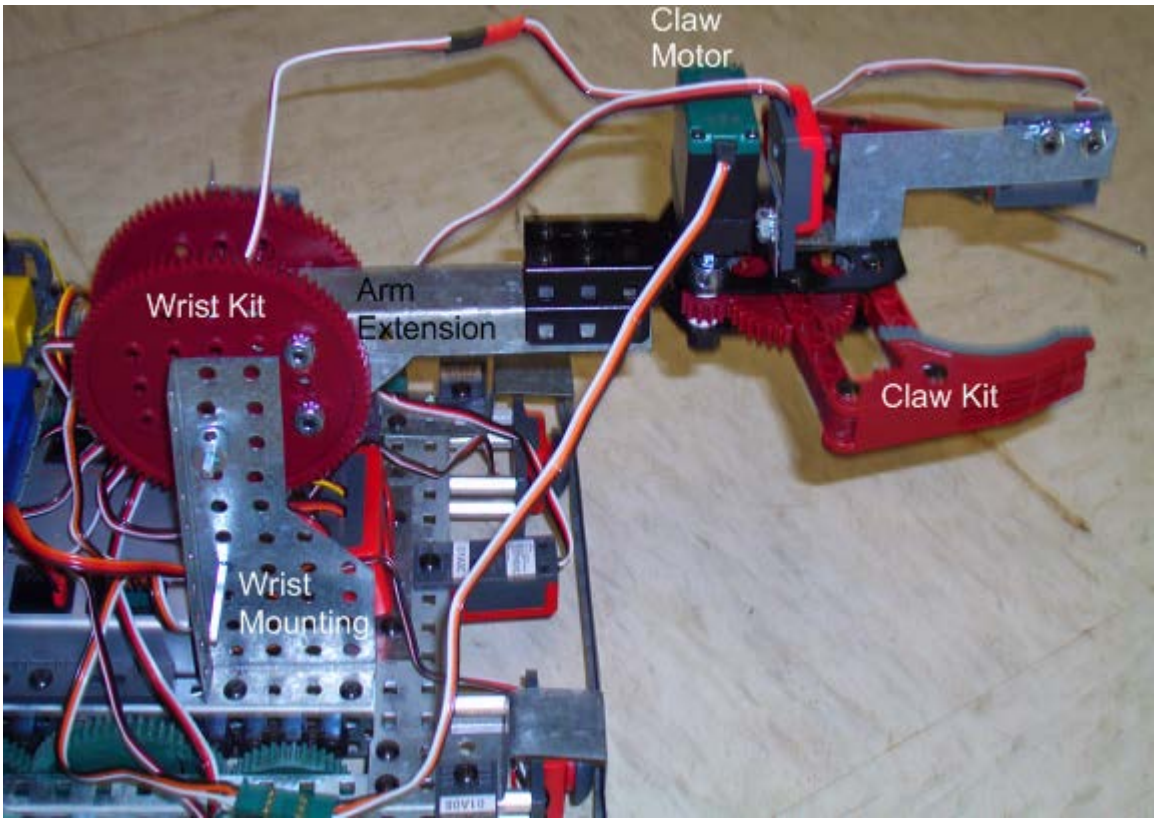


Figure 3.9 Final claw Mounting

3.5 Sensor Mountings

Several mountings had to be made so that the different sensors could be mounted in the right positions to perform their functions without any hindrance.

3.5.1 Light Sensor Mounting

The light sensor needed to be mounted on top of the claw so that it could easily sense the light source. Mounting it on top of the claws got it to the same height as the light source to be placed at the walls. Again a small mounting plate was made for the sensor.

3.5.2 Claw Limiter Mounting

A limiter switch was mounted on top of the claw to detect if the object has been gripped by the claw. The height had to be adjusted several times by making changes to the

mounting plate so that the switch was in the right position to be pressed when an object was detected.

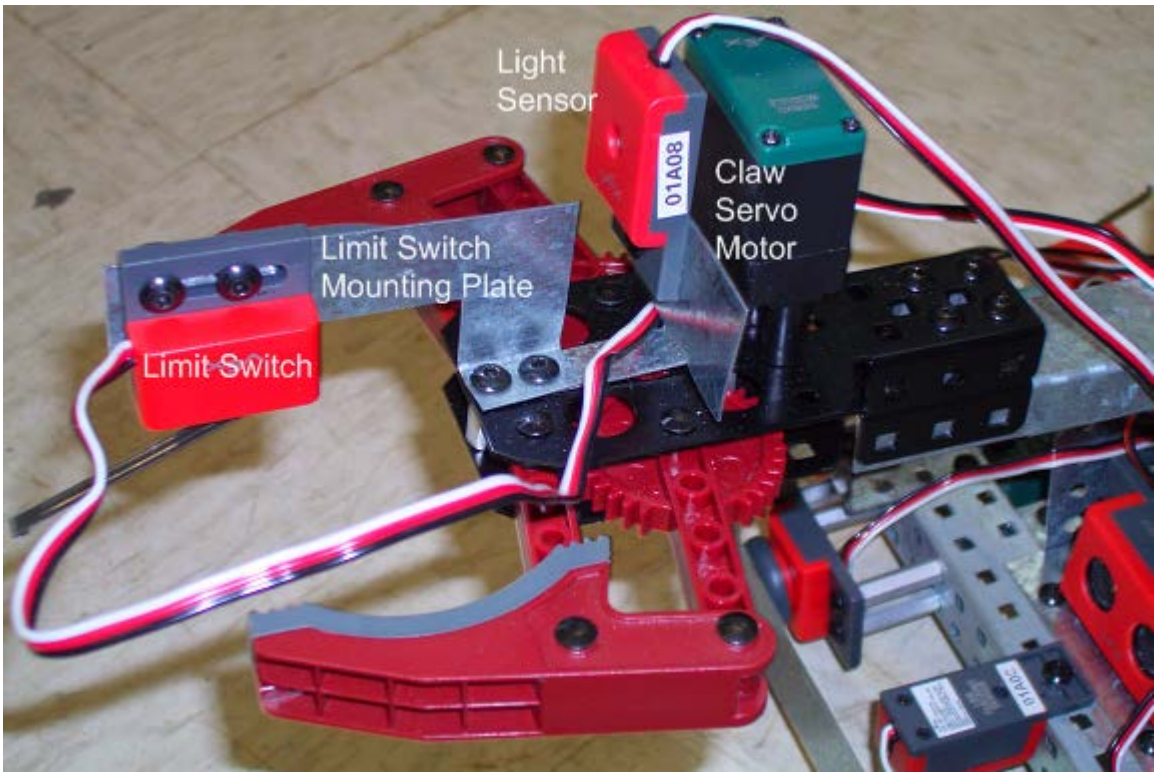


Figure 3.10 Light Sensor and Claw Limiter Mounting

3.5.3 Ultrasonic sensor mounting

The ultrasonic range finder was to be mounted in front of the robot where it would get a clear view. This was because the sensor would be sending high frequency sound waves which should reflect only when they come in contact with the walls. So to mount this sensor, a small L-shaped bracket was made from sheet metal.

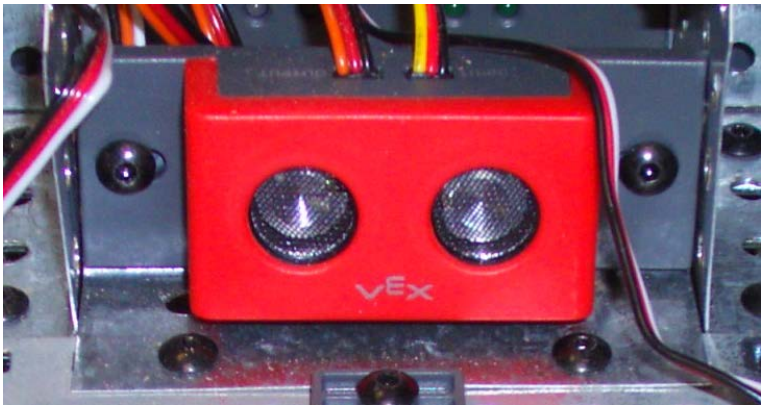


Figure 3.11 Ultrasonic Sensor Mounting

3.5.4 Front Bumper Panel

A bumper panel was designed to be placed on the front of the robot's structure so that it also covered the area in front of the wheels. This was done after some testings were done and it was observed that when the robot hit the walls of the platform at a certain angle, the line tracking sensors on the sides hit the walls and the robot got stuck causing hindrance in further operations. This bumper panel was designed to be flexible enough, so that it would bend when the robot hit the walls and while bending, it would press the Bumper sensors on the front.

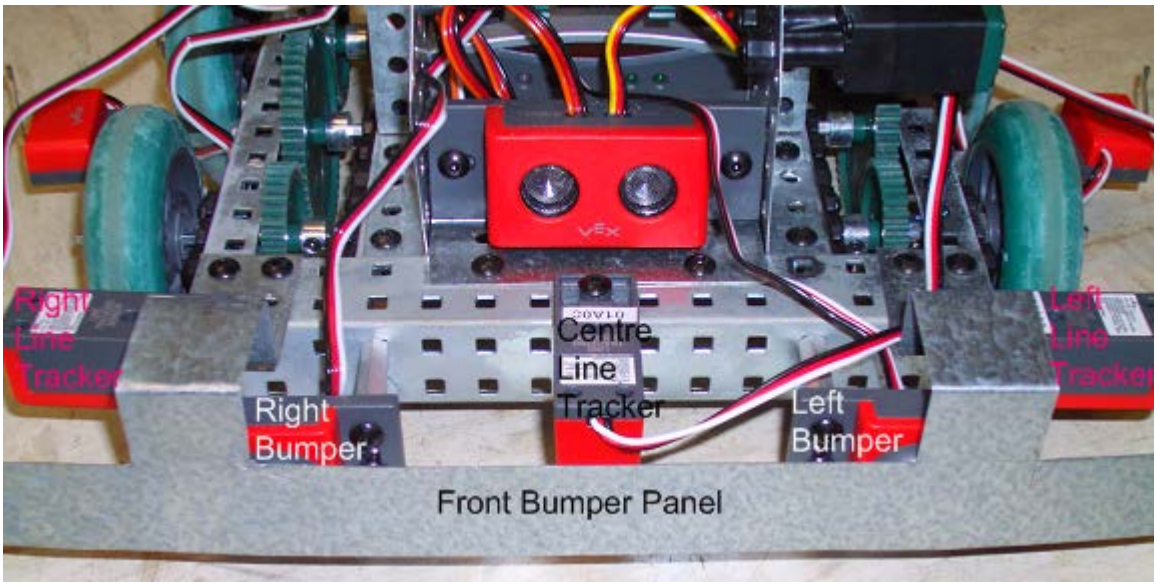


Figure 3.12 Front Bumper Panel and Sensor Positions

3.6 Final Robot Design

The final look of the robot is a combination of the Squarebot and the VEXplorer. The structure and mobility design is from Squarebot while the claw design is from VEXplorer. The picture below shows the final design of the robot used in the project.

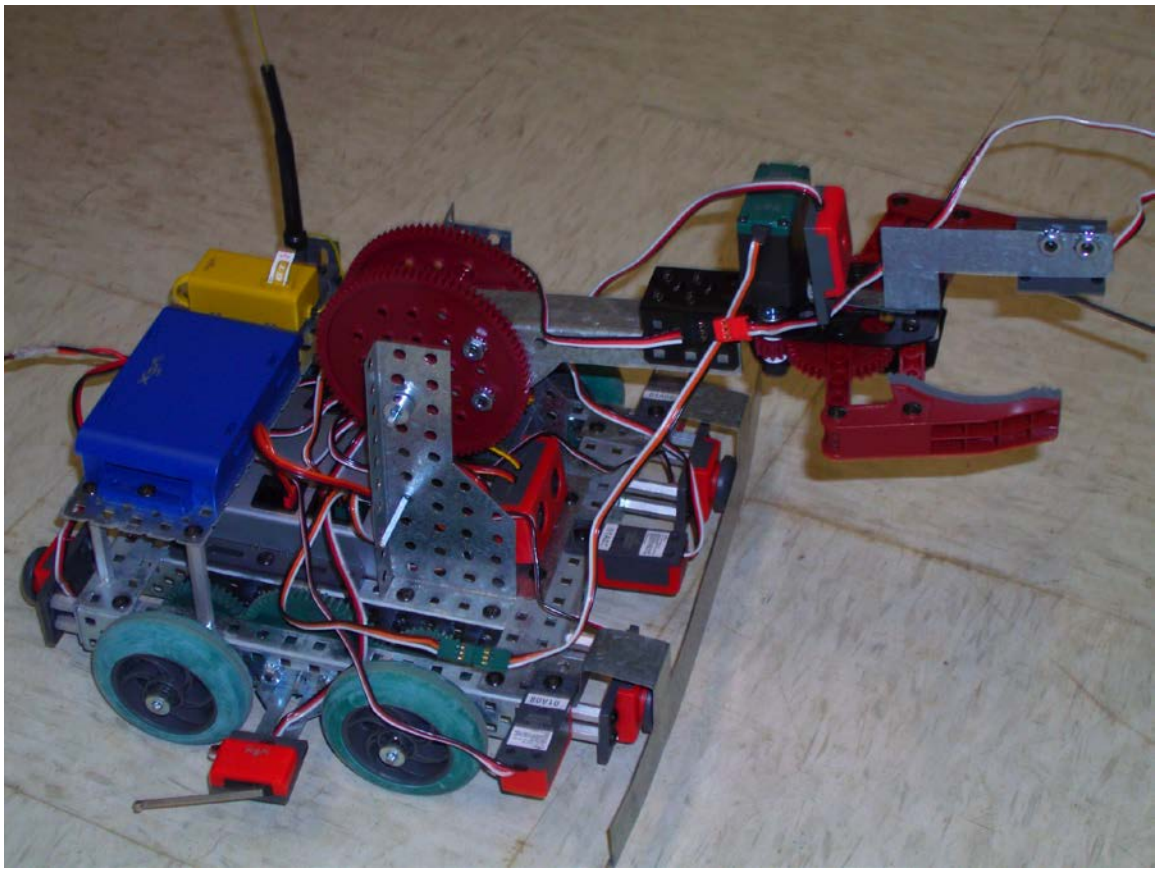


Figure 3.13 Final Robot Design

Chapter 4

Sensing and Electronics

The electrical system has been a phased development of the designing, implementing and testing. Paramount consideration has been given to the idea of reliability on the electrical system.

4.1 Sensors

The robot used in this project has been equipped with a variety of sensors to help it execute the behaviours to interact with its surrounding. These digital and analog sensors have been very helpful in meeting the objectives of this project. All the sensors used, have been manufactured by VEX.

4.1.1 Bumper Switches

This sensor is used to control the robot on the platform when it bumps into walls. The bumper switch gives digital input to the VEX controller: 0 when pressed and 1 when released. It enables the robot to perform certain functions with programming when the bumper switch is activated by contact. Four of these sensors have been used on the robot (two on the front panel and two on the rear panel). The robot changes direction whenever it bumps into walls, whether it is in the front or at the back.

Figure 4.1 Bumper Switches

4.1.2 Limit Switches

This switch can be used in more sensitive applications. This is also a digital sensor giving the same inputs as the Bumper Switch. It can be used to detect whether a gripper has

grasped an object or for the same applications as the Bumper Switch. Two of these switches have been used on the robot (one on each side). They help the robot to steer away from the walls whenever the sides rub against the walls.

Figure 4.2 Limit Switches

4.1.3 Light Sensor

The light sensor is an analog sensor, which uses a Cadmium Sulfoselenide Photoconductive photocell that allows the robot to detect and react to light. The robot can be programmed to follow the beam of a flashlight, or using it to help the robot to avoid getting stuck, by making it steer away from shadows. A whole new range of capabilities can be added to the robot. It can also be given a color vision by putting colored filters. The sensor has been vertically placed on the robot so that it is facing the front of the robot. This has been done because the light source is located on one of the walls and hence it would be easy for the sensor to detect the light source.

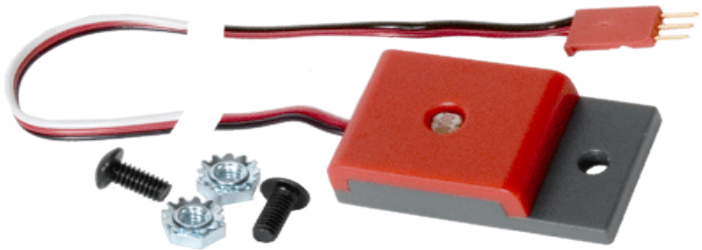


Figure 4.3 Light Sensor

Being an analog sensor, its output covers a range of values from zero to five volts rather than being only high (5 volts) or low (0 volts), as in the case of a digital sensor. This range of outputs from 0V to 5V is sent to the microcontroller, which reads it as a range of integer values from 0 to 255. For this sensor, a lower value (around 0) corresponds to very bright light and a high value (around 255) corresponds to darkness. The light sensor has a usable range of 0 to 6 feet, meaning it can distinguish a light source from ambient

light up to six feet away. A light source more than 6 feet away may blend into the ambient light and be lost. The output range is dependent on the intensity of the light source as well as the intensity of the ambient light in the environment.

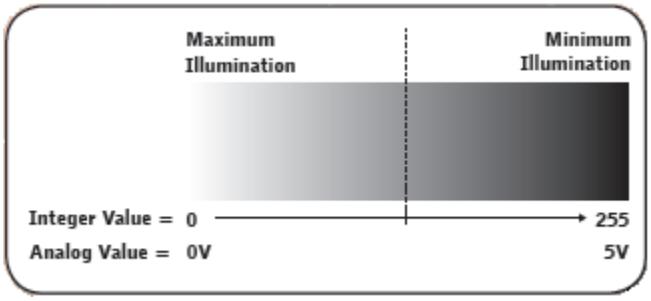


Figure 4.4 Respective Integer values for Varying Illumination

The sensor is sensitive to visible light only; it will not provide useful data for infrared or ultraviolet light sources.

4.1.3.1 Light Dependent Resistor (LDR)

Everything has an electrical resistance, some more than others. The light dependent resistor (LDR) will have a resistance that varies according to the amount of visible light that falls on it i.e. the resistance decreases in the presence of more light. The close up of an LDR is shown below:

The light falling on the brown zigzag lines on the sensor causes the resistance of the device to fall. This is known as a negative co-efficient. There are some LDR's that work in the opposite way i.e. their resistance increases with light (called positive co-efficient).

Figure 4.5 LDR Sensor

LDR's are made of semiconductors as light sensitive materials, on an isolating base. The most common semiconductors used in this system are cadmium sulphide, lead sulphide, germanium, silicon and gallium arsenide [15].

In order to use this device in a simple circuit, it is required to put a voltage across it and measure the current flowing through it. However, measuring current can be a little

tricky. So another resistor is placed in series, and the voltage measured across the LDR. This makes it a potential divider, and the voltage across the LDR is proportional to the current [16]. The diagrams below show the concept.

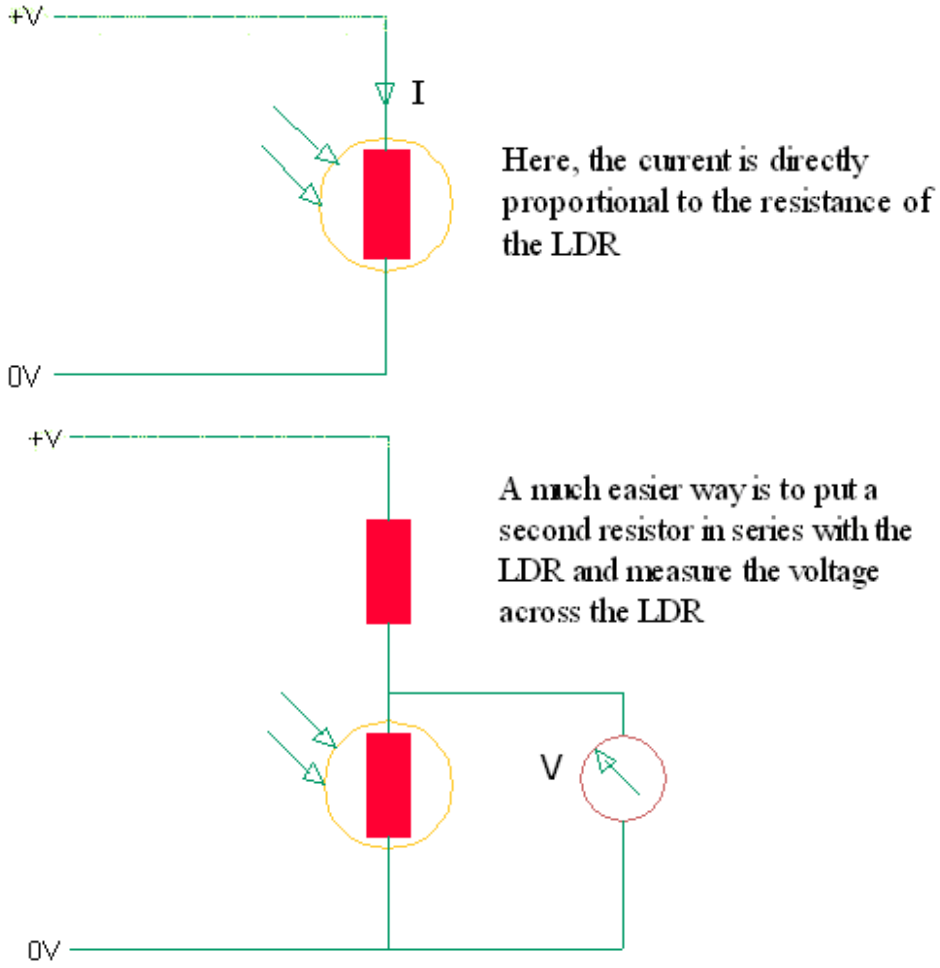


Figure 4.6 Circuit Diagrams for the use of LDR Sensors

4.1.4 Line Tracking Sensor

A line follower consists of an infrared light sensor and an infrared LED. It works by illuminating a surface with infrared light. The sensor then picks up the reflected infrared radiation and, based on its intensity, determines the reflectivity of the surface being used. Light-colored surfaces will reflect more light than dark surfaces, resulting in their appearing brighter to the sensor. This allows the sensor to detect a dark line on a pale surface, or a pale line on a dark surface or any other object with its color contrasting to that of the working plane.

The line follower can be used to help a robot navigate along a marked path, or in any other application involving discriminating the boundary between two high-contrast surfaces. A typical application uses three line follower sensors, such that the middle sensor is over the line the robot is following. All the three sensors from the kit have been used on the robot (one in front of the right and left front wheels and one in the centre). The major purpose of the sensors on the sides is to avoid any object that may come under the wheels and the sensor in the front centre is for object detection.

Figure 4.7 Line Tracking Sensors

Like the light sensor, this is also an analog sensor; its output covers a range of values from zero to five volts rather than being only high (5 volts) or low (0 volts), as in the case of a digital sensor. This range of outputs from 0V to 5V is sent to the microcontroller, which reads it as a range of integer values from 0 to 255. For this sensor, a lower value (around 0) is detected when the infrared light bounces back to the detector. In other words, when the surface is pale or highly reflective. A high value (around 255) is detected when the light is absorbed and does not bounce back. The optimal range for the line follower is approximately 0.05cm to 2.54cm.

4.1.4.1 How does the Line Tracing Sensor Work

The principles are very simple since the sensor consists of just two components. The first is an Infra-Red (IR) transmitter (usually an LED), while the second is an Infra-Red receiver (usually a transistor). IR is transmitted out of the sensor unit. If the IR is reflected back, it is picked up by the IR receiver transistor.

IR is basically classified as heat just like the heat from the sun is predominantly in the IR part of the spectrum. Black is known to absorb heat and when black can absorb heat, it can also absorb IR. Hence this is the principle of how it works. While the sensor is over

a black line, no IR is reflected back to the receiver. If the sensor strays away from the line, then IR is reflected back. For best results the black line is placed on a white background or vice versa, which will give the extreme two cases - white reflects IR [16].

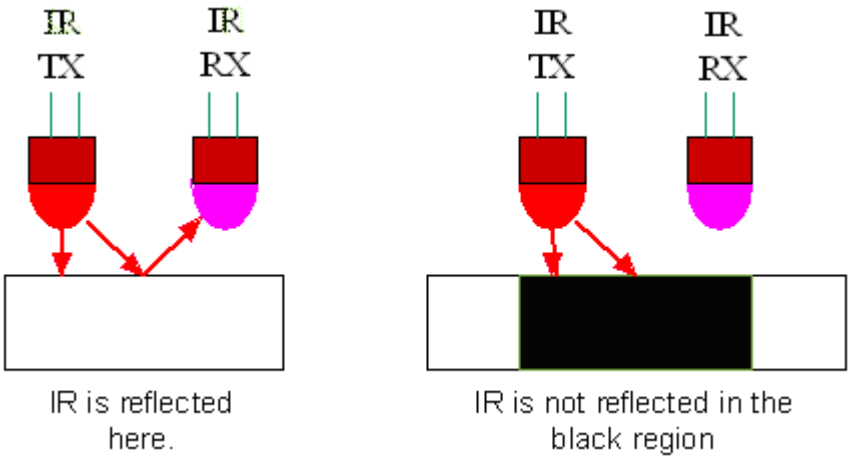


Figure 4.8 IR reflection on Different Surfaces

4.1.5 Ultrasonic Range Finder

This sensor is used to detect obstacles from 3 centimeters to 3 meters. It sensor works on the principle of sound- echo location. It sends out a high frequency sound wave and measures the time taken for it to return and the finds the distance of the obstacle. The sensor is mounted on the front beam of the robot to give a clear area for the waves to be sent and received. The purpose of this sensor in the project is to detect the wall when transporting an object.

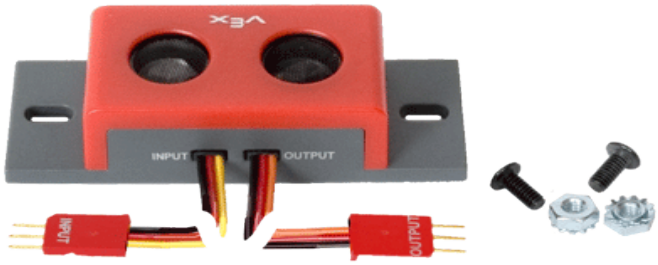


Figure 4.9 VEX Ultrasonic Sensor

The distance to the object can be calculated using the following formula:

Distance to object = ½ (speed of sound) X (round trip delay)

The speed of sound varies with altitude and temperature. At sea level and room temperature, it's approximately 344.2m/s. This increases with temperature and decreases with altitude. Therefore;

Distance in meters = 172.1 m/s X (round trip delay)

4.1.5.1 How Ultrasonic Sensors work

These sensors emit a very high frequency sound. In fact it is so high, that it can not be heard by the human ear. A picture of two ultrasonic sensors is shown below:

Figure 4.10 Ultrasonic Transmitter and Reciever

Two sensors work in union, one as the transmitter and one as the receiver. The transmitter typically sends out a constant beam of sound at a frequency of 40 KHz (note that the human hearing barely goes above 17 KHz). The receiver detects any sound coming in and gives a voltage out. Therefore, to detect an object, the transmitter sends out a signal. If there isn't an object in front of it, then the sound wave will carry on but there is a limit to the distance. Only if there is an object in the way, the sound waves will bounce back along the same path, and hence will be picked up by the receiver [16].

Therefore, if the time between transmitting the sound, and receiving it, can be noted, the distance of the object from our sensors can be worked out.

4.2 Processing and System Integration

In order for the VEX Robot (microcontroller) to have communication with the computer, some computer settings need to be done. Firstly, after software installation, the USB – to

– Serial Cable needs to be connected with the USB end to the computer. Then from the control panel, it needs to be identified to which COM Port it has been connected to. Once this is known, the COM Port settings need to be changed in the software. This settings need to be changed in the Terminal Window and the Loader Settings. Unless these changes are made no program can be downloaded to the microcontroller. The diagram below shows the windows where the settings need to be changed for the robot to communicate with the computer.

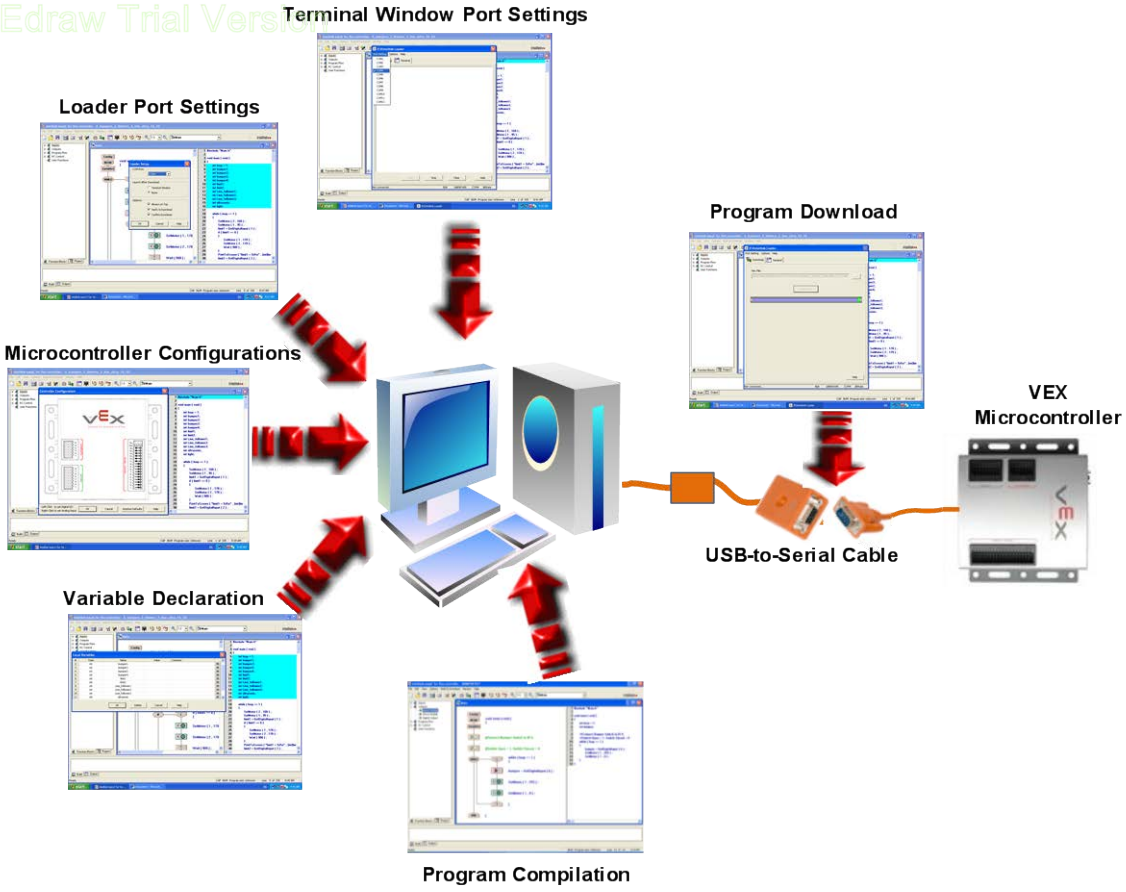


Figure 4.11 Communication of the VEX Microcontroller with the Computer

4.3 Power

Originally the VEX Robot should be powered up by 7.2V Rechargeable Nickel Cadmium (NiCd) battery supply connected to the Vex Microcontroller module. The controller & receiver draw minimum current of 62mA. The continuous rotation motors use 5mA to 2A

per motor depending on the speed at which they operate while the servomotor draws 1.5A. Both the motors draw the current from the Vex Microcontroller. The sensors, like all other electronic components, require power in order to function. Vex sensors draw power indirectly from the Power Subsystem through the Vex Micro Controller (Logic Subsystem).

Due to the unavailability of the batteries, a DC Power Supply was used to supply power to the robot using a 3m cable to avoid any obstruction in the mobility of the robot.
The use of the DC Power Supply to power up the robot helped it to perform efficiently and the components were able to draw enough current to perform the required tasks without difficulties.

4.4 Connection Diagram

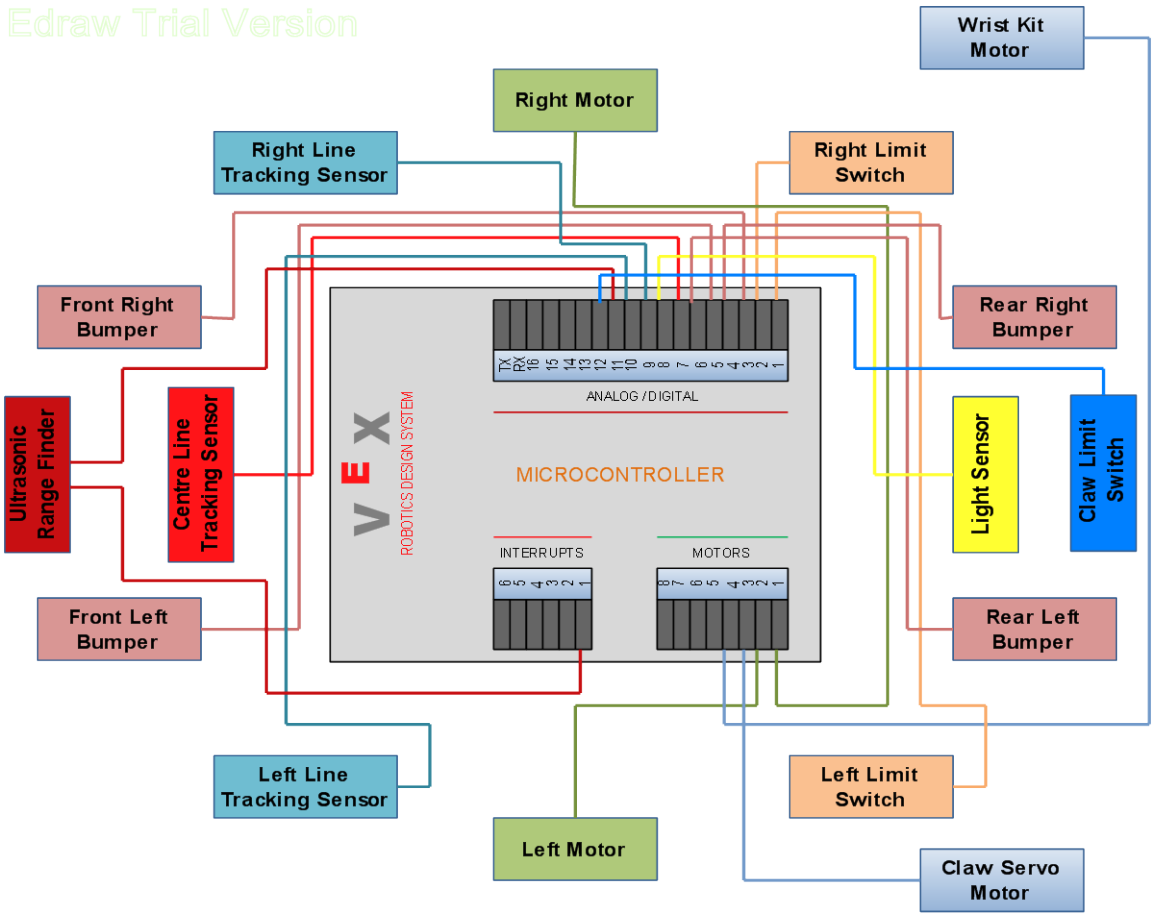


Figure 4.12 Connection of each component to the Microcontroller

4.5 Sensor Subsystem

The sensor subsystem is designed to gather information (actively and passively) about the environment. It will process this information and store it for use by the other subsystems. Active data gathering is the use of devices such as laser range finders and ultrasound systems that emit radiation into the environment. A passive data collection system would capture data such as reflected light and sound [17].

The robot can hence sense its environment, and can adjust its own behaviors based on that knowledge. Sensors are the beginning of that process. A sensor will generally tell the robot about one very simple thing in the environment, and the robot's program will interpret that information to determine how it should react.

4.6 Interaction of the Sensor Subsystem with other Subsystems

4.6.1 Structure Subsystem

The structure subsystem provides a mounting and stabilization platform for sensors on the robot. Often, sensors need to be held in a specific position to function properly, and the structural subsystem must be designed to accommodate these needs.
On the Squarebot, the structure subsystem provides a mounting spot for the bumper switch sensors, where they can detect collisions from the front and rear.

4.6.2 Motion Subsystem

Robots often have motors and other Motion components controlled by sensors (for instance, the emergency stop function stops the motors when the bumper switch sensor is pushed). However, the Sensor Subsystem does not directly control the Motion Subsystem. Instead, the Sensors provide information to the Micro Controller, which takes that information into account, and then decides what command to send to the Motion Subsystem.

4.6.3 Control Subsystem

The Control subsystem complements the Sensor Subsystem to achieve better control of the robot. The Control Subsystem provides human control over the robot, but the human operator does not always have perfect control, or the perfect point of view to see the robot's position. The Sensor Subsystem gives the robot the ability to make its own informed decisions, and can be a substantial aid to the operator.

On Squarebot, the bumper sensors can function as both tag game sensors and as emergency stop triggers. In the event that the operator accidentally drives into a wall or other obstacle, the sensor will stop the robot for a few seconds to prevent motor damage that could occur if the operator attempts to continue running the motors when the robot is stuck. The analog sensors are helpful for autonomous control of the robot.

4.6.4 Logic Subsystem

The Logic Subsystem relies on feedback from the Sensor Subsystem to provide information about the robot's environment. It uses this data to make informed decisions about how the robot should behave.

The actual behaviour that is activated when a sensor is triggered depends on which port the sensor is plugged into on the Micro controller and what the program commands for those ports are.

The software on command from the control program probes the ultrasound, light, line tracking and bumper sensors and gets the result, and on command from the control program runs the motors in the desired directions.

Chapter 5

Software System

5.1 Introduction

The robot can be programmed using three different C based programming softwares. Given below are the names of the softwares, which can be used to program the VEX microcontroller.

- MPLAB, Microchip compiler
- Robot C
- Easy C, Vex programming language

The Easy C software was used to program the robot's Microcontroller in this project. This software has been designed by Intellitek, specifically for programming the VEX Microcontroller. A programming guide is provided with the software package, showing some simple examples of controlling the motors and sensors. It can be quite useful for beginners or even the learned ones as it gives a brief idea of VEX Programming. This software has two modes of programming, one being Block Programming and the other Code Programming. The picture below shows the materials from the EasyC programming Kit, which includes: the EasyC Software CD, USB to Serial Cable, VEX Programming Module and the Programming Guide.

Figure 5.1 EasyC Programming Kit

The following pictures show screenshots of the programming windows for the above software

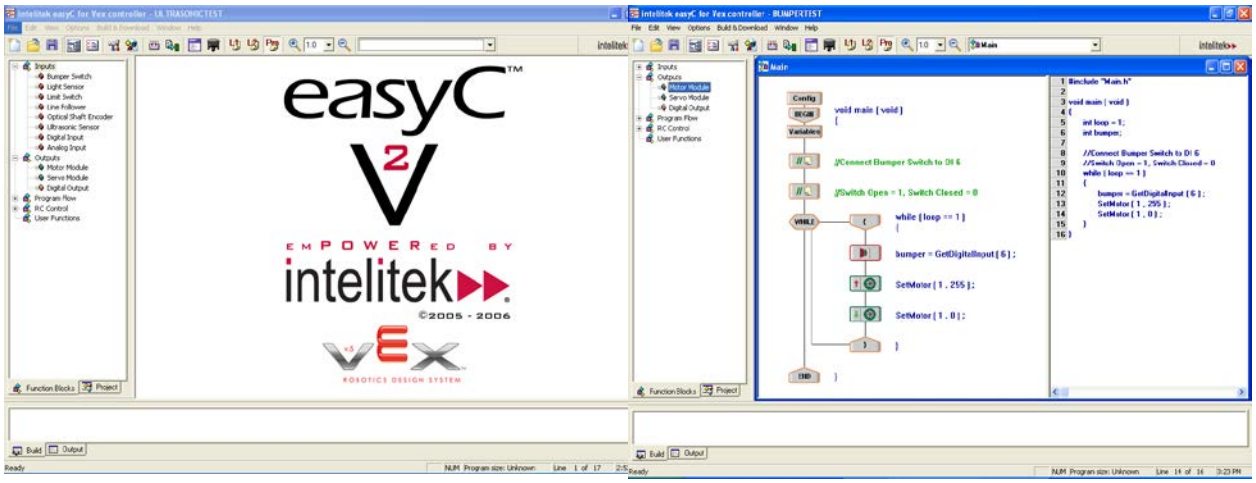
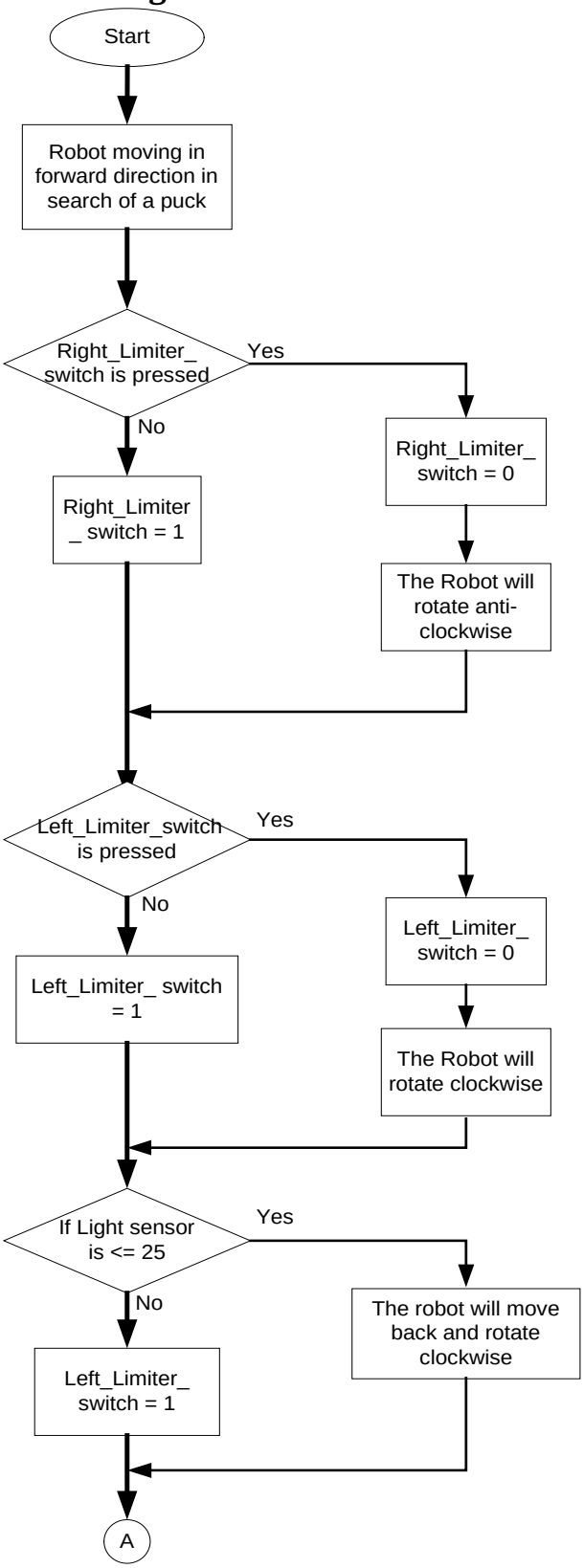


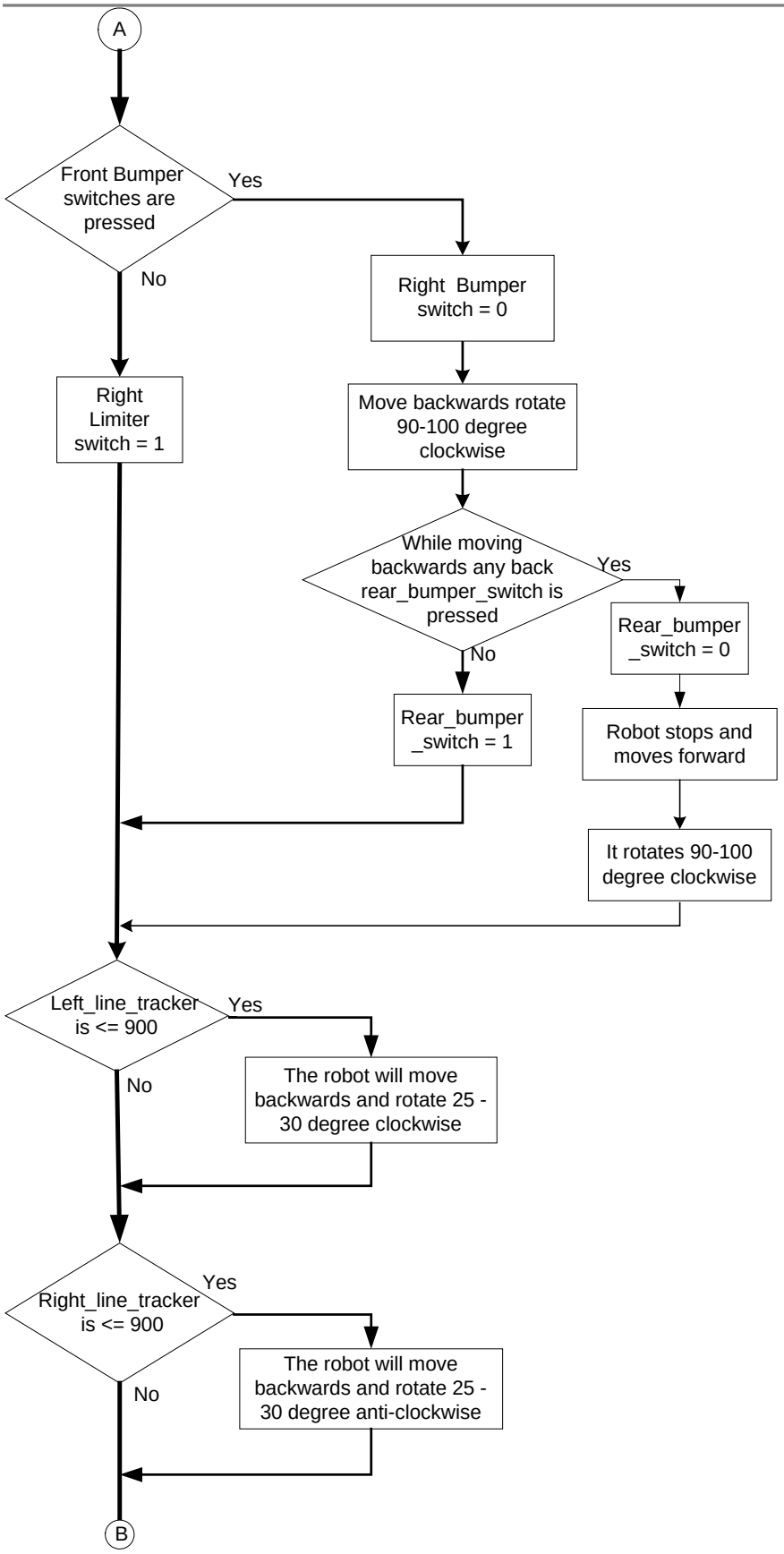
Figure 5.2 Screenshots of EasyC Programming Windows

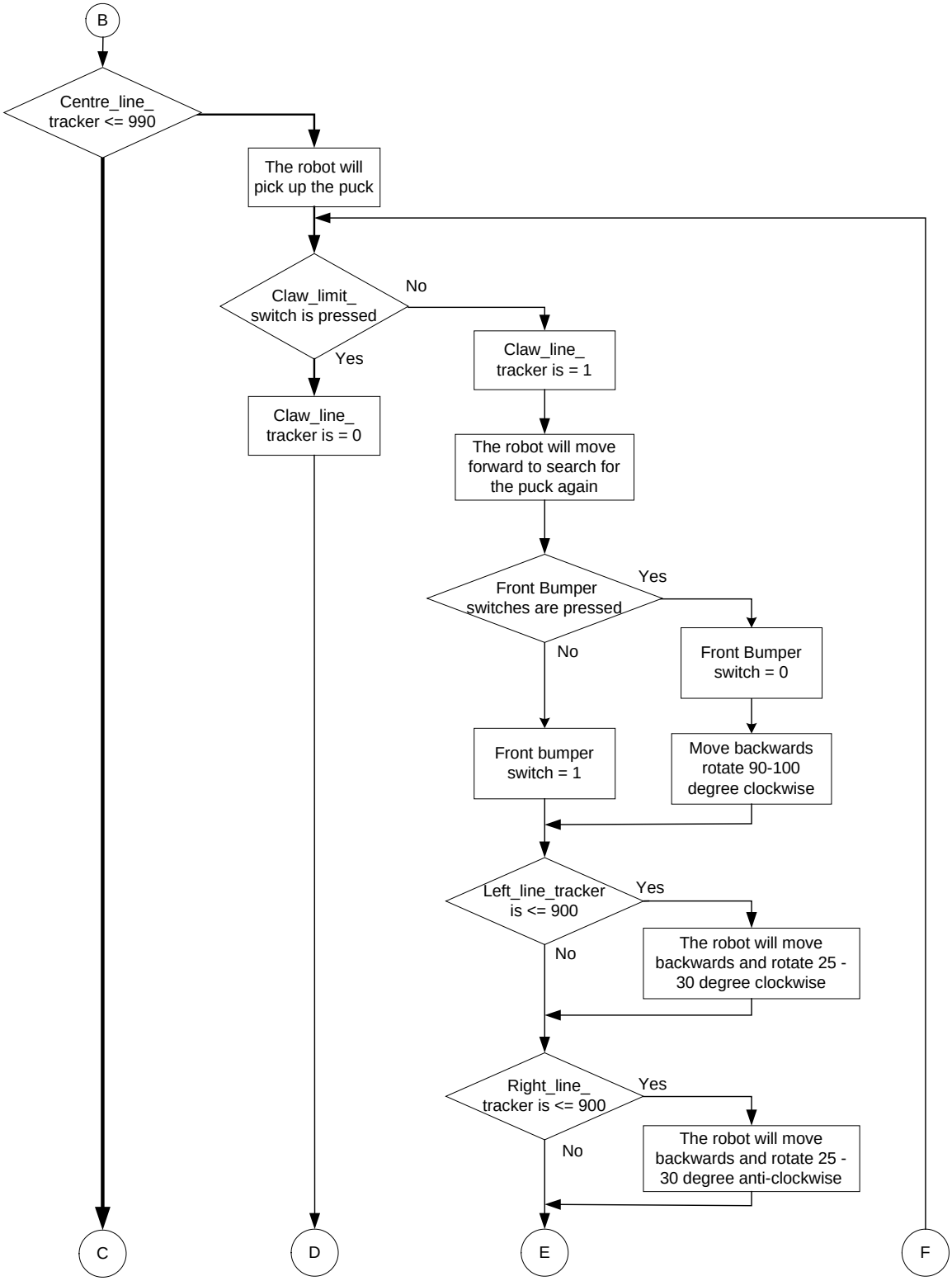
5.2 Control

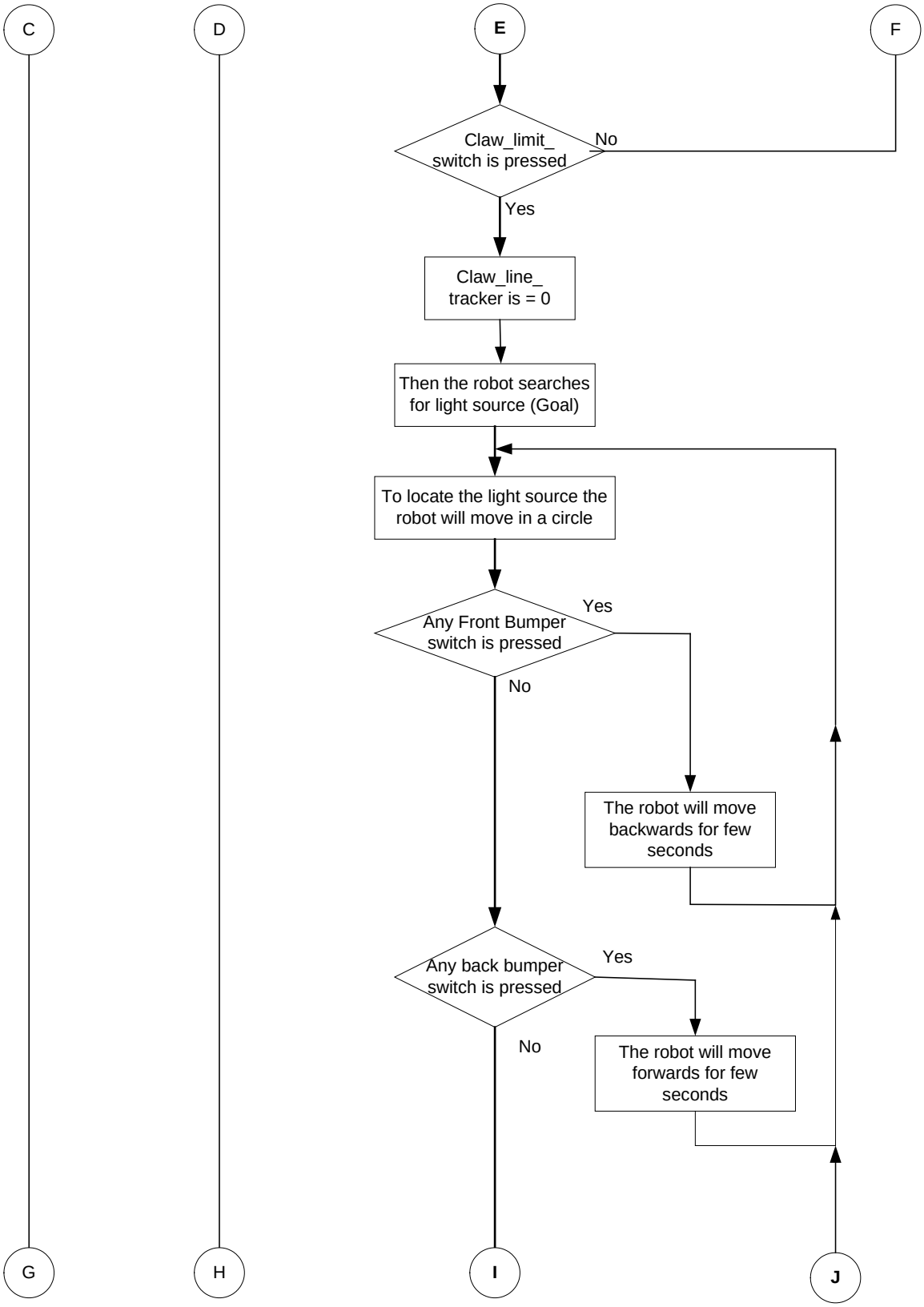
The robot control system is a special piece of software that must be incorporated on a robot. It must cope with data in many different grades of granularity, from sensor readings to user-supplied mission data; it must yield purposeful action on different time scales, from collision reflexes to optimal long- term mission organization. Consequently, a robot control program needs a special structure and organization that integrate these jumbled pieces into coherent overall action, that is, it needs a special architecture [1]. As discussed earlier, the reactive and the deliberative, plan based aspects of robot control cannot be a priori ordered by importance or precedence. On a short term, reaction is first in order to avoid bumping into walls and obstacles or falling down staircases. On a longer term, deliberated action is first in order to account for fulfilling the mission goals. The control of the VEX Robot is thoroughly discussed from the flowchart in the following pages. Modern robot control architectures are hybrid, i.e., they contain different layers for reactive and deliberative control components. Typically, a middle layer (often called sequencing layer) mediates between the reactive and the deliberative components, resulting in a three-layer architecture [18]. The control of this robot can be said to be of a similar structure, since the reactions of the robot are based on what feedback of the environment is received by the sensors. The robot then links between what it had been doing, to what it should do next.

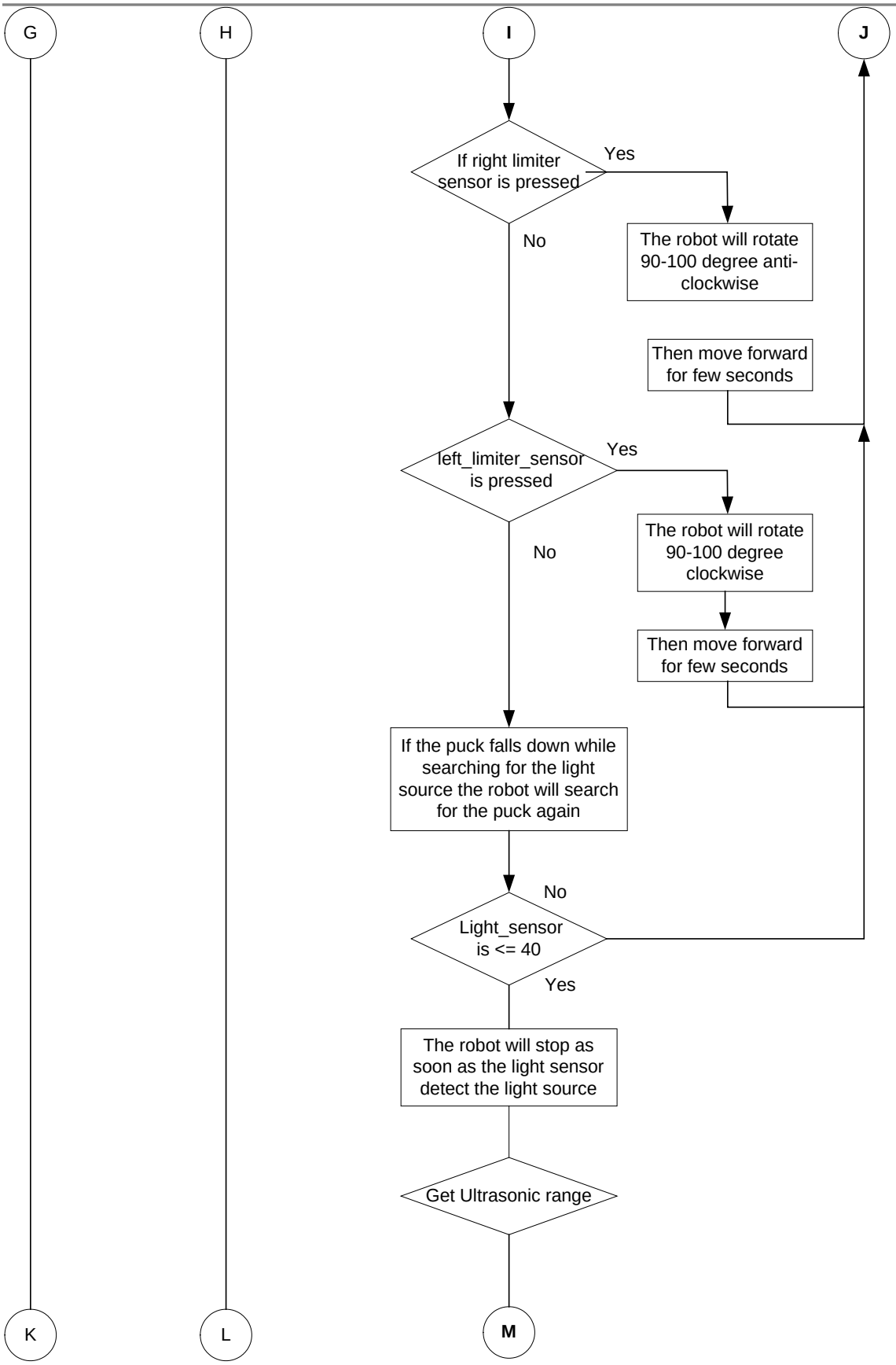
5.3 Program Flowchart




A
Front Bumper switches are pressed
Yes
No
Right Bumper switch = 0
Right Limiter switch = 1
Move backwards rotate 90-100 degree clockwise
While moving backwards any back rear_bumper_switch is pressed
Yes
No
Rear_bumper _switch = 0
Rear_bumper _switch = 1
Robot stops and moves forward
It rotates 90-100 degree clockwise
Left_line_tracker is <= 900
Yes
No
The robot will move backwards and rotate 25 - 30 degree clockwise
Right_line_tracker is <= 900
Yes
No
The robot will move backwards and rotate 25 - 30 degree anti-clockwise
B


B
Centre_line_
tracker <= 990
The robot will
pick up the puck
Claw_limit_
switch is pressed
No
Yes
Claw_line_
tracker is = 1
Claw_line_
tracker is = 0
The robot will move
forward to search for
the puck again
Front Bumper
switches are pressed
Yes
No
Front Bumper
switch = 0
Front bumper
switch = 1
Move backwards
rotate 90-100
degree clockwise
Left_line_tracker
is <= 900
Yes
No
The robot will move
backwards and rotate 25 -
30 degree clockwise
Right_line_
tracker is <= 900
Yes
No
The robot will move
backwards and rotate 25 -
30 degree anti-clockwise
C
D
E
F


C
D
E
F
Claw_limit_ switch is pressed
No
Yes
Claw_line_ tracker is = 0
Then the robot searches for light source (Goal)
To locate the light source the robot will move in a circle
Any Front Bumper switch is pressed
Yes
No
The robot will move backwards for few seconds
Any back bumper switch is pressed
Yes
No
The robot will move forwards for few seconds
G
H
I
J


G
H
I
J
If right limiter sensor is pressed
Yes
No
The robot will rotate 90-100 degree anti-clockwise
Then move forward for few seconds
left_limiter_sensor is pressed
Yes
No
The robot will rotate 90-100 degree clockwise
Then move forward for few seconds
If the puck falls down while searching for the light source the robot will search for the puck again
No
Light_sensor is <= 40
Yes
The robot will stop as soon as the light sensor detect the light source
Get Ultrasonic range
K
L
M

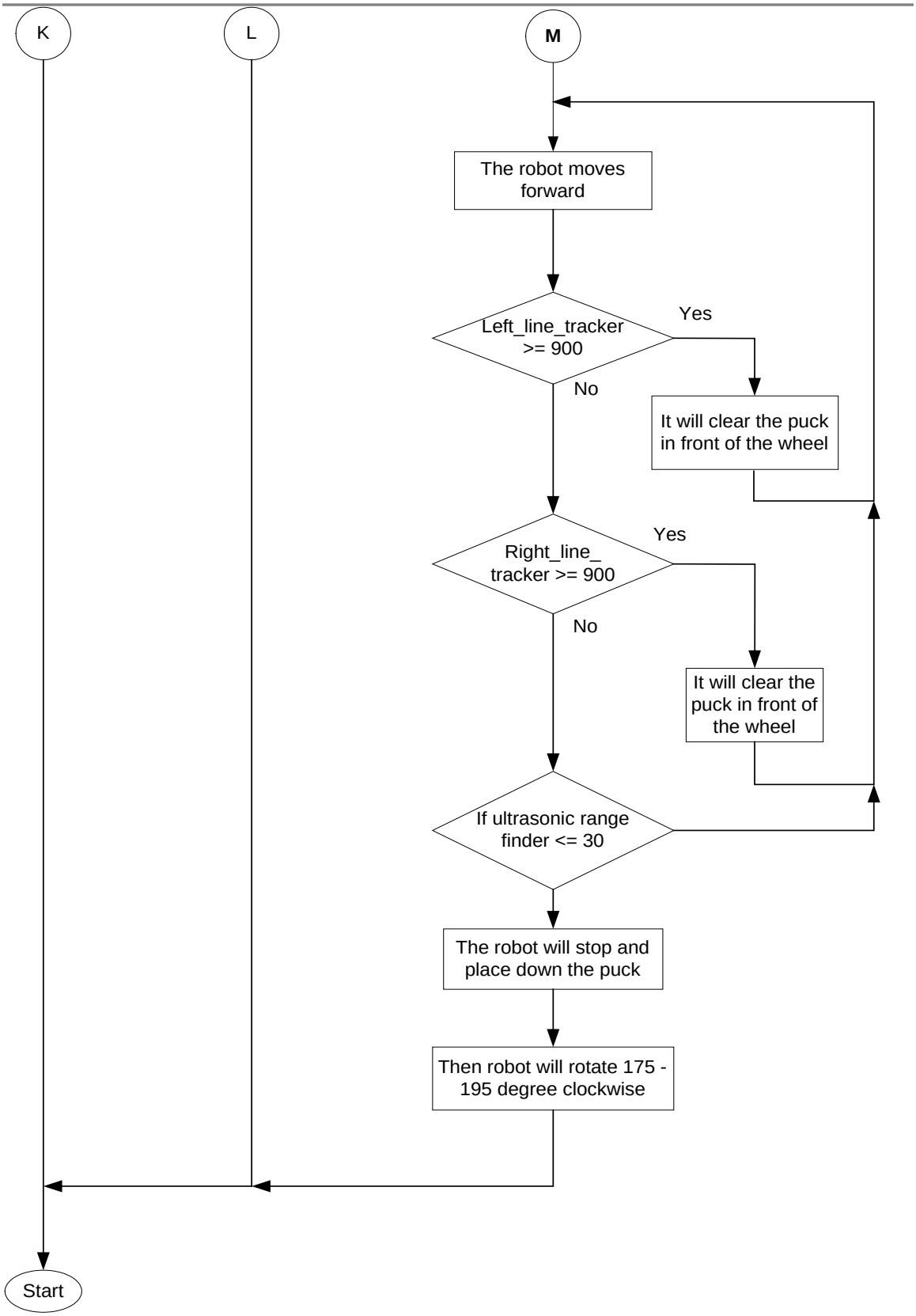


Figure 5.3 Program Flowchart

The flowchart on the previous pages shows, how the codes for the project were developed. The flowchart was done in this way so that it became self-explanatory with the easy to understand language. A general discussion of how the robot works according to the flowchart is given below.

When switched on, the robot starts to move in the forward direction. It would change course if any of the limiter or the bumper switches are pressed against the walls. The line tracing sensors will be active while the robot is doing this. If the sensors on the sides detect an object, the robot will stop, move back and turn a little so that the object is detected by the center- line tracer. When the object has been detected by this sensor, the robot will move back a bit to align itself to pick up the object. The motor for the arm will be rotated to lower the claw to hold the object. The claw limit switch will be pressed, indicating that the object has been picked up. If it is not pressed, it means that the object has not been picked up, so the robot will again try to pick it up. When the object has been picked up, the robot will start rotating to search for the light source. The light sensor will be doing this task. If the object falls while the robot is searching for light, the robot will again start searching the object, else it will stop when the light source has been detected. The robot will then start moving towards the light source. While going there if any of the switches are pressed, it will change its direction and again rotate to search for the light source. While going towards the light source, if the line trackers detect another object, the robot will avoid them, taking a different path to the goal location. While going towards the light source, the robot will keep track of the intensity of the light. If the value is out of the range, the robot will search for the light source again. The ultrasonic sensor will be active when going towards the goal location. When the light source value is in the minimum range and the ultrasonic sensor measures a value of 30, the robot will place the puck down. After placing the object, the robot will back up, turn and start searching for another object, i.e. it goes to the start of the program.

Chapter 6

Robot Field (Environment)

6.1 Introduction

Robot platform is the work plane or the environment in which a robot performs its functions. This varies for different robots designed for different applications. Some areas where the robots can be divided into are Industrial Robots, Mobile Robots (AGV's), robots used in Agriculture, Telerobots and Service Robots. According to these divisions, the platforms or the environment of operation will also change.

6.2 Industrial robots

Robots today are being utilized in a wide variety of industrial applications. Any job that involves repetitiveness, accuracy, endurance, speed, and reliability can be done much better by robots, which is why many industrial jobs that used to be done by humans are increasingly being done by robots [19]. Industrial robots are required to be performing operations in factories or production lines. Therefore they are designed in that way to perform well in their respective environments. For example, robots used for building cars, welding, spray-painting, assembly operations and palletizing and material handling.

Perhaps the most popular applications of robots is in industrial welding. The repeatability, consistency, quality, and speed of robotic welding is unrivaled. The two basic types of welding are spot welding and arc welding, although laser welding is done. Some environmental requirements should be considered for a successful operation in this application.

Another popular and efficient use for robots is in the field of spray painting. The consistency and repeatability of a robot's motion have enabled near perfect quality while at the same time wasting no paint. The spray painting applications seems to exemplify the proper applications of robotics, relieving the human operator from a hazardous, although skillful job, while at the same time increasing work quality, uniformity, and cutting costs.

Hence these robots would also be operating in their specific environments so that they become of proper application.

Assembly Operations is another area where industrial robots are widely used. Robots lend themselves well to the tedious and cyclical nature of assembly tasks provided that the proper planning and design have been done. In addition, their high level of repeatability has allowed the development of some new technologies in electronic assembly.

In Semiconductor Industry's IC Chip manufacturing facilities, various processes take place within a clean room. This requires that personnel as well as robots not introduce dirt, dust, or oil into the area. Since robots do not breath, sneeze, or have dandruff, they are especially suited to the clean room environment demanded by the semiconductor industry.

6.3 Mobile Robots

Mobile Robots commonly known as Automated Guided Vehicles, or AGV's, are used for transporting material over large sized places like hospitals, container ports, and warehouses, using wires or markers placed in the floor, or lasers, or vision, to sense the environment they operate in. An advanced form of the AGV is the SGV, or the Self Guided Vehicle, like PatrolBot Gofer, Tug, and Speci-Minder, which can be taught to autonomously navigate within a space, or do it by being given a map of the area. These robots have the ability of performing tasks that are non-sequential and non-repetitive in environments that are complex, hence are defined as intelligent robots [19].

6.4 Robots Used in Agriculture

Although the idea of robots planting seeds, ploughing fields, and gathering the harvest may seem straight out of a futuristic science fiction book, nevertheless there are several

robots in the experimental stages of being used for agricultural purposes, such as robots that can pick apples, prune grapevines, transplant seedlings, and so on. In fact, there already is a type of robot that shears sheep in Australia [19]. Again, the field of operation is of great consideration.

6.5 Telerobots

These robots are used in places that are hazardous to humans, or are inaccessible or far away. A human operator located at a distance from a Telerobot controls its action, which was accomplished with the arm of the space shuttle. Some other examples of telerobots are laparoscopic surgery being done with the help of a telerobot, or doctors using remotely located robots to communicate with their patients, which enables them to treat patients anywhere in the world. This has the potential of patients in remote places of the world, without adequate medical facilities, being able to consult doctors across the world, or even in the next town, and the doctors in turn having the ability to monitor them. Telerobots are also useful in nuclear power plants where they, instead of humans, can handle hazardous material or undertake operations potentially harmful for humans.

As is evident, telerobots are particularly useful for space exploration. Some of the applications in space that are on anvil are robots used for the maintenance of satellites, robotic arms for manufacturing in space, robots used for constructing space ships, space stations and so on.

Telerobots are also being increasingly used for military purposes, for instance the Unmanned Aerial Vehicle used for surveillance and also fire at targets. Some of them have even advanced to the level of having the ability to automatically make decisions like choosing the location to fly to, and deciding which enemy target to engage with. Many telerobots are being used by the US military in Afghanistan and Iraq to diffuse Improvised Explosive Devices [19].

6.6 Service Robots

This category comprises of any robot that is used outside an industrial facility, although they can be sub-divided into two main types of robots: one, robots used for professional jobs, and the second, robots used for personal use. Amongst the former type are the above mentioned robots used for military use, then there are robots that are used for underwater jobs, or robots used for cleaning hazardous waste, and the like.

Personal use robots are becoming more and more popular, with increased sophistication in Artificial Intelligence and with them becoming increasingly affordable, and are being seen in areas like care giving, pet robots, house cleaning and entertainment. Although it is more expensive and difficult to make highly intelligent and sensitive machines, but service robots designed with minimal intelligence are already fairly common, such as the vacuum cleaning robots.

Another area where personal use robots are being introduced is in the care for the elderly. In countries where there are increasing numbers of the aged with comparatively fewer numbers of young people to provide them with care, due to low birth rate and increased longevity, such as is the case in Japan and a growing number of Western countries, robots are increasingly thought to be the answer. These robots are being designed to provide physical services such as carrying bedridden elderly people (or even the handicapped), or washing for them, and doing various other day-to-day tasks. And then there are robots being designed to provide mental services, such as offering the therapeutic effect of interacting with the often-lonely elderly people [19].

Hence, as is evident, the trend is towards developing more and more sophisticated humanoid types of robots, with human-like physical features and intellectual abilities to suit the different environments of application.

6.7 Robot Platform Details

The platform used as the work plane for this project is a black surface of dimension: length 2.4m x width 2.4m, with boundaries of height 5 inches. The boundaries are for the purpose of keeping the robot within the enclosed area. A light source will be mounted on one of the walls, symbolizing the goal location. The light source being a 240V AC, 4 feet Fluorescent Lamp, so that there is even distribution of light on the field. The platform will have a number of white colored objects (pucks) scattered on it, which are supposed to be picked up by the robot.

The picture below shows the goal location of the robot field with the light source and the pieces of wood used as the pucks (objects.)

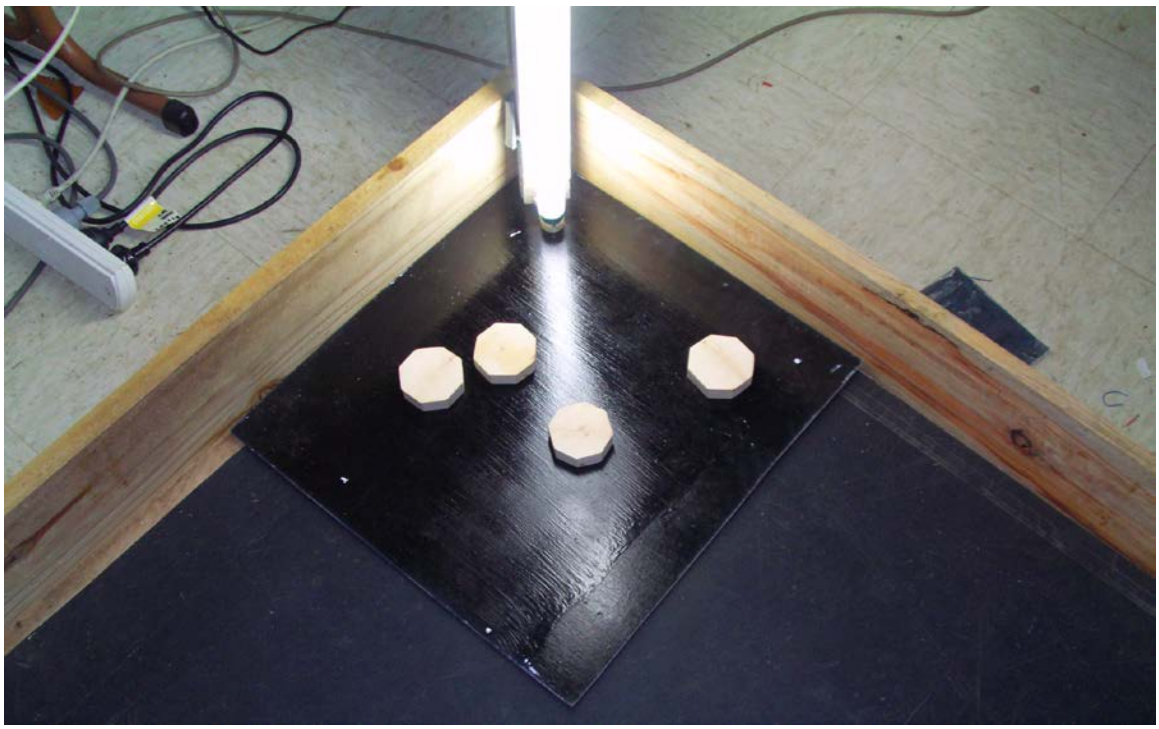


Figure 6.1 Robot Field

The picture below was taken when the robot was in operation on the field.

Figure 6.2 Robot Field with The Robot

Chapter 7

Performance Estimations

7.1 Speed

The speed of the VEX Robot varies according to the type of gear or chain system used to move it. The size of wheels used also makes a difference in the speed relative to the ground. Keeping the gear ratios and the wheel size constant, the speed can also be varied in the programming. It can be done by assigning different values of speed to the motors, where the Pulse Width Modulation supplied, will differ the speed.

Motor Rotation Clockwise = 128-255

Motor Rotation Anti-clockwise = 0-126

Motor Stop = 127

The further the value is from 127 to either side, the greater the speed is. The table below shows the speeds for different combinations of gears and wheels in feet per second.

Table 7.1 Robot Velocity for different combinations of Gears and Wheels

Reduction Details					Robot Velocity (fps)			
Reduction Type	Driving Size	Driven Size	Overall Reduction	Output % of Motor	Tank Tread	2.75" Wheel	4" Wheel	5" Wheel
Gear	12	84	7.00	14%	0.15	0.17	0.25	0.31
Gear	12	60	5.00	20%	0.20	0.24	0.35	0.44
Chain	10	48	4.80	21%	0.21	0.25	0.36	0.45
Chain	10	40	4.00	25%	0.26	0.30	0.44	0.55
Chain	15	48	3.20	31%	0.32	0.37	0.55	0.68
Gear	12	36	3.00	33%	0.34	0.40	0.58	0.73
Chain	15	40	2.67	38%	0.38	0.45	0.65	0.82
Chain	10	24	2.40	42%	0.43	0.50	0.73	0.91
Gear	36	84	2.33	43%	0.44	0.51	0.75	0.93
Chain	24	48	2.00	50%	0.51	0.60	0.87	1.09
Gear	36	60	1.67	60%	0.61	0.72	1.05	1.31
Chain	24	40	1.67	60%	0.61	0.72	1.05	1.31
Chain	15	24	1.60	63%	0.64	0.75	1.09	1.36
Chain	10	15	1.50	67%	0.68	0.80	1.16	1.45
Gear	60	84	1.40	71%	0.73	0.86	1.25	1.56
Chain	40	48	1.20	83%	0.85	1.00	1.45	1.82
Gear	12	12	1.00	100%	1.02	1.20	1.74	2.18
Gear	36	36	1.00	100%	1.02	1.20	1.74	2.18
Gear	60	60	1.00	100%	1.02	1.20	1.74	2.18
Gear	84	84	1.00	100%	1.02	1.20	1.74	2.18
Chain	10	10	1.00	100%	1.02	1.20	1.74	2.18
Chain	15	15	1.00	100%	1.02	1.20	1.74	2.18
Chain	24	24	1.00	100%	1.02	1.20	1.74	2.18
Chain	40	40	1.00	100%	1.02	1.20	1.74	2.18
Chain	48	48	1.00	100%	1.02	1.20	1.74	2.18

Chain	48	40	0.83	120%	1.23	1.44	2.09	2.62
Gear	84	60	0.71	140%	1.43	1.68	2.44	3.05
Chain	15	10	0.67	150%	1.54	1.80	2.62	3.27
Chain	24	15	0.63	160%	1.64	1.92	2.79	3.49
Gear	60	36	0.60	167%	1.71	2.00	2.91	3.63
Chain	40	24	0.60	167%	1.71	2.00	2.91	3.63
Chain	48	24	0.50	200%	2.05	2.40	3.49	4.36
Gear	84	36	0.43	233%	2.39	2.80	4.07	5.09
Chain	24	10	0.42	240%	2.46	2.88	4.19	5.23
Chain	40	15	0.38	267%	2.73	3.20	4.65	5.81
Gear	36	12	0.33	300%	3.07	3.60	5.23	6.54
Chain	48	15	0.31	320%	3.28	3.84	5.58	6.98
Chain	40	10	0.25	400%	4.10	4.80	6.98	8.72
Chain	48	10	0.21	480%	4.92	5.76	8.37	10.47
Gear	60	12	0.20	500%	5.12	6.00	8.72	10.90
Gear	84	12	0.14	700%	7.17	8.40	12.21	15.26

7.2 Battery life

The robot was initially using 6 x AA size batteries and these lasted for only around two hours of continuous operation. Since re-chargeable batteries were not available, it would become very expensive to operate the robot with single use batteries. Therefore, a DC Power Supply was used for this project purposes. Even though, the supply current was sometimes less, it was enough to drive the robot.

7.3 Sensor Performance

The performance of the robot in general depends on how each of the components perform. The sensors determine what actions the robot needs to take; therefore the performance of the sensors for the specific tasks should be analyzed. The sensors can be assessed on how well they are able to operate to accomplish the task they are indulged in.

7.3.1 Height at which Objects are Detected

The range of the line tracker sensors used for object detection is 0.05cm to 2.54cm and the distance at which the objects are to be detected in this project is about 1.5 centimeters from the line tracker to the objects surface. The sensors can be said to be quite effective

at this distance since it falls in the mid-range. It is not that the objects are detected at all the times. This is sometimes because of the speed at which the robot is moving, where the sensor crosses the object before a reflection is detected.

7.3.2 Distance at which objects are placed

When the robot has picked up an object, it would need to take it to the goal location and place it there. The ultrasonic sensor is used to detect the distance at which to place the object. The ultrasonic has a working range of 3cm to 3m. The distance at which the robot is supposed to place the object is 25 centimeters from the ultrasonic sensor to the walls at the goal location, which is at one of the corners.

7.3.3 When are the Obstacles Avoided

When the robot is moving in the forward direction and an object comes in the path of either of the wheels, it should stop and take a different path so that the object does not come under the wheels. To accomplish this task, line-tracking sensors have been placed in front of the wheels. While the robot is searching for objects, these sensors will help in guiding the robot so that the sensor placed in the centre detects the object.

When the robot has picked up an object and is in the process of transporting it to the goal location, any object in its path will be avoided but not picked up by the center line-tracker. If the object is detected by the left line-tracker, the robot first moves back and turns a little towards right and then continues to move ahead and if the object is detected by the right line-tracker, the robot will move back a little and then turn towards left and again move forward. This keeps happening whenever these sensors detect an object.

7.3.4 Object Gripping

The robot was not always able to pick up the objects with the claws at the first instant. Sometimes it started to go to the goal location without the object. To overcome this problem, a limit switch was placed on top of the claw, so that it would be pressed when an object would come in the grip of the claws. The programming was modified so that

the robot will not start detecting the light source until it had picked up an object. When the robot has not picked up the object in the first instant. It will go forward to sense it again, and move forward to pick it up. These actions will be repeated until the object has been picked up.

7.3.5 Light Source Detection

The intensity of the light varies for different areas on the field. The brightest being close to the fluorescent lamp and it becomes dimmer going further from the light source. Hence, for the robot to be able to detect the light source and be able to go to the correct position, values at several instances need to be taken from the light sensor. After the robot has picked up an object, it starts to rotate and the first value it searches for is between 85-80. When it has detected this value, it moves a little forward and then rotates again to search the values between 72-68. After this value is also detected, it moves a little forward and then rotates again to search the values between 67-60. When it has detected this value, it moves a little forward and then rotates again to search the values between 59-53. Finally it detects for values between 52-45 and again moves forward. Depending on the position where the robot first starts to rotate, it may miss out on some of the ranges values starting from the closest. The robot searches for the light source until the values are greater than 40 because by then it is quite close to the goal location. As soon as a value less than 40 is detected, the robot starts moving in that direction and the ultrasonic sensor is activated to detect the walls.

7.3.6 Accuracy of arrival at Goal Location

At most of the trials, the robot goes to the desired location to place the objects. But in some cases it misses out a little. This happens because when the robot rotates to detect the light source, it may not stop at the exact position where it had detected the source because of the processing time. Also sometimes, because of the position in which the robot is when it picks up the object. When the robot is really close to the walls, it goes to a slightly further position of the goal location because it is moving away from the walls.

Chapter 8

Experimentation Results

8.1 Introduction

This chapter presents the results of some of the tests carried out with the robot to analyze its performance and efficiency. Tests were carried out in terms of the time taken for the robot to complete the given tasks. Results were taken by increasing the number of objects, going up to a maximum of 5 objects. For each number of objects, the time was taken for 10 trials. This was done so that graphs could be plotted showing the variations in the times for each trial.

8.2 Test for the Time Taken to Detect and Transport One Object.

Three experiments were carried out to measure the time taken to transport the object. These were for the different number of attempts at which the robot was allowed to pick up the object. For the trial s where only one object was used, the results were taken for three different positions of the puck (object).

8.2.1 First Position

The first test was carried out with the object furthest away from the robot. The diagram below shows the position of the puck.

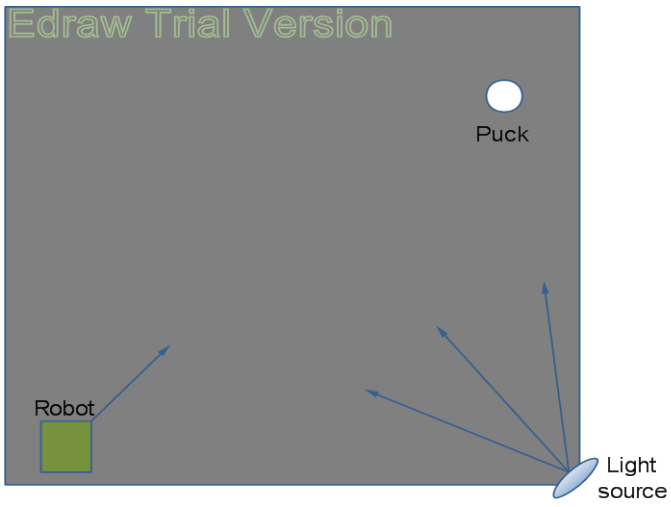


Figure 8.1 Position of the object on the field relative to the robot for first test.

8.2.1.1 First Attempt

These results were obtained when the robot was able to grab the object at the first attempt of detection.

Table 8.1 Time taken for the robot to transport One Object for first test at First Attempt.

No. of Trials	Time Taken (sec)
1	53
2	65
3	57
4	62
5	58
6	49
7	46
8	56
9	51
10	67

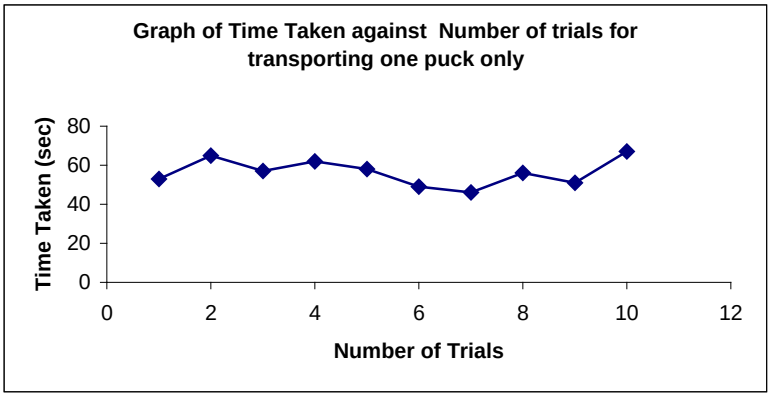


Figure 8.2 Graph of the time taken to transport one object on first test at the first Attempt.

8.2.1.2 Second Attempt

These results were obtained when the robot was able to handle the object at the second attempt, i.e. the object was removed from its place when the robot first detected it. The object was placed back when the robot was unsuccessful at the first attempt.

Table 8.2 Time taken for the robot to transport One Object for first test at Second Attempt.

No. of Trials	Time Taken (sec)
1	77
2	72
3	73
4	75
5	71
6	65
7	69
8	70
9	69
10	73

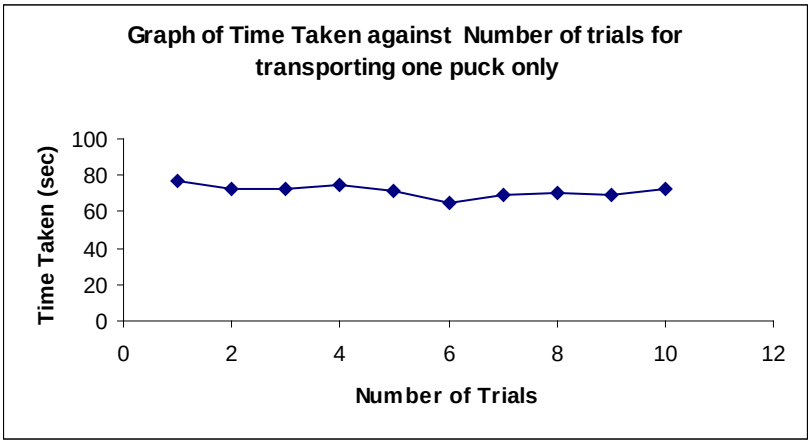


Figure 8.3 Graph of the time taken to transport one object on first test at the second Attempt.

8.2.1.3 Third Attempt

These results were obtained when the robot was allowed to handle the object at the second attempt, i.e. the object was removed from its place when the robot first detected it and this was repeated for the second attempt. The object was placed back when the robot was unsuccessful at the first two attempts.

Table 8.3 Time taken for the robot to transport One Object for first test at Third Attempt.

No. of Trials	Time Taken (sec)
1	75
2	77
3	74
4	75
5	76
6	75
7	79
8	81
9	79
10	76

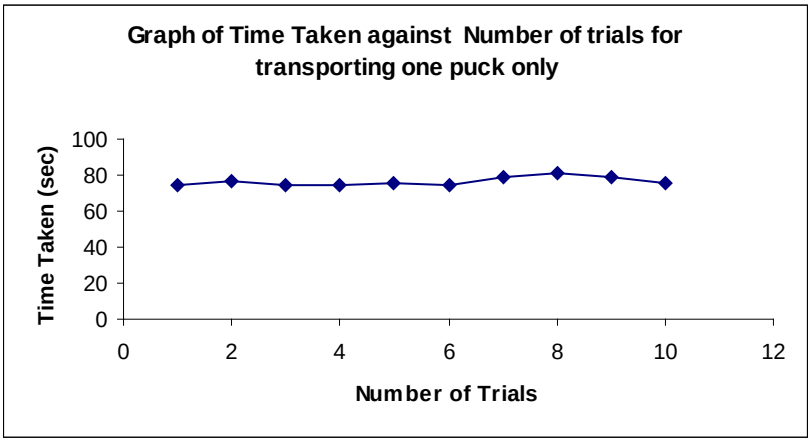


Figure 8.4 Graph of the time taken to transport one object on first test at the Third Attempt.

8.2.1.4 Comparison of the Three Attempts

To show the difference between the time taken at the various attempts, a comparison graph was plotted.

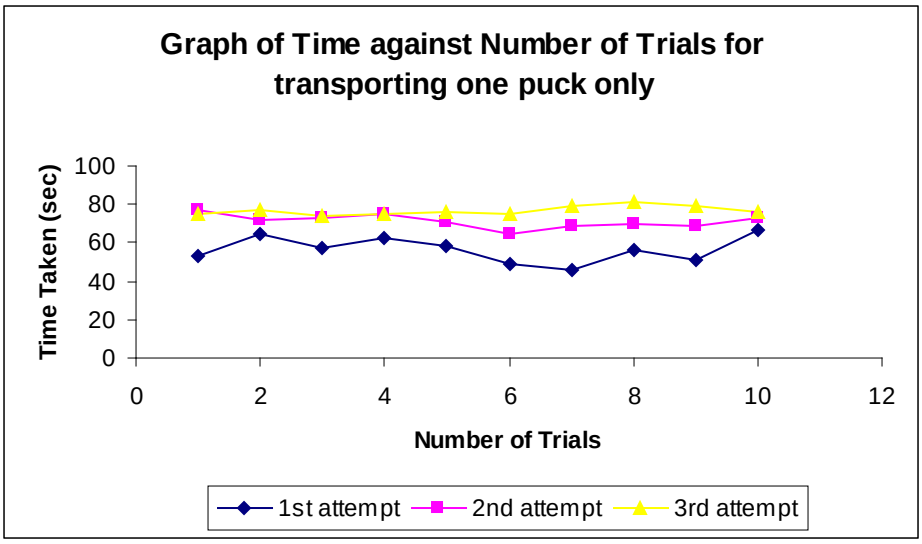


Figure 8.5 Graph showing the comparison of the result of one object for the three attempts.

8.2.2 Second Position

The second test was carried out with the object directly in front of the robot but the robot first starts to move diagonally. The diagram below shows the position of the puck.

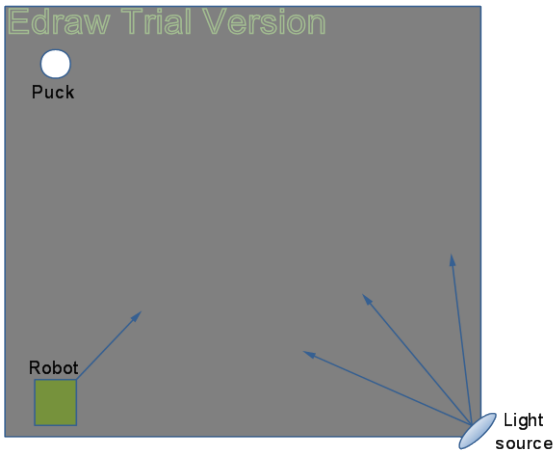


Figure 8.6 Position of the object on the field relative to the robot for second test.

8.2.2.1 First attempt

These results were obtained when the robot was able to grab the object at the first attempt of detection.

Table 8.4 Time taken for the robot to transport One Object for second test at First Attempt.

No. of Trials	Time Taken (sec)
1	45
2	48
3	42
4	55
5	51
6	50
7	46
8	60
9	55
10	52

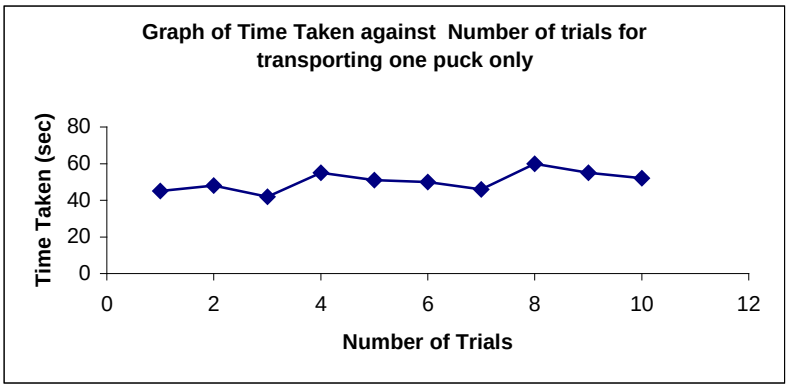


Figure 8.7 Graph of the time taken to transport one object on second test at the first Attempt.

8.2.2.2 Second Attempt

These results were obtained when the robot was able to handle the object at the second attempt, i.e. the object was removed from its place when the robot first detected it. The object was placed back when the robot was unsuccessful at the first attempt.

Table 8.5 Time taken for the robot to transport One Object for second test at Second Attempt.

No. of Trials	Time Taken (sec)
1	67
2	70
3	58
4	49
5	51
6	58
7	63
8	60
9	60
10	66

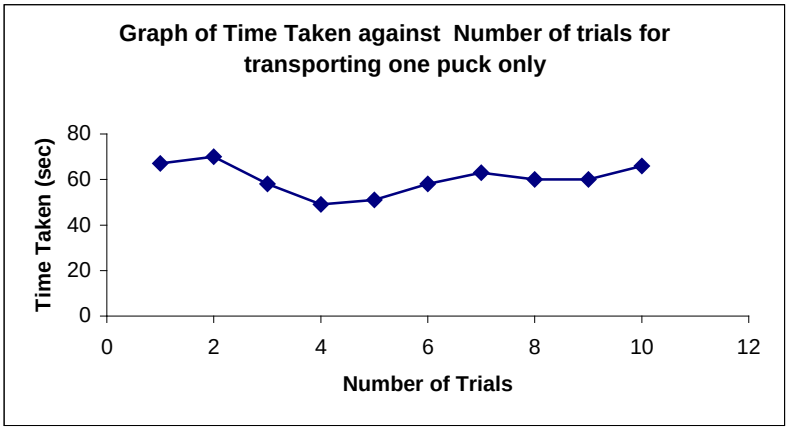


Figure 8.8 Graph of the time taken to transport one object on second test at the second Attempt.

8.2.2.3 Third Attempt

These results were obtained when the robot was allowed to handle the object at the third attempt, i.e. the object was removed from its place when the robot first detected it and this was repeated for the second attempt. The object was placed back when the robot was unsuccessful at the first two attempts.

Table 8.6 Time taken for the robot to transport 1 Object for second test at Third Attempt.

No. of Trials	Time Taken (sec)
1	80
2	71
3	72
4	73
5	82
6	78
7	76
8	58
9	79
10	81

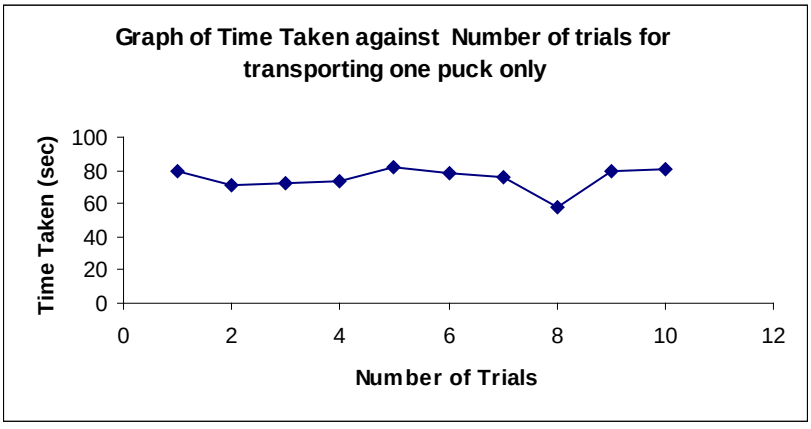


Figure 8.9 Graph of the time taken to transport one object on second test at the Third Attempt.

8.2.2.4 Comparison of the Three Attempts

To show the difference between the time taken at the various attempts, again a comparison graph was plotted.

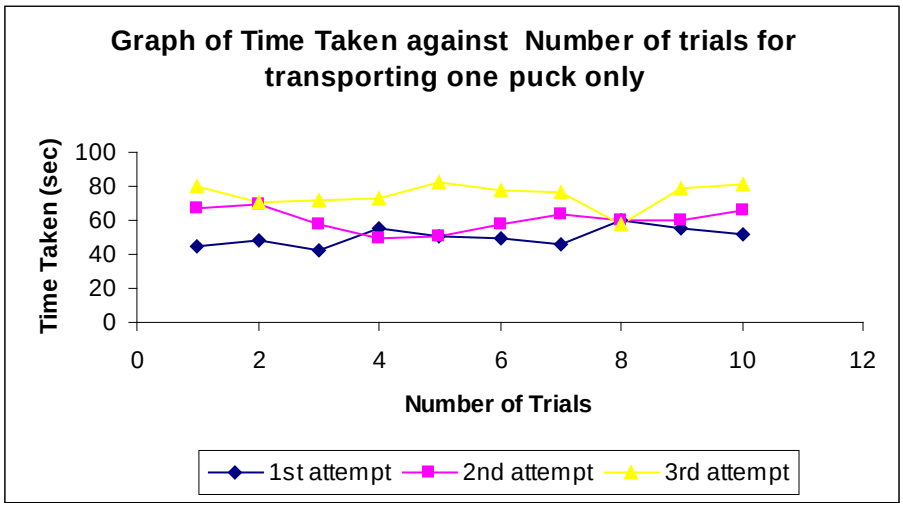


Figure 8.10 Graph showing the comparison of the time taken for three attempts.

8.2.3 Third Position

The third test for one object was carried out with the object almost in the center of the field, with the robot going and detecting it directly. The diagram below shows the position of the puck.

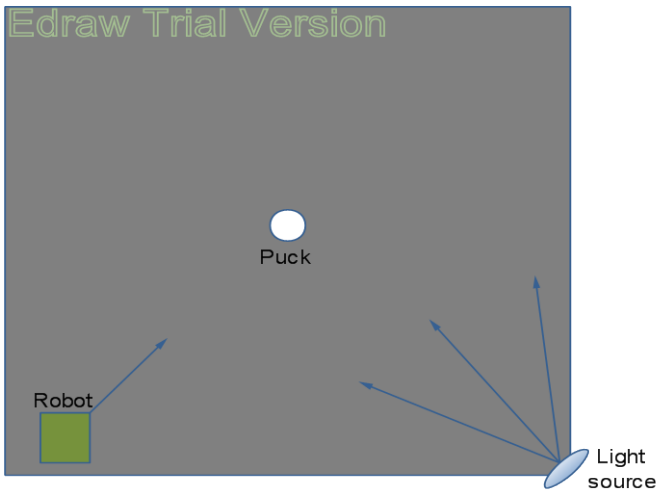


Figure 8.11 Position of the object on the field relative to the robot for third test.

8.2.3.1 First Attempt

These results were obtained when the robot was able to grab the object at the first attempt of detection.

Table 8.7 Time taken for the robot to transport One Object for third test at First Attempt.

No. of Trials	Time Taken (sec)
1	45
2	48
3	42
4	55
5	51
6	50
7	46
8	60
9	55
10	52

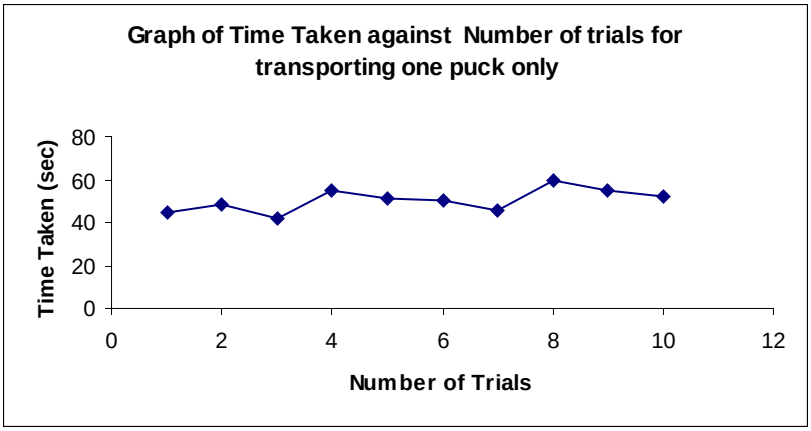


Figure 8.12 Graph of the time taken to transport one object on third test at the first Attempt.

8.2.3.2 Second Attempt

These results were obtained when the robot was able to handle the object at the second attempt, i.e. the object was removed from its place when the robot first detected it. The object was placed back when the robot was unsuccessful at the first attempt.

Table 8.8 Time taken for the robot to transport One Object for third test at second Attempt.

No. of Trials	Time Taken (sec)
1	67
2	70
3	58
4	49
5	51
6	58
7	63
8	60
9	60
10	66

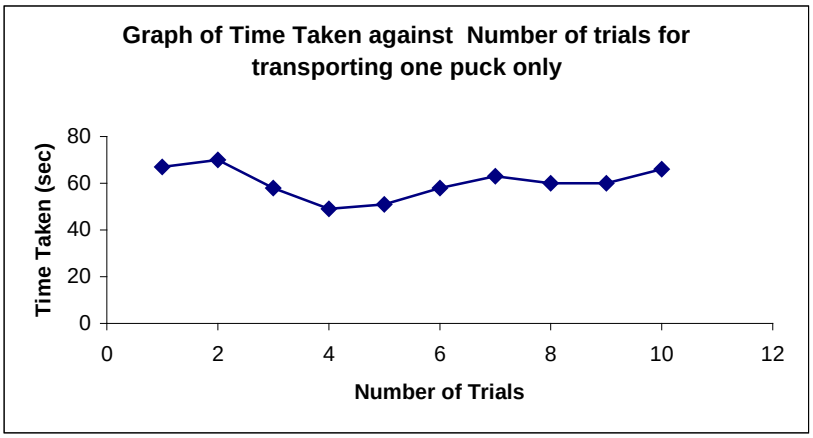


Figure 8.13 Graph of the time taken to transport one object on third test at the second Attempt.

8.2.3.3 Third Attempt

These results were obtained when the robot was allowed to handle the object at the third attempt, i.e. the object was removed from its place when the robot first detected it and this was repeated for the second attempt. The object was placed back when the robot was unsuccessful at the first two attempts.

Table 8.9 Time taken for the robot to transport One Object for third test at third Attempt.

No. of Trials	Time Taken (sec)
1	80
2	71
3	72
4	73
5	82
6	78
7	76
8	58
9	79
10	81

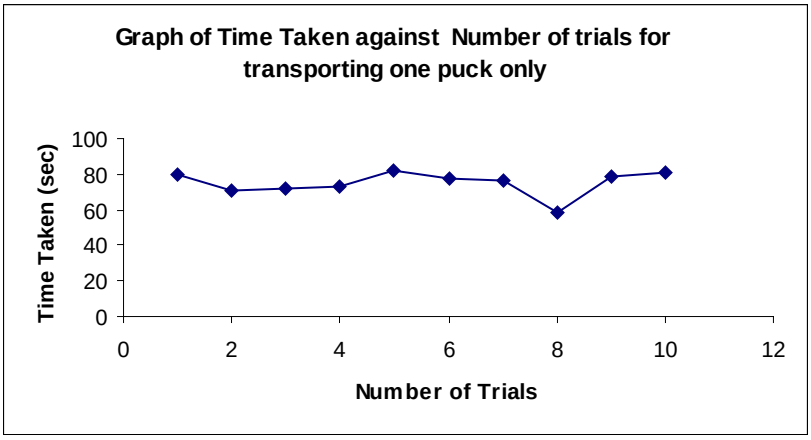


Figure 8.14 Graph of the time taken to transport one object on third test at the Third Attempt.

8.2.3.4 Comparison of the Three Attempts

To show the difference between the time taken at the various attempts, again a comparison graph was plotted.

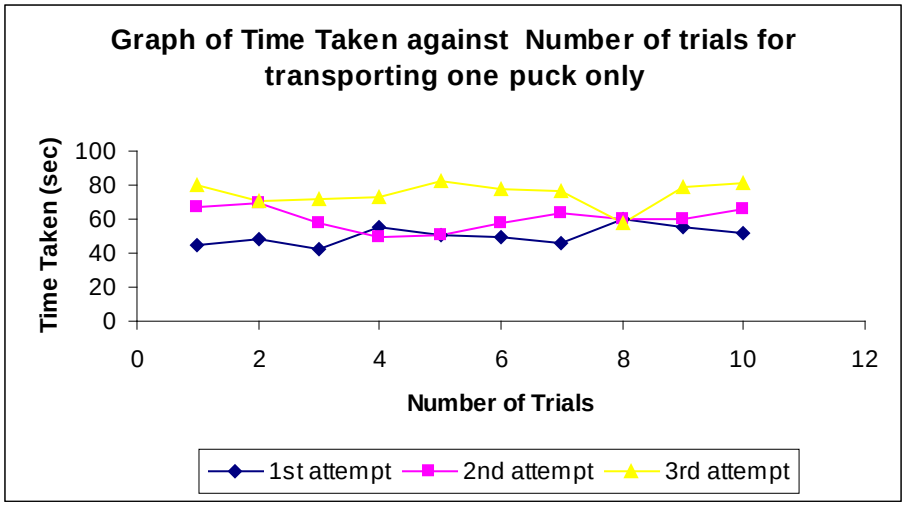


Figure 8.15 Graph showing the comparison of the third test results of one object for the three attempts.

8.3 Test for the Time Taken to Detect and Transport Two Objects.

The results were then taken by placing two objects on the field. The positions of the pucks are in the diagram below.

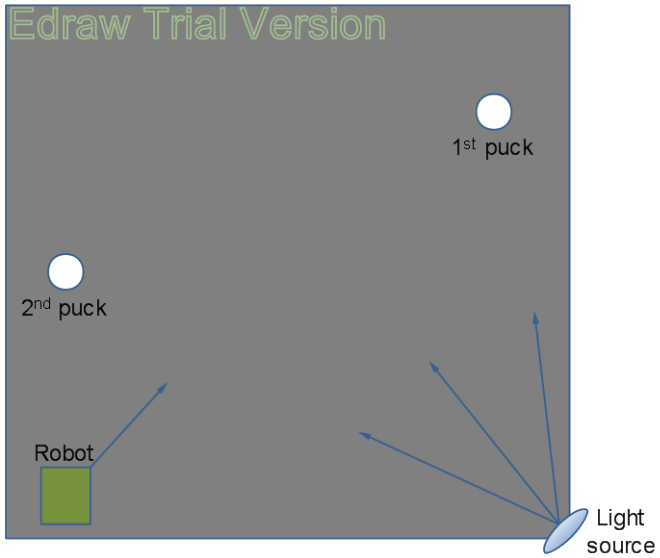


Figure 8.16 Positions of two object on the field relative to the robot.

8.3.1 First Attempt

These results were obtained when the robot was able to grab the objects at the first attempt of detection for both.

Table 8.10 Time taken for the robot to transport two objects at First Attempt.

No. of Trials	Time Taken (sec)
1	96
2	113
3	109
4	92
5	112
6	105
7	96
8	107
9	108
10	113

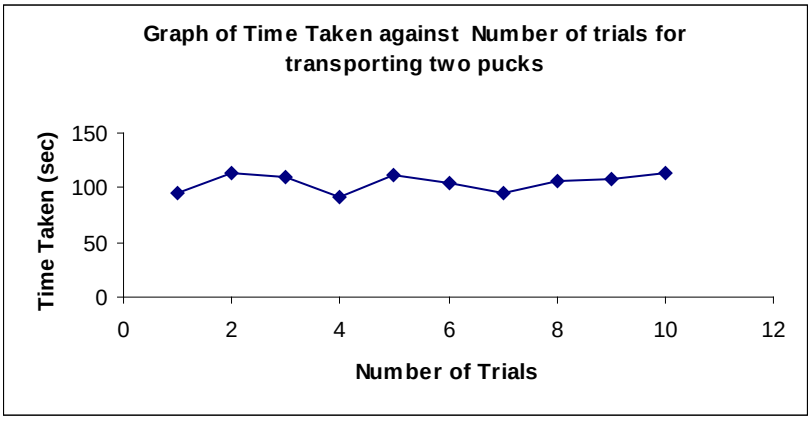


Figure 8.17 Graph of the time taken to transport two objects at first Attempt.

8.3.2 Second Attempt

These results were obtained when the robot was able to handle both the objects at the second attempt, i.e. the objects were removed from their places when the robot first detected them. They were placed back when the robot failed at the first attempt.

Table 8.11 Time taken for the robot to transport two objects at second Attempt.

No. of Trials	Time Taken (sec)
1	144
2	153
3	148
4	140
5	145
6	147
7	151
8	148
9	150
10	147

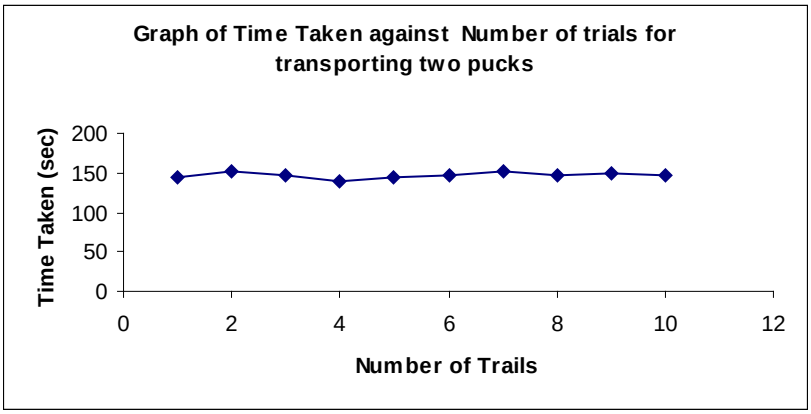


Figure 8.18 Graph of the time taken to transport two objects at second Attempt.

8.3.3 Third Attempt

These results were obtained when the robot was allowed to handle the objects at the third attempt, i.e. the objects were removed from their places when the robot first detected them and this was repeated for the second attempt. The objects were placed back when the robot was unsuccessful at the first two attempts.

Table 8.12 Time taken for the robot to transport two objects at third Attempt.

No. of Trials	Time Taken (sec)
1	156
2	154
3	149
4	153
5	155
6	158
7	160
8	155
9	153
10	159

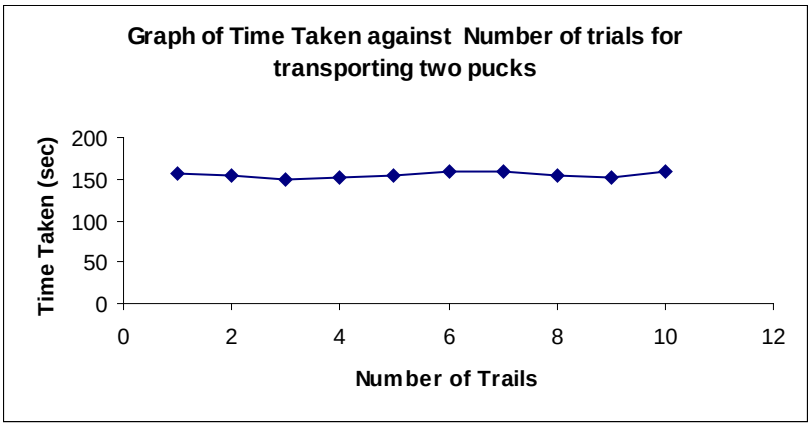


Figure 8.19 Graph of the time taken to transport two objects at third Attempt.

8.3.4 Comparison of the Three Attempts

To show the difference between the time taken at the various attempts, a comparison graph was plotted with all the three attempts.

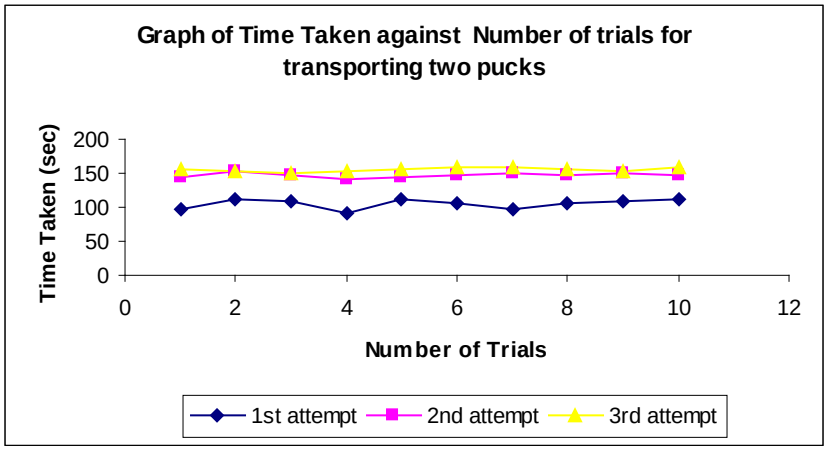


Figure 8.20 Graph showing the comparison of the results of two objects for three attempts.

8.4 Test for the Time Taken to Detect and Transport Three Objects.

Another test was then done with three objects on the field. The positions for the objects were randomly selected. The positions of the pucks are in the diagram below.

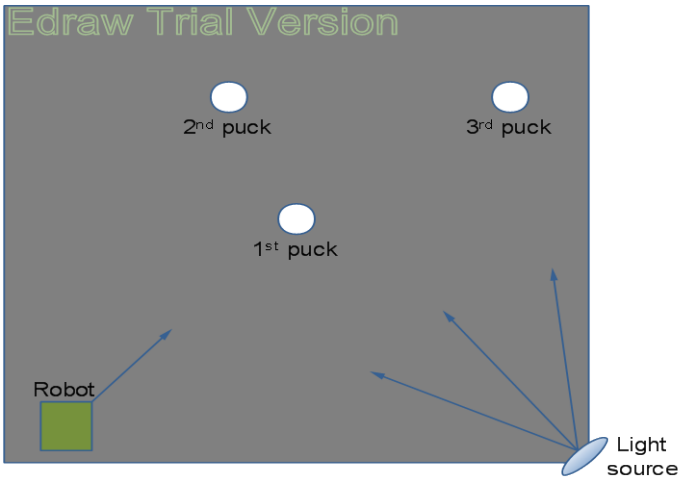


Figure 8.21 Positions of three objects on the field relative to the robot.

8.4.1 First Attempt

These results were obtained when the robot was able to grab the objects at the first attempt of detection for all the three objects.

Table 8.13 Time taken for the robot to transport three objects at First Attempt.

No. of Trials	Time Taken (sec)
1	161
2	156
3	164
4	158
5	157
6	155
7	162
8	157
9	163
10	161

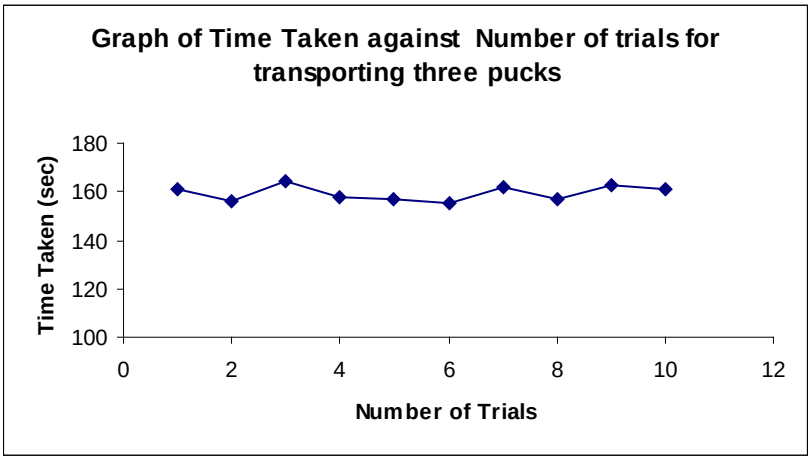


Figure 8.22 Graph of the time taken to transport three objects at first Attempt.

8.4.2 Second Attempt

These results were obtained when the robot was able to handle all the objects at the second attempt, i.e. the objects were removed from their places when the robot first detected them. They were placed back when the robot failed at the first attempt.

Table 8.14 Time taken for the robot to transport three objects at second Attempt.

No. of Trials	Time Taken (sec)
1	182
2	187
3	179
4	190
5	183
6	178
7	184
8	189
9	192
10	178

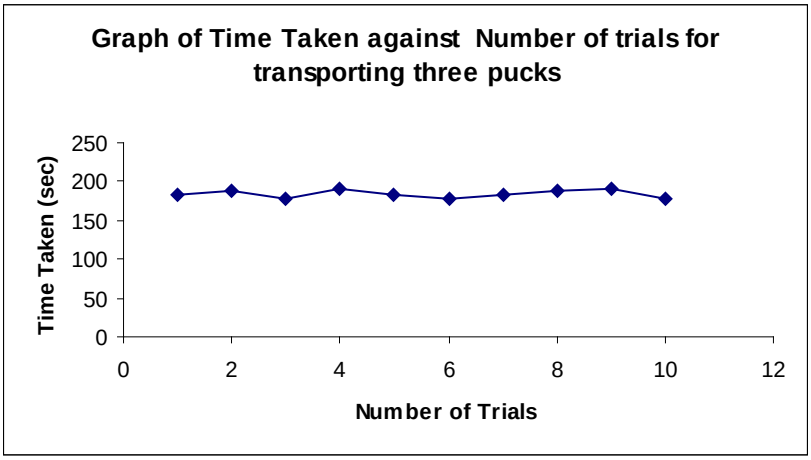


Figure 8.23 Graph of the time taken to transport three objects at second Attempt.

8.4.3 Third Attempt

These results were obtained when the robot was able to handle the objects at the third attempt, i.e. the objects were removed from their places when the robot first detected them and this was repeated for the second attempt. The objects were placed back when the robot was unsuccessful at the first two attempts.

Table 8.15 Time taken for the robot to transport three objects at third Attempt.

No. of Trials	Time Taken (sec)
1	221
2	227
3	216
4	222
5	219
6	220
7	217
8	223
9	220
10	214

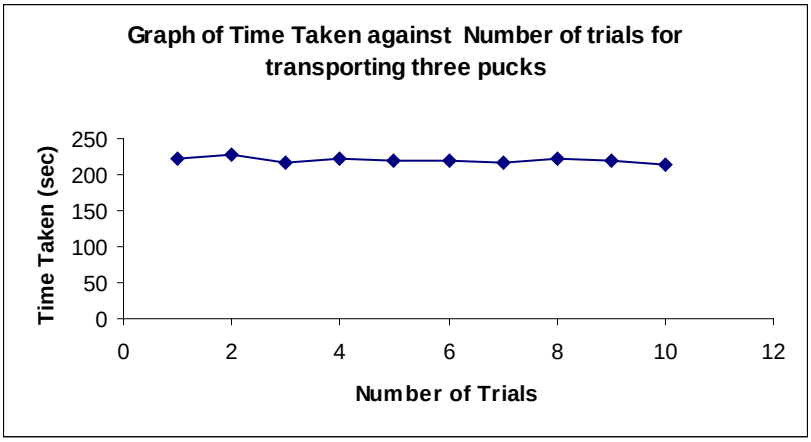


Figure 8.24 Graph of the time taken to transport three objects at third Attempt.

8.4.4 Comparison of the Three Attempts

To show the difference between the time taken at the various attempts, a comparison graph was plotted with all the three attempts for three objects.

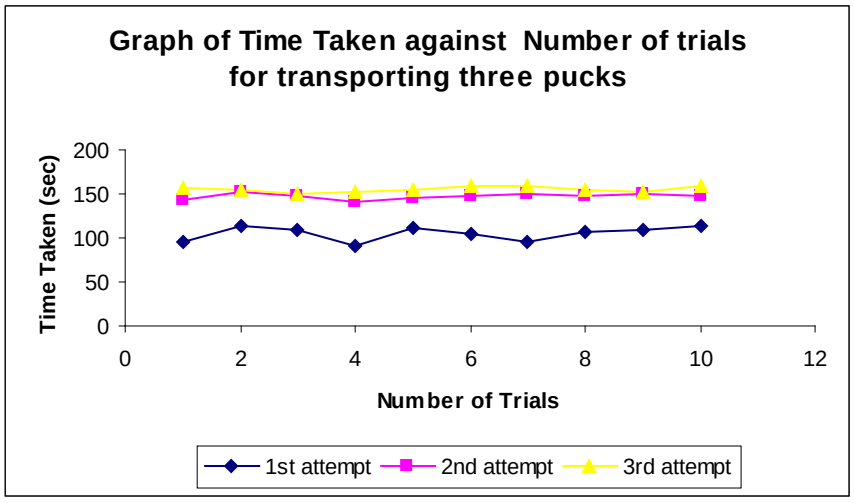


Figure 8.25 Graph showing the comparison of the results of three objects for three attempts.

8.5 Test for the Time Taken to Detect and Transport Four Objects.

The next test was then done with four objects on the field. The positions of the objects were again randomly selected. The positions of the pucks have been illustrated below.

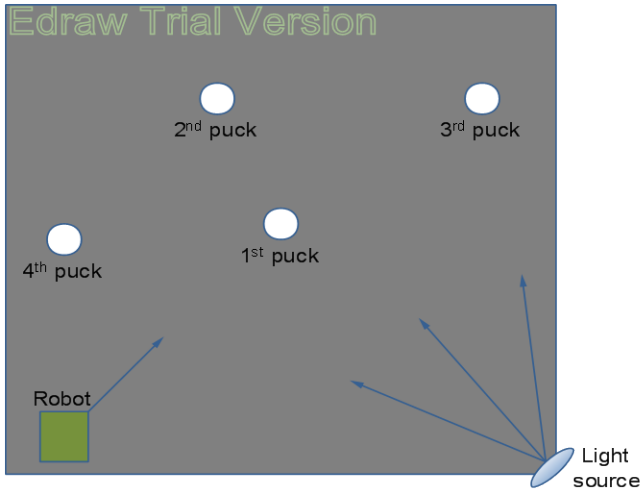


Figure 8.26 Positions of the four objects on the field relative to the robot.

8.5.1 First Attempt

These results were recorded when the robot was able to grab the objects at the first attempt of detection for all the four objects.

Table 8.16 Time taken for the robot to transport four objects at First Attempt.

No. of Trials	Time Taken (sec)
1	241
2	258
3	251
4	254
5	244
6	248
7	253
8	250
9	249
10	252

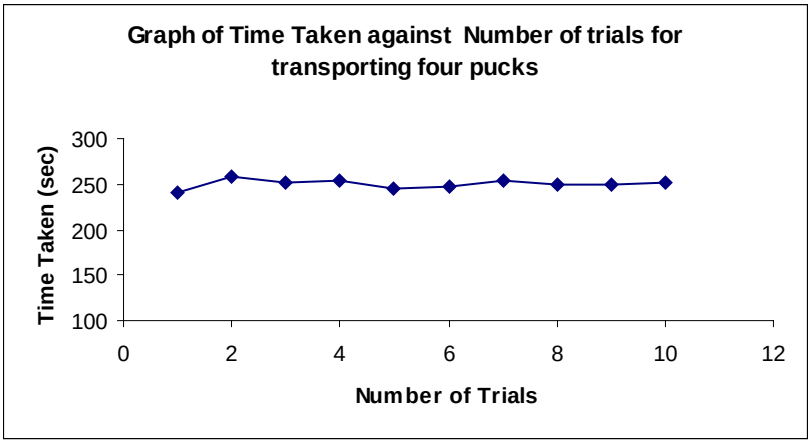


Figure 8.27 Graph of the time taken to transport four objects at first Attempt.

8.5.2 Second Attempt

These results were obtained when the robot was able to grab all the objects at the second attempt, i.e. the objects were removed from their places when the robot first detected them. They were placed back when the robot failed at the first attempt.

Table 8.17 Time taken for the robot to transport four objects at second Attempt.

No. of Trials	Time Taken (sec)
1	262
2	267
3	273
4	268
5	262
6	265
7	259
8	258
9	267
10	263

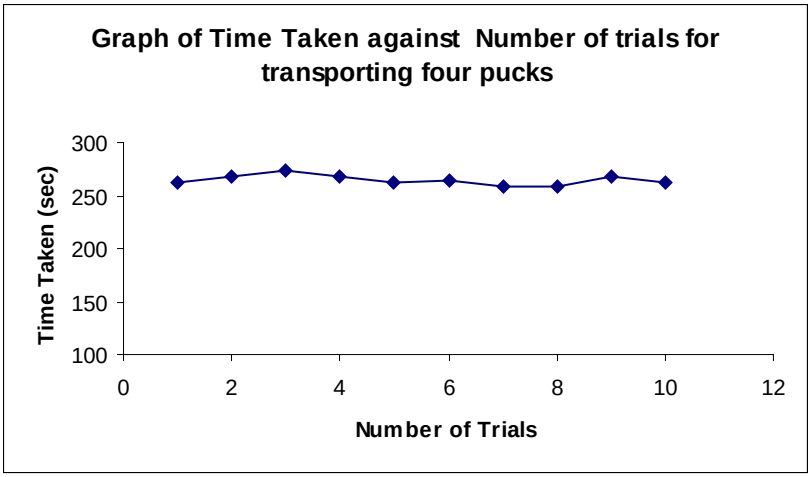


Figure 8.28 Graph of the time taken to transport four objects at second Attempt.

8.5.3 Third Attempt

These results were obtained when the robot was able to pick up the objects at the third attempt, i.e. the objects were removed from their places when the robot first detected them and this was repeated for the second attempt. The objects were placed back when the robot was unsuccessful at the first two attempts.

Table 8.18 Time taken for the robot to transport four objects at third Attempt.

No. of Trials	Time Taken (sec)
1	273
2	282
3	273
4	279
5	273
6	274
7	281
8	276
9	274
10	280

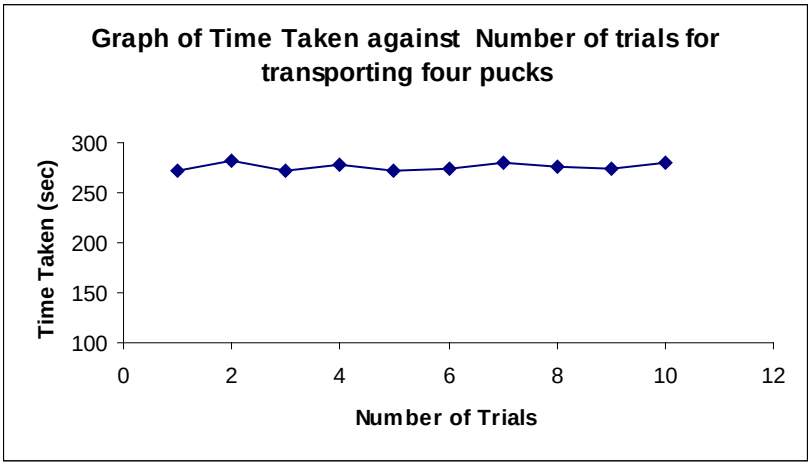


Figure 8.29 Graph of the time taken to transport four objects at third Attempt.

8.5.4 Comparison of the Three Attempts

To show the difference between the time taken at the various attempts, a comparison graph was plotted with all the three attempts for the four objects.

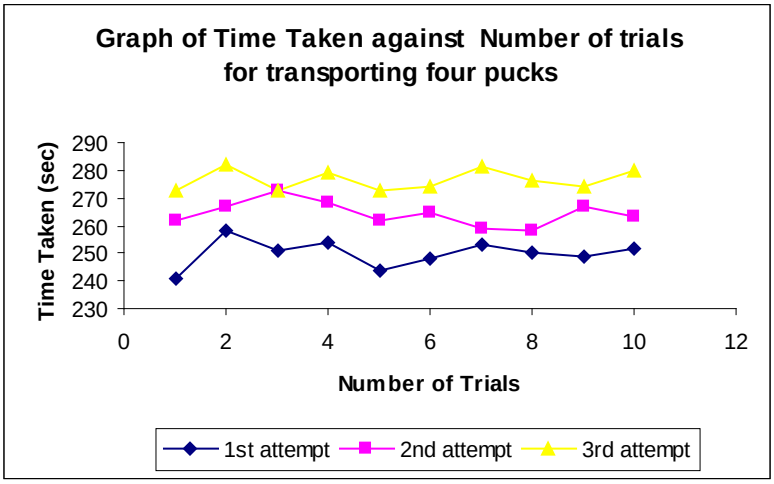


Figure 8.30 Graph showing the comparison of the results of four objects for three attempts.

8.6 Test for the Time Taken to Detect and Transport Five Objects.

The final test was then done with five objects on the field. The positions of the objects were again randomly selected. The positions of the pucks were as shown below.

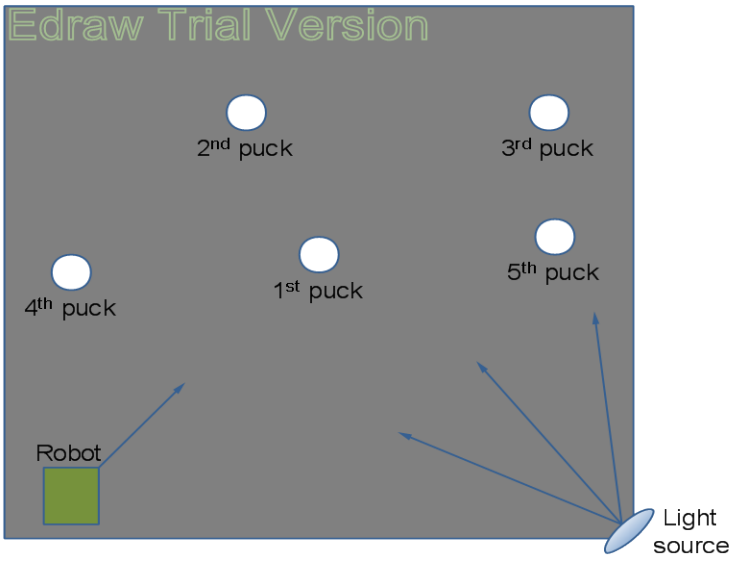


Figure 8.31 Positions of the five objects on the field relative to the robot.

8.6.1 First attempt

These results were obtained when the robot was able to grab the objects at the first attempt of detection for all five of the objects.

Table 8.19 Time taken for the robot to transport five objects at First Attempt.

No. of Trials	Time Taken (sec)
1	278
2	266
3	270
4	276
5	281
6	276
7	277
8	280
9	279
10	273

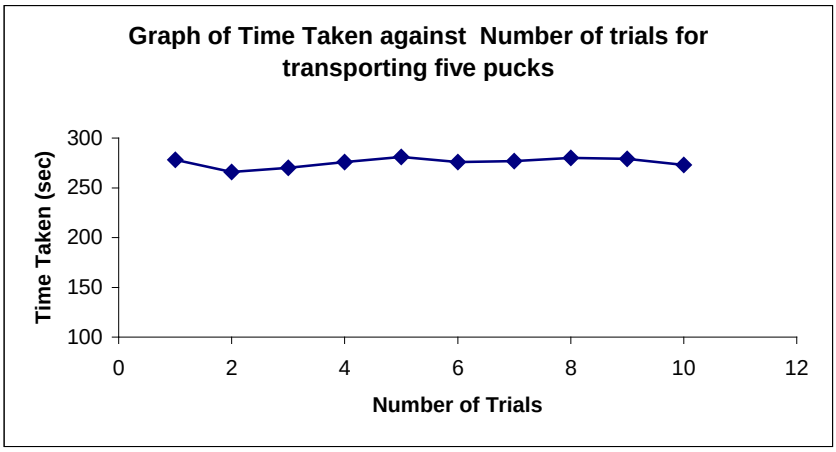


Figure 8.32 Graph of the time taken to transport five objects at first Attempt.

8.6.2 Second Attempt

These results were obtained when the robot was able to grab all the objects at the second attempt, i.e. the objects were removed from their places when the robot first detected them. They were placed back when the robot failed at the first attempt.

Table 8.20 Time taken for the robot to transport five objects at second Attempt.

No. of Trials	Time Taken (sec)
1	281
2	284
3	280
4	279
5	289
6	282
7	287
8	289
9	286
10	279

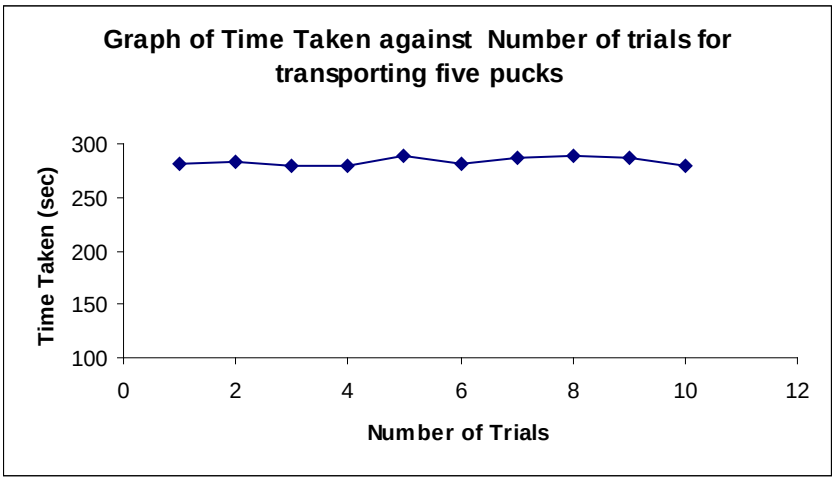


Figure 8.33 Graph of the time taken to transport five objects at second Attempt.

8.6.3 Third Attempt

These results were obtained when the robot was allowed to pick up the objects at the third attempt, i.e. the objects were removed from their places when the robot first detected them and this was repeated for the second attempt. The objects were placed back when the robot was unsuccessful at the first two attempts.

Table 8.21 Time taken for the robot to transport five objects at third Attempt.

No. of Trials	Time Taken (sec)
1	298
2	297
3	308
4	296
5	309
6	302
7	304
8	299
9	306
10	300

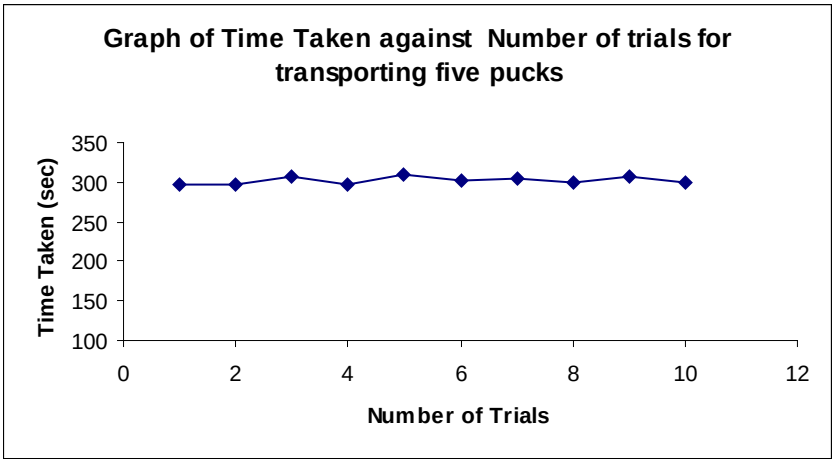


Figure 8.34 Graph of the time taken to transport five objects at third Attempt.

8.6.4 Comparison of the Three Attempts

Again to show the difference between the time taken for the various attempts, a comparison graph was plotted with all the three attempts for the final five objects.

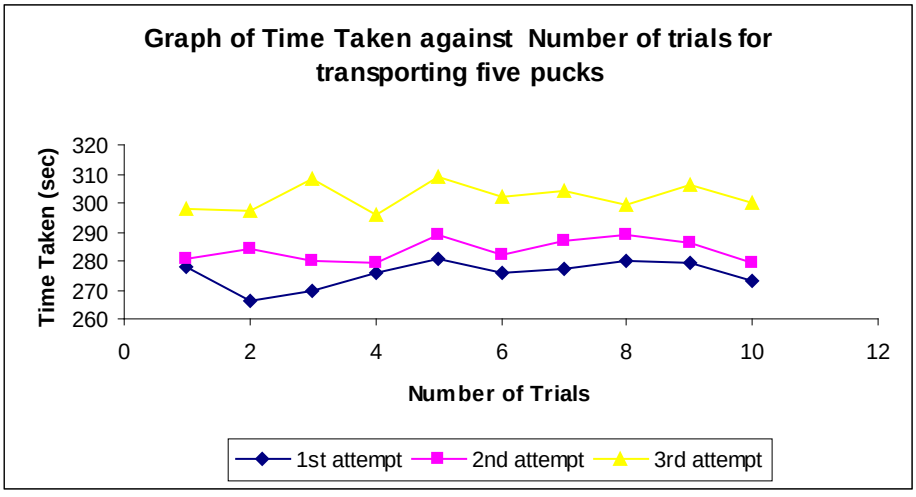


Figure 8.35 Graph showing the comparison of the results of five objects for three attempts.

8.7 Overall Performance

Looking at all the results given above, it can be said that the robot is quite efficient in performing its task. This can be said to be true because the difference between the times of the ten trials is not much. It is not that the robot is consistent throughout the test trials but from the graphs it can be noticed that there is not much variations. A better performance of the robot can be obtained with improvements in the programming. Sometimes the variations are due to the presence of brighter light from the environment which causes difficulties in object detection and light source detection. Finally, it can be said that the VEX Robotic Kit can be quite helpful for research work or for introduction to robotics for students.

Chapter 9

Conclusion and Recommendations

The behaviour- based approach was successfully used to autonomously control a mobile robot. The robot was assembled from the VEX Design Kit, which was bought for research work in the previous year but was not worked on.

The major part of this project was the programming of the robot to make it perform the required task. The EasyC Programming Kit which was specifically designed for the VEX microcontroller, was used for the programming. Since this was a new software, its usage had to be learned first before starting with the final programming. The task required for the project was to detect objects on the robot field and then transport them to a goal location represented by a light source. This task was successfully completed for five objects.

A variety of sensors, such as the ultrasonic sensor, line tracing sensor, light sensor and digital sensors were used to complete the task. The different sensors were really helpful in their respective areas. Some mechanical modifications were also made to the original Squarebot to get it to be able to perform the tasks properly. One major modification was the implementation of the claw for object handling. This idea was taken from another robotic kit from VEX which was named the VEXplorer.

Some problems were also faced while trying to complete the project. These were basically with getting the sensors to work properly and also with the control of the motors. A lot of tests had to be carried out to set the timings for each operation. Another problem was the usage of the right light source. Several types of lights were used when testing and finally the fluorescent lamp was used because of the even distribution of light.

Despite these problems, the Behavior-Based Control over the robot was quite easily implemented. Dividing up the task into simpler tasks made the development of the individual behaviors quite simple to manage and implement. In addition, once the key components of the system were in place, it was easy to design and debug each individual behavior by isolating it and examining its actions in various situations.

Even though, the primary objective of this project which was the use of different behaviors and a variety of sensors to implement the foraging task of locating objects and transporting them to a goal location was accomplished, there is some further work that can be carried out with this kit to make it more efficient. Some of the improvements, which can be made are:

- Firstly, the software (codes) can be further developed so that robot is able to detect and stop at the precise values of the light detected for the goal location. This would also help the robot to place the objects at the correct positions.

- Secondly, the applications of the sensors can be changed to give more behaviours to the robot. At the moment, object detection is being done with line tracing sensors. The light sensor or the ultrasonic sensor can also be used for this. If bigger objects were used, the best sensor to use would be the ultrasonic ones.

- Furthermore, a battery should be used to power up the robot, which can hold charge for long hours and can be recharged quickly. A rechargeable battery is available from Innovation First, Inc., which comes with the charger.

- Finally, further modification can be made with the mechanical structure of the robot. The size of the robot can be increased with the use of bigger chassis rails available with the Kit. While increasing the structure size, the wheel size can also be increased to give more speed to the robot.

References

[1] McClain, J. T. A Behavior- Based Blackboard Architecture for Multi-Robot Control. *MSc Thesis*, The University of Georgia, Athens, Georgia, 2004.

[2] Arkin, R. C. *Behavior-Based Robotics*. Cambridge, MA: The MIT Press. 1998.

[3] Brooks, R. A. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, 1986.

[4] Woolley, B. and Woolley, E. Vex Robotics Design System, *The Vex Epic*, 2005.

[5] VEX Robotics by Innovation First, Inc. [Online], Available: http://www.studica.com/VEXRobotics/, 02/09/08.

[6] Murphy, T. The Hitchhikers' Vexperience, *VEX Robotics Design System Bulletin Board.* [online], Available: http://client.methodsofmadness.com/radioshack/0905/email/, 05/09/08.

[7] Products, *VEX Robotics Design System.* [online], Available: http://www.vexrobotics.com, 02/09/08.

[8] Microcontroller, [online], Available: http://en.wikipedia.org/wiki/Microcontroller, 05/09/08.

[9] Mack, G. Vex Robotics, Controller Brain, [online], Available: http://www.robotgroup.net/index.cgi/VexRobotics, 02/09/08.

[10] Microchip, PIC18FXX20 Data Sheet: 64/80-Pin High-Performance, 256 Kbit to 1 Mbit Enhanced Flash Microcontrollers with A/D, *Technical Reference Document DS39609B*, Microchip Technology Inc., 1994.

[11] Sweeney, L. *Definition of Artificial Intelligence.* [online], Available: http://privacy.cs.cmu.edu/people/sweeney/aidef.html, 05/09/08.

[12] Brown, N. C. C. *The History of AI.* [online], Available: http://www.generation5.org/aihistory.shtml, 02/09/08.

[13] Artificial Intelligence versus Classical Robotics, *Robot Control Architectures.* [online], Available: http://web.cecs.pdx.edu/~mperkows/CLASS_479/hardware/014.Robot-Control-Architectures.ppt, 05/09/08.

[14] Building Intelligent Robots, *Autonomous Control Architectures*. [online], Available : http://www.cs.brown.edu/courses/cs148/old/2004fall/lecture_slides/cs148_4.pdf, 02/09/08.

[15] Bauman, R.P. *Absorption spectroscopy*; John Wiley & Sons, New York, 1962; pp. 72-78.

[16] Stracey, M. The Sensors, *Line Sensor*. [online], Available: http://www.mstracey.btinternet.co.uk/technical/Theory/theorysensors.htm, 15/10/08.

[17] The Sensor Subsystem. [online], Available: http://www.complexity.org.au/ci_louise/vol03/gaudi2/node6.html, 15/10/08.

[18] Hertzberg, J. and Schonherr, F. *Plan-Based Control of Autonomous Robots.* [online], Available: http://www.ercim.org/publication/Ercim_News/enw42/hertzberg.html, 05/09/08.

[19] Putatunda, R. *Types of Robots*. [online], Available: http://www.buzzle.com/articles/types-of-robots.html, 15/10/08.

Appendix A: Bill of Materials

Bill of Materials				
Item	**Qty**	**Unit Price**	**Total**	**Supplier**
Electronics				
VEX Bumper Switch Kit	1	12.99	12.99	Innovation First. Inc
VEX Light Sensor	2	19.99	39.98	Innovation First. Inc
VEX Limit Switch Kit	1	12.99	12.99	Innovation First. Inc
VEX Line Tracking Kit	1	39.99	39.99	Innovation First. Inc
VEX Motor Kit	1	19.99	19.99	Innovation First. Inc
VEX Optical Shaft Encoder	1	19.99	19.99	Innovation First. Inc
VEX Ultrasonic Range Finder	2	29.99	49.98	Innovation First. Inc
PWM 12" Extension Cable 4pk	1	19.95	19.95	Innovation First. Inc
Energizer Battery AA 1.5Volts 4pk	5	7.56	37.8	Vinod Patel
"AA" size DSE NiMH Battery Charger	1	49.99	49.99	Dicksmith NZ
"AA" size DSE Rechargeable Batteries 4pk	1	19.99	19.99	Dicksmith NZ
Total Electronics cost			***323.64***	
Hardware				
VEX Robotic Design System Kit	1	299.99	299.99	SEP-In-House
VEX Claw Kit	1	19.99	19.99	Innovation First. Inc
VEXplorer Wrist Kit	1	19.99	19.99	Innovation First. Inc
VEX Gear Kit	1	12.99	12.99	Innovation First. Inc
Pine Run of Mill H3 "6*1"	2	19.80	39.60	Vinod Patel
Exterior Plywood 2400mm x 1200mm x 9mm	2	60.84	121.68	SEP-In-House
Total Hardware cost			***514.24***	
TOTAL COST			**837.88**	

Appendix B: Final C Codes

```
#include "Main.h"

void main ( void )
{
    int loop = 1;
    int left_limit_switch;
    int right_limit_switch;
    int claw_limit_switch;
    int front_right_bumper;
    int rear_right_bumper;
    int front_left_bumper;
    int rear_left_bumper;
    int right_line_tracker;
    int left_line_tracker;
    int centre_line_tracker;
    int light_sensor;
    int ultrasonic_range_finder;

    while ( loop == 1 )
    {
        // The robot moving forward
        // The motor number 2 represent the left motor for the whole program
        // The motor number 1 represent the right motor for the whole program
        SetMotor ( 2 , 155 ) ; // Left motor
        SetMotor ( 1 , 100 ) ; // Right motor
        left_limit_switch = GetDigitalInput ( 1 ) ;
        PrintToScreen ( "left_limit_switch =%d\n" , (int)left_limit_switch ) ;
        // If right_limit_switch is pressed
        // switch open = 1, switch closed = 0
        if ( left_limit_switch == 0 )
        {
            SetMotor ( 1 , 170 ) ;
            SetMotor ( 2 , 170 ) ;
            Wait ( 600 ) ;
        }
        // If left_limit_switch  is pressed
        // switch open = 1, switch closed = 0
        right_limit_switch = GetDigitalInput ( 2 ) ;
        PrintToScreen ( "right_limit_switch = %d\n" , (int)right_limit_switch ) ;
        if ( right_limit_switch == 0 )
        {
            SetMotor ( 1 , 85 ) ;
            SetMotor ( 2 , 85 ) ;
```

```
    Wait ( 600 ) ;
}
// Line trackers represent object detection
// Left & Right_line_tracker will aline the puck to centre_line_tracker
// Light = 900, Dark = 1000
right_line_tracker = GetAnalogInput ( 9 ) ;
PrintToScreen ( "right_line_tracker =%d\n" , (int)right_line_tracker ) ;
if ( right_line_tracker <= 900 )
{
    SetMotor ( 1 , 160 ) ;
    SetMotor ( 2 , 95 ) ;
    Wait ( 1000 ) ;
    SetMotor ( 2 , 170 ) ;
    SetMotor ( 1 , 170 ) ;
    Wait ( 150 ) ;
}
// Light = 900, Dark = 1000
left_line_tracker = GetAnalogInput ( 10 ) ;
PrintToScreen ( "left_line_tracker =%d\n" , (int)left_line_tracker ) ;
if ( left_line_tracker <= 900 )
{
    SetMotor ( 1 , 160 ) ;
    SetMotor ( 2 , 95 ) ;
    Wait ( 1000 ) ;
    SetMotor ( 2 , 85 ) ;
    SetMotor ( 1 , 85 ) ;
    Wait ( 150 ) ;
}
// if the robot is going to the light source without picking any  puck
light_sensor = GetAnalogInput ( 8 ) ;
PrintToScreen ( "light_sensor =%d\n" , (int)light_sensor ) ;
if ( light_sensor <= 25 )
{
    SetMotor ( 1 , 160 ) ;
    SetMotor ( 2 , 95 ) ;
    Wait ( 1500 ) ;
    SetMotor ( 2 , 170 ) ;
    SetMotor ( 1 , 170 ) ;
    Wait ( 900 ) ;
}
// Switch open = 1,switch closed = 0; for all bumper switches
// Avoides the robot from getting trapped along the boundary
// The robot hits the wall then it moves back
// rotates and move in the other direction
front_right_bumper = GetDigitalInput ( 3 ) ;
PrintToScreen ( "front_right_bumper = %d\n" , (int)front_right_bumper ) ;
```

```
if ( front_right_bumper == 0 )
{
    SetMotor ( 1 , 160 ) ;
    SetMotor ( 2 , 95 ) ;
    Wait ( 2000 ) ;
    // while move back the rear_right_bumper is pressed
    rear_right_bumper = GetDigitalInput ( 4 ) ;
    PrintToScreen ( "rear_right_bumper = %d\n" , (int)rear_right_bumper ) ;
    if ( rear_right_bumper  == 0 )
    {
        SetMotor ( 1 , 127 ) ;
        SetMotor ( 2 , 127 ) ;
        Wait ( 2000 ) ;
        SetMotor ( 2 , 160 ) ;
        SetMotor ( 1 , 95 ) ;
        Wait ( 1000 ) ;
    }
    // while move back the rear_left_bumper is pressed
    rear_left_bumper = GetDigitalInput ( 6 ) ;
    PrintToScreen ( "rear_left_bumper = %d\n" , (int)rear_left_bumper ) ;
    if ( rear_left_bumper == 0 )
    {
        SetMotor ( 1 , 127 ) ;
        SetMotor ( 2 , 127 ) ;
        Wait ( 2000 ) ;
        SetMotor ( 2 , 160 ) ;
        SetMotor ( 1 , 95 ) ;
        Wait ( 1000 ) ;
    }
    SetMotor ( 2 , 170 ) ;
    SetMotor ( 1 , 170 ) ;
    Wait ( 900 ) ;
}
// Switch open = 1, switch closed = 0
front_left_bumper = GetDigitalInput ( 5 ) ;
PrintToScreen ( "front_left_bumper = %d\n" , (int)front_left_bumper ) ;
if ( front_left_bumper == 0 )
{
    SetMotor ( 1 , 160 ) ;
    SetMotor ( 2 , 95 ) ;
    Wait ( 2000 ) ;
    // while moving back the rear_right_bumper is pressed
    rear_right_bumper = GetDigitalInput ( 4 ) ;
    PrintToScreen ( "rear_right_bumper = %d\n" , (int)rear_right_bumper ) ;
    if ( rear_right_bumper  == 0 )
    {
```

```
            SetMotor ( 1 , 127 ) ;
            SetMotor ( 2 , 127 ) ;
            Wait ( 2000 ) ;
            SetMotor ( 2 , 160 ) ;
            SetMotor ( 1 , 95 ) ;
            Wait ( 1000 ) ;
        }
        // while moving back the rear_left_bumper is pressed
        rear_left_bumper = GetDigitalInput ( 6 ) ;
        PrintToScreen ( "rear_left_bumper = %d\n" , (int)rear_left_bumper ) ;
        if ( rear_left_bumper  == 0 )
        {
            SetMotor ( 1 , 127 ) ;
            SetMotor ( 2 , 127 ) ;
            Wait ( 2000 ) ;
            SetMotor ( 2 , 160 ) ;
            SetMotor ( 1 , 95 ) ;
            Wait ( 1000 ) ;
        }
        SetMotor ( 2 , 170 ) ;
        SetMotor ( 1 , 170 ) ;
        Wait ( 900 ) ;
    }
    // Light = 920, Dark = 1000
    // If the centre_line_tracker detects a puck, it will pick up the puck using the claw
    centre_line_tracker = GetAnalogInput ( 7 ) ;
    PrintToScreen ( "centre_line_tracker = %d\n" , (int)centre_line_tracker ) ;
    if ( centre_line_tracker  <= 990 )
    {
        SetMotor ( 2 , 127 ) ;
        SetMotor ( 1 , 127 ) ;
        Wait ( 1000 ) ;
        SetMotor ( 2 , 95 ) ;
        SetMotor ( 1 , 160 ) ;
        Wait ( 760 ) ;
        SetMotor ( 2 , 127 ) ;
        SetMotor ( 1 , 127 ) ;
        Wait ( 300 ) ;
        SetServo ( 3 , 0 ) ;
        Wait ( 500 ) ;
        SetMotor ( 4 , 85 ) ; // Arm motor used to move the claw up and down
        Wait ( 525 ) ;
        SetMotor ( 4 , 127 ) ;
        Wait ( 500 ) ;
        SetServo ( 3 , 255 ) ; // Claw servo motor used for the claw opening and
            closing
```

```
Wait ( 1000 ) ;
SetMotor ( 4 , 180 ) ;
Wait ( 900 ) ;
SetMotor ( 4 , 127 ) ;
Wait ( 500 ) ;
// This limit switch is used in case the claw is unsuccessful in
//  picking the puck, limit == 0 means the claw was able to pick up the puck
// If limit == 1 that mean it was unsuccessful and it will follow the while
// below.
claw_limit_switch = GetDigitalInput ( 12 ) ;
PrintToScreen ( "claw_limit_switch =%d\n" , (int)claw_limit_switch ) ;
while ( claw_limit_switch == 1 )
{
    claw_limit_switch = GetDigitalInput ( 12 ) ;
    PrintToScreen ( "claw_limit_switch =%d\n" , (int)claw_limit_switch ) ;
    if ( claw_limit_switch == 1 )
    {
        SetServo ( 3 , 0 ) ;
        Wait ( 500 ) ;
        SetMotor ( 2 , 150 ) ;
        SetMotor ( 1 , 105 ) ;
        centre_line_tracker = GetAnalogInput ( 7 ) ;
        PrintToScreen ( "centre_line_tracker = %d\n" ,
        (int)centre_line_tracker ) ;
        if ( centre_line_tracker  <=  990 )
        {
            centre_line_tracker = GetAnalogInput ( 7 ) ;
            PrintToScreen ( "centre_line_tracker = %d\n" ,
            (int)centre_line_tracker ) ;
            while ( centre_line_tracker  <= 990 )
            {
                if ( centre_line_tracker  <= 990 )
                {
                    SetMotor ( 2 , 127 ) ;
                    SetMotor ( 1 , 127 ) ;
                    Wait ( 1000 ) ;
                    SetMotor ( 2 , 95 ) ;
                    SetMotor ( 1 , 160 ) ;
                    Wait ( 725 ) ;
                    SetMotor ( 2 , 127 ) ;
                    SetMotor ( 1 , 127 ) ;
                    Wait ( 500 ) ;
                    SetMotor ( 4 , 85 ) ;
                    Wait ( 525 ) ;
                    SetMotor ( 4 , 127 ) ;
                    Wait ( 500 ) ;
```

```
                SetServo ( 3 , 255 ) ;
                Wait ( 1000 ) ;
                SetMotor ( 4 , 180 ) ;
                Wait ( 900 ) ;
                SetMotor ( 4 , 127 ) ;
                Wait ( 1000 ) ;
            }
            break ;
        }
}
// When the centre_line_tracker is not able the detect the puck and the
// robot move forward searching for others pucks, the front_bumpers
//  and side_line_trackers will guide the robot.
// The front_bumpers will be activated when the robot hits the wall,
// thus the robot will move back and rotate 90 - 100 degrees.
front_right_bumper = GetDigitalInput ( 3 ) ;
PrintToScreen ( "front_right_bumper =%d\n" ,
(int)front_right_bumper ) ;
if ( front_right_bumper == 0 )
{
    SetMotor ( 2 , 95 ) ;
    SetMotor ( 1 , 160 ) ;
    Wait ( 1200 ) ;
    SetMotor ( 2 , 85 ) ;
    SetMotor ( 1 , 85 ) ;
    Wait ( 1200 ) ;
}
front_left_bumper = GetDigitalInput ( 5 ) ;
PrintToScreen ( "front_left_bumper =%d\n" , (int)front_left_bumper );
if ( front_left_bumper == 0 )
{
    SetMotor ( 2 , 95 ) ;
    SetMotor ( 1 , 160 ) ;
    Wait ( 1200 ) ;
    SetMotor ( 2 , 170 ) ;
    SetMotor ( 1 , 170 ) ;
    Wait ( 1200 ) ;
}
// When a puck comes under the wheel of the robot, line_trackers
//  will help in alining the puck with the centre_line_tracker .
right_line_tracker = GetAnalogInput ( 9 ) ;
PrintToScreen ( "right_line_tracker =%d\n" , (int)right_line_tracker ) ;
if ( right_line_tracker <= 900 )
{
    SetMotor ( 1 , 160 ) ;
    SetMotor ( 2 , 95 ) ;
```

```
                Wait ( 1000 ) ;
                SetMotor ( 2 , 160 ) ;
                SetMotor ( 1 , 160 ) ;
                Wait ( 20 ) ;
            }
            left_line_tracker = GetAnalogInput ( 10 ) ;
            PrintToScreen ( "left_line_tracker =%d\n" , (int)left_line_tracker ) ;
            if ( left_line_tracker <= 900 )
            {
                SetMotor ( 1 , 160 ) ;
                SetMotor ( 2 , 95 ) ;
                Wait ( 1000 ) ;
                SetMotor ( 2 , 100 ) ;
                SetMotor ( 1 , 100 ) ;
                Wait ( 20 ) ;
            }
            // if the robot is going to the light source without picking any  puck
            light_sensor = GetAnalogInput ( 8 ) ;
            PrintToScreen ( "light_sensor =%d\n" , (int)light_sensor ) ;
            if ( light_sensor <= 25 )
            {
                SetMotor ( 1 , 160 ) ;
                SetMotor ( 2 , 95 ) ;
                Wait ( 1500 ) ;
                SetMotor ( 1 , 170 ) ;
                SetMotor ( 2 , 170 ) ;
                Wait ( 900 ) ;
            }
        }
        else
        {
            break ;
        }
    }
    // light = 40, dark = 1000
    // The light source is our goal where the robot will transport the pucks to.
    // Thus a light sensor is used to detect the source for transportation.
    while ( 4 )
    {
        light_sensor = GetAnalogInput ( 8 ) ;
        PrintToScreen ( "light_sensor = %d\n" , (int)light_sensor ) ;
        if ( light_sensor  >= 40 )
        {
            SetMotor ( 2 , 170 ) ;
            SetMotor ( 1 , 170 ) ;
            light_sensor = GetAnalogInput ( 8 ) ;
```

```
PrintToScreen ( "light_sensor = %d\n" , (int)light_sensor ) ;
if ( light_sensor <= 85 &&  light_sensor >= 80 )
{
    SetMotor ( 2 , 160 ) ;
    SetMotor ( 1 , 95 ) ;
    Wait ( 3000 ) ;
    // Will be helpful if any puck comes under the robots wheel
    // Then Line_trackers will change the robots direction of
    // movement
    right_line_tracker = GetAnalogInput ( 9 ) ;
    PrintToScreen ( "right_line_tracker =%d\n" ,
    (int)right_line_tracker ) ;
    if ( right_line_tracker <= 900 )
    {
        SetMotor ( 1 , 160 ) ;
        SetMotor ( 2 , 95 ) ;
        Wait ( 1000 ) ;
        SetMotor ( 2 , 85 ) ;
        SetMotor ( 1 , 85 ) ;
        Wait ( 100 ) ;
    }
    left_line_tracker = GetAnalogInput ( 10 ) ;
    PrintToScreen ( "left_line_tracker =%d\n" , (int)left_line_tracker );
    if ( left_line_tracker <= 900 )
    {
        SetMotor ( 1 , 160 ) ;
        SetMotor ( 2 , 95 ) ;
        Wait ( 1000 ) ;
        SetMotor ( 2 , 170 ) ;
        SetMotor ( 1 , 170 ) ;
        Wait ( 100 ) ;
    }
}
light_sensor = GetAnalogInput ( 8 ) ;
PrintToScreen ( "light_sensor = %d\n" , (int)light_sensor ) ;
if ( light_sensor <= 72 &&  light_sensor >= 68 )
{
    SetMotor ( 2 , 160 ) ;
    SetMotor ( 1 , 95 ) ;
    Wait ( 2500 ) ;
    right_line_tracker = GetAnalogInput ( 9 ) ;
    PrintToScreen ( "right_line_tracker =%d\n" ,
    (int)right_line_tracker ) ;
    if ( right_line_tracker <= 900 )
    {
        SetMotor ( 1 , 160 ) ;
```

```
            SetMotor ( 2 , 95 ) ;
            Wait ( 1000 ) ;
            SetMotor ( 2 , 85 ) ;
            SetMotor ( 1 , 85 ) ;
            Wait ( 100 ) ;
        }
        left_line_tracker = GetAnalogInput ( 10 ) ;
        PrintToScreen ( "left_line_tracker =%d\n" , (int)left_line_tracker );
        if ( left_line_tracker <= 900 )
        {
            SetMotor ( 1 , 160 ) ;
            SetMotor ( 2 , 95 ) ;
            Wait ( 1000 ) ;
            SetMotor ( 2 , 170 ) ;
            SetMotor ( 1 , 170 ) ;
            Wait ( 100 ) ;
        }
    }
    light_sensor = GetAnalogInput ( 8 ) ;
    PrintToScreen ( "light_sensor = %d\n" , (int)light_sensor ) ;
    if ( light_sensor <= 67 &&  light_sensor >= 60 )
    {
        SetMotor ( 2 , 160 ) ;
        SetMotor ( 1 , 95 ) ;
        Wait ( 1200 ) ;
    }
    light_sensor = GetAnalogInput ( 8 ) ;
    PrintToScreen ( "light_sensor = %d\n" , (int)light_sensor ) ;
    if ( light_sensor <= 59 &&  light_sensor >= 53 )
    {
        SetMotor ( 2 , 160 ) ;
        SetMotor ( 1 , 95 ) ;
        Wait ( 1000 ) ;
    }
    light_sensor = GetAnalogInput ( 8 ) ;
    PrintToScreen ( "light_sensor = %d\n" , (int)light_sensor ) ;
    if ( light_sensor <= 52 &&  light_sensor >= 45 )
    {
        SetMotor ( 2 , 160 ) ;
        SetMotor ( 1 , 95 ) ;
        Wait ( 500 ) ;
    }
    // If the puck falls down while the robot is searching for the light
    // source then the robot will search for the puck again.
    claw_limit_switch = GetDigitalInput ( 12 ) ;
    PrintToScreen ( "claw_limit_switch =%d\n" , (int)claw_limit_switch );
```

```
while ( claw_limit_switch == 1 )
{
    claw_limit_switch = GetDigitalInput ( 12 ) ;
    PrintToScreen ( "claw_limit_switch =%d\n" ,
    (int)claw_limit_switch ) ;
    if ( claw_limit_switch == 1 )
    {
        SetServo ( 3 , 0 ) ;
        Wait ( 500 ) ;
        SetMotor ( 2 , 150 ) ;
        SetMotor ( 1 , 105 ) ;
        centre_line_tracker = GetAnalogInput ( 7 ) ;
        PrintToScreen ( "centre_line_tracker = %d\n" ,
            (int)centre_line_tracker ) ;
        if ( centre_line_tracker <=  990 )
        {
            centre_line_tracker = GetAnalogInput ( 7 ) ;
            PrintToScreen ( "centre_line_tracker = %d\n" ,
            (int)centre_line_tracker ) ;
            while ( centre_line_tracker <= 990 )
            {
                if ( centre_line_tracker <= 990 )
                {
                    SetMotor ( 2 , 127 ) ;
                    SetMotor ( 1 , 127 ) ;
                    Wait ( 500 ) ;
                    SetMotor ( 2 , 95 ) ;
                    SetMotor ( 1 , 160 ) ;
                    Wait ( 750 ) ;
                    SetMotor ( 2 , 127 ) ;
                    SetMotor ( 1 , 127 ) ;
                    Wait ( 500 ) ;
                    SetMotor ( 4 , 85 ) ;
                    Wait ( 525 ) ;
                    SetMotor ( 4 , 127 ) ;
                    Wait ( 500 ) ;
                    SetServo ( 3 , 255 ) ;
                    Wait ( 2000 ) ;
                    SetMotor ( 4 , 180 ) ;
                    Wait ( 900 ) ;
                    SetMotor ( 4 , 127 ) ;
                    Wait ( 500 ) ;
                }
                break ;
            }
        }
```

```
front_right_bumper = GetDigitalInput ( 3 ) ;
PrintToScreen ( "front_right_bumper = %d\n" ,
    (int)front_right_bumper ) ;
if ( front_right_bumper == 0 )
{
    SetMotor ( 2 , 95 ) ;
    SetMotor ( 1 , 160 ) ;
    Wait ( 2000 ) ;
    SetMotor ( 2 , 85 ) ;
    SetMotor ( 1 , 85 ) ;
    Wait ( 1200 ) ;
}
// The front_bumpers and line_trackers will again guide the
// robot in locating the pucks
front_left_bumper = GetDigitalInput ( 5 ) ;
PrintToScreen ( "front_left_bumper = %d\n" ,
    (int)front_left_bumper ) ;
if ( front_left_bumper == 0 )
{
    SetMotor ( 2 , 95 ) ;
    SetMotor ( 1 , 160 ) ;
    Wait ( 2000 ) ;
    SetMotor ( 2 , 170 ) ;
    SetMotor ( 1 , 170 ) ;
    Wait ( 1200 ) ;
}
right_line_tracker = GetAnalogInput ( 9 ) ;
PrintToScreen ( "right_line_tracker =%d\n" ,
(int)right_line_tracker ) ;
if ( right_line_tracker <= 900 )
{
    SetMotor ( 2 , 95 ) ;
    SetMotor ( 1 , 160 ) ;
    Wait ( 1000 ) ;
    SetMotor ( 2 , 170 ) ;
    SetMotor ( 1 , 170 ) ;
    Wait ( 150 ) ;
}
left_line_tracker = GetAnalogInput ( 10 ) ;
PrintToScreen ( "left_line_tracker =%d\n" ,
(int)left_line_tracker ) ;
if ( left_line_tracker  <= 900 )
{
    SetMotor ( 2 , 95 ) ;
    SetMotor ( 1 , 160 ) ;
    Wait ( 1000 ) ;
```

```
                SetMotor ( 2 , 85 ) ;
                SetMotor ( 1 , 85 ) ;
                Wait ( 150 ) ;
            }
            // if the robot is going to the light source without picking any
            // puck
            light_sensor = GetAnalogInput ( 8 ) ;
            PrintToScreen ( "light_sensor =%d\n" , (int)light_sensor ) ;
            if ( light_sensor <= 25 )
            {
                SetMotor ( 1 , 160 ) ;
                SetMotor ( 2 , 95 ) ;
                Wait ( 1500 ) ;
                SetMotor ( 1 , 170 ) ;
                SetMotor ( 2 , 170 ) ;
                Wait ( 1000 ) ;
            }
        }
        else
        {
            break ;
        }
    }
    // If the robot hits the wall when it's locating the light source then
    // bumpers, limiters, will help the robot to move away for the wall.
    front_right_bumper = GetDigitalInput ( 3 ) ;
    PrintToScreen ( "front_right_bumper =%d\n" ,
    (int)front_right_bumper ) ;
    if ( front_right_bumper == 0 )
    {
        SetMotor ( 2 , 95 ) ;
        SetMotor ( 1 , 160 ) ;
        Wait ( 2000 ) ;
        SetMotor ( 2 , 85 ) ;
        SetMotor ( 1 , 85 ) ;
        Wait ( 700 ) ;
    }
    rear_right_bumper = GetDigitalInput ( 4 ) ;
PrintToScreen ( "rear_right_bumper =%d\n" , (int)rear_right_bumper );
    if ( rear_right_bumper == 0 )
    {
        SetMotor ( 2 , 160 ) ;
        SetMotor ( 1 , 95 ) ;
        Wait ( 2000 ) ;
        SetMotor ( 2 , 170 ) ;
        SetMotor ( 1 , 170 ) ;
```

```
        Wait ( 700 ) ;
    }
    front_left_bumper = GetDigitalInput ( 5 ) ;
    PrintToScreen ( "front_left_bumper=%d\n" , (int)front_left_bumper ) ;
    if ( front_left_bumper == 0 )
    {
        SetMotor ( 2 , 127 ) ;
        SetMotor ( 1 , 127 ) ;
        Wait ( 1000 ) ;
        SetMotor ( 2 , 95 ) ;
        SetMotor ( 1 , 160 ) ;
        Wait ( 2000 ) ;
    }
    rear_left_bumper = GetDigitalInput ( 6 ) ;
    PrintToScreen ( "rear_left_bumper =%d\n" , (int)rear_left_bumper ) ;
    if ( rear_left_bumper  == 0 )
    {
        SetMotor ( 2 , 127 ) ;
        SetMotor ( 1 , 127 ) ;
        Wait ( 1000 ) ;
        SetMotor ( 2 , 160 ) ;
        SetMotor ( 1 , 95 ) ;
        Wait ( 2000 ) ;
    }
    left_limit_switch = GetDigitalInput ( 1 ) ;
    PrintToScreen ( "left_limit_switch =%d\n" , (int)left_limit_switch ) ;
    if ( left_limit_switch == 0 )
    {
        SetMotor ( 1 , 170 ) ;
        SetMotor ( 2 , 170 ) ;
        Wait ( 900 ) ;
        SetMotor ( 2 , 160 ) ;
        SetMotor ( 1 , 95 ) ;
        Wait ( 2000 ) ;
    }
    right_limit_switch = GetDigitalInput ( 2 ) ;
   PrintToScreen ( "right_limit_switch = %d\n" , (int)right_limit_switch );
    if ( right_limit_switch == 0 )
    {
        SetMotor ( 1 , 85 ) ;
        SetMotor ( 2 , 85 ) ;
        Wait ( 900 ) ;
        SetMotor ( 2 , 160 ) ;
        SetMotor ( 1 , 95 ) ;
        Wait ( 2000 ) ;
    }
```

```
        }
        else
        {
            SetMotor ( 2 , 127 ) ;
            SetMotor ( 1 , 127 ) ;
            Wait ( 500 ) ;
            break ;
        }
    }
    // Near = 2, Far = 100
    StartUltrasonic ( 1 , 11 ) ;
    while ( 5 )
    {
        ultrasonic_range_finder = GetUltrasonic ( 1 , 11 ) ;
        PrintToScreen ( "ultrasonic_range_finder =%d\n" ,
        (int)ultrasonic_range_finder ) ;
        if ( ultrasonic_range_finder >= 30 )
        {
            SetMotor ( 2 , 160 ) ;
            SetMotor ( 1 , 95 ) ;
            light_sensor = GetAnalogInput ( 8 ) ;
            PrintToScreen ( "light_sensor =%d\n" , (int)light_sensor ) ;
            if ( light_sensor >= 50 )
            {
                while ( 4 )
                {
                    light_sensor = GetAnalogInput ( 8 ) ;
                    PrintToScreen ( "light_sensor =%d\n" , (int)light_sensor ) ;
                    if ( light_sensor >= 30 )
                    {
                        SetMotor ( 2 , 170 ) ;
                        SetMotor ( 1 , 170 ) ;
                    }
                    else
                    {
                        SetMotor ( 2 , 127 ) ;
                        SetMotor ( 1 , 127 ) ;
                        Wait ( 1000 ) ;
                        break ;
                    }
                }
            }
            front_right_bumper = GetDigitalInput ( 3 ) ;
            PrintToScreen ( "front_right_bumper =%d\n" ,
             (int)front_right_bumper ) ;
            if ( front_right_bumper == 0 )
```

```
{
    SetMotor ( 2 , 95 ) ;
    SetMotor ( 1 , 160 ) ;
    Wait ( 1000 ) ;
    SetMotor ( 2 , 85 ) ;
    SetMotor ( 1 , 85 ) ;
    Wait ( 100 ) ;
}
front_left_bumper = GetDigitalInput ( 5 ) ;
PrintToScreen ( "front_left_bumper =%d\n" , (int)front_left_bumper );
if ( front_left_bumper  == 0 )
{
    SetMotor ( 2 , 95 ) ;
    SetMotor ( 1 , 160 ) ;
    Wait ( 1000 ) ;
    SetMotor ( 2 , 170 ) ;
    SetMotor ( 1 , 170 ) ;
    Wait ( 100 ) ;
}
right_line_tracker = GetAnalogInput ( 9 ) ;
PrintToScreen ( "right_line_tracker =%d\n" , (int)right_line_tracker ) ;
if ( right_line_tracker <= 900 )
{
    SetMotor ( 1 , 160 ) ;
    SetMotor ( 2 , 95 ) ;
    Wait ( 1000 ) ;
    SetMotor ( 2 , 85 ) ;
    SetMotor ( 1 , 85 ) ;
    Wait ( 100 ) ;
}
left_line_tracker = GetAnalogInput ( 10 ) ;
PrintToScreen ( "left_line_tracker =%d\n" , (int)left_line_tracker ) ;
if ( left_line_tracker <= 900 )
{
    SetMotor ( 1 , 160 ) ;
    SetMotor ( 2 , 95 ) ;
    Wait ( 1000 ) ;
    SetMotor ( 2 , 170 ) ;
    SetMotor ( 1 , 170 ) ;
    Wait ( 100 ) ;
}
centre_line_tracker = GetAnalogInput ( 7 ) ;
PrintToScreen ( "centre_line_tracker =%d\n" , (int)centre_line_tracker) ;
if ( centre_line_tracker  >= 1 )
{
    continue ;
```

```
                    }
                }
                else
                {
                    SetMotor ( 2 , 127 ) ;
                    SetMotor ( 1 , 127 ) ;
                    Wait ( 1000 ) ;
                    SetMotor ( 4 , 85 ) ;
                    Wait ( 500 ) ;
                    SetMotor ( 4 , 127 ) ;
                    Wait ( 1000 ) ;
                    SetServo ( 3 , 0 ) ;
                    Wait ( 1000 ) ;
                    SetMotor ( 4 , 170 ) ;
                    Wait ( 1200 ) ;
                    SetMotor ( 4 , 127 ) ;
                    Wait ( 1000 ) ;
                    SetMotor ( 2 , 95 ) ;
                    SetMotor ( 1 , 160 ) ;
                    Wait ( 3000 ) ;
                    SetMotor ( 2 , 170 ) ;
                    SetMotor ( 1 , 170 ) ;
                    Wait ( 1750 ) ;
                    break ;
                }
            }
        }
    }
}
```