NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

SOFTWARE SIMULATION FOR ROBOT CONTROL

Graduation Project
COM- 400

Student: Mohammed Asfour

Supervisor: Assoc.Prof.Dr Adnan Khashman

Nicosia - 2003

ACKNOWLEDGMENT

To my parents, my professors, my supervisor Assoc.Prof.Dr. Adnan Khashman and my colleagues whose influence on me is as profound as a rising sun across the ocean…

I would like to thank all for your help and encouragement me to develop this project, especially my brothers Ayman and Zaid, also I'll never forget my friends, Nader Ibraik, Mohammed Shuqair, Ala'a Boulos, and my teachers in the high school. It is all your support, which motivated me all the way to bring this work.

Sincerely, I couldn't have done without all your support. It gives me great pleasure to say that I had received great motivation when you added extra words to me. Words such as Good Luck, Go Ahead and Good Job; have worked wonders for me.

I dedicate this project to my University (Near East University) and to the Various Computer Engineers all over the World.

Thanking you from the deepest of my heart…

Mohammed Khalid Asfour

ABSTRACT

Software simulation is a useful tool during the development of any complex system. By simulating the system before it is implemented, it is possible to specify the system requirements and to ensure that new components being developed are working correctly, moreover, many of robot systems use this technique to be under control, and this technique can simulate the reality environment and change it to virtual state.

However, the main advantage for this simulation is to avoid some dangerous situations like in military, emergency, and nuclear places.

This project deals with simple software simulation using Visual Basic programming language to force a simple vehicle to do some preprogrammed jobs as moving, map tracking, and detecting.

TABLE OF CONTENTS

INTRODUCTION

Robotics is a rapidly expanding field of study. As robots become more sophisticated, new applications are being found for which they can be used. Improvements in controller power and robot design have allowed robots to be used in a number of different industries including nuclear power, forestry, defense, automotive and others.

The software simulation is the main field that many of the bigger companies try to develop it, that for making the robots control more visible and under control. Moreover, these days some of the famous laboratories like MIT labs deal with many A.I researches to increase the efficiency of robots.

The aims of this project divided in two main parts:

- Introduce the main function for a simple robot in software as well as in hardware.
- Getting an experience for dealing with this simple type of robot by building and testing the main function of it.

In chapter one, we have introduced the software simulation and the robot control, by illustrated some examples of them.

In chapter two, we have concerned in hardware setup, beginning from the main robot hardware and finishing with the main interface circuit.

In chapter three, we have defined the software setup, and we have shown the main functions of the robot, the main GUI, as well as shown the used algorithms and codes.

Finally, we have illustrated the main results, modifications, and the future of this robot in chapter four.

CHAPTER ONE
ROBOT CONTROL AND SOFTWARE SIMULATION

1.1. Overview

In the last decade, the use of robot systems have increased dramatically although the high cost of design these robots, the components, and we never forget the software development. However, the robot control system has many types of controlling.

This chapter will show us the main types general robot control system and my own robot control system configurations.
It will also define the software simulation techniques and the simple way for simulates the robot movements.

1.2. General robot control system

The robotics in general distinguished in controlling. The robot systems differentiate between each other by the mechanism of control. Here we shall see the main robot control and the differences between every type.

Simply the robot control systems featured to three main types:

1-Normal control (human-aided control system).
2-Automatic control.
3-Servomechanisim controls.

1.2.1. Normal control (human-aided control system)

This kind of control was initially used before using the high technology of automatic control system, here the human-aided control is controlled directly by the human, as shown in figure 1.1.

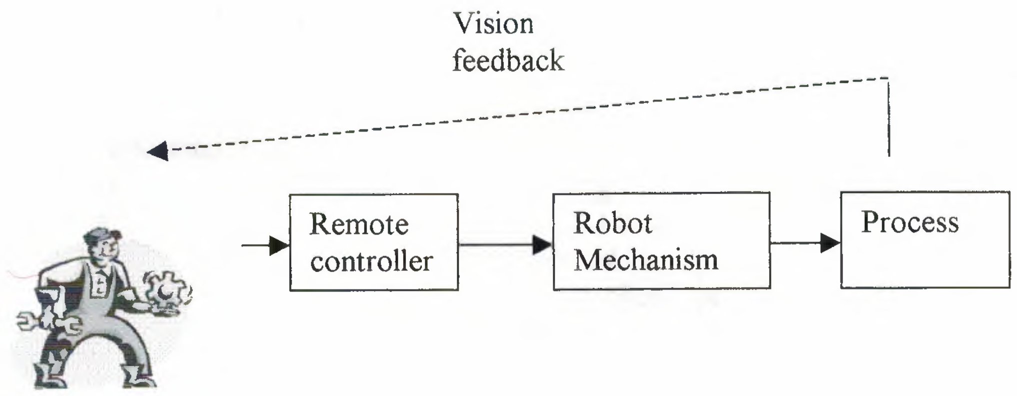


Figure 1.1 Robot normal control system

As shown, the human controls the robot using the remote controller, may be wired or wireless, this process is still used until this year but by modifying the control mode to be more accurate, this is using the servomechanism system.

1.2.2. Automatic control system

To provide automatic control to robot system, machines, electronics, should modify the system or computers replace the operations of human. So here the robot can be controlled by itself using predefined software with the hardware interface components as shown in figure 1.2.

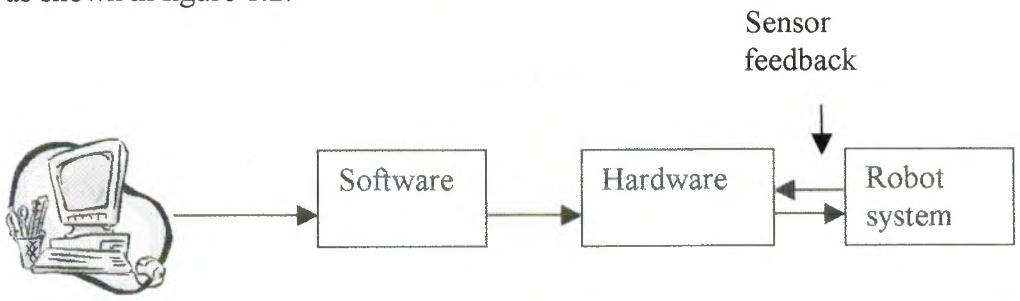


Figure 1.2 Robot automatic control system

The robot automatic control may be as simple as being using in simple toys, or very complex as in Artificial Intelligent (AI) system. The main importance to use this kind of robots is to make the big process as in industrial operation more flexible and comfortable.

Also in these days we can find the automatic control system in many complicated areas of applications as shown in this figure 1.3.

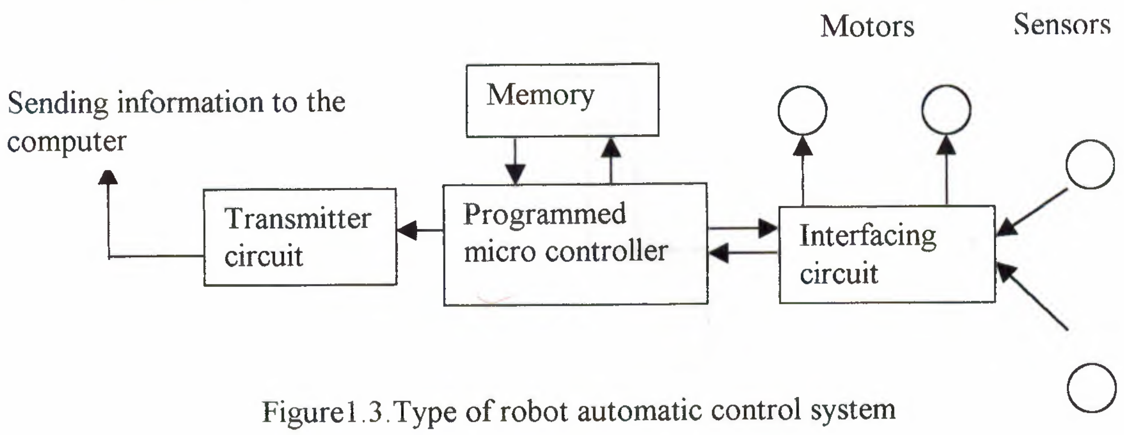

Motors
Sensors
Sending information to the computer
Memory
Transmitter circuit
Programmed micro controller
Interfacing circuit

Figure1.3.Type of robot automatic control system

As we see in this simple black box, this kind of robot system depends on the main chip that may be called the heart of the robot; it is the micro controller, the main job is to control the robot depending on the predefined program, may be you ask is it the same as the main automatic control? , The answer will be yes, but here also we can support the robot by intelligent software to control itself without any cables or any information sent by the computer system.
However, the transmitter can send a signal to the personal computer to make a virtual tracking for robot movement. Therefore the sensors can detect objects and send a signal to the controller to make a calculation to decrease motor speed and avoid this object directly.
These statements work depending on control algorithm.

1.2.3.Servomechanisim control system

This kind of control is the most important and useful for robotics in general application especially in industrial operations. In this case the objective is to force some parameter to vary in specific manner. This may be called a tracking control system. Instead of regulating a variable value to a set point, the servomechanism forces the controlled variable value to follow variation of the reference value. For example, in industrial robot arm as shown in figure 1.4 servomechanisms force the robot arm to follow a path from point A to point B, this is done by controlling the speed of motors driving the arm and the angles of the arm parts.

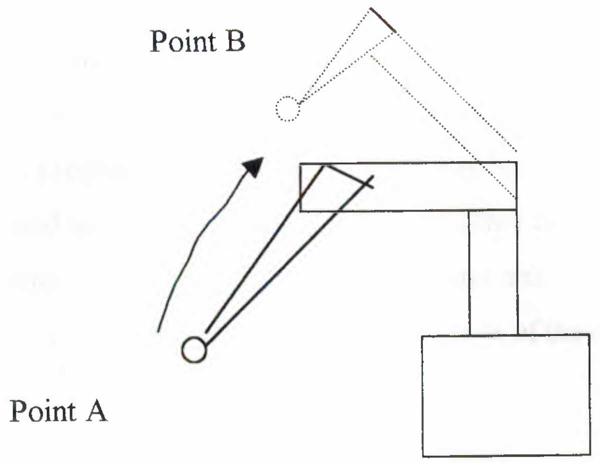


Figure 1.4 Robot control in servomechanism system

1.3. My robot control system

In this project the type of robot is a sample of simple robot system, on the other hand it contains many simple and useful components. Here we shall describe the main control system of the project architecture that presented here.
This system divided for three partitions:

- Software components.
- Hardware interface components.
- The vehicle.

As shown in figure 1.5 we can see the parts of the developed robot system.

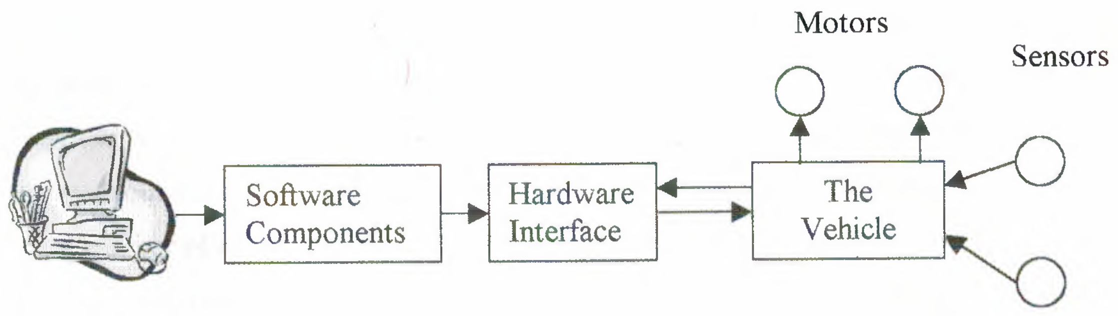


Figure 1.5 my robot system black box

In this figure I shall discuss it briefly because, these parts will be covered in detail in chapter two and chapter three.
The main idea is to guide the vehicle to do some movements depending to the predefined program, the software components has many algorithms for controlling, tracking, and saving the movements. On the other hand, the hardware interface includes a simple interface circuit between the computer and the vehicle. Here the vehicle group both of sensors and driven motors. More details of these components will be covered in the coming chapters.

1.4.Software Simulation

Software simulation is a useful tool during the development of any complex system. By simulating the system before it is implemented, it is possible to specify the system requirements and to ensure that new components being developed are working correctly. These components can then be incorporated without fear of damaging the physical system and this, in turn, simplifies the debugging process.

The simulation of a mechanical system is a useful tool to investigate the behavior of a real world system without having to realize a physical model. It is also a useful tool for predicting the performance of a control system for an existing physical model, without fear of damaging or breaking it. This ability is increasingly important as the complexity of the model grows. With more and more variables to control, the overall effect of a control system on a working model can be difficult to predict.

The control of a walking robot is an example of an extremely complex control problem. Due to the complexity of the control system, a computer simulation of the physical system is needed. The goal of the software simulation is to be able to control the simulation in a similar manner as the method used to control the actual system. This way, new control algorithms can be prototyped in simulation first without having to worry about damaging the actual robot if the control algorithm is initially faulty. There are a number of other variables to worry about with the actual system as well, such as power connections, lack of robustness in the mechanical design, and other issues that complicate the development of walking algorithms. The simulation is a useful tool for simplifying the entire system so that the software designer can focus on the walking algorithms and not have to worry about the other possibly unstable systems on the actual robot.

1.4.1.Software Simulation Design

The main idea of software simulation design is to know what is the reality and simulate it in virtual, as in the previous example for walking robot, the simulation should be for both robot motion, and sensing objects.

In the complex robot design as in AI systems, the need for simulation is more important than the motion itself, however, the simulation techniques in AI systems depends on:

- AI software simulation.
- The type of motion and type of intelligence that it uses.

Here the general software simulation is divided to main two types:

- The software simulation will show the actual motion for the robot.
- The software simulation will track the robot in its motion.
- Using a special method of saving can save the movement.

Now some examples of these types of software simulations:

In figure 1.6 we can see the first type of simulation technique.

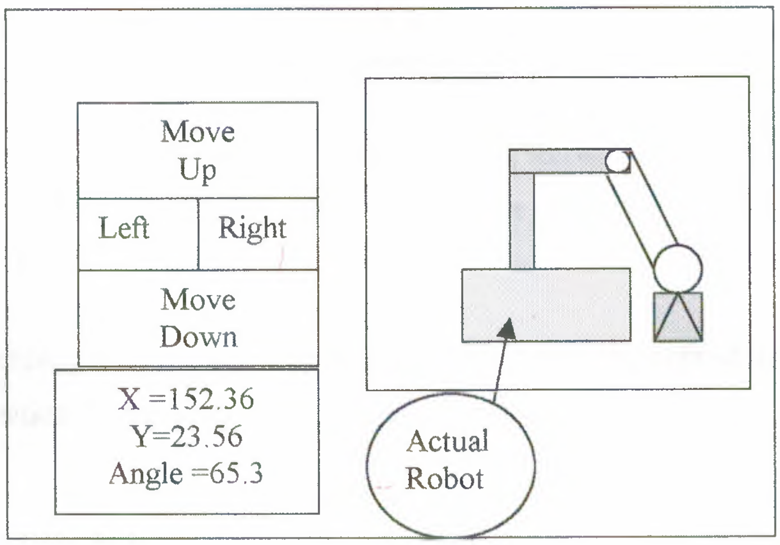


Figure 1.6 The software simulation showing the motion for the robot

As shown in figure 1.6, this type of simulation is complex, although it gives the user a freedom to move the robot to any motion he decide to do, actually this type is useful in the places that needs to monitor and control any motion, as in micro-industrial machines or also now a days in surgery operations.

The next type is the tracking one, as we shall see later, this type may be a complex one or a simple one depending on the way of tracking. On the other hand this type is for more serious application than the other one especially in emergency status or in military applications.

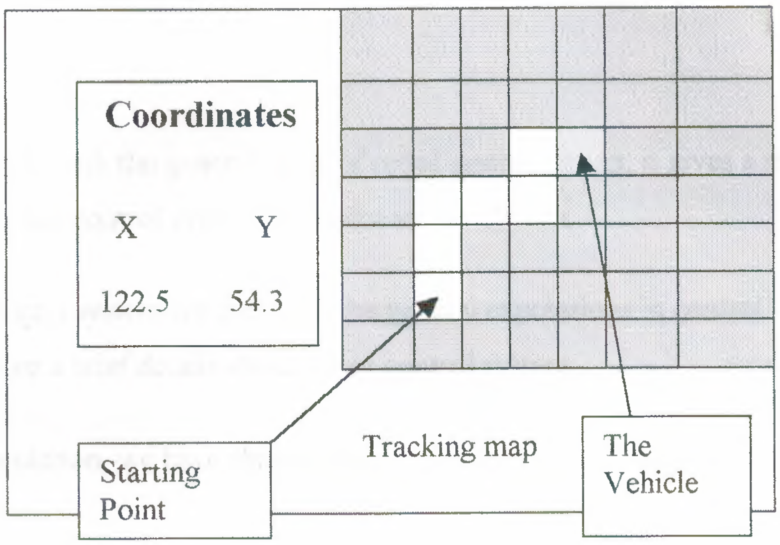


Figure1.7 An example for simple tracking software

As shown in figure1.7, this type is a simple tracking software, this type is used for simple robot system, as illustrated, the user interface shows the grid of tracking and the position of the robot in reality.

In more complex software and hardware, the position may be defined by using GPS (Global Position System) or by using active sensors.

1.4.2. Programming Languages for Software Simulation

Using different programming languages can program the software simulation. However, many people think that the language should be visual language as visual basic, or visual C++, but actually we can simulate the robot by basic languages as basic, Q- Basic, Pascal, and others.

Most popular scientists around the world prefer to use the visual language, for flexibility, comfort ability, and it is easier to modify. The most commonly used languages are Visual Basic, Delphi, and Visual C++.

1.5. Summary

This chapter deals with the general topic of robot system design; it gives a simple discussion on robot control system and software simulation.

In the robot control system we deal with the general expressions in control design, although we gave a brief details about robot control system.

In software simulation, we have shown some examples on the main software design.

CHAPTER TWO
HARDWARE SETUP

2.1. Overview

In the previous chapter some of the general descriptions about robots were introduced. In this chapter, we shall see more descriptions about hardware setup and the circuits that we have used to develop this project.

2.2. Hardware Configurations

The hardware configurations are the most important section of robot system, because this section defines many of hardware components, robot process control, and also the interface between hardware and software.

The hardware in general, consists of three major sections:

- Computer hardware.
- Robot circuit interface.
- The vehicle.

Moreover, the software that used in the user interface control should be compatible with the computer hardware.

In hardware configuration, as we are going shortly to start, should be active with the software components. In chapter three, the connection between the hardware and software components will be defined in details.

Therefore, any robot system consists of three main parts:

- Input process.
- Interface circuit.
- Output process.

As shown in figure 2.1

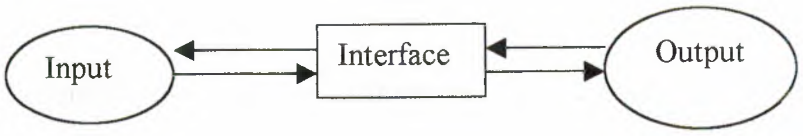


Figure 2.1. Robot process system

2.3. Input process

Every process control has an input process; moreover, there are two types of input for the robot process control:

- Input process with unidirectional.
- Input process with bi-directional.

These types is the most important, that's, the both inputs depending on the feedback process. In this project we shall see the two types.

The developed robot deals with both types individually, however, the input control signal in this robot may consists of two types of signals:

- Guided input signal (without using the sensors).
- Unguided input signal (using the sensors).

2.2.1. Guided input signal (without using the sensors)

Here the input signal contains only the control data, therefore, it forces the robot to do some predefined commands using the input devices.

As shown in figure 2.2 we can see the user can enter many commands to force the robot without any feedback signal.

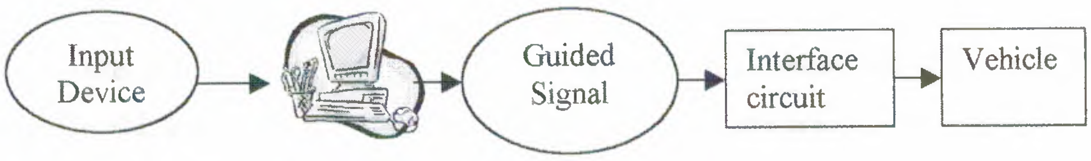


Figure 2.2. Guided input signal

2.2.2. Unguided input signal (using the sensors)

This type of signal contains smart predefined commands, depending on the sensors feedback signal. The user in this type only gives a command using the input device, to begin the operation.

As shown in figure 2.3, the computer should be fed by the sensor as a detect signal.

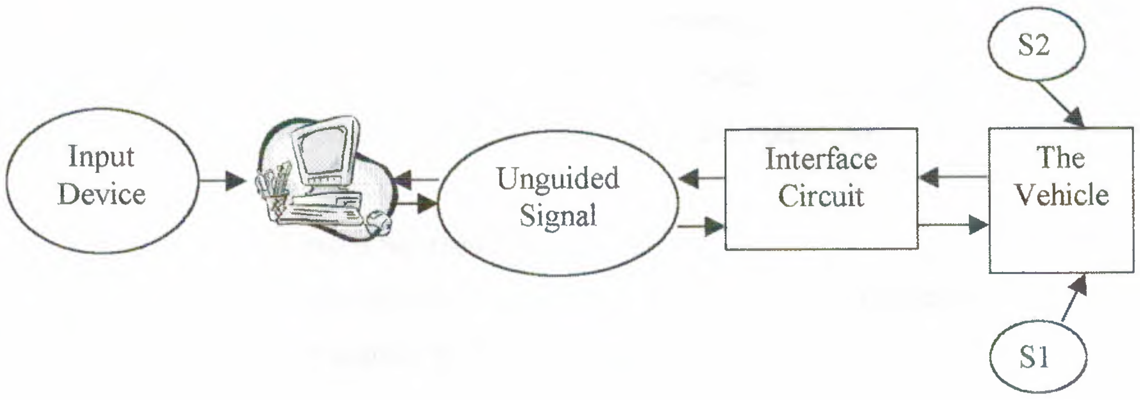


Figure 2.3. Unguided input signal (using the sensors)

- S1 and S2 are the sensors, S1 represents right sensor, and S2 represents left sensor.
- Here the computer sends a guided signal via the interface signal to the vehicle.
- The computer has programmed to wait the feedback signal through the interface circuit.
- By using the predefined algorithm, the computer sends the correction for the process.

2.4. The parallel input/output (I/O) port

The parallel port allows the microcomputer to communicate directly with the outside digital world, and handles most of the problems of control and synchronization with the microprocessor address and data buses. The most convenient is the bi-directional port, which has separate input and output lines, simplifying the connection to external devices. Since all bits are transferred in parallel (at the same time), it is generally faster than the serial port.

The parallel I/O port usually includes the following addressable internal registers:

1- **A data register**: it is for reading and writing.
2- **A control register**: that can be written to set the mode of operation and control the logic level of external lines. It permits the program to communicate with external circuits and tell them when the program has new output or is ready to accept new input.
3- **A status register**: that can be read for determining the status of the data register. Its contents can be read by the program to determine:
    - I- When a device connected to the data output port is ready to accept new data.
    - II- When it has read the last data output.
    - III- When new data have been latched by the external device on the input port and are ready to be read.

2.4.1. The parallel output port

The parallel output port consists of a set of registers that can be addressed and loaded from memory under program control. Whenever the output enable line is asserted, the contents of the register are available on the external output lines.
As shown in figure 2.4 the parallel output port using an octal edge-triggered D-flip-flop, address decoder, and tristate output buffer.[9]

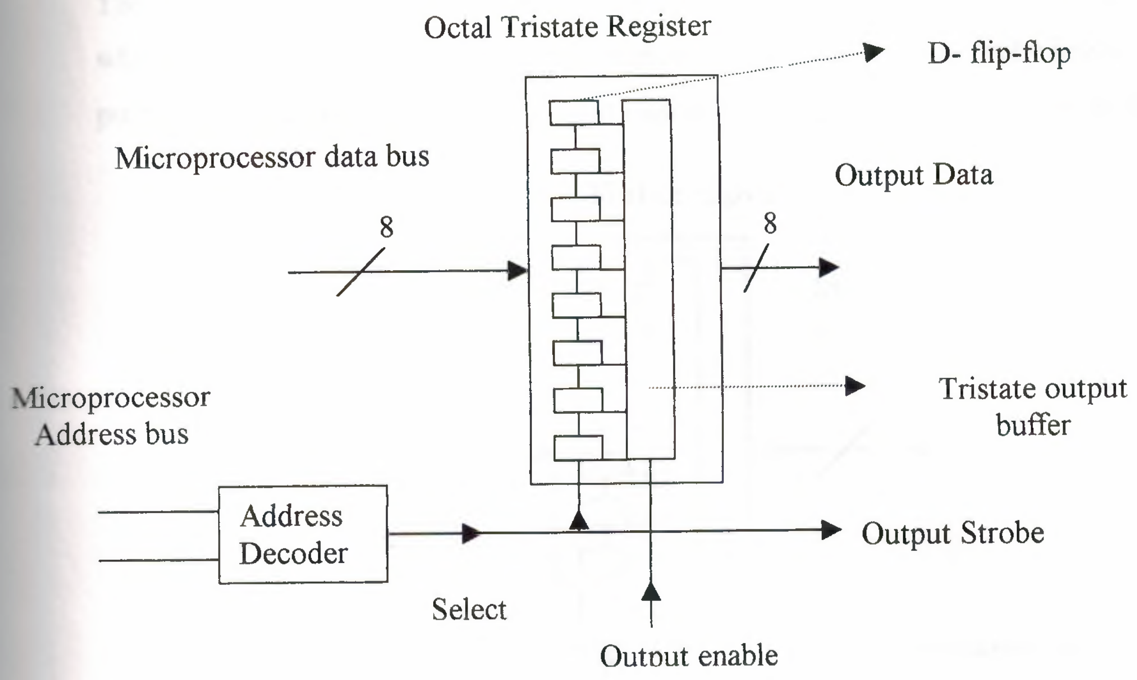


Figure 2.4. Parallel output port

Here when addressed by the microprocessor, data from memory is transferred to the registers and is available at the inputs of the tristate buffers until a new byte is written. The data appears at the output lines whenever the external circuit provides an output enable.

2.4.2. The parallel input port

The parallel input port converts a bit pattern of logic voltage levels in the world outside the computer to a number in computer memory. Additional control lines and status registers may also be provided, so that:

I- The computer program can notify the external circuit that it is ready to receive data.

II- The external circuit can notify the program that has written data to the port's internal registers.

III- The computer program can notify the external circuit that the data have been read.

The parallel input port as shown in figure 2.5, consists of a set of registers that stores whatever data are on the external input lines at the instant, the input strobe line is pulsed. The microcomputer program can address and read these registers at a later time.

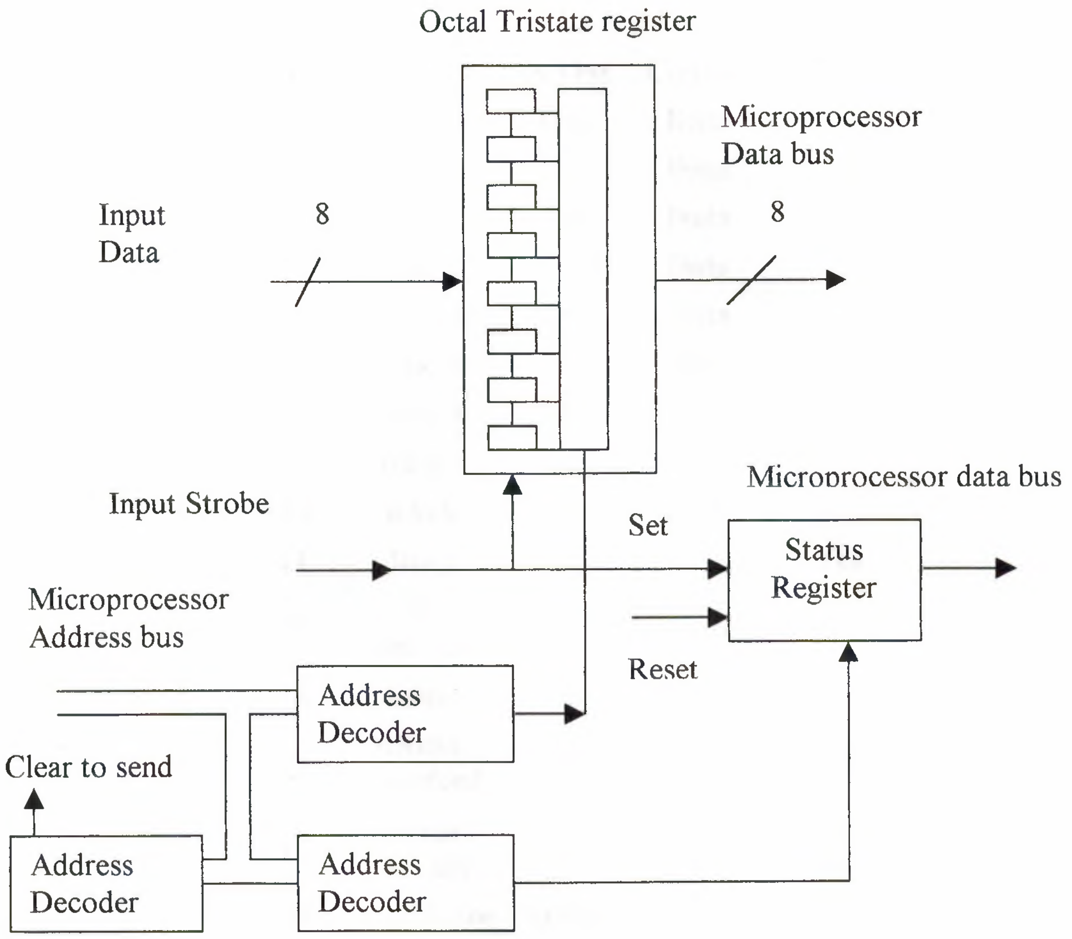


Figure 2.5 parallel input port

2.4.3. The D-Type 25 Pin connector

Below is a table of the "Pin Outs" of the D-Type 25 Pin connector. The D-Type 25 pin connector is the most common connector found on the Parallel Port of the computer. The IEEE 1284 standard however specifies 3 different connectors for use with the Parallel Port. The 1284 Type A is the D-Type 25 connector found on the back of most computers IEEE 1284. This connector is claimed to have a better clip latch, better electrical properties and is easier to assemble. It also contains two more pins for signals, which can be used to see whether the other device connected, has power.

Table 2.1. Pin Assignments of the D-Type 25 pin Parallel Port Connector.

Pin No (D-Type 25)	SPP Signal	Direction In/out	Register	Hardware Inverted
1	nStrobe	In/Out	Control	Yes
2	Data 0	Out	Data	
3	Data 1	Out	Data	
4	Data 2	Out	Data	
5	Data 3	Out	Data	
6	Data 4	Out	Data	
7	Data 5	Out	Data	
8	Data 6	Out	Data	
9	Data 7	Out	Data	
10	nAck	In	Status	
11	Busy	In	Status	Yes
12	P-out/End	In	Status	
13	Select	In	Status	
14	nAuto-Linefeed	In/Out	Control	Yes
15	nError / nFault	In	Status	
16	nInitialize	In/Out	Control	
17	nSelect-Printer / nSelect-In	In/Out	Control	Yes
18 - 25	Ground	Gnd		

This table uses "n" infront of the signal name to denote that the signal is active low. e.g. nError. If the printer has occurred an error then this line is low. This line normally is high, should the printer be functioning correctly. The "Hardware Inverted" means the signal is inverted by the Parallel card's hardware. Such an example is the Busy line. If +5v (Logic 1) was applied to this pin and the status register read, it would return back a 0 in Bit 7 of the Status Register.

The output of the Parallel Port is normally TTL logic levels. The voltage levels are the easy part. The current you can sink and source varies from port to port. Most Parallel Ports implemented in ASIC, can sink and source around 12mA. However these are just some of the figures taken from Data sheets, Sink/Source 6mA, Source 12mA/Sink 20mA, Sink 16mA/Source 4mA, and Sink/Source 12mA.The best bet is to use a buffer, so the least current is drawn from the Parallel Port.

2.4.4. Hardware interrupts method

The hardware interrupts method permits data analysis during data acquisition, therefore, the reading of the parallel input port is initiated by an interrupt, which causes the program to pause whatever it is doing while a special jump occurs to an interrupt-service routine as shown in figure 2.6. When the routine has serviced the interrupt, it transfers back to the main program.

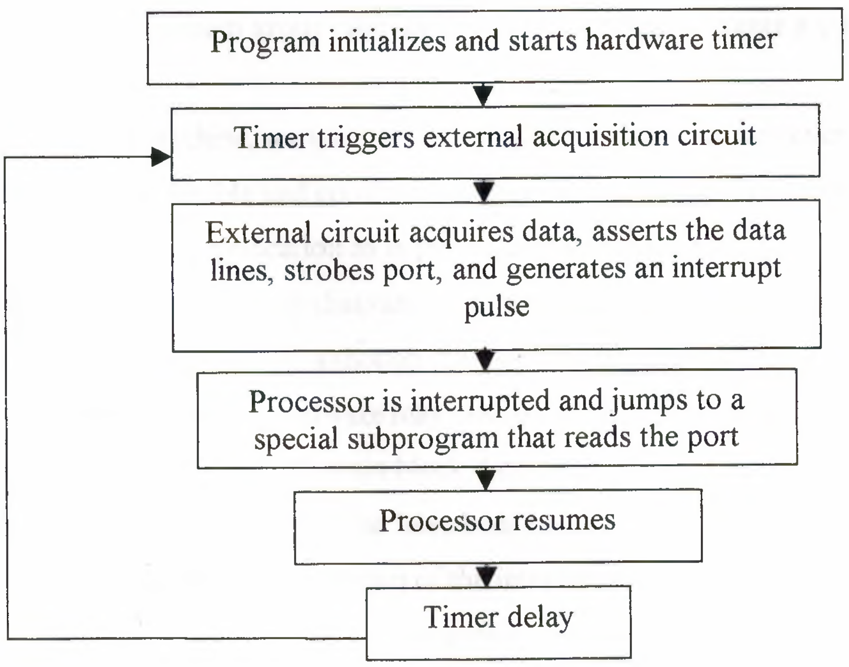

Program initializes and starts hardware timer
Timer triggers external acquisition circuit
External circuit acquires data, asserts the data lines, strobes port, and generates an interrupt pulse
Processor is interrupted and jumps to a special subprogram that reads the port
Processor resumes
Timer delay

Figure 2.6 Flow chart for hardware interrupt method

Here are descriptions for the previous flow chart:

1- The program initializes the hardware timer to repetitively load a number and put a pulse every time the terminal count is reached. The counter is then armed to initiate the trigger pulses. An interrupt-service routine initialized.

2- Each pulse triggers the external circuit.

3- After the external circuit has new data, it asserts them on the input lines of the parallel port, pulses the strobe line to latch the data and set the status bit, and generates an interrupt pulse.
4- The interrupt pulse causes a number of registers to be saved in a "stack", and jump to special interrupt service routine that reads the parallel input port, restores the registers, and returns to the main program.
5- Go to step 2.

2.5. The Interfacing

Any robot system should be interfaced to some operating system, like microprocessor circuit, microcontroller circuit, or microcomputer system. Furthermore, some complicated robot system groups the last systems together, to create a complex operation.
In this developed machine, we use the microcomputer interface, however, this type of interfacing is more flexible and comfortable. But, for more freedom robot in future we need to use a wireless application as in microcontroller systems.
As we shall see the main block diagram for the interface circuit, we should be careful for dealing with the computer, although the quick operation that will occurred, but the sensitivity of the computer hardware may destroy some critical parts in it.
Here, this section will show the main block diagram, the main circuit diagram, and the details of electronic components that have been used.
Figure 2.7 is shown the block diagram of the interface circuit that has used.

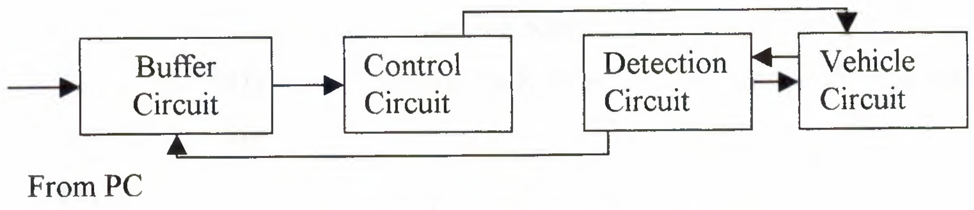


Figure 2.7 Interface block diagram

2.5.1. The buffer circuit
The buffer circuit is an external circuit that has been used in this robot; furthermore, this circuit is the first circuit that the control signal entered to the main control circuit of the robot.

Once the signal entered to the buffer circuit, the control circuit begins to work, and after take a look for the buffer circuit we shall to enter the control circuit.

Here we have used the TTL chip (74373) as a buffer, however, this chip operates using a regulated +5 volts.

In figure 2.8, we can see the chip with all usable pins.

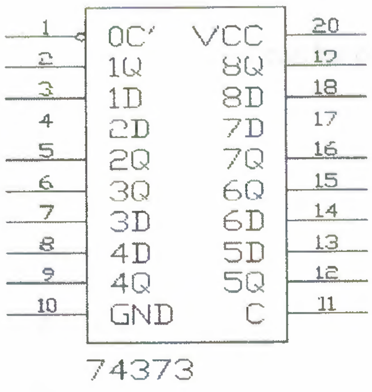


Figure 2.8 TTL (74373) Buffer chip

This chip also we can call it a latch circuit; therefore, in this circuit we use it as a latch more than a buffer. The latch is like controlled gates that used to control each signal that comes from the parallel port.

As shown in the last figure:

- Pin 1(OC'): This pin is used for output enable, that's, when this pin is connected to the ground, the output will be enabled.
- Pins (1Q to 8Q): These pins are used for output the data that buffered in the chip; this will be happened when we connect Pin 1 to ground.
- Pins (1D to 8D): These pins are ready to accept a data from the computer after Pin 11 has enabled.
- Pin 11 (C): This pin is used to control the input data from the computer; therefore, the data will be not accepted except this pin is connected to logic 1.
- Pins (VCC, Gnd): These pins are used to feed the chip with power, VCC connected to +5V,where Gnd connect to ground.

2.5.2. The control circuit

The control circuit is an important part of the circuit, as operates the vehicle. This circuit has been designed in very simple way, that's using simple electronic components.

The main idea is to switch the motors ON or OFF depending on the incoming signal; on the other hand, we have used a 7-segment display that displays the motion (forward, right, and left).

We use two relays that operate on +5V, and a 5A can be passed through its contacts. In figure 2.9, it shows the used relays.

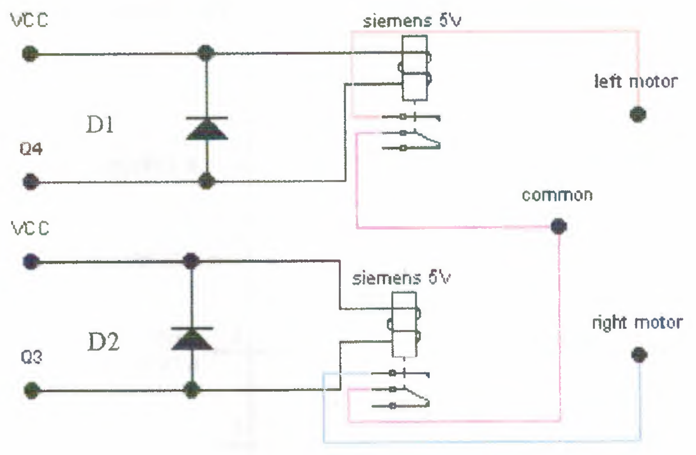


Figure 2.9 The motor control circuit

- Where Q3 and Q4, is connected to the TTL chip (74373).
- D1, D2 are diodes connected parallel to the relays and opposite to the supply, however, we have used these diodes to reduce unexpected inductance currents that's may cause a problem in the circuit.
- The contactors of the relays are connected to the vehicle circuit, for controlling the left motor and right motor.

2.5.3. The detection circuit

This robot has a special circuit for detecting the object, furthermore, this type of circuit has many kinds of operation, and it is vary from robot application to another, depending on the application. For example, the robot that uses the detection circuit to differentiate objects, this needs a high performance circuit, to make an image processing to detect.

The developed robot has used only to detect an objects, however, we have used a simple analog circuit to detect objects. Also the main sensors that have used are IR (InfraRed) object detection sensors, we should not forget the operation amplifier chip that used to amplify the Phototransistor signal.[3]

In figure 2.10, we show the operational amplifier LM358.

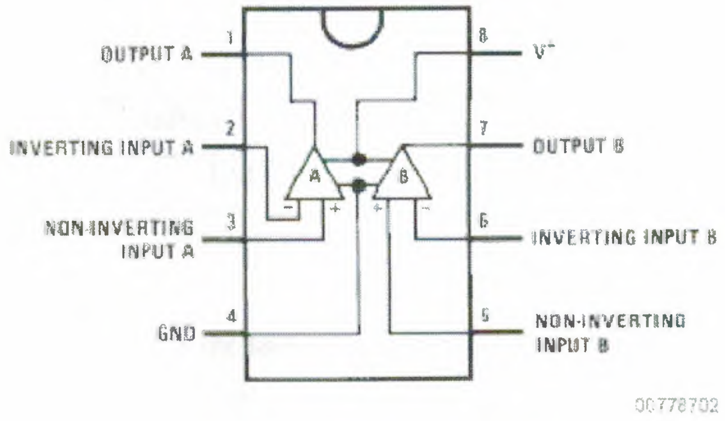


Figure 2.10 The operational amplifier LM358

Also in figure 2.11 we can see the IR detection circuit that has used in this project

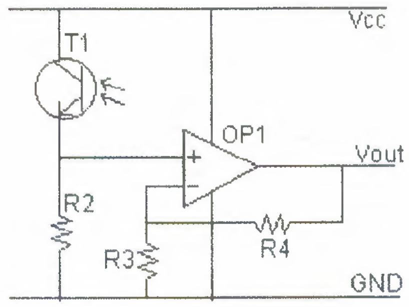


Figure 2.11 The IR detection circuit

In the developed project we have used two circuits for the same type of sensing, where:

- R2 should be larger than the maximum resistance of the detector. We should measure the resistance of the detector when it is pointing into a dark area and then choose the next larger resistor.
- R3 and R4 determine the amplification of the OpAmp, where, Gain=1+R4/R3, and Vout=(1+R4/R3) Vin.

As shown in table 2.2 the main electronic parts that have been used in this circuit.

Table 2.2 A list of used components

Component	Value/number	Cost in $	Company
R2	1MΩ	0.2	-----
R3	4.7kΩ	0.2	-----
R4	6.8kΩ	0.2	-----
T1	Infrared transistor	1.0	Radio shack
OP1	LM358	1.0	National

We should build two circuits the same of the previous one; therefore, we have two directions to detect, right and left.

On the other hand, we should build a circuit for transmitting the IR beam, so the below figure 2.12 has shown the circuit for this purpose.

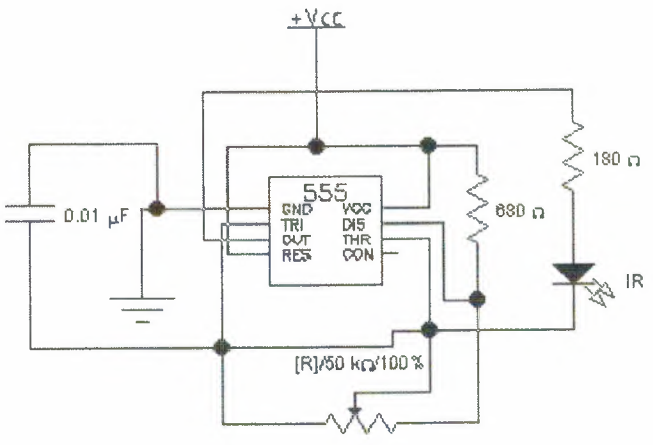


Figure 2.12 The IR transmitter circuit

This circuit is used to transmit the infrared beam to detect the object, however, this infrared beam should be 40KHz frequency, so this circuit should be calibrated.
The list of the used components is shown in table 2.3.

Table 2.3 List of used components in IR transmitter circuit

Components	Values / numbers	Cost in $	Company
R1	680 Ω	0.2	----------
R2	Variable resistor 50 k Ω	0.5	----------
R3	180 Ω	0.2	----------
C1	0.01 μF	0.2	---------
IC1	LM555	1.0	National
IR LED	A cheap IR LED	0.5	Radio shack

Moreover, this circuit is built on the vehicle to be more sensitivity.
The output of the detection circuit is connected to quad analog switches; this type of switches is used to give pulse to the parallel input port. Here we have used a CMOS chip for more accuracy and sensitivity.
As shown figure 2.13 we can see the quad analogue switches (4066).

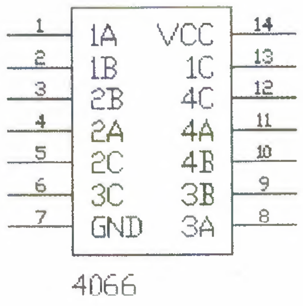


Figure 2.13 Quad Analogue Switches (4066)

Where:

- (1A to 4A) are the bi-directional input/output analogue signal.

- (1B to 4B) are the bi-directional output/ input analogue signal.
- (1C to 4C) are the switch control signal.
- (VCC and Gnd) VCC for the power supply and its range from 5V to 15V, and Gnd is the common ground.

From this chip we have used only two analogue switches, therefore, the parallel input port does not accept only a digital signal, so, the signal that has been detect should approach to threshold and then the detector circuit creates a signal and passes it to the analogue switches, then to the input port.

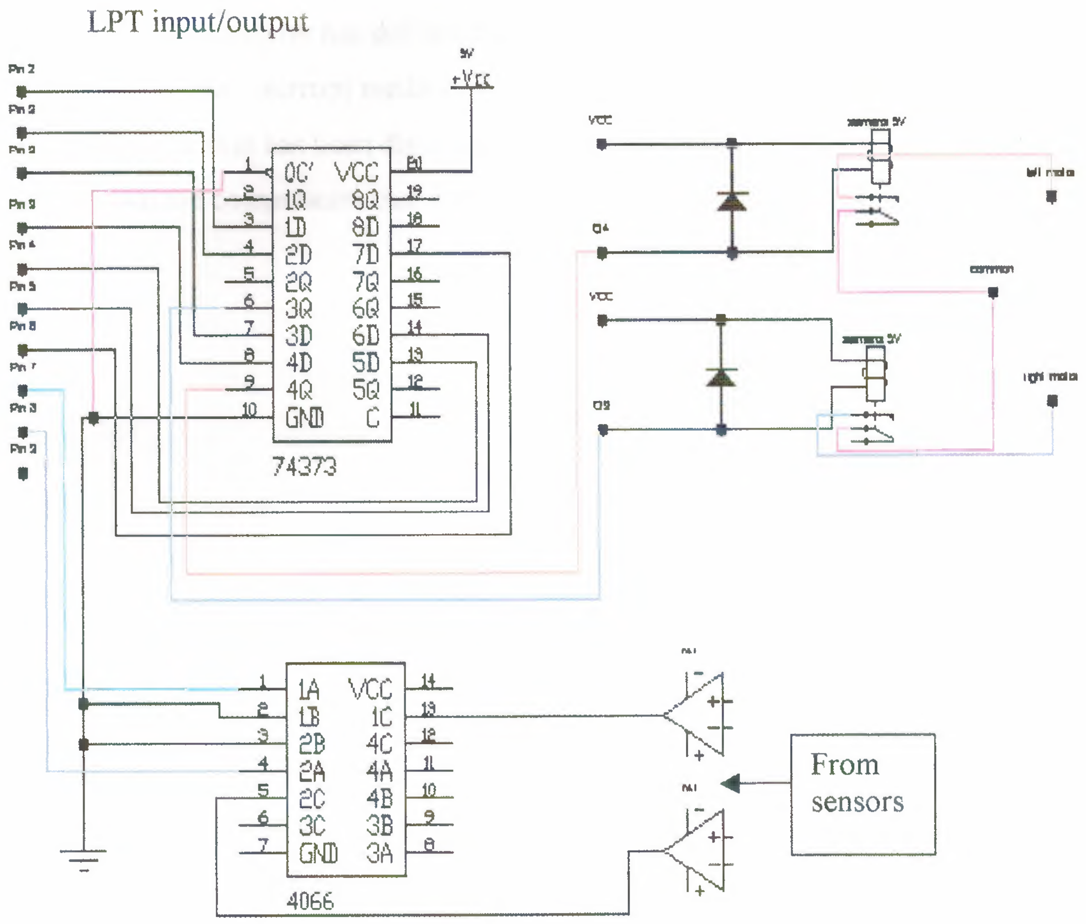


Figure 2.14 The application circuit

2.6. Summary

We can summarize this chapter as follows:

- This chapter has defined the main hardware topics beginning from the basic process and ending with the application circuit.
- It has also defined the input process and its features.
- We have shown the parallel input output port pins, as we have also shown the design of the parallel port.
- Moreover, this chapter has defined the hardware interrupt model and given a flow chart of the interrupt method.
- The last point that has been discussed is the interface circuit, on the other hand it has shown the components that have been used and how to connect it.

CHAPTER THREE

SOFTWARE SETUP

3.1. Overview

In the previous chapter we have defined the hardware setup, while in this chapter the software setup will be defined. We shall begin with the program language used for the main algorithms and codes. Furthermore, we shall give an example of the designed GUI and the technical specification of the software structure.

3.2. What is Visual Basic?

Visual Basic (VB) is an object-oriented, event-driven GUI programming language developed by Microsoft Corporation for Windows application development.

Bill Gates, the CEO and cofounder of Microsoft, developed the original DOS Basic interpreter as one of his last programming projects more then a decade ago. Since then, Visual Basic has grown to more then several hundreds keywords reflecting its rich object-oriented nature. [6]

3.3. Why Visual Basic?

Visual Basic is a useful language for many reasons:

- We can have multiple windows on one screen. These windows have full access to the clipboard and to the information in most other windows application running at same time.
- Microsoft Visual Basic is the quickest and easier way to create applications for Microsoft Windows, which run quickly.
- Visual Basic makes it easy to build large programs by allowing modern modular programming techniques. This means you can break a program into easy-to-handle modules.

- The programming language built into Visual Basic has easy-to-use graphics statements, powerful built-in function for mathematics and string manipulations, and sophisticated file-handling capabilities.

3.4 Modules

Code in Visual Basic is stored in modules. There are three kinds of modules: form, standard, and class.

Simple applications can consist of just a single form, and all of the code in the application resides in that form module. As your applications get larger and more sophisticated, you add additional forms. Eventually you might find that there is common code you want to execute in several forms. You don't want to duplicate the code in both forms, so you create a separate module containing a procedure that implements the common code. This separate module should be a standard module. Over time, you can build up a library of modules containing shared procedures.

Each standard, class, and form module can contain:

· Declarations. You can place constant, type, variable, and dynamic-link library (DLL) procedure declarations at the module level of form, class or standard modules.

· Procedures. A Sub, Function, or Property procedure contains pieces of code that can be executed as a unit. These are discussed in the section "Procedures" later in this chapter.

3.4.1 Form Modules

Form modules (.FRM file name extension) are the foundation of most Visual Basic applications. They can contain procedures that handle events, general procedures, and form-level declarations of variables, constants, types, and external procedures. If you were to look at a form module in a text editor, you would also see descriptions of the form and its controls, including their property settings. The code that you write in a form module is specific to the particular application to which the form belongs; it might also reference other forms or objects within that application.

3.4.2 Standard Modules

Standard modules (.BAS file name extension) are containers for procedures and declarations commonly accessed by other modules within the application. They can contain global (available to the whole application) or module-level declarations of variables, constants, types, external procedures, and global procedures. The code that you write in a standard module isn't necessarily tied to a particular application; if you're careful not to reference forms or controls by name, a standard module can be reused in many different applications.

3.4.3 Class Modules

Class modules (.CLS file name extension) are the foundation of object-oriented programming in Visual Basic. You can write code in class modules to create new objects. These new objects can include your own customized properties and methods. Actually, forms are just class modules that can have controls placed on them and can display form windows.

3.4.4 Input /Output module

In this project, the I/O module should be installed; this module is responsible to make a connection between the software and hardware environments.

Furthermore, we have installed this module as (IO.DLL) through windows platform to system32. After that we should call this module to Visual Basic program before building the main forms or the user interface.

3.5. The GUI (Graphical User Interface)

The GUI is a kind of user interface; it might be a complex or a simple user interface depending on the application. The GUI includes some hidden codes that generate the application as shown in figure 3.1 we can imagine the main operation for this topic.

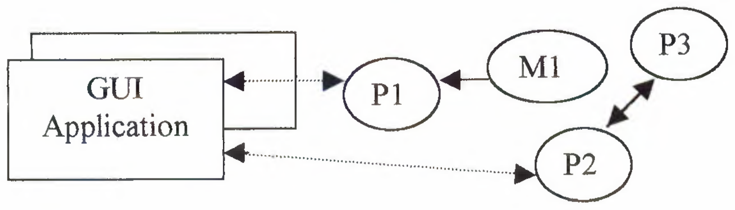


Figure 3.1 the GUI and the hidden processes

Here P1, P2, and P3 are a processes, while M1 is a module connecting with process1, these processes has been called using the Graphical Controls, moreover, here we shall show some of the useful Graphical Controls:

- Command Buttons: Most VB application have command buttons that allows the user to simply click them to perform an action, when the user chooses the button, it not only carries out the appropriate action, it also looks if its being pushed in and released. Whenever the user clicks a button, the click event procedure is invoked.
- Text Boxes: Text Boxes are controls that can be used to get input from the user to display text.
- Message Boxes: Message boxes display information in dialog box superimposed on the form.
- List Boxes: the List Boxes display data in a list, they are used to output a data and display it either by sorting or without sorting, moreover, they are used for both numeric or string data type.
- Picture Boxes: they are used for displaying shapes or string separately of the main form; they have their own coordinates for controlling where the picture will be shown, and their own control.
- Labels: they are used to display some texts as a label, but they are not used for input data.

3.6. The developed GUI

The developed GUI is one of important parts of this project. The designed GUI is a simple one, without any complicated ideas, except using a track drawing, however, our user interface consists of:

- Command buttons
- Labels
- Picture Box
- List box
- Forms

As shown in figure 3.2 we can see the main user interface that has been used

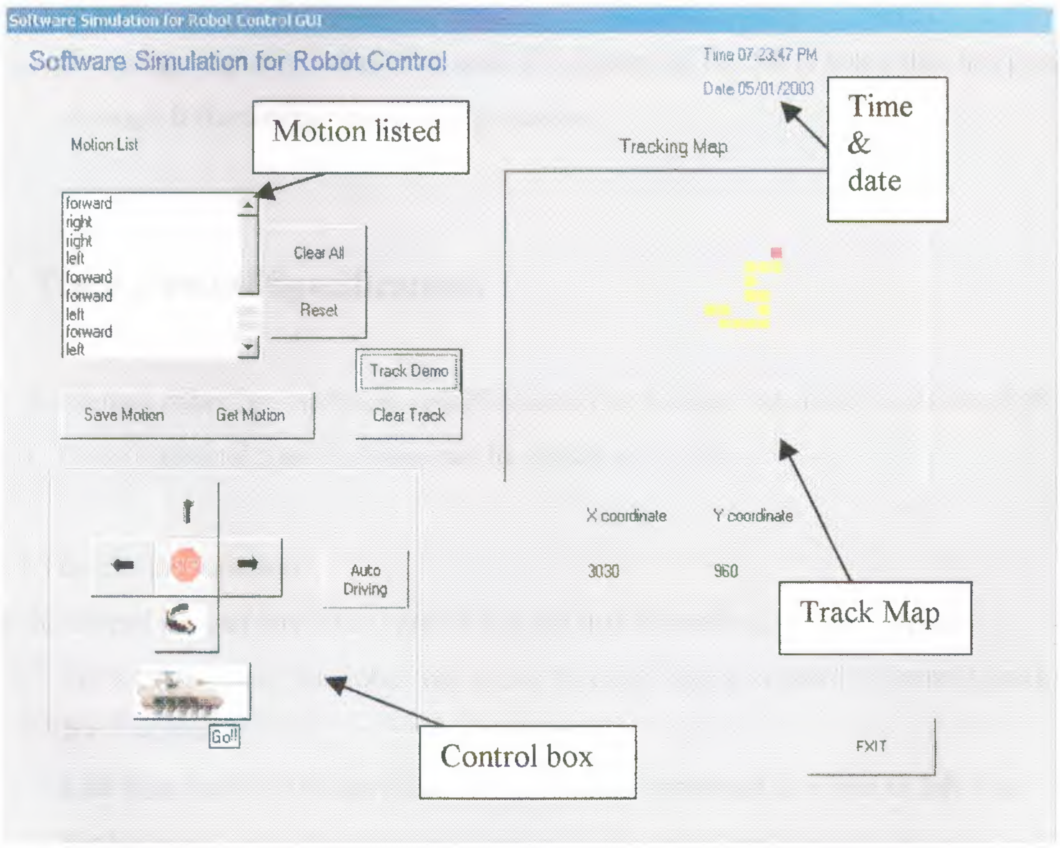


Figure 3.2 The designed GUI

As shown in previous figure, this is the developed user interface, after a while we shall support this chapter the main function that this user interface can control. Furthermore, we can summarize this GUI by these features depending on functions:

- The Control Box: this region is specified for robot control, it is also contains a motion control (forward, left, right, rotate, run, auto drive).
- Motion listed: this region is for listing the motions depending on the commands, here we have:
  1- Clear all button: this is for clearing the list and the picture box without reset the main variables.
  2- Reset button: this is used for reset all the form with the variable.
  3- Get motion: this is used for getting a saved motion from a file.
  4- Save motion: this is used for saving a prefer motion to a file.
- Date & Time: these labels display the currently date and time depending on the system.
- Tracking map: this is a picture used for displaying the robot track that has gone through it from origin point to destination.

3.7. The Technical Specifications

The developed robot has technical specifications that contain the main functions of this robot. These technical specifications can be shown as follow:

3.7.1 The motion function:

The developed project has four types of motion that depending on the program:

- Forward Motion: this robot can move forward, every forward command work for 3 seconds.
- Left side motion: the developed project has a command to move to left side, furthermore, once the command operates, the robot can move 90 degrees to left, and it operates for 5 seconds.
- Right side motion: this kind of motion can force the robot to move to right side, moreover, once the command operates, the robot can move 90 degrees to right, and it operates for 5 seconds.

- Rotate 180 degrees: this function used for rotating, that's, since the command operates the robot can rotate 180 degrees, this simply move to right side by doubling the time.
- The developed robot doesn't need the backward command, because the rotate function is used instead of the backward function.

3.7.2 The tracking map

While the upper commands are running we need to track what is currently happen in the reality without having a look after the robot, this option is offering that; the developed software is able to draw a track illustrates the reality of the motion, however, every command should be changed to draw a track depending on:

- The origin coordinates
- The old coordinates, that's we have here X1, and Y1
- The new coordinates, that's we have here X2, and Y2

These coordinates will help the drawing track to be accurate.
Furthermore, we have a button for make a test for tracking before run the robot ; this will decrease the error of unexpected motion. It also gives the new coordinates for a new position.

3.7.3 Saving and Getting the track

The developed software is able to save any favorite motion ins a special file and get it in any time, moreover, the motion should not exceed the allowed number, for example, in this project the max motion that we can operate it are 30 motions, if we give an order to save it will clear the old commands automatically, but actually, the program can be modified in order to save and get more than these commands in the future.

3.7.4 Auto Driving function

This function is used to make the robot move depending on the built in sensors.

NOTE: all the previous functions will be shown in details in the algorithms and codes section, and any modifying will be announced in chapter four.

3.8. Input and storing commands algorithm

This algorithm used to decide the input then storing it in an array, moreover the input devices are the mouse and keyboard.

Here the used array stores up to 30 data or we call it motion, in figure 3. we shall see the flow chart diagram for this operation.

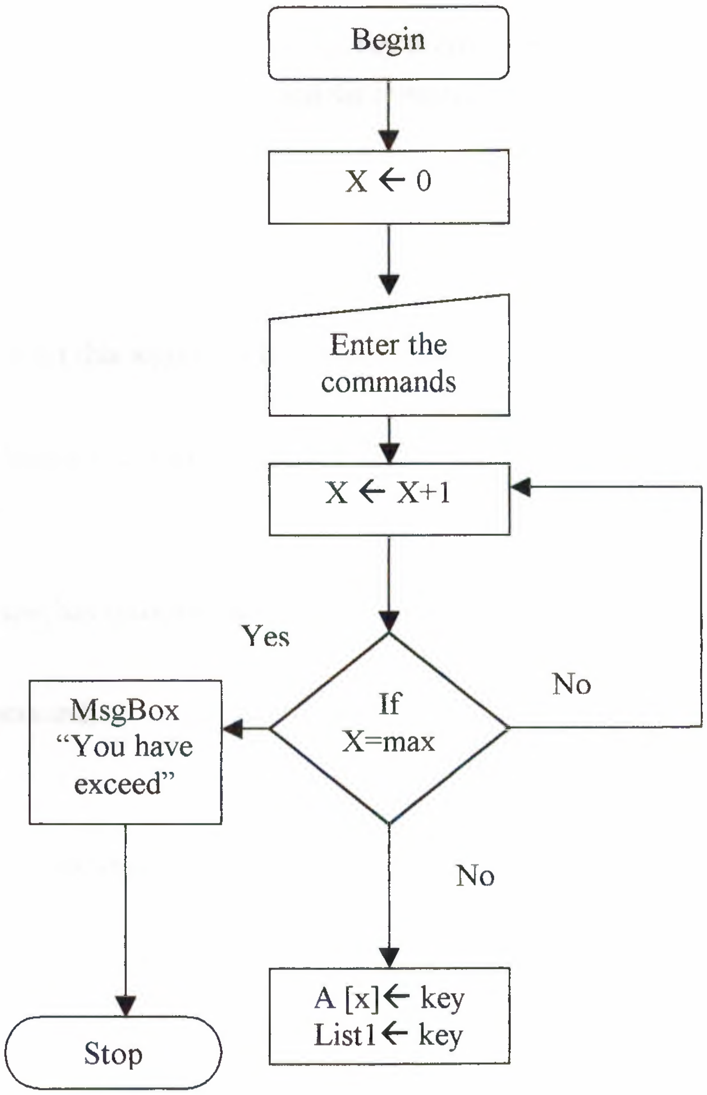


Figure 3.3. Input and storing commands flow chart

The algorithm of the previous flow chart is as follow:

```
1- begin                    'start using the interface'.
2- X←0                      ' assign counter to zero'.
3- ready                    ' ready to enter using mouse or keyboard'
4- X←X+1                    ' the counter will be increased every command'
5- if X= max array storage
6- msgbox " you have exceeded the limit of storage"
7- Else
8- a [x] ← key               ' assign key to array a[]'
9- list1 ← key               ' add the command name to list 1'
10- Goto 4
11- End
12- Stop
```

Now we shall convert this algorithm to sample visual basic code:

```
Dim X as integer
'When the form is loaded X will automatically initiate'
Private Sub command_Click()
If X + 1 = n Then
MsgBox "This motion has exceeded the maximum allowed number of moves!!"
Else
List1.AddItem (command)
a (X) = command
X = X + 1
last_element_index = last_element_index + 1
End If
End Sub
```

3.9. Track drawing algorithm

The track drawing algorithm is responsible to convert the command to tracking pict furthermore, every motion will be tracked in virtual on the GUI mode. It is difficult arrange between the virtually and reality environments, but this simple algorithm sha help the program to give an approximation point from the origin point (home) to destination. In figure 3.4, the main flow chart will be shown.

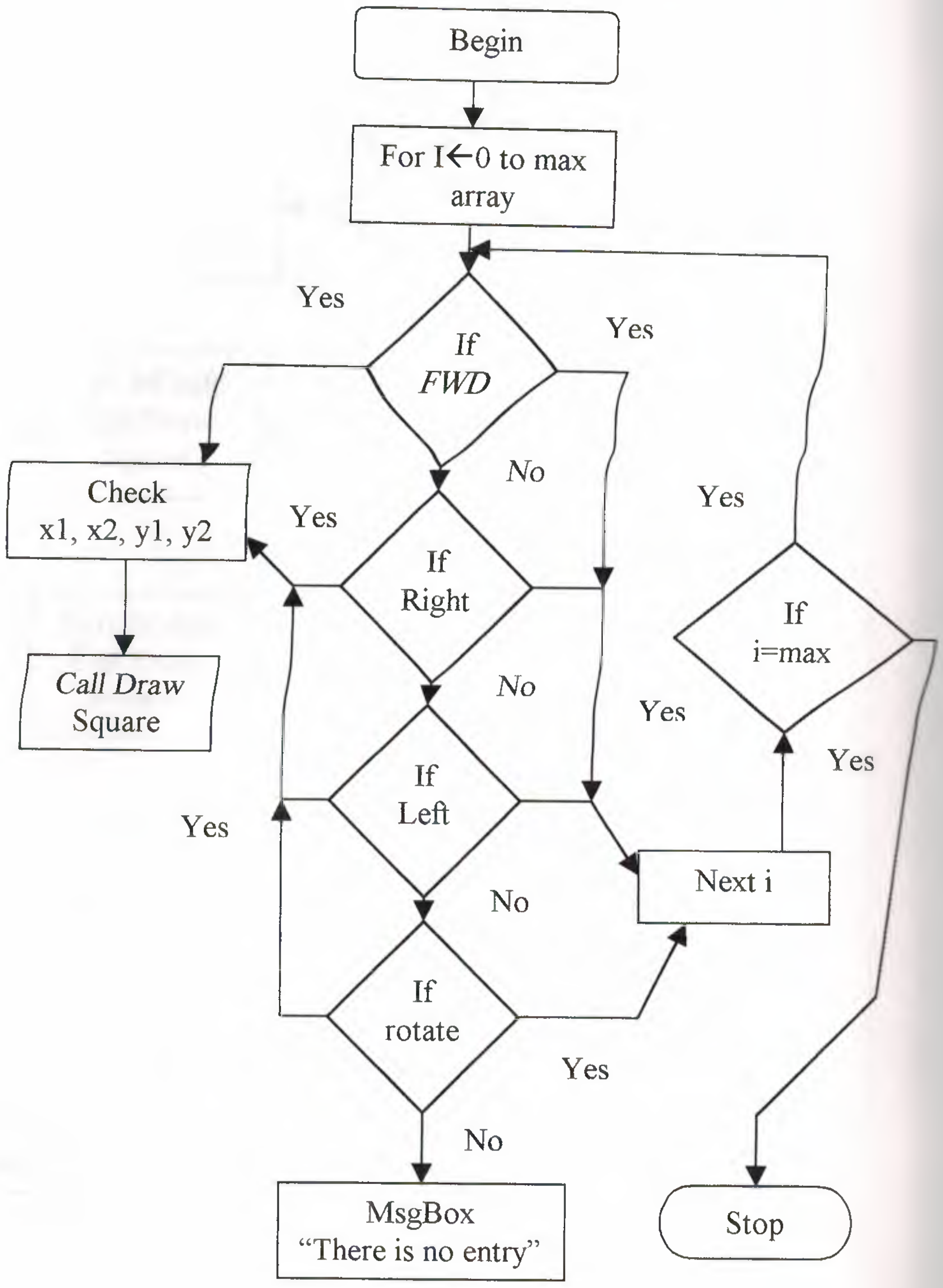


Figure 3.4 Track drawing flow chart

For the check points X1, X2, Y1, and Y2, it should be an algorithm to decide the point of the vehicle, here we shall see in figure 3.5, the main algorithm that illustrates the check point.

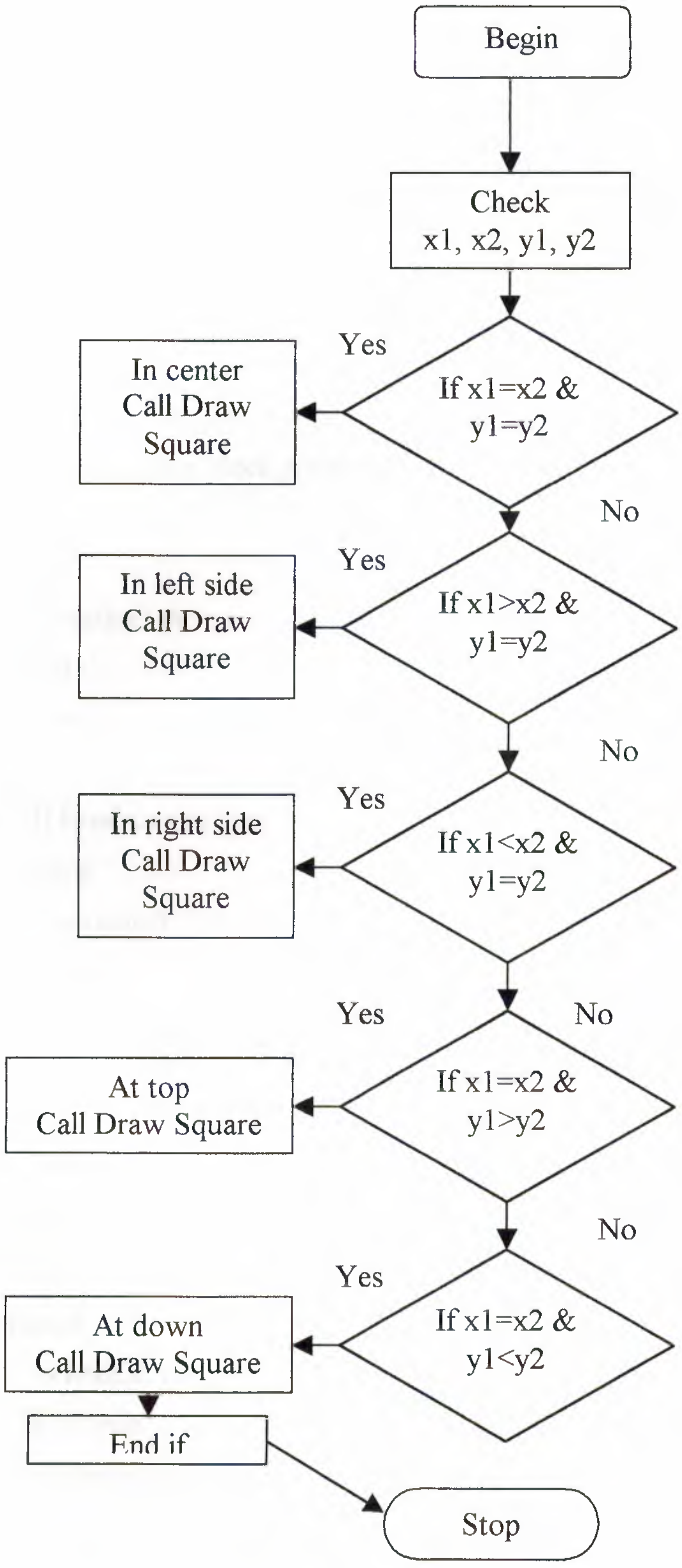


Figure 3.5 Check points flow chart for drawing

For the check points X1, X2, Y1, and Y2, it should be an algorithm to decide the point of the vehicle, here we shall see in figure 3.5, the main algorithm that illustrates the check point.

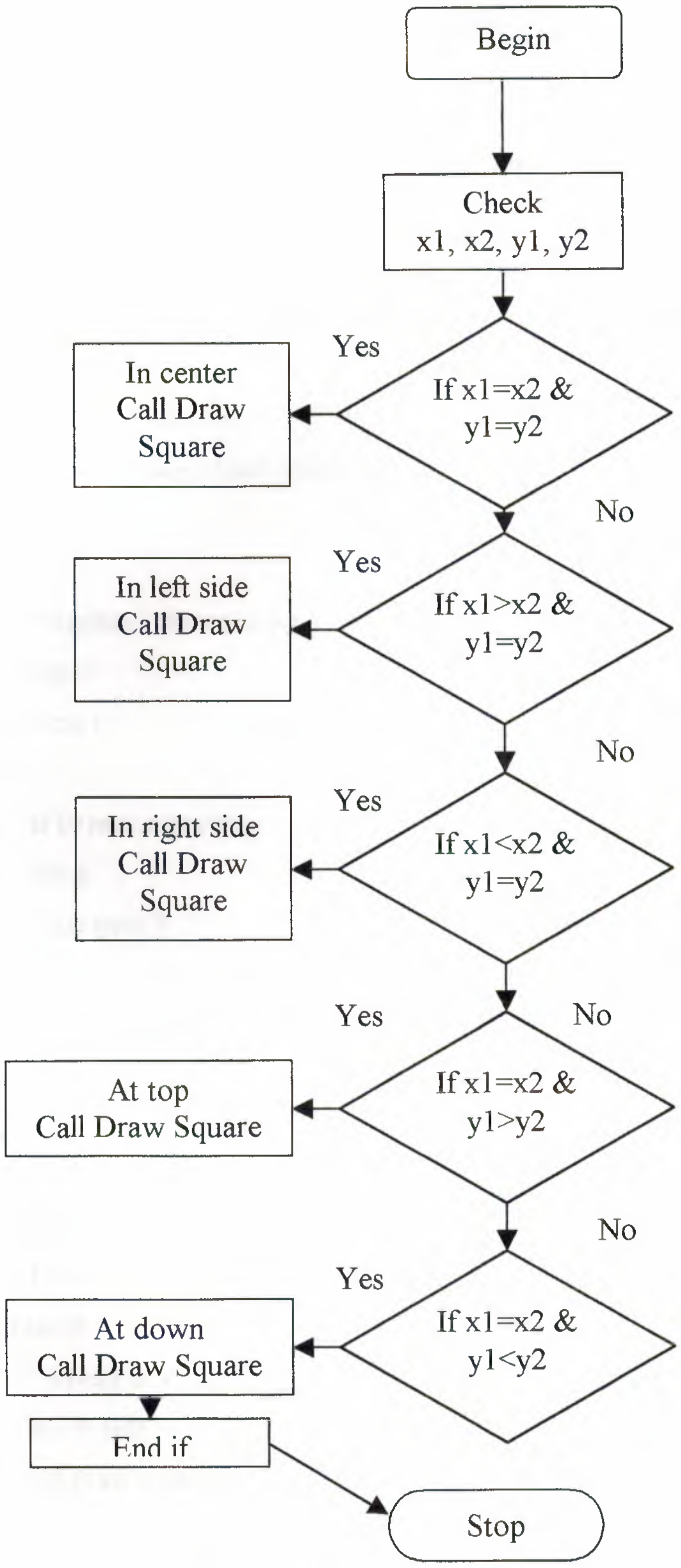


Figure 3.5 Check points flow chart for drawing

The algorithm of the track drawing is as follow:

```
1-  Begin
2-  For I←0 to max array do
3-    If a [I] = FWD then
4-     Call function check points (x1, x2, y1, y2)
5-    Else if
6-      a [I]= FWDright then
7-      Call function check points (x1, x2, y1, y2)
8-    Else if
9-      a [I]= FWDleft then
10-     Call function check points (x1, x2, y1, y2)
11-   Else if
12-     a [I]= Rotate then
13-     Call function check points (x1, x2, y1, y2)
14-   Else
15-
16-     MsgBox" there are not a command"
17-   End if
18-    Next I
19-
20-    If I= max array then
21-     Stop
22-     Else goto 3
```

Check points (x1, x2, y1, y2) function algorithm:

```
1-  Begin
2-  If x1=x2 & y1=y2 then
3-     In center
4-     Call draw square (x1, x2, y1, y2)
5-  Else if
6-       x1>x2 & y1=y2 then
7-     In left side
8-    Call draw square (x1, x2, y1, y2)
9-   Else if
10-    x1<x2 &y1=y2 then
11-    In right side
```

```
12-  Call draw square (x1, x2, y1, y2)
13-   Else if
14-    x1=x2 &y1>y2 then
15-    At top
16-      Call draw square (x1, x2, y1, y2)
17-   Else if
18-     x1=x2 & y1<y2 then
19-    At down
20-     Call draw square (x1, x2, y1, y2)
21-  End if
22-  Stop
```

Now we shall convert the algorithm to sample Visual Basic code, moreover, we will not give the code for every statement, it will be given for the main statements.

The VB code for the first algorithm is:

```
Private sub command1_click ()
For i = 0 To n
'checking if it is forward
If a(i) = FWD
        'checking for drawing the track
         'checking if it is at the center
         '*******************************
          If x1 = x2 And y1 = y2 Then
               y2 = y2 - s
             Call check(i)
          '*****************************

          'checking if it is at the right side
          '*****************************

          ElseIf x1 < x2 And y1 = y2 Then
               x1 = x1 + s
               x2 = x2 + s
               Call check(i)
          '*****************************

          'checking if it is at left side
          '*****************************

            ElseIf x1 > x2 And y1 = y2 Then
               x1 = x1 - s
        x2 = x2 - s
        Call check(i)
    '****************************
    'checking if it is at down side
    '****************************
    ElseIf y1 < y2 And x1 = x2 Then
        y1 = y2 - s
        y2 = y2 + s
        Call check(i)
    '****************************
    'checking if it is at top side
    '****************************
    ElseIf y1 > y2 And x1 = x2 Then
        y1 = y1 - s
        y2 = y2 - s
        Call check(i)
 '****************************************************

  End If
```

This VB code is a sample for track drawing algorithm, moreover, if we change to any direction it will be the same idea, but by changing X's and Y's.

Here call check (i) is used as a call check points but here the return value is (i), this function depends here on (i) to a make a decision to draw.

Furthermore, the main code for track drawing after checking points is shown as follow:

```
Sub check (ByVal i As Integer)
 Picture1.Line (x2, y2)-Step (100, 100), RGB (0, 255, 0), BF
        For t = 0 To 500000
        Next t
        Picture1.Line (x2, y2)-Step(100, 100), RGB(255, 255, 0), BF
```

3.10. Call port out algorithm

Every hardware control system should be connected with a special software that can open port, and close ports. LPT has a unique code for calling e.g. call portout (888,255), the code 888 illustrates the LPT port for output, while 255 illustrates the port number as decimal, and its from 0 to 255.

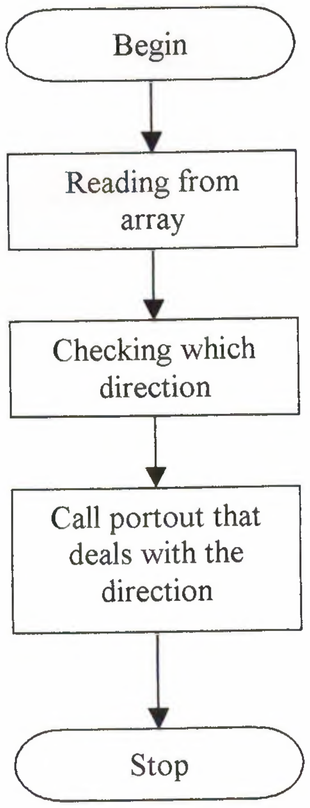


Figure 3.6 Calling output port flow chart

The previous flow chart is shown how to call output port after reading from the array that includes the direction for moving. On the other hand, every motion has its time. This time will be defined depending on the angle that we have suggested particularly, for example, forward motion needs about 3 sec for being compatible with the virtual tracking draw, therefore, to move to right side, it needs about 5sec, These times will be created using for loop or while loop, but in this project we have used the for loop.

Calling output port has been shown in details using the coming algorithm.

The main algorithm for calling output port is written as follow:

```
1-  Begin
2-  For I←0 to max array do
3-    If a [I]= FWD then
4-      For z←0 to T1 do
5-        Call portout 192
6-    Next z
7-    Call portout 255
8-    Else if a [I]=FWDright then
9-        For s←0 to T2 do
10-   Call portout 210
11-   Next s
12-    Call portout 255
13-   Else if a [I] =FWDleft then
14-       For d←0 to T2 do
15-       Call portout 142
16-     Next d
17-      Call portout 255
18-    Else if a [I]= Rotate then
19-      For f←0 to T3 do
20-      Call portout 210
21-    Next I
22-        Call portout 255
23- Stop
```

If we convert this algorithm to VB code it will be as follow:

```
Private sub command2_click ( )
For i = 0 To n
'Checking if it is forward
If a(i) = FWD Or f(i) = FWD
    For z = 0 To t1
    Call PortOut(888, 192)
    Next z
```

The rest of this code is the same by change only the direction and the port, furthermore, the output port for right, left, and rotate are:

1- Right (call portout (888,210))
2- Left (call portout (888,142))
3- Rotate (call portout (888,210))

Note: the main difference between rotate and right is the time.

The rotate direction moves about 180 degree, for that the time for rotate motion is bigger than the right direction by 5 sec or in other words the double.

3.11. Object Detection and Software Control

The developed project contains a detection circuit, this circuit is connected to the computer through the interface circuit, moreover, it needs a simple smart program to work.

Here we shall see the main algorithm for this application, the figure 3.7, illustrates the flow chart.

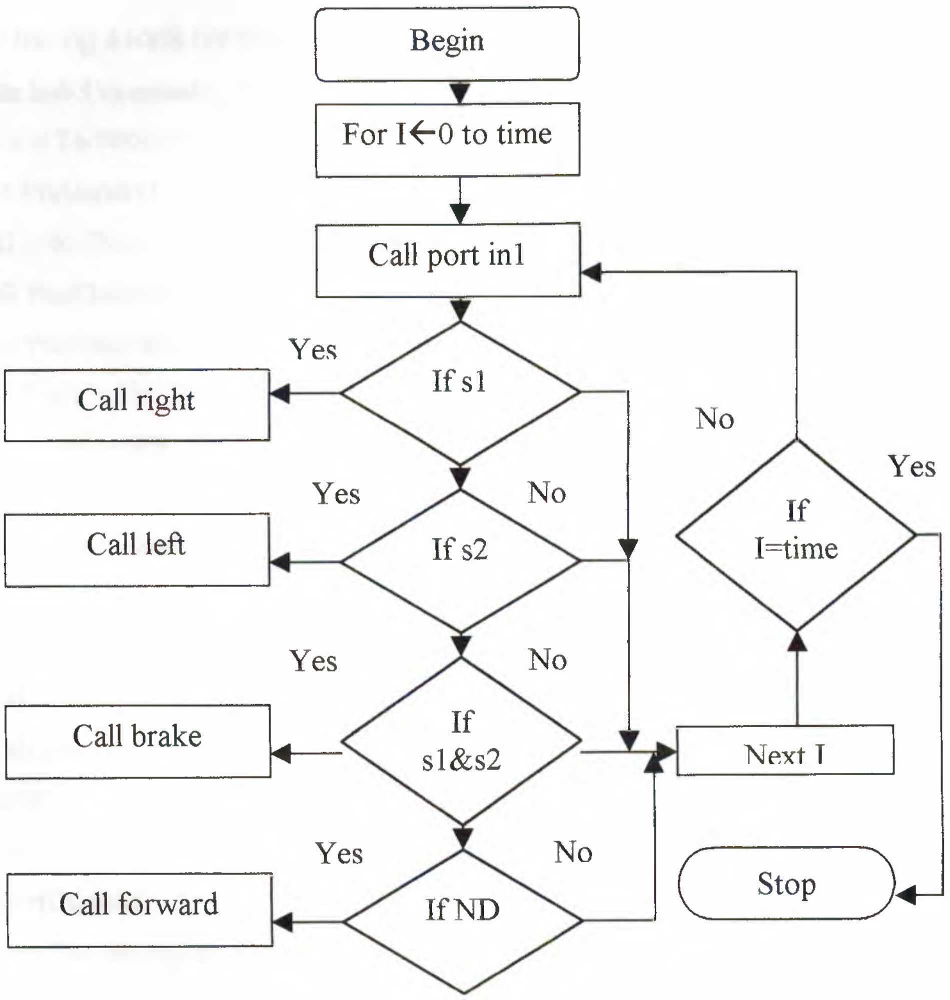


Figure 3.7 Object detection and software control flow chart

The algorithm for the previous flow chart has written as follow:

```
1-  Begin
2-  For I←0 to Time do
3-     If in1=s1 then
4-     Call right
5-    Else if in1=s2 then
6-     Call left
7-      Else if in1=s1&s2 then
8-     Call brake
9-     Else if no detection (ND)
10-     Call forward
11-   Next I
12-   Stop
```

After having a look for the algorithm, let us have a look for the VB code:

```
Private Sub Command1_Click()
For h = 0 To 500000
  in1 = PortIn(889)
  If in1 = 63 Then
    Call PortOut(888, 255)
    Call PortOut(888, 210)
    ElseIf in1 = 191 Then
    Call PortOut(888, 255)
    ElseIf in1 = 255 Then
     Call PortOut(888, 255)
     Call PortOut(888, 142)
      ElseIf in1 = 127 Then
      Call PortOut(888, 255)
      Call PortOut(888, 192)
    End If
 Next h
 Call PortOut(888, 255)
 MsgBox "the driving has finished"
 End Sub
```

Note: number 63 means that sensor1 is working, while number 255 means sensor2 is working, number 191 means that both sensors are working, eventually, number 127 means both sensors are probably not working.

3.12 Summary

In this chapter we have defined the following:

- We have provided an introduction about Visual Basic language and the reasons for using this language.
- Moreover, we have discussed the Graphical User Interface; on the other hand we have shown the designed GUI.
- We have presented a general idea about modules that is used in Visual Basic.
- Some of the technical specifications have been introduced in this chapter.
- Eventually, we have shown the main algorithms, flow charts, and codes that have been used.

CHAPTER FOUR

RESULTS AND MODIFICATIONS

4.1. Overview

In this chapter we shall illustrate some results that we have observed, some of them in software and the others in hardware. Moreover, this chapter will deal with some modifications that have been applied on the project also in software and hardware. Furthermore, we shall suggest some applications that may be implemented using this simple project in the future.

4.2. Results

This project has some results; here we shall divide the results for two parts:

- Software results
- Hardware results

4.3. The Software results

The software is the main tool that has been used in this project; however, the software components should be connected perfectly to avoid any unexpected bugs through running the program.

In software results we have faced many problems that have been solved eventually, especially in dealing with array. These problems will be illustrated in the modification section of this chapter.

The main software result that has been noticed in this project was the robot's routine, therefore, if we repeat the same commands more than one time and not more than four times, the robot will return back to the origin point, but this theory has an exception, the track should be not a step track or a straight track as shown in figure 4.1.

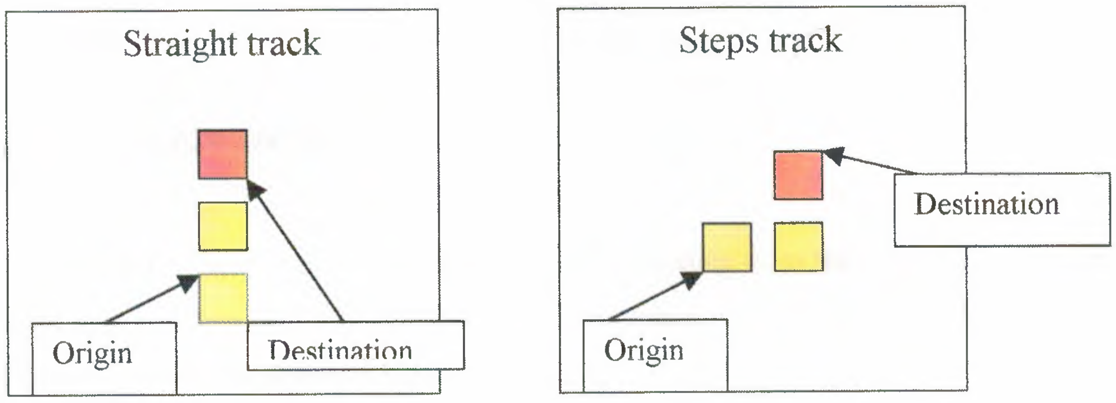


Figure 4.1 The exception tracks

Else, the robot can be returned easily by repeating the commands as shown below in figure 4.2:

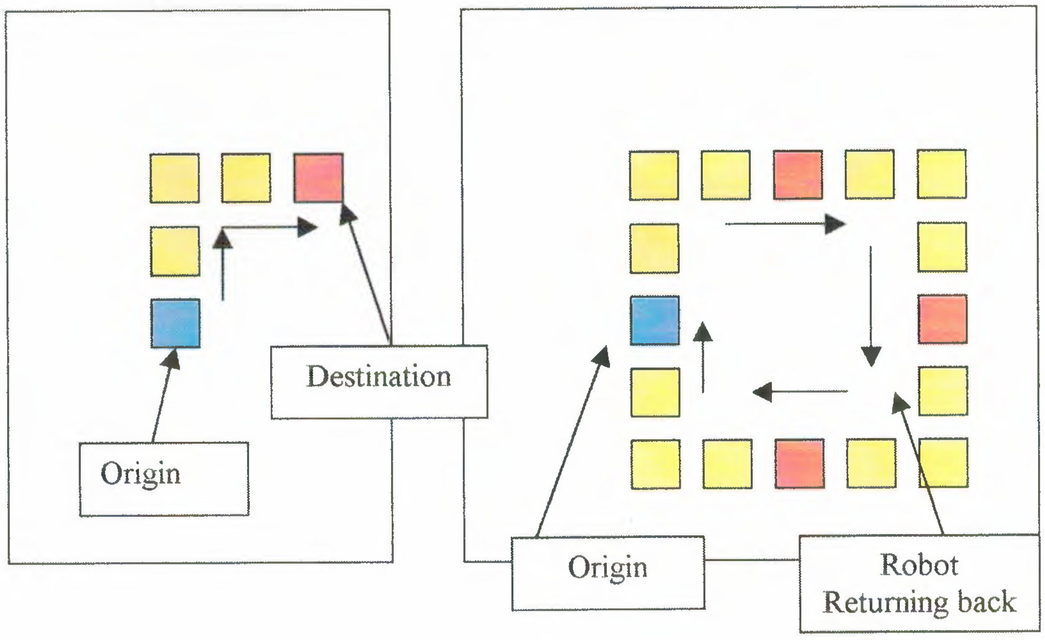


Figure 4.2 The robot returning back to the origin point

This algorithm will be shown in modifications section.

Eventually, the software results is the way for modifying and developing the project, here the previous result is an important idea for this kind of project, therefore, this result might be developed into intelligence software.

4.4. The Hardware Results

The hardware section is an important section after finishing from the software results. In this project we have built a simple circuit that will help us running the project perfectly. This circuit picture is shown in figure 4.3:

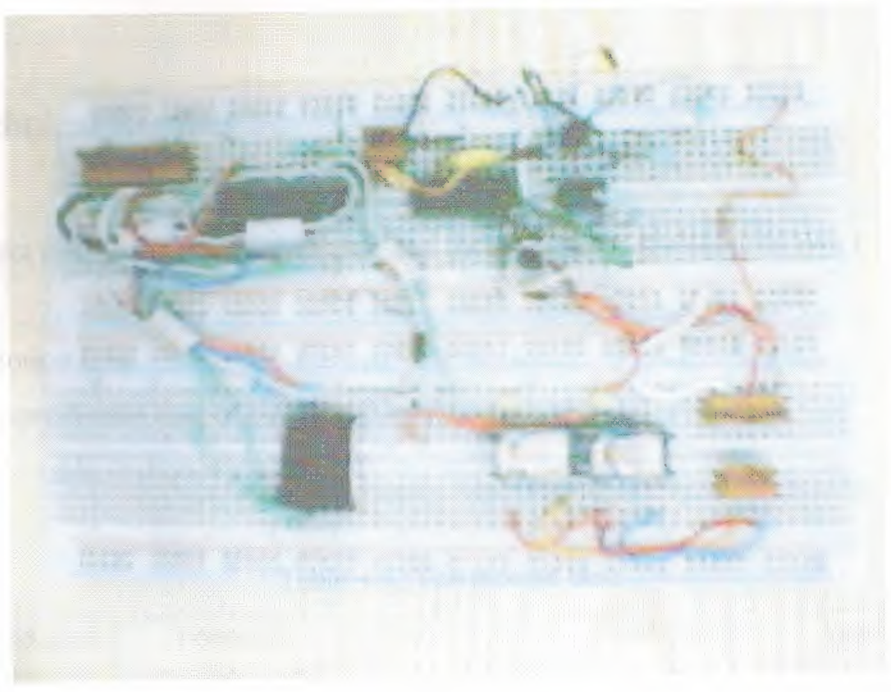


Figure 4.3 The main Interface circuit

The robot machine (a toy military tank) is shown also in figure 4.4

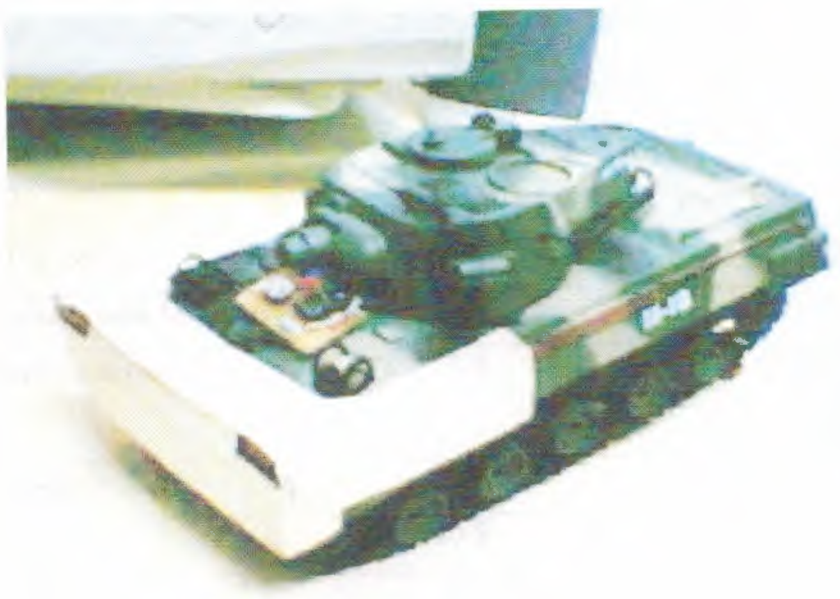


Figure 4.4 The used vehicle

4.5. Technical Results

This project also has a technical results, and these results is divided into two types:

- Motion time
- Approximate cost of building this system

The motion table is shown in table 4.1

Table 4.1. The motion time

Motion type	Motion	Forward motion time	Right motion time	Left motion time	Rotate motion time
Normal motion		Needs about 3 seconds	Needs about 5 seconds	Needs about 5 seconds	Needs about 8 seconds
Return journey		This motion depends on the previous track, between 8 seconds and 30 seconds and may be more if the track complex.			
Auto driving		This motion depends on the detection sensors, the reaction on time, once the detection sensor detects an object.			

The approximate cost of building this system is shown in table 4.2

Table 4.2. The approximate cost

The device	Cost in U.S dollars $	Company
A toy military tank M-18	10 $	China made
Interface circuit	15 $	National / siemens / radio shack
Power supply 5Volts	10 $	Hand made
Sensors	10 $	Radio shack / national
Cables	5$	--------------------
Total cost	50 $	

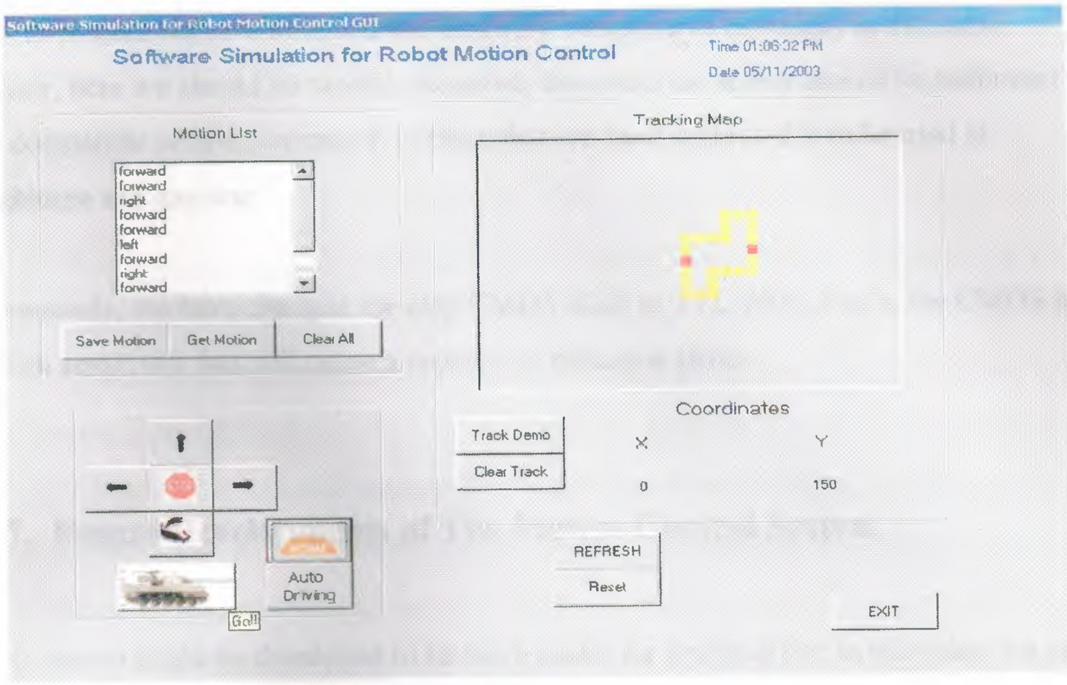
4.6. Modifications

Every project has some modification at the end, this depending on the developing mission, furthermore, the modification may be in software or hardware, in this project we have modified both of them.

4.5.1 Software Modifications

In this project we have modified some codes and the user interface, therefore, some codes have been modified and changed for some errors, for example, if we didn't initiate the array every new order this will cause an error when adding new commands.

The user interface through which robot motion control can be made is shown in figure 4.5:



forward
forward
right
forward
forward
left
forward
right
forward

X 0
Y 150

Figure 4.5 The New GUI

As we can see, a new button have been added, this is a home button, and below we can define the algorithm for this mission:

```
1-  Begin
2-  Reading current x1, x2, y1, y2
3-  If x1=x2 and y1<y2 then
4-  Call return            'here we call this function one time'
5-  Else if x2>x1 and y1=y2 then
6-      For s←0 to 2 do
7-      Call return
8-   Next s
9-  Else if x1>x2 and y1=y2 then
10-     For s←0 to 3 do
11-     Call return
12-  Next s
13- Else if x1=x2 and y1>y2 then
14-     MsgBox" please add right command then try again"
15- End if
16- Stop
```

4.6.2. Hardware Modifications

In this project we have modified the hardware components especially in detection sensor, here we should be careful, therefore, the detection sensor should be calibrated to be compatible with environment, in this robot we have calibrated it to be used at nighttime and daytime

.

Eventually, we have changed the chip CMOS 4066 to TTL 7400, that's, the CMOS has a high sensitivity that will cause a problem in detection circuit.

4.7. Future Developments of The Motion Control System

This system might be developed to be more useful for practical life; in this robot we can modify it in two sections:

- Software Vision: this robot may be controlled using intelligent software depending on A.I algorithms, as in Genetics algorithms, Expert Systems, and Fuzzy logic.

- Hardware Vision: this robot also can be controlled using a micro controller, and we can support this device from the PC by intelligent software. We can add a digital camera to the robot, this camera can send images from real life to the PC, and by using intelligent software (Image Processing Software) the robot can differentiate between objects and save them in its memory. All these operation should be wireless to be more flexible and usable.

4.8. Summary

This chapter can be summarized as follow:

- In this chapter we have defined some of the important results in software and hardware structures.
- We have shown the technical results by illustrate the motion time as well as the approximation cost.
- There are some modifications has been illustrated in this chapter also.
- Finally, we have discussed some of the future visions of this robot.

CONCLUSION

Software simulation for robot motion control is a big subject that may be we could not cover every subject. However, we have presented an introduction about this subject in general in chapter one.

And every practical subject should have a hardware and software setup to give an idea about the real project. In chapter two and three we have defined some of the hardware components beginning from the main interface block diagram and ending by the used interface circuit without forget the detection circuit, furthermore, in chapter three we have illustrated the main software components including the related algorithms, flow charts, and Visual Basic codes.

Moreover, in chapter four we have proved the main ideas by present the results and modifications, as well as the future of the project.

In this project we have achieved our aims, building this simple project and getting an experience in writing a program give us a step forward in designing a good projects, however, we have compared this project with other commercial robots, it might be better in some actions and cheaper to be done.

In conclusion, this project is only a sample of future one, therefore, this type of robots can be developed by supporting it by smart software to be able to use in the reality environment, however, the future one should be wireless and by adding a digital camera it will be more flexible and usable.

REFERENCES

[1] www.alanmacek.com

[2] www.robotics.com

[3] www.robotbooks.com

[4] http://isdl.ee.washington.edu/CE/ConsElectHome.html

[5] http://real.uwaterloo.ca/~robot

[6] http://www.cit.nih.gov/

[7] http://www.najadojo.com/jamie/

[8] M.Morris Mano, *Digital Design*, Prentice/ Hall International, Inc., Englewood Cliffs, New Jersey, USA, 1984.

[9] Stephen E. Derenzo, *Interfacing*, Prentice/ Hall International, Inc., Englewood Cliffs, New Jersey, USA, 1990.

[10] The college, *Microsoft Visual Basic Courseware*, Microsoft Authorized Training Center, Amman- Jordan, 1998.

APPENDIX A

SOFTWARE CODES

1. Open a saved motion

```
'OPENING THE SAVED MOTION

Private Sub open_motion_Click()

Dim i As Integer

Open "c:\project\motion" For Input As #1

   Do While Not EOF(1)

   Input #1, a(i)

   List1.AddItem (a(i))

   i = i + 1

   Loop

   Close #1

   MsgBox "GETTING THE  SAVED MOTION"

End Sub
```

2. Save a motion to a file

```
'SAVING THE MOTION

Private Sub save_motion_Click()

Open "c:\project\motion" For Output As #1

     For i = 0 To n
```

```
    If a(i) <> "" Then
    Print #1, a(i)
    End If
    Next i
   Close #1
   MsgBox "The motion is saved"
End Sub
```

3. Reset all

```
' RESET THE MAIN VALUES
Private Sub reset_Click()
q = MsgBox("Are you sure you want to reset all value?", 65, "reset all values")
If q = 1 Then
'intiate the values
X = 0
Erase a
x1 = 2280
x2 = 2280
y1 = 1560
y2 = 1560
 Label1.Caption = 0
 Label2.Caption = 0
 Picture1.Cls
 List1.clear
  End If
End Sub
```

4. Loading the main form

```
'LOADING THE FORM

Private Sub Form_Load()

last_element_index = -1

x1 = 2280

x2 = 2280

y1 = 1560

y2 = 1560

Call PortOut(888, 255)

Show

frmSplash.Show

clear.Caption = "Clear All"

open_motion.Caption = "Get Motion"

save_motion.Caption = "Save Motion"

reset.Caption = "Reset"

Command1.Font = "tahoma"

Command1.Caption = "Auto Driving"

Label1.Caption = ""

Label2.Caption = ""

Label3.Caption = "X "

Label4.Caption = "Y "

Label7.Caption = "Motion List"

End Sub
```

5. Time and Date

```
'SHOW THE DATE AND TIME

Private Sub Timer1_Timer()

Label5.Caption = "Time" & " " & Format$(Now, "hh:mm:ss AM/PM")

Label6.Caption = "Date" & " " & Format$(Now, "mm/dd/yyyy")

End Sub
```