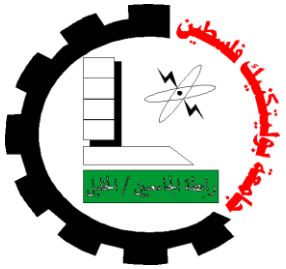
Palestine Polytechnic University



College of Engineering and Technology

Electrical and Computer Engineering Department
Mechanical Engineering Department

Graduation Project

Chess Playing Robot

Project Teams:

Mechatronics Team:
Mohammad Alsharif
Rami Yaghmour
Safwat Haddad

Computer Systems Team:
Ahmad Alsaheb
Mahmoud Jallaq

Project Supervisors:

Dr.Yousef Sweiti Prof. Dr. Karim Tahboub Dr. RadwanTahboub

Hebron-Palestine

2011

Palestine Polytechnic University

Hebron-Palestine
College of Engineering and Technology
Department of Electrical & Computer Engineering
Department of Mechanical Engineering

Chess Playing Robot

Project teams:

Mechatronics team:
Mohammed Alsharif
Rami Yaghmour
Safwat Haddad

Computer Systemsteam:
Ahmad Alsaheb
Mahmoud Jallaq

According to the directions of the project supervisor and by the agreement of all examination committee members, this project is presented to the department of Mechanical Engineering and Computer Engineering at College of Engineering and Technology, for partial fulfillment Bachelor of engineering degree requirements.

Supervisor Signature

…… .

Supervisor Signature

……

Committee Member Signature

Department Head Signature

Abstract

Since hundreds of years, human played with small stones for entertainment, cleverness and challenge, one of those games was the chess, since more than 1500 years chess has been considered as an indication for genius and leadership.

The project is an industrial robot arm from kind SCARA (Selective Compliance Assembly Robot Arm) that can play chess game against a human, this Robot can think, make decisions and also it can move the chess stones between the different locations. SCARA has three Degrees of Freedom (DoF) which are two rotational motions and one prismatic, this enables the Robot to reach any of the 64 locations on the chess board in order to pick and place the stones.

The Electrical board (E-Board) on which the robot picks and places the stones is connected to the computer using interfacing cards also, 64 switches are put at the center of each square in the board to sense each movement done by the human or the robot.

A suitable chess engine on the computer is used as the software that manages the game. A monitor is placed beside the robot to show the game steps.

Basically the project depends on two software, chess game and MATLAB. MATLAB is the software that controls the robot; it includes the controlling algorithm and the motion algorithm to control the robot's joints.

Dedication

To our parents
To our families
To our supervisors
To our friends for their support
To our teachers and professors for their help
To our University Palestine Polytechnic University

List of contents

Chapter 2: Design Options 17

Chapter 3: System Design 41

List of Figures

List of Tables

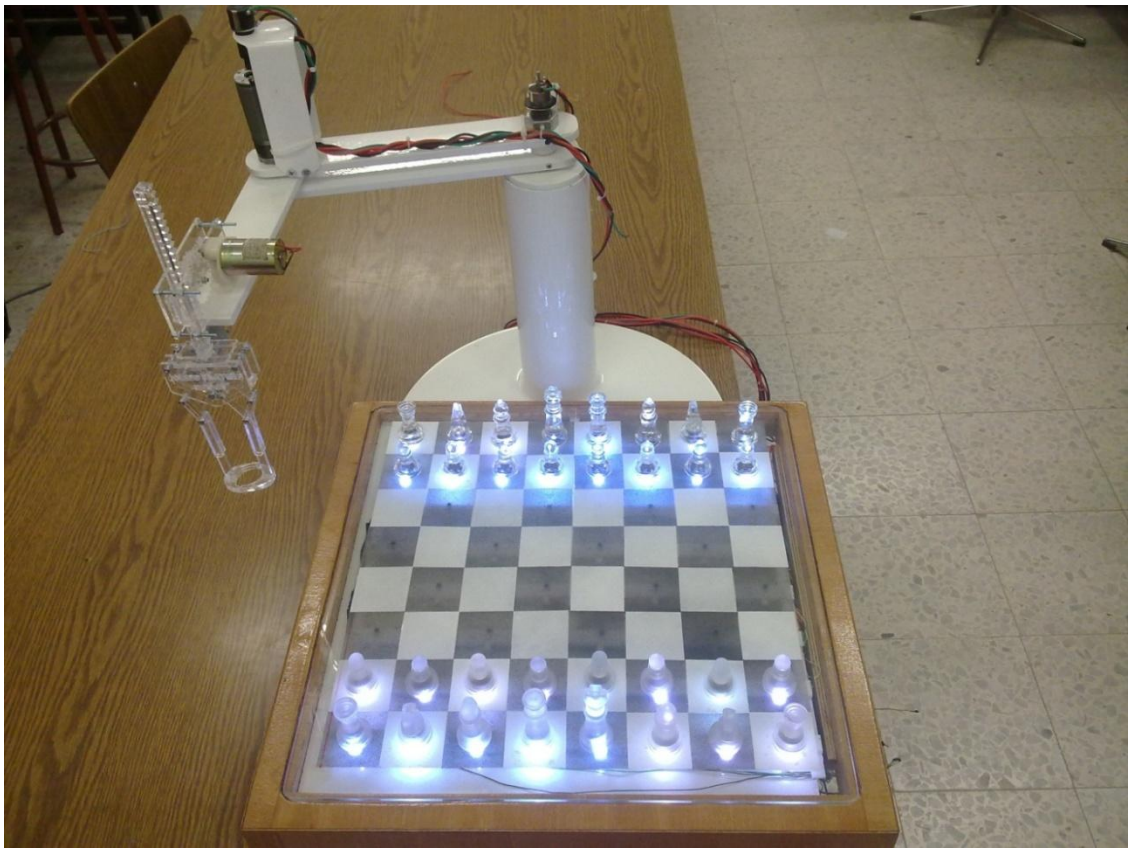
CHAPTER ONE 1

Introduction

1.1 Overview

Chess Playing Robot (CPR) against humans is a very interesting field of study. In this application the user can play chess against a robot arm on a real physical chess board and rather than performing the game as a software installed on the computer.

The robot shown in Figure (1.1) is of the SCARA (Selective Compliance Assembly Robot Arm) kind which has three links and three joints, two revolute and one prismatic, the SCARA mechanism and motion is very similar to the human shoulder and arm but in a plane, to enable the robot to move above the board using the first two links, and to pick and place the stone using the third link.



Figure(1.1):SCARA Robot and Chess Board

The board can be considered as an organ of the robot arm, thus the robot will feel with the board as well as the stones placed on it. This can be achieved by connecting the board physically with the computer and then with the robot, by the use of interfacing cards such as the Data Acquisition Cards and the keyboard encoder cards. The interfacing between the three pieces of the project (Board, PC and the robot) can be done.

When a stone is put on the board, the magnet which is fixed under it will affect the Hall Effect sensor under the board, this will generate a signal that can be read by the keyboard-encoder interfacing card, and then this piece of information will be translated to the chess engine on the PC. Now it is the turn of the robot to play its game, the chess engine will decide its next movement. This order will be sent through local network to be by the mind of the robot, the MATLAB software, which calculates the motion of the robot arm.

Using MATLAB/Simulink and data acquisition cards (DAQ) these mathematical calculations will be converted to motion, enabling the robot to do the action decided by the chess engine, then the turn of the player will come to repeat the loop.

The monitor of the PC is placed beside the robot; its function is to show the whole process happening on the chessboard. In addition, it shows the countdown timer and the level of difficulty that can be chosen using buttons beside the board.

The timer will give the player a specific time that if exceeded; the player's turn will be ignored and converted to the robot. This interval of time depends on the difficulty level chosen by the user.

Lighting LEDs will interact with the game; they are placed as one LED under each square. The LEDs will tell the user about the possible positions for the hold stone to place on.

This flow of information between the different pieces of the project is shown is figure (1.2).

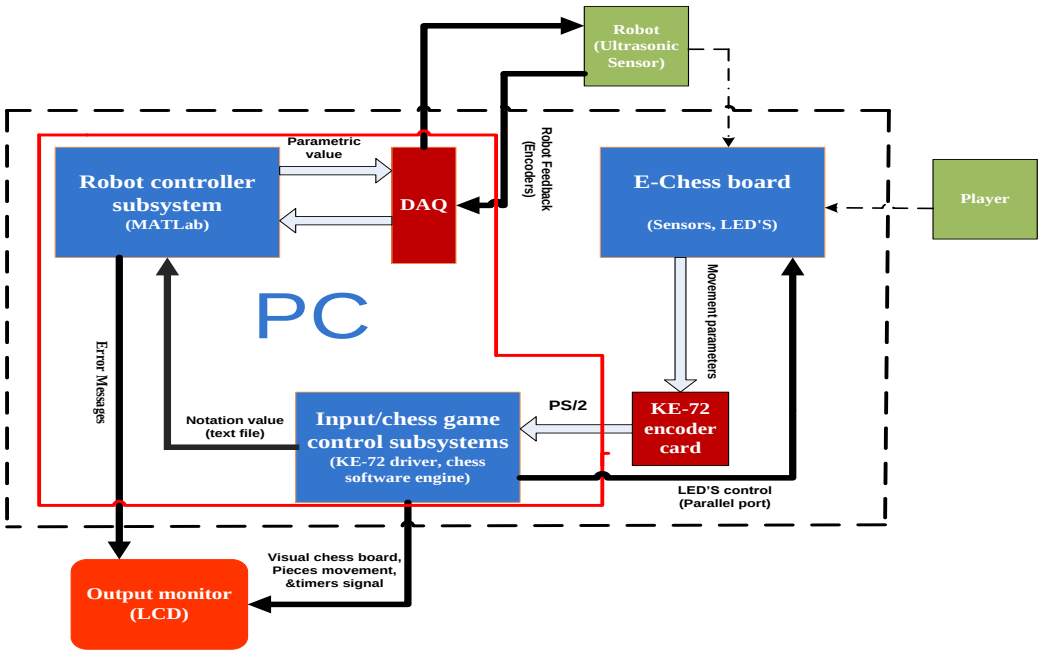


Figure (1.2): Chess Robot Playing main block Diagram

1.2 Project Objectives

The main objectives of this project are:

1. Use the robotics in colleges and universities as an educational tool because it combines many technological systems.

2. Highlight the integration between the software programming and mechanical designing.

3. Using this project as starting point for other projects.

4. Building an interfacing environment between a Microsoft product (C#) and other software such as MATLAB.
   .

1.3 System Requirements

This section describes the functional requirements and the nonfunctional requirements. The functional requirements provide the functions and the operations that the system must satisfy. The nonfunctional requirements are related with nonfunctional constraints, such as the time and budget constraints, efficiency and ease of use. The next section will discuss the specifications of the functional requirements.

1.3.1 Functional Requirements

1. A mechanical Robot arm with its robot controller algorithms.
2. A sensory chessboard that can sense any movements occur.
3. Light chess board's squares.
4. Interacting chess board and Robot subsystem through customized chess software.
5. Cheating discovery algorithms
6. Multi-level game.
7. Historical archiving of the game.
8. Game's Timer.
9. Educational capabilities

1.3.2 Non Functional Requirements

1. Playing against the robot with no faults.
2. Accurate and error-free game.
3. Buttons for starting and choosing the game's difficulty.
4. Safety with a collision-free game.
5. Robustness of the structure
6. Beautiful appearance of the robot.
7. Fast robot-response and fast game with a countdown timer.
8. Human interface using a monitor and board-place LEDs.
9. Developing the project to be considered as a commercial product.
10. 3D graphical user interface

1.3.3 Requirements Specifications

1. This robot arm is responsible for moving stones from one position to another ,the manipulator should follow a specific path, depending on data received from chess engine.
2. Magnetic fields affect the board squares, sensing determine move type, up or down, and move source and destination.
3. Board's Squares light to help player and guide him.
4. Chess software to interface board events with robot controller.
5. Cheating discovery

1.4 Recognition of the Need

Chess playing robot can be considered as one of the important scientific applications because it combines many knowledge fields in one work. First the science of Robotics, its motion and programming, also the Control methods used in the robot such as the cascade, state feedback and digital control using high level programming languages such as MATLAB.

All these aspects make this project a distinctive one, the interfacing techniques used to connect the board with the computer and then translating the action in the chess engine into a motion in the robot can be considered as excellent exercises for both Mechatronics and Computer engineering students. On the other hand, this project can be converted into a commercial product in local and universal markets.

Finally, Chess Playing Robot can be adapted in the colleges and universities as an educational tool because it combines many technological systems such as:

- Sensors and Actuators
- Signal processing
- Embedded systems
- Hardware-Software interfacing
- Programming languages
- Mechanical Design

Another advantage for this study is that it can be a graduation project for later students; they can add a lot of things to it, for example the vision system. They can also improve it, trying other kinds of robots, using other sensing techniques in the board, or they may implement the chess engine and the other software on microcontrollers such as PIC or DSP.

1.5 Project Description

As previously mentioned, this project consists of three main pieces, the board, the PC and the SCARA robot. These three divisions are very general and should be anatomized separately. For example the robot consists of three motors, the gripping mechanism, the ultrasonic sensors for safety and the mechanical structure.

The PC contains the chess engine and the robot operating software, which is MATLAB, the PC contains the interfacing cards for both the robot (PCI-6024 DAQ) and for the chessboard (KE72).

1.5.1 Chess Board

The used board is a wooden 50cm*50cm with approximately 8cm height stones, it is fixed on a table between the robot and the player, and connected with the computer using the key board encoder KE72 that will be discussed in chapter three in detail, each square in the board contains a sensor that is activated when a stone is put over it.

This switch will generate the signal to the key board encoder KE72.This signal should inform the PC whether there is or there is not a stone in each square.

The 64 squares are addressed using (0X88) representation[11], and named using the intersection of the row and the column as in Figure (1.3).

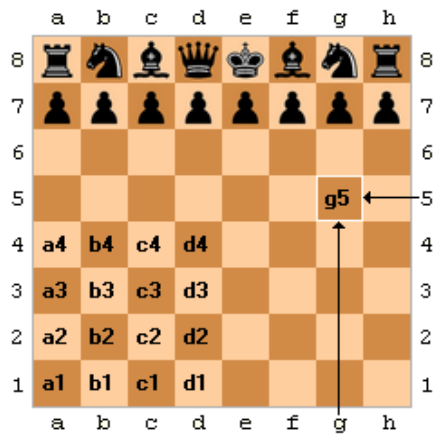


Figure (1.3): Chess Board divisions

1.5.2 Chess Engine/Programs

The Microsoft .Net Framework is a software framework for Microsoft windows operation system. A Framework is a set of class libraries that provide the developers with the common system functions and services that can be used by a programming language. But the .NET Framework extends this concept. The .NET Framework is a large set of class libraries that can be used for many programming languages, like Microsoft's C#, Visual Basic, Managed C++ and more.

C# is an Object Oriented programming language of its own, and it has its own syntax, control structures, and construction building statements. At the same time, it has been created to be used with the .NET Framework, and the C# compiler generates code that runs under the .NET Environment [23].

The used chess software is built using the C#.Net technology, its integrated classes are used to manage the chess board inputs and the intermediate process between MATLAB files and C#.NET[23].

From the name of the project, using chess game program is essential as it will be the brain of the Robot by giving it the legal movement against the human player. It also helps the user playing by showing him the possible movement for each piece and discovers any non-legal or unknown movement.

The chess engine that is selected must be customized so as to satisfy the project's requirements and needs. These requirements and properties that should be available on the chess game are as follows:

- Ability to play against the Computer: this requirement is very important since the legal movement that will be given to the Robot to play against the player will be from the intelligence side in the chess engine.
- Many preset difficulty levels, or an option to play using timers: the player may want to play against the Robot with difficult level or in combination with timers.
- Ability to help the player by showing all possible movements when the user selects any piece.
- Using chess engine that is implemented according to the chess game rules and standards.
- Real time computer thinking to make the game as fast as possible.
- Graphical chess board and user interface.

.

1.5.3 MATLAB

MATLAB (*Mat*rix *lab*oratory) is a numerical computing environment and fourth-generation programming language. Developed by Math Works, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, and Fortran.[16]

MATLAB is used to program the Robots motion, this software acts as a translator, it receives the order from the chess engine and translates it into voltages supplied to the three motors, and thus motion can be performed.

1.5.4 DAQ Card

Data acquisition (abbreviated DAQ) is the process of sampling of real world physical conditions and conversions of the resulting samples into digital numeric values that can be manipulated by a computer. The components of data acquisition systems include Sensors, Signal conditioning circuitry and Analog-to-digital converters.

DAQ card is used as interfacing card in the project, it is fixed on the PCI slot in the PC and programmed using MATLAB/Simulink, the core function of this card is to

convert the analysis done on MATLAB into voltages supplied to the motors, or in other words to connect the Robot with the PC.

1.5.5 xPC target

The xPC target technique is a solution for prototyping, testing and developing real time systems, using standard PC hardware and its peripherals such as DAQ cards. This technique is a part of MATLAB software provided by MathWorks Company. In particulate xPC is a toolbox with MATLAB/Simulink.

In xPC target technique two PCs are used, host and target. With the host PC, one can design the controller, simulate it and download it to the PC. The target PC which is connected to the controlled plant is just used to run control functions in real time and monitor the controlled application.

xPC target technique is considered as excellent solution for educational and rapid prototyping purposes due to the following facts:[27]

- Changes and modifications in the controller design are easily introduced to the host PC, and the modified controller is downloaded to the target PC almost with no effort. Online tuning of some parameters is also possible.
- Hard real-time requirements can be satisfied since the target PC processor is fully dedicated for running the controller.
- Host and target PC can be connected serially, through a network or even through the internet. Furthermore the target PC can operate alone without any connection with the host.
- xPC target technique supports a wide range of DAQ cards and IO boards, which gives the designer a high level of flexibility to choose the suitable hardware.

This technique makes the system faster, more accurate and easier to modify.

1.5.6 Mechanical Structure of SCARA

In general, SCARA (Selective Compliance Assembly Robot Arm) robot, is a type of horizontal drive, and is used in the arrangement of parts to a printed wiring board and a product assembly, and is usually controlled by PID compensator to attain an exact moving. In controlling the SCARA robot, the arm head is moved by motors and the system is a MIMO (Multiple-Input Multiple-Output) system, so each link has the required accuracy and incoherency.

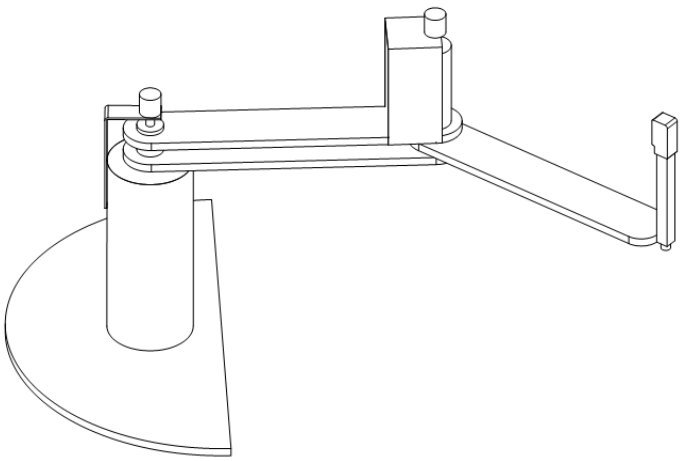


Figure (1.4):SCARA Robot

In order to achieve the motion two rotational and one linear DC motors are used and will be discussed in next section.

1.5.7 DC Motors

DC motor is an electromechanical component that yields a displacement output for a voltage input. DC motors are used in either open loop or closed loop application. Most used in robotics is closed loop. Encoders, potentiometers, or tachometers are attached to the output of the motor to form the feedback loop.

DC motors power supply are connected to the DAQ card, when the chess engine decides to move the stone in B3, MATLAB will receive this address and will do quick analysis, DAQ card will receive this result and supply the motors.

1.5.8 Gripper

Robot grippers are a type of tooling used to move parts from one location to another. They can be driven hydraulically, electrically, and pneumatically. Dual grippers offer quick part changeovers. Vacuum grippers and electromagnets also offer other benefits.

Robot grippers are often selected by the amount of force that the gripper applies to a part. In Chess Playing Robot application the load to be moved is very light; each stone is less than 40gm, so two racks and a pinion mechanism is an adequate selection to achieve this aim.

1.5.9 Ultrasonic Sensors

Ultrasonic sensors (also known as transceivers when they both send and receive) work on a principle similar to radar or sonar which evaluate attributes of a target by interpreting the echoes from radio or sound waves respectively. Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor. Sensors calculate the time interval between sending the signal and receiving the echo to determine the distance to an object.

Placing an ultrasonic sensor at the nearest point of the robot (which is the end of the second link) to the human enables the robot to sense any obstacle, thus if by mistake the human puts his face or hand in the robot's path, the robot will sense it, stopping and going back to its home position.

1.5.10 Potentiometers

A potentiometer is a three-terminal resistor with a sliding contact that forms an adjustable voltage divider. Potentiometers are commonly used to control electrical devices such as volume controls on audio equipment.

Two potentiometers are needed to be fixed on the two motors' shaft in order to measure the links positions. If a sudden energy loss happens, the system will identify these positions immediately when power turns on i.e. the potentiometer may be considered as an absolute sensor.

Figure (1.5): Multi-turn potentiometer

1.5.11 Controllers

The controller is the "brain" of a servo system. It is responsible for generating the motion paths and for reacting to changes in the outside environment. Controllers can be something as simple as an ON/OFF switch or a dial controlled by an operator. They can also be as complex as a multi-axis controller that actively servos several drives as well as monitors I/O and maintains all of the programming for the machine.

Typically, the controller sends a signal to the drive; the drive provides power to the motor; and the feedback from the motor is sent back to the controller and drive. Feedback from the load is also routed to the controller. The controller analyzes the feedback signal and sends a new signal to the amplifier to correct for errors. The controller is considered to be the intelligent part of the servo, closing the velocity and/or position loops while the amplifier closes the current loop.[5]

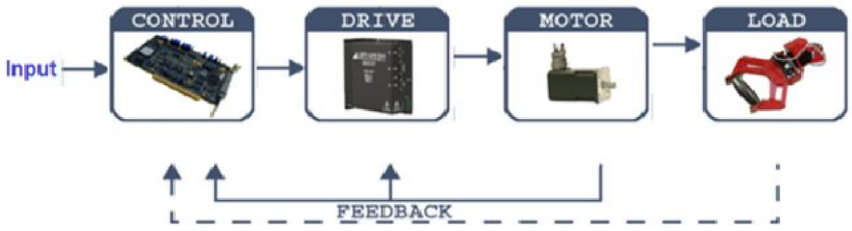


Figure (1.6): A Servo System

There are two types of classical controller in the control system; continuous controller and digital controller. Continuous controllers are divided into various categories as follows:

1- P controller
2- I controller
3- D controller
4- PD controller
5- PI controller
6- PID controller.

Controllers used in SCARA robot maybe of the PID kind or other types to control both the speed and the steady state accuracy of the robot, also the controllers take part in designing and planning the trajectory of the robot while moving, its velocity and acceleration.

1.6 Previous Studies

1-Lego Chess Robot

By Stewart Gracie, Jonathan Matthey, David Rankin, KonstantinosTopoglidis, University of Glasgow 2004/2005.This project was a Cartesian Lego robot on rail connected to the computer using infra-red transition. The chess board was a physical-USB with 64 switches. LEDs and LCD were used showing different levels of difficulties. The gripper was a Lego tripod one. [1]

2-Chess Playing Robot

Del Hatch made a chess playing robot, he bought the three pieces of the project which were the sensory chess board from ”Digital Game Technology”, the articulated robot “SIR-3” by Scien-Tech and the chess engine “GNUChess”. [2]

3-MCU Chess

The designers used a Cartesian robot on rail with 4 stepper motors controlled by PWM (Pulse Width Modulation). The project was implemented on microcontroller (not on a computer). [3]

4-Autonomous Chess Playing Robot

The designers used an articulated robot with 4 revolute joints and potentiometers for feedback. Interface was done using ISA cards, controlling the robot using vision system and programming on Turbo C.[4]

5-MarineBlue: A Low-Cost Chess Robot

David URTING and Yolande BERBERS, KULeuven, Department of Computer Science, Celestijnenlaan 200A, B-3001 LEUVEN, Belgium. Using vision system and a robot “Robix RC_6” with bipod gripper and implementation on computer they build their project.[5]

1.7Project Scheduling and Cost Estimation

The Project's working processes in the first semester was planned to achieve a parallel working between the two teams, the Mechatronics team, and the Computer team, this guaranteed to achieve so much work and to spread the knowledge between the two teams.

This plan focused on the designing steps and choosing the project's parts after doing the required analysis. This plan is shown in table (1.1).

Table (1.1): Timing Table

shTasks week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Choosing the project.														
Gathering information about the project.														
Mechanical analysis.														
Electrical analysis.														
Robotics analysis														
Board design.														
Writing the documentation														

The cost of the project at this level (noncommercial level) is precisely studied, most of the project parts are available in local market but some are not, such as the PCI-6024 DAQ card and the key board encoder KE72 which are relatively expensive.
The Parts and its price on their land are listed in Table (1.2).

Table (1.2): Parts' Price Table

Components	**Cost($)**
DAQ-6024	1500
computer	200
Magmatic sensors	100
Mechanical structure	200
Two DC motor	200
Total cost	2200

1.8 Summary

Chess Playing Robot is a scientific application that combines hardware with software, programming with designing and entertainment with science. The integration and synchronization between Sharp Chess, C# and MATLAB is important.Also the integration between the physical parts such as the mechanical structure, motors and the potentiometers is important, but the core of this importance is beyond the integration between the physical parts and the software on the PC.

This chapter presented the general ideas of chess playing Robot, the system objectives, system description, and recognition of the need.

Chapter two talks about design options, discussing and comparing these options with each other to find the most suitable design to be considered in the project.

Chapter three presents the system design and requirements, designing the SCARA Robot including its mechanical structure, actuators and gripping mechanism. In addition, it talks about the chess engine algorithms and chess board design.

CHAPTER TWO 2

Design Options

2.1 Introduction.

2.2 Structure Design Options.
- 2.2.1 Robot Options.
- 2.2.2 Safety Sensor Options.
- 2.2.3 Motors & Actuators Options.
- 2.2.4 End Effecter's Gripper Options.

2.3 Chess Engine design Options.
- 2.3.1 SharpChess game.
- 2.3.2 Chess game (Chris Meijers 2001).
- 2.3.3 Chess game implemented in UNIX.
- 2.3.4 Chess games that is implemented using other programming languages.

2.4 E-Board Design options.
- 2.4.1 The Keyboard encoder card KE72.
- 2.4.2 Scanning circuit using AND's gates.
- 2.4.3 Scanning circuit using transceivers.
- 2.4.4 Scanning circuit using Data Acquisition Cards (DAQ).

2.5 Switches design Options.
- 2.5.1 Photodiodes.
- 2.5.2 Push-Button.
- 2.5.3 Fixed Poles Switch.
- 2.5.4 Magnetic Switch.

2.6 Board's LEDs.
- 2.6.1 Eight vertical –Eight horizontal LED's.
- 2.6.2 One LED under each square.

2.1Introduction

The ‘Chess Playing Robot’ project can be designed and achieved using different methods, the chess engine, the board; the actuators and even the robot itself have many options to be selected from. Thus, each part in the project should be studied deeply in order to decide the most suitable choice.

In the following sections, these options are listed, showing the advantages and disadvantages of each, authorization and no authorization reasons.

Part A:

This chapter is divided into two parts, the first part (Part A) talks about the research done by the Mechatronics Engineering team such as the structure design, kind of robots suitable for such an application, the actuators and the safety sensors. Part B discusses the research done by the Computer Engineering team and is listed after Part A.

2.2 Structure Design Options

This section talks about the different design options available and convenient to be adapted in such an application, the robot itself, the safety sensors, the actuators and the gripping mechanism all have variety in types with advantages and disadvantages for each.

2.2.1 Robot Options

Many kinds of robots can be used to pick and place objects in industry or for entertainment. These robots differ from each other in structure, shape, size and number of degrees of freedom and in work space. In Chess Playing Robot one may be interested in shape, size, simplicity and efficiency. Minimal number of degrees of freedom should be taken in consideration, for these reasons and others the best choice was to use SCARA robot.

2.2.1.1 SCARA Robot

The SCARA acronym stands for Selective Compliant Assembly Robot Arm or Selective Compliant Articulated Robot Arm.

SCARA robot shown in figure (2.1) is a 3-axis manipulator consists of three joints, two revolute and one prismatic, the z-axes of all three joints are parallel, they can move to any X-Y-Z coordinate within their work envelope as shown in figure (2.2). There may be a fourth axis of motion, which is the wrist; it is not included in this application because there is no need for it.

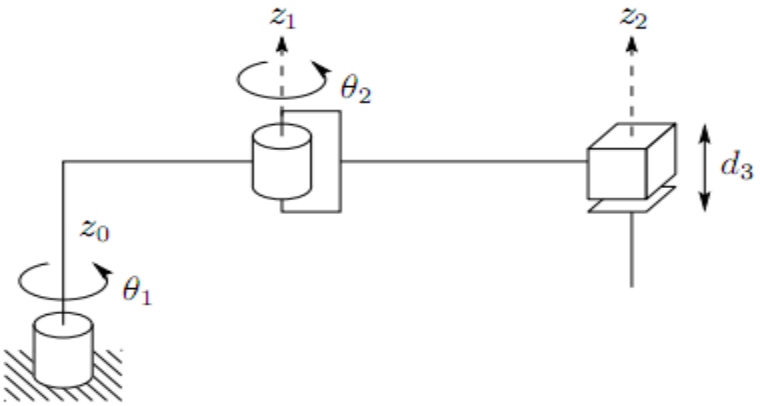


Figure (2.1): The SCARA manipulator.

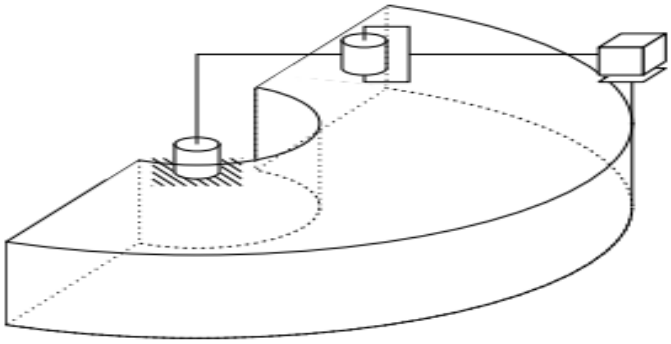


Figure (2.2): Work space of SCARA manipulator.

From Figure (2.2) one can notice that the workspace of SCARA robot is almost matching the shape of the chess board and include it. Besides, it is the most compacted robot in size which makes this option the best one.

2.2.1.2 Cartesian Robot:

Cartesian robot, also known as a gantry, Cartesian coordinate robot, is a common type of industrial robot.

This is a manipulator whose first three joints are prismatic as shown in figure (2.3). It consists of three arms, each arm can move only along a two dimensional axis (forward & backward or up & down). It is constructed in a way that allows the robot to utilize the motions of all three arms to reach various points in a three-dimensional space with relative ease. The work space is shown in Figure (2.4). And as a result of this feature, its most common application is as a CNC machine. The size of these arms can vary, depending on the application used for.

The main advantage of the Cartesian robot is that all of its three joint are prismatic (linear not rotational), and this simplifies the kinematic analysis of this manipulator.

It has relatively a huge size when it is needed to cover a workspace such as the chess board, thus the sight of this robot besides the board and the pieces will be uncomfortable.

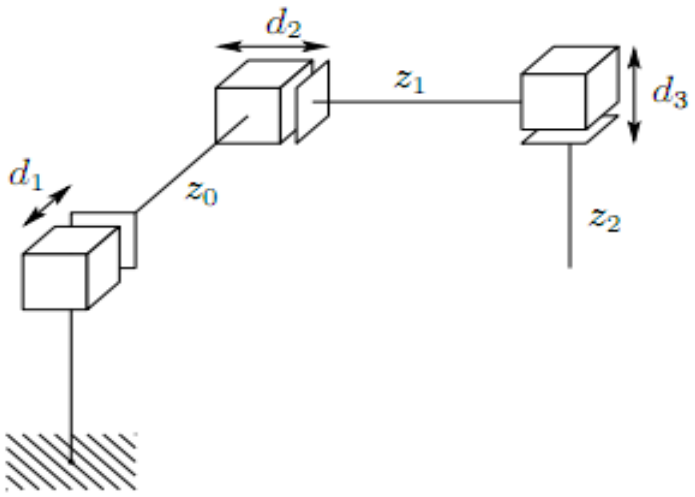


Figure (2.3): The Cartesian manipulator.

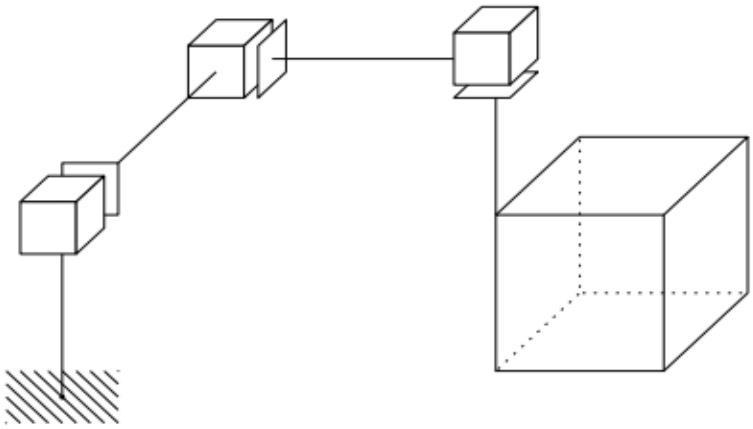
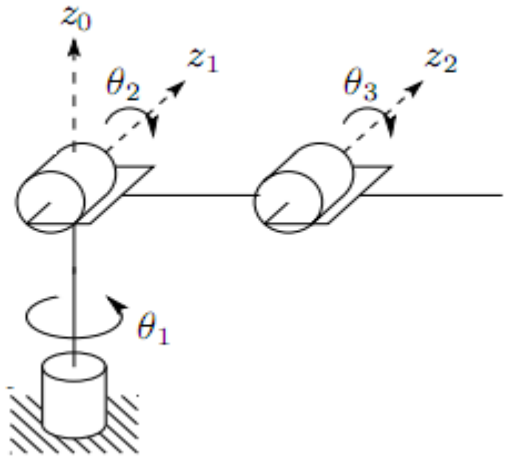


Figure (2.4): Work space of Cartesian robot.

2.2.1.3 Articulated Robot

The Articulated manipulator shown in figure (2.5) is also called a Revolute, or Anthropomorphic manipulator. This kind of manipulators consists of a chain of revolute joints only. This feature increases the capabilities of the robot considerably and allows reaching a bigger workspace as shown below in figure (2.6). One disadvantage is the analysis complexity compared with the previous two robots.



Figure(2.5): The Articulated manipulator.

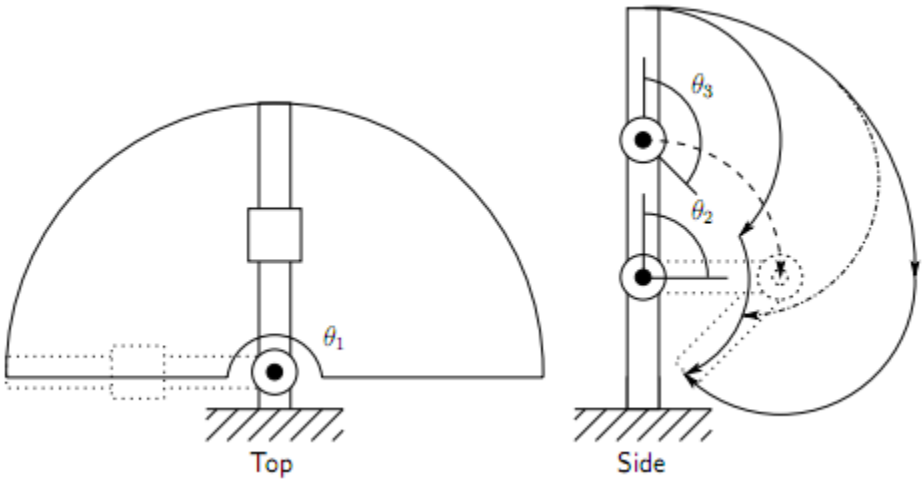


Figure (2.6): Workspace of Articulated manipulator.

2.2.2 Safety Sensor Options

Safety is the process by which one can ensure that the player (human) cannot be injured. The robot is a robust mechanical structure that may harm and hit the user. The usage of the obstacle detection sensors guarantees that this will not happen.

2.2.2.1 Ultrasonic Sensor

A device sends and receives radio or sound waves in order to determine the position and speed of a specific object. Its principle is similar to radar or sonar depending on echoes and transmission time.

Among different types of sensors, ultrasonic one was the choice adapted in the project; its small size, low cost and low weight made it perfect to be put at the end of the robot to detect the existence of obstacles [19]

2.2.2.2Infrared Sensor

It differs from ultrasonic sensors in principle and usage, this kind of sensor is commonly used in detecting motion in closed areas, rather than putting it on dynamic systems to sense the obstacles around.

Apparent motion is detected when an infrared source with one temperature, such as a human, passes in front of an infrared source with another temperature such as a wall.

Infrared sensors were not the best choice because of many reasons:

- -relatively high and huge detection range that may detect the player while he is setting far from the robot.
- -depending on the difference in temperature, this will make a problem if the obstacle is the side wall.
- -high chance of false alarms, this generated from the sensor's detection volume that may detect the stones or the table considering it as an obstacle.
- -large in size when compared with ultrasonic sensors.

2.2.3 Motors & Actuators Options

SCARA has three degrees of freedom, that means that three motors are needed to fully control the robot, its motion and path. The choices are limited, servo motors is not an option for such an application due to its huge size and weight, so the only remaining two options are DC and Stepper motors.

2.2.3.1 DC Motors

The first two links have a rotational motion, so it was decided to use two compacted DC motors. These compacted motors (gearbox and the motor) enable the robot to create the motion with the desired speed and torque. For safety reasons the second motor was selected with relatively low torque, thus if the ultrasonic sensor turned off and the robot hit the player, the robot will boomerang and will go to its home position.

The third motion is a linear motion to pick and place the stones. So a 15cm stroke motor will be the best choice for such a function in case it has a high speed, but

unfortunately these motors are slow, so the only option was to design a ‘Rack and Pinion’ mechanism.

Adequate speed of the links is achieved easily using DC motors; a simple PID controller will fix both the speed and accuracy as desired. Another advantage of the DC motors is the recovery ability, with the existence of the potentiometer to create the feedback part of the loop. The positions of the links can be known every moment, even if the robot hit the user and change its position, it will be able to cover this mistake and do the right action.

2.2.3.2 Stepper Motors

These kinds of motors do not have a continuous rotation; its motion consists of sequential steps until achieving the desired angle. The smallest step is about one degree, which means approximately 2.5cm at the end of a 25cm link, this phenomenon obviously restricts the accuracy of the robot while trying to pick and place the stones, besides there is no feedback in the stepper motors which limits their usage so much.

As a result one can summarize the main differences between the DC motors and Stepper motors in the following points:

1. Stepper motors are easily controlled with any programmable device; however logic and drive electronics are more complex.
2. Stepper motors are brushless and brushes contribute several problems, e.g., wear, sparks, electrical transients.
3. DC motors have a continuous displacement and can be accurately positioned, whereas stepper motor motion is incremental and its resolution is limited to the step size.
4. Stepper motors can slip if overloaded and the error can go undetected. (A few stepper motors use closed-loop control.)
5. Feedback control with DC motors gives a much faster response time compared to stepper motors.

The fourth point previously mentioned was enough to persuade the designer to use the DC motors.

2.2.4End Effecter's Gripper Options

The industrial robot's applications contain a lot of gripper styles and mechanisms, varying in its level of complexity, cost, weight and size depending on the application itself, such grippers maybe mechanical, vacuum, pneumatic or even electrical ones.

In the Chess Playing Robot, the desired gripper should have some important properties, light weight, accuracy, robustness, small size and fast action should all be in one gripper. Starting with the electro-mechanical gripper shown in Figure (2.7)[13], it satisfies all the properties needed. Without any change on the stone themselves, the robot will be able to pick and place these stones quickly and with good accuracy.

Choosing this kind of grippers offers a lot of advantages:

- Manual and automated machine loading/unloading
- Simplicity in control
- Grasps odd shaped or perforated parts
- Increases production and reduces shot air costs
- Designed for long, maintenance-free operation
- Operates effectively in any orientation
- Will not drop parts if system air-loss occurs
- Reduces noise
- Maximum operating temperature 140°F (60°C)

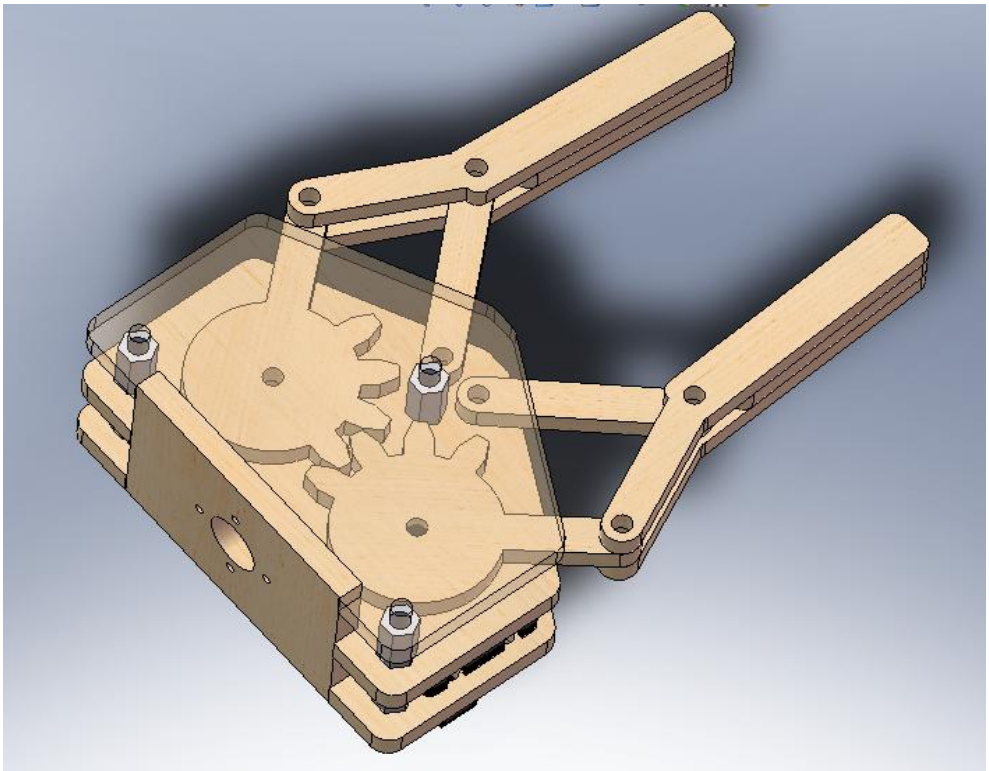


Figure (2.7): The electro-mechanical gripper of the robot.

Another option that may be considered is the pneumatic gripper. It has a high weight and a noisy action due to the compressor which will make the game uncomfortable at all. Its weight increases both by the torque needed to move the links, and the pending moment on the robot resulting in more wear between the links.

Part B:

The rest of the activities in this chapter are done by the computer engineering team, such as converting the chessboard into E-board, type of interfacing between the PC and E-board and the type of switches to be considered as input for the chess game classes. On the other hand, a comparison between different chess engines that are programmed using different environments is held. Finally the selection of the best software design and best chess board circuit to satisfy the following requirements:

- E-chess board that senses the position and movement of the stones.
- Configure the LED's on the board to light the possible movement.
- Real time-counter for the player.
- Customize the chess software to receive input from E-board, output notation value to the Robot.
- A display monitor to display the visual chess board with every movements and also all expected messages.

2.3 Chess Engine design Options

Chess software will be able to run chess game with standard rules, it is expected to lead the robot by giving it the correct legal movements. Chess software will also be able to help the player when he requests a help. These expected jobs must exist in the adopted chess software. The available chess engine programs can't achieve the project requirements, so the selected chess software should accordingly be customized to be compatible.

2.3.1SharpChess Game

Sharp chess is an open source chess game that is implemented in .Net framework using the C sharp.Net. This software will be the best choice to be used in the project for many reasons:

- The ability to play against computer opponent
- Implemented according to chess game rules and strategy game programming.
- Play with two modes, 16 level of difficulty or specify custom settings using combinations of Move Time, Maximum Search Depth, and Pondering.
- Easy to play with graphical user interface
- Real-time chess clocks.

On other hand, the chess engine features[10] are listed below making it the best option to be considered in the project as needed.

- WinBoard compatible.
- Alpha-Beta search with Pricipal-variation (PVS).
- Iterative Deepening.
- Hashtable (transition table) using Zobrist Keys. (Separate King-Pawn & In-Check hashtables).
- Adaptive Null-Nove Forward Pruning.
- 0x88 board representation.
- Move ordering using Hash table, MVV/LVA, SEE, 2-slot Killer-Move, History Heuristic.
- Search Extensions (Check Evasion, Single response, Pawn push to 7th rank, Re-capture of same-value piece).
- Quiescence Search (Re-Capture, Pawn-promotion, Enprise piece, pieces attacked by pieces of lower value).
- Pondering (thinking during opponent's time).
- "n moves in x minutes" Move time-allocation algorithm.
- Opening book containing over 1300 varied opening positions.

From above, the reasons beyond choosing SharpChess as the chess engine for this project are:

Table 2.1: Sharp Chess Features

	Yes	No
Open Source	✓	
Customization	✓	
Suitable Environment	✓	
High Speed Recognition	✓	

2.3.2 Chess game (Chris Meijers 2001)

This open source game is implemented using C sharp .Net, with poor in programming methodology and chess standard rules. The main advantages and disadvantages of this software are listed below.

Advantages:

- Graphical user interface
- Enable to undo the movement

Disadvantages:

- Two players needed. There is no intelligence in the game
- One level of difficulty to play the game
- Complex to customize the source code to be integrated with project needs
- No timers for player

According to these disadvantages, it is impossible to integrate this software with the project needs. Since the robot movements come from the intelligence side, it is hard to configure it with the SCARA robot.

2.3.3 Chess game implemented in UNIX

There are many open source games implemented in UNIX operating system environment. This software was ignored for many reasons, the inability to run any .Net frame work program in UNIX and most programs that are running in Windows operating system. In addition, it is hard to customize the source code to be integrated with the project needs.

2.3.4 Chess games that is implemented using other programming languages

Many chess software are available, built using programming languages such as Java, C, C++, VB. These software are not suitable for many reasons:

- None of these software apply the chess game standard rules.
- Do not have the intelligence classes in the game.
- Poor programming methodology make the customization complex.

According to the above comparison between software that can run in Windows and in UNIX, the best choice that is integrated very well, and offer all functions and properties for the Chess playing robot project is Sharpchess engine.

2.4Board Design options

The wooden chess board should sense the position and movement of the stones, positions detection means knowing the sixty four positions situation, which positions are changed and which are not, while Movement Detection means how to make the chess engine know that the player has made a movement on the board.

There are several choices to implement the chess board hardware. After comparing and studying four choices and options that will be discussed later, the best choice was the usage of the Keyboard encoder (KE72). Besides, the other choices have disadvantages that make it hard for them to be used according to the following explanations.

2.4.1 The Keyboard encoder card KE-72

The KE-72 Keyboard Encoder shown in figure (2.8) is a product designed to interface switches or keypads to the computer's PS/2 keyboard port .Keyboard encoder is the adopted choice for the board design, so the KE-72 appears as a keyboard to any computer system that has the PS/2 port.

The main special property of the KE-72 is the number of recognized pushed keys, KE-72 can deal with 72 simultaneous key events, besides the activation of any of the 72 individual inputs on the KE-72 may be programmed to do a specific work.

In chapter three, this option is presented and discussed in details.

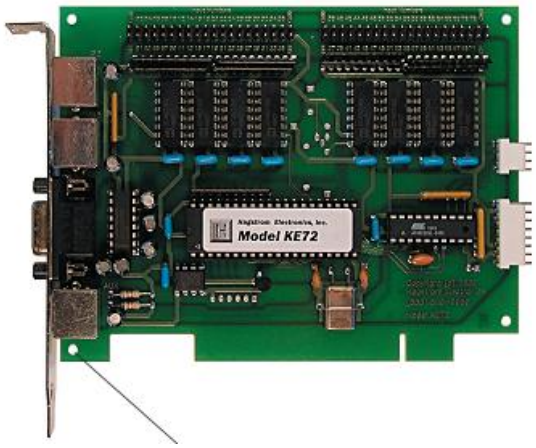


Figure (2.8): KE-72 interfacing card.

2.4.2 Scanning circuit using AND gates

- Components:

Figure (2.9) shows part of the circuit implementation. It contains a 3X8 decoder, eight AND gates and sixty four OR gates. The circuit must be connected to the PC using the Parallel port; three control signals get out from the parallel port. These are connected to the decoder. Eight data bits get in the port connected to the AND gates.

- Operation process:

The main operations of this scanning circuit are position detection and movement detection. Position detection should be done in two steps. Firstly, the software commands give the row addresses through parallel port control signals (three bits). Secondly, at the instant the row address is given, the AND gates read the situation of the selected row. By reading the eight addresses for the eight rows all position situation will be known.

An example of this is to scan the first row, first select row one through control signals, control signals has a value $(000)_{2,}$ following the selection, the position situation on row one will appear on the AND gates .Then the parallel port can get the AND gates through eight data lines.

The movement detection process is done as follows: each specific time (ex. 2 second), or before telling the player to make a movement, position detection is applied, then the comparison for each corresponding position in the new and the old scanning is done. This comparison detects which positions are changed.

- Advantages and disadvantages:

The main advantage is that it can provide position and movement detection, while the disadvantage is the implementation of this circuit. It needs large numbers of digital gates, which leads to high delay time effect, and is difficult to be implemented.

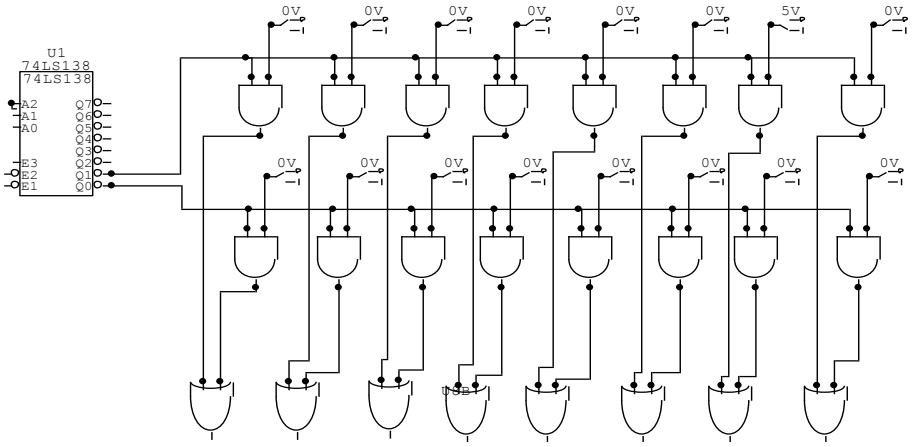


Figure (2.9): Part of scanning circuit using AND gates.

2.4.3 Scanning circuit using transceivers

- Components:

The circuit shown in figure (2.11) is implemented using a decoder, transceivers, OR gates and inverters. Using the parallel port, eight data lines plus three control signals it is possible to decode which line should be read, and the eight data lines are connected with the AND gates. AND gates combine the output of all transceivers, the transceiver's function is to transfer the inputs of transceiver to its outputs when the transceiver is enabled.

- Operations process:

This option requires two steps. First is doing the position detection by sending the addresses of the rows starting from 0 to 7, in each address. Reading the outputs of AND gates and saving them must be accomplished.

Secondly, movement detection is done by applying position detection continuously, so the movement detection needs to scan the board and save the condition of each line, then another scan must be done. By comparing the new reading with the old reading, changes in the positions can be detected.

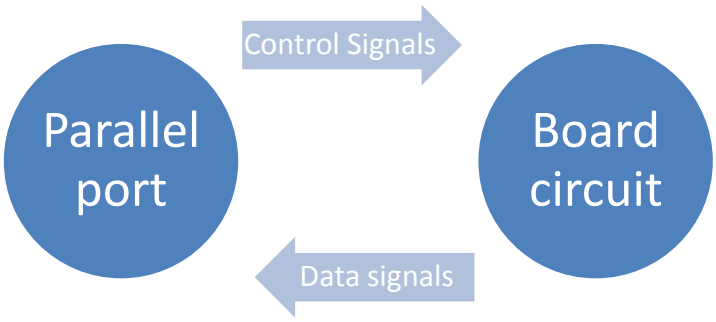


Figure (2.10): Illustrative Block Diagram.

The implementation of this method is simpler than the previous method, but still has its own software to be used correctly and the synchronization of the steps must be done accurately.

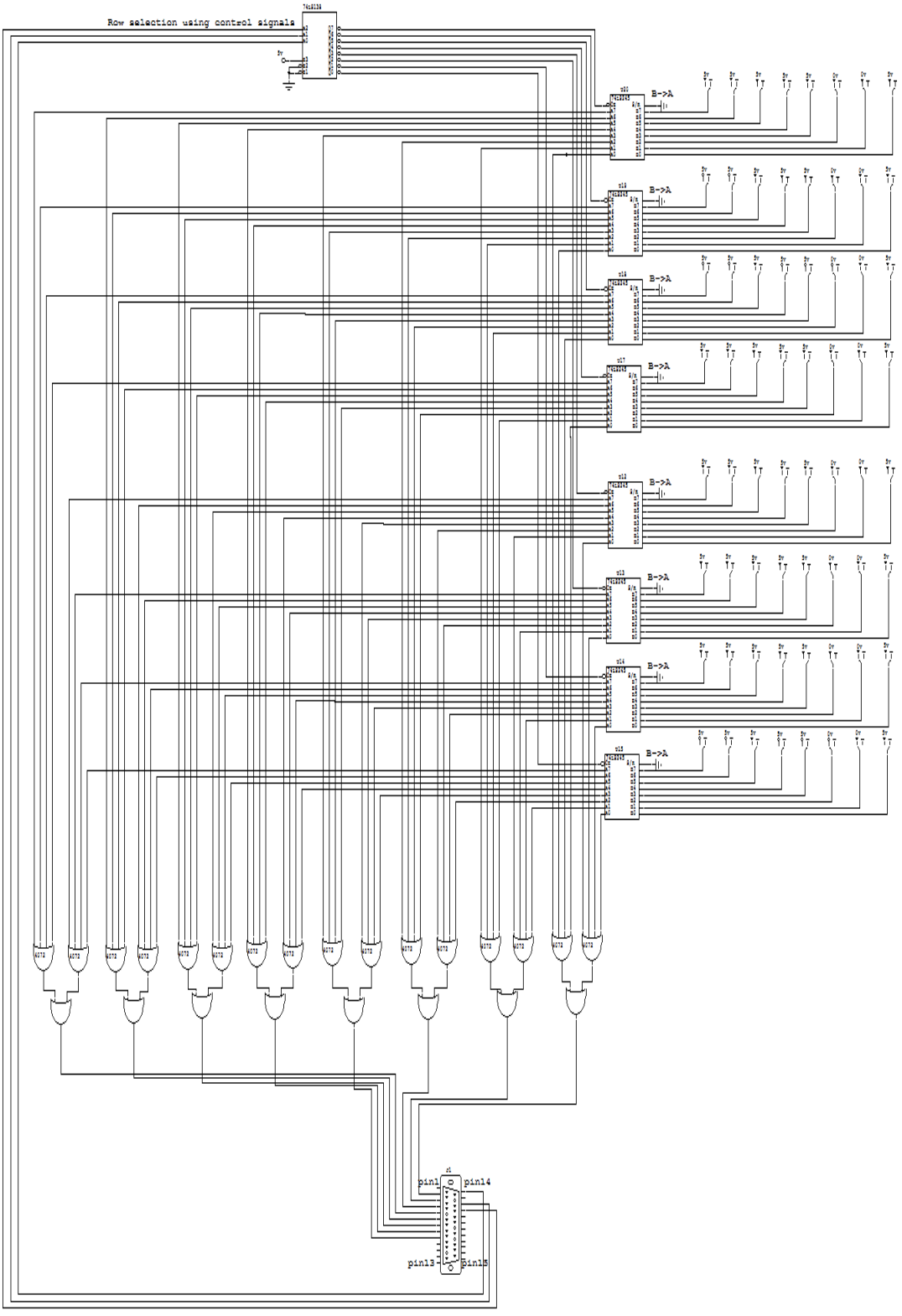


Figure (2.11): Scanning circuit using transceivers.

2.4.4 Scanning circuit using Data Acquisition Cards (DAQ)

This choice requires a data acquisition card that has at least sixty four digital I/O, each position on the board is connected through a digital I/O, this leads to read the sixty four positions at each scanning time, to accomplish position and movement detection. The following block diagram shows how this choice would be implemented.

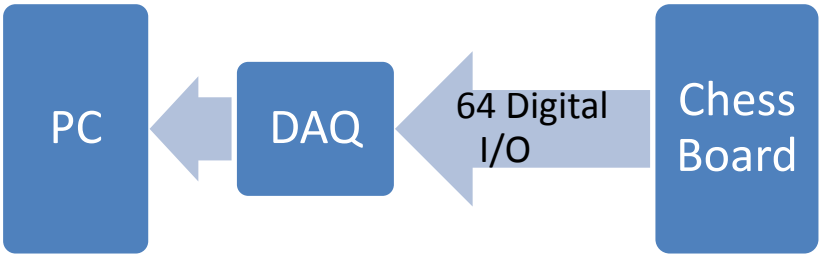


Figure (2.12): Illustrative Block Diagram.

- Advantages:

  1- Combines a lot of tools in one card (DAC, ADC, Timers, Counters, etc...).
  2- Facilitate the process of system controlling
  3- Relatively easy to use

- Disadvantages:

  1- High cost, more than one thousand USD.

  2- The available DAQ cards in the university labs can take only 16 digital inputs at the same time, so there is a need for four DAQ iterations to read all square at the same time.

  3- Needs compatible software (MATLAB, LabView … etc.).

According to the above comparison, option two and three are too complicated and have many disadvantages, in addition, DAQ choice makes a huge processing each reading time besides its high cost. On the contrary, of the adopted choice, which is the Keyboard Encoder, it has the easiest implementation and the simple stone; also, it has the main properties, which is required in this project. All details of the keyboard encoder are explained in chapter three.

2.5 Switches design Options

Each square at the board has a switch fixed under it, regardless the type of this switch it should make a closed circuit (or a signal) when a stone is put on the square. The purpose of using the best choice of switches is to get correct, fast and accurate reading values form chessboard. Many options for designing switch are available, the Hall Effect sensors, the photo diodes, small push buttons, or designing a suitable mechanism to perform a switching.

2.5.1 Photodiodes

A photodiode is a type of photo detector capable of converting light into either current or voltage, depending upon the mode of operation.

As shown in Figure (2.13), Photodiodes are similar to regular semiconductor diodes except that they may be either exposed or packaged with a window or optical fiber connection to allow light to reach the sensitive part of the device. Usage of this choice is done by putting a stone on a square which will create darkness and when the player or robot pick up the stone will allow light to activate the photodiode.

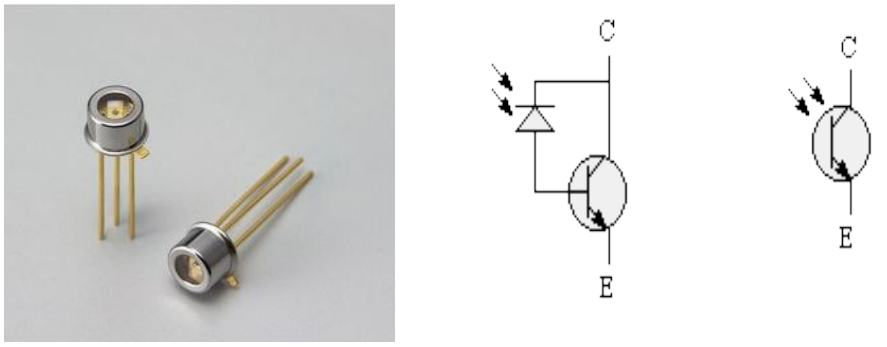


Figure (2.13): Photodiode.

- Advantages:

  1- Long life time
  2- Low cost
  3- Simple to implement its circuit.

- Disadvantages:

1- Output is not linear. The value of the output is not linear and the keyboard encoder input must be zero or 5V, so it is not compatible with KE72.
2- High read noise.
3- High dark current

2.5.2Push-Button

A push-button or simply a button is a simple switch mechanism for controlling some aspects of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, to be easily depressed or pushed. Buttons are most often biased switches, though even many un-biased buttons (due to their physical nature) require a spring to return to their un-pushed state.

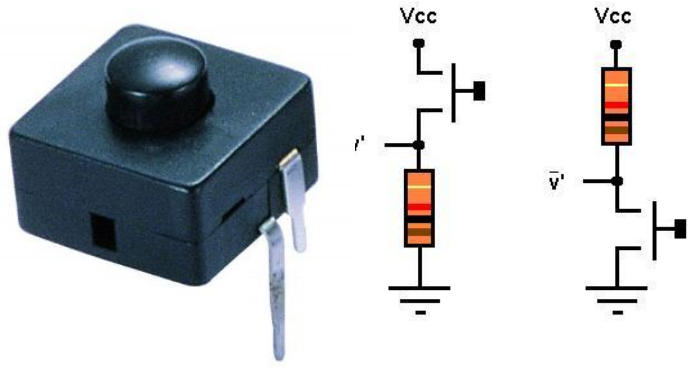


Figure (2.14): Push Button**.**

This choice was not used due to the following disadvantages:

1- Its operation is press/push to make it active
2- Chess pieces must be located at the button's center to make it active
3- Relatively the stones should have a sufficient weight to push the switch.

- Advantages of this choice:

1- Low cost
2- Accurate

3- Linear output voltage

2.5.3 Fixed Poles Switch

These switches do its job through closing the circuit between the negative and positive poles; this choice needs to fix the positive and the negative pole on the board, while using the chess piece to make the contact between the two poles, causing opening, or closing the circuit.

So a conducting metal must be fixed on the stone's bottom, and fixing the poles on each square on the board. When the chess stone is placed on a square, it makes a close circuit with 5 volts, and when it is removed, the switch becomes an open circuit with zero output Volt. This method doesn't offer so much accuracy because of the way of foxing these poles on the board, so it was not adapted.

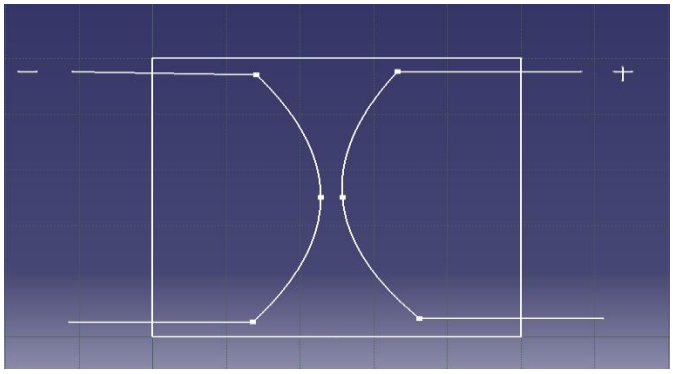


Figure (2.15): Implementation of fixed switch poles.

2.5.4 Magnetic Switch:

The type of the switch that is used as input for chessboard, is the magnetic switch sensor, it must be a toggle switch not latched since the latched will save the status for every square on the chessboard that was changed. Therefore, it needs a reset circuit to reset the latch to make correct reading .On the other hand the KE72 inputs required to be as toggle switch.

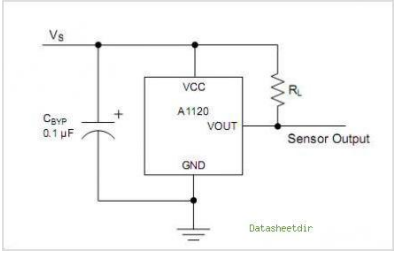


Figure (2.16): Hall Effect Sensor

- **Hall -Effect Features and Benefits:**

1. Continuous-time operation
2. Fast power-on time
3. Low noise
4. Stable operation over full operating temperature range
5. Reverse battery protection
6. Solid-state reliability
7. Factory-programmed at end-of-line for optimum performance
8. Regulator stability without a bypass capacitor.

According to the above comparison between different switches to be the input device in the chess board, the best choice is the Hall Effect sensor for many reasons and advantages as listed below:

1. Most types of switches need controllers to read the value of the output and to compare it with a saved value to decide if it is on or off, however this type do not need such a controllers.

2. Most sensors need reset circuit since they are latched sensors and also some of the switches need accurate in position movement to activate the input. However this type works on magnetic field with suitable distance and its toggle sensor.

3. The Keyboard encoder require that the input must be form CMOS technology ,so some types of switches and sensors can't be used.

4. Low cost small size and that can be hidden under the wooden chess board.

2.6 Board's LEDs

One of the board requirements is using LEDs for guiding purpose, the guiding may done in two cases:

1- Every event on the board triggers the LED's, so when the player or the robot, pick chess piece, or places a piece at a location, this specific location must be targeted using the LEDs.

2- When the player ask for help to do a movement, and it is his turn to play, the player pick up a chess piece, then the chess software must light all location which are legal to move the stone to.

These two cases may be implemented using one of the following two ways:

2.6.1 Eight vertical –Eight horizontal LED's

In this method, the target square uses one LED of the horizontal row and another one from the vertical column, the intersection between the vertical and horizontal lighted LEDs is the target square. Figure (2.17) show how the board looks like[1].

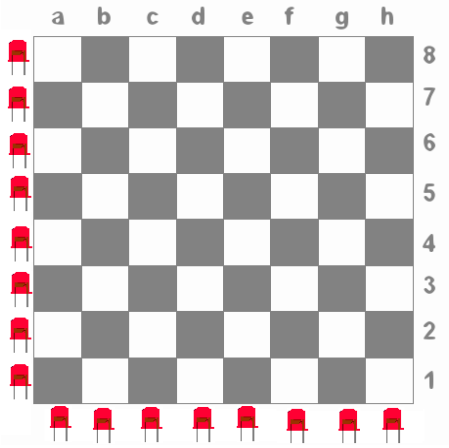


Figure (2.17): Horizontal/Vertical LEDs

The connection circuit of this choice shown in Figure (2.18), it shows the vertical and horizontal LEDs using two 3*8 decoders.

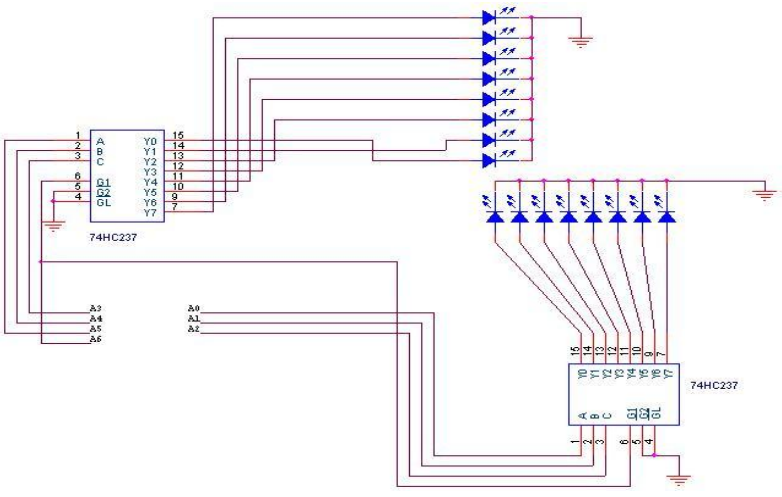


Figure (2.18) Connections Circuit for the Vertical-Horizontal LEDs.

[1] APPENDIX A-2

2.6.2 One LED under each square

In this method, there is a LED under each square, the target square lights when its LED is activated. Figure (2.20) show how the board looks like.

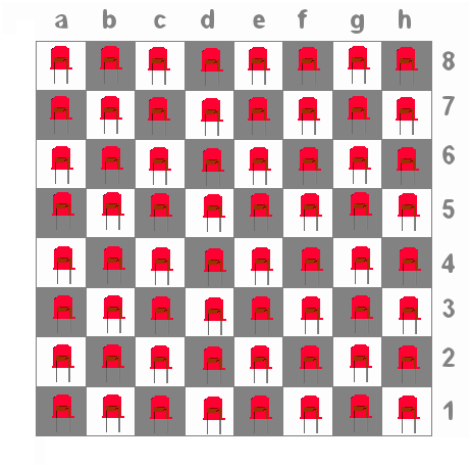


Figure (2.19): Board's LEDs.

The connection circuit for the 64 LEDs is designed using 4 decoders as shown:

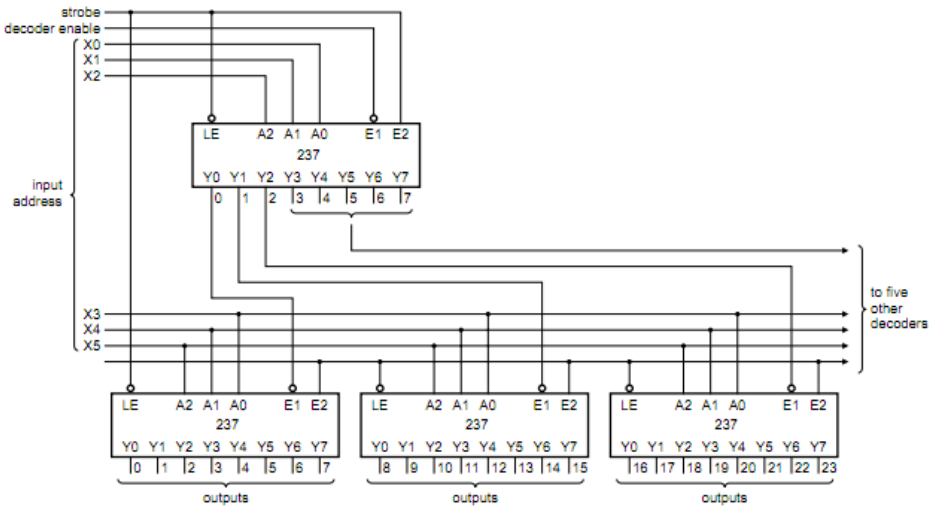


Figure (2.20): Connection circuit for the 64 LEDs Board.

The “One LED Under Each square” method offers some advantages over the “Vertical-Horizontal LEDs” method, it is better if the guiding “help” mode is selected, also it gives a more beautiful shape for the board while playing, so it is adapted.

CHAPTER THREE 3

System Design

3.1 Introduction

3.2 Structure Design
- 3.2.1 SCARA Structure
- 3.2.2 Robotics Calculations
- 3.2.3 Path Planning of SCARA Robot
- 3.2.4 Trajectory Planning of SCARA Robot
- 3.2.5 Mechanical Calculations

3.3 Board Design
- 3.3.1 Hall effect sensor
- 3.3.2 E-board circuit using KE-72

3.4 Board LEDs

3.5 Software Design
- 3.5.1 Input Class (interaction class)
- 3.5.2 Chess Engine
- 3.5.3 Output Class

3.6 Cheating Discovery and Solutions

3.7 MATLAB
- 3.7.1 Robots motion steps
- 3.7.2 NI PCI-6024 DAQ card

3.1 Introduction

In this chapter the design of the project is discussed, from the options previously mentioned in chapter two one option is selected here, analyzed and discussed, beginning with the structure design which include the robot design, its calculations and mechanical calculation, then the design of the board, its switches and interfacing technique. The last section deals with the PC and the software design, explaining the interfacing cards and software used in the project.

3.2 Structure Design

In this section, the mechanical structure is discussed in details, the kind of the robot arm, its robotics analysis, and mechanical analysis such as the applied torques and moments.

3.2.1 SCARA Structure

This kind of robot consists of three links as mentioned previously, the first two links have a rotational motion, while the third link have a linear motion up and down in order to pick and place the stones. The dimensions of the first two links are designed simply depending on the dimensions of the board, which is selected to have a 50x50 cm to achieve the most comfortable situation for the human. The first two links have the same length of 26.3 cm to reach the farthest point in the board while not exceeding its borders as shown in figure (3.1); this may be considered as another factor that will increase the safety by forbidding the robot from reaching nearer points to the human.

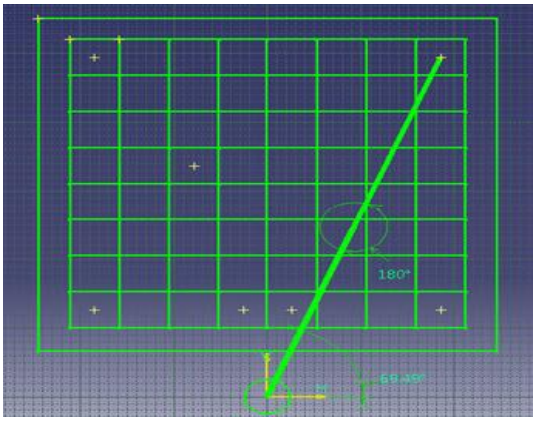
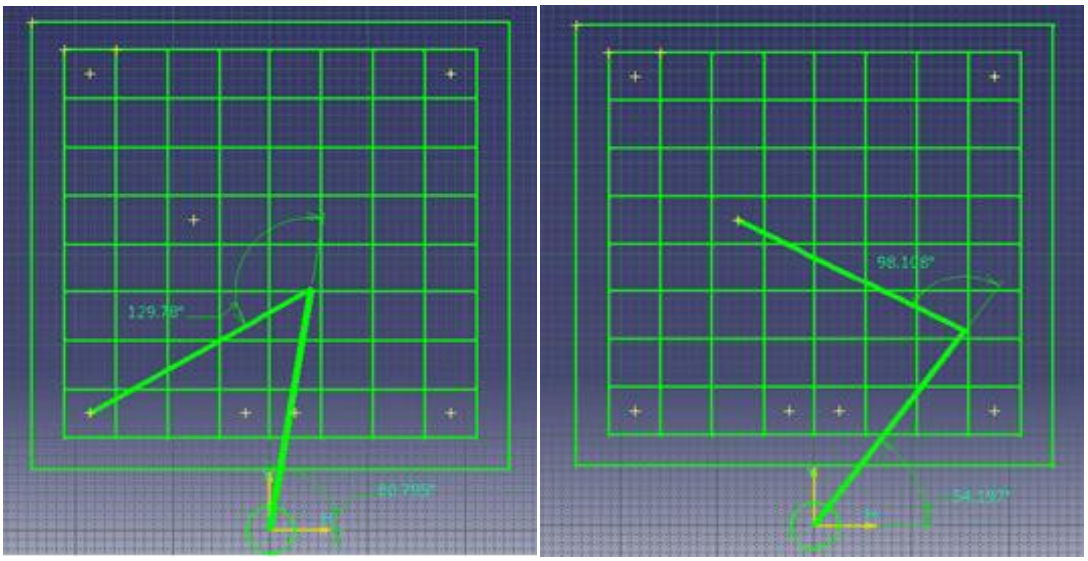


Figure (3.1): Robot while reaching the farthest point on the board.

The robot links are constructed from Aluminum material; because of its low weight. The first two links are about 264gm for each which will decrease the torque needed to be applied by the motors to move the robot.

The robot will receive the addresses to go to from the chess engine, then by using MATLAB the angles of rotation of the two motors can be calculated from the inverse kinematic program. Then it is useful to use a software such as CATIA to ensure that these results are correct, taking two examples as shown in the figures below, these are the same results obtained from inverse kinematics program.



(a) (b)

Figure (3.2): Examples of obtaining the angles using CATIA.

In figure (3.2.a), the two local angles obtained are the same as what were obtained from the inverse kinematics programs, which are 80.795^{o} and 129.78^{o} respectively, the same results was found regarding Figure (3.2.b).

3.2.2. Robotics Calculations

In order to identify positions and addresses in the robot's environment a lot of calculations must be done. Programming the Forward and Inverse Kinematics equations in the robots controller is necessary; it enables it to calculate and identify the rotation

angles that should be applied by each motor to reach the desired address. In the following sections, this analysis is discussed in details.

3.2.2.1 Forward Kinematics

First establishing the joint coordinate frames using the DH convention as showed in Figure (3.3), where frame 0 is the base frame and frame 3 is the tool frame. This industrial robot has 3 degrees of freedom and has 3 links.

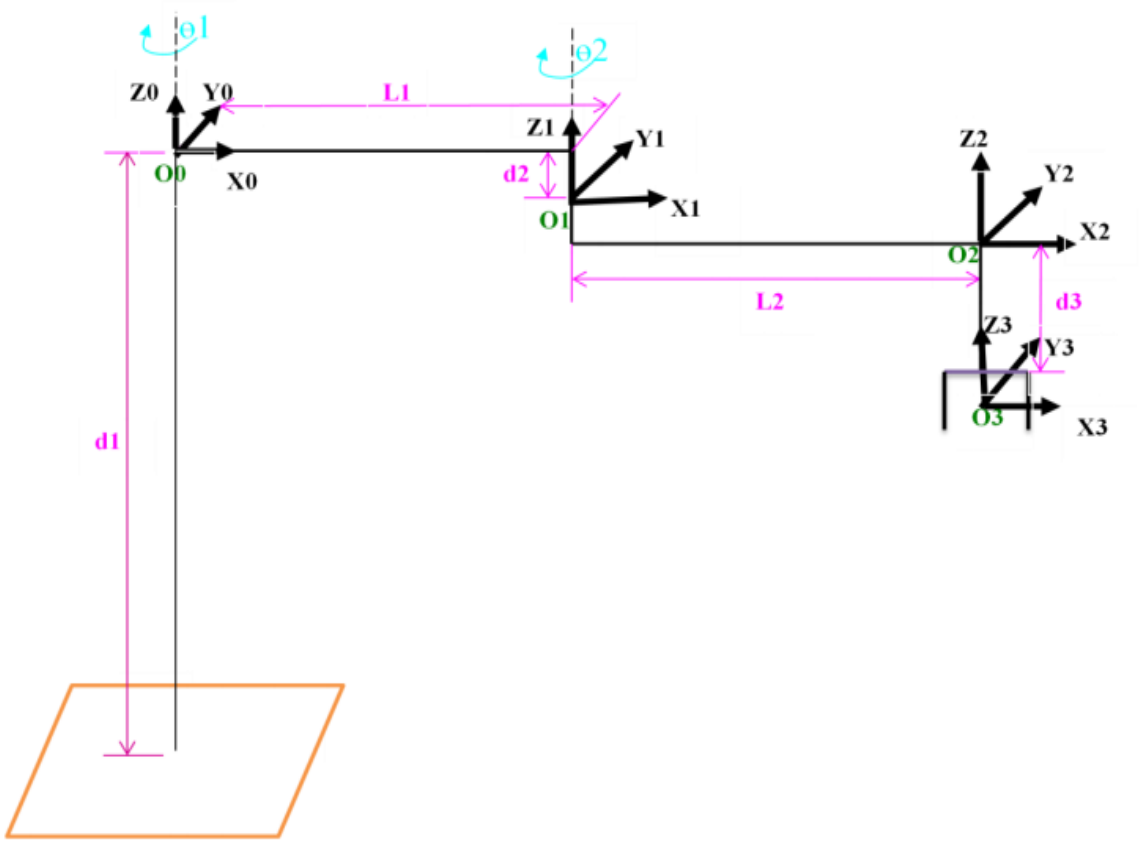


Figure (3.3): DH coordinate frame assignment for the SCARA manipulator.

Finding the DH parameters of these 4 link where d_2 =0, will give:

Table (3.1): DH parameters of the SCARA robot (* joint variable).

Link	a(mm)	α(degree)	d(mm)	θ(degree)
1	L_1	0	0	θ_1^*
2	L_2	0	0	θ_2^*

3	0	0	d_3^*	0

Constructing the homogenous transformation matrices by substituting these values in equation (3.1): [7]

$$T_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.1)$$

Where c denotes cosine and s denotes sine, will give:

$$T_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & L_1c_1 \\ s_1 & c_1 & 0 & L_1s_1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2)$$

$$T_2^1 = \begin{bmatrix} c_2 & -s_2 & 0 & L_2c_2 \\ s_2 & c_2 & 0 & L_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$

$$T_3^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d_3^* \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4)$$

The resulting homogenous transformation is:

$$T_3^0 = T_1^0 * T_2^1 * T_3^2 \quad (3.5)$$

$$T_3^0 = \begin{bmatrix} c_1c_2 - s_1s_2 & -c_1s_2 - s_1c_2 & 0 & L_2c_1c_2 - L_2s_1s_2 + L_1c_1 \\ s_1c_2 + c_1s_2 & -s_1s_2 + c_1c_2 & 0 & L_2s_1c_2 - L_2c_1s_2 + L_1s_1 \\ 0 & 0 & 1 & -d_3^* \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.6)$$

3.2.2.2 Inverse kinematics

This section is concerned with the inverse problem of finding the joint variables in terms of the end-effector position and orientation. This is the problem of inverse

kinematics. Subsections will discuss the problem of inverse position, velocity and acceleration of SCARA robot.

3.2.2.2.1Inverse solution of position

The end-effector orientation and position with respect to the home coordinates can be found using:

$$T_3^0 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.7)$$

Where the first three columns are the orientation of the end-effector with respect to the home coordinates, the fourth column represents the position of the end-effector in the x, y and z direction respectively of the home coordinate.
From (1,4) and (2,4) elements of the equations (3.6) and (3.7): [6]

$$p_x = L_2c_1c_2 - L_2s_1s_2 + L_1c_1 = L_2c_{12} + L_1c_1 \quad (3.8)$$

$$p_y = L_2s_1c_2 - L_2c_1s_2 + L_1s_1 = L_2s_1 + L_1s_1 \quad (3.9)$$

Then after some calculations we get: [6]

$$\theta_2 = tan^{-1}\left(\frac{s_2}{c_2}\right) \quad (3.10)$$

$$\theta_1 = tan^{-1}\left(\frac{s_1}{c_1}\right) = tan^{-1}\left(\frac{(L_1+L_2c_2)p_y - L_2s_2p_x}{(L_1+L_2c_2)p_x + L_2s_2p_y}\right) \quad (3.11)$$

And from (4, 4) elements of the equation (3.6) and (3.7):

$$d_3^* = -p_z \quad (3.12)$$

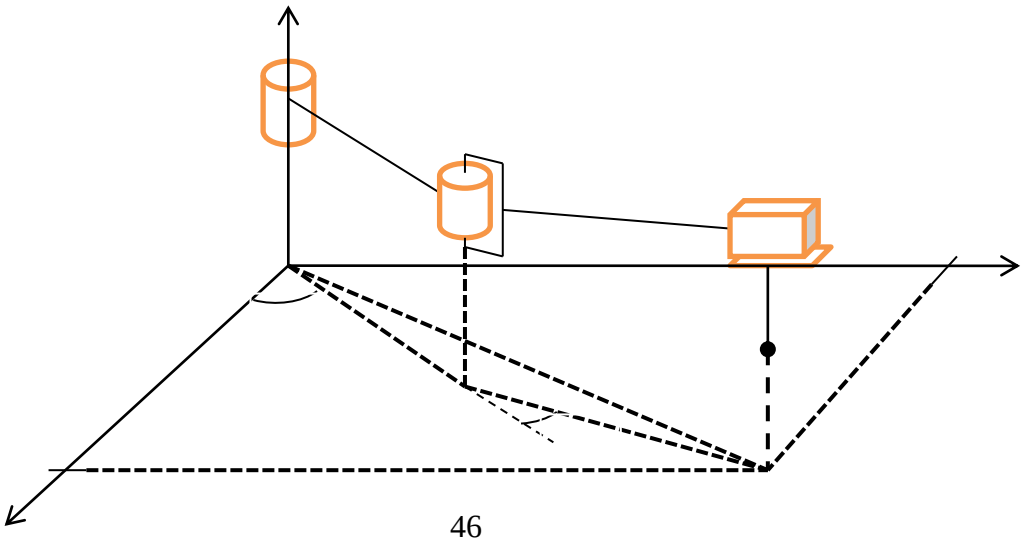


Figure (3.4): SCARA manipulator.

3.2.2.2.2 Inverse solution for velocity

From equation (3.8) and equation (3.9):

$$\dot{p_x} = -L_1 s_1 \dot{\theta}_1 - L_2 s_{12}\left(\dot{\theta}_1 + \dot{\theta}_2\right) = -(L_1 s_1 + L_2 s_{12})\dot{\theta}_1 - L_2 s_{12}\dot{\theta}_2 \quad (3.13)$$

$$\dot{p_y} = L_1 c_1 \dot{\theta}_1 + L_2 c_{12}\left(\dot{\theta}_1 + \dot{\theta}_2\right) = (L_1 c_1 + L_2 c_{12})\dot{\theta}_1 + L_2 c_{12}\dot{\theta}_2 \quad (3.14)$$

Using Kramer's rule to solve equation (3.13) and equation (3.14):

$$\dot{\theta}_1 = \frac{\dot{p_x} c_{12} + \dot{p_y} s_{12}}{L_1 s_2} \quad (3.15)$$

$$\dot{\theta}_2 = \frac{-\dot{p_y}(L_1 s_1 + L_2 s_{12}) - \dot{p_x}(L_1 c_1 + L_2 c_{12})}{L_1 L_2 s_2} \quad (3.16)$$

And the translational velocity is:

$$\dot{d}_3^* = -\dot{p_z} \quad (3.17)$$

3.2.2.2.3 Inverse solution for acceleration

$$\ddot{\theta}_1 = \frac{(-\ddot{p_x} s_{12} + \ddot{p_y} c_{12})\dot{\theta_{12}} + (\ddot{p_x} c_{12} + \ddot{p_y} s_{12}) - L_1 c_2 \dot{\theta}_1 \dot{\theta}_2}{L_1 s_1} \quad (3.18)$$

$$\ddot{\theta}_2 = -\frac{\left[\left(\ddot{p_y} s_1 - \ddot{p_x} c_1\right)L_1 + \left(\ddot{p_x} c_{12} + \ddot{p_y} s_{12}\right)L_2 + \left(-\dot{p_x} s_1 + \dot{p_y} c_1\right)L_1 \dot{\theta}_1 + \left(\dot{p_x} s_{12} + \dot{p_y} c_{12}\right)L_2 \dot{\theta_{12}} + L_1 L_2 C_2 \dot{\theta}_2^2\right]}{L_1 L_2 s_2} \quad (3.10)$$

$$\ddot{d}_3^* = -\ddot{p_z} \quad (3.20)$$

3.2.2.2.4 Jacobian Matrix of SCARA Robot

The Jacobian matrix of SCARA has the dimensions of 6*3 since it has three links, so this gives:

$$J(q) = \begin{bmatrix} z_0 \times (o_3 - o_0) & z_1 \times (o_3 - o_1) & z_2 \\ z_0 & z_1 & 0 \end{bmatrix} \quad (3.21)$$

Where:

$$z_0 = \begin{bmatrix}0\\0\\1\end{bmatrix}, \qquad o_3 = \begin{bmatrix}L_2c_1c_2 - L_2s_1s_2 + L_1c_1\\ L_2s_1c_2 + L_2c_1s_2 + L_1s_1\\ -d_3^*\end{bmatrix}$$

$$o_0 = \begin{bmatrix}0\\0\\0\end{bmatrix}, \qquad o_1 = \begin{bmatrix}L_1c_1\\L_1s_1\\0\end{bmatrix}, \qquad z_1 = z_2 = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

Substituting these in equation (3.21) will give:

$$J(q) = \begin{bmatrix} -L_2s_1c_2 - L_2c_1s_2 - L_1s_1 & -L_2s_1c_2 - L_2c_1s_2 & 0\\ L_2c_1c_2 - L_2s_1s_2 + L_1c_1 & L_2c_1c_2 - L_2s_1s_2 & 0\\ 0 & 0 & 1\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 1 & 1 & 0 \end{bmatrix} \qquad (3.22)$$

3.2.3 Path Planning of SCARA Robot

Path planning is the process of defining the shape of the path should be crossed by the robot, thus it is a geometric description of the robot motion, but it doesn't specify any dynamic aspects of the motion. [7]

The robot geometry can be calculated to reach each one of the 65 address from inverse kinematics analysis, then it is simply a signed voltage value powered to each motor using the DAQ (Data Acquisition) cards to reach the desired position. This process starts with the chess program telling the MATLAB the new addresses, a programmed inverse kinematics equations will calculate the desired angles for each motor, an m-file will calculate the shortest path depending on the current geometry, DAQ cards will translate these angles into analogue voltages supplied to each motor, thus, rotating clockwise or counterclockwise is just a polarity of the power.

3.2.4 Trajectory Planning of SCARA Robot

The end-effector of manipulator has to move in a particular fashion to accomplish the specific task. The execution of the specific task requires the manipulator to follow a pre-planned path, which is major problem of motion or trajectory planning and motion control. The goal of trajectory planning is to describe the request motion of a manipulator as a time sequence of joint/link/end-effector location and derivative of locations, which are generated by "interpolating" or "approximating" the desired path by a polynomial function. Some techniques for trajectory planning are point-to-point motion and continuous-path motion. Main objective of the trajectory planning is to get a smooth motion of manipulator and trajectories. The smooth motion has an important advantage of reducing the vibrations and wear of mechanical system.

3.2.4.1 Steps in Trajectory Planning

1) Task Description:

The first step in motion planning is to identify the kind of motion required. Task can be grouped into 3 different categories.

In the case of application such as *pick and place* operation, task is specified as initial and final end-effector location. This is called point to *point-to-point* motion. No particular specification about the intermediate locations of end-effector.

If in addition to start and finish point, a specific path between them is required to be traced by the end-effector in Cartesian space, this is called *continuous path* motion and trajectory. Operations such as welding and plotting are example of continuous path motion.

There is third type of task description, where more than two points of path are specified. This is done to ensure a better monitoring of executed trajectory in case of application similar to point-to-point motion.
Apart from required task attributes, user can also include temporal attribute such as travel time in task specification.

2) Selecting and employing a Trajectory Planning Technique:

The various trajectory-planning falls into one of two categories: *Joint space techniques* or *Cartesian space techniques.* In case of point-to-point motion, Joint space techniques are employed in which motion planning is done at joint level. The Joint space planning scheme generate time-dependent function of all joint variable and their first two derivatives to describe their motion of manipulator. For application

requiring continuous path motion, *Cartesian space techniques* are used. The Cartesian space planning scheme provide time history of the location, velocity, acceleration of the end-effector with respect to the base. The corresponding joint variables and their derivatives are computed, using *inverse kinematics.*

3) Computing the Trajectory:

Time sequence values are attained by the functions generated from the trajectory planning technique are computed at a particular path update or sampling rate. Path update rate in real time lies between 20Hz to 200Hz in typical industrial manipulator systems.

3.2.4.2 Joint Space Technique

The first step in these techniques is to obtain the corresponding set of joint variable valued for each specified path point, either by employing the inverse kinematics algorithm or variable can directly recorded if trajectory planning is performed by *technique-by-showing- technique*. The next step is to find smooth function $q(t)$for each n-joints of an n-DOF manipulator. For that, it is needed *travelling time, initial and final locations*. Three functions are examined here:

3.2.4.2.1 Cubic Polynomial Trajectories: [7]

In the case of cubic polynomial, the velocity of the links will increase then it will decrease, also the robot will accelerate. The following figure shows an example of this kind of trajectory. The disadvantage of the cubic polynomial is the discontinuity in the acceleration, leading to an impulse jerk (the derivative of the acceleration) which will excite vibrational modes and reduce the tracking accuracy of the robot.

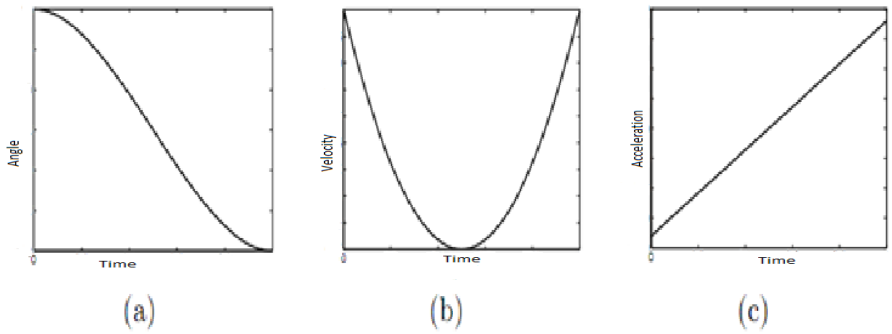


Figure (3.5): (a) Cubic polynomial trajectory, (b) Velocity profile for cubic polynomial trajectory, (c) Acceleration profile for cubic polynomial trajectory.

Let $q_0$and $q_f$be the initial and the final point values, respectively, of the joint variable q. Generalized joint variableq_i $(\theta_i or\ d_i)$ denotes for joint i of n-DOF manipulator. Motion begins at $t = 0$ and ends at $t = t_f$where t_f is travelling time decided by user or program. Both $q(t)$and its derivatives should be smooth from $t = 0$to$t = t_f$. Therefore, to describe joint motion, we assume that cubical polynomial is:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

which gives parabolic velocity and linear acceleration profile

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2$$

and $$\ddot{q}(t) = 2a_2 + 6\,a_3 t$$

From above we can get the values of a_0, a_1 ,a_2 and a_3.

So for the initial (o) and final (f) conditions, noting that we have $q = \theta$here,

$$\theta_o = a_0 + a_1 t_o + a_2 t_o^2 + a_3 t_o^3$$
$$\dot{\theta}_o = a_1 + 2a_2 t_o + 3a_3 t_o^2$$
$$\theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3$$
$$\dot{\theta}_f = a_1 + 2a_2 t_o + 3a_3 t_o^2$$

Arranging these equations in matrix form, we get

$$\begin{bmatrix} 1 & t_o & t_o^2 & t_o^3 \\ 0 & 1 & 2t_o & 3t_o^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \theta_o \\ \dot{\theta}_o \\ \theta_f \\ \dot{\theta}_f \end{bmatrix}$$

The motion will go from rest to rest, thus substituting in the above equations will give:

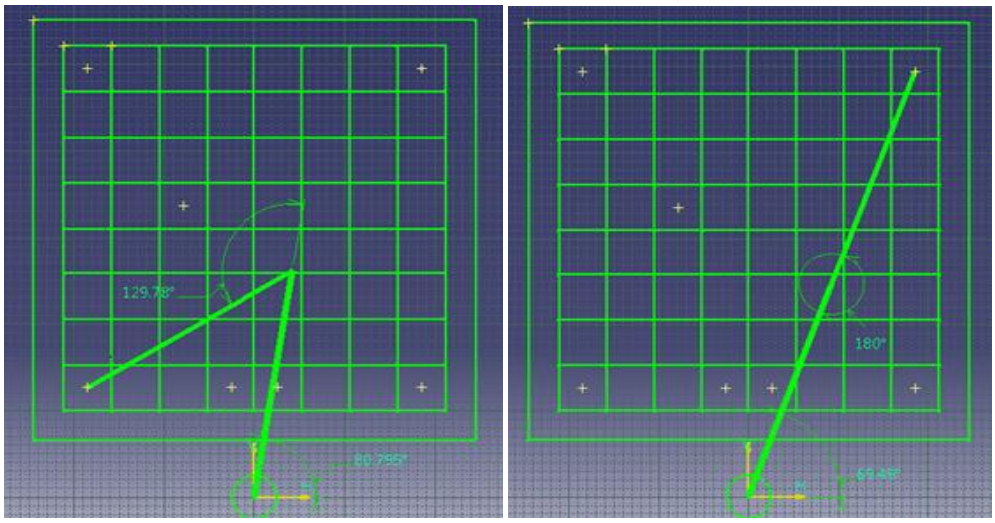
$$\theta(t) = 80.8 - 8.843\ t^2 + 2.828\ t^3$$
$$\dot{\theta}(t) = -16.97\ t + 8.484\ t^2$$

$$\ddot{\theta}(t) = -16.97 + 16.968\ t$$

Now taking an example of a motion that may happen while playing, that is to go from the position a1 (lowest most left point) to h8 (highest most right point), the distance between the two addresses are the longest and thus this motion will be the highest time-consumption, the largest traveling time is adjusted to 3 seconds, and the lowest is adjusted to 1 second, so the traveling time (t_f) can be calculated in general to be:

$$t_f = 1 + d/261.1$$

Where d is the distance between the two addresses and 261.1mm is the length of the diagonal of the board.



(a) (b)

Figure (3.6):(a) The robot in position A1. (b) The robot in position H8.

From the previous information, one can write a simple program on MATLab to sketch the trajectory of the robot to obtain the following figures:

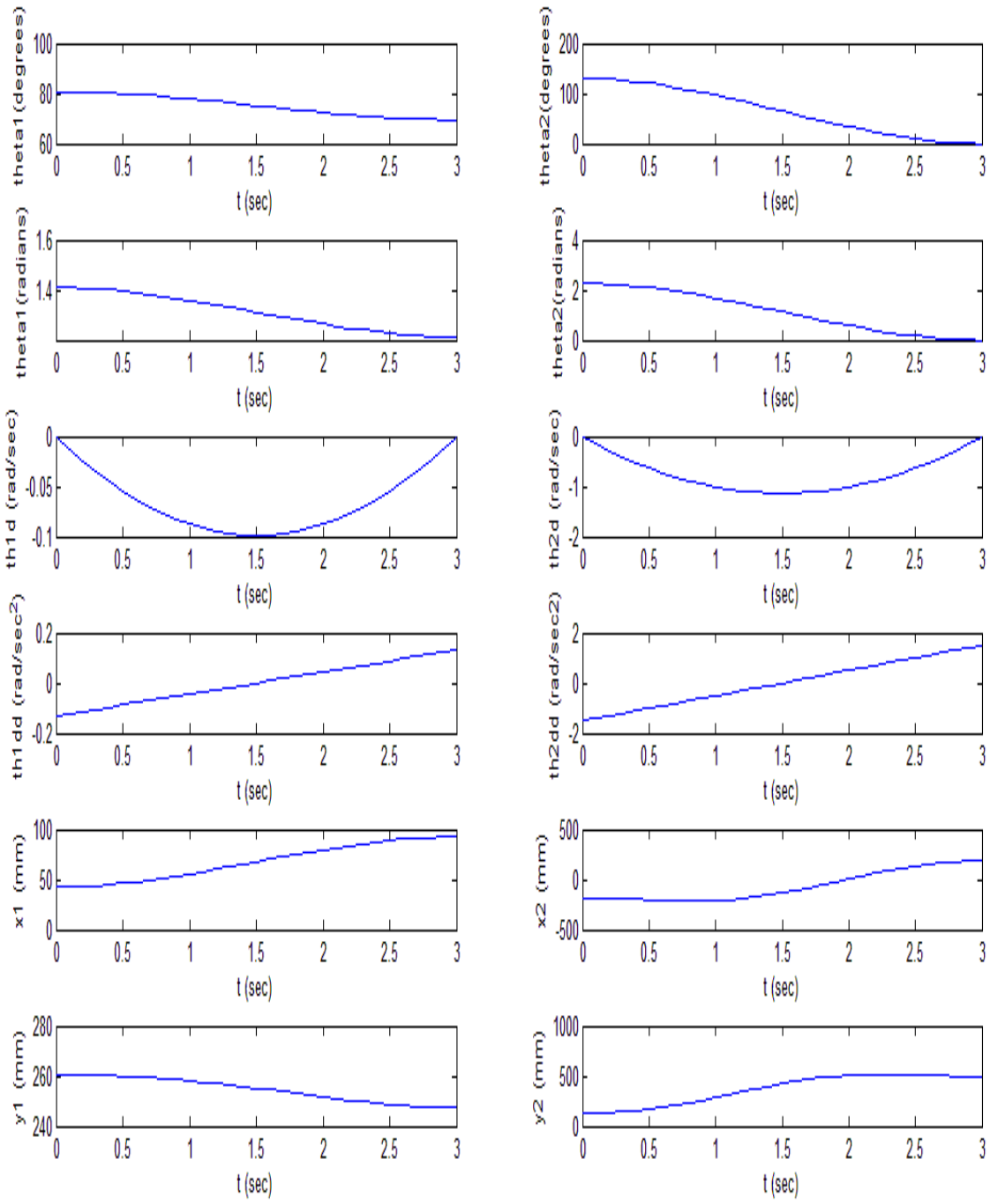


Figure (3.7): The Robot's response while traveling from position a1 to h8.

Where:

theta1: angle of rotation of the first motor
theta2: angle of rotation of the second motor
th1d: angular velocity of the first motor
th1dd: angular acceleration of the first motor
th2d: angular velocity of the second motor
th2dd: angular acceleration of the second motor
x1: x-coordinates of the end of the first link
x2: x-coordinates of the end of the second link (end-effector)
y1: y-coordinates of the end of the first link
y2: y-coordinates of the end of the second link (end-effector)
t: response time

Now doing a simple forward kinematics on the robot only in the plane x_o-$y_o$as shown in the following figure will give the equations bellow:

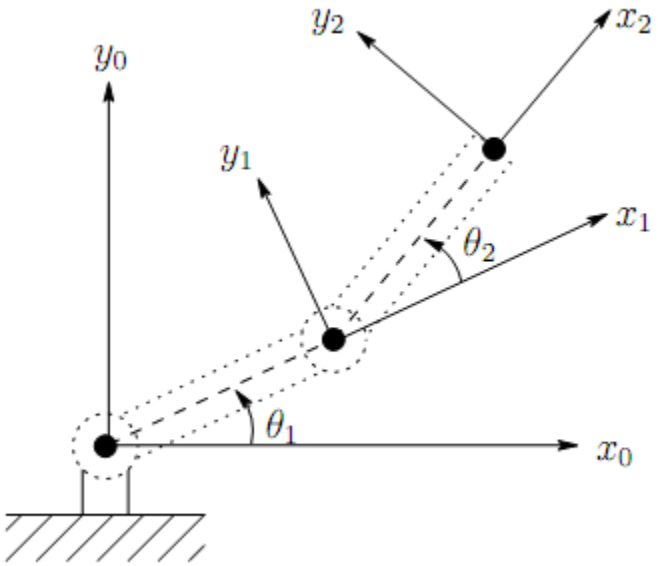


Figure (3.8):Planner coordinate frames for SCARA Robot.

$$x_1 = \alpha_1 cos(\theta_1)$$
$$y_1 = \alpha_1 sin(\theta_1)$$
$$x_2 = \alpha_1 cos(\theta_1) + \alpha_2 cos(\theta_1 + \theta_2)$$
$$y_2 = \alpha_1 sin(\theta_1) + \alpha_2 sin(\theta_1 + \theta_2)$$

Where α_1 and α_2 are the lengths of the first link and the second link respectively.

Now sketching the coordinates of the end-effector in the previous example using MATLAB will give:

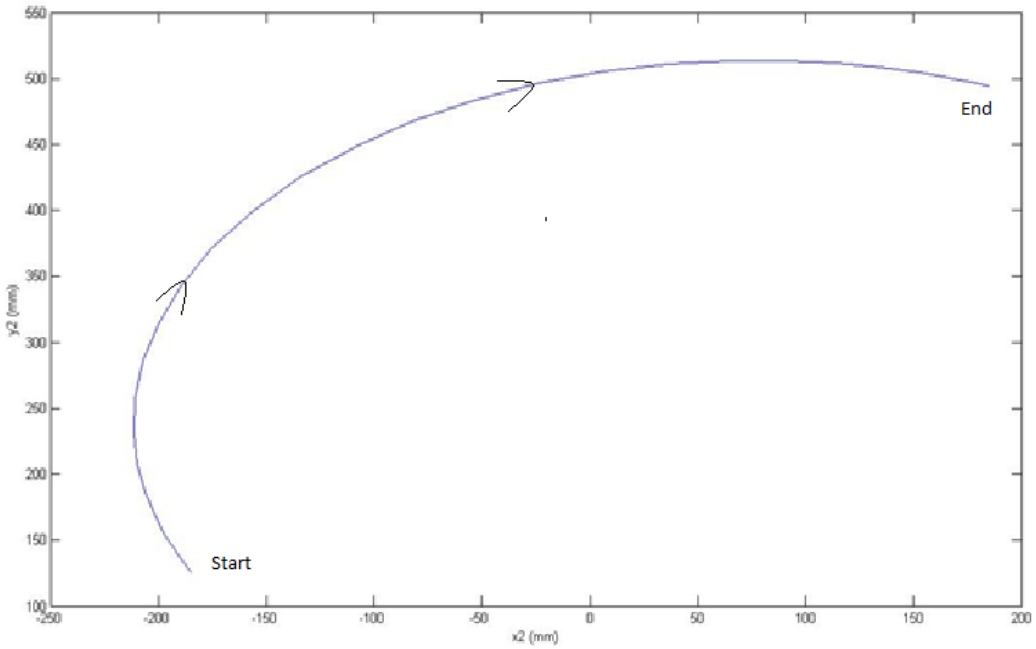


Figure (3.9): The path shaped by the end-effector while going from a1 to h8.

3.2.4.2.2 Quintic Polynomial Trajectories

This kind of trajectory differs from the previous one in the acceleration behavior. As seen in the following figure the robot will accelerate then decelerate with a continuous function.

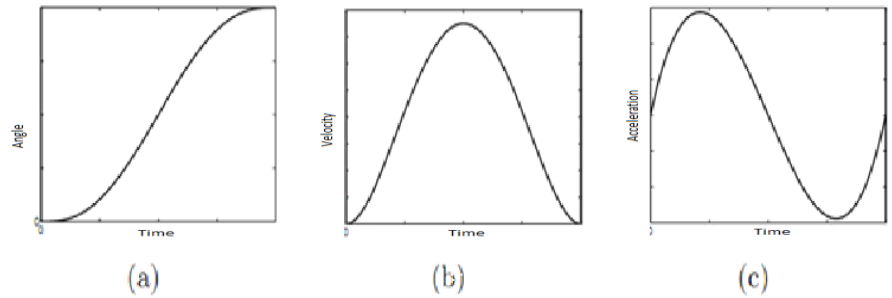


Figure (3.10): (a) Quintic Polynomial Trajectory, (b) Velocity Profile for Quintic Polynomial Trajectory, (c) Acceleration Profile for Quintic Polynomial Trajectory.

There will be a constant acceleration up to certain time say t_b and then constant velocity say *v* i.e. zero acceleration and then for time period t_g constant retardation to make velocity from *v* to zero. There will be a trapezoidal velocity profile. Also we can make the triangular velocity profile for small value of joint parameter. Here we have to choose the travel time in parabolic blends and if we fix the total time of motion then we can calculate the value of acceleration else if we fix the value of acceleration then we can calculate the value of acceleration.

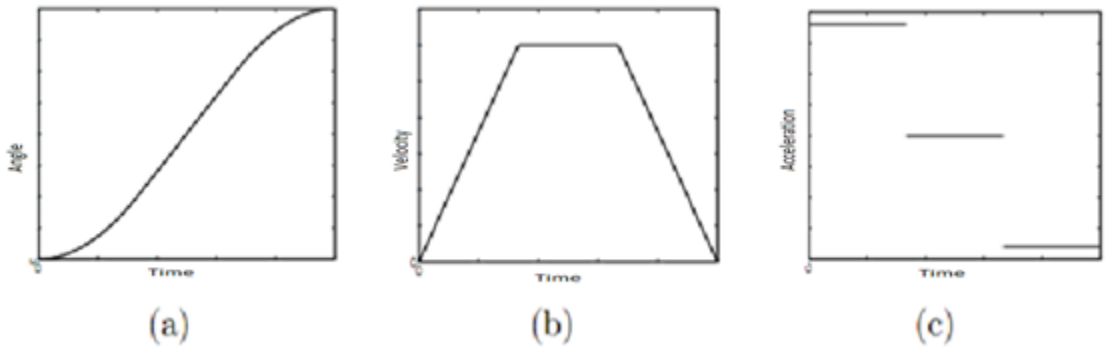


Figure (3.11): *(a) LSPB trajectory, (b) Velocity profile for LSPB trajectory (c) Acceleration for LSPB trajectory.*

The complete LSPB trajectory is given by:[7]

$$q(t) = \begin{cases} q_0 + \frac{a}{2}t^2, & 0 \le t \le t_b \\ \frac{q_f + q_0 - Vt_f}{2} + Vt, & t_b < t \le t_f - t_b \\ q_f - \frac{at_f^2}{2} + at_f t_b - \frac{a}{2}t^2, & t_f - t_b < t \le t_f \end{cases}$$

Where:

q: is the angle of rotation
q_0: is the initial angle (at the starting position)
q_f: is the final angle (at the end of the motion position)
V: is the velocity at the constant portion
t_b:is the blend time
In order to make this motion possible, the velocity must be between two limits:

$$\frac{q_f - q_0}{t_f} < V \le \frac{2(q_f - q_0)}{t_f}$$

The time consumed by the robot while moving is not constant for any motion, thus for each motion the distance between the two addresses will be calculated, by knowing the speed of the first two motors then the time needed for this motion is calculated and transferred to the equations of the trajectory planning.

3.2.4.3 Cartesian Space Technique

In Cartesian space, the position and orientation of rigid body can be clearly defined. The problem of planning trajectories that enable the manipulator's end effector to track a given path in Cartesian space is investigated in this section. The user specifies the desired end-effector path, the travelling time, and tool orientations along the path. It is helpful when there is an obstacle in the path but it is much more complex than joint space technique. Some common techniques for Cartesian space trajectory planning are:

(i) Parametric Description of a Path

(ii) A Straight-Line Path

(iii) A Circular path

(iv) Position planning

(v) Orientation Planning etc.

An alternative method that was used to get a cubic polynomial trajectory is using a second order transfer function with a step input. Each time the Robot should move, MATLAB calculates the desired angles to be crossed by each motor, this piece of information is generated as a step input that it is inputted to the second order transfer function, the result will be a cubic trajectory transferred to the controller as the reference input for each motor.

3.2.5 Mechanical Calculations

The downward bending moment at the farthest part of the robot arm shown in figure (3.12) is:

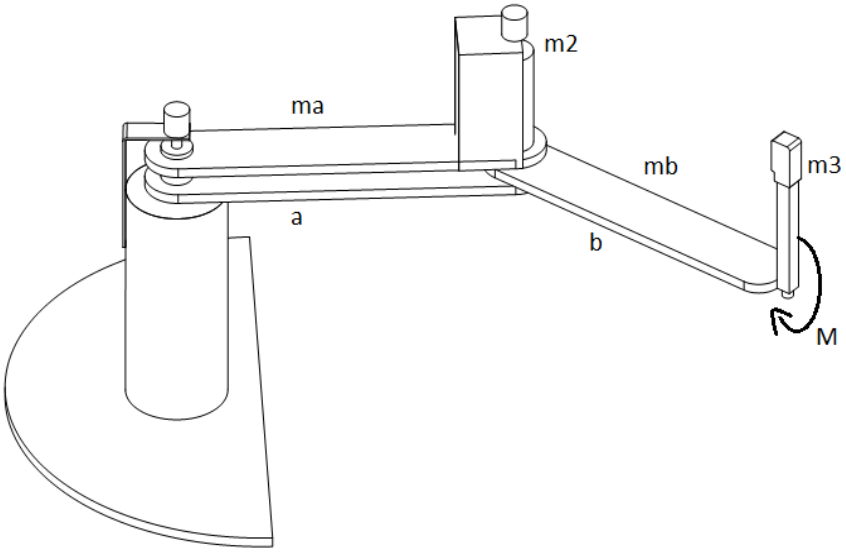


Figure (3.12): Bending Moment affecting the robot.

$$M = m_3 * 9.81 * ({m_3}^2 + b^2 - 2ab\cos(\theta_2)) + m_2 * 9.81 * a + m_b * 9.81 * \left(a^2 + \left(\frac{b^2}{2}\right) - 2a\left(\frac{b}{2}\right)\cos(\theta_2)\right) + m_a * 9.81 * (\frac{a}{2})$$

Where:

M: bending moment

a:length of the first link

b:length of the second link

θ_2:angle of rotation of the second motor

m_a: mass of the first link

m_b : mass of the second link

m_2: mass of the second motor

m_3: mass of the linear link and the gripper mechanism

The applied torques of a two link planer arm ignoring the gravity-containing terms are:[14]

$$T_1 = m_2 l_2^2\left(\ddot{\theta}_1 + \ddot{\theta}_2\right) + m_2 l_1 l_2 c_2\left(2\ddot{\theta}_1 + \ddot{\theta}_2\right) + (m_1 + m_2) l_1^2 \ddot{\theta}_1 - m_2 l_1 l_2 s_2 \dot{\theta}_2^2 - 2m_2 l_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2 \quad (3.23)$$

$$T_2 = m_2 l_1 l_2 c_2 \ddot{\theta}_1 + m_2 l_1 l_2 s_2 \dot{\theta}_1^2 + m_2 l_2^2 (\ddot{\theta}_1 + \ddot{\theta}_2) \quad (3.24)$$

Where:

$$m_1 = \frac{2}{3} m_{Al} + m_{m2}$$
$$m_2 = \frac{1}{3} m_{Al} + m_{gr}$$

Where:

m_1: Equivalent mass placed at the end of the first link
m_2:Equivalent mass placed at the end of the second link
m_{Al}:Mass of the Aluminum link.
m_{m2}: Mass of the second motor.
m_{gr}:Mass of gripping mechanism.
l_1: Length of the first link.
l_2: Length of the second link.
$c_2 = cos\theta_2$
$s_2 = sin\theta_2$

Ignoring the higher order terms and with the assumption of very small angles θ_1 and θ_2 ($\sin\theta = \theta$, $\cos\theta = 1$), the above two equations (4.23) and (4.24) becomes:

$$T_1 = m_2 l_2^2\left(\ddot{\theta}_1 + \ddot{\theta}_2\right) + m_2 l_1 l_2\left(2\ddot{\theta}_1 + \ddot{\theta}_2\right) + (m_1 + m_2) l_1^2 \ddot{\theta}_1 \quad (3.25)$$

$$T_2 = m_2 l_1 l_2 \ddot{\theta}_1 + m_2 l_2^2 (\ddot{\theta}_1 + \ddot{\theta}_2) \quad (3.26)$$

Now:

Now applying these equations in specific motion of the robot, that is going from the lower left corner a1 to the upper left corner h8, this motion needs the highest torque to be applied by the second motor. As it is shown in equations (3.1) and (3.2) the torques are a function displacement, velocity and acceleration of each joint, which is called 'The Trajectory', this trajectory is specified and calculated in details in section 3.2.5, so plotting the two torques as varied with time going from a1 to h8 will give the result that is shown in Figure(3.3).

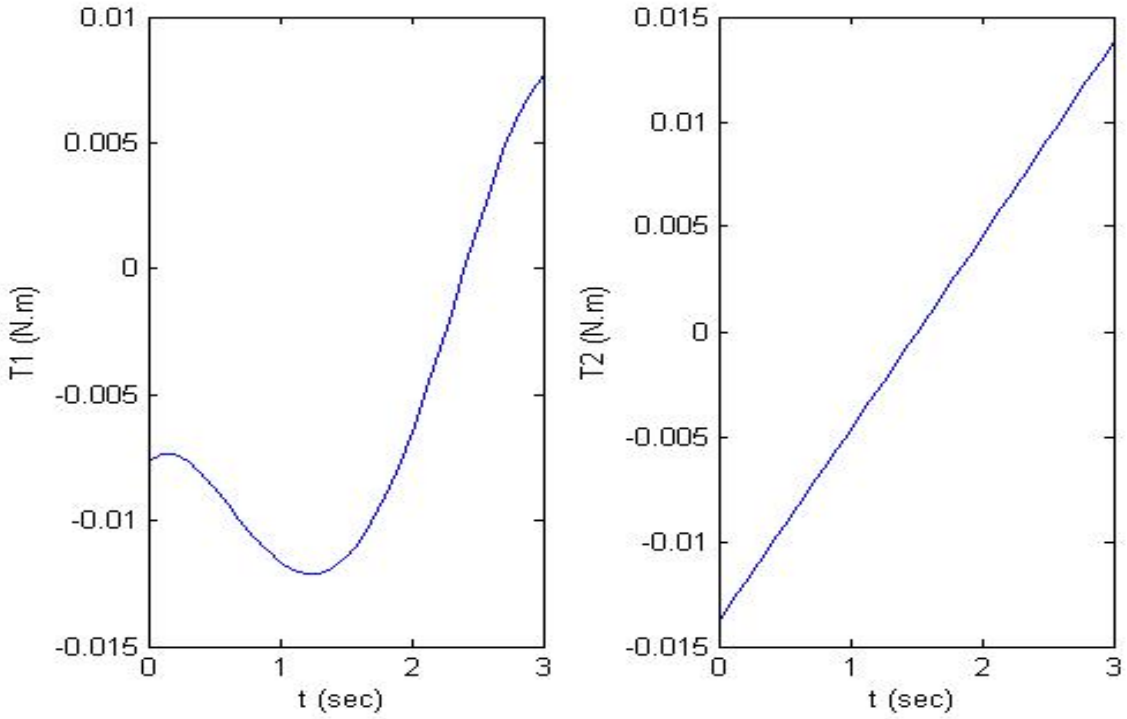


Figure (3.13): Torques applied by the two joints when going from a1 to h8.

In Figure (3.13) T_1 is less than T_2 because the first motor will apply only a deflection of about 11 degree, which means low torque is needed.
The maximum torque that will be applied by the first motor will result in going from position a8 to h1; this motion requires the maximum angular deflection crossed by the first link. This applied torque is shown in figure (3.14).

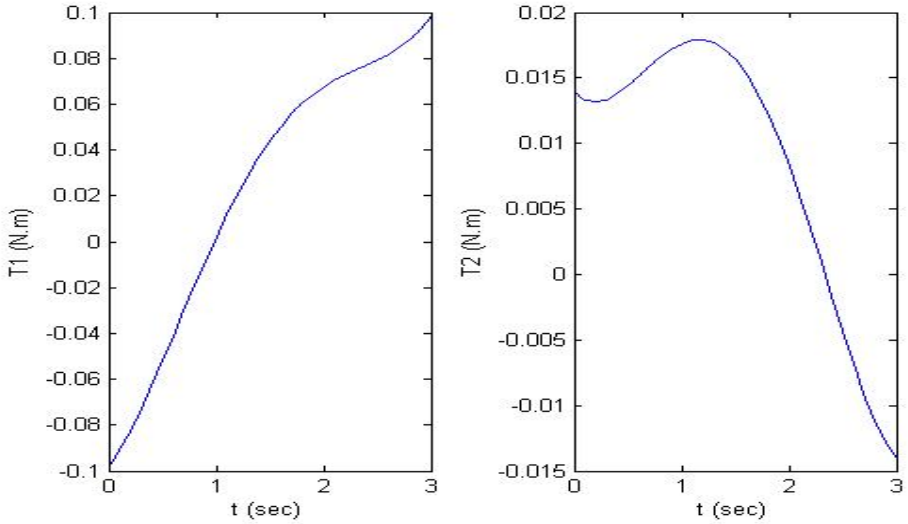


Figure (3.14): Torques applied by the two joints when going from a8 to h1.

The rest of this chapter discusses the main project components, as shown in Figure (3.15) below. The designing of the chessboard according to the project requirements, and customization the chess software to interact with the, thus to achieve the main project requirements.

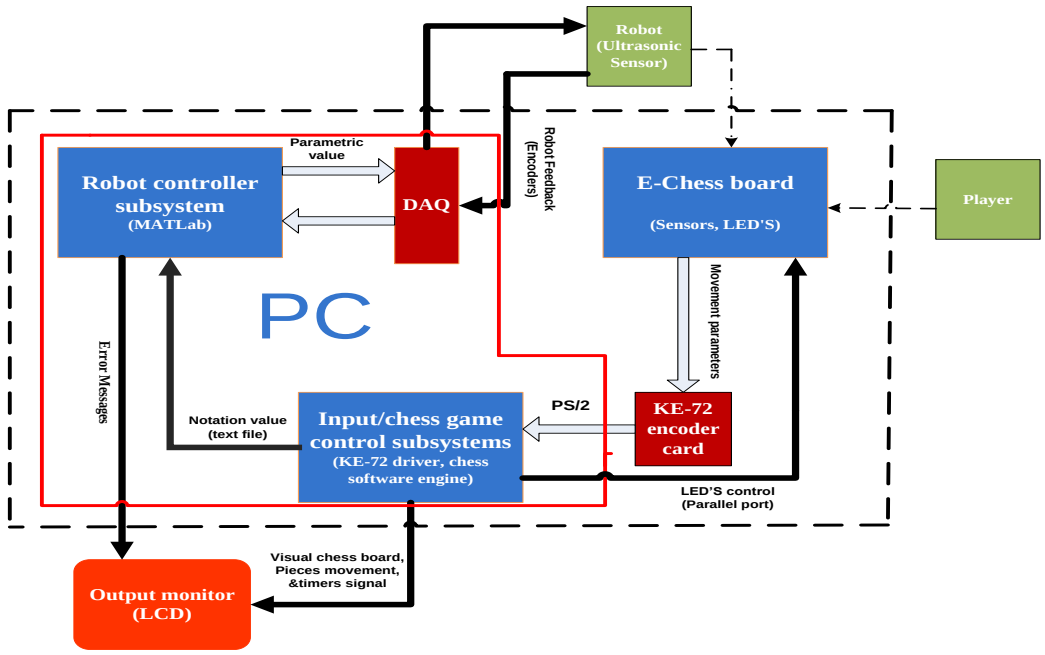


Figure (3.15): Chess Playing Robot Main Block Diagram

3.3.Overview

This part discusses the main project's components in details, which are done by students of Computer System Engineering. It explores selected option forms, system hardware and system software that are mentioned in the previous chapter.

It will explain the hardware chessboard design, sensing, and LED's controller design; also, it will explain the design phases of the system software in details. As an addition to the main customized phases to meet the project requirements.

3.4Hardware Design Requirements

E-board is a highly active component, which is fixed between the player and the robot. All chess activities during the whole game will be on this board. So designed hardware should satisfy the following objectives:

- Sense any changes on the board for the presence or absence of a piece.
- Locates any changes on the board for the presence or absence of a piece.
- Flashes possible squares using LED's.
- Configures chess game through buttons.

These objectives are accomplished using KE-72 encoder, Magnetic sensors, and LED's driver circuit, these are the main components that will be used

3.5 High Level Hardware Design

This section describes major mechanisms of chessboard to achieve its hardware objectives. Figure3.15 shows these mechanisms and its interfaces, the detection process will be through keyboard encoder KE-72, while the lightning process is done through a decoder circuit.

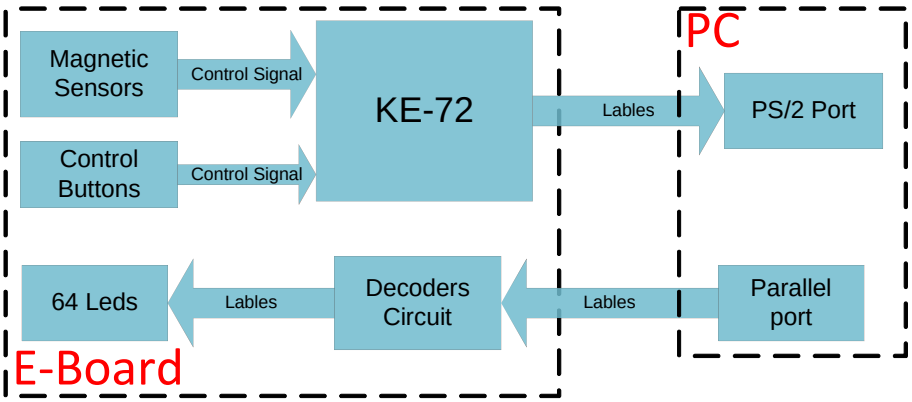


Figure (3.16): E-Board Block Diagram

3.5.1- Electrical Chess Board (E-Board)

3.5.1.1- Sensing Inputs (Sensors)

3.5.1.1.1 - Magnetic Sensors

Chessboard sensing technique is done through Hall Effect Magnetic sensors. One sensor is placed under each board square; these sensors are affected by magnetic flux from magnetic pieces. The magnetic piece is fixed inside the chess pieces underneath, as shown in the Figure3.16.

Figure (3.17): Chess Piece

The output of each sensor is connected directly to a keyboard encoder (KE72) input, each KE72 input sends two values to the PC ,one value when it is closed (Place), the other value is when it is opened (Pick up).

However, the Chessboard is able to sense the status of each square position.

The basic interface for a digital Hall Effect Sensor[25]is a single resistor. When a resistor is used in conjunction with a current sinking sensor, it is normally tied between the output and the plus power supply; it is referred to as a pull-up resistor. Figure 3.17, illustrates pull-up resistor (R) connected between the sensor and its load.

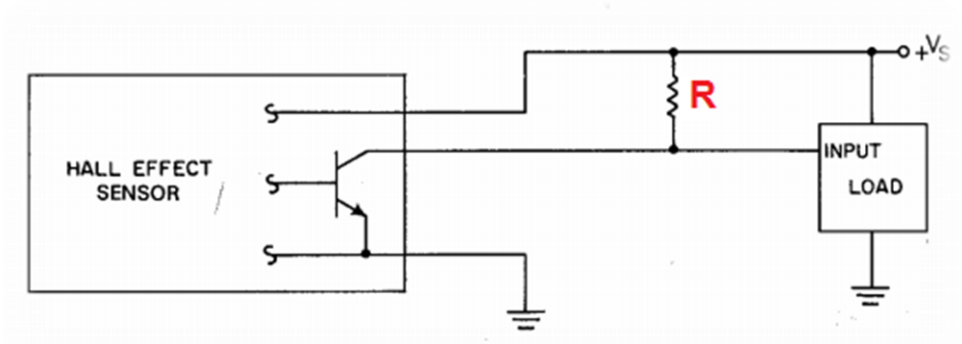


Figure (3.18): Sensor Connection to operate as digital switch

When the sensor is activated, the input to the load falls to near ground potential, independent of the pull-up resistor.
On the other hand, when the device is de-activated, the input to the load is pulled-up to near VS. If the pull-up resistor is not present, the input to the load could be left floating, neither at ground nor VS potential.

Each row of the eight sensors are connected as shown in Figure 3.18, to voltage source (VDD) and grounded with (GND). Likewise, each output pin is connected to a specific KE72 input pin. This connection of sensors is meant to operate one row; and the other rows are implemented in the same way.

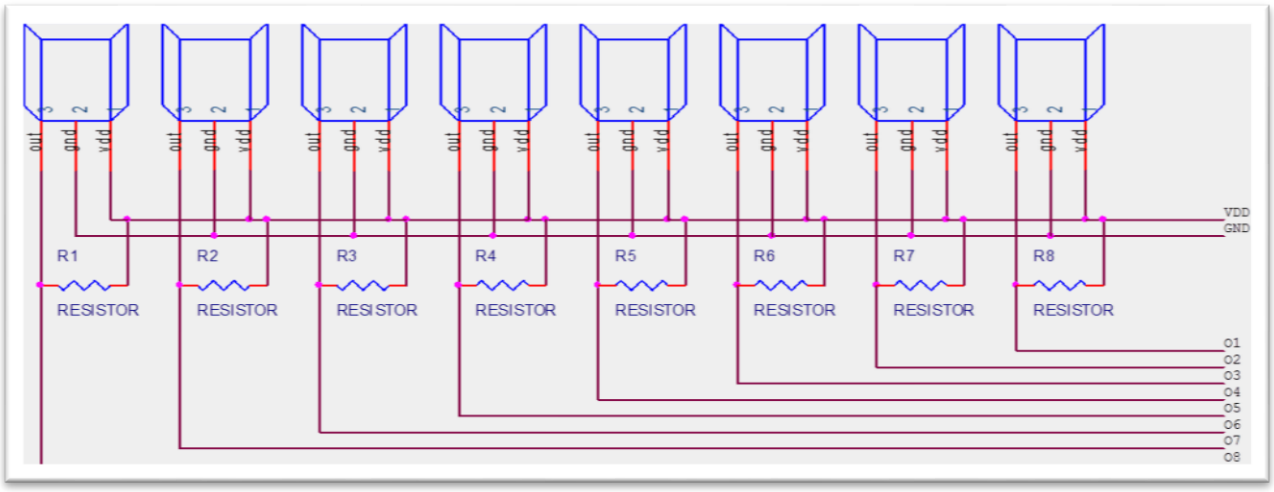


Figure (3.19): Sensor Connection Schematic

3.5.1.1.1.1 Estimated Calculations

This section shows estimated calculation for a magnetic sensor. voltage supplied is (+5V), where the pull-up resistor equal 47 Ohms, that is:

$$Is = \frac{Vs}{R}$$

$$Is = \frac{5}{47} = 106mA \ , foronesensor$$

Thus the 64 sensors consume current of It,

$$It = currentforOnesensorIs * numberofsensors$$

$$It = 106 * 32 = 3.404A$$

Where It equals the maximum current consumed by sensors.

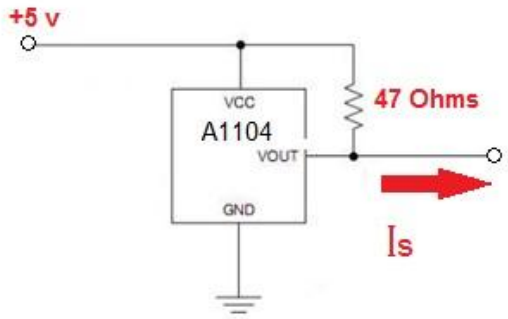


Figure (3.20): Sensor Circuit

3.5.1.1.2 - Interface KE-72

The magnetic sensors are interfaced with KE72; the KE72 features 72 individual inputs. These inputs are activated by connecting the input (through a Magnetic sensor) to the common ground on the KE72 I/O header.

Figure 3.19, shows the KE-72 input header, Figure3.20 shows one line of sensors interfaced with the KE-72. Each input is programmed to output a unique single sign when it's activated and when the input is released.

The configuration for the input responses is stored on board KE72 in EEPROM. This memory is maintained until the configuration is changed again.[24]

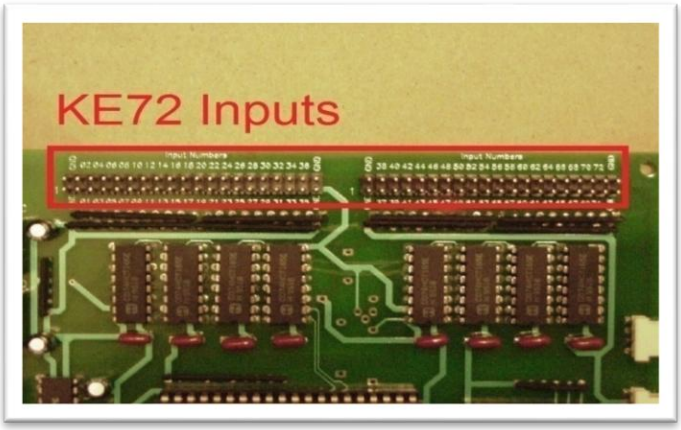


Figure (3.21):KE-72 Input Header

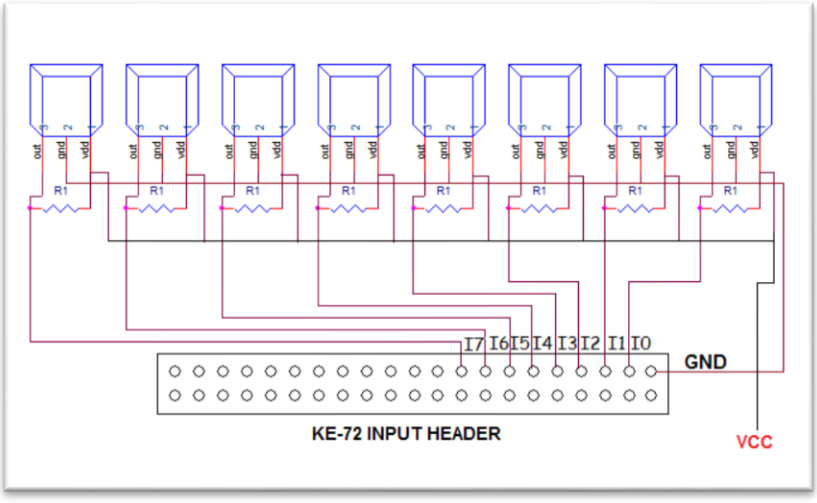


Figure (3.22): Interface sensors with KE-72 header.

3.5.1.1.1 Programming the KE72 Input Responses

Each of the KE72 inputs is programmed to output a standard keystroke when activated. In addition, Inputs also is set up to send a different keystroke when the input is released.

Programming of the inputs is accomplished by creating a text file that designates which inputs are being used, and determines the keystroke responses. Once the text file has been saved, program KE72LOAD.EXE is used to save the configuration onto the KE72. The configuration is loaded through the keyboard port.[24]

When KE-72 unit responses to inputs, KE-72 direct these responses to the PC keyboard port PS/2 as keystrokes then, the responses are handheld by operating system.

According 0X88 methods for chessboard presentation, specific values must assign to each KE-72 input response, the following table shows these values[11].

Table 3.2- KE-72 Input values

	Pin	Value	Pin	Value	Pin	Value	Pin	Value	Pin	Value	Pin	Value	Pin	Value	Pin	value
Row1	0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7
Row2	8	16	9	17	10	18	11	19	12	20	13	21	14	22	15	23
Row3	16	32	17	33	18	34	19	35	20	36	21	37	22	38	23	39
Row4	24	48	25	49	26	50	27	51	28	52	29	53	30	54	31	56
Row5	32	64	33	65	34	66	35	67	36	68	37	69	38	70	39	71
Row6	40	80	41	81	42	82	43	83	44	84	45	85	46	86	47	87
Row7	48	96	49	97	50	98	51	99	52	100	53	101	54	102	55	103
Row8	56	112	57	113	58	114	59	115	60	116	61	117	62	118	63	119

3.5.1.2- Square Lighting (LED's)

"One LED under each square" is the adapted method for light chess board; sixty-four positions are needed to be decoded. The sixty-four positions are treated as eight rows, where each row has eight LED's, one of eight rows must be selected in order to select one LED in the row.
The following block diagram shows the complete module for the LED's driver circuit.[2]

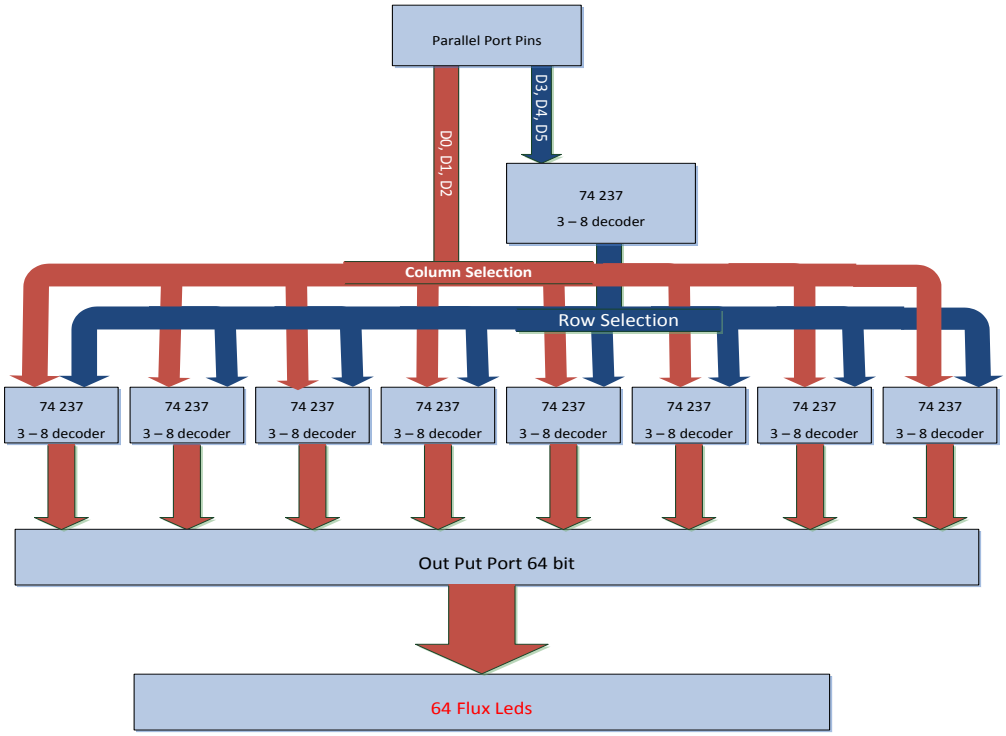


Figure (3.23):LEDs Driver Block diagram

On the other hand, the following schematic shows the connection between the LEDs and driver circuit.

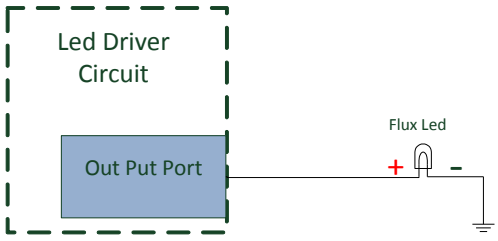


Figure (3.24): LEDs Connection with LED Driver circuit.

[2] Appendix A

Parallel port (PLT) is used to accomplish this method, PLT offers eight data bits, as shown in Figure 3.23, six data bits are enough to decode the sixty-four LED's. using first three data bits (D2,D1,D0) to decode eight LED's (Column Selection) on the same row, followed by three bits (D5,D4,D3) to decode eight rows(Row Selection) .

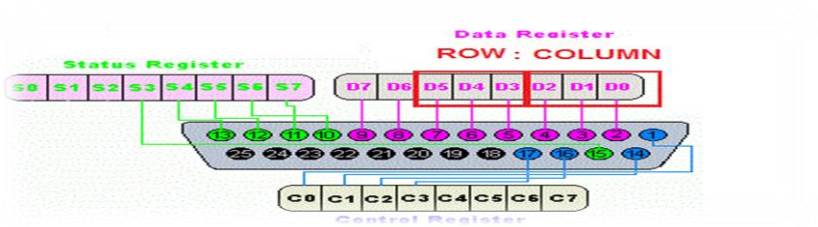


Figure (3.25): Parallel Port Socket

3.5.1.2.1 Estimated Transistor Calculations

LEDs Driver circuit contains transistors used as switches, each transistor control one led, following the calculation estimated to operate this transistors, Collector Resistor is chosen to give the best light intensity with least current consumption, this is done using Variance Resistance Box, the collector resistor equal 200 Ohms. Then:

$$CollectorCurrentIC = \frac{CollectorVoltage}{CollectorResistor}$$

$$IC = \frac{5V}{200\Omega} = 25mA$$

Then, Base transistor current is calculated using:

$$BaseCurrentIB = \frac{collecterCurrent}{\beta}$$

$$\beta; is\ D.C\ current\ gain\ equal\ 250$$

So,

$$\boldsymbol{IB = \frac{25m}{200} = 0.125mA}$$

Now, Finding base resistor using the following equation:

$$BaseResistor = \frac{BaseVoltage - base_{EmmiterVoltage}}{Base\ current}$$

Base-Emitter Voltage equal 0.7 Volts

$$RB = \frac{(5-0.7)V}{0.125mA} = 34.4K\Omega$$

So, each transistor will consume a current of:

$$TransistorCurrent = IC + IB$$
$$Transistorcurrent = 25mA + 0.125mA = 25.125mA$$

If all transistors are in the saturation region, then the maximum current will be:

$$MaximumtransistorsCurrent = TransistorCurrent * Transistornumber$$
$$MaximumtransistorCurrent = 25.125mA * 70 = 1.758A$$

3.5.1.3- Control Buttons

At the beginning when a player starts a game, he/she should make some configurations and settings such as level selection, help mode and time settings, these settings are controlled by using four push buttons that are placed beside the chessboard. According to that, buttons must do these four actions:

- UP: Go up through menu.
- DOWN: Go down through menu.
- OK: Make selection.
- EXIT: Exit menu or return to previous menu.

The following block diagram shows these four push buttons which are connected and interfaced through the KE-72, each button is interfaced to give specific value when pressed.

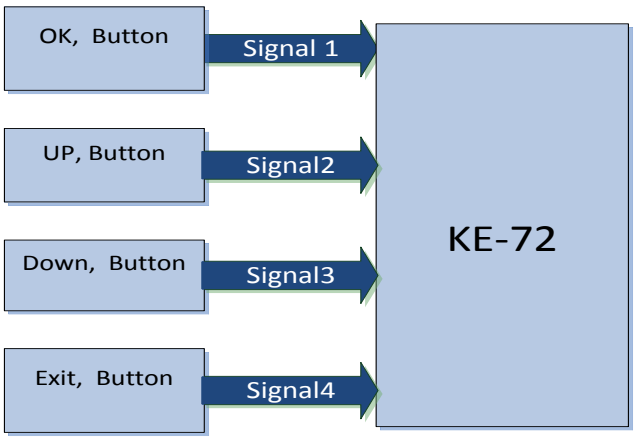


Figure (3.26): Control Buttons Connection with KE-72

3.5.2- Electrical Chess Board Interface

The following block diagram shows the connection between E-board and PC, magnetic sensors are interfaced with KE-72; andKE-72 is interfaced with the PC through the PS/2 Port.

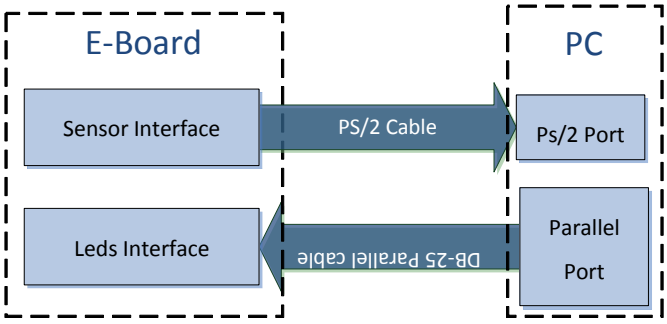


Figure (3.27): Interface E-board with PC

LED's are interfaced through parallel port, parallel port has eight data bits, LED's driver make use of six data bits just to drive LED's.

3.6- System Software Design Requirements

The open source chess game that is used does not fully satisfy the project requirements. Thus, the open source chess game was studied and customized to meet the main design requirements of the system software. The software must be modified to achieve the followings:

- Design Start up Main menu screen.
- Design 3D chess board.
- Design Interactive classes that can read the E-board.
- Design LEDs circuit controller to light the suitable squares.
- Configure the intelligence class with the Robot's controller which will be designed using MATLAB
- Design cheating algorithms, able to detect most cheating cases.

3.7- Software Design

This section will discuss the system software that implemented to run the project as the requirements.

3.7.1-Interactive Input and Output with E-board

The designed E-board will be interact to send changes value to the system, in addition to receive value form the system, the following subsection will discuss these phase

3.7.1.1- Startup Main Menu

Unlike common chess games on a computer where the player uses a keyboard or a mouse, the player here uses an actual chessboard. In order to facilitate an easy control for the player, the E-board has few buttons to enable the player start a new game, modify the game's option and to resume or to exit the game. The previous functions must be considered in the design of startup main menu and the buttons input must be adjusted to activate the selected function. The main menu design gives the players a good user interface, which makes it easy to play.

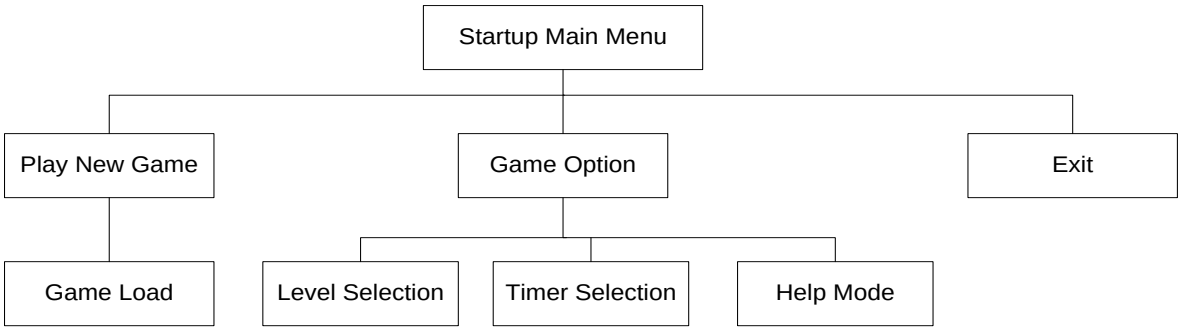


Figure (3.28): Startup Main Menu Block Diagram

3.7.1.2 -E-Board Inputs

Interactive environment must exist between the physical E-board and System input. Therefore, E-board consists of sixty-four square each has a unique ID. Each ID stores the picking and placing of a specific piece in Encoder memory. Any change that occurs on the E-board will generate a specific value for the system input that dictates what type of movement to generate.

3.7.1.3- E-Board Outputs

When the player picks any piece, it will lead to generate output from the software to lighten up the LED's that are placed underneath each square. The LEDs lightened according to the possible square that the piece can placed on. The Values that come out to the LEDs circuits through the parallel port cable. The light square will be ON until the user place the picked up piece on valid position or release it, and if the player places the piece on Invalid Square.

Lightening the possible square is one of the game options to give the beginner player help hints, so if the player does not select this option, the system will light just the selected piece square.

3.7.2- Chess Engine

The open source chess engine is not compatible with the projects needs and requirements, as a result, the important classes will be customized to work as the project requirements. Other classes will not be used.

The main process is to customize the core chess engine to read E-board changing values to decide whether the movement valid or invalid, convert the 2D chess visual board to 3D chess board, and to customize the intelligence classes to be integrated with Robot controller in MATLAB environment.

3.7.2.1- Human Player Turns

As known in chess game rules, human player has to play first at the beginning of the game to enable the intelligence to play back.

All chess pieces should be placed at the starting position, as the chess game rules impose. When the player picks up any piece, the interactive input class will handle this change.

When the system input gets the value for this change, it will be decoded before it sends it to the proper class in chess engine in order to check this change for accuracy and to get the possible movement value for it.

The parallel port function will receive these possible movements value to light up the possible square on the E-board.

While the system input waits for the next move, the lightened square will be ON. When the next move occurs, the "Checking" algorithm verifies the validity of this motion, which is by comparing it with the chess game standards. If the result is correct then it passes it to chess engine to activate the intelligence, and switches the turn for the Robot.
The player can play with the defined game options, which are as the following:

- Play new game with level of difficulty from level one to level sixteen, each level is harder than the previous one, and it depends on the number of search depth for the intelligence search.

- Play new game with timer countdown with minimum period of60 seconds and maximum period of 5 minutes, if the player exceeded these interval time, he will automatically lose the round.

- Playing the game with help mode or not.

Any cheating occurs the game will pause till the player plays his last move according to the chess rules.

3.7.2.2- Robot Player Turns

Chess engine has great intelligence in planning the next move. When the human player places out the last piece of his turn, the chess engine senses it and plans the upcoming move accordingly. After the decision is made, a signal is sent to the robot to make the correct move.

Robot plays back its move according to the passed intelligence level. When the intelligence completes thinking, it will send the value to the Robot controller. The human player waits for the Robot to do its move by picking up the piece and place it on the valid square.

When the Robot picks the piece up and place it, this makes the system input to handles the changes on the E-Board. Getting these values and comparing them with intelligence value to make sure that the Robot plays correctly.

If the Robot makes a movement of fraud, the game will pause then signals will be sent to the Robot to return the piece to its original place. After making sure that the robot returns the piece on a valid square, the chess engine gives the turn to the user to play his round, if it's correct, then the turn will repeated. In addition, signals of datum will be sent to the robot to play his turn again.

Interactive class can receive feedback data from the robot controller such as when the robot picks the piece and places it. These values help the system making sure that the player is the robot and to know if the human is intervening in the Robots turn.

3.7.2.3- Special Cases for Handled Input

As mentioned before, any changes occur on the E-board, will activate the system input to receive these changes.

However, as known the chess game has more than one type of movements, each move differs from other moves by the amount of picking up or placing the pieces. In addition, by the action that will be generated after played each move.

Eventually, the system input should be compatible and able to distinguish these types of moves. The following cases will be followed with a brief explanation for each case:

- Normal move: player or robot can pick up and place one piece in each turn , so the system input get these two changes, otherwise it will be shifted to cheat mode.

- Kill Move: in this case, the player picks first his piece, then picks the enemy's piece and finally places the piece. Three moves occur at the same turn one placing move and two picking up moves.

- Promotion Move: when the pawn reaches the end of the board on the other side, promotion will be activated. User selects the new wanted piece instead of the pawn.

- Castle Move: as chess rules, the player can replace the two positions of the rook and the king if certain conditions were met. Therefore the system have to handle two types of picking up, and tow type of placing movement.

All these type of moves will process independently by the system input without errors or cheating.

3.7.2.4- 3D Chess Board

The open source chessboard was implemented in 2D. Switching the 2D to 3D is one of the main processes in the customization steps.

Any valid movements that may occur on the E-board will reflect on the 3D board, therefore this increase the excitement, and entertainments for the game. In addition, increase the educational views how this system works and all these units are interact with each other.

3.7.2.4.1- 3D Chess Board Design Phases

In order to design a 3D chessboard with its segments and animations, it requires many phases, as the following:

Phase 1: 3D Models

To design the 3D chess game, it is required to use 3D chess piece, and chessboard squares. The used model in this project is from the open source community that was designed using 3D Max and converted to FBX files.

Phase 2: World Transform Model

Before loading the 3D models, the world transform of each model must be defined. The world transform is meant to define the position for each model to be drawn according to that position.
The chess board consists of sixty four square; each square has unique value of world transform that differs from one another. In addition, the world transform of chess pieces will change based on each movement.

Phase 3: Display 3D Models

Since the screen that will be used is a 2D. Therefore a method should be used in order to allow 3D model to be displayed. By using triangle picking pipeline that converts the model from local view to world space view, and views matrix to change the positions and directions of the models to be shown as needed.
Eventually, required model will be displayed according to the world transform.

Phase 4: Interfacing 3D with Chess Engine

The player picks up the piece, which generates a move on the E-board. This movement generates a value, which passes to the chess engine. The 3D receives this value and selects which piece will move and what are the possible positions for that movement. When the player places the piece, the chess engine also receives a value that defines the type of the movement and the new position. The 3D receives those changes to create an animation representing the player movement.

3.7.3 Intelligence and Robot Controller Combination

This subsection discusses the used ways to combine chess intelligence system with robotics system.

3.7.3.1 .Net and MATLAB integration.

Since there are two different environments, .Net those runs the chess engine software and MATLAB that runs the robot controller system. Connection between them must exist, by using UDP Protocol to create server with static IP on the XPc target that

has the robot controller and client with static IP on the personal computer that has the chess engine.

When the intelligence activated and the values become ready; the connection established between client and server to send these data, which are name of each piece, start position, next position and the type of this move (Normal, Kill, Castle, or Promotion).

3.7.3.2 MATLAB Feedback

In order to increase the efficacy of the game, the robot has the ability to send feedback to the chess engine for accuracy. The robot feedback will tell if the piece is being picked up or placed on E-board.

The feedback from the robot is essential to monitor possible cheating, any other moves outside of the feedback will be considered as cheating.

3.7.4 Cheating Discovery Solution

Discovering any cheating during the game is important, many cases are existed that may be done by the user. This section explores these cases and designs the best solution to solve these problems. Below, there are lists of cheating cases and the solution for each one.

3.7.4.1Pick more than one piece at the same time

If there is no possible moves to kill or castle or promote, the player should pick up one piece at his turn.

Each time a player picks up any piece, the system will check the current position and history, thus the system will determines the piece category, and will determine whether the coming location valid or invalid.

If the player picks up more than one piece the system looks up for three possibilities:

- Killing move: in this case; first the system checks if the player is selecting the valid piece to be killed or not. If not it will be move of fraud.

- Castle move: when the player chooses the castle move, he should pick up the king piece to place it into a valid position. Then picks up the rook piece to place it on its valid position.

- Promotion move: At this case, the system will wait the player to pick up new piece from outside the board to replace it with old piece (pawn).

3.7.4.2Play with invalid move

If the player places any piece in invalid location, the cheating algorithm must realize this error by sending the user a message regarding it. In addition, the game will be paused until the piece is returned to its original place, and then places it into a valid position.
If the player selects to play with timer mode, and tries to make a move with cheating, he will lose if the timer exceeds the defined interval period.

3.8 System Modeling

This section expose the system design by using different ways of modeling,.

3.8.1 Unified Modeling Language (UML)

This subsection exposes the chess engine and the 3D chess board using the unified modeling language

3.8.1.1 Startup Main Menu

The following diagrams show the main menu classes of the chess playing Robot game.

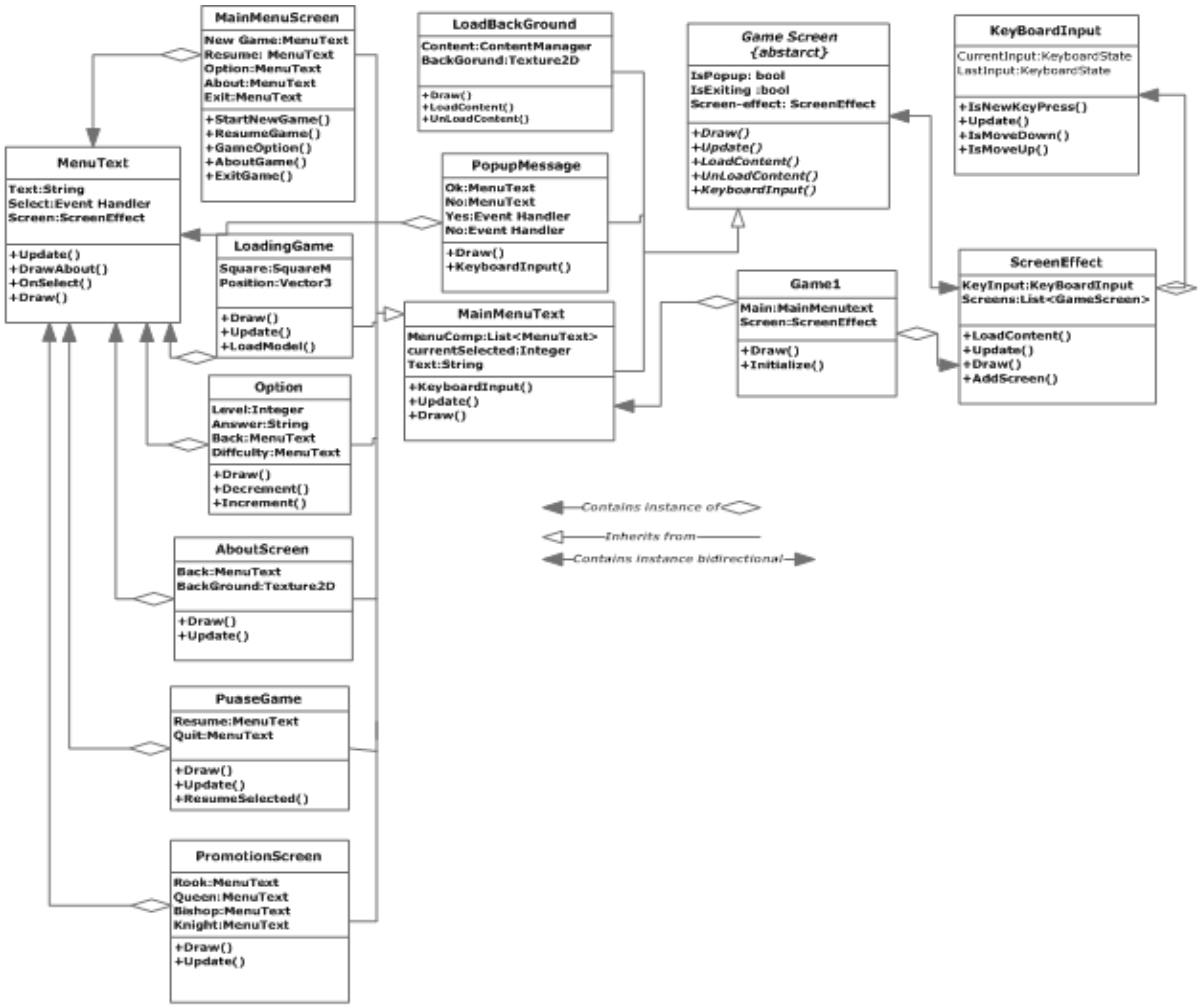
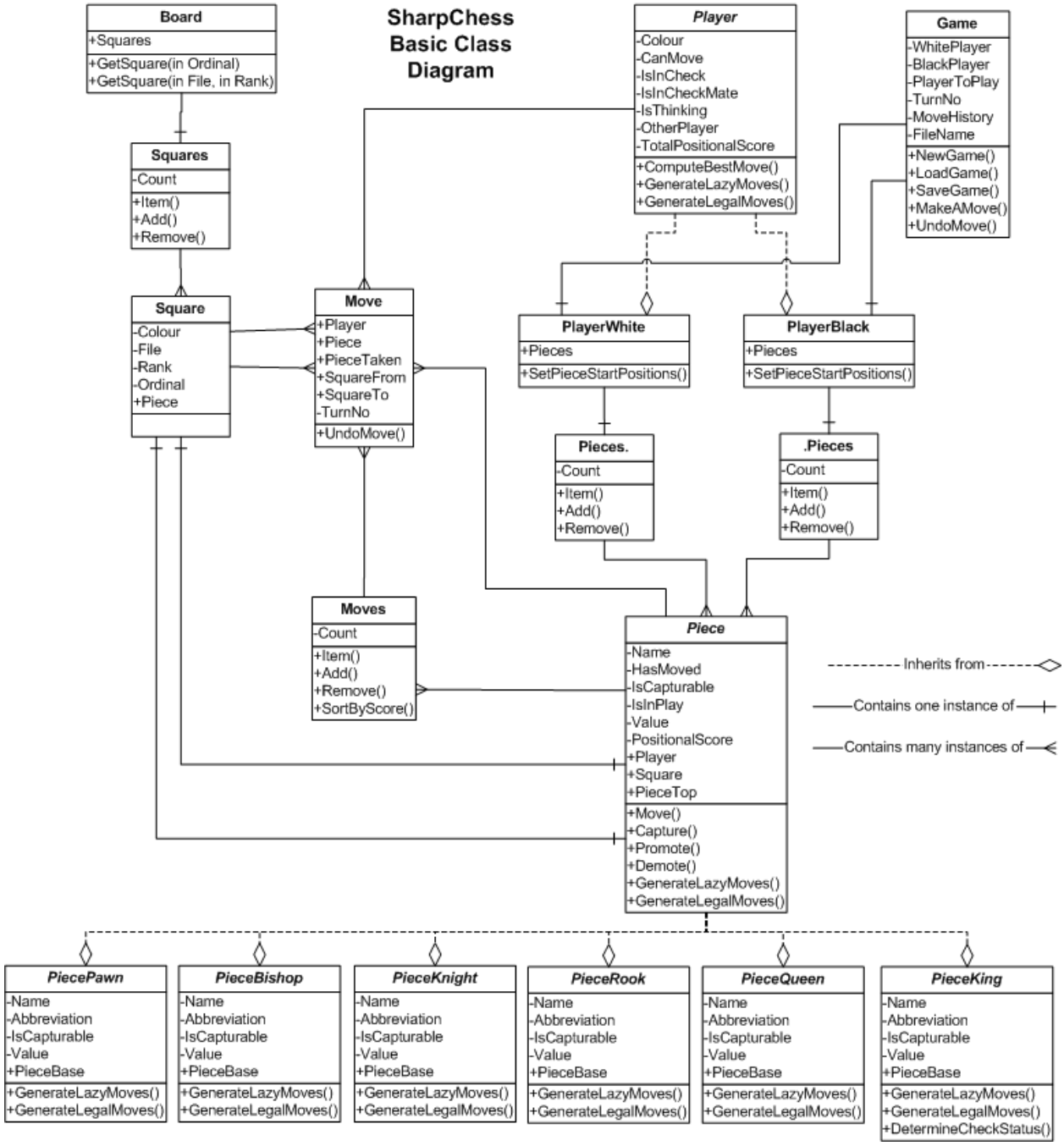


Figure (3.29): Startup Main Menu Class Diagram

3.8.1.2 Sharp Chess Game

This diagrams[10]shows the basic class diagram for the open source sharp chess



game.

Figure (3.30): Chess Engine UML Class Diagram

3.8.1.3 Chess Board (3D)

This class diagrams show the main classes that used to implement 3D chessboard.

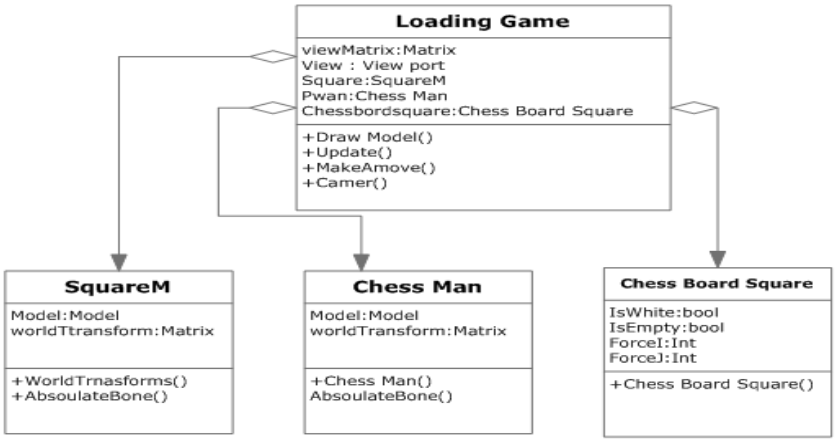


Figure (3.31): Chess Board Basic Class Diagram (3D)

3.8.2 General Flow Chart of System and Algorithms

The main classes and function of the system

3.8.2.1 Interactive Input with E-board

This flowchart shows the main process of interactive input with E-Board

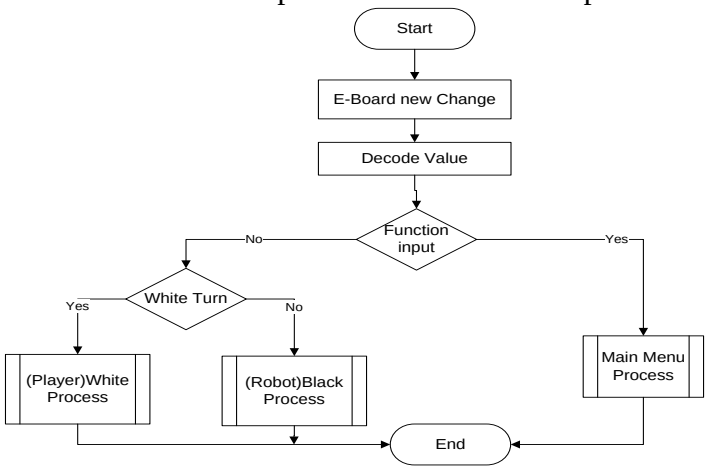


Figure (3.32): Interactive Input with E-board General Flow Chart

3.8.2.2 Interactive Output with E-board.

This flowchart shows how the possible movements out to parallel ports to light the possible square on the E-board.

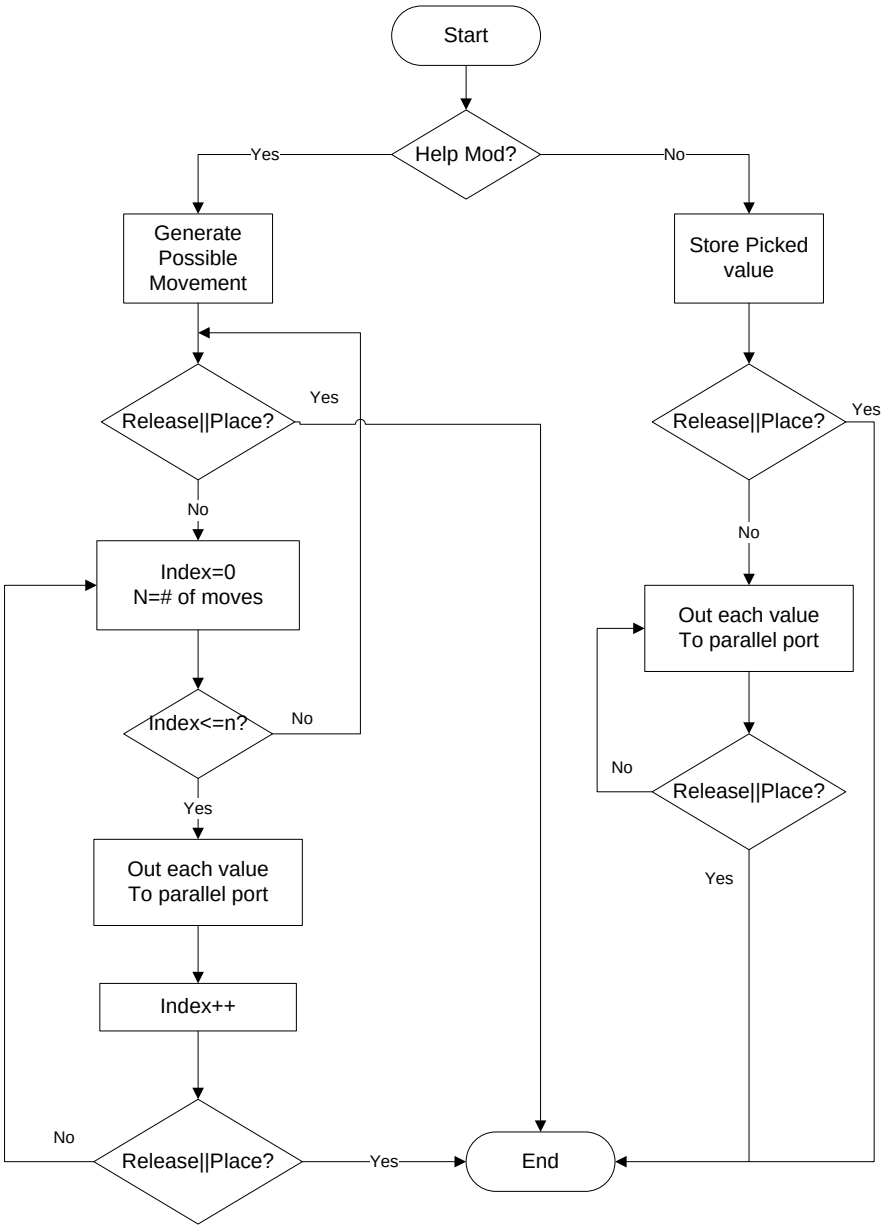


Figure (3.33): Parallel Port Controller Flow Chart

3.8.2.3 (White) Human Player Turn

Picking and placing or releasing the piece by the player is shown by the following flow chart

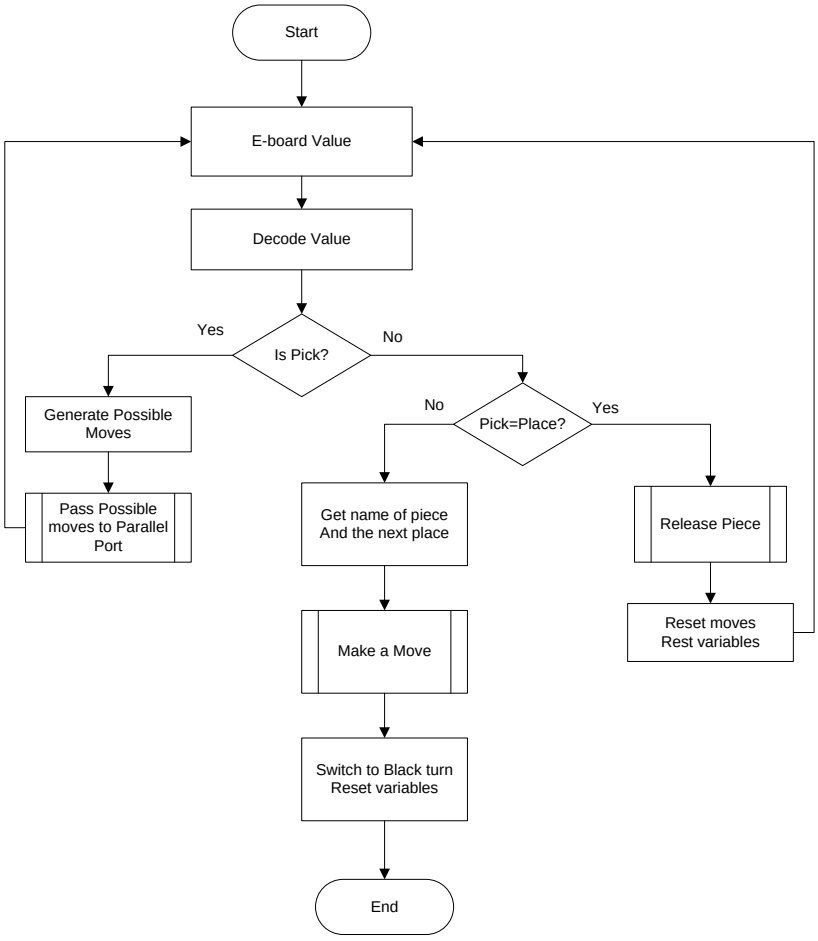


Figure (3.34): White Player Turn

3.8.2.4 (Black) Robot Player Turn

This flowchart shows the process how to configure intelligence with robot controller:

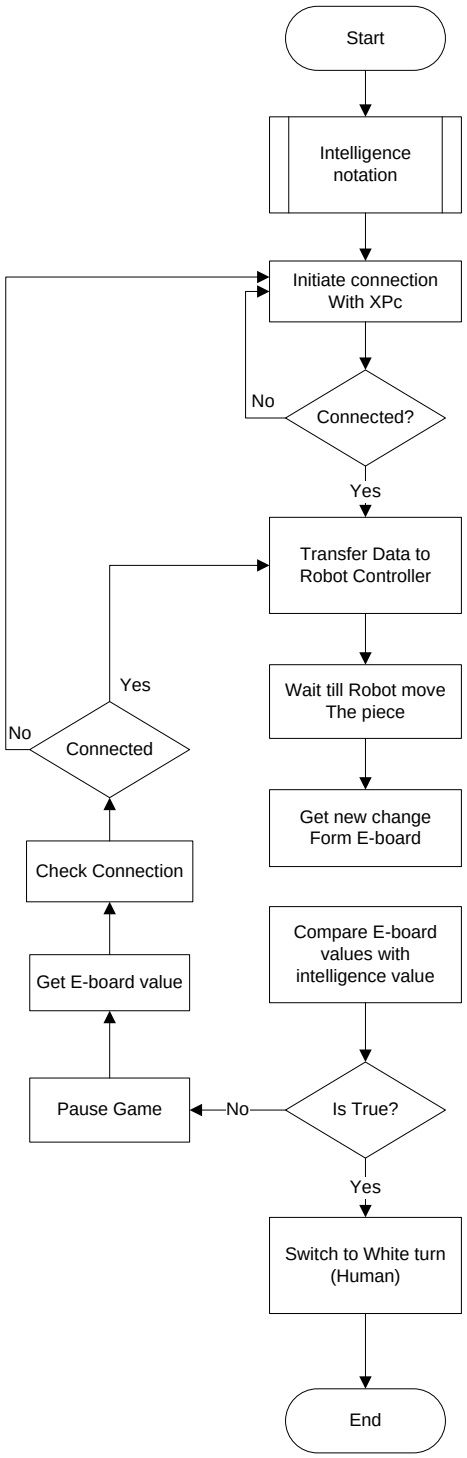


Figure (3.35): Robot Turns Flow Chart

3.8.2.5 Castle Move

This flowchart shows the steps when the Castle move occurs for human player or Robot player.

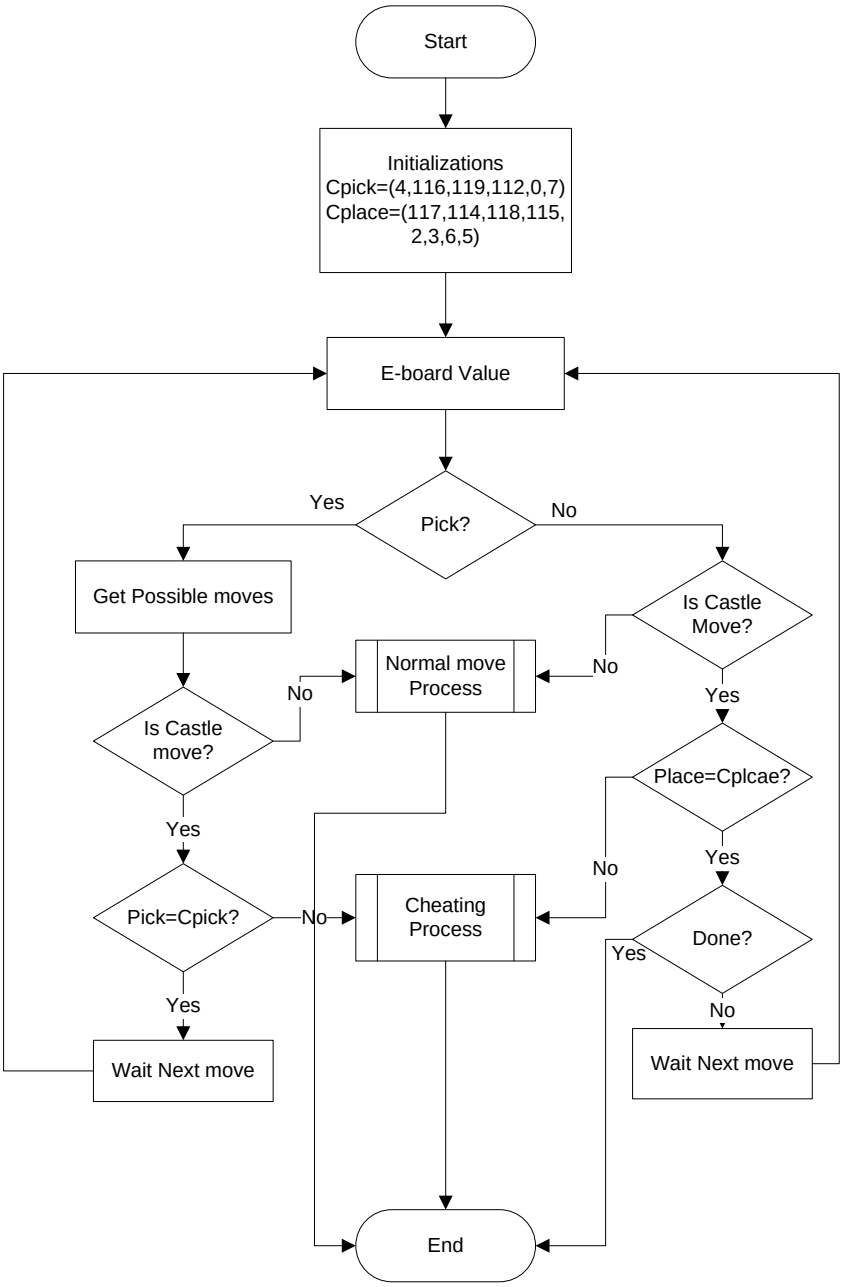


Figure (3.36): Castle Move Flow Chart

3.8.2.6 Promotion Move

When the player gets the promotion case, the following flowchart shows the process steps of how to handle the changes that occur on the E-board.

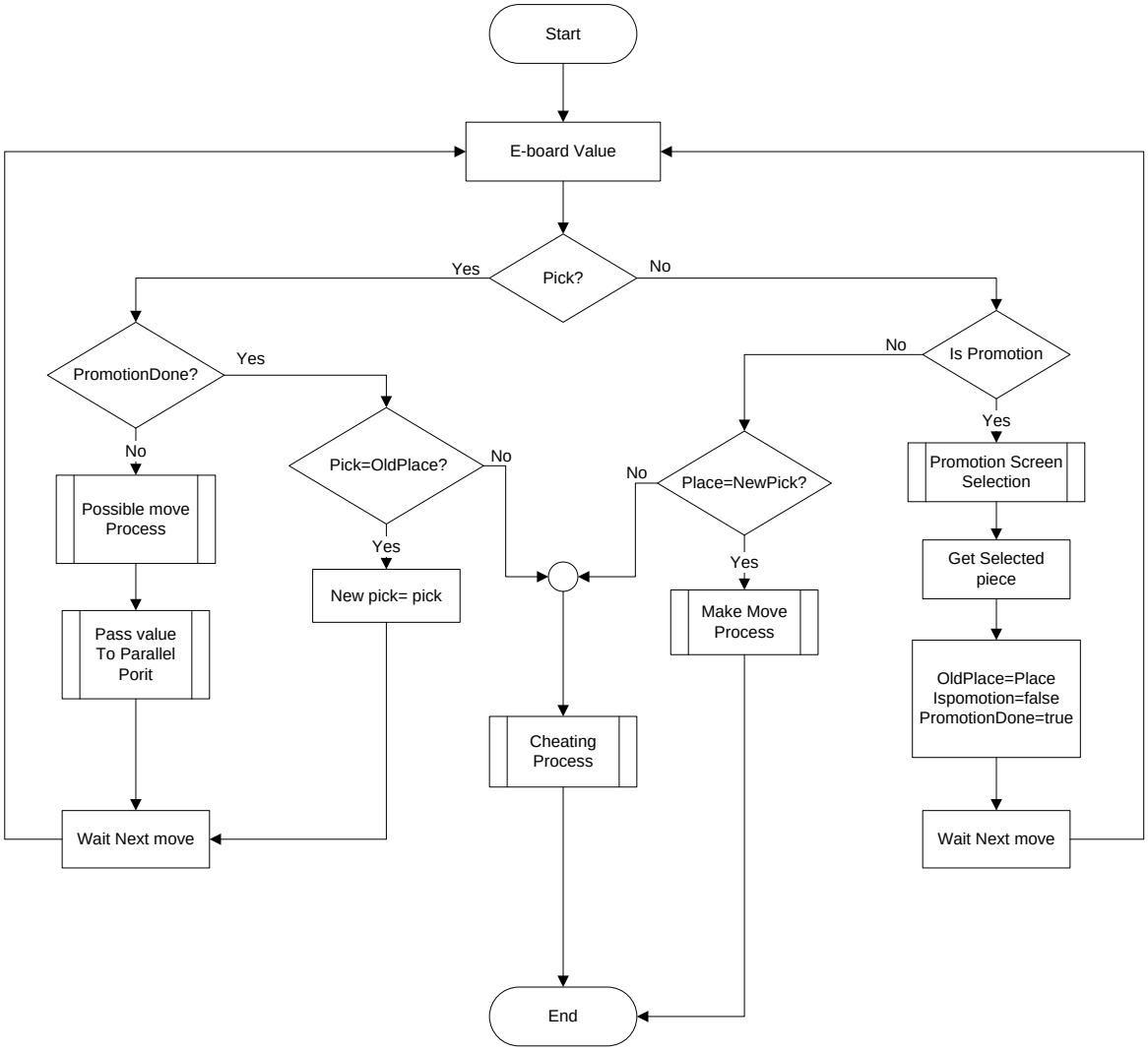


Figure (3.37): Handle Input of Promotion Move

3.8.2.7 Cheating Discovery

This flowchart shows how to determine cheating cases like picking up more than one piece or placing a piece that is out of the game.

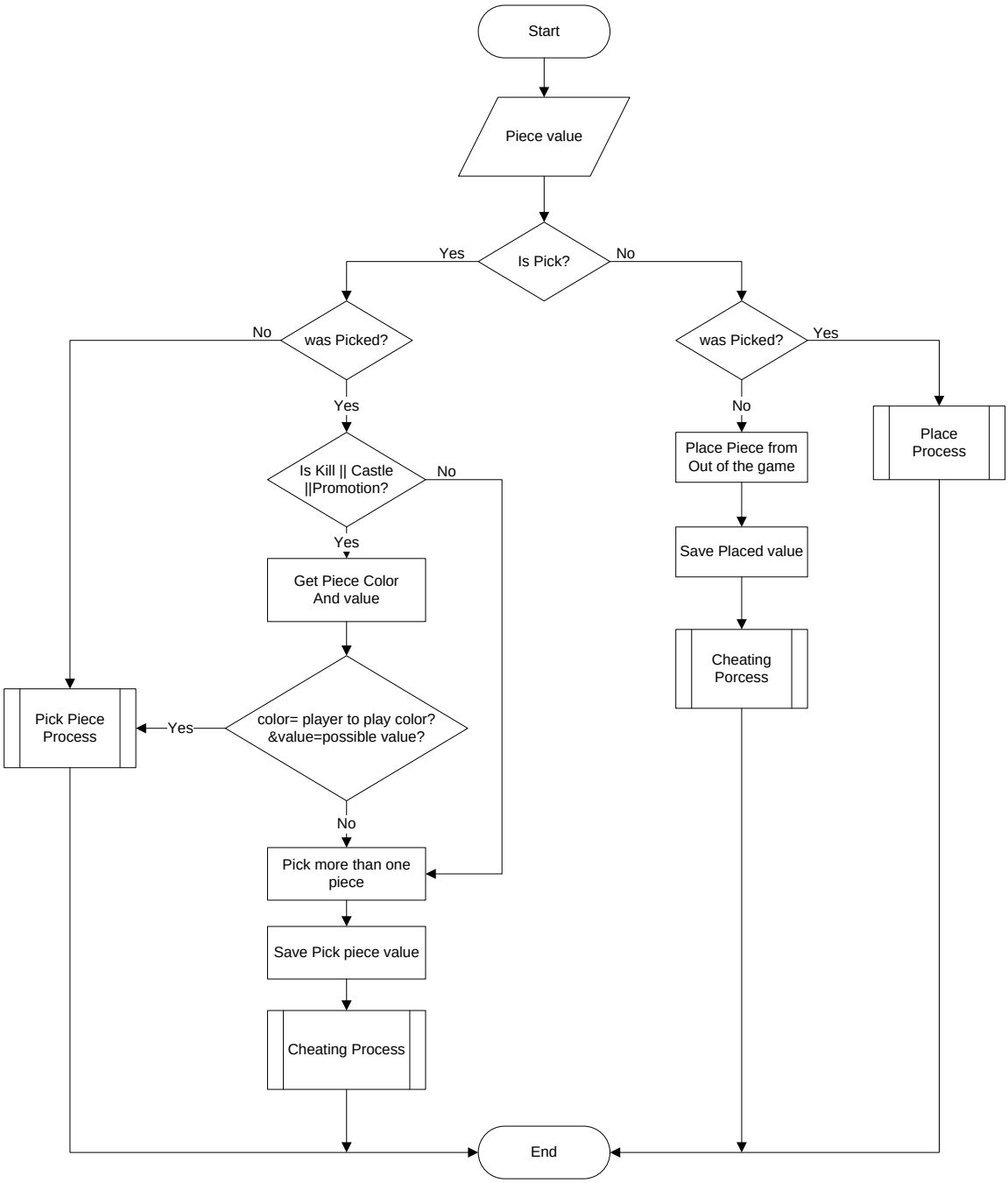


Figure (3.38): Cheating Discovery Flow Chart

3.8.2.8Cheating Solution

This flowchart shows the main process of how to solve the cheating case when it occurs while the player is playing the game.

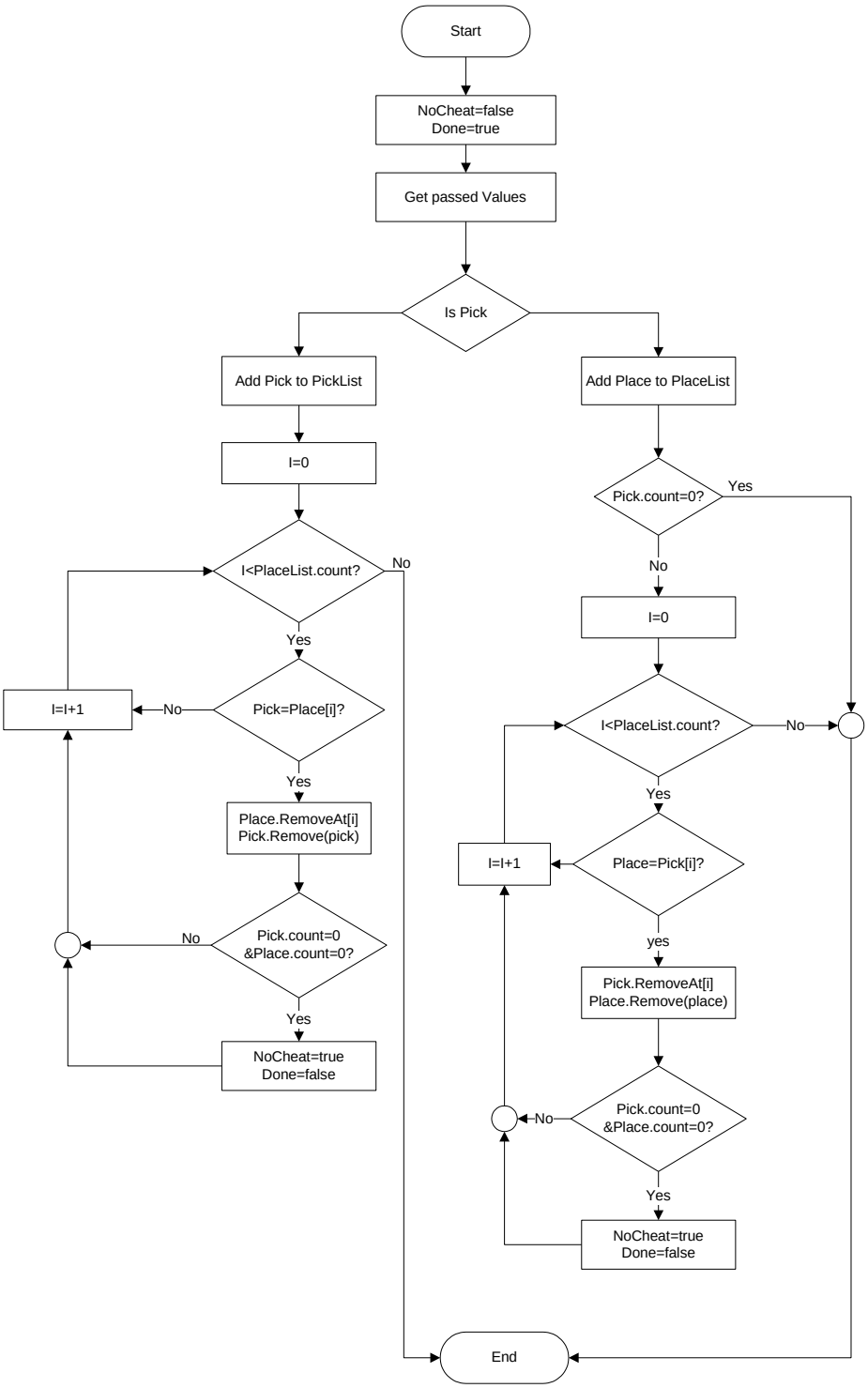


Figure (3.39): Cheating Solution Flow Chart

3.8.3 Sequence Diagram

The system interaction between the human player and Robot player, and how the process with one another and in what order, will be shown as the following subsections

3.8.3.1 Sequence Diagram of White Player Turn

If the player play with default game option which are played in level number one, and guiding the player by light the possible squares. In addition, if the player pick and place the piece as normal move.

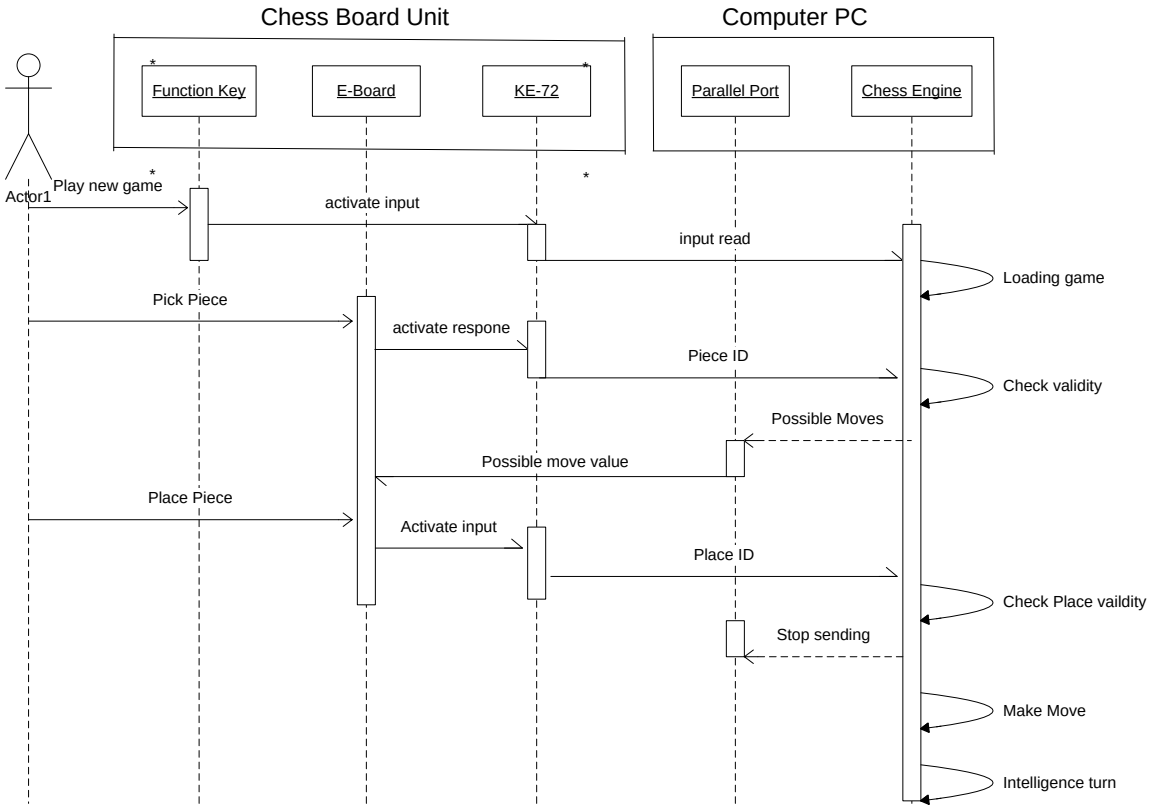


Figure (3.40): Sequence Diagram of Human Turn

3.8.3.2 Sequence Diagram of Robot Turn

Sequence diagram after the human player plays his turn with no errors, and plays the game with default option, the intelligence generate the contrast move to send it to Robot controller through local area network (Client/Server). Robot plays the normal move of pick and place.

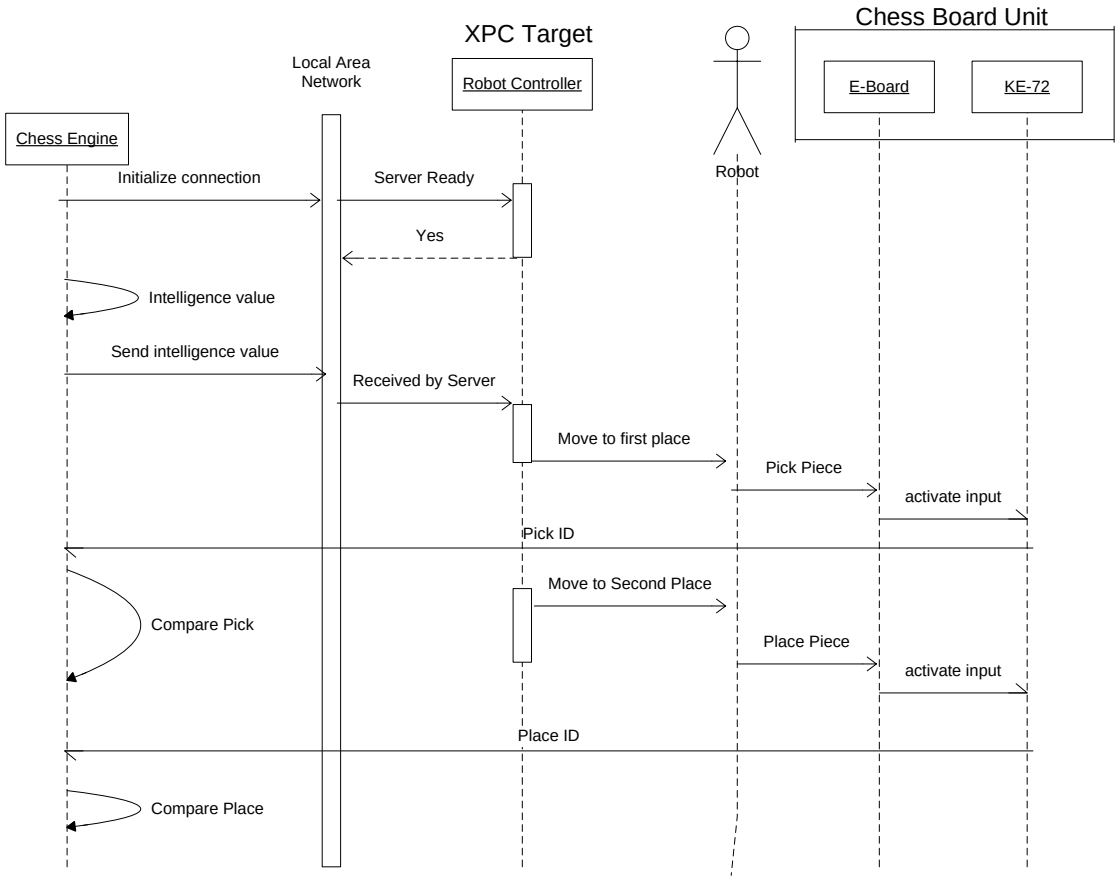


Figure (3.41): Sequence Diagram of Robot Turn

3.9 Summary

In this chapter, we exposed the system design in details, design the E-board components such as pieces, sensors, and LED's driver circuit. As well as the interfacing techniques, that is used.

In addition that it explores the system software design, system integration with E-board, system integration with Robot controller, the customization on chess engine software, and cheating solution.

The following table summarizes the changes in the chess engine to meet the project requirements.

pen Source (Sharp Chess Game)	**Customized Sharp chess Game**
Play the game in visual screen	Play the game on Physical Board
2D Chess board	3D chess board
Play against inelegance	Play against Robot according to passed value
	Playing with help mode by light the possible squares
	Good user interface that controlled by buttons on the E-board.
	Cheating discovery solution
	Messages notifications texts and voice

CHAPTER FOUR 4

Control System Design

4.1 Introduction

Chess Playing Robot is a mechatronics system that includes a lot of control theories starting with the classical ones until reaching the modern ones dealing with nonlinear systems.

The attention in such field of research comes from the high combination of many scientific issues, programming, interfacing techniques between the hardware and software systems, controlling algorithms used to control the robot and the sensors variance and connection. This project also has such an importance because of the challenges embedded in it.

The challenges in this research grow in parallel with the achieved steps in it, from the control point of view, one may think in many problems and difficulties in it such as:

- The nonlinearity in the dynamics of the mechanical system, the robot contains actuators and rotational motion in its links; this will result in differential equations that contain nonlinear terms in all of the Robot's range of motion.

- Sequential motion, there should be a sequence that the Robot with its links and parts should follow. The controller for such a problem should consider the fact that the Robot will not have the same motion sequence all the time, this depends on the chess engine and its decision for the next motion.

- Disturbances with its variety in kinds, either those which come as signals from the power sources or even those which caused by the human, like hitting the Robot or moving it, this will generate signals from external sensors that are inputted to the computer in order to do the right actions.

- Suitable actuators that can be used to operate the system need to be controlled also, this means a further modeling, programming and controlling.

- Fixing limit switches on the Robot's gripper in order to generate signals used in defining the exact position of the Robot at any instant.

4.2 Control Strategy

This section talks about the strategy and procedures followed to design the Robot's controller. The way of thinking growth starting with a simple controller for each joint depending on the Independent Joint Control method, then it is advanced to design a Robust-tracking controller for the system.

An important issue that should be taken in consideration is the fact that besides the needed controller for the mechanical structure, we need another controller that is responsible to dominate the logic of the motion, in other words, a software should be programmed to control the sequence in motion regarding the actuators, that is because in the chess game one may execute one of the following four motion:

- Move the stone from a position to another empty one.
- Make a movement that includes killing.
- Castling, which is the process of interchanging the positions of the Rock and the King.
- Promotion, it is the ability to alter the Soldier with another stone (Queen, Knight, Rock or Bishop) when reaching the first row of the other player.

MATLAB/Simulink is the software that combines these two controllers, the first one is done using the ordinary Simulink blocks but the second one is done using State Flow toolbox.

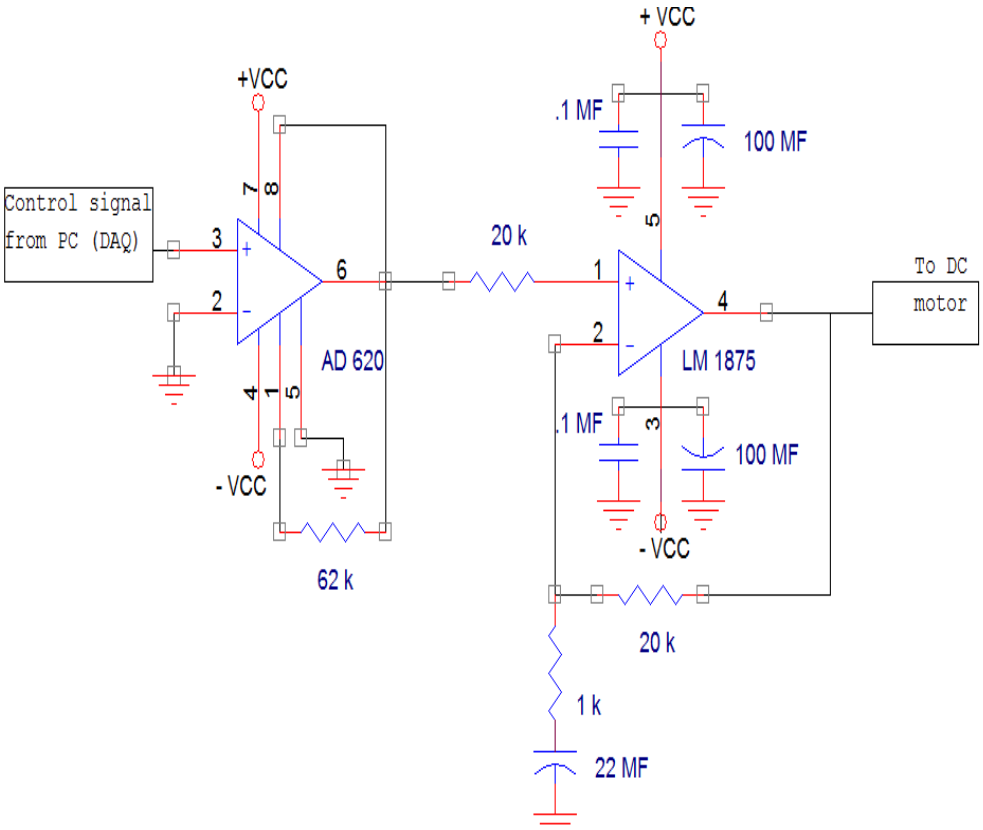
The upcoming sections show the exact steps done in details to design a controller for the system; they also include the electrical and interfacing circuits that is used to connect the Robot with the PC. At the end of this chapter, theoretical and experimental results in addition to the conclusions are discussed.

4.3Driving and Interfacing Circuits

Almost for each electrical part in the Robot a driving circuit should be built, this circuit is either used to amplify the current and the voltage or to switch the direction of rotation of the DC motors or both. The properties of the DC motors and the controlling signals from the DAQ card are the two things that should be taken in consideration to design such circuits, these details and others are discussed in the following sub-sections.

4.3.1 Power Amplifier Circuit

Data acquisition cards DAQ can provide a 10V output maximum with a maximum current of ±5mA, the used DC motors in the Robot operates at 18V with a rated current of 0.5A, so driving circuits are necessary to connect the DC motors with the DAQ card. These circuits are shown starting with Figure (4.1).



Figure(4.1): interfacing circuit between DAQ and DC motors.

The circuit of Figure (4.1) contains two major elements:

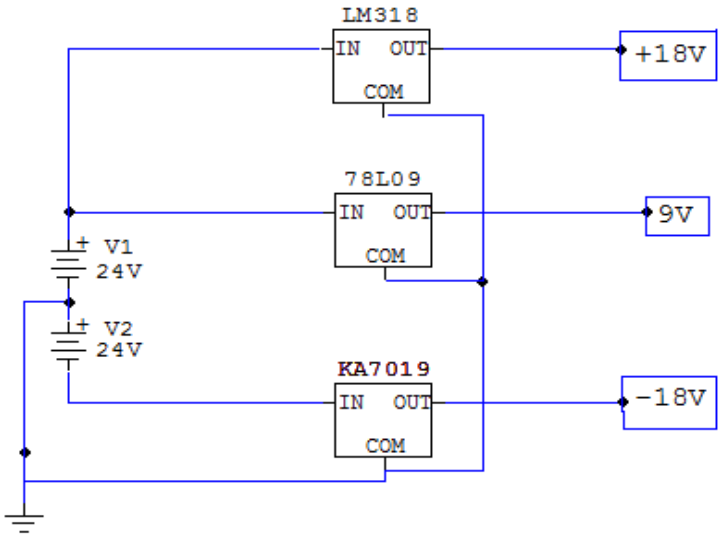
1- AD620 operational amplifier: this is used as a preamplifier; it receives the signal from the DAQ card and amplifies its output voltage from 10V to 18V but without current amplification.

2- LM1875 power amplifier: it receives the 18V signal from AD620 operational amplifier; then it amplifies the current in the circuit in order to put the motor into operation.

Others parts in the circuit are calculated and used to build the whole circuit that can amplify both the voltage and the current depending on the data sheets of the previous two parts[22].

4.3.2 Power Supply Circuit

The power supply circuit must be modified to generate the needed voltage to supply the DC motors and other circuits with the needed power. For example the first two DC motors should be supplied with ±18V, besides the other two DC motors in the Gripper mechanism operates at 10V DC, other electrical circuits such as the ICs need much lower DC voltage.The power supply circuit is shown in Figure (4.2).



Figure(4.2): Power supply circuit.

There are two power supplies in this circuit, each one can provide 24V DC, and they are connected in series in order to take + 24V and -24V voltage.
The other elements in the circuit are:

1- Two +18V regulators connected in parallel to take higher current up to 2A.
2- Two -18V regulators connected in parallel to take higher current up to 2A.
3- 9V regulator and 10v regulator. That is used for the potentiometers and in the Gripper mechanism.

4.3.3 H-bridge Circuit

There are two DC motors in the Gripper mechanism, these are controlled by the DAQ card through the digital I/O to drive the motors clockwise and counterclockwise in order to move the gripper up and down or to open and close it, the following circuit shownin Figure(4.3) shows how the signal outputted from the DAQ card was used to generate the bidirectional rotation[20].

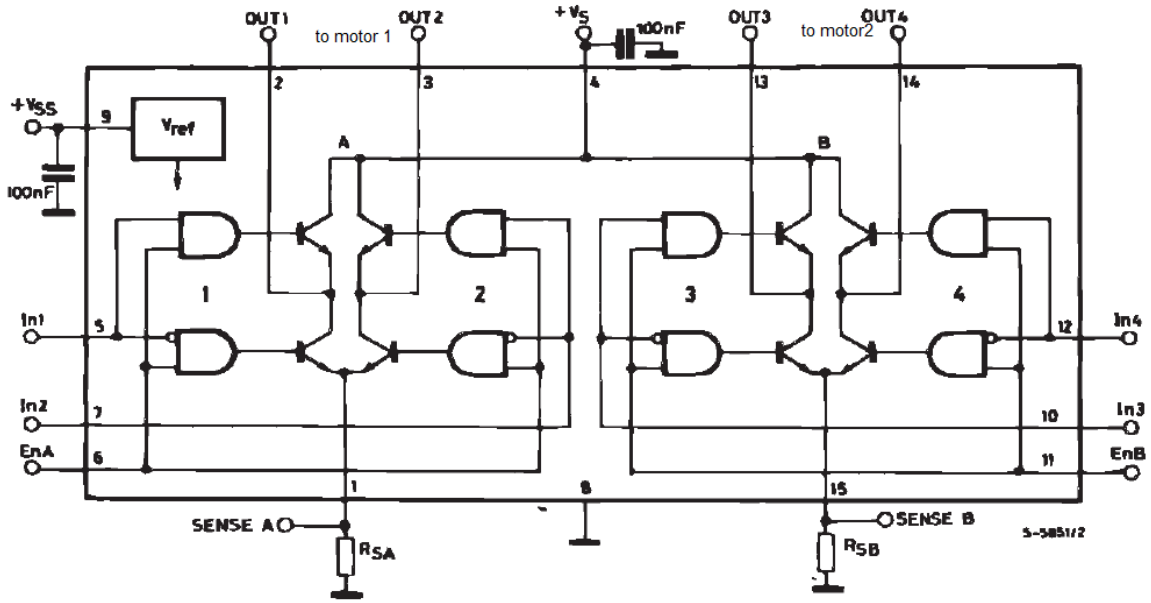


Figure (4.3): H-bridge Circuit

Where OUT1 and OUT2 are connected to the first motor and OUT3 and OUT4 are connected to the second motor. The following table shows the parameters of the previous circuit:

Table (4.1):H-bridge circuit parameters

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

This circuit needs additional components other than the H-Bridge itself. To achieve the bidirectional rotation for each motor, four diodes and two capacitors must be used, the detailed circuit is shown in the following figure.

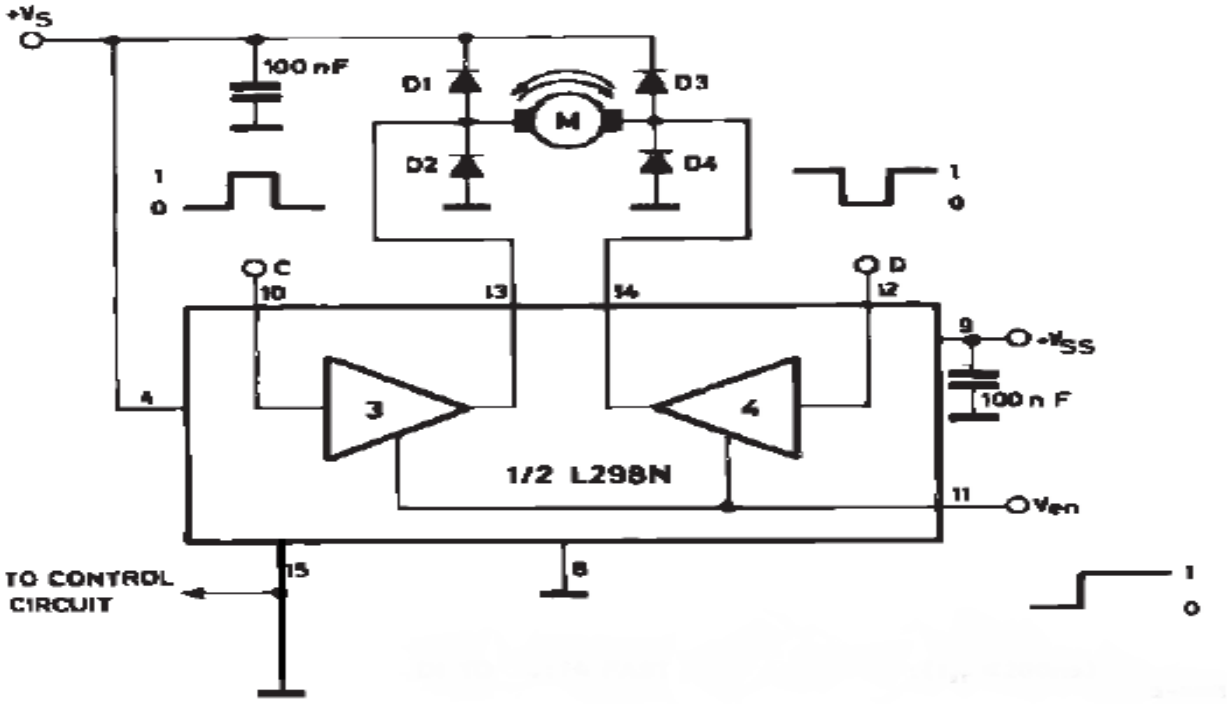


Figure (4.4): Bidirectional DC Motor Control

Where C and D are the digital inputs and they define the motor's rotation according to the following table.

Table (4.2): Bidirectional DC Motor Control Logic

Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

4.4 Potentiometer Calibration

As mentioned before, SCARA robot structure used in this project has three degrees of freedom, the first two links move using two motors, one for each link. Two potentiometers are used, one for each motor to get a voltage feedback to close the loop of the motor's controller, that is to know the position of each link of the robot at any desired instant.

In order to get a relationship between the output voltage of the potentiometer and the angle of rotation of the motor, a conversion factor (degree to voltage) must be calculated.

4.4.1 First Potentiometer

It is a 5 kΩ variable resistance, one turn potentiometer, it is fixed to be coaxial to the first motor's shaft to get a 1:1 rotation ratio between the two shafts of the motor and the potentiometer.

- First attempt:

The AutoCAD program is used to draw lines that intersect in the same point with an angle of 30 degree between each line and the previous one that is to create a huge protractor on an A0 paper.

After this a small laser pointer was fixed on the lateral surface of the first link, and then the link was manually moved to generate readings of the potentiometer for every 30 degrees of motion. After taking the readings, MATLAB was used to plot the relation between these readings to create the suitable line that connects these points and as a result to find the equation that combines these points.

The disadvantage of this method is that it results in inaccurate equations that may not combine all the given points in one line, that was because the way of fixing the laser beam on the link, so it was necessary to search for another way that is more accurate.

- Second attempt:

This experiment was held by connecting a load with a rope freely with the link under the effect of the gravitational force, then the link was moved manually as before 30 degrees at each time in order to record the readings of the potentiometer. Then using MATLAB, these readings (angle vs. voltage) are programmed as shown in Figure (4.5) and.

```
Command Window
>> x=[0 30 60 90 120 150 180]; %Degree
>> y=[1.636 2.832 4.063 5.22 6.24 7.212 8.101]; %Volt
>> cftool
```

Figure (4.5): Potentiometer's voltage corresponding to the angles (degrees).

Using the command shown in Figure(4.5) which is the "cftool" command, the corresponding points of the readings obtained as:

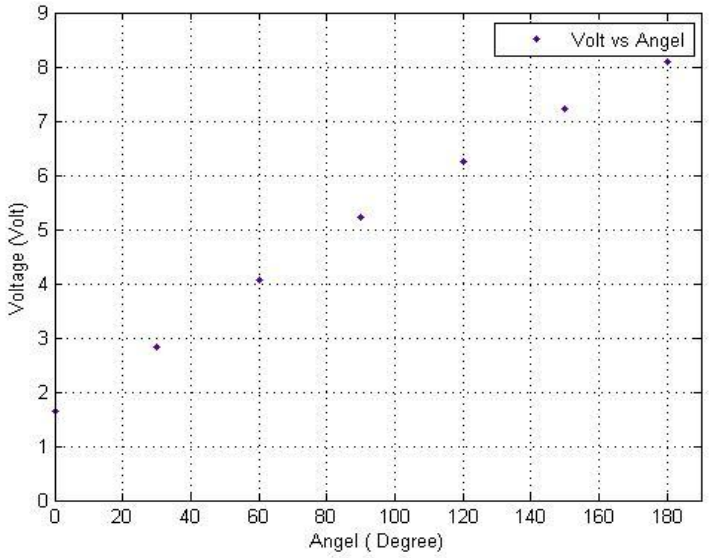


Figure (4.6): Voltage vs. Degrees of the first potentiometer

Next step is to choose the linear polynomial that describes all these points in one equation. This is done also using MATLAB and "cftool" command. This command offers many options for this chosen polynomial, linear, quadratic cubic and others as shown in Figure (4.7).

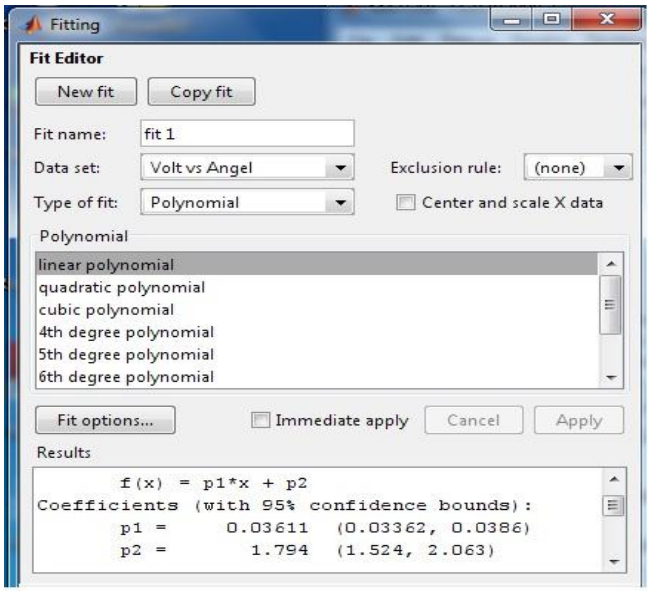


Figure (4.7): Selecting the linear polynomial approximation.

At first, the linear polynomial was chosen as the one that will combines all these points with one line, from the previous figure the equation of this line is obtained and considered as the conversion factor, this equation is:

$$f(x) = 0.03611x + 1.794 \ldots (4.1)$$

Plotting these lines in addition to the linear polynomial in one plot is shown in Figure (4.8). It is obvious that there is not a 100% match between the points and the linear polynomial, so it was necessary to try another polynomial other than the linear one.

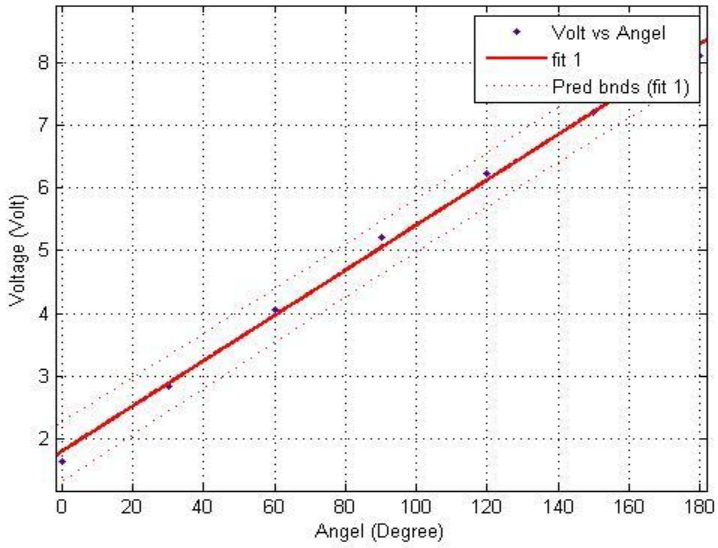


Figure (4.8): The linear polynomial approximation between the points.

Next step was to choose the quadratic polynomial and test it if satisfying all the given points, as shown in the following figure this was done and the desired equation was also obtained.

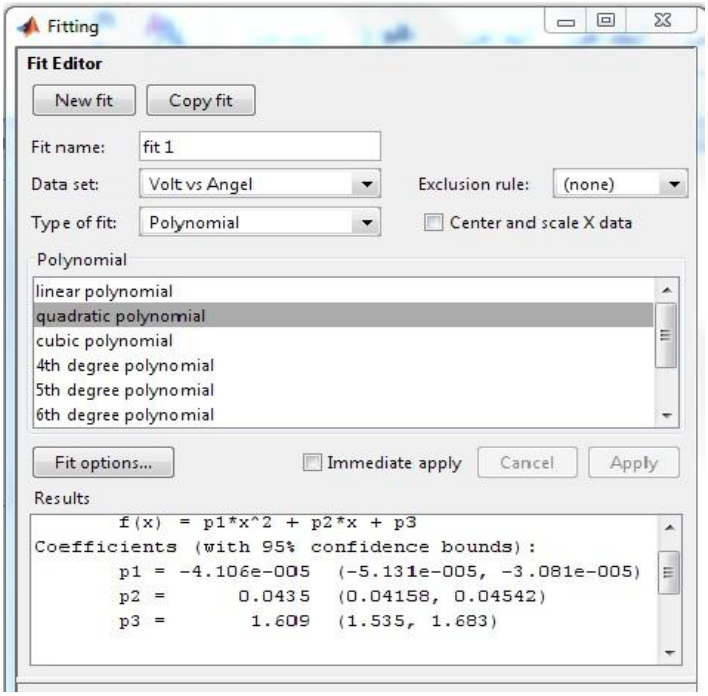


Figure (4.9): Selecting the quadratic polynomial approximation.

The resulted quadratic polynomial function is:

$$f(x) = -4.1 * 10^{-5}x^2 + 0.04435x + 1.609..(4.2)$$

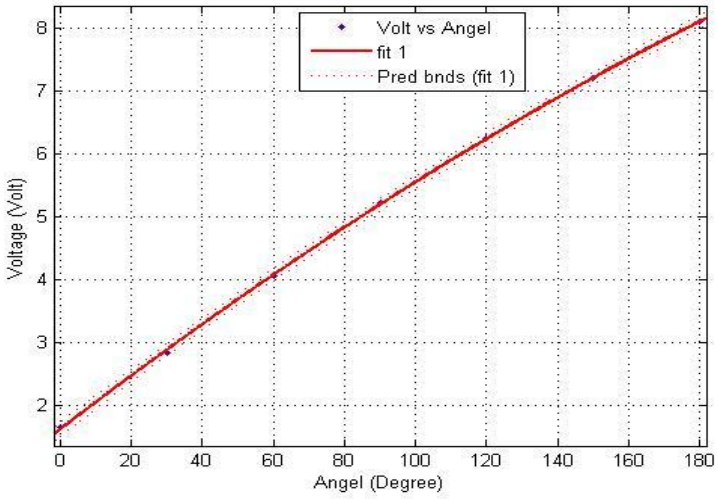


Figure (4.10): the quadratic approximation between the points.

4.4.2 Second potentiometer

It is a 5 kΩ variable resistance, one turn frictionless potentiometer, it is fixed to be coaxial to the second motor shaft.

As what was done for the first potentiometer a protractor on an A2 sheet was drown. The point of intersection is fixed at the center of the second motor in order to take accurate readings between the voltage and the angles of the second potentiometer. The readings were written and plotted using the "cftool" command in MATLAB as shown below:

```
Command Window
>> x=[0 30 60 90 120 150]; %Degree
>> y=[7.9 7.1 6.27 5.43 4.57 3.76]; %Volt
>> cftool
```

Figure (4.11): Potentiometer's voltage corresponding to the angles (degrees).

By using "cftool" command the resulted point corresponding to the reading are shown in Figure (4.12)that follows.

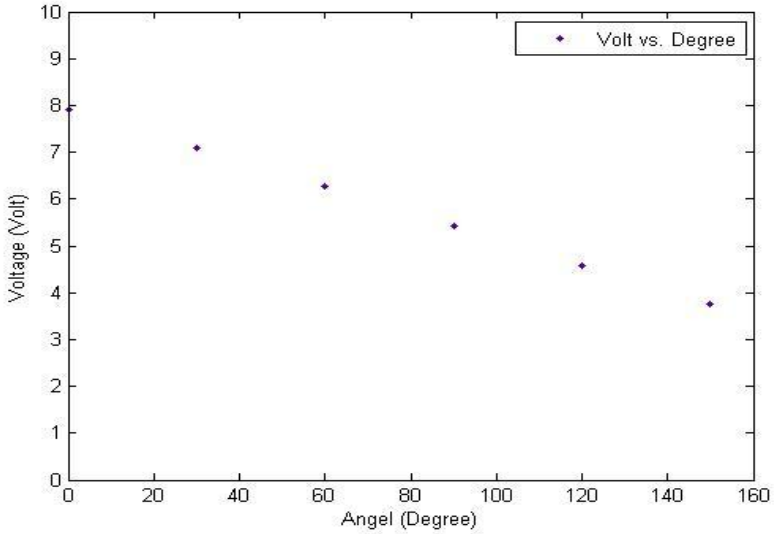


Figure (4.12): The corresponding points for the second potentiometer readings.

Next step was to choose the linear polynomial to sketch all the line that combines all the points as in Figure (4.13).

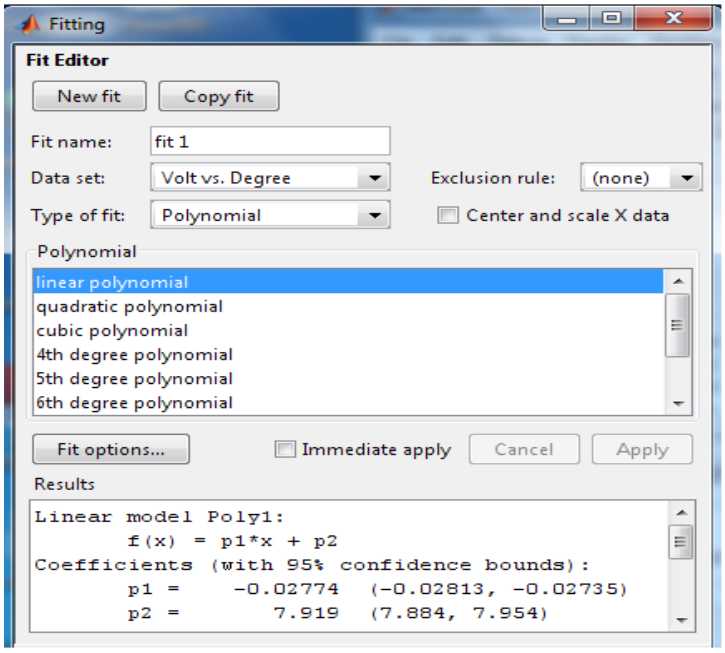


Figure (4.13):selecting the linear polynomial approximation.

And the resulted linear equation is:

$$f(x) = 0.02774 + 7.919 \ldots (4.3)$$

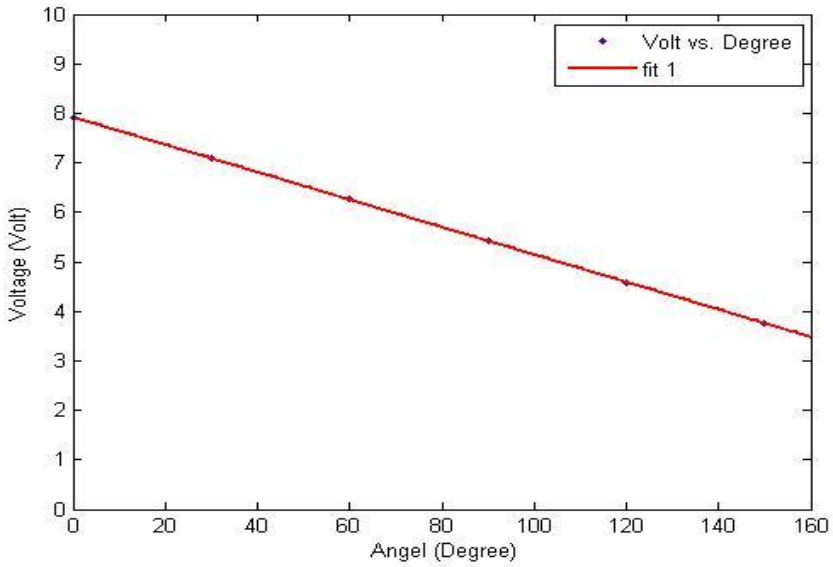


Figure (4.14):the linear line approximation between the points.

Finally, from the calibration process we get an equation for each potentiometer.

$$f(x) = -4.1 * 10^{-5}x^2 + 0.04435x + 1.609..(4.4)$$
$$f(x) = 0.02774x + 7.919.(4.5)$$

These equations are then used as the conversion factors that will convert the input angles in degrees which are calculated from the inverse kinematics analysis, to the corresponding voltages that will be inputted to the motors which will allow the robot's arms to achieve the exact motion.

4.5Controller Design

SCARA Robot in this case has four states which are an angle and the velocity for each link, two out of these four states must be regulated to zero which are the two velocities, and the other two states must go to a specific reference calculated by the inverse kinematics program, this represents the final position for each link, thus a good choice for a controller design will be the a Robust tracking one that in addition will give a zero error.

4.5.1Robust Tracking Controller Design

This section talks about designing a Robust Tracking for the system to track the desired trajectory inputted to each one of the DC motors, it is necessary for this controller to achieve the desired response and also to improve the steady state error in the system, otherwise, the Robot will go to wrong locations.

Recalling the dynamic equations previously mentioned in Chapter Three with emitting nonlinear terms:

$$T_1 = m_2 l_2^2\left(\ddot{\theta_1} + \ddot{\theta_2}\right) + m_2 l_1 l_2\left(2\ddot{\theta_1} + \ddot{\theta_2}\right) + (m_1 + m_2) l_1^2 \ddot{\theta_1} \quad (4.6)$$
$$T_2 = m_2 l_1 l_2 \ddot{\theta_1} + m_2 l_2^2(\ddot{\theta_1} + \ddot{\theta_2}) \quad (4.7)$$

Finding the State Space representation for the system with the assumption of:
$x_1 = \theta_1$
$x_2 = \dot{\theta}_1$
$x_3 = \theta_2$
$x_4 = \dot{\theta}_2$

Then $\dot{x}_1 = \dot{\theta}_1 = x_2$ (4.8)

$\dot{x}_2 = \ddot{\theta}_1$

$\dot{x}_3 = \dot{\theta}_2 = x_4$ (4.9)

$\dot{x}_4 = \ddot{\theta}_2$

Now substituting these equations in equation (4.6) and (4.7):

$T_1 = m_2 l_2^2(\dot{x}_2 + \dot{x}_4) + m_2 l_1 l_2(2\dot{x}_2 + \dot{x}_4) + (m_1 + m_2) l_1^2 \dot{x}_2$

$T_1 = (m_2 l_2^2 + 2m_2 l_1 l_2 + m_1 l_1^2 + m_2 l_1^2)\dot{x}_2 + (m_2 l_2^2 + m_2 l_1 l_2)\dot{x}_4$ (4.10)

$T_2 = m_2 l_1 l_2 \dot{x}_2 + m_2 l_2^2(\dot{x}_2 + \dot{x}_4)$

$T_2 = (m_2 l_1 l_2 + m_2 l_2^2)\dot{x}_2 + m_2 l_2^2 \dot{x}_4$ (4.11)

Solving equation(4.11) for$\dot{x}_4$

$$\dot{x}_4 = \frac{T_2 - (m_2 l_1 l_2 + m_2 l_2^2)\dot{x}_2}{m_2 l_2^2} \quad (4.12)$$

Substituting this equation in equation (4.10) will give:

$$\dot{x}_2 = \frac{T_1 - \frac{bT_2}{m_2 l_2^2}}{a - \frac{b^2}{m_2 l_2^2}} \quad (4.13)$$

Now substituting in equation (4.12):

$$\dot{x}_4 = \frac{\left(1 + \frac{b^2}{m_2 l_2^2 e}\right) T_2 - \frac{d}{e} T_1}{m_2 l_2^2} \quad (4.14)$$

Where:

$a = m_2 l_2^2 + 2m_2 l_1 l_2 + m_1 l_1^2 + m_2 l_1^2$

$b = m_2 l_2^2 + m_2 l_1 l_2$

$c = m_2 l_1 l_2$

$e = a - \frac{b^2}{m_2 l_2^2}$

Now from these four equations one can construct the matrices A and B, which are:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ \frac{1}{e} & \frac{-b}{m_2 l_2^2 e} \\ 0 & 0 \\ \frac{-d}{m_2 l_2^2} & \frac{1}{m_2 l_2^2} + \frac{b^2}{m_2^2 l_2^4 e} \end{bmatrix}$$

This system has two inputs, one for each motor with four states, so one should expect a 2*4 dimensions for the feedback matrix k, so:

$$\vec{u} = -[k]\vec{x} \tag{4.15}$$

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = -\begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{4.16}$$

x_1 and x_3 are to track a specific reference since they stand for the angle of rotation of the first and second links respectively, so they should stop at given angles that represent their final position, on the other hand, $x_2$and x_4 stand for the velocities of these links that should be regulated to zero when reaching this final position, and:

$$e_1 = x_1 - r_1 \tag{4.17}$$

$$e_2 = x_3 - r_3 \tag{4.18}$$

Where:
$e_1$is the error related to the state x_1
$e_2$is the error related to the state x_2
$r_1$is the reference value inputted to the first motor
$r_2$is the reference value inputted to the second motor

Then:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = -[k]\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} - \begin{bmatrix} r_1 \\ 0 \\ r_3 \\ 0 \end{bmatrix}\right) \tag{4.19}$$

Simplifying will give

$$\vec{u} = -[k]\vec{x} + [k]\begin{bmatrix} r_1 \\ 0 \\ r_3 \\ 0 \end{bmatrix} = -[k]\vec{x} + \begin{bmatrix} k_{11} & k_{13} \\ k_{21} & k_{23} \end{bmatrix} \tag{4.20}$$

So the input matrix that will regulates state 2 and 4, and in the same time will eliminate the steady state error in state 1 and three is:

$$\vec{u} = -[k]\vec{x} + [k_s]\vec{r} \quad (4.21)$$

$k = lqr(A, B, Q, R)$= 2.4869 1.1128 -1.9533 -0.7373
-1.9533 -0.7373 -2.4869 -0.7646 (4.22)

And k_s = [k(1,1) k(1,3) ; k(2,1) k(2,3)] = 2.4869 -1.9533
-1.9533 -2.4869 (4.23)

With Weight matrices of

Q=[10 0 0 0;0 1 0 0;0 0 10 0 ; 0 0 0 1] (4.24)

And

R=eye(2) (4.25)

The analysis and results of applying this controller are discussed later in section 4.7.

4.6 Simulink Model

4.6.1 Stateflow

The State flow product is an interactive graphical design tool that works with Simulink software to model and simulate event-driven systems, also called reactive systems. Event-driven systems transition from one operating mode to another in response to events and conditions. These systems are often used to model logic for dynamically controlling physical device such as a fan, motor, or pump. [15]

The reason beyond the usage of the State flow product came from the existence of the four kinds of motion previously mentioned in section 4.2. This Robot is similar to any other machine in having a sequence of motion that should be followed each time, the new thing here is that the Robot should perform a different motion each time depending on the kind of motion (order) received from the Chess Engine, an example of that is the "Move without Kill" motion, in this kind of motion the Robot should go the first address, opening the gripper, going down, closing the gripper and catching the stone, going up, going to the second address to put the stone in it and then going to the home position. This actions sequence obviously differs from one motion to another.
Next chart shows this sequence for the first motion"Move without Kill".

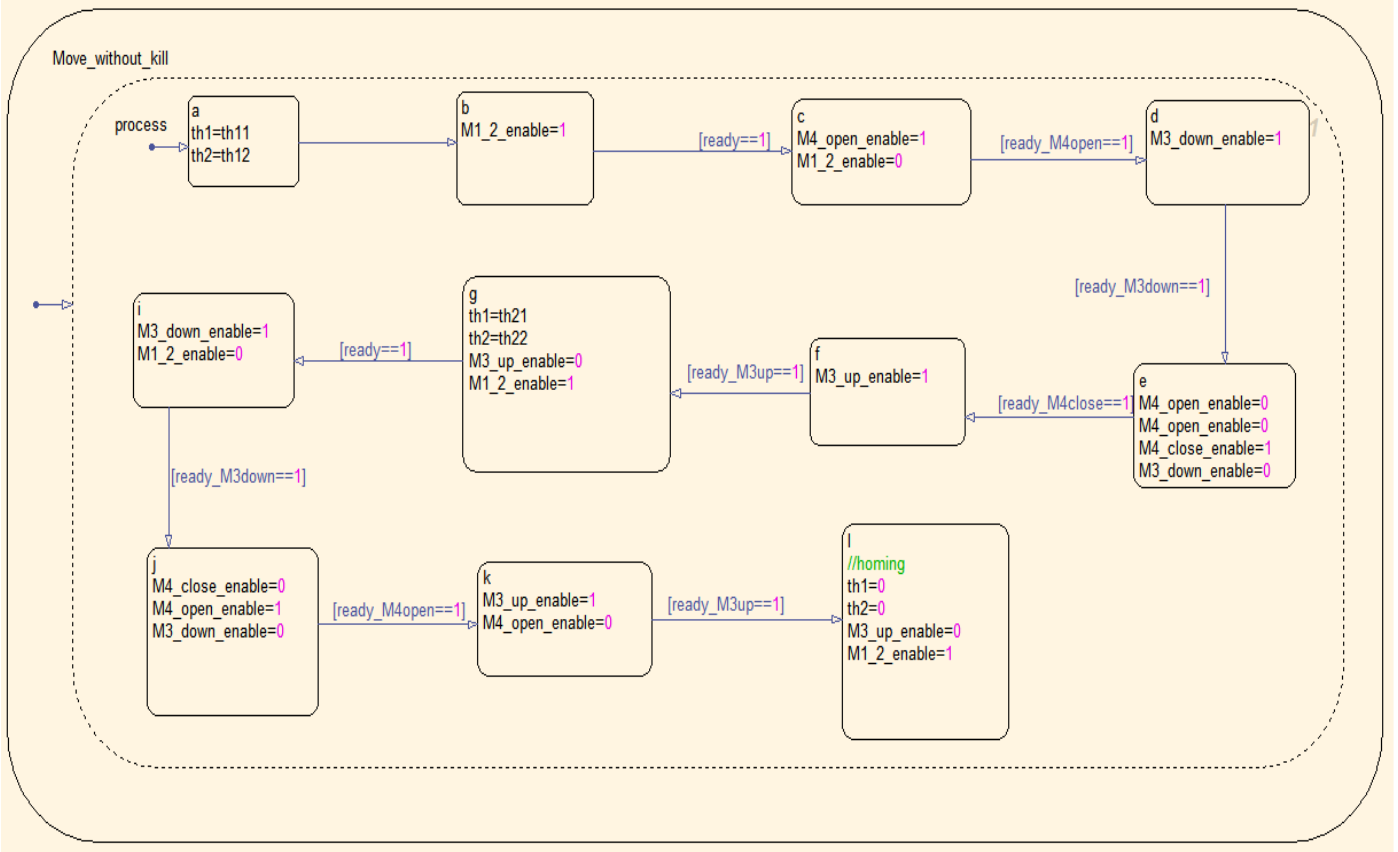


Figure (4.15):"Move without Kill" State flow

The State flow used for the whole system is embedded in one block (see Figure (4.16)). This block receives the desired angles, motion type, stone kind and the ready signals, these signals are inputted from the physical system using the DAQ card to allow the sequence in actions, for example, if the signal "ready" is enabled (becomes one) means that the desired location was reached, then it is the gripper's turn to open and to go down.

The State flow block also has many outputs, starting with th1 and th2 which are the two angles transferred to the two motors respectively indicating a specific address, and ending up with the "enable" signals, these signal are the ones responsible to activate the actuators sequentially using the DAQ card.

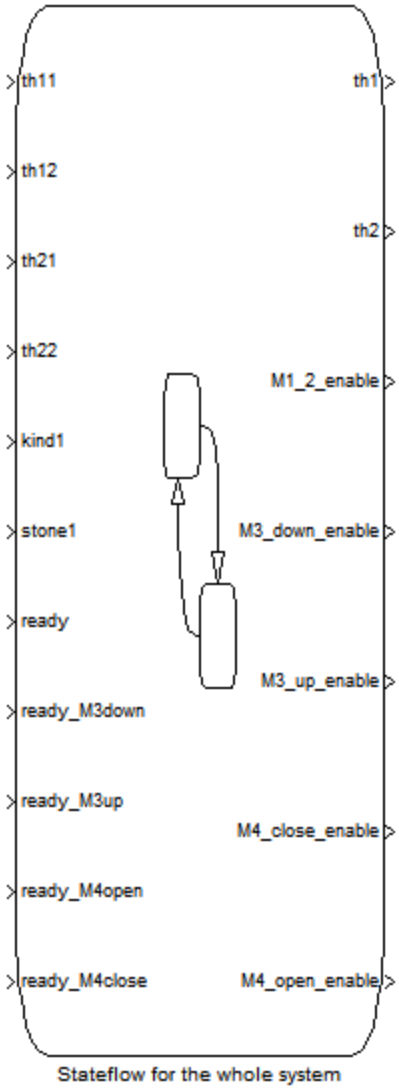


Figure (4.16): State flow block including the four types of motion

4.6.2 The Whole Simulink Model

MATLAB/Simulink was the software used in this project; it is responsible for controlling the system including all its aspects like the compensators, the motion sequence, the timers and the sensors readings. Subsystems and Imbedded Functions were used to program such a model as showed in the following figure.

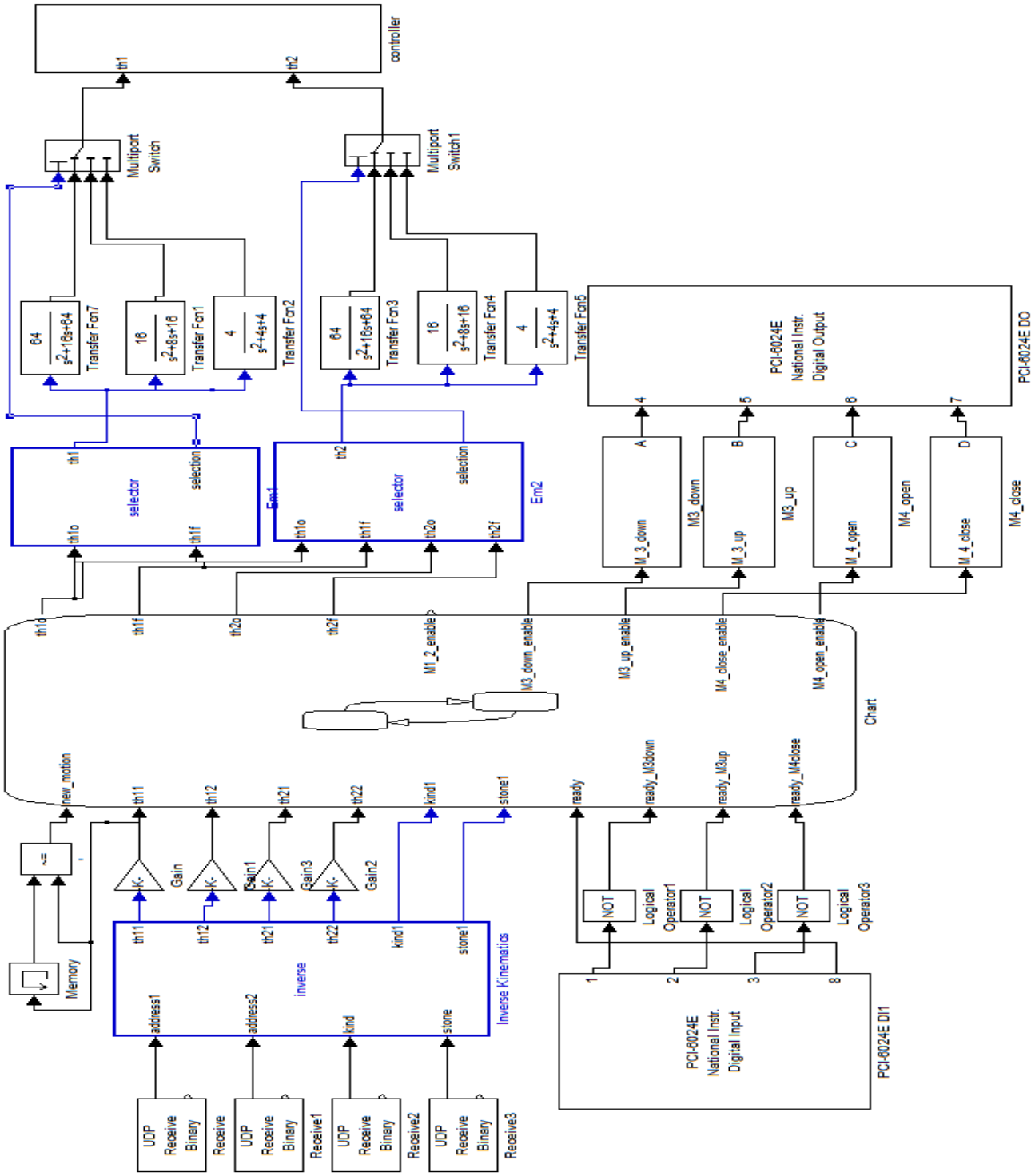


Figure (4.17): The Whole Simulink Model

Looking at the previous figure and going from left to right, one may notice the following procedures:

- Receiving the data and addresses from the chess engine
- Calculating the desired angles using inverse kinematics analysis
- Controlling the logic and motion sequence through the State Flow
- Converting the inputted signals into cubic polynomial trajectory using the transfer functions. These transfer functions are also used to control the velocity of the motion, depending on the distance to be crossed by each link that is divided into three categories: less than 10cm, between 10cm and 20cm, more than 20cm.
- Inputting the trajectories to the controller of the system.

4.7 Results and Conclusions

4.7.1 Theoretical results

First step in the controller design was designing a State Feedback controller to improve the system's response and specially the steady state error. Applying the controller obtained in section4.5.1 to the nonlinear theoretical representation of the system as showed in Figure (4.18):

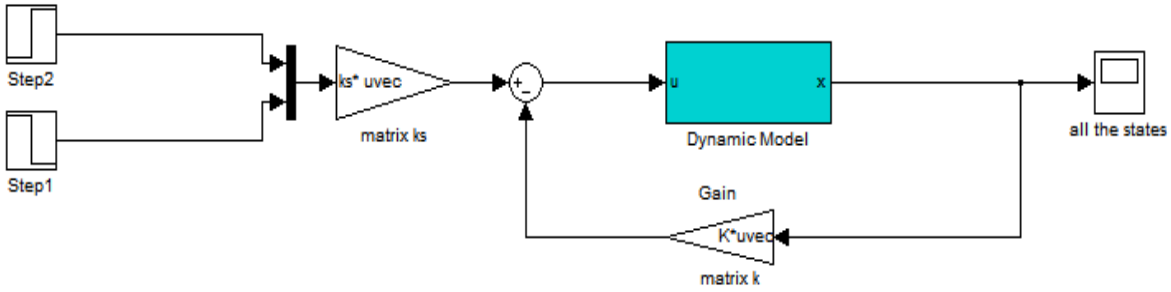


Figure (4.18): State Feedback Controller applied for the theoretical model

This model will output on its scope the response of the four states as follows:

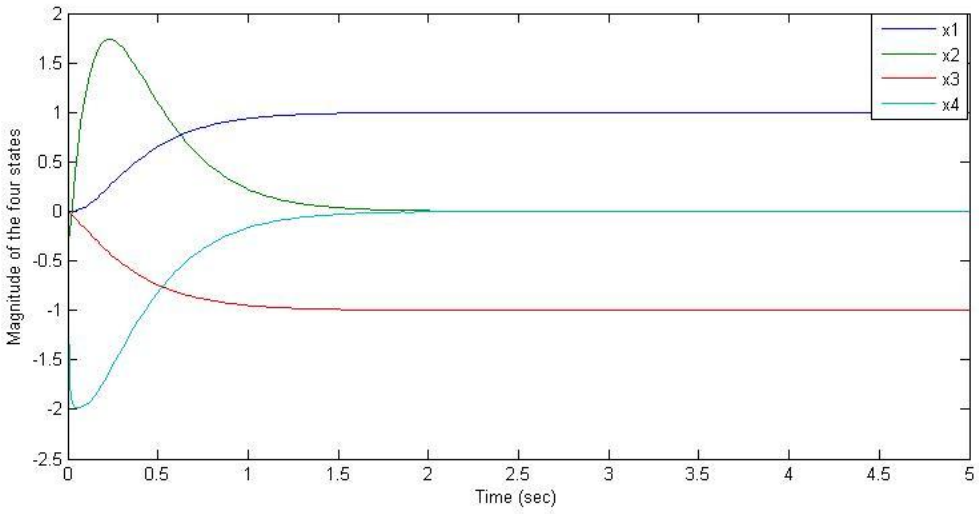


Figure (4.19): Four states response using state feedback

It is clear that states x1 and x3, which are the angle of rotation of the first and the second motor respectively, reached the final value as desired with zero steady state error, also the other two states which are the velocities are regulated and reached zero as a final value.

The weight matrix Q can be tuned to get the most suitable response for each state, that is to say, changing the magnitude of the diagonal of the Q matrix will change directly the response for each state, its settling time, peak time and speed.

4.7.2 Experimental results

Applying the designed Robust tracking controller on the physical system will not give results that perfectly match what were obtained in the simulation analysis, this is due to the immeasurable parameters in the robot itself that were not embedded in the mathematical model like the viscous and Coulomb friction, on the other hand the exist of the noise signals is a critical problem, this appears as a continuous vibration in the robot even when it is not in motion progress.

All this force the designer to use additional filters and further tuning in the controller to improve the system's response. For example and regarding the safety sensors, when any of the sensors detect an obstacle in the robot's path the robot should "freeze" in its place, that is done by switching the input to the controller to the current position, then error should be zero and the controller will stop the motors, but what was obtained when an obstacle is detected is that the robot stopped but with continuous soft vibration, this is due to the noise signals.

The following two figures show the response for each link while going from the home position to the address D4. It is clear in the actual signal the effect of the noise on it. The first link has an output of a cubic polynomial that follows the reference signal inputted to it from the second order transfer function, the second link gets its reference signal from a first order filter. As mentioned earlier, both links don't follow the exact input signal due to the noise and immeasurable parameters. On the other hand the error in each link is almost zero and the link reached its final position.

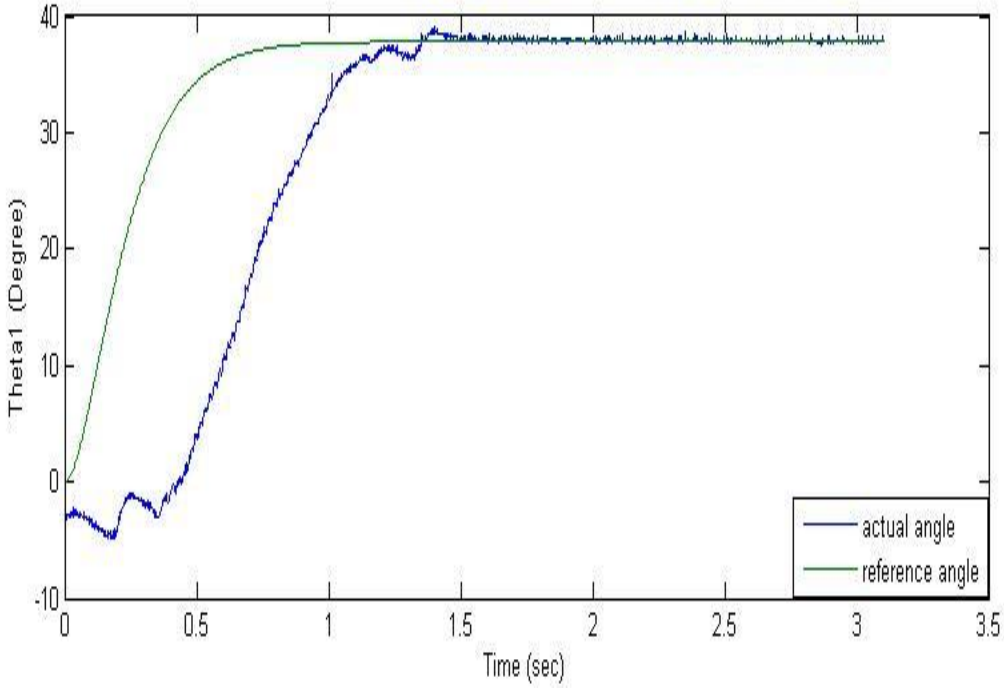


Figure (4.20): First link response

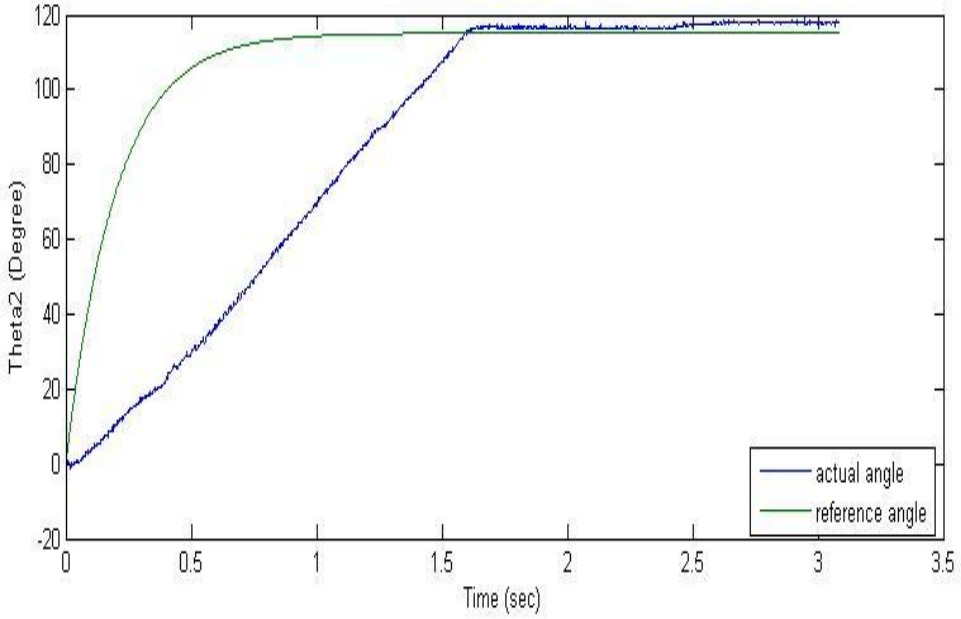


Figure (4.21): Second link response

Giving the robot the same order many times and sketching the results will give an indication of the repeatability and accuracy in the system. This is done in the following two sketches. The robot was ordered to go to the address D4 starting from the home position, as showed in Figure (4.21) the majority of the values are between .03 and .04, Figure (4.22) shows a repeatability between .06 and .07 for the second link.

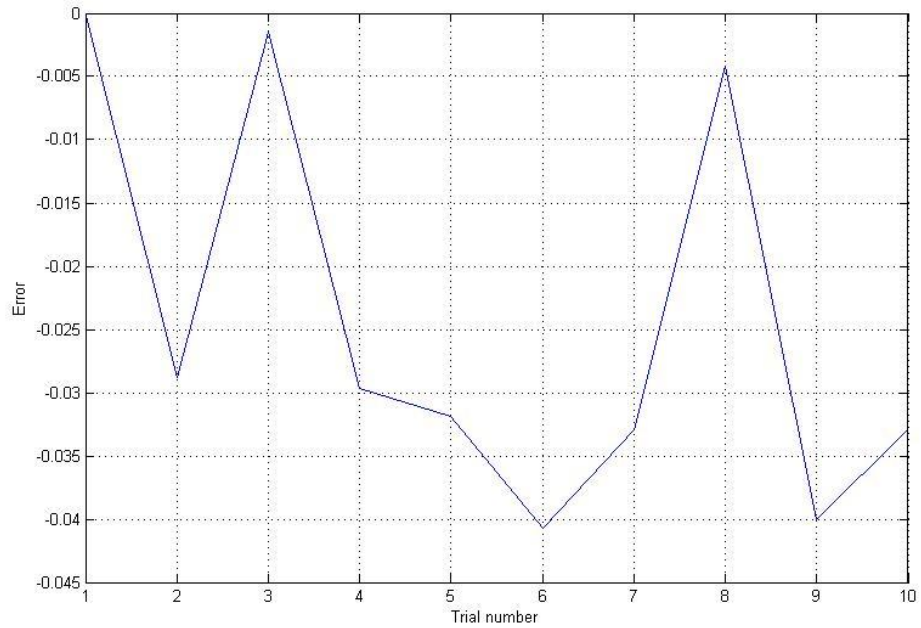


Figure (4.22): repeatability of the first link

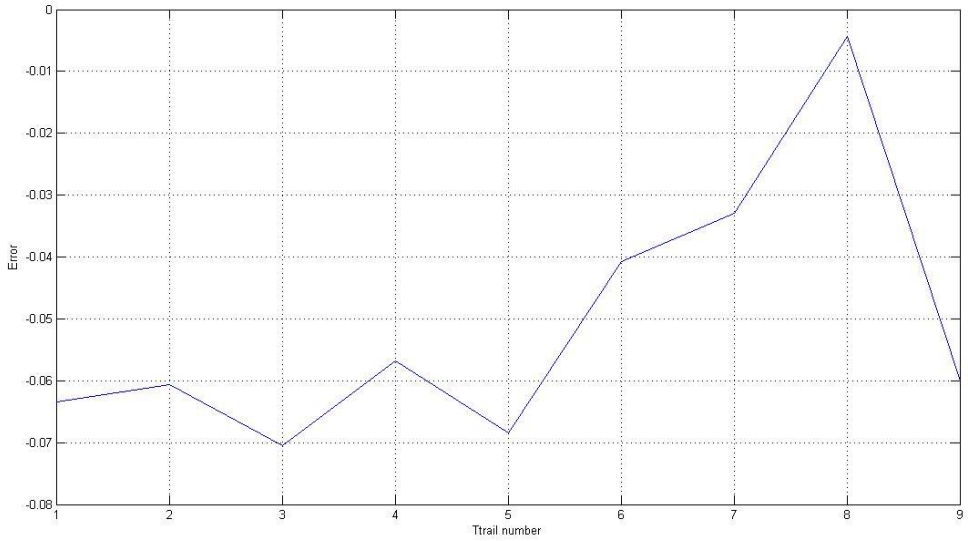


Figure (4.23): repeatability of the second link

CHAPTER FIVE 5

Hardware and Software Implementation

5.1 Overview

This chapter discusses the most important phases in sides, the software and hardware systems. Talking about the details for each phase that are implemented to achieve the project requirements, which are as follow:

1- Graphical user interface (GUI).
2- Playing full game with Robot.
3- Ability to play the game according to the defined options
4- Cheating discovery while the game starting.

5.2 Development Environment Overview

Implement the whole and complete System required software enlivenments and software environments.

5.2.1 Hardware Environment

A number of applications and tools are used to implement hardware system, these software were used to facilitate and speed manufacturing and implementing process, these software explained as follows:

- Orcad 9.2
  is a suite of tools from Cadence for design and layout of printed circuit boards (PCBs), Orcad contains two tools to facilitate design and implement PCB's, Capture tool ,used to draw schematics for sensor circuit and LED's driver circuit, and Layout tool to print layout on PCB's[26].

- KE72 Loader
  Is a utility program is used to load KE-72 configuration file.

- Corel Draw 13
  It is used to form static parts using CNC machines, plastic bars to fix magnetic sensors and flux LED's, draw on wooden pieces to form a place for buttons and promotion cases. Also gluey plastic square pieces are cut to form the chess board cover face.

- 3Ds MAX 2010

  It is Design software that provides powerful, integrated 3D modeling, animation, rendering, and compositing tools that enable designers to more quickly ramp up for production, used to model E-board before actual model is done.

5.2.2 Software Environment

The programs and tools that are used in order to develop the system can be explained as the following:

- Microsoft XNA Studio 3.0

  XNA is an integrated development environment (IDE) that includes tools and code development libraries to develop game programming using C#. So XNA was used to develop a 3D chess game and program with a good graphical user interface (GUI). Many advantages for using XNA over other 3D programs such as easy to use and the compatibility with C#.Net.[17]

- Microsoft Visual C#

  Using C#.Net to customize the Chess engine that is implemented using C# and to program the 3D chess game with game startup menu screen.

- .Net Framework:

  To compile the C# and XNA source code into machine code to be executable and in order to provide us the libraries that the system needs [23].

5.3 System Implementation Phases

This section shows the system phases including the software phases, and hardware phases.

5.3.1 Hardware Implementation Phases

E-Board implementation is separate into two main parts, static part and electrical part, static part talk about implement static pieces, such as, wooden frame. Electrical part talk about implementing electrical parts, such as sensors circuit, static and electrical phases are done in parallel.

5.3.1.1 Static parts phase

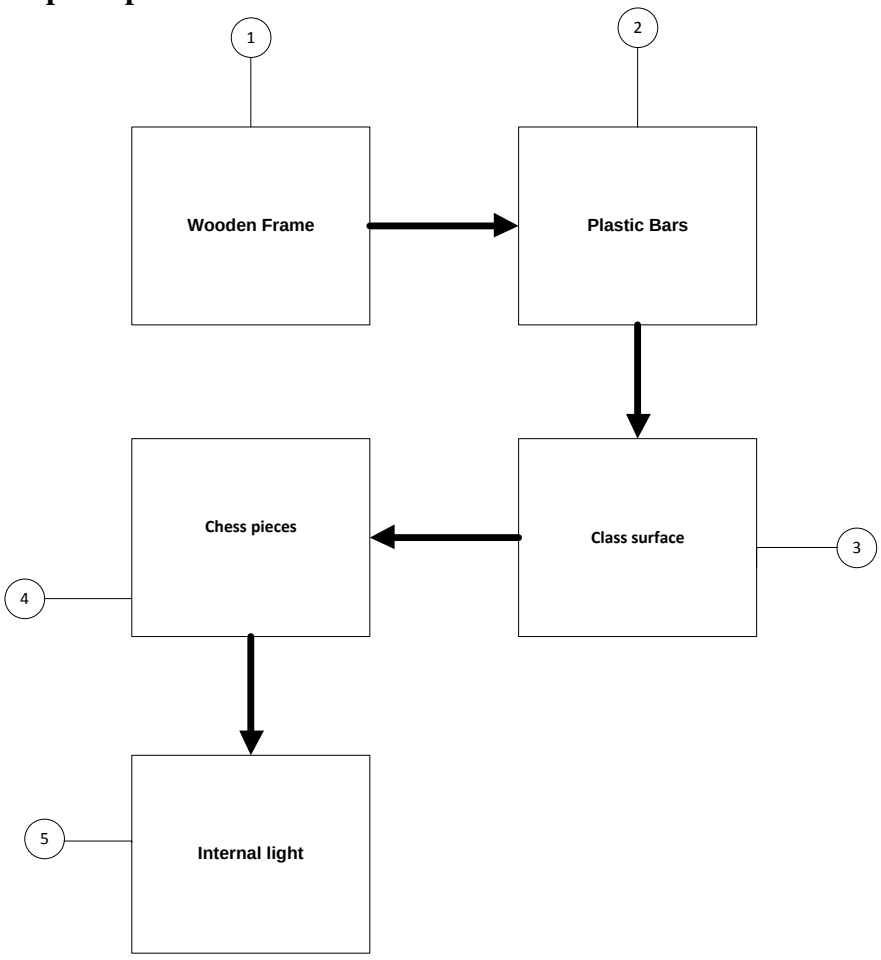


Figure (5.1): Static Implementation Phases

Phase 1: Wooden Frame

It is the main part of E-Board, as shown in figure5.3 A, the frame dimensions, and figure 5.3 B shows the actual board.

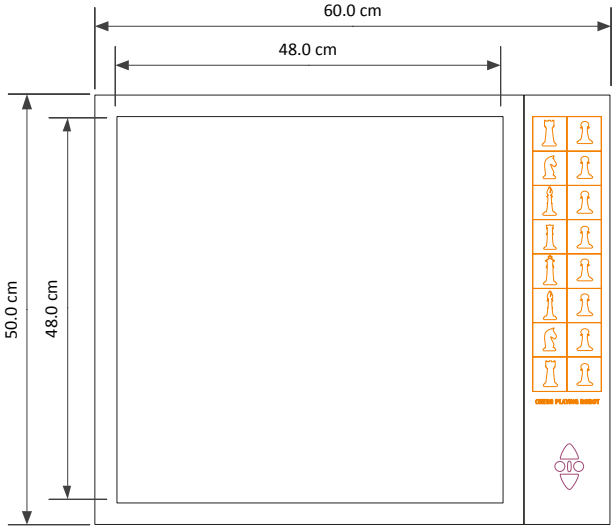


Figure (5.2-A): Basic Wooden chessboard dimensions

Figure (5.2-B): Wood Frame for E-Board

Phase 2: Plastic bars

Plastic bars are cut using CNC machine, these bars are used to fix LEDs and sensors in their proper positions.

Each line has two bars with same size, one for sensors and other one for Leds, Figur (5.4-A) shows the dimentions of these bars, Figure (5.4-B) shows sensors fixed on plastic bars, while Figure (5.4-C) shows two bars fixed togather, and Figure (5.4-D) present all fixed bars.

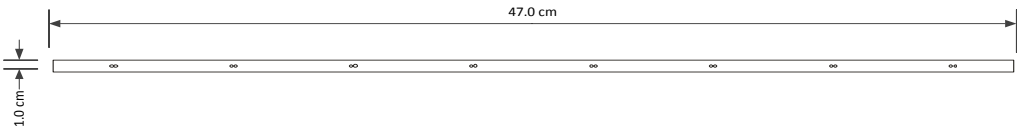


Figure (5.3-A): Bars dimensions.

Figure (5.3-B): Magnetic sensors fixed on bars.

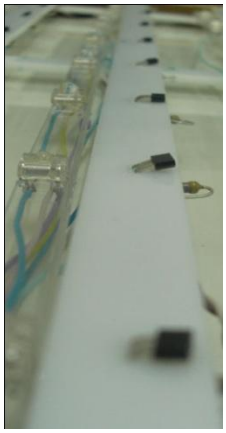


Figure (5.3-C): Bars when fixed together.

Figure (5.3-D): All bars fixed together.

Phase 3: Glass of surface

The upper side of the E-board is made of glass with size of 50cm X 50cm, the sixty four squares are made from gluey plastic and fixed on the glass board, these squares are cut using CNC machine with size 5.2 cm X 5.2 cm, then they were colored to give the chess board its special appearance.

Figure (5.4): Glass of surface

Phase 4: chess pieces

E-Board makes use of glass rocks, each one has magnetic pieced fixed at its bottoms shown in the following figure.

Figure (5.5): Chess Piece

The following table shows rocks dimensions, this is relative with gripper mechanism

Table 5.1: Pieces Dimensions

	Min	Max
Length	4cm	7.5
Base diameter	2.9	2.9

Phase 5: Internal Light

According to the chess game and because the project presents a game, the internal blue light is added to let player more induced with it, also these lights give attraction sight for player. Figure (5.7) shows these internal lights.

Figure (5.6): Internal lights

5.3.1.2 Electrical Implementation Phase

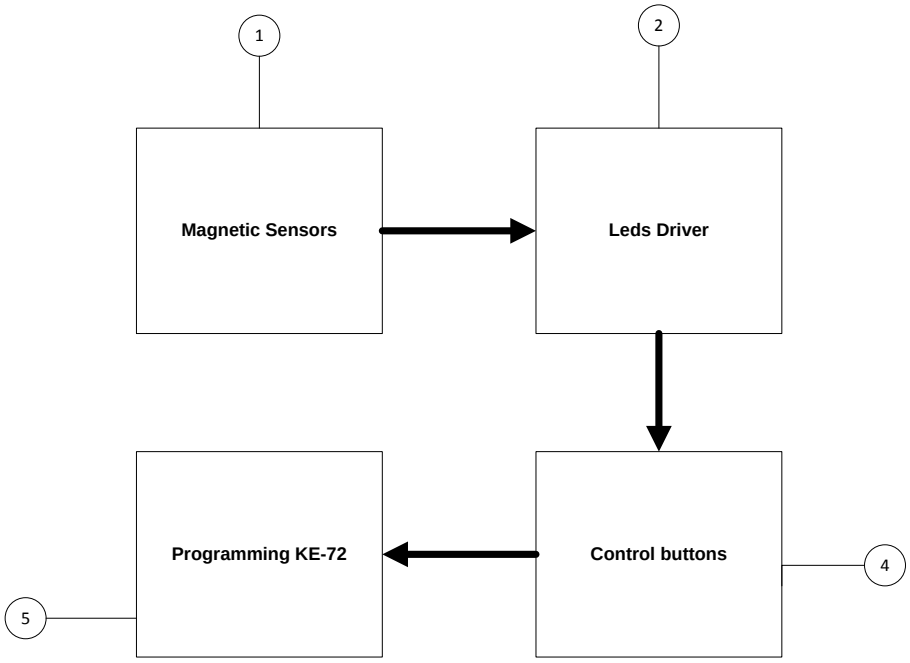


Figure (5.7): Electrical Implementation Phases

Phase 1: Magnetic sensors

Each row (contains eight sensors) are fixed on one plastic bar, also it is connected to the voltage source and is grounded. The output of these sensors are connected to KE-27 terminal as shown in Figure (5.9),these terminals are an intermediate connection between the sensors and the KE-72 Input headers, Figure (5.10)shows the sensors fixed on plastic bars.

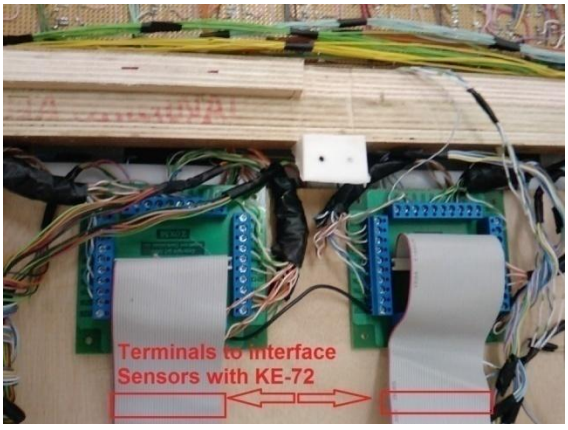


Figure (5.8): Two terminals to interface sensors with KE-72.

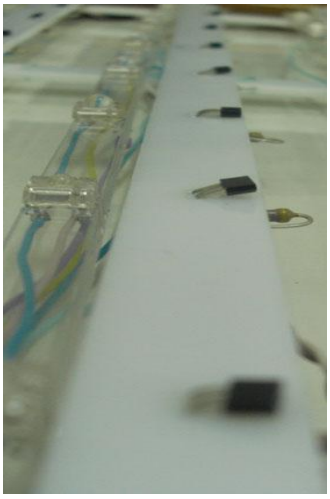


Figure (5.9): Magnetic sensors fixed on a plastic bar.

Phase 2: LED's driver

Figure (5.12) shows the driving circuit, the Orcad application is used to prepare a layout files in order to make a printed cupper board (PCB), all file are made with "Zero Errors", Figure (5.12) shows this layout application. See appendix for top and bottom layout files.

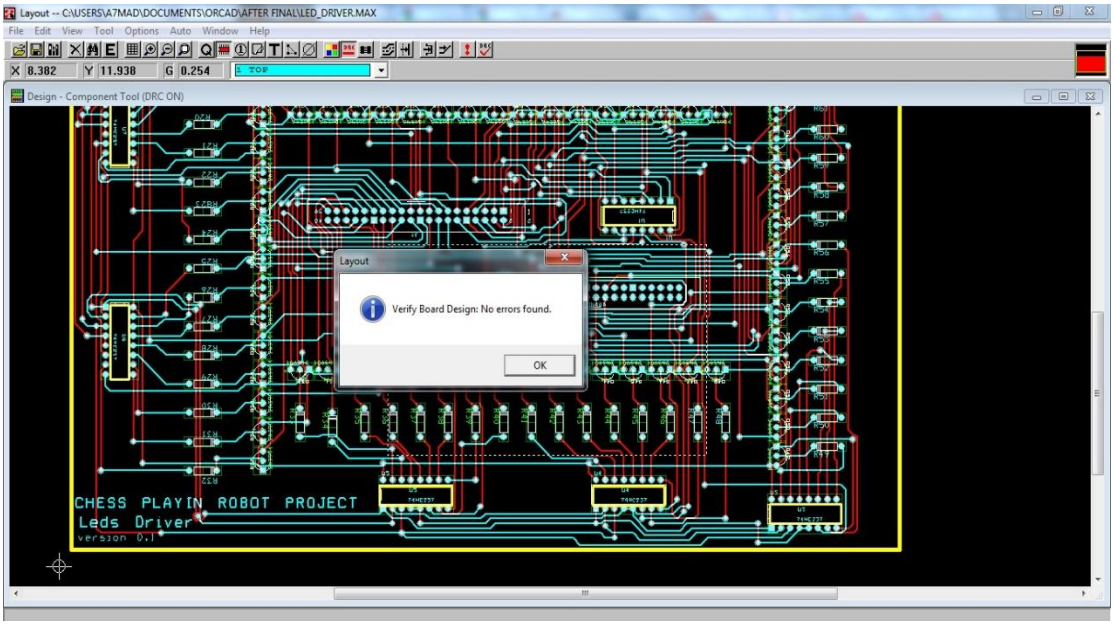


Figure (5.10): Layout application shows driver circuit layouts with zero error.

Unfortunately, the CNC machine in the university is broken, so LEDs driving circuit was welded by hand.

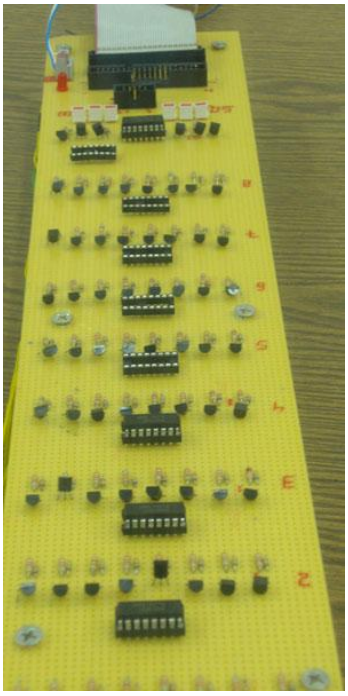


Figure (5.11):LED Driver Circuit

The following table summarizes the LEDs driving circuit components:

Table 5.2:LEDsDriveCircuit Components

Parts	Quantity
74237 decoder	9 pcs
Transistor	70 pcs
resistors	128 pcs
cable	4 feet -16 wire in
IDE 40-pin slot	2 pcs
flux LEDs	64

Phase 3: Control Buttons

The following Figure shows the push buttons circuit, using these buttons the player is able to start his game with easy steps, all what he need to do is to follow the instructions on the LCD screen using these four control buttons.

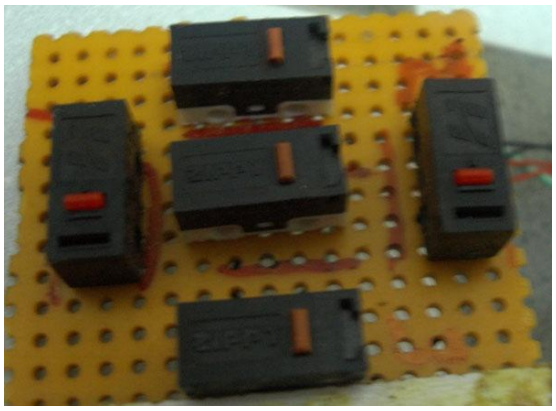


Figure (5.12-A): Control Buttons

The figure below shows buttons circuit fixed with wooden board.

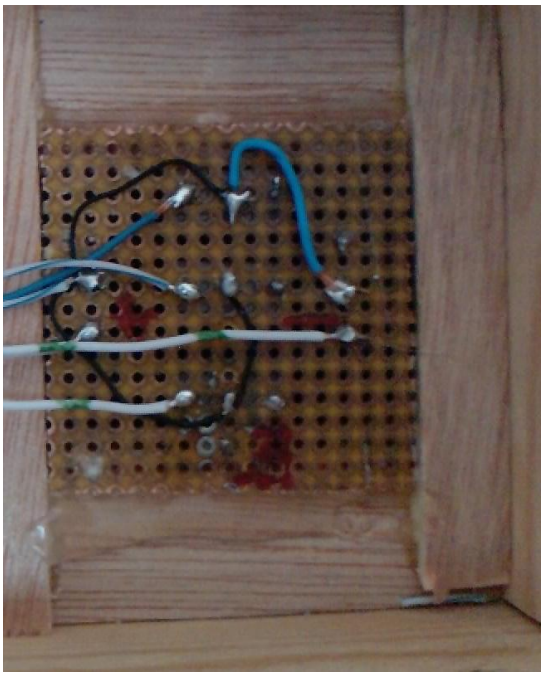


Figure (5.12-B): Buttons Circuit fixed with chessboard.

The complete wooden chessboard is shown below, Figure (5.14-A) shows the board from Top, while Figure (5.14-B) shows the board from bottoms, and all electric circuit is appeared.

Figure (5.13-A): Wooden chess board Top view

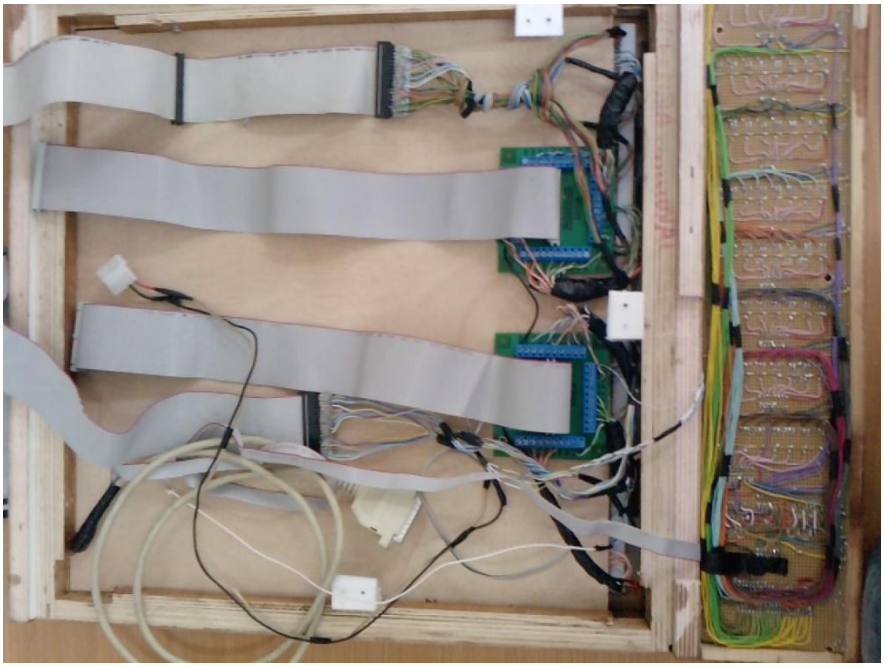


Figure (5.13-B): The wooden chessboard bottom view.

Phase 4: Programming KE-72

Using KE72 Loader, configuration file is loaded to KE-72 with the proper values mentioned in previous chapters.

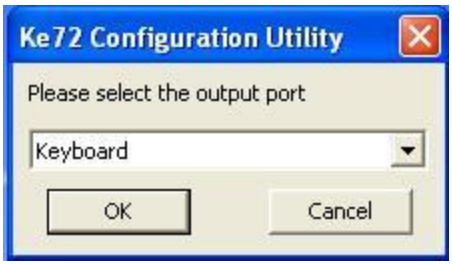


Figure (5.14):KE-72 Loader

5.3.2 Software Implementation Phases:

This section discusses the system software phases that were implemented according to the project requirements.

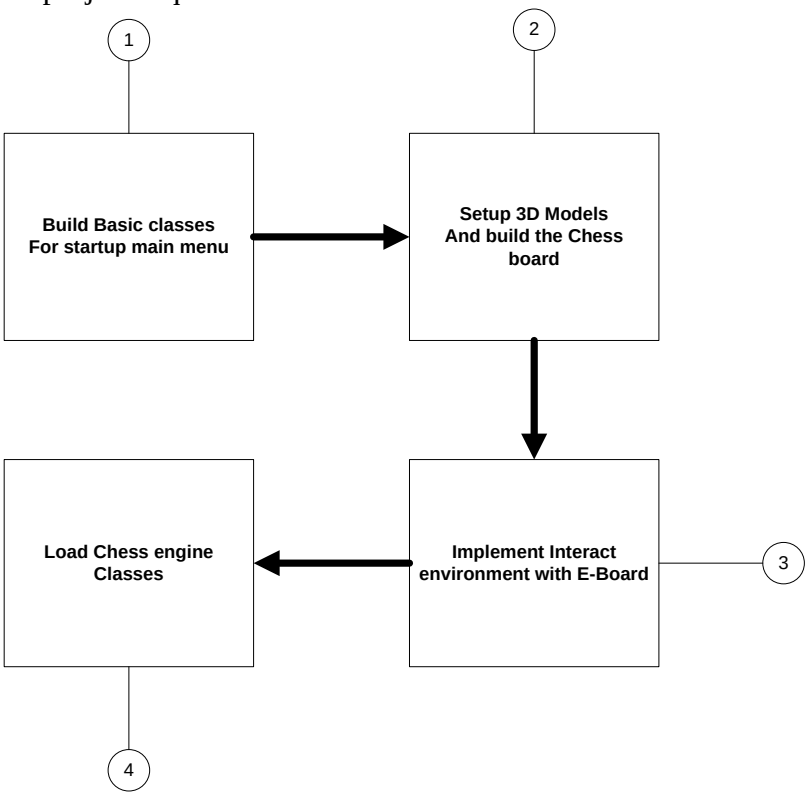


Figure (5.15): Software Implementation Phases

Phase 1: Building the Basic Classes for Startup Main Menu

This phase shows the process of how the main menu was implemented. The main menu consists of many classes, each one has many functions.

- Game1 Class

  This class was implemented to load the main menu on the screen. The main functions for this class are creating the graphics device manager that initialize the graphic device before the game is initialized, and to load the game content on the screen

- Screen Effect Class

  Many classes share the same types of effects such as textures, fonts, sound and models. According to that, the Screen Effect Class was implemented. This class responsible to load all effects that will be used by other classes and to update the changes that may occur such as adding new screen, remove screen, and screen resolution.
  The following code shows sample of how this class loads the effects.

```
protectedoverridevoid LoadContent()
      {
ContentManager content = Game.Content;

          SpritBatch = newSpriteBatch(GraphicsDevice);
          SpritFont = content.Load<SpriteFont>("menufont");
          Shadow = content.Load<Texture2D>("BlueT");
          background = content.Load<Texture2D>("About");
          Boarder = content.Load<Texture2D>("blank");

          Place = content.Load <SoundEffect>("Sound\\place");
          Kill = content.Load<SoundEffect>("Sound\\Kill");

//Load 3d Model for the chess board and pieces
for (int i = 0; i<ChessBoardModel.Length; i++)
          {
           ChessBoardModel [i]= content.Load
<Model>("Model\\"+ModelName [i]);
          }
       }
```

- Game Screen Class

This class is an abstract class that is inherited by many derived class. Using abstract class to provide common definition of base class that multiple derived classes can be shared. It shows the important methods that other derived classes can share. The following code shows these methods that may be override by derived class.

```
publicvirtualvoid KeyPressEvent(Form1  key) { }
publicvirtualvoid Update(GameTime gametime) { }
publicvirtualvoid Draw(GameTime gametime) { }
publicvirtualvoid LoadContent() { }
publicvirtualvoid UnLoadedContent() { }
publicvirtualvoid Initialize() { }
```

- Load Back Ground Class:

This class was implemented to load the background texture for the main menu.

- Menu Screen Class

This class define the main menu list which are “Play” new game, ”Resume” game, game “Options”, and “Exit” game. Each one of these has its own events to lead the user to an action.

```
MenueText Start = newMenueText(Strings.Main_Play);
MenueText Resume = newMenueText(Strings.Main_Resume);
MenueText Option = newMenueText(Strings.Main_Option);
MenueText About = newMenueText(Strings.Main_About);
MenueText Exit = newMenueText(Strings.Main_Exit);
```

- Menu Text Class

By implementing this class one can get any event occurs to do its reaction. In addition to draw each menu list according to the passed values. The following code shows the event definitions.

```
publicclassMenueText
    {
publiceventEventHandler<EventArgs> Selected;
protectedinternalvirtualvoid OnSelect()
        {
if (Selected != null)
                Selected(this, newEventArgs());
        }
    }
```

- Main Menu Text Class

  This class was implemented to draw the screen texts with their highlights shadow.
  This class inherits form the Game Screen Class to override the needed methods such as KeyPressEvent, that is responsible to handle the control key changes, and the draw method to load the screen content on the specified position.

- Popup Message Class

  When the user selects a specified action such as exit game, this requires confirmation from the user. According to that the Popup Message appears for the user to take the confirmation

- Option Class

  According to the chess game rules, some options are required such as playing the game with multi-level difficulty modes or with a countdown timer and playing with help mode or without by lighting the LED's .
  This class was implemented to create these options after the user start his new game.
  The option value is saved on XML file to be loaded when the game starts. If the player starts the game without selecting the option game, the game will load the last values from XML file.
  However, if the XML file does not exist, the game will use the default value to start the game

- Loading Game Class

  This class is an interactive class between the main menu and 3D chessboard, it is responsible to load the 3D chessboard models and build the required classes for this process.

- Promotion Screen Class:

  Promotion case occurs in the chess game if any pawn piece arrives at the end of the other side of the board. According to that, the pawn can be upgraded to one of the four possible pieces (Queen, Rook, Knight, bishop). This screen will pop up if the promotion occurs while the game is running; waiting the player to select the piece from this screen to continue the playing.

- Game Over Class

  The player may loss his game if he exceeds the timer intervals, or if the enemy had checkmate or stalemate. According to that the game over screen will appear.

- Form1 Class :

  This form was created to load the content of XNA classes on it. In addition, to use the keypress event handler to get any changes occurs on the E-Board or function buttons.

Phase 2: Setup laibaries and Build 3D Models:

This phase shows the process of implementingthe 3D model for the chess board.

- Loading Game Class:

This class was implemented to load and setup the 3D chess board . it consists of many methods, these methods are shown as follows with their function:

- Model World Transform

  Chess board consist of squares and pieces model, these model must be shown onthe 3D space. According to that, each model needs to idenitfy the world transofrm. By defining the world transform the models will have their position located on it.

  This method defines the world transform position for the chess pieces which are eight for both white and black players, and for the chess squares which are sixty four squares.

- Load Models

  This method was implemeted to load the chess pieces and the squares model, and to define the matrix model that contains the model bons number.

- Draw Model

  This method was implemented to draw the models according to their world trasnform. Using the triangle picking pipline library the modelsareconverted from local view to world space view, then they will be drawnon the screen[18].

- Draw Available Square

  This method is responsible to draw the possible squaresonthe3D board. Thus, this method receives the possible movement form the chess engine for the selected piece; then it finds the possible square on the board, and changes the color for these squares to show that these are the possible squares.

- KeyBoard Input

  This method is responsible to receive any change occurs on the E-board. After that it passes these values to the specified class in the chess engine to make the decoding process.

- UpdatePicking

  To reflect the changes that occurs in the E-board on the 3D board. It requires knowing the name and the position of the piece.
  According to that, this method obtains the piece data, which are the name, ID, and type of the move from the chess engine classes.

- Make a Move

  This method is responsible to find the piece that have to be moved, the next place, and the type of the movement on the 3D board, according to the obtained values form Update Picking method.

- Animation Move

  This method is called after Make a Move method to find the where the specified piece is to move. After that this method finds the linearly interpolates between pick piece and place piece position, and generated the animation to move the piece on 3D board to the new valid square.

- SquareM Class

This class was implemented as atemplate defintion for chess board square.It contains methods for the model,and model world transform.

```
publicclassSquareM
    {
publicModel Model { get; set; }
publicMatrix WorldTransforms { get; set; }
publicMatrix[] AbsoluteBoneTransforms { get; set; }
    }
```

- Chess Man Class

This class was implemented as template defintion for chess pieces to determine the chess pieces name,model, and their world transform.

The following sample code shows the defintion of the Chessman class with its properties

```
publicclassChessman
    {
publicForce Force { get; set; }
publicModel Model { get; set; }
publicMatrix WorldTransforms { get; set; }
publicMatrix[] AbsoulateBoneTransforms { get; set; }
    }
```

- Chess Board Square Class

This class was implemented to define the whole chessboard. It also determines the color of the piece that is held up, and if this square is empty, also

```
publicclassChessboardSquare
    {
publicbool? IsWhite { get; set; }
publicbool? IsEmpty { get; set; }
publicint ForceJ { get; set; }
publicint ForceI { get; set; }
publicForce? Force { get; set; }
}
```

Phase 3: Implement Interact environment with E-Board

- Possible Movement Class

  This class was implemented to control the LED's driving circuit by sending the possible movement value through the parallel port. According to that, this class receives the possible movement form the human player class.

- Parallel Port Class

  This class import the Dynamic-Link Library "inpout32.dll ", to enable the data to send through the parallel port to LED's controller.

Phase 4: Load Chess Engine Classes

This phase deals with the most important classes that are used to run the system correctly.

- Human Class

  This class is responsible to receive the player movement that was done on the E-board. However, the most important functions for this class are as follows:
  - Decode the E-board value to decide if it is picking up or place value.
  - Make sure that these values are valid
  - Generate the possible movement according to the picking up value.
  - Active the intelligence classes

- Robot Class

  This class is responsible to send the intelligence value to the Robot controller through the local network. In addition, this class has many functions such as the following:

  - Initialize the connection between Robot class and Robot controller
  - Send the intelligence value to Robot controller
  - Receive the feedback from Robot controller.
  - Receive the E-board value resulting from the playing Robot.

- Cheating Class

  During the game a cheating movement may occur, according to that, this class was implemented to handle such a problem.
  In addition, this class provides a solution to solve the cheating cases.

- Timer Class:

  The players can play a new game in two different modes; one of these modes is playing with countdown timer. According to that, this class was implemented to display a real countdown timer.

- Difficulty Class

  One of the options is to play the game in level mode. Therefore, this class is responsible of saving the user options, and loading a new game according to these options

- Game Class

  This class is used to reflect the player movement on the chess engine, and to activate the intelligence to generate the next move. The main functions for this class are such as the following:
    - Make an internal move according to human player movement.
    - Add game movement to backup
    - Activate the intelligence side

- Player Class:

  This class contains the most intelligent algorithms that are run to generate next move.
  After the player did his move correctly, this class will be activated to think in the next move.

- Board Class

  This class is responsible of defining the chessboard approach, which is the 0X88. Each square has a unique ID according to the 0X88. The valid squares that have zero resulting by applying logic, with 88 hexadecimal, and with the square ID.

- Piece Class :

  This class was implemented to determine the pieces type, and their possible movement. In addition, it keeps tracking each piece movement during the game.

5.4 Graphical User Interface

Designing an interactive user interface increases the interaction between the user and the system. Chess Playing Robot interface should have a good design to give the user a high facility to play his game and to do other functions such as game's options or pause.

The player can interact with the system interface by using the function key which are fixed on the E-Board.

Figure (5.16): Startup Main Menu

5.4.1 Game Options:

Game options screen give the user many choices to start his new game, including playing the game in level mode or playing it in custom counter down timer. In addition, the player can select the help mode by lighting the possible square or not.

- General Difficulty Level

Figure (5.17): Game Option (General difficulty Level)

In this screen, the default option value is shown, playing the game with level one and guiding the player by lighting the possible moves. When the player finishes the option selection he should click Back to return to the Main menu. These option's parameters are saved when the player close this screen and goes back to the main menu.

- Custom Counter Down Timer

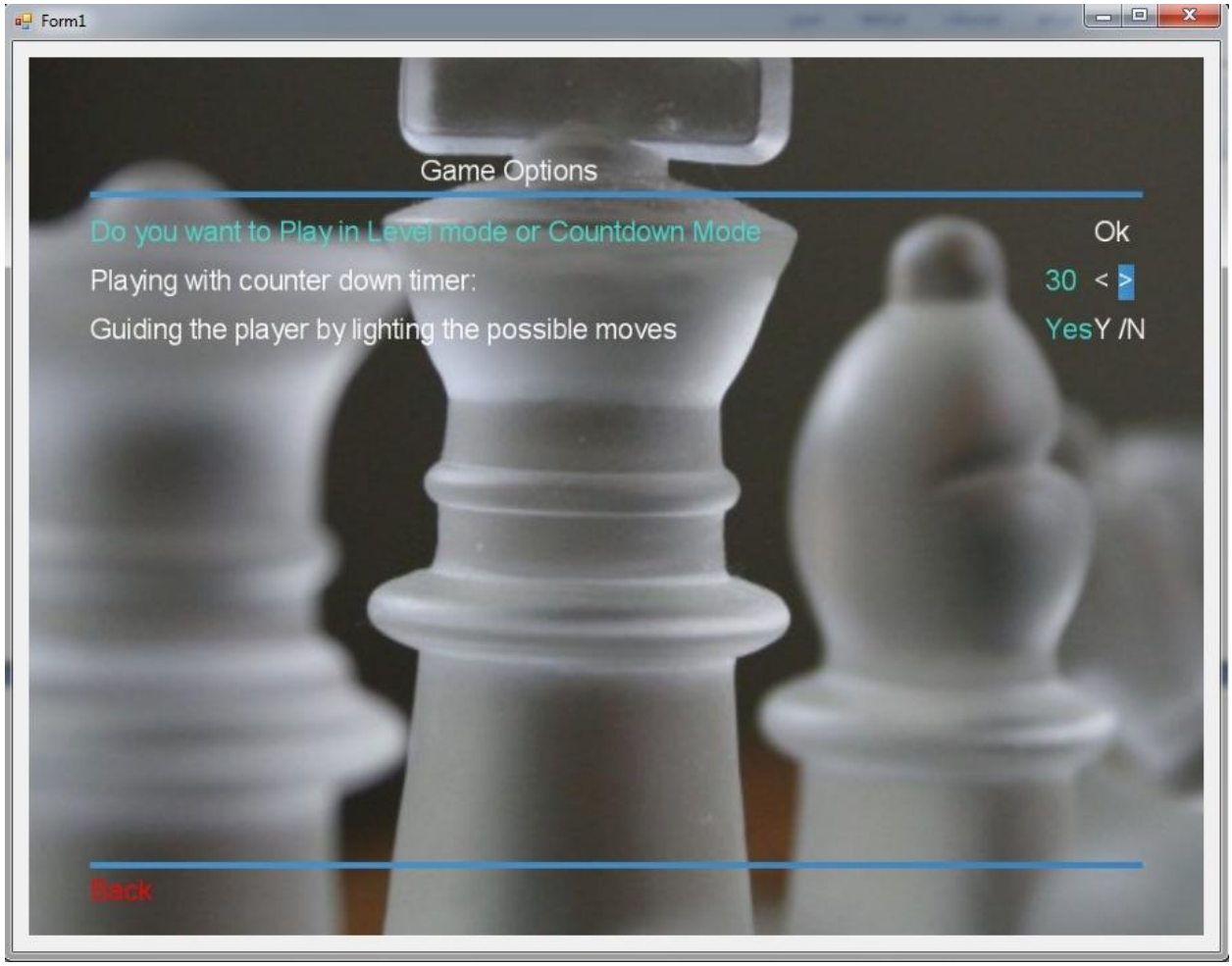


Figure (5.18): Custom Counter Down Timer

If the player wants to play with counter down timer he must click on Ok then the user can select the period of time in seconds using up down key.

5.4.2 Chess Playing Robot Board.

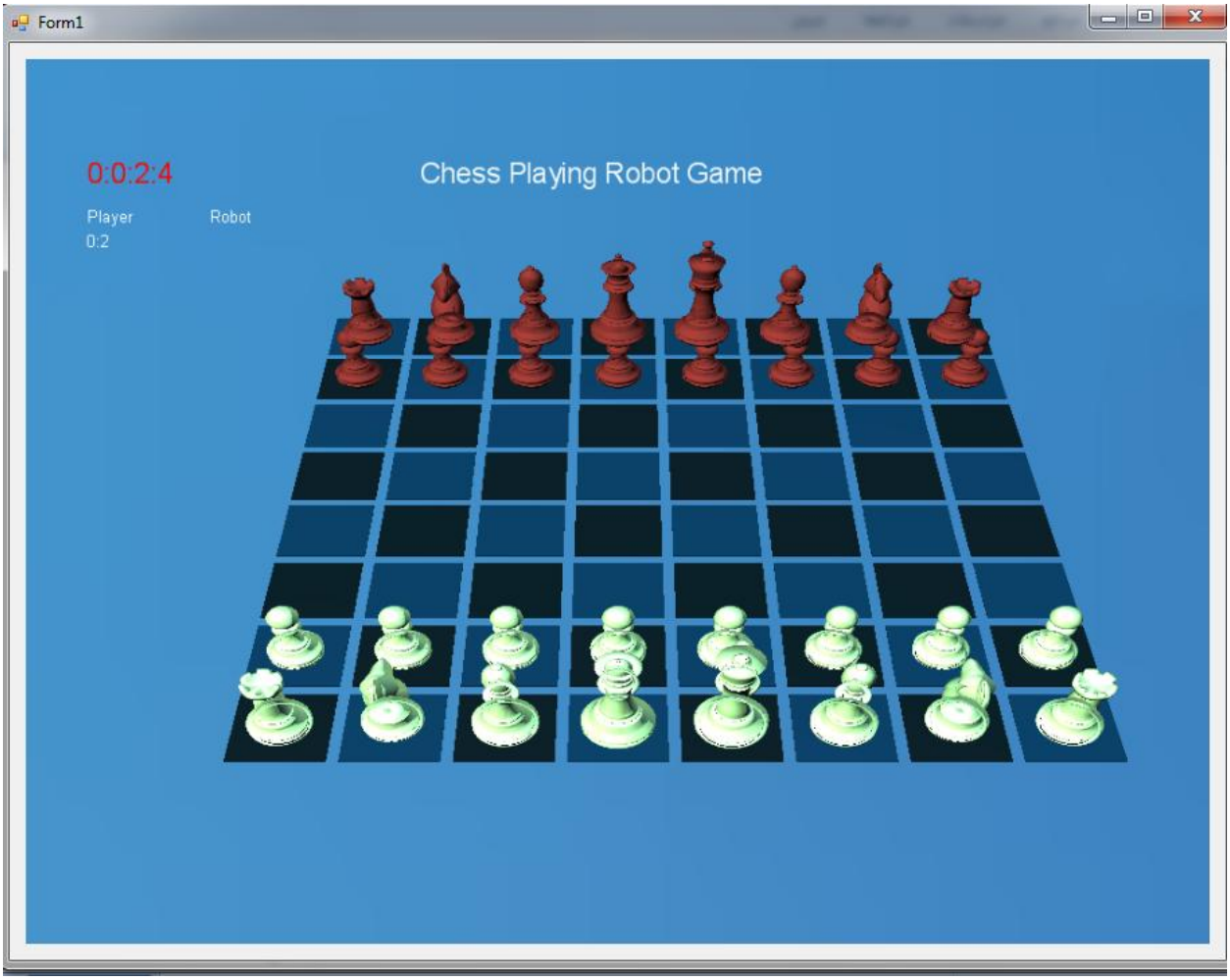


Figure (5.19): Chess Playing Robot Board.

This screen shows the Chessboard and the pieces at the starting position, timers for each player (Human and Robot). Any changes occurs on the E-board will reflect on the board.

5.4.3 Pause Screen

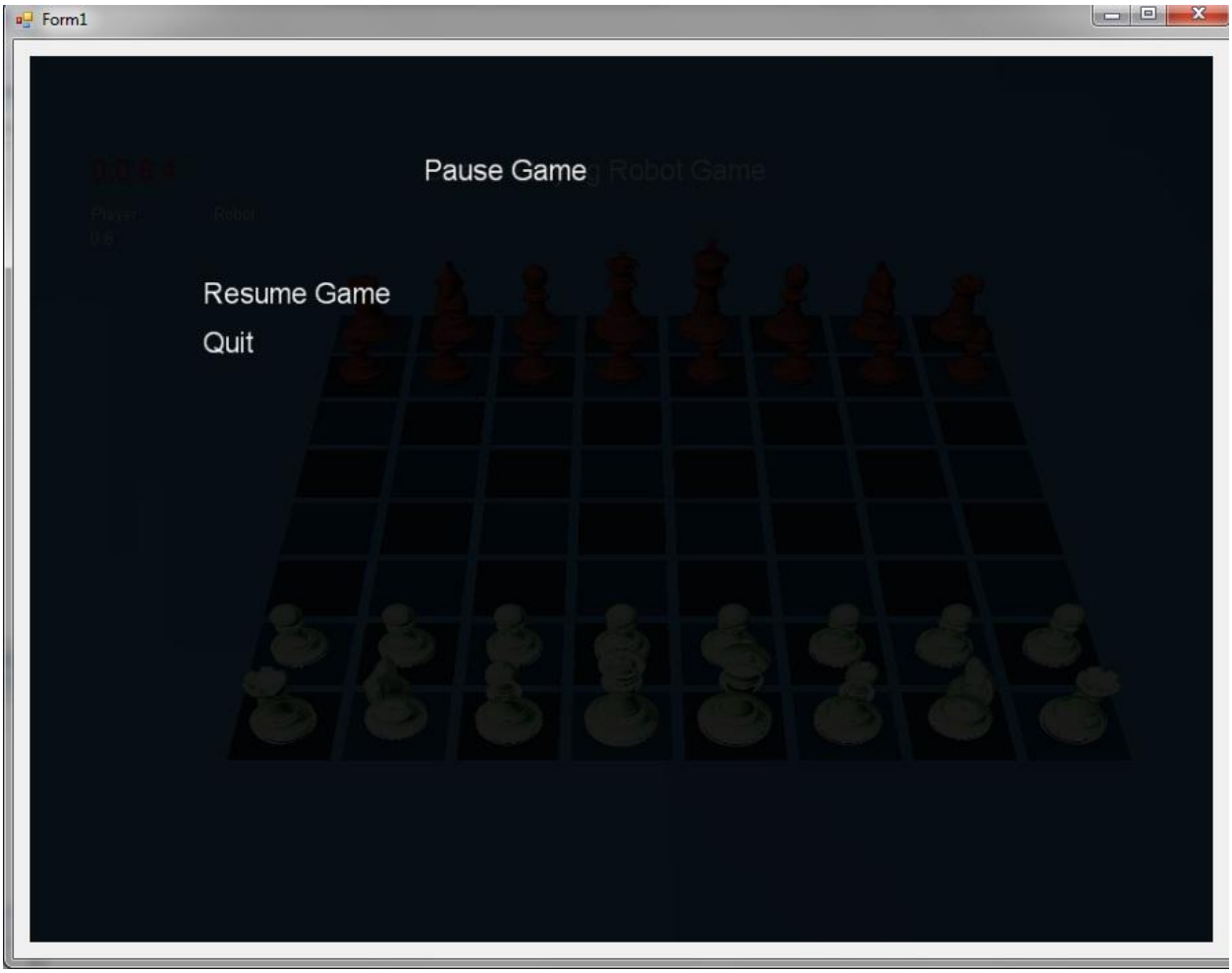


Figure (5.20): Pause Game

When the game starts, the player can pauses the game by clicking on the ESC button, this screen will pop up. If the player wants to resume the game he must click on Resume Game, otherwise he may click Quit to go back to the Main menu.

The player can pause the game just if plays in the level mode, otherwise he will not be able to pause the game.

5.5 Testing and Validation

This section provides the system testing in term of system software and hardware testing.

5.5.1 System Software Testing

This section shows the result of testing the system in term of its load game option, exceeding the timer intervals, error messages, and promotion screen.

5.5.1.1 Game options

When the player selects the game option, the selected option will be saved on Xml file to be loaded when the game is run. Otherwise, the game will load the default value to start the game.
The following figure shows the player when selecting level nine to play his new game with help mode.

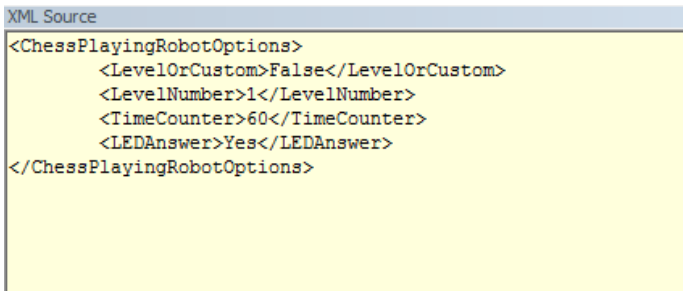


Figure (5.21): Xml File (Level Mod)

However, this figure shows the player when selecting countdown timer mode with 60 second interval.

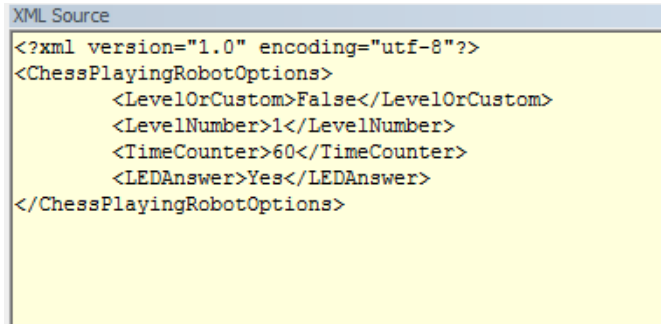


Figure (5.22): Xml File (countdown timer mod)

5.5.1.2 Game over

Game over screen will pop up if the player does not play during the defined interval time, or if the play has checkmate or stalemate. When this screen pops up, it shows messages telling the user that the game was ended.

Figure (5.23): Snapshot for Game over Screen

By pressing any button, the player returns to the main menu to play new game or quit.

5.5.1.3Promotion Screen

This screen will pop up, if the promotion case occurs during the game allowing the human player to select the new piece nested pawn piece.

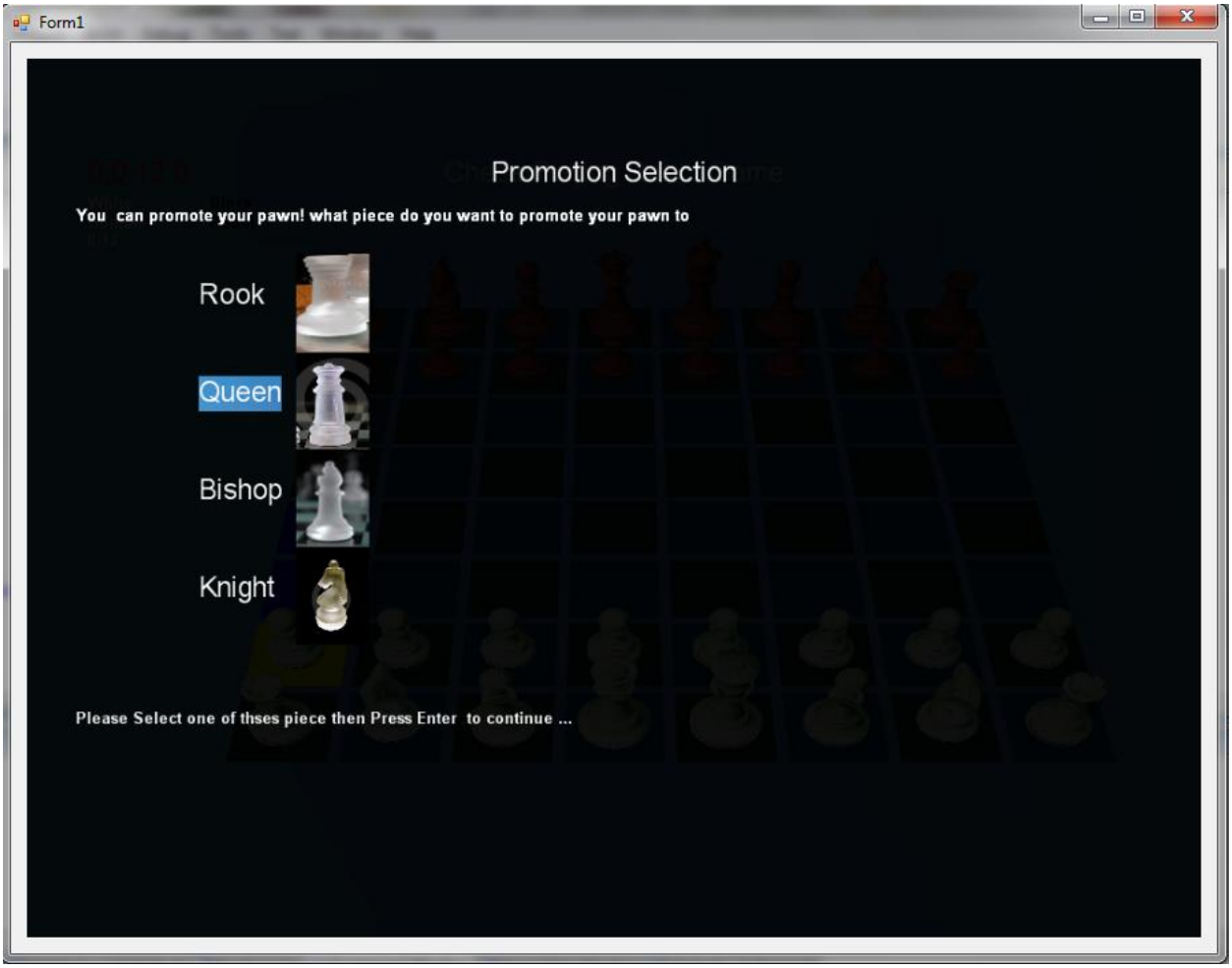


Figure (5.24): Snapshot for Promotion Screen.

5.5.1.3 Error Messages

The error messages will appears if the player try to play incorrectly, the game will pause till the player returns the piece to the original place then moving it to valid position.

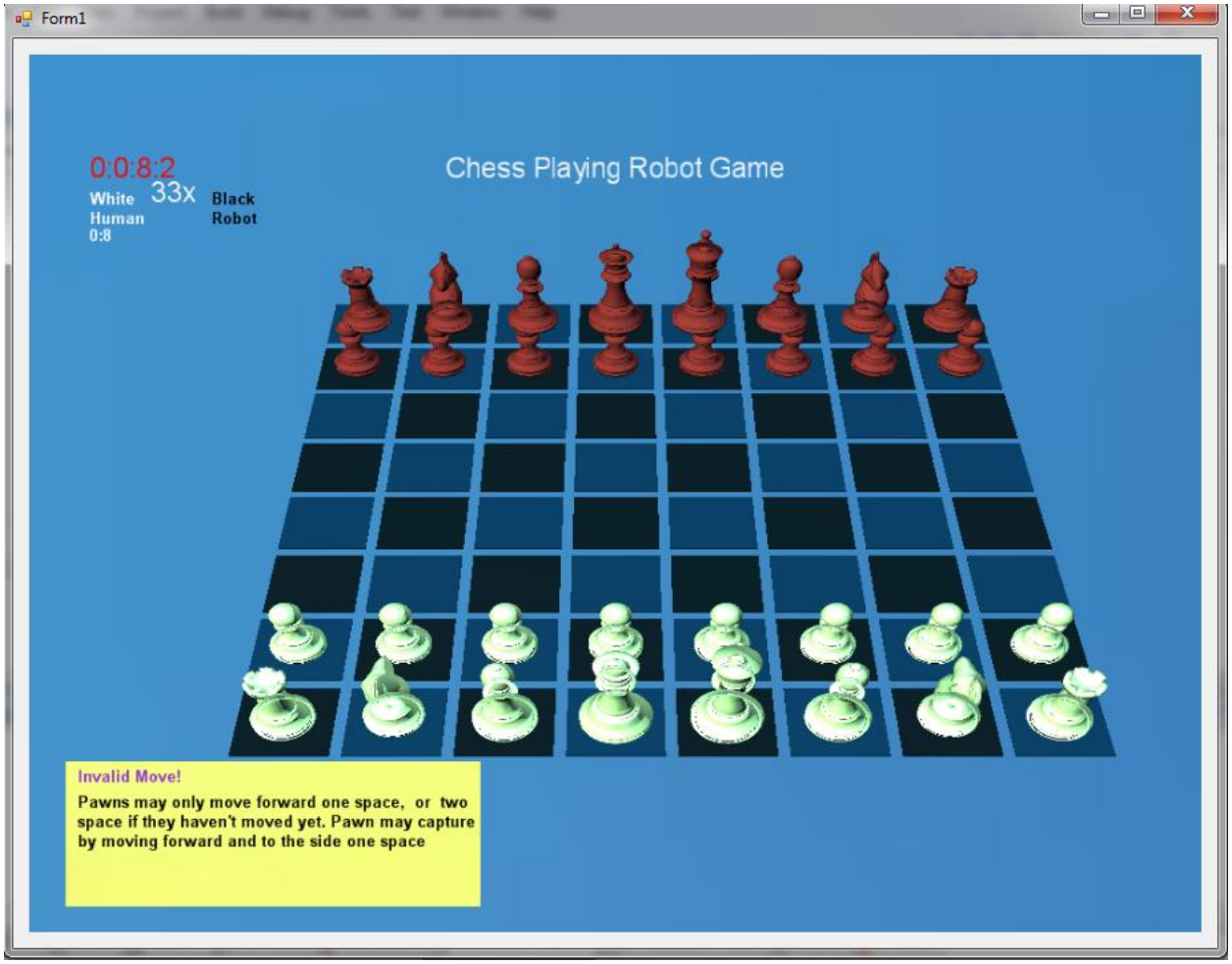


Figure (5.25): Snapshot for Error Message.

This screen shows that the player picking pawn piece then placing it on an invalid square, according to that the error message appeared.

5.5.2Hardware Testing

In this section many tests on the system's parts are done to make sure that they work properly and that they do their job accurately. After that the whole hardware system is tested to make sure that all subsystems work correctly together.

5.5.2.1 Magnetic Sensors Calculations

5.5.2.1.1 Measured calculations

Using voltmeter and ammeter, figures below shows the practical values. The circuit consumes 0.431 mA (I_L) ,without any actuated sensor.

Each actuated sensor consumes 95 mA (I_S), this leads to the maximum current that could be consumed (IT);

$$IT = CurrentforOnesensor * MAXnumberofactuatedsensors + Lekagecurren$$

$$IT = 95mA * 32 + 0431mA = 3.471A \quad (5.1)$$

5.5.2.2 LED's Driving Circuit Calculations

The calculation of the total current consumed by LED's driving circuit is shown in this section. To calculate the total current, few steps are needed to be done starting from transistor calculations to IC decoders calculations,

5.6 Summary

In this chapter the implementation and testing of both hardware and software components of the system, were successfully completed. The testing made as specified in the requirement list at the beginning of this project. More specifically those tasks were completed:

- A software interface between two different environments C sharp and MATLAB.
- An attractive graphical user interface to enforce actions made by the control keys that are fixed on the E-board.
- The completion of a highly sensitive chessboard to sense any changes that may occur on the board.
- A good LEDs driving circuit controlled with a very advanced system control.
- A full 3D chessboard that interacts with the E-board changes. plus cheating discovery algorithms and error message system.

References

1. Gracie, S., Matthey, J., & Rankin, D. (2004). *Lego Chess Robot*. Glasgow: University of Glasgow.
2. Chess Playing Robot. (n.d.). *Chess Playing Robot*. Retrieved January 19, 2011, from http://www.chessplayingrobot.com/index.html
3. MCU CHESS. (n.d.). *courses.cit.cornell.edu*. Retrieved January 19, 2011, from http://courses.cit.cornell.edu/ee476/FinalProjects/s2006/umh3_fcf3/index.htm
4. COUR, T., LAURANSON, R., & VACHETTE, M. (2002). Autonomous Chess-playing Robot. *Autonomous Chess-playing Robot*, *27*(ECOLE POLYTECHNIQUE), 1-27. Retrieved January 10, 2010, from http://www.seas.upenn.edu/~timothee/papers/ChessAutonomousRobot.pdf
5. URTING, D., & BERBERS, Y. (2002). A Low-Cost Chess Robot. *MarineBlue*, *6*, 1-6. Retrieved October 10, 2010, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.6961&rep=rep1&type=pdf
6. Alshamasin, M., Ionescu, F., & Al-Kasasbeh, R. (2009). Kinematic Modeling and Simulation of a SCARA Robot by Using Solid Dynamics and Verification by MATLAB/Simulink. *European Journal of Scientific Research*, *37*(EuroJournals Publishing), 388-405. Retrieved October 10, 2010, from http://www.eurojournals.com/ejsr_37_3_05.pdf

7. Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). *Robot modeling and control* . Hoboken, NJ: John Wiley & Sons.

8. PCI-QUAD04. (n.d.). *Measurment Computinh*. Retrieved March 12, 2010, from www.mccdaq.com/PDFs/Manuals

9. Using Quadrature Encoders with E Series DAQ Boards - Developer Zone - National Instruments. (n.d.). *NI Developer Zone*. Retrieved January 19, 2011, from http://zone.ni.com/devzone/cda/tut/p/id/4623#0

10. pawn, o. (n.d.). Computer Chess Programming Theory. *Colin Frayn's Home Page*. Retrieved January 19, 2011, from http://www.frayn.net/beowulf/theory.html

SharpChess: Home. (n.d.).*SharpChess: Home*. Retrieved January 19, 2011, from http://www.sharpchess.com

11. Retrieved January 19, 2011, from http://www.cis.uab.edu/hyatt/boardrep.html

12. Ways of integrating MATLAB with the .NET - CodeProject. (n.d.). *CodeProject - Your Development Resource*. Retrieved January 19, 2011, from http://www.codeproject.com/KB/dotnet/matlabeng.aspx

13. My standard servo gripper Â« J-J Shortcut's Blog. (n.d.). *J-J Shortcut's Blog*. Retrieved May 22, 2011, from http://jjshortcut.wordpress.com/2010/03/05/my-standard-servo-gripper/

14. Introduction To Robotics, Mechanics And Control, John J Craig

15. MathWorks - MATLAB and Simulink for Technical Computing. (n.d.). *MathWorks - MATLAB and Simulink for Technical Computing*. Retrieved May 22, 2011, from http://www.mathworks.com/

16. NET Framework - Wikipedia, the free encyclopedia. (n.d.). *Wikipedia, the free encyclopedia*. Retrieved January 26, 2011, from http://en.wikipedia.org/wiki/.NET_Framework

17. Getting Started with XNA Game Studio Development. (n.d.). *MSDN | Microsoft Development, Subscriptions, Resources, and More*. Retrieved May 20, 2011, from http://msdn.microsoft.com/en-us/library/bb203894.aspx

18. Triangle picking pipeline - extracting more data. - App Hub Forums. (n.d.). *Forums - App Hub Forums*. Retrieved May 20, 2011, from http://forums.create.msdn.com/forums/p/72167/440312.aspx

19. Electrofun LTD. (n.d.). *Refresh*. Retrieved May 22, 2011, from http://www.electrofun.biz/catalog/product_info.php?products_id=90&osCsid=24a6aba2d53c8b93cd29fee54a0dd292

20. DUAL FULL-BRIDGE DRIVER.Retrieved May 22, 2011, from http://www.st.com/stonline/books/pdf/docs/1773.pdf

21. Instrumentation Amplifier AD620 .Retrieved May 22, 2011, from http://www.analog.com/static/imported-files/data_sheets/AD620.pdf

22. Power Amplifier LM1875.Retrieved May 22, 2011, from

http://www.national.com/ds/LM/LM1875.pdf

23. Introducing C# and the .NET Framework :: ASP Free. *ASP Help, ASP Tutorials, ASP Programming, ASP Code :: ASP Free*. Retrieved January 19, 2011, from http://www.aspfree.com/c/a/C-Sharp/Introducing-CSharp-and-the-NET-Framework

24. Hagstrom Electronics - Products - KE72. (n.d.). *Hagstrom Electronics - PC Keyboard Interface Products*. Retrieved May 22, 2011, from http://www.hagstromelectronics.com/products/ke72detail.html

25. Hall Effect Sensors,Retrieved May 22, 2011, from http://content.honeywell.com/sensing/prodinfo/solidstate/technical/chapter2.pdf

26. OrCAD Introduction. (n.d.). *Connexions - Sharing Knowledge and Building Communities*. Retrieved May 22, 2011, from http://cnx.org/content/m11650/latest/

27. Internet-Based Control of Double Inverted Pendulum on Cart, Mohammad Albakri, Aziz Arafe

Appendix

A

Layouts and Schematics

Appendix

B

Source Code