INTELLIGENT ROBOT PLAYER

Submitted by

Tariq Mostareeh

1009906

Faculty of Computing and Information Technology

King Abdul Aziz University

Jeddah – Saudi Arabia

08, 1436 – 06 ,2015

بسم الله الرحمن الرحيم

قال الله تعالى :

{ إِنَّمَا يَخْشَى اللَّهَ مِنْ عِبَادِهِ الْعُلَمَاءُ } فاطر:28

وقال الرسول عليه الصلاة والسلام :

" من سلك طريقًا يطلبُ فيه علمًا ، سلك اللهُ به طريقًا من طرقِ الجنةِ ... "

المحدث : الألباني
خلاصة حكم المحدث : صحيح

Certification of Research

I certify that I have reviewed this project done by: ***Tariq Mostareeh – 1009906*** during the **1st or 2nd** semester of the academic year **2015**. And I approve the project submission for discussion by the project responsible committee to grade it as the graduation project degree in ***Computer Science*****.**

Supervisor: Ph.D. Anas Fattouh

Signature: ……

Declaration

We certify that the implementation of this graduation project has been done by our own efforts. This includes the preparation and writing of the analysis, design, programming and documentation. We have got assistance during the implementation period of this project, after Allah, from the references mentioned in this thesis, and Allah is the witnesses of what we say.

Student Name: Tariq Mostareeh

Student ID:1009906

E-mail: Tariqmym@gmail.com

Mobile: 0590903131

Signature: .....

Intelligent Robot player(using Connect4 game)

Abstract

- Build a robot that can play Connect4 game with an intelligent thinking similar to human .

.

Aim of the project

In our project we will use robotics and AI algorithms to build an intelligent robot player that can play a physical games with a human like Connect 4 , so in the final stage it expected from the Robot Player to be :

- Smart : that it can win and do tricks on the game.
- Learn-able : it could learn from his mistake .
- Fast : it should be fast on taking the decision and in his movement.

User-Friendly : it should be easy to use and operate.

Dedicated to

To our great families who provided a generous assistant as always during this in order to complete this project.

Acknowledgments

Special thanks to Dr. Fattouh (Anas) for his help in achieving the project goals. Thanks to Dr. Aiiad Albeshri for his coordination and management for the senior project.

Wish you all the best ,

Table of Content

Certification of Research ..... **iv**

Declaration ..... **v**

Abstract ..... **vi**

Aim of the project vi

User-Friendly : it should be easy to use and operate. ..... **vi**

Dedicated to ..... **vii**

Acknowledgments ..... **viii**

Table of Content ..... **ix**

Chapter 1 : Introduction **2**

1.1 Motivation 3

1.2 Consideration 3

1.3 Project Proposal 4

1.3.1 Problem definition 4

1.4 Project scope 4

1.5 Project type 4

1.6 The outcome 5

1.7 Target Users 5

1.8 Suggested solution 6

1.9 Project Management 7

1.9.1 Work Breakdown Structure (WBC) 7

1.9.2 Time Estimates 9

Chapter 2 : Literature Search and Evaluation **11**

2.4 Skills 15

Chapter 3 ..... **17**

Analysis & design **17**

Chapter 3 : Analysis & design **18**

3.1 Requirements Specification 18

3.1.1 Functional Requirements :- 18

3.1.2 Nonfunctional Requirements :- 18

3.2 Data Requirements :- 18

3.3 Hardware :- 19

3.4 Software :- 19

3.5 Activities and Actions 20

3.5.1 Use case diagram :- 20

3.5.2 Flow chart :- 21

3.5.3 Class diagram :- 22

Chapter 4 : Implementation **24**

4.1 The coding is divided into 5 parts 24

4.2 Maintenance 24

4.3 Interface 25

Chapter 5 ..... **26**

Testing **26**

Chapter 5 : Testing **27**

5.1 Some of the test cases kind 1 (simulation) : 27

5.2 Calibrate the sensors 30

5.3 The Results 31

Chapter 6 32

Conclusion 32

Chapter 6 : Conclusion 33

6.1 List of references :- 35

Appendix A 36

Matlab code 36

FIRST CLASS C4 CLASS: 37

SECOND CLASS chec class: 44

THERD CLASS Update class: 44

FOURTH ClASS Next play class: 46

FIFTH CLASS Win class: 52

SIXTH CLASS Sensors class: 53

SEVENTH CLASS Move class 58

Arduino Code 62

Appendix B 69

Algorithm explanation 69

Appendix C 74

Photos and Videos 74

Chapter 1

Introduction

Chapter 1 : Introduction

The title of the project is Intelligent robot player. It means that Build a robot that can play Connect4 game with an intelligent thinking similar to human . As we know there are a lot of AI algorithm the support all kinds of thinking in game. For example: alpha beta , min max and fuzzy logic algorithm . With the help of these algorithm and our own algorithm we will achieve our goal. Our project will be physical human against robot. Our project is related to the family of CS because it is working with AI. The moving objects needs to a lot of hardwires .

Additionally, the project is touching a lot of technologies PC, Robot, sensors , programming .

Connect4 game has a lot scenarios to win . These scenarios are too difficult for the human to save. by using the CPU These scenarios will be saved and handled

1.1 Motivation

The motivation for the project is learning new things , and since there are so many huge technologies ,by the AI algorithm we can handle a robotic.

1.2 Consideration

In this project we will consider the objects will be able to receiving the player steps by technologies such as (Sensors ,Arduino , … etc.).

1.3 Project Proposal

1.3.1 Problem definition

Build an intelligent robot agent that can implement the artificial intelligent games and searching algorithms on it to study the efficiency of that algorithm.

1.4 Project scope

The scope of this project is:

Use a robot kit , microprocessor controller and PC to implement the algorithm and the movement of the robot. All these should be integrated in one system to achieve the expected goal .

1.5 Project type

The type of the project is AI.

1.6 The outcome

A Robot player which can play against human**,** And it will be a good competitor.

1.7 Target Users

- computer science researcher to implement other algorithm on the system.
- regular game player.

1.8 Suggested solution

- Choose a proper algorithm like minimax, fuzzy and reinforcement learning algorithms or create our own algorithm.
- Design the robot to fit with connect 4 game module.
- Using IR sensors to read the game input.
- Using Screen to show what the robot do right now and also to let the user to interact with it.

1.9 Project Management

1.9.1 Work Breakdown Structure (WBC)

Work Breakdown Structure is one of the best ways to break a large or difficult task into more smaller, more manageable subtasks. So, Work Breakdown Structure helps team members to know what tasks that must be done to complete the project and to understand how all of the tasks fit into the overall design project. In summary, Work Breakdown Structure defines and groups a project's discrete work elements in a way that help us organize and define the total work scope of the project . WBC shown in figure

Figure 1.1 shows you “The work break down structure for the project”

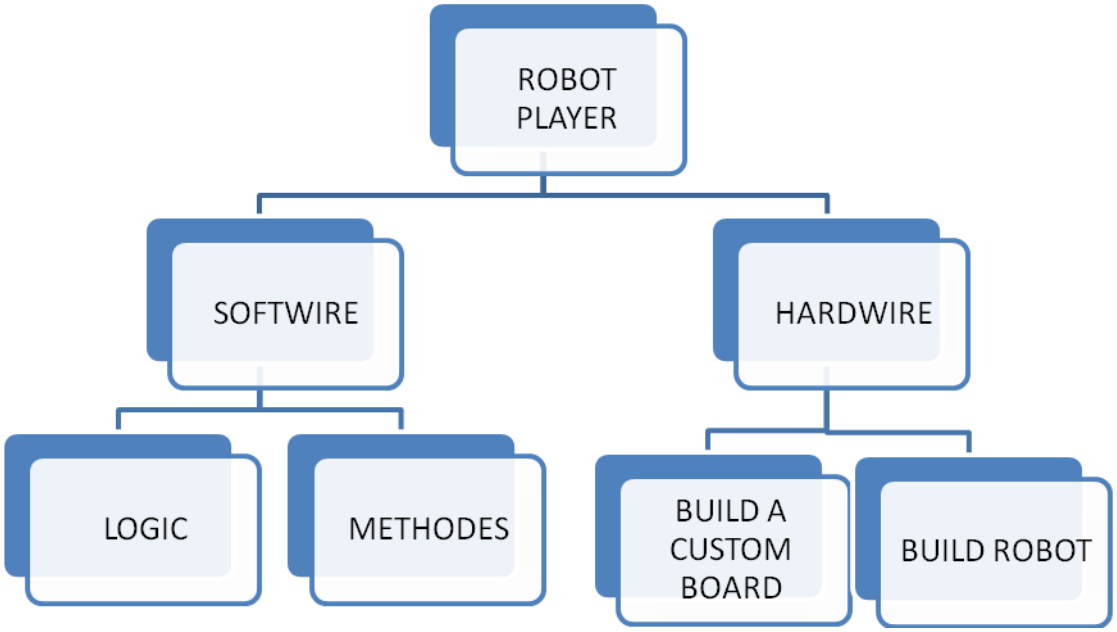


Figure 1.1 - WBS

1.9.2 Time Estimates

Figure 1.2 shows you "The time estimates schedule for the project"

No.	Task Name	Time
	First Semester	14 weeks
1	Understand the project scoop	2 weeks
2	Research	2 weeks
3	Analysis	4 weeks
4	Initial design	4 weeks
5	Writing reports	2 weeks
	Second Semester	14 weeks
6	Design	2 weeks
7	Implementation	6 weeks
8	Testing	3 week
9	Deployment	2 weeks
10	Writing reports	1 weeks
	Total of time	28 weeks

Figure 1.2 - Time Estimates

Chapter 2

Literature Review

Chapter 2 : Literature Search and Evaluation

do three literature search and evaluation

Item	Criteria	Ref [1]	Ref [2]	Ref [3]
Presentation				
1	Language and layout and structure is clear or difficult to read.	Clear	Clear	Clear
2	Are the diagrams or figures understandable.	Yes	Yes	Yes
3	Do you need additional information, Knowledge and or skills to make sense of it.	No	No	No
Relevance				
1	Does information match your needs	Yes	Yes	Yes
2	What is it About	his paper, are study applying Reinforcement Learning to design	This paper looks at implementing artificial intelligence, AI,	this paper, we examine sev- eral ways to improve

		a automatic agent to play the game Super Mario Bros. One of the challenge is how to handle the complex game environment. By abstracting the game environment into a state vector and using Q learning	into an existing Connect 4 game. The AI for this game is based on influence mapping	performance of Minimax and Alpha- Beta search in the game of Connect- k , a generalized version of the more famous Connect-4 game
3	What use can you make of it in your project.	Using Ai algorithm for playing	Using Ai algorithm for playing	Using Ai algorithm for playing
4	Is this the best source of the information you need.	Yes	Yes	Yes

Objectivity				
1	Is the authors position of interest made clear.	Yes	Yes	Yes
2	Do the writers state their position on the issue.	Yes	Yes	Yes
3	Are there vested Interests.	Yes	Yes	Yes
4	Dose the paper represent an independent or corporate view.	corporate	independent	independent
5	Can you trust what it says how do you know.	Yes , from the source	Yes , from the source	Yes , from the source

Method				
1	Is it clear how the data collected	Yes	Yes	Yes
2	Can you identify the authors or organizations.	Standford university , Yizheng Liao and Kun Yi google Inc ,Zhe Yang	University of Michigan Dr. Adnan Shaout	UC Irvine, California Jenny Lam
3	Is it from a reputable source.	Yes	Yes	Yes
4	How do you know.	CV	CV	CV

Timeliness				
1	Is it clear where the information was produced.	Yes	Yes	Yes
2	Dose the date of the Of the information meet your requirements.	Yes	Yes	Yes
3	Is it up to date	Yes	Yes	Yes
4	When was it published or the website reviewed .	2012	2007	Not mention
5	Has it been recently revised.	Yes	Yes	Yes
6	Is it obsolete	No	No	No

3. Methods to find information :

	Www.google.com	Www.ieee.org
Advantages	-search all types (papers,books,video…).	- Search paper and Books.
Disadvantages	- All areas. - All users	- In science - Need an Account
Library	- Search project. - Manual search.	- Avatar - Especially for Faculty members.

2.4 Skills

The main skills needed are :-

- Programming :- to manage the system and implement the AI algorithm using Matlab.
- Robot handling :- The system needs mobility and a robot was chosen for the task.
- Basic electric :- to connect the sensors and make the wiring for the system

The most important skills are :-

- Programming: The core of the project is in the coding so this skill is fundamental.
- Robot handling :- The second core of the project is in the robot handling so this skill is also fundamental.

Chapter 3

Analysis & design

Chapter 3 : Analysis & design

3.1 Requirements Specification

3.1.1 Functional Requirements :-

- Robot mobility.
- Streaming data from sensors to MatLab.
- Identify the playing round and where the coins are.
- Check The game rules.
- Execute AI algorithm to determine where to place the new coin.

3.1.2 Nonfunctional Requirements :-

- Graphical user interface.
- Collect data from sensors.

3.2 Data Requirements :-

Each step which played by the human player.

3.3 Hardware :-

- Lego NXT EV3 MindStorm robot.
- Arduino.
- IR Sensors.

.

3.4 Software :-

- Lego NXT IDE.
- Arduino IDE.
- Matlab.

3.5 Activities and Actions

3.5.1 Use case diagram :-

Table 3.1 shows you "The use case diagram which shows you what kind of query the user can choose from"

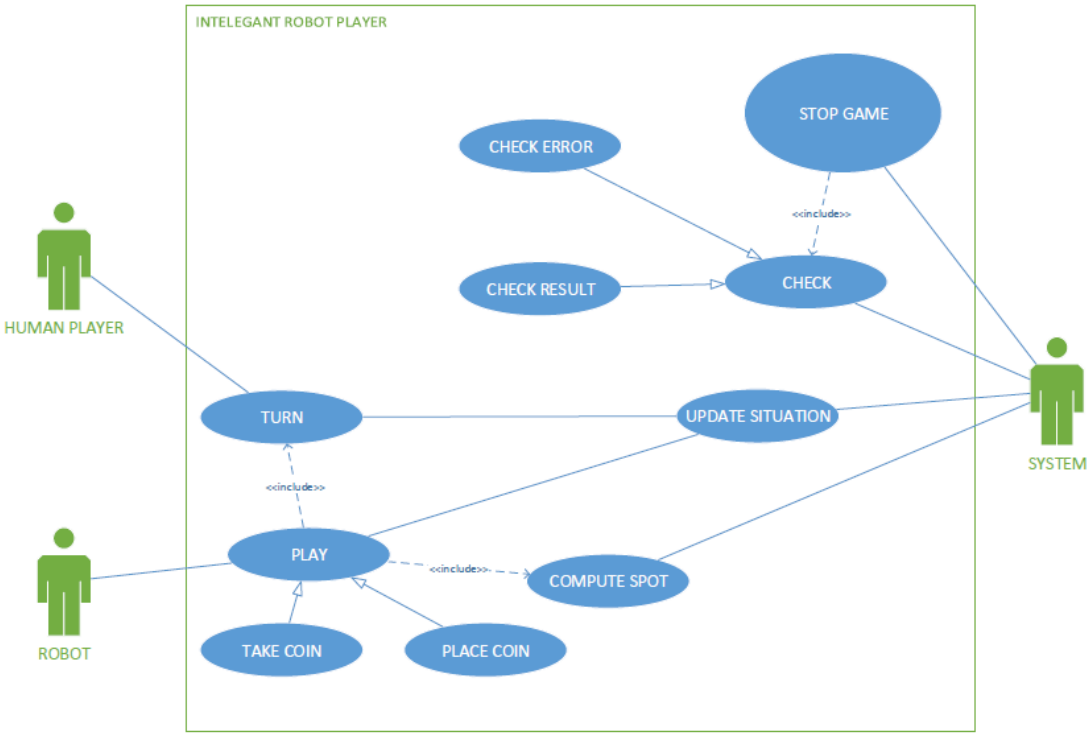


Table 3.1 - Use case diagram

3.5.2 Flow chart :-

Table 3.2 shows you “The flow chart which explains to you how the system will work”

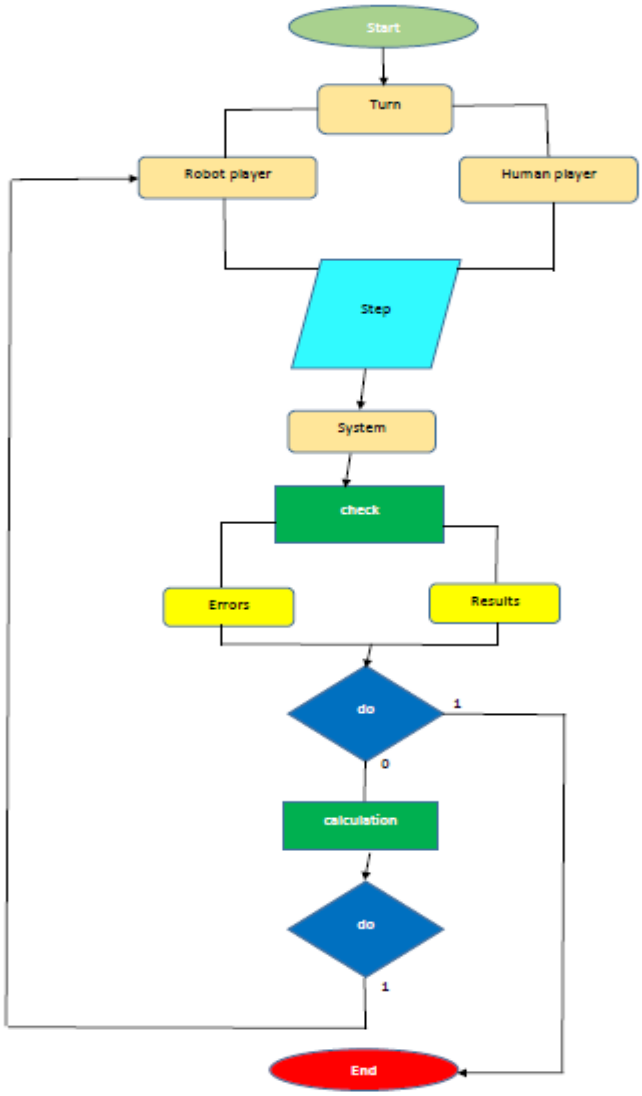
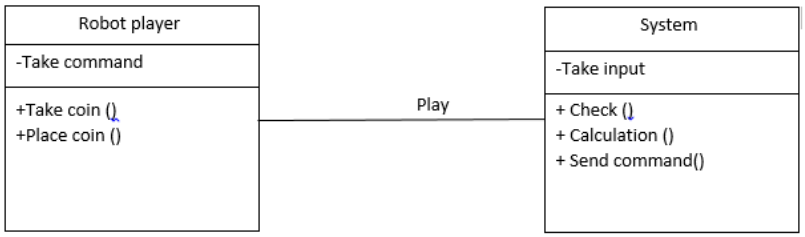


Table 3.2 - Flow chart

3.5.3 Class diagram :-

Table 3.3 shows you "Class diagram which shows to you the role of the server and the role of the client"

Table 3.3 - Class diagram



ER (Entity Relationship) diagram :-

Our project does not need a database , and according to our supervisor we just need to store real-time data in a data structure (Arrays) .

Tool (software) used for design :-

Microsoft Visio .

Chapter 4

Implementation

Chapter 4 : Implementation

4.1 The coding is divided into 5 parts :

we started coding the project . and the coding is divided into 5 parts :

1. Write game rules checking methods.
2. Write method for taking input from sensors.
3. Write methods to save the input data.
4. Write the logic (the brain of the robot).
5. Write method to send the output to the robot.

4.2 Maintenance

We maintained the program by making a lot of tests in order to remove the errors.

4.3 Interface

```
want to start? [y=1, n=0] 0
     1     0     0     0     0     0     0     0
     2     0     0     0     0     0     0     0
     3     0     0     0     0     0     0     0
     4     0     0     0     0     0     0     0
     5     0     0     0     0     0     0     0
     6     0     0     0     2     0     0     0
     0     1     2     3     4     5     6     7

your play? [x]
```

```
your play? [x] 6
your play? [y] 6
     1     0     0     0     0     0     0     0
     2     0     0     0     0     0     0     0
     3     0     0     0     0     0     0     0
     4     0     0     0     0     0     0     0
     5     0     0     0     2     0     0     0
     6     0     0     1     2     2     1     0
     0     1     2     3     4     5     6     7

your play? [x]
```

Chapter 5

Testing

Chapter 5 : Testing

We did brake the test unit to 2 kinds of test , the first kind is testing the game it self by programming a simulation. The second kind is testing the robot and taking input from the sensors .

5.1 Some of the test cases kind 1 (simulation) :

```
want to start? [y=1, n=0] 0
     1     0     0     0     0     0     0     0
     2     0     0     0     0     0     0     0
     3     0     0     0     0     0     0     0
     4     0     0     0     0     0     0     0
     5     0     0     0     0     0     0     0
     6     0     0     0     2     0     0     0
     0     1     2     3     4     5     6     7

your play? [x]
```

```
your play? [x] 3
your play? [y] 6
     1     0     0     0     0     0     0     0
     2     0     0     0     0     0     0     0
     3     0     0     0     0     0     0     0
     4     0     0     0     0     0     0     0
     5     0     0     0     0     0     0     0
     6     0     0     1     2     2     0     0
     0     1     2     3     4     5     6     7

your play? [x]
```

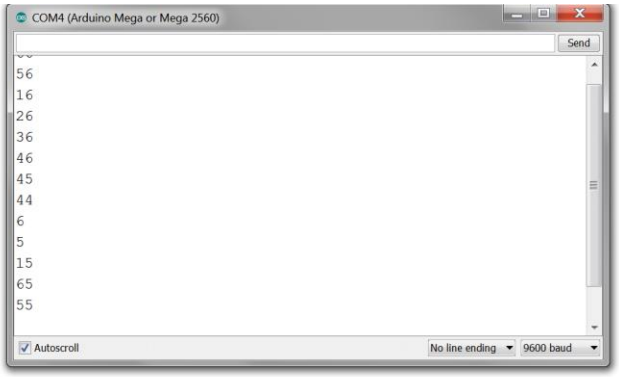
```
your play? [x] 6
your play? [y] 6
     1      0      0      0      0      0      0      0
     2      0      0      0      0      0      0      0
     3      0      0      0      0      0      0      0
     4      0      0      0      0      0      0      0
     5      0      0      0      2      0      0      0
     6      0      0      1      2      2      1      0
     0      1      2      3      4      5      6      7

your play? [x]
```

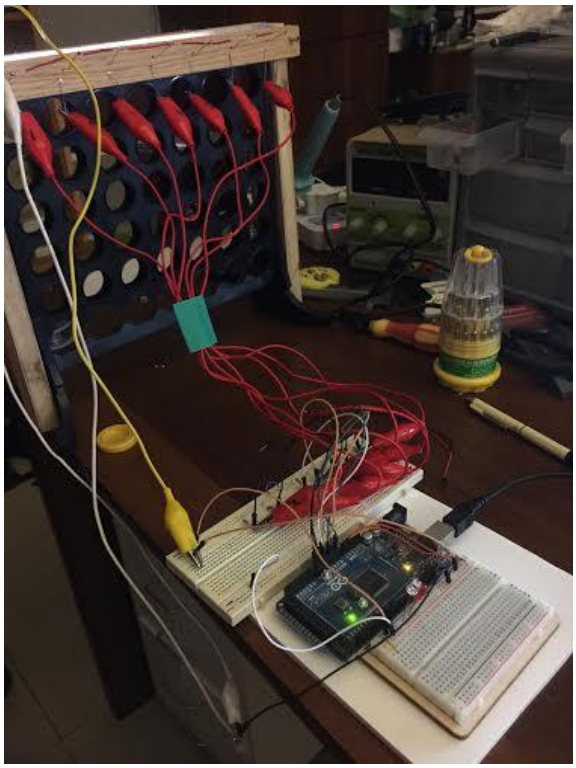
```
your play? [x] 5
your play? [y] 4
Your play is not admissible
Please choose another play
your play? [x] 5
your play? [y] 5
     1      0      0      0      0      0      0      0
     2      0      0      0      0      0      0      0
     3      0      0      0      0      0      0      0
     4      0      0      0      2      0      0      0
     5      0      0      0      2      1      0      0
     6      0      0      1      2      2      1      0
     0      1      2      3      4      5      6      7

your play? [x]
```

```
your play? [x] 5
your play? [y] 4
     1      0      0      0      0      0      0      0
     2      0      0      0      0      0      0      0
     3      0      0      0      2      0      0      0
     4      0      0      0      2      1      0      0
     5      0      0      0      2      1      0      0
     6      0      0      1      2      2      1      0
     0      1      2      3      4      5      6      7

     1      0      0      0      0      0      0      0
     2      0      0      0      0      0      0      0
     3      0      0      0      2      0      0      0
     4      0      0      0      2      1      0      0
     5      0      0      0      2      1      0      0
     6      0      0      1      2      2      1      0
     0      1      2      3      4      5      6      7

You lost
>>
```

5.2 Calibrate the sensors

```
  int sensor1 = A1;
  int sensor2 = A2;
  int sensor3 = A3;
  int sensor4 = A4;
  int sensor5 = A5;
  int sensor6 = A6;
  int sensor7 = A7;
  int count1 = 7;
  int count2 = 17;
  int count3 = 27;
  int count4 = 37;
  int count5 = 47;
  int count6 = 57;
  int count7= 67;
  int red = 2;
  int green = 4;


void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
    pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);

}

void loop() {
  // read value from sensor1
    digitalWrite(red , LOW);
  digitalWrite(green , HIGH);
  if(analogRead(sensor1)<=80){

   digitalWrite(green , LOW);
```





5.3 The Results

Test	Note
Checking methods	Worked well
Compute next play	Strongly worked.
Sensors	Worked well
Robot	Worked well

5.4 Conclusion for testing phase

The results were good ,reasonable and as expected.

Chapter 6

Conclusion

Chapter 6 : Conclusion

In the end , the outcomes of the project are :

- Robot player plays and thinks like a human.

List of References

6.1 List of references :-

1. http://en.wikipedia.org/wiki/Robot
2. http://en.wikipedia.org/wiki/Connect_Four
3. http://www.generation5.org/content/2000/boardai.asp
4. http://www.pomakis.com/c4/connect_generic/c4.txt
5. N. P. Padhy, "State Space Search: Implementation and Applications," in Artificial Intelligence and Intelligent Systems, New York: Oxford Univ. Press,2005, ch. 4, sec. 4.10, pp. 161-171

6. http://en.wikipedia.org/wiki/Minimax
7. arduino.cc/en/Reference/HomePage
8. https://legoev3.codeplex.com/documentation
9. www.raspberrypi.org/documentation/hardware/raspberrypi
10. V. Allis. A knowledge-based approach of Connect-4. The game is solved: White wins. Master's thesis, Department of Mathematics and Computer Science, Vrije Universiteit, Amsterdam, The Netherlands, 1988.
11. http://www.gm.fh-koeln.de/~konen/Publikationen/ppsn2012_RL-CFour.pdf
12. https://www.csustan.edu/sites/default/files/honors/documents/journals/Stirrings/Darmousseh.pdf
13. http://www-personal.engin.umd.umich.edu/~shaout/connect4.pdf

Appendix A

Part 1

Matlab code

Part 1 : Code

The first part contains 5 classes:

FIRST CLASS C4 CLASS:

```
close all

clear

clc

%% Initialize

global pp pm winn sensorX sensorY


  global aa


% possible play

pp=[6 6 6 6 6 6 6;1 2 3 4 5 6 7];


% playing matrix

pm=zeros(6,7);


% Displaying matrix

dm=[[[1;2;3;4;5;6] pm];[0 1 2 3 4 5 6 7]];
```

```
% win table
r1=[6 6 6 6 6 6 6;1 2 3 4 5 6 7;0 0 0 0 0 0 0];
r2=[5 5 5 5 5 5 5;1 2 3 4 5 6 7;0 0 0 0 0 0 0];
r3=[4 4 4 4 4 4 4;1 2 3 4 5 6 7;0 0 0 0 0 0 0];
r4=[3 3 3 3 3 3 3;1 2 3 4 5 6 7;0 0 0 0 0 0 0];
r5=[2 2 2 2 2 2 2;1 2 3 4 5 6 7;0 0 0 0 0 0 0];
r6=[1 1 1 1 1 1 1;1 2 3 4 5 6 7;0 0 0 0 0 0 0];

r7=[6 5 4 3 2 1;1 1 1 1 1 1;0 0 0 0 0 0];
r8=[6 5 4 3 2 1;2 2 2 2 2 2;0 0 0 0 0 0];
r9=[6 5 4 3 2 1;3 3 3 3 3 3;0 0 0 0 0 0];
r10=[6 5 4 3 2 1;4 4 4 4 4 4;0 0 0 0 0 0];
r11=[6 5 4 3 2 1;5 5 5 5 5 5;0 0 0 0 0 0];
r12=[6 5 4 3 2 1;6 6 6 6 6 6;0 0 0 0 0 0];
r13=[6 5 4 3 2 1;7 7 7 7 7 7;0 0 0 0 0 0];

r14=[6 5 4 3;4 3 2 1;0 0 0 0];
r15=[6 5 4 3 2;5 4 3 2 1;0 0 0 0 0];
r16=[6 5 4 3 2 1;6 5 4 3 2 1;0 0 0 0 0 0];
r17=[6 5 4 3 2 1;7 6 5 4 3 2;0 0 0 0 0 0];
```

```
r18=[5 4 3 2 1;7 6 5 4 3;0 0 0 0 0];
r19=[4 3 2 1;7 6 5 4;0 0 0 0];

r20=[6 5 4 3;4 5 6 7;0 0 0 0];
r21=[6 5 4 3 2;3 4 5 6 7;0 0 0 0 0];
r22=[6 5 4 3 2 1;2 3 4 5 6 7;0 0 0 0 0 0];
r23=[6 5 4 3 2 1;1 2 3 4 5 6;0 0 0 0 0 0];
r24=[5 4 3 2 1;1 2 3 4 5;0 0 0 0 0];
r25=[4 3 2 1;1 2 3 4;0 0 0 0];

sp=[0;0;3]; %Separator

win1=[r1 sp r2 sp r3 sp r4 sp r5 sp r6];
win2=[r7 sp r8 sp r9 sp r10 sp r11 sp r12 sp r13];
win3=[r14 sp r15 sp r16 sp r17 sp r18 sp r19];
win4=[r20 sp r21 sp r22 sp r23 sp r24 sp r25];
winn=[sp win1 sp win2 sp win3 sp win4 sp];


%% Start play
```

```
aa = serial('COM4');
fopen(aa);
a=input('want to start? [y=1, n=0] ');
ch=[]; % check if the play is admissible
if a==1
    while true
        disp(dm)
        sensors()
        j=sensorX;
        i=sensorY;

       % j=input('your play? [x] ');
      %  i=input('your play? [y] ');
        ch=chec(i,j);
        if ch==0
            disp('Your play is not admissible')
            disp('Please choose another play')
        else
            break;
        end
```

```
    end
    updat(i,j,1);
    dm=[[[1;2;3;4;5;6] pm];[0 1 2 3 4 5 6 7]];
    a=2; % computer play
else
    % my play
    i=6;
    j=4;
   % move(j)

    updat(i,j,2);
    dm=[[[1;2;3;4;5;6] pm];[0 1 2 3 4 5 6 7]];
    a=1; % enemy play
end
disp(dm)

w=0; % win flag
while w==0 && i>0 && j>0
    w=win();
        if w==1
```

```
    disp(dm)
    disp('You won')
    break;
elseif w==2
    disp(dm)
    disp('You lost')
    break;
else
    if a==1 % enemy play
        while true

            sensors()
    j=sensorX;
    i=sensorY;
            ch=chec(i,j);
            if ch==0
                disp('Your play is not admissible')
                disp('Please choose another play')
            else
                break;
```

```
            end
        end
        updat(i,j,1);
        dm=[[[1;2;3;4;5;6] pm];[0 1 2 3 4 5 6 7]];
        a=2; % computer play
    else % computer play
        [i,j]=next_play();
       % move(j)

        updat(i,j,2);
        dm=[[[1;2;3;4;5;6] pm];[0 1 2 3 4 5 6 7]];
        a=1; % enemy play
        disp(dm)

    end
  end
end
fclose(aa);
```

SECOND CLASS chec class:

```
function ch=chec(i,j)

global pp

ch=not(isempty(find(and(pp(1,:)==i,pp(2,:)==j)==1)));
```

THERD CLASS Update class:

```
function updat(i,j,p)

global pp pm winn

% p=1 enemy
% p=2 player


% update pp
```

```
m= and(pp(1,:)==i,pp(2,:)==j)==1;
if i-1<0
    pp(1,m)=0;
else
    pp(1,m)=i-1;
end

% update pm for displaying
% and win table
m=and(winn(1,:)==i,winn(2,:)==j)==1;
if p==1
    winn(3,m)=1;
    pm(i,j)=1;
else
    winn(3,m)=2;
    pm(i,j)=2;
end
```

FOURTH ClASS Next play class:

```
function [i,j]=next_play()

global pp winn

% Find if computer win
b=[2 2 2];
c=[];
for k=1:length(winn)-length(b)+1
   if length(find(winn(3,k:k+length(b)-1)==b))>length(b)-1
      c=[c;k];
   end
end
if ~isempty(c)
   for k=1:length(c)
      if winn(3,c(k)-1)==0 && chec(winn(1,c(k)-1),winn(2,c(k)-1))==1
            i=winn(1,c(k)-1);
            j=winn(2,c(k)-1);
            return;
```

```
        elseif winn(3,c(k)+3)==0 &&
chec(winn(1,c(k)+3),winn(2,c(k)+3))==1
                i=winn(1,c(k)+3);
                j=winn(2,c(k)+3);
                return;
        end
    end
end

% Find if enemy win
b=[1 1 1];
c=[];
for k=1:length(winn)-length(b)+1
    if length(find(winn(3,k:k+length(b)-1)==b))>length(b)-1
        c=[c;k];
    end
end
if ~isempty(c)
    for k=1:length(c)
        if winn(3,c(k)-1)==0 && chec(winn(1,c(k)-1),winn(2,c(k)-1))==1
                i=winn(1,c(k)-1);
```

```
            j=winn(2,c(k)-1);
            return;
        elseif winn(3,c(k)+3)==0 &&
chec(winn(1,c(k)+3),winn(2,c(k)+3))==1
            i=winn(1,c(k)+3);
            j=winn(2,c(k)+3);
            return;
        end
    end
end

% Find if computer has [2 2]
b=[2 2];
c=[];
for k=1:length(winn)-length(b)+1
    if length(find(winn(3,k:k+length(b)-1)==b))>length(b)-1
        c=[c;k];
    end
end
if ~isempty(c)
    for k=1:length(c)
```

```
        if winn(3,c(k)-1)==0 && chec(winn(1,c(k)-1),winn(2,c(k)-1))==1
                i=winn(1,c(k)-1);
                j=winn(2,c(k)-1);
                return;
        elseif winn(3,c(k)+2)==0 &&
chec(winn(1,c(k)+2),winn(2,c(k)+2))==1
                i=winn(1,c(k)+2);
                j=winn(2,c(k)+2);
                return;
        end
    end
end

% Find if enemy has [1 1]
b=[1 1];
c=[];
for k=1:length(winn)-length(b)+1
    if length(find(winn(3,k:k+length(b)-1)==b))>length(b)-1
        c=[c;k];
    end
end
```

```
if ~isempty(c)
   for k=1:length(c)
      if winn(3,c(k)-1)==0 && chec(winn(1,c(k)-1),winn(2,c(k)-1))==1
            i=winn(1,c(k)-1);
            j=winn(2,c(k)-1);
            return;
      elseif winn(3,c(k)+2)==0 &&
chec(winn(1,c(k)+2),winn(2,c(k)+2))==1
            i=winn(1,c(k)+2);
            j=winn(2,c(k)+2);
            return;
      end
   end
end

% Find if computer has [2]
b=2;
c=[];
for k=1:length(winn)-length(b)+1
   if length(find(winn(3,k:k+length(b)-1)==b))>length(b)-1
      c=[c;k];
```

```
    end
end
if ~isempty(c)
    for k=1:length(c)
        if winn(3,c(k)-1)==0 && chec(winn(1,c(k)-1),winn(2,c(k)-1))==1
                i=winn(1,c(k)-1);
                j=winn(2,c(k)-1);
                return;
        elseif winn(3,c(k)+1)==0 &&
chec(winn(1,c(k)+1),winn(2,c(k)+1))==1
                i=winn(1,c(k)+1);
                j=winn(2,c(k)+1);
                return;
        end
    end
end

% Otherwise (any play)
k=randi(7,1);
i=pp(1,k);
j=pp(2,k);
```

FIFTH CLASS Win class:

```
function w=win()

global winn

w=0;

% Find if enemy win
b=[1 1 1 1];
c=[];
for k=1:length(winn)-length(b)+1
    if length(find(winn(3,k:k+length(b)-1)==b))>length(b)-1
        c=[c;k];
    end
end
if isempty(c)
    % Find if player win
```

```
    b=[2 2 2 2];
    c=[];
    for k=1:length(winn)-length(b)+1
        if length(find(winn(3,k:k+length(b)-1)==b))>length(b)-1
            c=[c;k];
        end
    end
    if ~isempty(c)
        w=2; % player win
    end
else
    w=1; % enemy win
end
```

SIXTH CLASS Sensors class:

```
function sensors % arduino comuncation
global sensorX  sensorY aa ;


  sensorX = 0;
```

```
  sensorY = 0 ;

%initilaize serial port for comuncation with arduino
%aa = serial('COM3');

% open serial port
%fopen(aa);

% read from serial port
sensor= fscanf(aa ,'%d');

%check witch sensor is reading the data

   if sensor>0 | sensor<8

         sensorY = sensor;
         sensorX =1;
         return

   end
```

```
if sensor>10 | sensor<18

        sensorY = sensor - 10;
        sensorX =2;
        return
end

if sensor>20 | sensor<28

        sensorY = sensor - 20;
        sensorX =3;
        return

end

if sensor>20 | sensor<28

        sensorY = sensor - 20;
        sensorX =3;
```

```
        return
end


if sensor>30 | sensor<38

        sensorY = sensor - 30;
        sensorX =4;
        return
end


if sensor>40 | sensor<48

        sensorY = sensor - 40;
        sensorX =5;
        return
end

if sensor>50 | sensor<58
```

```
        sensorY = sensor - 50;
        sensorX =6;
        return
    end

    if sensor>60 | sensor<68

        sensorY = sensor - 60;
        sensorX =7;
        return

    end

% close serial port
%fclose(aa);
```

SEVENTH CLASS Move class

```
function move_test1(j)

robot = legoev3('usb');

mymotorL = motor(robot,'C');
mymotorR = motor(robot,'B');
kick = motor(robot,'A');
mycolorsensor = colorSensor(robot);
resetRotation(mymotorL)
count = j;

while (count ~= 0)

    while ~strcmp(readColor(mycolorsensor),'black')

    mymotorL.Speed = 50;
    mymotorR.Speed = 50;
    start(mymotorL)
    start(mymotorR)
```

```
end

      stop(mymotorL,1)
stop(mymotorR,1)

while ~strcmp(readColor(mycolorsensor),'brown')
   if count == 0
      break;
   end
mymotorL.Speed = 50;
mymotorR.Speed = 50;
start(mymotorL)
start(mymotorR)
```

```
        end

            stop(mymotorL,1)
        stop(mymotorR,1)

        count = count-1;


end


 resetRotation(kick)
 pause(1);
while ~(readRotation(kick)<=-100)
            kick.Speed = -75;
```

```
        start(kick)
end
        stop(kick)
         resetRotation(kick)
       while ~(readRotation(kick)>=100)
            kick.Speed = 75;

        start(kick)

        end
        stop(kick)
        resetRotation(kick)
pause(5);

while true
        if ~strcmp(readColor(mycolorsensor),'white')
        mymotorL.Speed = -50;
        mymotorR.Speed = -50;
        start(mymotorL)
        start(mymotorR)
```

```
        else
        stop(mymotorL,1)
        stop(mymotorR,1)
           break;
        end
end
```

PART 2

Arduino Code

```
  int sensor1 = A1;
  int sensor2 = A2;
  int sensor3 = A3;
  int sensor4 = A4;
  int sensor5 = A5;
  int sensor6 = A6;
  int sensor7 = A7;
```

```
  int count1 = 7;
  int count2 = 17;
  int count3 = 27;
  int count4 = 37;
  int count5 = 47;
  int count6 = 57;
  int count7= 67;
  int red = 2;
  int green = 4;


void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
    pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);

}

void loop() {
```

```
// read value from sensor1
  digitalWrite(red , LOW);
digitalWrite(green , HIGH);
if(analogRead(sensor1)<=80){

 digitalWrite(green , LOW);
 digitalWrite(red , HIGH);

  count1 = count1-1;
Serial.print(count1);
Serial.print("\n");
delay(1000);
}

// read value from sensor2
if(analogRead(sensor2)<=80){

  digitalWrite(green , LOW);
  digitalWrite(red , HIGH);
  count2 = count2-1;
```

```
Serial.print(count2);
Serial.print("\n");
delay(1000);
}

// read value from sensor3
if(analogRead(sensor3)<=80){
  digitalWrite(green , LOW);
  digitalWrite(red , HIGH);

  count3 = count3-1;
Serial.print(count3);
Serial.print("\n");
delay(1000);
}

// read value from sensor4
if(analogRead(sensor4)<=80){
  digitalWrite(green , LOW);
  digitalWrite(red , HIGH);
```

```
  count4 = count4-1;
Serial.print(count4);
Serial.print("\n");
delay(1000);
}

  // read value from sensor5
if(analogRead(sensor5)<=80){
  digitalWrite(green , LOW);
  digitalWrite(red , HIGH);

  count5 = count5-1;
Serial.print(count5);
Serial.print("\n");
delay(1000);
}

  // read value from sensor6
if(analogRead(sensor6)<=80){
```

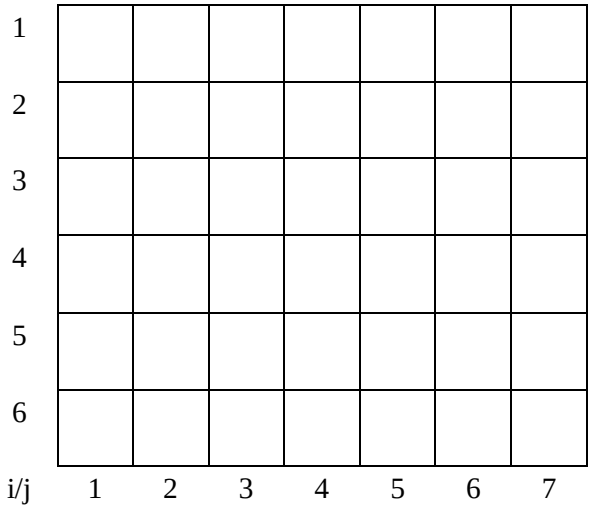
```
  digitalWrite(green , LOW);
  digitalWrite(red , HIGH);

  count6 = count6-1;
Serial.print(count6);
Serial.print("\n");
delay(1000);
}

  // read value from sensor7
if(analogRead(sensor7)<=80){

  digitalWrite(green , LOW);
  digitalWrite(red , HIGH);

  count7 = count7-1;
Serial.print(count7);
Serial.print("\n");
delay(1000);
```

```
  }
}
```

Appendix B

Algorithm explanation

Connect-Four Algorithm



Possible play

initial	6-1	6-2	6-3	6-4	6-5	6-6	6-7

After each play (i,j), replace (i,j) by (i-1,j). For example, if a play is (6,5), the possible play is

current	6-1	6-2	6-3	6-4	5-5	6-6	6-7

Possible win: four successive in any row in the following matrix (two copies, for player and for enemy).

	4 Horizontals						
1	6-1	6-2	6-3	6-4	6-5	6-6	6-7
2	5-1	5-2	5-3	5-4	5-5	5-6	5-7
3	4-1	4-2	4-3	4-4	4-5	4-6	4-7
4	3-1	3-2	3-3	3-4	3-5	3-6	3-7
5	2-1	2-2	2-3	2-4	2-5	2-6	2-7
6	1-1	1-2	1-3	1-4	1-5	1-6	1-7
	4 Verticals						
7	6-1	5-1	4-1	3-1	2-1	1-1	
8	6-2	5-2	4-2	3-2	2-2	1-2	
9	6-3	5-3	4-3	3-3	2-3	1-3	
10	6-4	5-4	4-4	3-4	2-4	1-4	
11	6-5	5-5	4-5	3-5	2-5	1-5	
12	6-6	5-6	4-6	3-6	2-6	1-6	
13	6-7	5-7	4-7	3-7	2-7	1-7	

	4 Left Diagonals						
14	6-4	5-3	4-2	3-1			
15	6-5	5-4	4-3	3-2	2-1		
16	6-6	5-5	4-4	3-3	2-2	1-1	
17	6-7	5-6	4-5	3-4	2-3	1-2	
18	5-7	4-6	3-5	2-4	1-3		
19	4-7	3-6	2-5	1-4			
	4 Right Diagonals						
20	6-4	5-5	4-6	3-7			
21	6-3	5-4	4-5	3-6	2-7		
22	6-2	5-3	4-4	3-5	2-6	1-7	
23	6-1	5-2	4-3	3-4	2-5	1-6	
24	5-1	4-2	3-3	2-4	1-5		
25	4-1	3-2	2-3	1-4			

- Initialize a matrix, winn, with size (3×170). In the first row put the first index from above table. In the second row put the second index from above table. In the third row put zeros elements except in separators put value 3 (with index (0,0) for the separator).

Algorithm:

1- If player starts or (6-4) is free: play (6-4), otherwise (the enemy played (6-4)), play (6-5).
2- **Update** ***play, win*** tables.
3- **Check** ***win*** tables for possible winner.
4- If yes stop, otherwise go to step 5.
5- **Find** the ***play*** which makes ***player wins*** (using ***play, win*** tables), play it and go to step 2.
6- Otherwise, **find** the ***play*** which makes ***enemy wins*** (using ***play, win*** tables).
7- If found, play it, go to step 2. Otherwise, go to step 8.
8- **Find the best play**, play it and go to step 2. (No play will make anyone wins).

Given (i,j) the current play of player 1 or 2.

Update *play*:

- Search (i,j) in play table, replace it by (i-1,j).

Update *winn*:

- Search (i,j) in winn table (i in first row and j in second row).
- Replace the winn (3, found index) by 1 or 2 according to player.

Check *winn table* for winner:

- Search the ***pattern 1111*** in winn table for enemy win.
- Search the ***pattern 2222*** in winn table for computer win.

Find the ***play*** using ***play, winn*** tables:

- Search the ***pattern 222*** in win table, if found search possible play to win (make computer win).
- Otherwise, Search the ***pattern 111*** in win table, if found search possible play to win (make enemy loose).
- Otherwise, Search the ***pattern 22*** in win table, if found search possible play.
- Otherwise, Search the ***pattern 11*** in win table, if found search possible play.
- Otherwise, Search the ***pattern 2*** in win table, if found search possible play.
- Otherwise, play randomly any possible play.

Find the best play:

- Search the ***pattern 11*** in win table.
- If found
    - Search possible play in same segment.
    - If found play it.
    - Otherwise, play any possible play.
- Otherwise, Search the ***pattern 1*** in win table
    - Search possible play in same segment.
    - If found play it.
    - Otherwise, play any possible play.

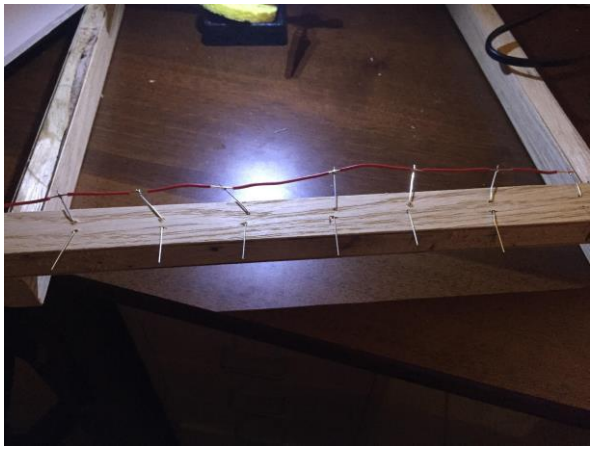
Appendix C

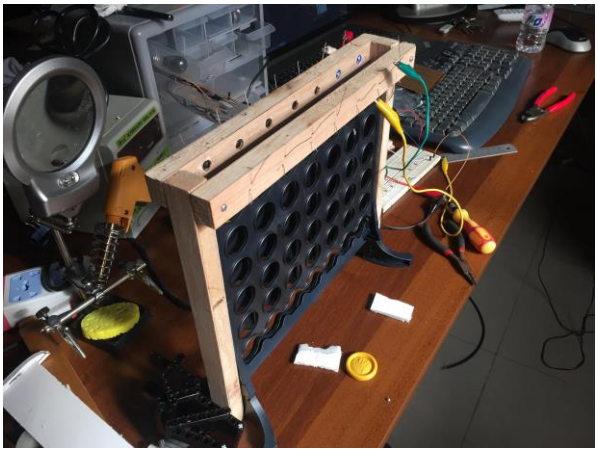
Photos and Videos

Videos :-

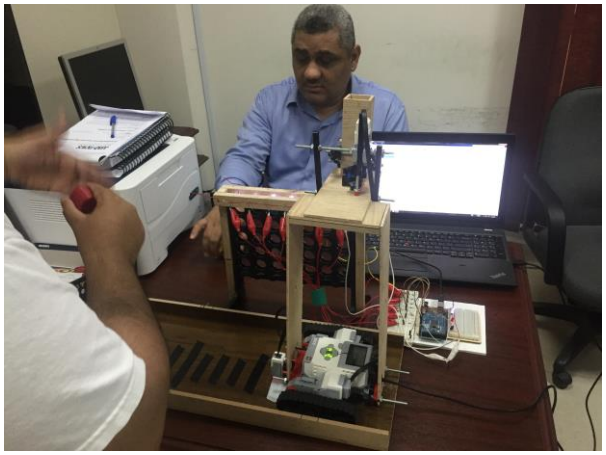
Attached in the project CD

Photos:-

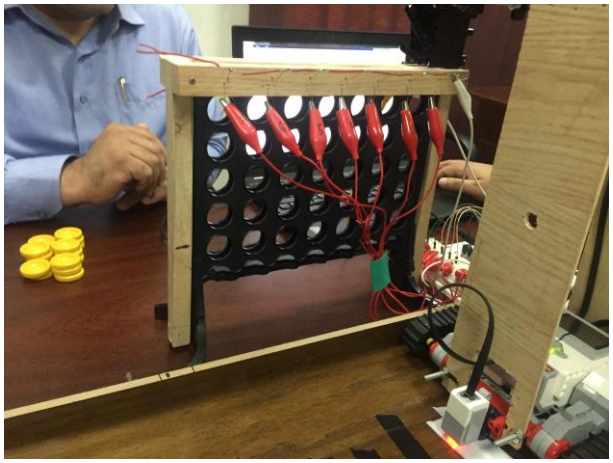


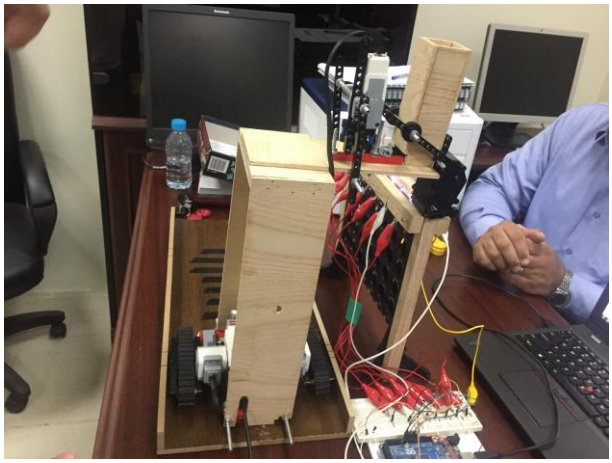


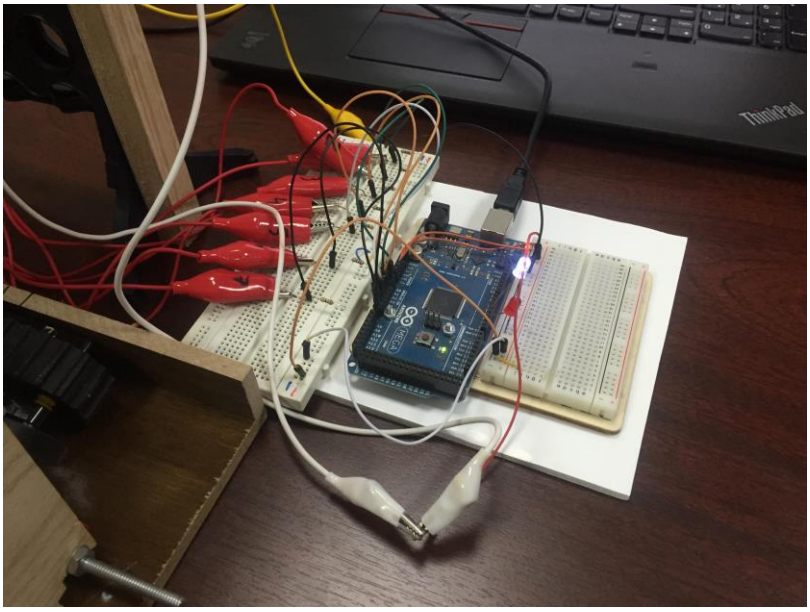

EV3




CONNECT 4






ThinkPad

The end

Thanks for your reading