PDE3413

System Engineering for Robotics

Group Members:

Mohammad Yusuf (M00528650)

Constantin-Alexandru Apetrei (M00536775)

Umut Ekerdiker (M00549321)

Oguzhan Janberk (M00567205)

Title: Arduino Powered Robot Playing Tic-Tac-Toe – "ToRo"

Date: 05/05/2019

Table of Contents

1. Introduction 3
1.1. Project Aim 3
1.2. Project Timeline & Group Meetings 4
2. Methodology & Background Research 4
2.1. Raspberry Turk 5
2.2. Jack the Dealer 6
3. System & Design Concept 7
4. Development & Implementation 9
4.1. Hardware Development & Implementation 9
4.2. Software Development & Implementation 10
5. Testing 11
6. Conclusion & Future Work 12
7. Evaluation 13
8. Screenshots 14

1. Introduction

Tic-tac-toe, also known as “noughts and crosses”, is a traditional paper and pencil game for two players. The first reference to the game was in 1858 in an issue of a magazine in Britain. Later in 1952, it was developed into one of the first known video games by the British computer scientist Alexander S. Douglas at the University of Cambridge.

Initially, players choose to draw X’s or O’s until the game is concluded. The game is played on a 3x3 grid and the player who manages to place three of their mark in a horizontal, vertical or diagonal row wins the game. Due to the simplicity and easy to adapt nature of the game, the game usually leads to a draw, however, tic-tac-toe has been an excellent pedagogical tool for teaching concepts of logic and intelligence to young children.

This project proposes to design a robotic arm that plays tic-tac-toe supported by Arduino Mega, a USB camera and a breadboard. The main goal is to bring the traditional, in-person way of playing tic-tac-toe with a reliable, cheap and fun robot.

1.1. Project Aim

The aim for the project was to create and implement a social robot that interacts with the user by playing the game. By having the social robot play the game with the user, it will have social qualities because it is interacting with the user and it will feature user-friendly features such as a “happy” animation when the robot wins the game. The aim was for our group to build a robot from scratch by assembling the parts together and getting the software to move the robot. A vital aspect that we needed to consider was to have the requirements that we were given. The requirements were to have a camera and have 2-degree movement. By having these requirements, it would make life easy into building the robots because the requirements have important parts that we need in our robot.

1.2. Project Timeline & Group Meetings

We were all put into group first before starting the project. Alongside group meetings in early time on Thursdays, we also scheduled days off to work on out project. We met on our days off and weekends to work on the project by having individual roles to take on. This allowed us to complete the robot much faster and easier because we all worked on different things. In total we had over 15 meetings, and this allowed us to work efficiently as a group and also keep track of the project.

2. Methodology & Background Research

In this section of the report, existing similar architectures will be reviewed and analysed. After analysing the existing products, advantages, disadvantages and improvements will be detailed for ToRo. Currently, there are many robots in the market that are used in pedagogical development of young children. These robots include programmable robots, robots for speaking disabilities, soccer playing robots, and chess playing robots. However, there are currently no advertised robot specialised in playing tic-tac-toe. Hence, the robots that will be analysed in this section will be similar types of architectures with relevant purposes.

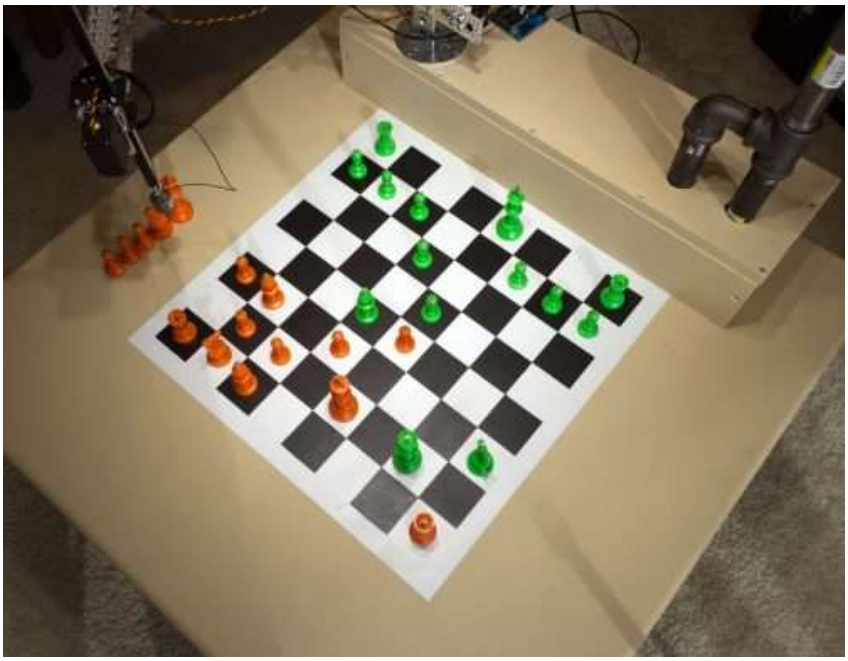
As explained in the introduction, ToRo is a robotic arm that will be able to play the traditional tic-tac-toe game. As the game is intended for young children, ToRo will fit into market of toys for young children from age 3 to 12. ToRo can be advertised in 2 main markets, for home use and for educational use in pedagogical treatment centres, education centres and social events.

Even though tic-tac-toe is available on almost any digital platform, the physical and traditional way of playing will gather attention from young children. This is expected to be one of the greatest advantage of the product. Also, the size of the robotic arm being transportable easily, it is also great for retail in toy shops and other venues.

2.1. Raspberry Turk

Raspberry Turk is an open source robot based on Raspberry Pi that can play chess physically against a human. It has been inspired by an 18th century mechanical chess playing machine that has gained popularity after touring Europe and America and playing against Benjamin Franklin and Napoleon Bonaparte. The working mechanism of the machine remains unknown.

Raspberry Turk's software is developed in Python and runs on Raspberry Pi. It includes a camera, a robotic arm, chess table and pieces. The successful part about this robot is its software is well developed and the machine is able to play professional level of chess. However, the biggest disadvantage is the robot has to include the special table and coloured chess pieces in order to play against a human. Due to lack of sensors and algorithms, it is unable to play on a regular chess board. This makes the robot expensive to make and a very large piece of hardware. Raspberry Turk can be improved by adding a functionality of being able to play on any board rather than its specialised board and chess pieces.



Raspberry Turk – Raspberry Powered Chess Playing Robot

2.2. Jack the Dealer

Jack the dealer is a patented product that is already in the market. The robot was released in July 2016. Jack the dealer can shuffle and distribute the cards. The robot can also assist players to comply with rules of the game to keep the game running which makes it perfect for home use. The machine is compact, cheap and commercially available.

On the other hand, the purpose of the robot is eliminating the dealer in casinos however, it still requires an operator to place the cards in the slots as well as to remove remaining cards from the machine. For these purposes, Jack the Dealer is suitable for home games however it is far from commercial space to be used as a dealer and it always has a major flow of requiring an operator which conflicts with its main purpose. It can be improved by adding more automated features that will allow the machine to run by itself and modify according to the game.

Jack the Dealer – Card Dealing Robot

3. System & Design Concept

The group came up with an idea of a social robot that plays tic tac toe with a human. The social robot will be playing the game by using an object to play it onto a table. There will be two sets of objects with two sets of colours. There will be a 3x3 grid printed on an A4 paper. There will be white and red button shaped items to be used by the player and the robot to fill the grids. The robot will include a robotic arm to place items and a camera to detect the player movements and changes within the grid. The robot will be seeing the drawing through a camera which will allow it to see what empty places there are. The robot will use its arm to place the object.

The robot has four servos to power the sections named base, shoulder, elbow and pin (claw). One of these motors will be placed in the base of the robotic arm to turn around and create a reaching angle, one in the middle of the robotic arm to bend for the intended grid, one at the tip of the arm to act as a wrist and the last one in the end part for gripping the items. The servos will help the robotic arm to move its joints and it is a required for the robot to have two or more degree of movement which the robot arm has four. The concept is very simple to understand, and it fits the criteria of the robot being social because it interacts with a human by playing a game. The requirements that was essential for the robot to have was a camera and a 2-degree movement.

After the ideas and design of the hardware part of the project, we designed the game in order to implement it within the code. For this purpose, we decided that we needed to code set of movements for the robotic arm to pick up and place the items on the grid. For the logic of the game, we designed an activity diagram in order to implement it in Visual Studio.

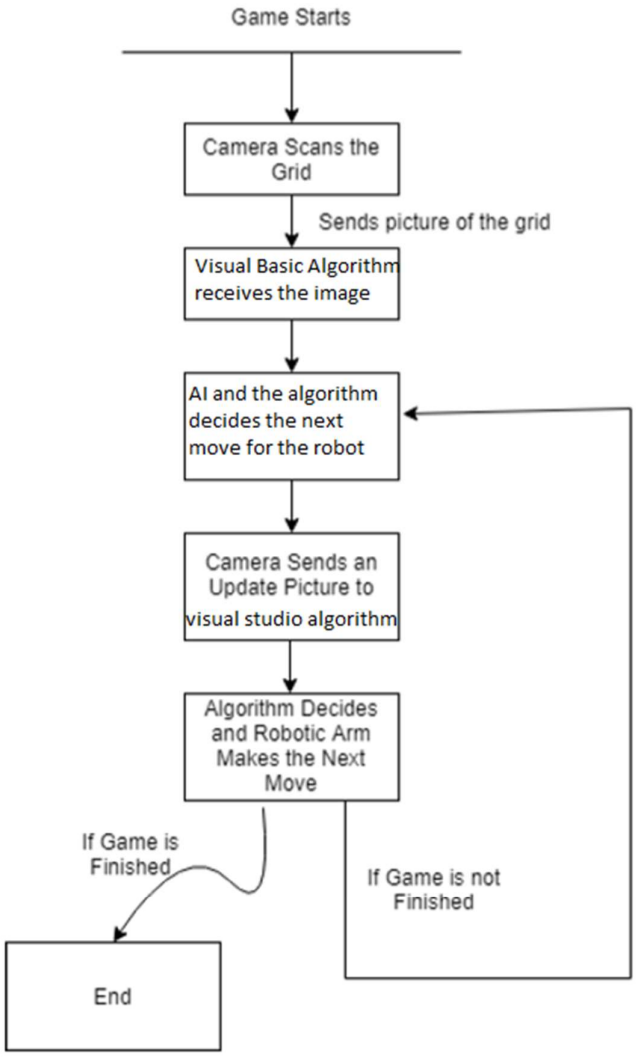

Game Starts
Camera Scans the Grid
Sends picture of the grid
Visual Basic Algorithm receives the image
AI and the algorithm decides the next move for the robot
Camera Sends an Update Picture to visual studio algorithm
Algorithm Decides and Robotic Arm Makes the Next Move
If Game is Finished
If Game is not Finished
End

Figure 1

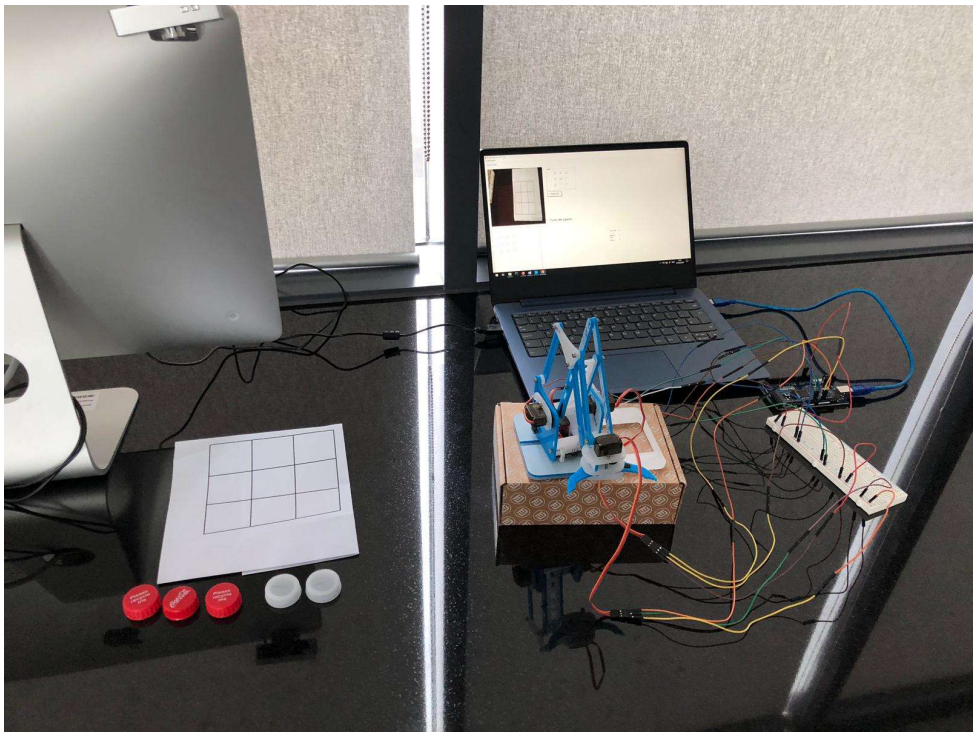
4. Development & Implementation

The implementation stage has a process to be taken for the implementation for the robot arm. There two sides of the implementation which was the hardware and software. For the robot to be successfully built and function, both software and hardware have to be completed. We started off the hardware side of the robot by building it.

4.1. Hardware Development & Implementation

The first step for implementing the hardware was to gather the robotic parts that were 3D printed and assemble them together. We used an Allen Key as a tool to tighten the parts with the screws on. The robot had a base to hold the arm and there were servos connected to allow the joints to move easily. There were four servos in total, two servos were to move the shoulder and elbow, the third servo was to move the claw that picks up the objects and the last one was to move the base.

We only needed the arm of the robot to be built because it required only the arm to function with the game. Before putting all the parts together, we looked at a data sheet which showed us which parts are needed and which parts fit together. Complete hardware setup of the project is shown below.



Components used:

Servos: The servos were used to move the joints or the arm. This component helped move the arm in all directions and was vital that it did because it needed this to play the game. Servo motors are small as they have built-in circuitry to control their movement they can be connected directly to an Arduino. The requirement of the robot needed at least 2-degree movement and the servos allowed the robot to this.

Camera: The camera is used to capture data from the user's interactions and based on the interaction, the robot will take feedback and take his next action. The camera was used for the robot to see the grid for it to know what places have been taken and where to place the object.

Breadboard: We used the breadboard to power the servos individually to improve the mobility of the robotic arm. By having more power to the servos, they can help move the arm and the joints faster and smoother. The ground and power were connected to the breadboard and the pins were connected to the Arduino.

Other External Tools: Allen key to tighten the parts, 10 x M3 Nut, 6 x M3-6mm Screws, 15 x M3-8mm Screws, 3 x M3-10mm Screws, 8 x M3-12mm Screws, 4 x M3-20mm Screws

4.2. Software Development & Implementation

During the design of the software part of our project, we initially improved ourselves on controlling the robot using Arduino Mega and Arduino IDE. For this purposes, we programmed our servos to move to certain positions and we initially accomplished to make the robotic arm pick-up items from the ground from predetermined locations. (Screenshots 1 to 4). After doing that, we started to set-up multiple levels of movements for the robot to pick up an item and place it on another place. At this stage, we have accomplished to automate the robot to move by itself by using the Arduino IDE to complete a full movement for the tic tac toe gameplay. (Screenshots 5 to 8). We also coded a movement called "happy" to use it when the robot wins the game.

For the second part of the software development, we moved on to research on creating the game logic. Therefore, we moved on to looking into other people's projects and developed a game logic file in Visual Basic. Therefore, we looked into how to make the computer make a decision and found about Minimax algorithm that is used in 2 player games to help computers give a decision. (Screenshots 9 and 10 as a part of the code).

Therefore, we have used the USB camera to take a picture and send it to the algorithm and by using the Minimax algorithm, Visual Basic algorithm makes a decision and pushes the Arduino code to move the robotic arm to place an item to the initially decided spot.

5. Testing

At the start of the project, we started with a robot arm called Dobot. This arm was already made and had its own software. As a group we tried to implement our own code in order for it to move into certain positions by itself. Unfortunately, this didn't work because due to the Dobot having its own software, we figured out that any other code that are added in the software wouldn't work. The Dobots own software had buttons that could be pressed in order for it to move the arm from left, right, up and down. This was not what our group wanted because we wanted to move the arm on its own in all directions with our own code base. There was another feature that the Dobot had which was a suction that could pick up the object by its suction functionality. Due to not finding our own code for this feature not working we didn't go through with this Robot arm. We wanted to have an arm where it be connected to an Arduino and have our own code on it and function the way we wanted.

The Dobot was very limited and this held us back a few days of our project. To overcome this problem, we decided to get rid of the Dobot and build a robot arm from scratch by 3D printing the parts and putting them together looking at a sheet. By building a robot by assembling its parts together, we had knowledge of how the architecture of the robot was. This allowed us to know everything about how the robot was built. If we had stuck to the Dobot, it would only allow us to move the arm with its own software and this wouldn't teach us anything.

Testing was the last stage. For testing, we tested both hardware and software. By testing it, it allowed us to see if the hardware and software was working in terms of functionality. This stage is very vital for the project because without testing the robot, we wouldn't know what parts are working and not. We needed to know and understand which part of architecture of the robot was functioning. By seeing that the hardware side was working, the software side had to work together with it. The software allowed the robot arm to play the game and to move in directions we set it to.

Hardware

- Tested to see if all the parts were together and there were no loose parts that could fall out
- Tested the wires to see if they were connected to the correct pins on the Arduino and on the Breadboard
- Tested to see if the robot functioned by running the code in Arduino to see if the armed moved.
- The power from the Arduino was not enough therefore breadboard was used.

Software

- Tested to see if the code on Arduino compiled successfully
- Tested to see that the code for the robot to move in certain positions worked
- We had a problem with opening an Arduino pin for the Visual Basic code, solved it via adding another line of code that force opens a pin

6. Conclusion & Future Work

The group has managed to complete the project, all the requirements have been met and the prototype is working as it is was designed to. The social robot that we have created needs to play the famous Tic Tac Toe game with a user by having certain requirements met. The requirements have been fulfilled by having a camera and having 2 degree movement. Our robot has a camera and more than 2 degree movement. To be able to create a fully working project we have used a lot of tools and techniques such as Visual Basic, Arduino IDE and minimax algorithm. We have used Visual Basic to create the game concept and the Arduino IDE to move the robot. All that we can say now is that the project has been successful, and we have managed

to create a social robot that plays a game with the user and shows social friendly feature by getting excited if the robot wins the game.

During the project, as a group we had little knowledge on building a robot and developing the software for it. By working together with individual roles and solving problem, we gained knowledge and experience from this.

For future work, we have decided that 2 improvements were necessary. As to run the robot we need a computer, we aim to bring Raspberry Pi on board to get rid of the necessity of the computer. For the second development, we aim to bring voice recognition features to create a more interactive experience.

7. Evaluation

To evaluate the whole project, we believed by creating our social robot it met all the requirements that were given to us. The requirements that it met was having a camera in which was used and having two-degree movement in which we used four-degree movement.

A social robot is an autonomous robot that interacts with humans or other autonomous by following social behaviours and rules attached to its role. The company has approached us with an idea for a social robotic system and has asked for help with possible applications we came up with the idea for the user to play tic tac toe with the robot. The system contains a camera and at least one more sensor to collect human input whilst having at least two degrees of freedom.

We've managed to fulfil this by building this robot. Tt is fully functional and works whilst set out to complete what it is meant to do. This is a concept of a robotic architecture, that will play a traditional tic-tac-toe game. It will include a robotic arm to place items and camera and detect where the object is placed as and changes within the grid by plotting either an" X" or an "O". On top of the robotic arm, a camera and an ultrasonic sensor will be statically placed. The robotic arm will contain 4 servo motors. One of these motors will be placed in the base of the robotic arm to turn around and create a reaching angle, one in the middle of the robotic arm to bend for the intended grid, one at the tip of the arm to act as a wrist and the last one in the end part for gripping the items.

8. Screenshots

arm_movement

```
void setup() {
#include <Servo.h>
int i,j;

Servo base;
Servo shoulder;
Servo elbow;
Servo pin;

int angle=0;
char fact;

int angleBase;
int angleshoulder;
int angleelbow;
int anglepin;
int anglegrippin;


  void setup()
  {
    Serial.begin(9600);


    base.attach(5);
    shoulder.attach(4);
    elbow.attach(9);
    pin.attach(10);

    angleBase=90;
    angleshoulder=90;
    angleelbow=90;
    anglepin=70;
    anglegrippin=130;
```

Screenshot 1

```
    base.write(90);
    delay(300);
    shoulder.write(90);
    delay(300);
    elbow.write(90);
    delay(300);
    pin.write(70);
    delay(300);


  }
void baseRightTurn()
{
  angleBase=angleBase+2;
  base.write(angleBase);
}

void baseLeftTurn()
{
  angleBase = angleBase-2;
  base.write(angleBase);
}

void shouldermoveup()
{
  angleshoulder=angleshoulder-2;
  shoulder.write(angleshoulder);
  Serial.println(angleshoulder);
}
void shouldermovedown()
{
  angleshoulder=angleshoulder+2;
  shoulder.write(angleshoulder);
  Serial.println(angleshoulder);
}
```

Screenshot 2

```
void elbowmoveup()
{
  angleelbow=angleelbow+2;
  elbow.write(angleelbow);
}

void elbowmovedown()
{
  angleelbow=angleelbow-2;
  elbow.write(angleelbow);
}

void openpin()
{
  anglepin=anglepin-5;
  pin.write(anglepin);
}

void closepin()
{
  anglepin=anglepin+5;
  pin.write(anglepin);
}

void initialPosition()
{
  for(i=90;i<=180;i++)
  {
    base.write(i);
  }
  delay(300);
  shoulder.write(90);
  delay(300);
  elbow.write(90);
  delay(300);
```

Screenshot 3

```
  pin.write(70);
}

void collectChip()
{
  for(i=90;i<=174;i++)
  {
    shoulder.write(i);
    delay(20);
  }
  delay(200);

  for(i=90;i<=114;i++)
  {
    elbow.write(i);
        delay(20);
  }
  delay(200);

  for(i=70;i<=anglegrippin;i++)
  {
    pin.write(i);
    delay(20);
  }
  delay(200);

  for(i=176;i>=140;i--)
  {
    shoulder.write(i);
        delay(20);
  }
}
```

Screenshot 4

```
//************************************
void firstPosition()
{
   for(i=180;i>=92;i--)
  {
    base.write(i);
    delay(20);
  }

  delay(200);

  for(i=140;i<=170;i++)
  {
    shoulder.write(i);
    delay(20);
  }

  delay(200);

  for(i=114;i<=124;i++)
  {
    elbow.write(i);
    delay(20);
  }

  delay(200);

  for(i=170;i<=176;i++)
  {
    shoulder.write(i);
    delay(20);
  }

  delay(200);
```

Screenshot 5

```
  delay(200);

  for(i=124;i<=130;i++)
  {
    elbow.write(i);
    delay(20);
  }

  delay(200);

  for(i=105;i>=70;i--)
  {
    pin.write(i);
    delay(20);
  }

  delay(200);

  for(i=176;i>=140;i--);
  {
    shoulder.write(i);
    delay(20);
  }

  initialPosition();

}
```

Screenshot 6

```
void secondPosition()
{
  //Base movement
  for(i=180;i>=88;i--)
  {
    base.write(i);
    delay(20);
  }

  delay(200);

  //Elbow movement  to 90 degrees
  for(i=114;i>=90;i--)
  {
    elbow.write(i);
    delay(20);
  }

  delay(200);


  //Shoulder movement to 160 degrees
  for(i=140;i<=170;i++)
  {
    shoulder.write(i);
    delay(20);
  }

  delay(200);
```

Screenshot 7

```
//Elbow movement
  for(i=90;i<=102;i++)
  {
    elbow.write(i);
    delay(20);
  }

  delay(200);

  //Opening the pin
  for(i=110;i>=90;i--)
  {
    pin.write(i);
    delay(20);
  }

  delay(200);

  //Raising the shoulder
  for(i=170;i>=140;i--)
  {
    shoulder.write(i);
    delay(20);
  }

  delay(200);

  initialPosition();

}
```

Screenshot 8

```
1   Public Class TicTacToe
2       Private m_iGrid(2, 2) As GridEntry    'A 3x3 Grid
3       Private m_bHasWinOccured As Boolean
4       Private m_iWinner As GridEntry
5
6       Public Event TicTacToeWinOccured(ByVal aWinner As GridEntry)
7       Public Event Cat()
8
        1 reference
9       Public Sub New()
10          Me.Clear()
11      End Sub
12
13      'Clear the board
        3 references
14      Public Sub Clear()
15          Dim iRow As Integer, iCol As Integer
16          For iRow = 0 To 2
17              For iCol = 0 To 2
18                  m_iGrid(iRow, iCol) = GridEntry.NoEntry
19              Next
20          Next
21          m_bHasWinOccured = False
22          m_iWinner = GridEntry.NoEntry
23      End Sub
24
25      'Three possible entries in a grid square
        58 references
26      Public Enum GridEntry
27          NoEntry = 0
28          PlayerX = 1
29          PlayerO = 2
30      End Enum
31
```

Screenshot 9

```
17      Public Function GetAIMove(ByVal aTTT As TicTacToe) As SquareCoordinate
18          Dim MoveList As New ArrayList
19          m_CurrentBoard = aTTT
20
21          Dim iRow As Integer
22          For iRow = 0 To 2
23              Dim possibleMoves() As SquareCoordinate
24              Try
25                  possibleMoves = GetPossibleRowMoves(iRow)
26              Catch ex As NullReferenceException
27                  Stop
28              Catch ex As Exception
29                  Stop
30              End Try
31
32              If Not possibleMoves Is Nothing Then
33                  Dim x As Integer
34                  For x = 0 To possibleMoves.Length - 1
35                      MoveList.Add(DetermineMoveEntry(possibleMoves(x)))
36                  Next
37              End If
38          Next
39
40          'Return a blocking move or choose a random normal move
41          Dim NormalMoveList As New ArrayList
42          Dim aMoveEntry As MoveEntry
43
44          'Make sure we check for winning moves
45          For Each aMoveEntry In MoveList
46              Select Case aMoveEntry.MoveType
47                  Case MoveType.Winning
48                      Console.Write(ControlChars.CrLf & aMoveEntry.ToString)
49                      Return aMoveEntry.SquareCoordinate
```

Screenshot 10