IOT Mini Project Report On

“TIC TAC TOE GAME”

Submitted in partial fulfillment for the award of the degree in

MASTER OF COMPUTER APPLICATION

By

ARUNABH DAS

(Enrolment Number – AJU/221788)

Under the esteemed guidance of

Dr. Arvind Kumar Pandey

(Dean)

&

Mr. Akash Kr. Bhagat

(Assistant Professor)

(Internal Guide)

ARKA JAIN UNIVERSITY

JAMSHEDPUR, JHARKHAND

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

2022 – 2024

CERTIFICATE

This is to certify that the project entitled, "**TIC TAC TOE GAME"**, is bonafide work of **Arunabh Das** bearing **Enrolment no – AJU/221788** under the guidance of **Mr. Akash Kumar Bhagat** submitted in partial fulfillment of the requirements for the award of degree of MASTER OF COMPUTER APPLICATION from ARKA JAIN UNIVERSITY during the year 2023 – 2024.

Internal Guide
Mr. Akash Kr. Bhagat
(Assistant Professor)
Department of Cs & IT
ARKA JAIN, University
Gamharia, Jharkhand

Dr. Arvind Kumar Pandey
Dean
ARKA JAIN, University
Gamharia, Jharkhand
dr.arvind@arkajainuniversity.ac.in

Date:

University Seal

ABSTRACT

The provided Arduino sketch implements a **"Tic Tac Toe Game"** with a graphical user interface on an LCD display. The game allows a player to interact with the system using two buttons to navigate and select moves. The computer opponent employs a basic artificial intelligence algorithm to make strategic moves. Graphics routines enable the rendering of X and O symbols on the LCD screen, enhancing the user experience.

The game logic is implemented in C++ and manages the state of the Tic Tac Toe grid, checking for wins or ties after each move. The computer opponent's decision-making process considers potential player wins and attempts to create winning opportunities for itself. The user interface is designed to be intuitive, and the graphical representation on the LCD enhances the gameplay experience.

Overall, this Arduino project demonstrates the integration of hardware input (buttons) and output (LCD display) to create an interactive and engaging gaming experience. It serves as a practical example of combining game logic, simple artificial intelligence, and graphical representation on an embedded system.

ACKNOWLEDGEMENT

We take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor to a successful culmination.

It is a genuine pleasure to express my profound gratitude and deep regard to my guide **"Mr. Akash Kr. Bhagat"** for their exemplary guidance, monitoring and constant encouragement.

I would like to specially thank **"Dr. Arvind Kumar Pandey "** our Dean who gave me the golden opportunity to do this wonderful project on the topic "**Tic Tac Toe Game**", which helped me in research and I came to know about so many things.

DECLARATION

I, Arunabh Das hear by declare that the Project entitled "**Tic Tac Toe Game**" done at **ARKA JAIN UNIVERSITY** has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

This project is done in the fulfillment of the requirement for the award degree of **MASTER OF COMPUTER APPLICATION** to be submitted as a mini project as part of our curriculum.

ARUNABH DAS
(AJU/221788)

Signature Of Student

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Introduction

Tic Tac Toe, a classic and universally recognized game, serves as the foundation for this Arduino-based project. The project focuses on creating an interactive and visually engaging implementation of Tic Tac Toe, leveraging an Arduino microcontroller and an LCD display. The primary goal is to offer users a tangible and enjoyable gaming experience while showcasing the integration of hardware and software components.

The project includes the utilization of two buttons for user input, enabling navigation within the game grid and the selection of moves. These inputs are processed by the Arduino, which manages the game state and updates the graphical representation on the LCD screen accordingly. The LCD display serves as the visual interface, presenting the Tic Tac Toe grid and providing feedback on the game's progress.

In addition to user interactions, the project features a computer opponent with a basic artificial intelligence algorithm. This algorithm enables the computer to make strategic moves, enhancing the challenge for the player. The game's graphics are brought to life through custom characters representing X and O, displayed dynamically on the LCD screen.

This project explores the synergy between hardware and software components, demonstrating how an Arduino microcontroller can serve as the brain of an interactive game. By combining game logic, graphical representation, and user input, the implementation showcases the versatility and creative potential of Arduino-based projects. The subsequent sections will delve into the details of the project, including hardware connections, game logic, user interface design, and the artificial intelligence algorithm employed by the computer opponent.

CHAPTER 2: PROPOSED MODEL

2.1 Tic Tac Toe Game

This Arduino-based project brings the classic game of Tic Tac Toe to life through an interactive and visually appealing experience. Using an Arduino microcontroller and an LCD display, users can navigate the game grid and make moves with two buttons. The project features a computer opponent with a basic AI algorithm, adding a strategic element to the game. Custom X and O graphics are dynamically displayed on the LCD, creating an engaging and enjoyable gaming environment. The project showcases the seamless integration of hardware and software components to deliver a tangible gaming experience.

2.2 HARDWARE REQUIREMENTS

2.2.1 Arduino Uno R3

Figure 1: Arduino Uno R3

The Arduino UNO is a standard board of Arduino. Here UNO means 'one' in Italian. It was named as UNO to label the first release of Arduino Software. It was also the first USB board released by Arduino. It is considered as the powerful board used in various projects. Arduino.cc developed the Arduino UNO board.

Arduino UNO is based on an ATmega328P microcontroller. It is easy to use compared to other boards, such as the Arduino Mega board, etc. The board consists of digital and analog Input/Output pins (I/O), shields, and other circuits.

The Arduino UNO includes 6 analog pin inputs, 14 digital pins, a USB connector, a power jack, and an ICSP (In-Circuit Serial Programming) header. It is programmed based on IDE, which stands for Integrated Development Environment. It can run on both online and offline platforms.

2.2.2 LCD 16 x 2

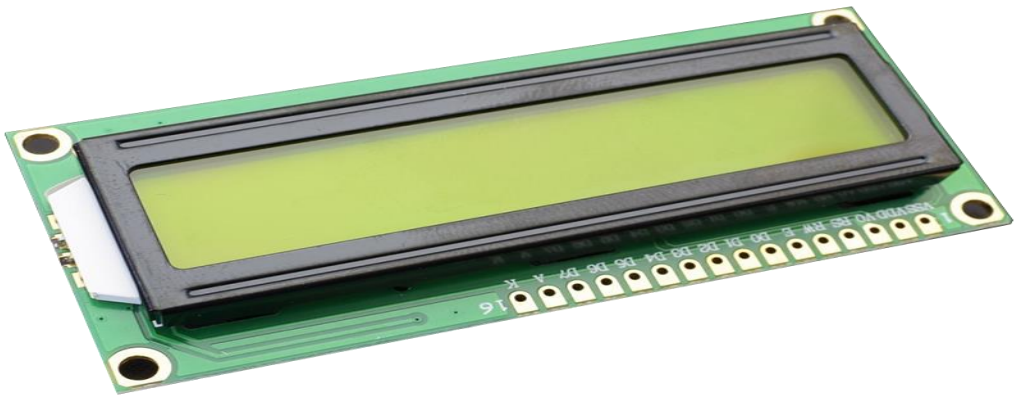

Figure 2: LCD 16 x 2

An LCD (Liquid Crystal Display) screen is an electronic display module and has a wide range of applications. A 16 x 2 LCD display is very basic module and is very commonly used in various devices and circuits. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. The 16 x 2

intelligent alphanumeric dot matrix display is capable of displaying 224 different characters and symbols. This LCD has two registers, namely, Command and Data.

2.2.3 Push Button

Figure 3: Push Button

A push-button (also spelled pushbutton) or simply button is a simple switch mechanism to control some aspect 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, so as to be easily depressed or pushed. Buttons are most often biased switches, although many un-biased buttons (due to their physical nature) still require a spring to return to their un-pushed state.

Terms for the "pushing" of a button include pressing, depressing, mashing, slapping, hitting, and punching. The "push-button" has been utilized in calculators, push-button telephones, kitchen appliances, and various other mechanical and electronic devices, home and commercial.

2.2.4 Resister

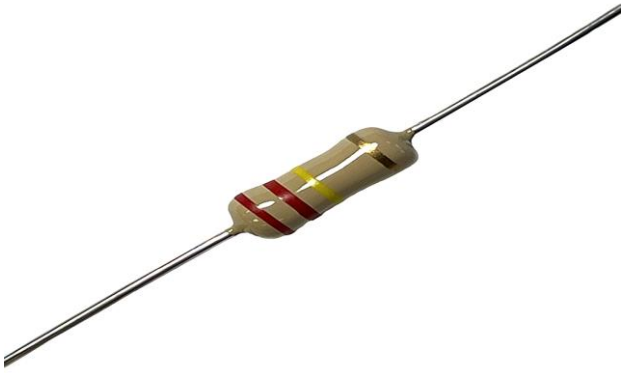

Figure 4: 1 kΩ Resistor

A resistor is a passive two-terminal electronic component that limits or regulates the flow of electric current in a circuit. Resistors are widely used in electronic circuits for various purposes, including voltage division, current limiting, signal conditioning, and biasing active elements like transistors. It is characterized by its resistance, measured in ohms (Ω). Resistors are commonly used to control current, divide voltage, and set bias points in electronic circuits. The resistance value is often indicated by colored bands on the resistor's body, following a standardized color code. Resistors play a crucial role in electronics by providing a predictable level of resistance, allowing engineers to design circuits with precise control over current and voltage.

2.2.5 Jumper Wires

Figure 5: Jumper Wires

Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with breadboards and other prototyping tools in order to make it easy to change a circuit as needed. Fairly simple. In fact, it doesn't get much more basic than jumper wires.

2.2.6 Breadboard

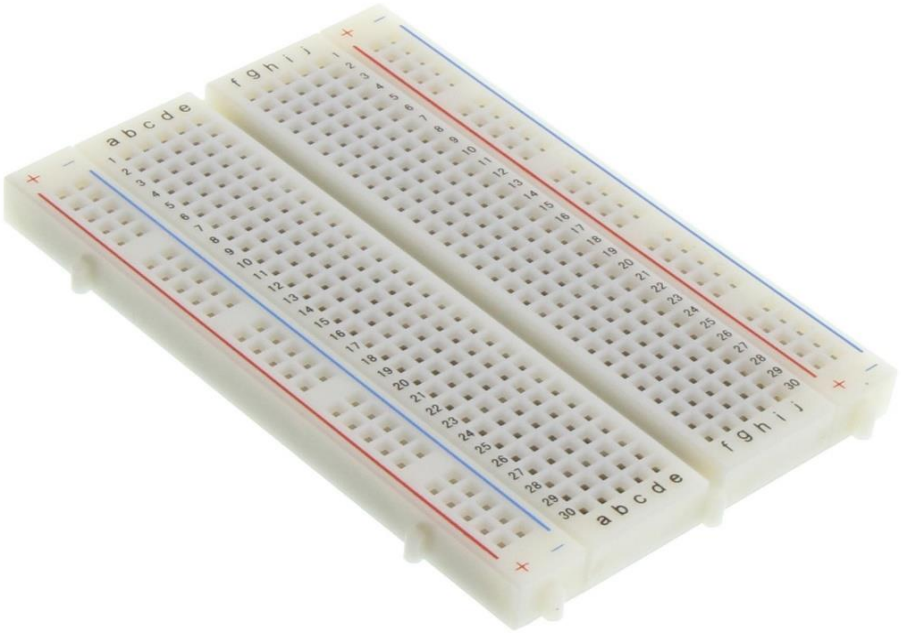

Figure 6: Breadboard

A breadboard is used to build and test circuits quickly before finalizing any circuit design. The breadboard has many holes into which circuit components like ICs and resistors can be inserted. The bread board has strips of metal which run underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontally while the remaining holes are connected vertically.

2.3 SOFTWARE REQUIREMENTS

2.3.1 Arduino Ide Software

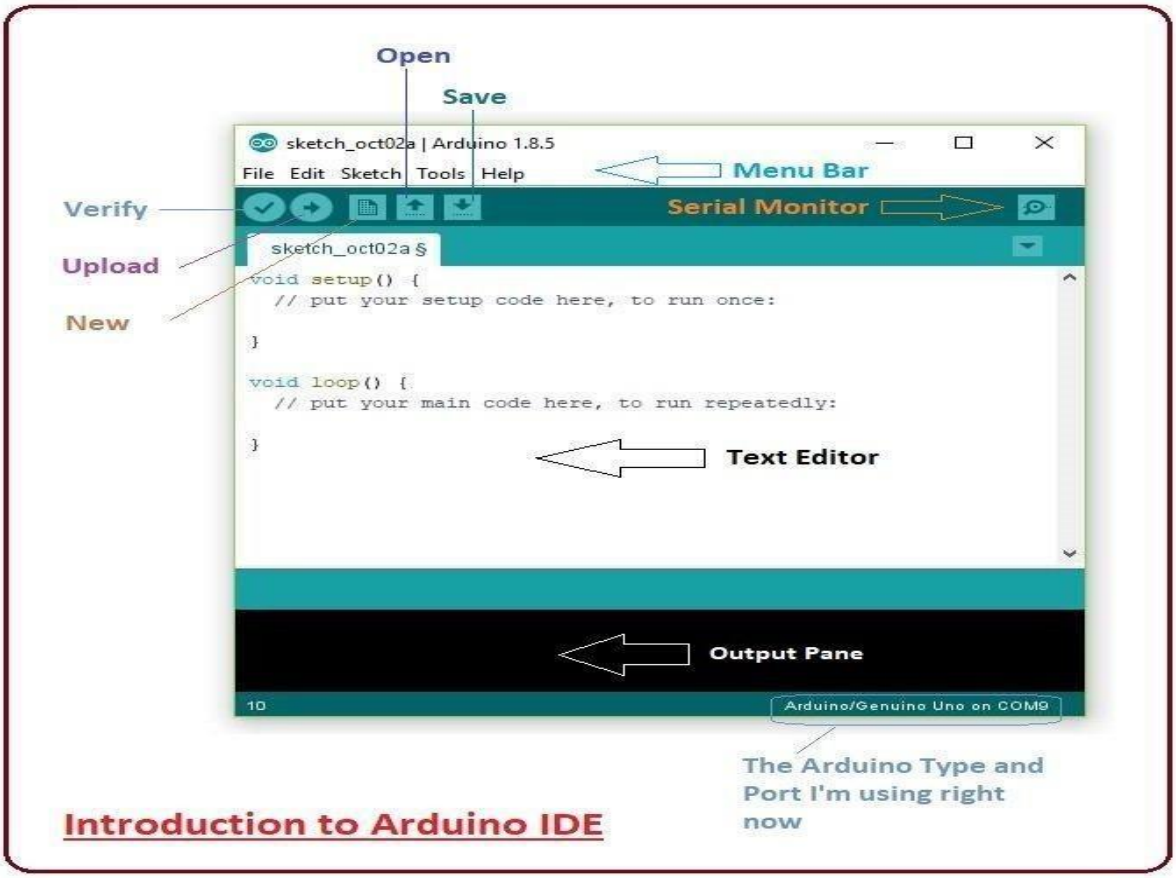

Figure 7: Arduino IDE Software

Arduino.cc that is mainly used for writing, compiling and uploading the code in almost all Arduino modules/boards. Arduino IDE is open-source software and is easily available to download & install from Arduino's Official Site.

Arduino IDE is an open-source software, designed by Arduino.cc and mainly used for writing, compiling & uploading code to almost all Arduino Modules.

It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.

It is available for all operating systems i.e., MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role in debugging, editing and compiling the code.

CHAPTER 3: WORKING PROCEDURE

3.1 Working

The Tic Tac Toe IoT project utilizes an Arduino Uno microcontroller, a 16x2 LCD display, and two push buttons to create an interactive gaming experience. The hardware setup involves connecting the LCD to display the game grid and player moves, and the push buttons serve as input devices for players to make their moves.

The Arduino code initializes the LCD and handles the game logic. It establishes a virtual Tic Tac Toe grid, monitors player button presses to record moves, and checks for win conditions or a draw after each move. The LCD dynamically updates to reflect the current state of the game, displaying the grid and player turns.

During gameplay, players take turns pressing the push buttons to place their respective marks (X or O) on the grid. The Arduino code ensures that each move is processed correctly, updating the internal game state and checking for a winner or a draw. Visual feedback is provided on the LCD, making the game interactive and engaging.

The project's simplicity makes it an accessible platform for learning about Arduino programming, electronics, and the basics of the Internet of Things. The combination of physical input devices and a visual display enhances the user experience, making it an educational and entertaining project suitable for various applications.

3.2 Circuit Diagram

Connect all the components as given in the circuit diagram.

1. **Buttons**:
    - NAVIGATE_BUTTON (Pin 7): Connect one button terminal to Pin 7 on the Arduino.
    - HIT_BUTTON (Pin 6): Connect one button terminal to Pin 6 on the Arduino.

2. **Liquid Crystal Display**:
    - lcd (Pin 12, 11, 5, 4, 3, 2): Connect the corresponding pins on the LCD to the Arduino as follows:
        - Pin 12: LCD RS (Register Select) - Connect to Arduino Pin 12.
        - Pin 11: LCD Enable - Connect to Arduino Pin 11.
        - Pin 5: LCD D4 - Connect to Arduino Pin 5.
        - Pin 4: LCD D5 - Connect to Arduino Pin 4.
        - Pin 3: LCD D6 - Connect to Arduino Pin 3.
        - Pin 2: LCD D7 - Connect to Arduino Pin 2.
        - Connect the VCC and GND pins of the LCD to 5V and GND on the Arduino, respectively.

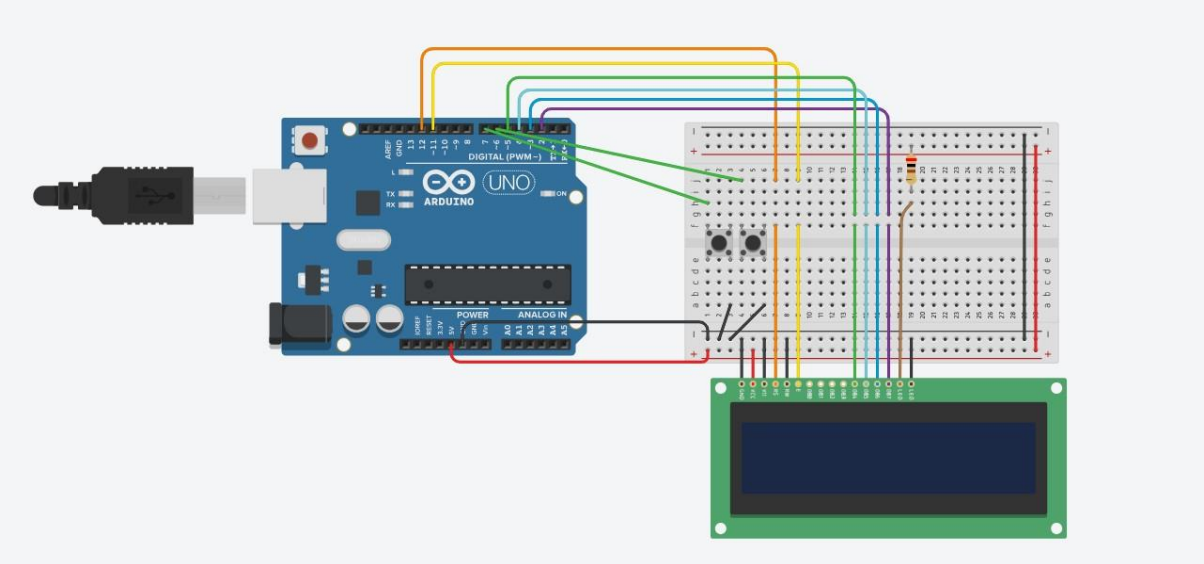

Figure 8: Circuit Diagram

3.3 Schematic View

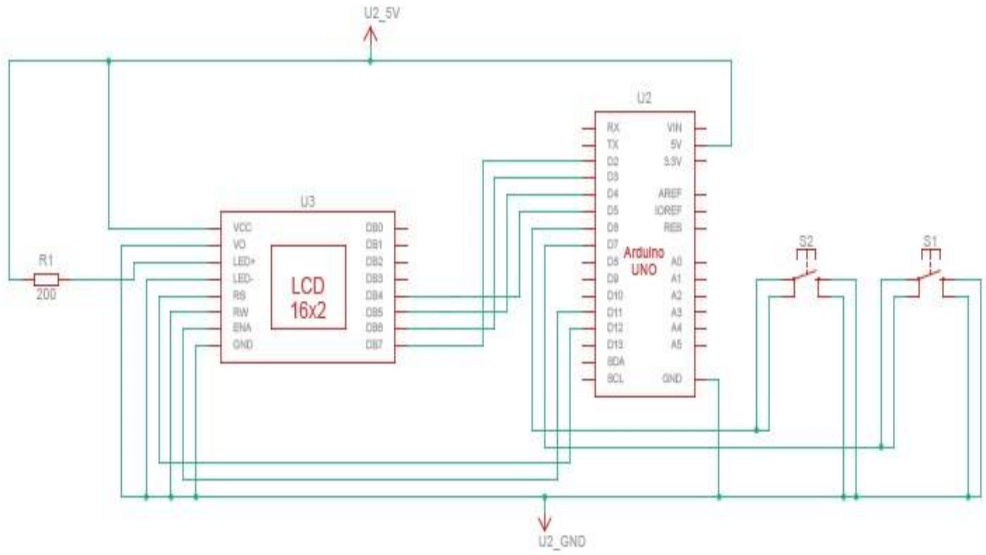

Figure 9: Schematic View

CHAPTER 4: RESULTS & DISCUSSION

4.1 Results

In this section, the setup of the whole research work is depicted in a step-by-step manner. Sample screenshots are displayed once the components are fixed and connected to each other. All the components are connected to each other and thus completes the system setup which helps one to understand the steps in a simple and easy way. With these steps, even when a person who is trying to implement the same, it makes it simple, clear and easy.

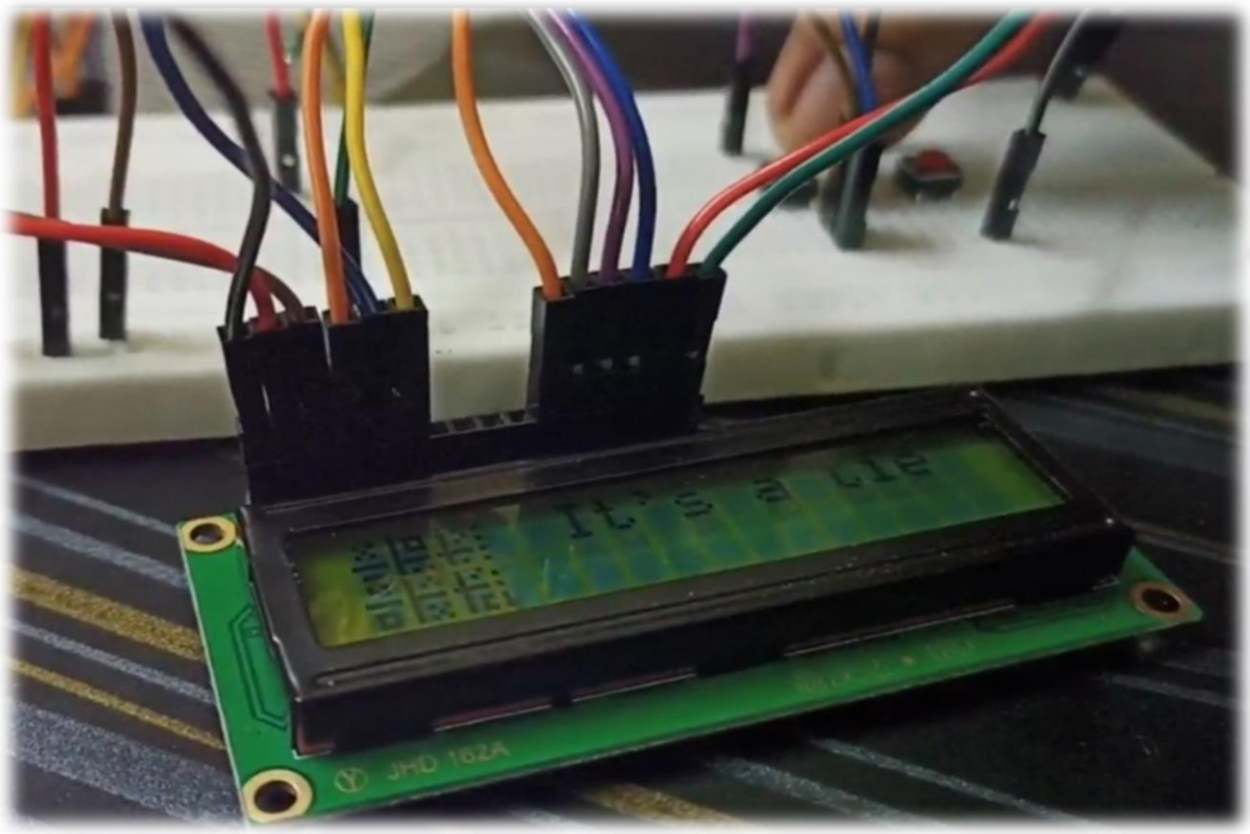

Figure 10: Result

CHAPTER 5: CODE

5.1 Coding

```
#include <LiquidCrystal.h>
#define LEFT_BUTTON 2
#define UP_BUTTON 7

LiquidCrystal lcd_1(12, 11, 5, 4, 3, 2);
byte graphics[64];

int squareState[3][3];
int moves = 0;
int markerX = 1;
int markerY = 1;
bool buttonPressed = false;

void setup()
{
  pinMode(LEFT_BUTTON, INPUT_PULLUP);
  pinMode(UP_BUTTON, INPUT_PULLUP);

  lcd.begin(16, 2);
  lcd.setCursor(1, 0);
  lcd.print("Welcome to");
  lcd.setCursor(0, 1);
  lcd.print("Tic-Tac-Toe!");
  delay(8000);
  lcd.clear();

  for (int i = 0; i < 8; ++i)
  {
        lcd.createChar(i, &graphics[i << 3]);
  }

  //set the first 4 characters of the 2 rows to our 8 custom characters
  lcd.setCursor(0, 0);
  for(int i = 0; i < 4; i++) lcd.write((char)i);
  lcd.setCursor(0, 1);
  for(int i = 4; i < 8; i++) lcd.write((char)i);

  ResetGame();
}
```

```
void loop() {
  bool userSelected = false;
  int button;
  int nextX, nextY;
  button = GetUserInput();
  switch(button)
  {
    case 0: //nothing yet
    break;

    case 1: //marker left
      ClearMarker(markerX, markerY);
      GetPrevFreeSquare(nextX, nextY);
      markerX = nextX;
      markerY = nextY;
      SetMarker(markerX, markerY);
      WaitForNoUserInput();
    break;

    case 2: //marker right
      ClearMarker(markerX, markerY);
      GetNextFreeSquare(nextX, nextY);
      markerX = nextX;
      markerY = nextY;
      SetMarker(markerX, markerY);
      WaitForNoUserInput();
    break;

    case 3: //select
      userSelected = true;
      WaitForNoUserInput();
    break;
  }

  if(userSelected)
  {
    ClearMarker(markerX, markerY);

    squareState[markerX][markerY] = 1;
    DrawX(markerX, markerY);
    moves++;
    delay(200);
```

```
    if(CheckForWin())
    {
      FlashWinner(1);
      ResetGame();
    }
    else if (moves == 5) //tie
    {
      FlashWinner(3);
      ResetGame();
    }
    else
    {
      DoComputerMove();
      GetNextFreeSquare(markerX, markerY);
      SetMarker(markerX, markerY);
      if(CheckForWin())
      {
        FlashWinner(2);
        ResetGame();
      }
    }
  }
}

void DoComputerMove()
{
  bool done = false;

  if(moves == 1)
  {
    //if first player move isn't the center, take the center
    if(squareState[1][1] == 0)
    {
      squareState[1][1] = 2;
      DrawO(1, 1);
    }
    else // take any corner.  A side could equal a loss later.
    {
      int row = random(0, 2) << 1;
      int col = random(0, 2) << 1;

      squareState[row][col] = 2;
      DrawO(row, col);
    }
```

```
}
if(moves == 2)
{
  done = false;
  // if player has 2 in a row, block it
  done = BlockPlayerWin();

  // look for any doubling traps.
  if(!done)
  {
    // Corner-opposite corner.  Take any side.  Corner would lead to loss.
    if (((squareState[0][0] == 1) && (squareState[2][2] == 1)) ||
        ((squareState[0][2] == 1) && (squareState[2][0] == 1)))
    {
      int row = random(0,3);
      int col = 1;
      if(row == 1) col = random(0,2) << 1;
      squareState[row][col] = 2;
      DrawO(row, col);
      done = true;
    }

    // Side-corner. Take inside corner.
    //top-center plus corner
    if(!done && (squareState[0][1] == 1))
    {
      if(squareState[2][0] == 1)
      {
        squareState[0][0] = 2;
        DrawO(0, 0);
        done = true;
      }
      else if (squareState[2][2] == 1)
      {
        squareState[0][2] = 2;
        DrawO(0, 2);
        done = true;
      }
    }

    //bottom -center plus corner
    if(!done && (squareState[2][1] == 1))
    {
      if(squareState[0][0] == 1)
```

```
  {
    squareState[2][0] = 2;
    DrawO(2, 0);
    done = true;
  }
  else if (squareState[0][2] == 1)
  {
    squareState[2][2] = 2;
    DrawO(2, 2);
    done = true;
  }
}

//center-left plus corner
if(!done && (squareState[1][0] == 1))
{
  if(squareState[0][2] == 1)
  {
    squareState[0][0] = 2;
    DrawO(0, 0);
    done = true;
  }
  else if (squareState[2][2] == 1)
  {
    squareState[2][0] = 2;
    DrawO(2, 0);
    done = true;
  }
}

//center-right plus corner
if(!done && (squareState[1][2] == 1))
{
  if(squareState[0][0] == 1)
  {
    squareState[0][2] = 2;
    DrawO(0, 2);
    done = true;
  }
  else if (squareState[2][0] == 1)
  {
    squareState[2][2] = 2;
    DrawO(2, 2);
    done = true;
```

```
      }
    }

    // Center-corner (if computer has opposite corner).  Take either remaining corner.
    if(!done)
    {
     if(squareState[1][1] == 1)
     {
      if(((squareState[0][0] == 2) && (squareState[2][2] == 1)) || ((squareState[0][0] == 1)
&& (squareState[2][2] == 2)))
       {
        if(random(0,2) == 1)
        {
         squareState[0][2] = 2;
         DrawO(0, 2);
        }
        else
        {
         squareState[2][0] = 2;
         DrawO(2, 0);
        }
        done = true;
       }
       else if(((squareState[0][2] == 2) && (squareState[2][0] == 1)) || ((squareState[0][2] ==
1) && (squareState[2][0] == 2)))
       {
        if(random(0,2) == 1)
        {
         squareState[0][0] = 2;
         DrawO(0, 0);
        }
        else
        {
         squareState[2][2] = 2;
         DrawO(2, 2);
        }
        done = true;
       }
     }
    }

    // Otherwise take any random spot
    if (!done)
    {
```

```
      int row = 1;
      int col = 1;
      do
      {
        row = random(0, 3);
        col = random(0, 3);
      }
      while(squareState[row][col] > 0);

      squareState[row][col] = 2;
      DrawO(row, col);
    }
  }
 }

 if(moves >= 3)
 {
  done = false;
  //if we have 2 in a row and can get 3 to win, do it
  done = FindComputerWin();

  //if player has 2 in a row, block it
  if(!done) done = BlockPlayerWin();

  // take any random spot
  if(!done)
  {
    int row = 1;
    int col = 1;
    do
    {
      row = random(0, 3);
      col = random(0, 3);
    }
    while(squareState[row][col] > 0);
    squareState[row][col] = 2;
    DrawO(row, col);
  }
 }
}
bool FindComputerWin()
{
 bool done = false;
```

```
  //down diag
  if((squareState[0][0] == 2) && (squareState[1][1] == 2) && (squareState[2][2] == 0))
  {
    squareState[2][2] = 2;
    DrawO(2, 2);
    done = true;
  }
  else if((squareState[2][2] == 2) && (squareState[1][1] == 2) && (squareState[0][0] == 0))
  {
    squareState[0][0] = 2;
    DrawO(0, 0);
    done = true;
  }

  //up diag
  if(!done)
  {
    if((squareState[0][2] == 2) && (squareState[1][1] == 2) && (squareState[2][0] == 0))
    {
      squareState[2][0] = 2;
      DrawO(2, 0);
      done = true;
    }
    else if((squareState[2][0] == 2) && (squareState[1][1] == 2) && (squareState[0][2] == 0))
    {
      squareState[0][2] = 2;
      DrawO(0, 2);
      done = true;
    }
  }
  if (done) return done;

  //rows
  for(int row = 0; row < 3; row++)
  {
    if((squareState[row][0] == 2) && (squareState[row][1] == 2) && (squareState[row][2] ==
0))
    {
      squareState[row][2] = 2;
      DrawO(row, 2);
      done = true;
      break;
    }
```

```
    else if((squareState[row][0] == 2) && (squareState[row][1] == 0) && (squareState[row][2]
== 2))
    {
      squareState[row][1] = 2;
      DrawO(row, 1);
      done = true;
      break;
    }
    else if((squareState[row][0] == 0) && (squareState[row][1] == 2) && (squareState[row][2]
== 2))
    {
      squareState[row][0] = 2;
      DrawO(row, 0);
      done = true;
      break;
    }
  }
  if (done) return done;

  //cols
  for(int col = 0; col < 3; col++)
  {
    if((squareState[0][col] == 2) && (squareState[1][col] == 2) && (squareState[2][col] == 0))
    {
      squareState[2][col] = 2;
      DrawO(2, col);
      done = true;
      break;
    }
    else if((squareState[0][col] == 2) && (squareState[1][col] == 0) && (squareState[2][col]
== 2))
    {
      squareState[1][col] = 2;
      DrawO(1, col);
      done = true;
      break;
    }
    else if((squareState[0][col] == 0) && (squareState[1][col] == 2) && (squareState[2][col]
== 2))
    {
      squareState[0][col] = 2;
      DrawO(0, col);
      done = true;
      break;
```

```
    }
  }
  return done;
}

bool BlockPlayerWin()
{
  bool done = false;

  //down diag
  if((squareState[0][0] == 1) && (squareState[1][1] == 1) && (squareState[2][2] == 0))
  {
    squareState[2][2] = 2;
    DrawO(2, 2);
    done = true;
  }
  else if((squareState[2][2] == 1) && (squareState[1][1] == 1) && (squareState[0][0] == 0))
  {
    squareState[0][0] = 2;
    DrawO(0, 0);
    done = true;
  }

  //up diag
  if(!done)
  {
    if((squareState[0][2] == 1) && (squareState[1][1] == 1) && (squareState[2][0] == 0))
    {
      squareState[2][0] = 2;
      DrawO(2, 0);
      done = true;
    }
    else if((squareState[2][0] == 1) && (squareState[1][1] == 1) && (squareState[0][2] == 0))
    {
      squareState[0][2] = 2;
      DrawO(0, 2);
      done = true;
    }
  }
  if (done) return done;

  //rows
  for(int row = 0; row < 3; row++)
  {
```

```
    if((squareState[row][0] == 1) && (squareState[row][1] == 1) && (squareState[row][2] ==
0))
    {
      squareState[row][2] = 2;
      DrawO(row, 2);
      done = true;
      break;
    }
    else if((squareState[row][0] == 1) && (squareState[row][1] == 0) && (squareState[row][2]
== 1))
    {
      squareState[row][1] = 2;
      DrawO(row, 1);
      done = true;
      break;
    }
    else if((squareState[row][0] == 0) && (squareState[row][1] == 1) && (squareState[row][2]
== 1))
    {
      squareState[row][0] = 2;
      DrawO(row, 0);
      done = true;
      break;
    }
  }
  if (done) return done;

  //cols
  for(int col = 0; col < 3; col++)
  {
    if((squareState[0][col] == 1) && (squareState[1][col] == 1) && (squareState[2][col] == 0))
    {
      squareState[2][col] = 2;
      DrawO(2, col);
      done = true;
      break;
    }
    else if((squareState[0][col] == 1) && (squareState[1][col] == 0) && (squareState[2][col]
== 1))
    {
      squareState[1][col] = 2;
      DrawO(1, col);
      done = true;
      break;
```

```
    }
    else if((squareState[0][col] == 0) && (squareState[1][col] == 1) && (squareState[2][col]
== 1))
    {
      squareState[0][col] = 2;
      DrawO(0, col);
      done = true;
      break;
    }
  }
  return done;
}


bool CheckForWin()
{
  for(int player = 1; player < 3; player++)
  {
    if(squareState[0][0] == player)
    {
      if((squareState[1][1] == player) && (squareState[2][2] == player)) return true; // right-
down diag
      if((squareState[1][0] == player) && (squareState[2][0] == player)) return true; //col 1
      if((squareState[0][1] == player) && (squareState[0][2] == player)) return true; // row 1
    }
    if((squareState[1][0] == player) && (squareState[1][1] == player) && (squareState[1][2]
== player)) return true; //row 2
    if((squareState[2][0] == player) && (squareState[2][1] == player) && (squareState[2][2]
== player)) return true; //row 3
    if((squareState[0][1] == player) && (squareState[1][1] == player) && (squareState[2][1]
== player)) return true; //col 2
    if((squareState[0][2] == player) && (squareState[1][2] == player) && (squareState[2][2]
== player)) return true; //col 3

    if((squareState[2][0] == player) && (squareState[1][1] == player) && (squareState[0][2]
== player)) return true; //right-up diag
  }
  return false;
}

void FlashWinner(int playerOrComp)
{
  lcd.setCursor(5, 0);
  if(playerOrComp == 1)
```

```
  {
    lcd.print("You win!");
  }
  else if (playerOrComp == 2)
  {
    lcd.print("You lose.");
  }
  else  //tie
  {
    lcd.print("It's a tie");
  }
  delay(2000);
  lcd.setCursor(5, 0);
  lcd.print("          ");
}

void ResetGame()
{
  moves = 0;
  markerX = 1; markerY = 1;

  //erase the squareState array
  for(int row = 0; row < 3; row++)
      for(int col = 0; col < 3; col++)
        squareState[row][col] = 0;

  //erase graphics
  for(int index = 0; index < 64; index++) graphics[index] = 0;

  // define 8 custom characters using our (now blank) bit data
  for (int i = 0; i < 8; ++i)
  {
    lcd.createChar(i, &graphics[i << 3]);
  }

  //set the first 4 characters of the 2 rows to our 8 custom characters
  lcd.setCursor(0, 0);
  for(int i = 0; i < 4; i++) lcd.write((char)i);
  lcd.setCursor(0, 1);
  for(int i = 4; i < 8; i++) lcd.write((char)i);

  //draw grid
  drawLine(5, 0, 5, 15, 1);
  drawLine(11, 0, 11, 15, 1);
```

```
  drawLine(0, 5, 15, 5, 1);
  drawLine(0, 11, 15, 11, 1);

  //draw marker
  SetMarker(markerX, markerY);
}

int GetUserInput()
{
  if (digitalRead(LEFT_BUTTON) == LOW) return 1;
  //if (digitalRead(RIGHT_BUTTON) == LOW) return 2;
  if (digitalRead(UP_BUTTON) == LOW) return 3;
  return 0;
}

void SetMarker(int x, int y)
{
   setPixel((x * 6) + 2, (y * 6) + 2, 1);
}

void ClearMarker(int x, int y)
{
   setPixel((x * 6) + 2, (y * 6) + 2, 0);
}

void GetPrevFreeSquare(int& nextX, int& nextY)
{
   int tempX = markerX;
   int tempY = markerY;

   do{
     tempX--;
     if(tempX == -1)
     {
       tempX = 2;
       tempY--;
       if(tempY == -1) tempY = 2;
     }
   }while (squareState[tempX][tempY] > 0);

   nextX = tempX;
   nextY = tempY;
}
```

```
}

//graphics routines:
void setPixel(int x, int y, int color)
{
  //sanity checks.  SetPixel would trash memory and crash if these were exceeded.
  if((x < 0) || (y < 0) || (x > 19) || (y > 15)) return;

  int col = x / 5; //will be 0 through 3
  int row = y >> 3; // div 8.  will be 0 or 1

  int col_inner = x % 5; //0 through 4
  int row_inner = y % 8; // 0 through 7
  col_inner = 4 - col_inner; //col 1 comes before col 2, but is a higher bit in the bitmap (lsb is
rightmost).

  int charnum = row * 4 + col;

  if(color == 1) //set pixel
  {
    graphics[(charnum << 3) + row_inner] |= color << col_inner;
  }
  else // clear pixel if set
  {
    int bitmask = B11111 - (1 << col_inner);
    graphics[(charnum << 3) + row_inner] &= bitmask;
  }

  lcd.createChar(charnum, &graphics[charnum << 3]);
  //apparently don't need to re-lcd.write, character updates on lcd automatically if line below is
used.
  lcd.setCursor(col, row);
}

void drawLine(int x1, int y1, int x2, int y2, int color)
{
  float incr;
  int xabs = abs(x1 - x2);
  int yabs = abs(y1 - y2);
  if((xabs == 0) && (yabs == 0)) //single pixel, just set it and bail
  {
    setPixel(x1, y1, color);
    return;
  }
```

```
 //line between 0 and 19 needs 20 pixels for example, so add one.
 xabs++;
 yabs++;

 if (xabs < yabs) //mostly vertical, x increments < 1
 {
  float x = x1 + 0.5;  //add 0.5 to deal with up-rounding vs. down-rounding.
  int yinc = (y1 < y2) ? 1 : -1;

  incr = (float)(x2 - x1) / (float)yabs;
  for(int y = y1; y != y2; y+= yinc)
  {
   setPixel(x, y, color);
   x += incr;
  }
  setPixel(x, y2, color);
 }
 else //mostly horizontal, y increments < 1
 {
  float y = y1 + 0.5;  //add 0.5 to deal with up-rounding vs. down-rounding.
  int xinc = (x1 < x2) ? 1 : -1;

  incr = (float)(y2 - y1) / (float)xabs;
  for(int x = x1; x != x2; x+= xinc)
  {
   setPixel(x, y, color);
   y += incr;
  }
  setPixel(x2, y, color);
 }
}
```

CHAPTER 6: APPLICATIONS & ADVANTAGES

6.1 Applications

1. Educational Tool.
2. Recreational Entertainment.
3. Learning Programming and Electronics.
4. Accessible Interactive Art.

6.2 Advantages

1. Hands-On Learning.
2. Customization and Creativity.
3. Low-Cost Hardware.
4. Versatility for Future Development.
5. Interactive User Experience.

CHAPTER 7: CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

In conclusion, the Tic Tac Toe IoT project represents a successful integration of hardware and software components, creating an engaging and educational gaming experience. By leveraging Arduino Uno, a 16x2 LCD display, and push buttons, the project introduces users to the fundamentals of electronics, microcontroller programming, and the Internet of Things (IoT). The interactive nature of the game, with players making moves through physical buttons and receiving real-time feedback on the LCD, enhances the overall learning experience.

The project's adaptability allows for customization and future expansion, offering avenues for advanced features such as cloud connectivity, online multiplayer capabilities, and integration with other IoT devices. Its low-cost hardware and simplicity make it accessible to a wide audience, making it an ideal tool for teaching and learning.

Through this project, users not only acquire practical skills in Arduino programming but also gain insights into basic game development and IoT concepts. As technology continues to advance, the Tic Tac Toe IoT project stands as a foundation for further exploration and innovation, showcasing the potential for interactive and connected applications in the realm of electronics and IoT. Overall, the project successfully merges education and entertainment, providing a tangible and enjoyable introduction to the exciting world of IoT development.

7.2 Future Scope

Looking towards the future, the Tic Tac Toe IoT project holds considerable potential for expansion and enrichment. One avenue for future development lies in enhancing its connectivity features. Integrating cloud-based platforms could open up possibilities for remote gameplay, data storage, and collaborative gaming experiences. Additionally, incorporating online multiplayer functionality could transform the project into a more socially interactive platform. The exploration of wireless communication between Arduino units could lead to the creation of expansive, interconnected gaming scenarios.

CHAPTER 8: REFERENCES

1. Arduino Documentation:
    - Arduino (n.d.). “ Arduino – Software.”.

      https://www.arduino.cc/en/Main/Software
2. LiquidCrystal_I2C Library Documentation:
    - Arduino Playground. (n.d.). "I2C Communications."

      https://playground.arduino.cc/Main/I2C
3. Push Button Tutorial:
    - Arduino Tutorial. (n.d.). "Button." https://www.arduino.cc/en/Tutorial/Button
4. Thinker cad ref:
    - https://www.tinkercad.com/things/fMDCqssVRwU-powerful-snicket/editel?returnTo=%2Fdashboard
5. You Tube ref:
    - https://www.youtube.com/shorts/xnCWfo_HG-U