Project Report

Arduino Tic-tac-toe game

By Andrew Finnerty

Overview:

My project is a user-interface Tic-tac-toe game, the game will be displayed on a breadboard using Leds and buttons for a player to enter their moves. The game will use an Arduino microcontroller to execute the code. The idea was inspired by my Young Scientist entry, and I will try and integrate elements of the project into this. This project will allow people to play tic-tac-toe against a computer algorithm. The main function of this device is to allow users to play against a computer algorithm. If a player wants to play Tic-tac-toe and doesn't have a sheet of paper and a pen, they can simply use this device. Furthermore, this device can be reset once the game is over and doesn't require a player to keep drawing lines in a copy. It also allows the user to play even if no human opponent is available thus maximizing the user's time efficiency.

Functions of the project:

This device displays a Tic-tac-toe board using Leds and allows a user to play against a computer algorithm. There are 9 bicolor Leds for display (red and blue), which are controlled by the game logic. There are 9 pushbuttons which when pressed, input data to the Arduino. The Arduino receives an input from the user (a button pressed), and outputs the move by the Leds. The game logic updates and displays the board, requests an input from the user, requests a move from the computer algorithm and checks if the game is over. If a player wins, the winning combination flashes.

Reflection:

The project I undertook turned out to be more of a challenge than I initially expected. I first wrote the code in python, with the BT Young Scientist in mind. However, the Arduino IDE only supports c++ so I struggled to adapt to the language. I ran many test circuits and programs to familiarize myself with the hardware, new software, and inspire ideas. Initially these projects were very simple, just testing a single LED, or button. However, quite quicky these test projects became complex and resulted in hours of debugging, google searches and rebuilding circuits. I quickly Identified that I needed 27 pins to support the project (18 for LEDs and 9 for buttons), however, the Arduino only has 17 pins. I ordered extension ports to connect to the Arduino with more pins. This brought another host of problems, trying to use the library I downloaded with these ports.

In these first steps of the project, I struggled massively with the knowledge and understanding required to write the code. Although I designed the flowchart, and wrote the code in python, writing in C++ and the Arduino IDE was difficult. I used a certain chatbot to help translate some of the functions over. I wrote a very primitive code in c++ which got the user input and output the voltage to the LEDs. This

code worked but was hard to fit with the rest of the game logic. It was recommended to me to use a structure to organize the pins for the extension ports. When I finally put the full thing together it was over 500 lines long! After some simplification involving arrays and structures, and functions, the code became a lot neater.

The circuit also caused me many problems. I started by experimenting with very small single component circuits, with the help of an Arduino book. This was working very well when I increased the scale until I added the extension ports. It appeared there wasn't enough power going through the extension ports to power both the LEDs and the buttons. After hours spent debugging, I concluded it had to be the circuit, not the code causing problems. However, the circuit had previously worked fine, leading me to believe that it was a lack of power problem. This encouraged me to change the design from the standard pulldown model, and instead to use a pullup model. This simplified the circuit and allowed me to put all the LEDs on the extension ports, and buttons on the Arduino. Designing the new circuit was also a challenge as there were few examples online. I did find some inspiration online, and set-up the circuit in the simplest way for me.

I struggled with time management throughout this project as everything seemed to take much longer than I expected. A large portion of this time was spent writing small test programs, debugging, and rewiring circuits, trying to find mysterious bugs. These included some c++ quirks, such as when an array is called it makes a copy instead of accessing the array from memory and misplaced brackets.

I pretty much followed my plan exactly with some deviations for unexpected errors, despite taking far longer than I expected. The flowchart I proposed at the start of the project was implemented in my final project. This helped me keep structure to my code and helped me stay on track.

Reflection on planning versus implementation.

- **What parts of the project worked out as expected?**

Initially I designed the code in python which was relatively straight-forward, I had a main function which set the grid size and looped through while the game was still on. I designed functions to create and display the grid, to get the user input, the computer input, and check if the game was won or drawn. I used top-down modeling as represented in my flowchart, passing each action off to a function.

- **What parts of the project did not work out as expected?**

Coding the Arduino didn't work out as expected, initially there was the challenge of working with the c++ syntax. I ran many test programs to familiarize myself with the syntax, input and output pins, and the hardware. There weren't enough pins on the Arduino board to support all the Leds and buttons, so I had to get an extension with more pins. Downloading the library for the extension brought more surprises and was confusing labeling of pins. Mislabeling pins became the bane of my existence for a few days and caused me severe headaches as to why the LEDs and buttons weren't working. As mentioned earlier, the pull-down design resulted in a lack of power and was unable to support the LEDs and buttons. For this reason, I changed to a pull-up model, which worked much better.

- **What were the biggest challenges of the project?**

There were many problems I encountered, largely due to being unfamiliar with the c++ language. This caused many syntax and logic errors, resulting in a large portion of my time spent on the project debugging. There also were not enough pins on the Arduino, to support 9 Leds and 9 buttons. I initially tried charlieplexing strategies to reduce the number of pins, however this proved too time consuming and complex, so I purchased an Arduino extension boards (pfc8574). Another challenge I faced was calling the pins on the extension board. The library I downloaded for the expansion board added another layer of complexity to the syntax.

- **Given more time to develop the project and your coding experience, what would you change or improve about your project?**

Given more time I would like to have added more advanced computer algorithms and AIs, a more user-oriented display, and other options for the user.

The AI algorithms I would have liked to have added are: a brute force algorithm which pre-calculates all positions in a table database format, a simple AI algorithm similar to the one I displayed, and an advanced learning AI model. It would be very interesting to compare their playing strengths, computational costs and other elements.

I would have liked to have made the display much more user-oriented, with an LCD screen to display messages to the user. I also wanted to add more buttons to allow the user to change the mode of the game. This would make it more interactive.

Finally, I would have liked to have added other options for the user including playing on a larger grid, giving the option to play against another human opponent or watch the computer play against itself.

Complete project code:

(In python):

1. **Tictactoe:** tictactoe.py
2. Create Grid: createGrid.py
3. Display Grid: displayGrid.py
4. User Input: userInput.py
5. Computer Input (random): randomCompMove.py
6. Check for a win: checkActive.py
7. Check for a draw: checkDraw.py

See bottom of document for Arduino Code

(In Arduino IDE):

```
#include "Arduino.h"
#include "PCF8574.h"

#define GRID_SIZE 3
```

```
struct ledControl {
  int redpin;
  PCF8574 *redpcf;
  bool redard;
  int bluepin;
  PCF8574 *bluepcf;
  bool blueard;
  int buttonpin;
  PCF8574 *buttonpcf;
  bool buttonard;
};

typedef char gridColumn[GRID_SIZE];

bool isButtonPressed(ledControl *led) {
  if( led->buttonard ) {
    return digitalRead(led->buttonpin) == LOW;
  }
  return led->buttonpcf->digitalRead(led->buttonpin) == LOW;
}

void setRedOn(ledControl *led) {
  if( led->redard ) {
    digitalWrite(led->redpin, HIGH);
  } else {
    led->redpcf->digitalWrite(led->redpin, HIGH);
  }
}

void setBlueOn(ledControl *led) {
  if( led->blueard ) {
    digitalWrite(led->bluepin, HIGH);
  } else {
    led->bluepcf->digitalWrite(led->bluepin, HIGH);
  }
}

void setRedOff(ledControl *led) {
  if( led->redard ) {
    digitalWrite(led->redpin, LOW);
  } else {
    led->redpcf->digitalWrite(led->redpin, LOW);
  }
}

void setBlueOff(ledControl *led) {
  if( led->blueard ) {
    digitalWrite(led->bluepin, LOW);
  } else {
    led->bluepcf->digitalWrite(led->bluepin, LOW);
  }
}

void setRedOutput(ledControl *led) {
  if( led->redard ) {
    pinMode(led->redpin, OUTPUT);
  } else {
    led->redpcf->pinMode(led->redpin, OUTPUT);
  }
}
```

```
void setBlueOutput(ledControl *led) {
 if( led->blueard ) {
  pinMode(led->bluepin, OUTPUT);
 } else {
  led->bluepcf->pinMode(led->bluepin, OUTPUT);
 }
}

void setButtonInput(ledControl *led) {
 if( led->buttonard ) {
  pinMode(led->buttonpin, INPUT_PULLUP);
 } else {
  led->buttonpcf->pinMode(led->buttonpin, INPUT_PULLUP);
 }
}

//////////////

ledControl grid[GRID_SIZE][GRID_SIZE];

PCF8574 pcf8574_1(0x20,A1,A0);
PCF8574 pcf8574_2(0x21,A1,A0);
PCF8574 pcf8574_3(0x22,A1,A0);
PCF8574 pcf8574_4(0x23,A1,A0);

///////////////

bool checkForWin(gridColumn game[], int gridSize, int row, int col) {
  char userValue = game[row][col];

  // horizontal check
  int hCount = 0;
  for (int c = 0; c < gridSize; c++) {
    if (game[row][c] == userValue) {
      hCount++;
    }
  }
  if (hCount == gridSize) {
    return true;
  }

  // vertical check
  int vCount = 0;
  for (int r = 0; r < gridSize; r++) {
    if (game[r][col] == userValue) {
      vCount++;
    }
  }
  if (vCount == gridSize) {
    return true;
  }

  // slope == 1
  bool diagonalOne = (row == col);
  // slope == -1
  bool diagonalTwo = (row == (gridSize - 1) - col);

  // diagonal check
  if (diagonalOne) {
    int countOne = 0;
```

```
      for (int i = 0; i < gridSize; i++) {
        if (game[i][i] == userValue) {
          countOne++;
        }
      }
      if (countOne == gridSize) {
        return true;
      }
    }

    // diagonal check
    if (diagonalTwo) {
      int countTwo = 0;
      for (int i = 0; i < gridSize; i++) {
        if (game[(gridSize - 1) - i][i] == userValue) {
          countTwo++;
        }
      }
      if (countTwo == gridSize) {
        return true;
      }
    }

    return false;
}

// 2. Comp Input
bool compInput(gridColumn game[], int gridSize, int *row, int *col) {
  for (int r = 0; r < gridSize; r++) {
    for (int c = 0; c < gridSize; c++) {
      if (game[r][c] == '_') {
        *row = r;
        *col = c;
        Serial.print("Computer: ");
        Serial.print(r);
        Serial.print(",");
        Serial.println(c);
        return true;
      }
    }
  }
  return false;
}

bool compRand(gridColumn game[], int gridSize, int *row, int *col) {
  while(true) {
    int r = random(gridSize);
    int c = random(gridSize);

    if (game[r][c] == '_') {
      *row = r;
      *col = c;
      Serial.print("Computer: ");
      Serial.print(r);
      Serial.print(",");
      Serial.println(c);
      return true;
    }
  }
}
```

```
bool checkDraw(gridColumn game[], int gridSize) {
  int qtyOfBlanks = 0;
  for (int r = 0; r < gridSize; r++) {
    for (int c = 0; c < gridSize; c++) {
      Serial.print("LookinAt: "); Serial.print(r); Serial.print(","); Serial.print(c); Serial.print(" - "); Serial.println(game[r][c]);
      if( game[r][c] == '_' ) {
        Serial.print("BlankAt: "); Serial.print(r); Serial.print(","); Serial.println(c);
        qtyOfBlanks += 1;
      }
    }
  }

  Serial.print("QuantityOfBlanks: "); Serial.println(qtyOfBlanks);
  if (qtyOfBlanks == 0) {
     return true;
  } else {
     return false;
  }
}

// Tic-tac-toe
void setup() {
  // Initialize your grid, buttons, and LEDs here
  grid[0][0] = {P7, &pcf8574_1, false, P6, &pcf8574_1, false, P2, &pcf8574_1, true};
  grid[0][1] = {P5, &pcf8574_1, false, P4, &pcf8574_1, false, P3, &pcf8574_1, true};
  grid[0][2] = {P3, &pcf8574_1, false, P2, &pcf8574_1, false, P4, &pcf8574_1, true};
  grid[1][0] = {P1, &pcf8574_1, false, P0, &pcf8574_1, false, P5, &pcf8574_1, true};
  grid[1][1] = {P7, &pcf8574_2, false, P6, &pcf8574_2, false, P6, &pcf8574_1, true};
  grid[1][2] = {P5, &pcf8574_2, false, P4, &pcf8574_2, false, P7, &pcf8574_1, true};
  grid[2][0] = {P3, &pcf8574_2, false, P2, &pcf8574_2, false, 8, &pcf8574_1, true};
  grid[2][1] = {P1, &pcf8574_2, false, P0, &pcf8574_2, false, 9, &pcf8574_1, true};
  grid[2][2] = {13, &pcf8574_1, true, 12, &pcf8574_1, true, 10, &pcf8574_1, true};

  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(1000);

  for(int i = 0; i < GRID_SIZE; i++ ) {
    for(int j = 0; j < GRID_SIZE; j++ ) {
      ledControl *led = &grid[i][j];
      setRedOutput(led);
      setBlueOutput(led);
      setButtonInput(led);
    }
  }

  if (pcf8574_1.begin()){
    Serial.println("OK1");
  }else{
    Serial.println("KO1");
  }

  if (pcf8574_2.begin()){
    Serial.println("OK2");
  }else{
    Serial.println("KO2");
  }

  if (pcf8574_3.begin()){
    Serial.println("OK3");
  }else{
```

```
   Serial.println("KO3");
  }

  if (pcf8574_4.begin()){
   Serial.println("OK4");
  }else{
   Serial.println("KO4");
  }

  for(int i = 0; i < GRID_SIZE; i++ ) {
   for(int j = 0; j < GRID_SIZE; j++ ) {
    ledControl *led = &grid[i][j];
    setRedOff(led);
    setBlueOff(led);
   }
  }
}

void clearGame(gridColumn game[], int gridSize ) {
  for(int i = 0; i < gridSize; i++ ) {
   for(int j = 0; j < gridSize; j++ ) {
    game[i][j] = '_';
   }
  }
}

void displayGrid(gridColumn game[], int gridSize) {
  for(int i = 0; i < gridSize; i++ ) {
   for(int j = 0; j < gridSize; j++ ) {
    if( game[i][j] == 'X' ) {
     setRedOn(&grid[i][j]);
     continue;
    }
    if( game[i][j] == 'O' ) {
     setBlueOn(&grid[i][j]);
     continue;
    }
    setRedOff(&grid[i][j]);
    setBlueOff(&grid[i][j]);
   }
  }
}

bool userInput(gridColumn game[], int gridSize, int *row, int *col) {
  for( int r=0; r<gridSize; r++) {
   for( int c=0; c<gridSize; c++ ) {
    ledControl *led = &grid[r][c];
    if( (game[r][c] == '_') && isButtonPressed(led) ) {
     *row = r;
     *col = c;
     Serial.print("User: ");
     Serial.print(r);
     Serial.print(",");
     Serial.println(c);
     return true;
    }
   }
  }
  delay(100);
  return false;
}
```

```
char game[GRID_SIZE][GRID_SIZE] = {{'_','_','_'},{'_','_','_'},{'_','_','_'}};
char blank[GRID_SIZE][GRID_SIZE] = {{'_','_','_'},{'_','_','_'},{'_','_','_'}};
int gridSize = GRID_SIZE;

void loop() {
-
 clearGame(game, gridSize);
 clearGame(blank, gridSize);

 while (true) {
  displayGrid(game, gridSize);

  // Prompt user to input move
  int row, col;
  bool userPress = userInput(game, gridSize, &row, &col);

  // Check the user move
  if( userPress && game[row][col] == '_' ) {
   game[row][col] = 'X';
   displayGrid(game, gridSize);

   if (checkForWin(game, gridSize, row, col)) {
    Serial.println("Game is Over - you won");
    break;
   }
   if (checkDraw(game, gridSize)) {
    Serial.println("Game is Over - draw");
    break;
   }

   delay(800);

   // Request computer move
   // bool compMoved = compInput(game, gridSize, &row, &col);
   bool compMoved = compRand(game, gridSize, &row, &col);

   if( compMoved == true) {
    game[row][col] = 'O';
    displayGrid(game, gridSize);

    // Check the computer move
    if (checkForWin(game, gridSize, row, col)) {
     Serial.println("Game is Over - you lost");
     break;
    }
    if (checkDraw(game, gridSize)) {
     Serial.println("Game is Over - draw");
     break;
    }
   } else {
    Serial.println("Error: Game is Over - computer could not find a move !");
    break;
   }
  }
 }

 for( int i=0; i<5; i++ ) {
  delay(1000);
  displayGrid(blank, gridSize);
  delay(1000);
```

```
    displayGrid(game, gridSize);
  }
}
```