Arduino Tic Tac Toe

Ethan Luong - eluong2 - eluong2@uic.edu
Antony Ni - ani3- ani3@uic.edu

Abstract

______The project has a human player against an AI in Tic Tac Toe. The human player will make the first move and the AI will use a minimax algorithm to find the best possible moves in accordance. Due to Tic Tac Toe being unwinnable if both players play perfectly, the human player will always lose or draw with the AI. The project uses buttons or a bluetooth communication as inputs for putting down the pieces and the arduino will display the results on 18 LEDs. The LEDs will be powered through a technique called Charlieplexing all on one arduino. Outputs will be a virtual board being displayed by an LED as well as LEDs confirming whose turn it is.

Description:

______The project idea came from a fascination with game theory. Games are interesting because there are a set of actions that players can take, and the more complex a game is the more complex the decision tree becomes (to an exponential extent). The project is going to have the arduinos accurately read inputs from a human player and play accordingly, which is doable with these arduinos due to the simple nature of tic-tac-toe. This is different from AIs playing each other due to humans potentially not moving optimally. Also, we wanted to implement bluetooth communication with the arduino in order to allow this to be played wirelessly or eventually even online.

Communication:

______The communication in this project will be used between the HC-05 Bluetooth Module and the Arduino UNO. The module will be used as a substitute for the buttons on the breadboard and the signals sent from the bluetooth to the arduino will affect the Tic Tac Toe board by navigating it and selecting options. The bluetooth will send up to three different signals telling the cursor on the board to select a space and maneuver around. The arduino will communicate back by updating the bluetooth module.

I/O:

Our input for this project is relatively simple. For our analog input, we have three different push buttons. They will control different things throughout the game but will mainly be used for choosing and selecting a move on the board. In addition to the analog option of input, there is also an option to use a bluetooth input. This is achieved by using the HC-05 bluetooth module with the arduino uno. We have included a program made with Processing that creates a GUI for the player to use to control the board through bluetooth. There are three buttons labeled left, right, and select which correspond to the three analog buttons.

The output for this project is straightforward but more complex. The tic tac toe board is represented by 18 different LEDs and there are an additional 2 LEDs to signify whose turn it is and who won or if there was a draw. Because we used an Arduino Uno with this, we had a limited number of pins to work with. To work around this issue, we used a method called Charlieplexing. How it works is that two LEDs are put on the same rails with the LEDs positive and negatives opposite each other. Two wires from the arduino will connect to the leds, one to each side. When one wire's output is LOW and the other is HIGH one of the LEDs will turn on. When the outputs are reversed the other LED will turn on. To turn them both off, we set the pinModes of the wires to be input. With this method, the number of LEDs we can use goes to n(n-1) where n is the number of pins available. Since we used 5 pins, we were able to use the 20 pins needed.

Original Work:

The original work of this project is that it is the Arduino AI interacting with a real world player as opposed to Arduino AI interacting with another Arduino AI. The bluetooth communication in substitution for the buttons is also original work in how it controls the board. The LED layout of a board through the use of Charlieplexing is also original. The use of buttons and multiple LEDs to navigate the board is done to conserve space so that only one UNO is needed, which hasn't been done with this method.

Building Project:

To start building this project, everything listed in the materials needed section should be acquired first. The wiring for the LEDs should be completed first as this is the bulk of the wiring and can be hard to find mistakes once there are other wires in the way. If a large breadboard is unavailable, a smaller one can be used. What is important is that each pair of LEDs is connected to the correct pair of wires in the correct order. Once the LEDs are wired up, the push buttons can be placed on the breadboard and wired to the arduino. The final step is to connect the HC-05 bluetooth module to the arduino.

Once the physical wiring is completed, a connection with the HC-05 module must be made. If opting to use the provided GUI, the Processing IDE must be installed. Connect to the bluetooth module through the PC with a pin of 1234. Check the modules COM number and change it in the sketch file if needed. Once this is done, both sketch files should be runnable.

Using Project:

The program will run like a normal game of Tic Tac Toe. The player will begin with the first move. The player can choose to either use the buttons or a bluetooth connection for input. If the player chooses to use the buttons as inputs, then the top button will be used for moving the selection to the right, the bottom button will move the selection to the left, and the middle will confirm the button. If the user chooses to use a bluetooth connection, commands can be sent instead of the buttons. 1 corresponds to right, 2 corresponds to left, and 3 corresponds to select.

There is also an included processing sketch for an option to use GUI to send bluetooth commands. After the player makes their move, the AI will make their move. The two LEDS near the bottom signify whose turn it is. This will continue until either the AI wins, the user wins, or there is a tie. If the user wins, the player colored LED on the bottom will stay on. If the AI wins, the AI's color will stay on. If there is a tie, both LEDs will stay on.

Timeline:

Project Contract Completed 10/9/20
LED Board Proof of Concept 10/16/20
LED Board Implementation 10/23/20
Button System Proof of Concept 10/30/20
Button System Implementation 11/4/20
Communication between Arduino and bluetooth Proof of Concept 11/11/20
Communication between Arduino and bluetooth Completion 11/19/20
Completed Project 11/20/20
Troubleshooting 11/21/20
Final Design Documentation 12/3/20

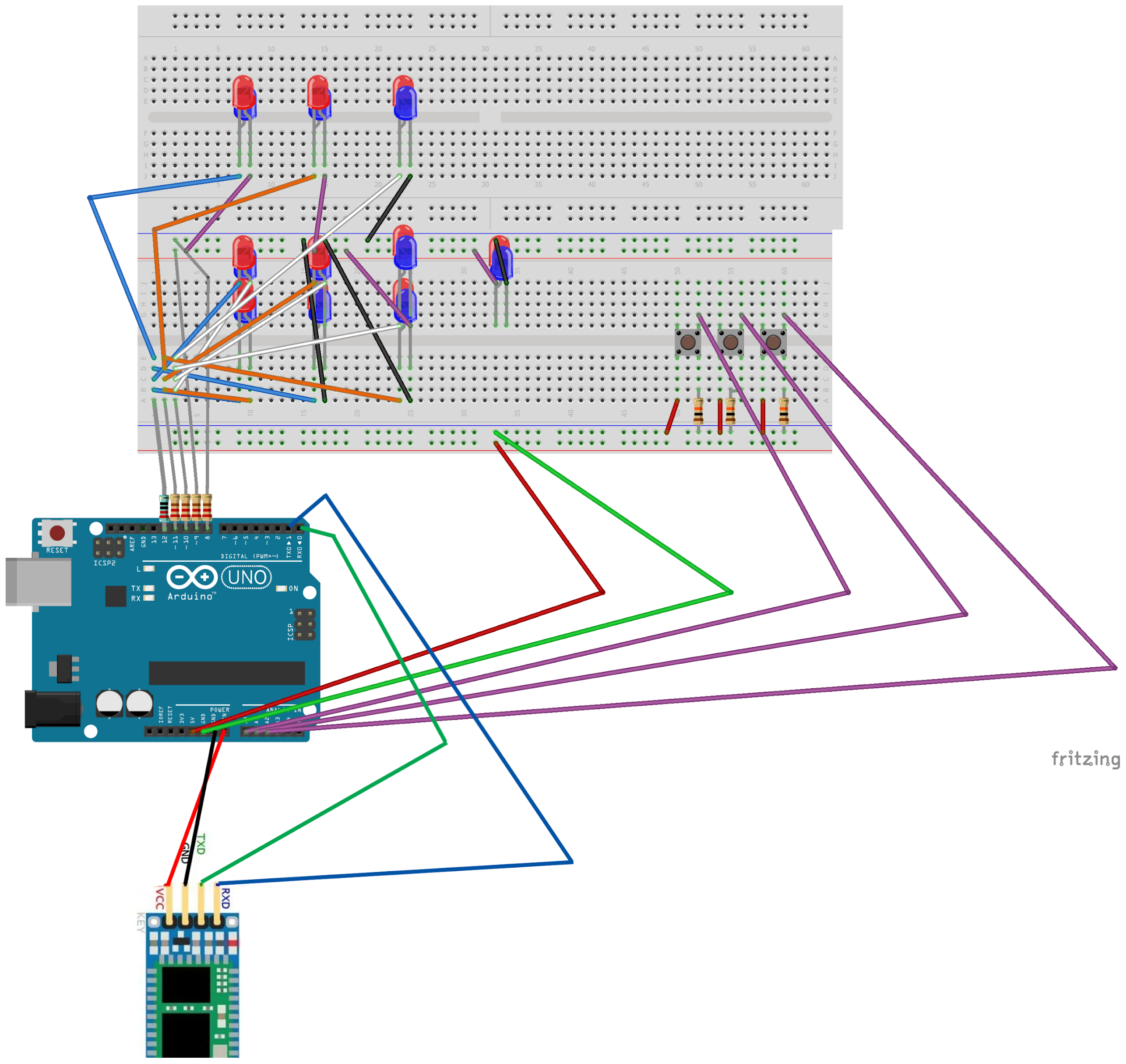
Materials Needed:

20 LEDS - Preferably 10 of one color and 10 of a different color
HC-05 Bluetooth Module
Three push buttons
Five 220 Ohm Resistors
27 Wires
One Arduino Uno
A device capable of a bluetooth connection
One Bread board

References:

1. Minimax Algorithm used for determining the best move for the AI. Utilizes a recursive call to find the best value of a move all the way down to a certain depth. https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/
2. Helpful example of how to alternate the use of minimax when planning out moves: https://runtimeprojects.com/2017/04/artificial-intelligence-on-arduino-an-invincible-tic-tac-toe-player/
3. Reference for learning how to charlieplex the LEDs representing the gameboard .https://www.instructables.com/Charlieplexing-the-Arduino/
4. Article used as an example of how to connect the HC-05 module to the arduino Uno. https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/

Fritzing Diagram



INO Files:

Arduino Sketch

```
//ani3 AI Controlled TicTacToe
// Uses a minimax algorithm to come up with the best moves for the AI to make and is
modified for TTT
//https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/
//https://runtimeprojects.com/2017/04/artificial-intelligence-on-arduino-an-invincible-tic-t
ac-toe-player/



#define A 12
#define B 11
#define C 10
#define D 9
#define E 8
#define left A0
#define right A2
#define select A1
boolean test = false;

#define PIN_COUNT 5
#define PIN_CONFIG 0
#define PIN_STATE 1


#define LED_COUNT 20


int matrix[LED_COUNT][2][PIN_COUNT] = {
    /*
     *  Each row in this matrix respresents the pin modes and pin states for a single LED
     */
    //              PIN_CONFIG                          PIN_STATE
    //    A       B       C      D      E          A     B    C    D    E
    { { OUTPUT, OUTPUT, INPUT, INPUT, INPUT }, { HIGH, LOW, LOW, LOW, LOW } }, // AB 0
    { { OUTPUT, OUTPUT, INPUT, INPUT, INPUT }, { LOW, HIGH, LOW, LOW, LOW } }, // BA 1
    { { OUTPUT, INPUT, OUTPUT, INPUT, INPUT }, { HIGH, LOW, LOW, LOW, LOW } }, // AC 2
    { { OUTPUT, INPUT, OUTPUT, INPUT, INPUT }, { LOW, LOW, HIGH, LOW, LOW } }, // CA 3
    { { OUTPUT, INPUT, INPUT, OUTPUT, INPUT }, { HIGH, LOW, LOW, LOW, LOW } },  // AD 4
    { { OUTPUT, INPUT, INPUT, OUTPUT, INPUT }, { LOW, LOW, LOW, HIGH, LOW } },  // DA 5
    { { OUTPUT, INPUT, INPUT, INPUT, OUTPUT }, { HIGH, LOW, LOW, LOW, LOW } },  // AE 6
    { { OUTPUT, INPUT, INPUT, INPUT, OUTPUT }, { LOW, LOW, LOW, LOW, HIGH } },  // EA 7
    { { INPUT, OUTPUT, OUTPUT, INPUT, INPUT }, { LOW, HIGH, LOW, LOW, LOW } }, // BC 8
    { { INPUT, OUTPUT, OUTPUT, INPUT, INPUT }, { LOW, LOW, HIGH, LOW, LOW } }, // CB 9
    { { INPUT, OUTPUT, INPUT, OUTPUT, INPUT }, { LOW, HIGH, LOW, LOW, LOW } },  // BD 10
    { { INPUT, OUTPUT, INPUT, OUTPUT, INPUT }, { LOW, LOW, LOW, HIGH, LOW } },  // DB 11
    { { INPUT, OUTPUT, INPUT, INPUT, OUTPUT }, { LOW, HIGH, LOW, LOW, LOW } },  // BE 12
    { { INPUT, OUTPUT, INPUT, INPUT, OUTPUT }, { LOW, LOW, LOW, LOW, HIGH } },  // EB 13
    { { INPUT, INPUT, OUTPUT, OUTPUT, INPUT }, { LOW, LOW, HIGH, LOW, LOW } }, // CD 14
    { { INPUT, INPUT, OUTPUT, OUTPUT, INPUT }, { LOW, LOW, LOW, HIGH, LOW } }, // DC 15
    { { INPUT, INPUT, OUTPUT, INPUT, OUTPUT }, { LOW, LOW, HIGH, LOW, LOW } }, // CE 16
    { { INPUT, INPUT, OUTPUT, INPUT, OUTPUT }, { LOW, LOW, LOW, LOW, HIGH } },  // EC 17
    { { INPUT, INPUT, INPUT, OUTPUT, OUTPUT }, { LOW, LOW, LOW, HIGH, LOW } }, // DE 18
    { { INPUT, INPUT, INPUT, OUTPUT, OUTPUT }, { LOW, LOW, LOW, LOW, HIGH } } // ED 19
};

void turnOn( int led ) {
    // set all the pin modes
    pinMode( A, matrix[led][PIN_CONFIG][0] );
    pinMode( B, matrix[led][PIN_CONFIG][1] );
    pinMode( C, matrix[led][PIN_CONFIG][2] );
    pinMode( D, matrix[led][PIN_CONFIG][3] );
    pinMode( E, matrix[led][PIN_CONFIG][4] );
    // set all the pin states
    digitalWrite( A, matrix[led][PIN_STATE][0] );
    digitalWrite( B, matrix[led][PIN_STATE][1] );
    digitalWrite( C, matrix[led][PIN_STATE][2] );
    digitalWrite( D, matrix[led][PIN_STATE][3] );
    digitalWrite( E, matrix[led][PIN_STATE][4] );
}


void setup(){
  Serial.begin(9600);
  pinMode(left, INPUT);
  pinMode(right, INPUT);
  pinMode(select, INPUT);

}


//Displays the current gameboard with LEDs
void displayBoard(int brd[9], int curSelection, boolean playerTurn) {
  for(int i = i; i < 9; ++i){
    if(brd[i] == 1){
      turnOn((i * 2) + 1);
      delay(1);
    }else if(brd[i] == -1){
      turnOn(i * 2);
      delay(1);
    }
  }
  if(curSelection != -1){
    turnOn(curSelection);
    delay(1);
  }
  if(playerTurn){
    turnOn(18);
  }
  else{
    turnOn(20);
  }
}



//checks the board if there is a win on the screen, which are the three across, three down,
and 2 diagonals
int winCheck(int brd[9]){
  Serial.println("Win");
   int winBoards [8][3] = {{0,1,2},{3,4,5},{6,7,8},{0,3,6},{1,4,7},{2,5,8},{0,4,8},{2,4,6}};
   for (int i = 0;  i < 8; i++){ //if the board matches a winning set, return the piece
owned by player
     if ((brd[winBoards[i][0]] != 0) && (brd[winBoards[i][1]] == brd[winBoards[i][2]]) &&
(brd[winBoards[i][0]] == brd[winBoards[i][1]]))
       return  brd[winBoards[i][0]];
   }
       return 0; //no winning board
}



int mm(int board[9], int turn, int left){//minimax algorithm
  Serial.println("Beginning of minimax");
  int win = winCheck(board);
  if (win !=0)
    return win * turn;
  int action = -1;
  int rating = -2; //score of the move
  for (int i = 0; i<9; i++){
      //checks if legal move
      if (board[i] ==0){
        board[i] = turn; //x or o
        int theoScore = 0; //theoretical score
        if (left < 8)
          theoScore = -1*mm(board,-1*turn,left+1);
        if (theoScore > rating){
          rating = theoScore;
          action = i;}
      }
  }
   if (action == -1)
     return 0 ;//invalid move
   return rating;
  }
//Function for Ai's turn
void ai(int board[9]){
    int brd[9];
    for(int i = 0; i < 9; ++i){
      brd[i] = board[i];
    }
    int action = -5;
    int rating = -2;
```

```
     Serial.println("AI");
     for (int i =0; i<9; i++){
       Serial.println("AI for loop");
       if (brd[i] ==0){
         brd[i] = 1;
         int theoScore = -mm(brd,-1,0);
         brd[i] = 0; //reset after calling
         if (theoScore > rating){
           rating = theoScore;
           action = i;      }
       }
     }
     board[action] = 1;}
//Gets input from player with option for bluetooth input
void human(int brd[9]){ //move based on button press
    boolean selected = false;
    int curSelection = 0;
    int state = -1;
    boolean buttonPressed = false;
    int prevButton = 0;

    do {
     while(brd[curSelection] != 0){
       if(prevButton == 0){
         curSelection++;
         if(curSelection > 8){
           curSelection = 0;
         }
       }
       else{
         curSelection--;
         if(curSelection < 0){
           curSelection = 8;
         }
       }
     }
     if(Serial.available() > 0){
     state = Serial.read();
     }
       if(digitalRead(right) == HIGH || state == '1'){
         if(!buttonPressed){
           curSelection++;
           buttonPressed = true;
           state = -1;
           prevButton = 0;
         }
       }
       if(digitalRead(left) == HIGH || state == '2'){
         if(!buttonPressed){
           curSelection--;
           buttonPressed = true;
           state = -1;
```

```
          prevButton = 1;
        }
      }
      if(digitalRead(select) == HIGH || state == '3'){
        if(!buttonPressed){
          selected = true;
          buttonPressed = true;
          state = -1;
        }
      }
      if(digitalRead(right) == LOW && digitalRead(left) == LOW && digitalRead(select) ==
LOW){
        buttonPressed = false;
      }
      if(curSelection > 8){
        curSelection = 0;
      }
      if(curSelection < 0){
        curSelection = 8;
      }
      displayBoard(brd, curSelection * 2, true);
   } while (selected == false);

   brd[curSelection] = -1;
   displayBoard(brd, -1, false);
   delay(1000);
}

int winner = 0;
int gameBoard[9] = {0,0,0,0,0,0,0,0,0};
void loop(){
  for (int i = 0; i < 9 ; i++){
    winner = winCheck(gameBoard);
    if(winner != 0){
      break;
    }
    if (i %2  == 0){//alternating turns
      ai(gameBoard);
      displayBoard(gameBoard, -1, false);
  }
    else {

      displayBoard(gameBoard, -1, true);
      human(gameBoard);
    }
  }
  int start = 0;

  if(winner == 1){
    turnOn(19);
    if(digitalRead(select) == HIGH || Serial.read() == 3){
      winner = 0;
      for(int i = 0; i < 9; ++i){
        gameBoard[i] = 0;
      }
      delay(1000);
    }
  }else if(winner == -1){
    turnOn(18);
    if(digitalRead(select) == HIGH || Serial.read() == 3){
      winner = 0;
      for(int i = 0; i < 9; ++i){
        gameBoard[i] = 0;
      }
      delay(1000);
    }
  }


}
```

Processing Sketch for Bluetooth GUI

```
import processing.serial.*;
Serial myPort;

void setup(){
  size(450, 300);
  myPort = new Serial(this, "COM5", 9600); // Starts the serial
communication
}

boolean clicked = false;
void draw(){
  background(237, 240, 241);
  fill(20, 160, 133); // Green Color
  stroke(33);
  strokeWeight(1);
  rect(75, 100, 100, 50, 10);  // Left Button
  rect(275, 100, 100, 50, 10); // Right Button
  rect(175,200,100,50,10);
  fill(255);

  textSize(32);
  text("Left",95, 135);
  text("Right", 280, 135);
  text("Select", 180, 235);
  textSize(24);
  fill(33);
  textSize(30);
  textSize(16);



  if(mousePressed && mouseX>75 && mouseX<175 && mouseY>100 &&
mouseY<150){
    if(clicked == false ){
      myPort.write('2');
      stroke(255,0,0);
      strokeWeight(2);
      noFill();
      rect(75, 100, 100, 50, 10);
      clicked = true;
    }

  }
  else if(mousePressed && mouseX>275 && mouseX<375 && mouseY>100 &&
mouseY<150){
    if(clicked == false){
      myPort.write('1');
      stroke(255,0,0);
      strokeWeight(2);
      noFill();
      rect(275, 100, 100, 50, 10);
      clicked = true;
    }
  }
  else if(mousePressed && mouseX > 175 && mouseX < 275 && mouseY >
200 && mouseY < 250){
    if(clicked == false){
      myPort.write('3');
      stroke(255,0,0);
      strokeWeight(2);
      noFill();
      rect(175, 200, 100, 50, 10);
      clicked = true;
    }
  }
  else {
      clicked = false;
  }
}
```