Jose Figueroa
Matthew Provotorov
YiQi Wu

Advanced Mechatronics
05/05/2020

Tic Tac Toe: Final Project

Background:

Major cities entered lock downs and quarantines as the coronavirus expanded throughout the globe. These efforts slowed down the spread of the virus, however, they did not stop it from becoming a pandemic. The pandemic forced governments to close businesses to ensure the well-being of their citizens. This put industries in difficult financial situations because they struggle to generate revenue. However, one industry does not share the same financial situation as the others. The entertainment industry flourished despite the pandemic. Netflix's stock went up by 30% and gained millions of subscribers since the beginning of the quarantine. Video games also benefited from the pandemic. CNBC wrote an article describing how video games sales soar due to the pandemic. They also reported that Activision Blizzard's Call of Duty: Warzone gained 30 million players in 10 days because people try to socialize through online based games. This shows that the entertainment industry remains strong even during the quarantine.

Figure 1

Objective:

Team 7 did not want to miss the opportunity to join this flourishing industry, therefore, the decision to join the market was made. The project's objective is to develop a game by implementing mechanical, electronical, and software engineering together. The game will also utilize two microcontrollers to enhance user interaction and achieve better performance. The game's name is Tic Tac Toe and aims to be engaging and competitive so people can have fun and forget all the craziness happening right now.

Methods & Plans:

Tic Tac Toe works by drawing Xs and Os in a 3x3 grid. Two players choose between one of the given symbols (X or O) and then the game starts. Players take turns to draw their symbols, but they can only draw once per turn. The player who first draws his/her respective symbol horizontally, vertically, or diagonally wins the game. The following picture depicts how the 3x3 grid looks after a player won the game.

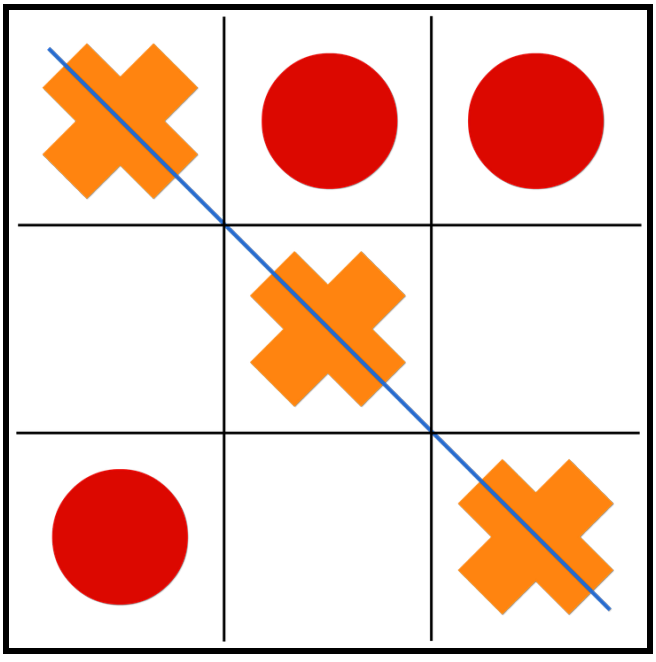


Figure 2

Tic Tac Toe is usually played on paper, however, this project aims to utilize different engineering disciplines and combine them together. Therefore, the following materials are necessary:

- Arduino Uno
- Propeller Activity Board
- Push buttons
- LCD
- LEDs
- 7-Segment Display

The previously mentioned materials come together and follow the instructions given in the codes to create the game. The game starts by randomly selecting who goes first. This is accomplished by using the random function which is given minimum and maximum values (1 and 3 respectively). The random function chooses a value within the given parameters and places it in the variable “initial”. The “initial” variable is used as a condition for an if statement that sets up the LCD to display the proper message depending on who goes first. Then the players can choose between 9 positions and the positions are visible in a 3x3 matrix. The 3x3 matrix consists of 9 LEDs (initially turned off) and they are connected to the Arduino Uno.

Players can swap between different positions by using a push button and there is a second push button that confirms the desired position. Both push buttons use pull up resistors which invert their states and that is used as conditions in if statements. The first if statement says if the button is pressed (low) add 1 to the iteration variable “num” and display it in the LCD. However, “num” is reset to zero if it happens to go beyond 9. That guarantees that the players only see numbers from 1 to 9. The second if statement says if the second push button is pressed (low) turn on the LED with the same number as “num”. Additionally, 1 is subtracted from “num” within the “winarray” to select a position (within the array) and change its value. The “winarray” was initialized at the beginning of the code, but its main function comes later when triggering the winning condition. This process occurs in the Arduino Uno, but the Propeller goes through a similar process. The propeller is where player 2 inputs his desired position and follows similar conditions to swap and confirm his desired position. This leads to the next point which is the communication between the microcontrollers.

The communication between the two microcontrollers is possible through libraries. Arduino Uno uses the “Software.Serial” library which allows digital pins to access serial communication. It lets them receive and transmit information as if they were the RX and TX pins. Similarly, the propeller uses the “fdserial” header to transmit and receive messages from other microcontrollers. The Arduino possesses the variable “num2” to exclusively take input from the Propeller and the Propeller has an exclusive variable named “num” to send the information to the Arduino Uno. This is what allows player 2 to participate because player 2’s input is taken only by the propeller.

The two players will continue to take turns and choose their desired positions until the winning condition is triggered. The condition consists of 8 sub conditions that say the same three symbols must be placed horizontally, vertically or diagonally to win the game. This is when the “winarray” comes into play because it stores the symbols of player 1 and 2 on their chosen positions. The LCD will display “Player 1 Wins” or “Player 2 Wins” depending on who met the winning condition first. However, there is a possibility that there will be no winner and the game ends up in a tie. This scenario is addressed by the following if statement that says if the variable “turn” equals 10 then the LCD displays “It’s a tie”. “Turn” initially starts at 0, but 1 is added to “turn” every time a player confirms the position they want to place their symbol. This means “turn” is in the second if statements where players pressed the second push button.

Everything up to this point, except selecting which player goes first, is inside an if statement. The if statement has the condition to end the game when the variable “over” becomes true. “Over” is a boolean variable and its initial value is false, but the winning or tieing condition changes that value to true. That ends the game and to play again, one must press the reset button on the Arduino Uno. Additionally, the symbols are preselected for each player. Player 1 has the static LEDs which present Xs while player 2 has the blinking LEDS that represent Os. This is how players know their symbols during the game.

Code:

Arduino Uno Code

```
#include<SoftwareSerial.h> /
SoftwareSerial mySerial(10, 9);                 //assign RX pin 10, TX pin 9
int led1 = 0;                                   // LED 1 to pin0
int led2 = 1;                                   // LED 2 to pin1
int led3 = 2;                                   // LED 3 to pin2
int led4 = 3;                                   // LED 4 to pin3
int led5 = 4;                                   // LED 5 to pin4
int led6 = 5;                                   // LED 6 to pin5
int led7 = 6;                                   // LED 7 to pin6
int led8 = 7;                                   // LED 8 to pin7
int led9 = 8;                                   // LED 9 to pin8
int button1 = 12;                               // button1 to pin 12
int button2 = 11;                               // button2 to pin 11
boolean button1state = 1;                       // State of button1
boolean button2state = 1;                       // State of button2
int winarray[] = {0, 0, 0, 0, 0, 0, 0, 0, 0};   //array to collect players' moves
int ledarray[] = {A1, A1, A1, A1, A1};          //player2' moves
int ledState = LOW;                             // ledState used to set the LED, used for LED
blinking
unsigned long previousMillis = 0;               // will store last time LED was updated
// constants won't change, used for LED blinking
const long interval = 1000;                     // interval at which to blink (milliseconds), used for
LED blinking
int turn = 0;                                   // players' turns
int num = 0;                                    //player1 choosing led's number
int num2;                                       //player2 choosing led's number
int p2 = 0;                                     // count number for blinkled function
int count = 0;                                  //count number
int initial;                                    //for which player to go first
```

```
boolean over = false;                          //game over condition

void setup()
{
  mySerial.begin(9600);                        // baud rate 9600
  pinMode(button1, INPUT_PULLUP);              // Setting button1 as Pullup button input
  pinMode(button2, INPUT_PULLUP);              // Setting button2 as Pullup button input
  pinMode(led1, OUTPUT);                       // Setting led1's pin to output
  pinMode(led2, OUTPUT);                       // Setting led2's pin to output
  pinMode(led3, OUTPUT);                       // Setting led3's pin to output
  pinMode(led4, OUTPUT);                       // Setting led4's pin to output
  pinMode(led5, OUTPUT);                       // Setting led5's pin to output
  pinMode(led6, OUTPUT);                       // Setting led6's pin to output
  pinMode(led7, OUTPUT);                       // Setting led7's pin to output
  pinMode(led8, OUTPUT);                       // Setting led8's pin to output
  pinMode(led9, OUTPUT);                       // Setting led9's pin to output
  mySerial.write(17);                          // Turn backlight on
  mySerial.write(12);                          //LCD, The cursor is moved to position 0 on
// line 0 and the entire display is cleared.
  delay(5);
  mySerial.print("Tic-Tac-Toe");               //print on LCD
  randomSeed(analogRead(A0));                  //initializes the pseudo-random number
generator
  delay(3000);                                 //shows game title for 3s
}

void loop()
{
  unsigned long currentMillis = millis();      //Returns the number of milliseconds since
// the Arduino board began running the current program
  button1state = digitalRead(button1);         //read if button1 is pressed
  button2state = digitalRead(button2);         //read if button2 is pressed
  initial = random(1, 3);                      //random which player goes first
```

```
  if (initial == 1 && turn == 0)                    //Player 1 goes first scenario
  {
        mySerial.write(12);                         //LCD, The cursor is moved to position 0 on
// line 0 and the entire display is cleared.
        delay(5);
        mySerial.print("PLAYER 1's turn");          //print on LCD
        count = 0;                                  //player1's move condition
        turn += 1;
        delay(500);
  }
  else if (initial == 2 && turn == 0)               //Player 2 goes first scenario
  {
        mySerial.write(12);                         //LCD, The cursor is moved to position 0 on
line 0 and the entire display is cleared.
        delay(5);
        mySerial.print("PLAYER 2's turn");          //print on LCD
        count = 1;                                  //player2's move condition
        turn += 1;
        delay(500);
  }
  if (over != true)
  {
        if (button1state == 0 && count == 0)        //Player1's left button, changing numbers
        {
        num += 1;
        if (num > 9)
        {
        num = 1;
        }
        mySerial.write(12);                         //LCD, The cursor is moved to position 0 on
// line 0 and the entire display is cleared.
        delay(5);
        mySerial.print("PLAYER 1 chooses pin");     //print on LCD
```

```
        mySerial.print(num);                    //print led num on LCD so player 1 can see
which led he is choosing
        delay(300);
        }
        if (button2state == 0 && count == 0)    //player1's right OK button
        {
        count = 1;
        turn += 1;
        mySerial.write(208);                    // 1/64 notes
        mySerial.write(229);                    // pitch
        turnonled(num);                         // call turnonled function
        winarray[num - 1] = 1;                  // store player 1's moves
        num = 0;
        mySerial.write(12);                     //LCD, The cursor is moved to position 0 on
line 0 and the entire display is cleared.
        delay(5);
        mySerial.print("PLAYER 2's turn");      //print on LCD
        delay(500);
        }
        if ( mySerial.available() && count == 1)    //player2's serial data
        {
        mySerial.write(12);                     //LCD, The cursor is moved to position 0 on
line 0 and the entire display is cleared.
        delay(5);
        mySerial.print("PLAYER 1's turn");      //print on LCD
        turn += 1;
        count = 0;
        num2 = mySerial.parseInt();             //returns the first valid (long) integer number
from the serial buffer
        p2moves(num2);                          //call p2moves
        winarray[num2 - 1] = 2;                 //store player 2's moves
        mySerial.write(208);                    // 1/64 notes
        mySerial.write(229);                    // pitch
```

```
        delay(500);
        }
        if ((winarray[0] == 1 && winarray[1] == 1 && winarray[2] == 1) ||
        (winarray[3] == 1 && winarray[4] == 1 && winarray[5] == 1) ||
        (winarray[6] == 1 && winarray[7] == 1 && winarray[8] == 1) ||
        (winarray[0] == 1 && winarray[3] == 1 && winarray[6] == 1) ||
        (winarray[1] == 1 && winarray[4] == 1 && winarray[7] == 1) ||
        (winarray[2] == 1 && winarray[5] == 1 && winarray[8] == 1) ||
        (winarray[0] == 1 && winarray[4] == 1 && winarray[8] == 1) ||
        (winarray[2] == 1 && winarray[4] == 1 && winarray[6] == 1)  )
//8 player 1's winning condition
        {
        mySerial.write(12);
        delay(5);
        mySerial.print("Player 1 Wins");
        over = true;                                                    //set Game over
condition to true
        }
        else if ((winarray[0] == 2 && winarray[1] == 2 && winarray[2] == 2) ||
        (winarray[3] == 2 && winarray[4] == 2 && winarray[5] == 2) ||
        (winarray[6] == 2 && winarray[7] == 2 && winarray[8] == 2) ||
        (winarray[0] == 2 && winarray[3] == 2 && winarray[6] == 2) ||
        (winarray[1] == 2 && winarray[4] == 2 && winarray[7] == 2) ||
        (winarray[2] == 2 && winarray[5] == 2 && winarray[8] == 2) ||
        (winarray[0] == 2 && winarray[4] == 2 && winarray[8] == 2) ||
        (winarray[2] == 2 && winarray[4] == 2 && winarray[6] == 2)  )                    //8
player 2's winning condition
        {
        mySerial.write(12);
        delay(5);
        mySerial.print("Player 2 Wins");
        over = true;                                                    //set Game over
condition to true
```

```
        }
        else if (turn == 10)                                    //tie condition
        {
        mySerial.write(12);
        delay(5);
        mySerial.print("It's a tie");
        over = true;                                            //set Game over
condition to true
        }
  }
  if (currentMillis - previousMillis >= interval) {             //Blink Player 2's
LEDS
        // save the last time you blinked the LED
        previousMillis = currentMillis;
        // if the LED is off turn it on and vice-versa:
        if (ledState == LOW) {
        ledState = HIGH;
        } else {
        ledState = LOW;
        }
        for (int i = 0; i < 5; i++)
        {
        digitalWrite(ledarray[i] - 1, ledState); //declaring them as outputs
        }
        // set the LED with the ledState of the variable:
        //digitalWrite(ledPin, ledState);
  }
}                                                       //main function ends


void turnonled(int ledp1) // function to turn player1's LEDs on
{
  digitalWrite(ledp1 - 1, 1); //Turn on LEDs
```

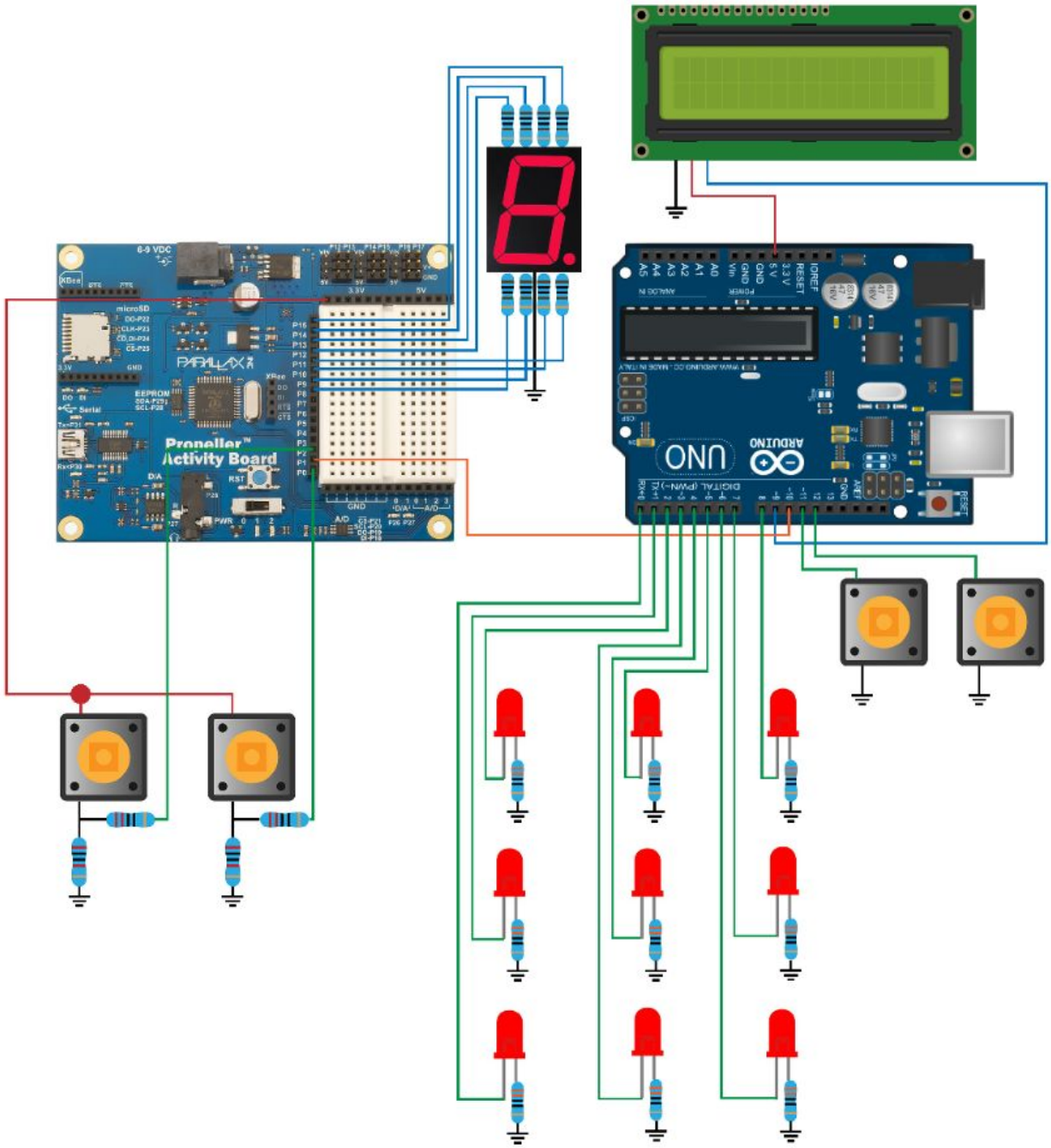
```
}
void p2moves(int ledp2) // function to store player2's chosen LEDs
{
  ledarray[p2] = ledp2;
  p2 += 1;
}
```

Propeller Code

```
#include "simpletools.h" // Include simpletools header
#include "fdserial.h"
fdserial *arduino;
int main() // Main function
{
pause(500);
int button1;
int button2;
int num = 1;
arduino = fdserial_open(-1,1, 0, 9600);// declare serial communication,no Rx pin, Tx pin=1,baud
rate 9600
int NumTable[] = {0b10000100, 0b11010011, 0b11010110, 0b10110100,
0b01110110, 0b01110111, 0b11000100, 0b11110111, 0b11110110}; //7-seg num 1-9
int ShowDigit;
set_directions(15, 8, 0b11111111); // Set P8-P15 as output pins.
set_outputs(15, 8, 0b10000100); // Show 1 as 7-seg initial state
while(1)
{
button1 = input(0); //Pulldown button1 connect to pin 0 as input
button2 = input(2); //Pulldown button2 connect to pin 2 as input
if (button1 ==1) //Click button1 to adjust the number
{
num+=1;
if (num>9)
```

```
{
num = 1;
}
ShowDigit=Lookup(num-1, NumTable);//call lookup function and assgin ShowDigit
set_outputs(15, 8, ShowDigit); //set p8-p15, shows the number on 7 seg
pause(300);
}
if (button2 ==1) //Click button2 to transfer serial data
{
set_outputs(15, 8, 0b10000100);//set 7 seg to show 1
dprint(arduino, "%d",num);//transfer num
pause(500);
num =1;//reset the num back to 1
}
}
}
int Lookup(int Index, int Array[]) // Lookup function
{
return Array[Index]; // Return result
}
```

Circuit Diagram:



Budget:

Quantity	Electronic Part	Cost
1	Arduino Uno	$20.00
1	Propeller Activity Board	$79.00
4	Push Buttons	$0.15
1	LCD	$8.99
9	LEDs	Free
1	7-Segment Display	$0.60
Total		$108.74

Results & Future Improvements:

The result is a fun game that can keep people entertained during this quarantine, however, there are shortcomings that diminish the project's value. For instance, players can not choose their symbol. The symbols are preselected and there is no way to change that which can be a let down for some. Additionally, one must press the reset button on the Arduino Uno to reset the game. This can not happen in the final product because people can mess up the microcontroller if they have access to it. This brings up that the prototype needs a better chasi because the buttons, LCD, microcontroller, and more are all over the place and exposed. Addressing these shortcomings, in future improvements, are a priority for our team because then our product is ready to hit the market.

All in all, the objective was accomplished. A Tic Tac Toe game was built implementing different engineering disciplines and resulted in a fun and competitive game. Although, there are a few shortcomings as mentioned earlier, the game achieves everything proposed in this report. Above all, the Propeller and Arduino Uno successfully communicated with each other. This was one of the main goals because it enhanced the performance and user experience. The product will be able to hit the market as soon as the shortcomings are addressed and hopefully will help people get through this quarantine.

References:

1) https://www.cnbc.com/2020/04/03/video-games-sales-soar-as-coronavirus-leaves-millions-trapped-at-home.html
2) https://metro.co.uk/2020/04/22/netflix-confirms-surge-users-says-coronavirus-will-affect-content-12592273/
3) https://towardsdatascience.com/tic-tac-toe-creating-unbeatable-ai-with-minimax-algorithm-8af9e52c1e7d?gi=349d398ece62