Tic-Tac-Go Final Report
GEEN 3010 - Circuits for Engineers

Sydney Kobak, Bryan Sirner, Yamelit Medina-Lopez and David Price

Abstract

Our classic game that we chose to design in a new light for our GEEN 3010 Circuits for Engineers class is Tic Tac Go. Tic Tac Go is much like the classic game of Tic Tac Toe however, our version we developed has a timed element thereby the “Go”. This game was designed with several requirements in mind, but we designed this game with the idea that sight and touch would be the two human senses that we would target for our audience and that the players would interact with the interface via buttons to select their X or O space as well as start the game. Throughout the entirety of our project, we had to be adaptable and always be able to apply our problem solving skills so that we could produce a successful product.

Project Scope

For our Fall 2021 GEEN 3010 Circuits for Engineers class we were tasked to produce a classic game by applying what we have learned over the course of this semester in conjunction with our mechanical skills. Our personal team goals that we set for ourselves is that we wanted to make a clean and visually enticing version of Tic Tac Toe that showcases our other mechanical engineering skills and the circuits knowledge that we developed over the course of the class. With our personal team goal set, we also had to hit the following deliverables:

- Background
    - User input must affect the game in a meaningful way -- inputs their square selection
    - Each game should include some way for a user to interact with the game (knobs, joystick etc) -- Buttons for players input
    - Game should stimulate at least two human senses -- touch and visuals
    - The game must keep to a set time limit as an indication of the passage of time toward the time limit should be included -- 10 and 20 second timers
    - Game should indicate score or progress toward a goal with some sort of display
    - Entire operation of machine must fit on table top
    - Game can have no more than two power supplies (power wall outlets can be used) -- single wall plug used
    - The finished circuitry must include a PCB that you design specifically for this game. -- Power cable plugs directly into the PCB
    - Safety -- No fires or exposed wires

In addition to the requirements written above, our project needed design plan deliverables that aided in keeping to a timeline (initial idea, informal design plan, prototype, final presentation and final report respectively)

Design Requirements

Beyond the course scoped design requirements our team created some personal design requirements to ensure the end product we all envisioned. Some of these additional requirements for our game for Tic-Tac-Toe were that we didn't want the light to shine between the X and O in one square. This was important to our team because it could confuse gameplay. Additionally our team wanted the game to be fun to play. This is why we made it Tic-Tac-Go, which has speed rounds of 10 or 20 seconds. This speed was designed to force the players to make mistakes simply because they are going fast. Between the speed and fun LED's the game had plenty more to offer than traditional Tic-Tac-Toe. Last, the team added our own design touch with a subtle Star Wars theme. The lights are red and blue, the font is Star Wars font, and R2D2 is engraved on the side. All of these design requirements were successfully met and were a large part of why our game was so successful and fun to play.

Final Design and Process

Our final design can be broken down into two larger pieces, the housing and the internal electrical components respectively. Our housing consisted of the top playing face, the angled front face, the side wood panels and the laser cut box base. All of these components that make up the housing are made of different materials and showcase different engineering skills. Our top playing face and angled face are both made on solidworks and 3D printed.

Figure 1

Figure 2

Starting by looking at the top face, we can simply look at one section at a time as the design was essentially a 3x3in square face where each section had an X and an O as well as a button hole. Each of the X and O's had an offset lip so that we could add a piece of laser cut acrylic on top to act as the light source for each time the X or O was pressed. On the underside of the top face there are sectioned off smaller cubes that go underneath each X and O so that when the LED was put behind the acrylic it would not bleed over into the X or O respectively.

Figure 3

The other portion of the housing that was laser cut was the box base as well as the side panels. Instead of 3D printing the lower housing box base, we decided to laser cut a box to cut down on printing time. We ended up making a 9 x 9in box where we took off the top and the front side so that we could lay

the top face of the housing as the lid and the angled face as the front side of the box. We then sealed the bottom of the base and made sure to print one large 9x13in rectangle base so that everything was just attached to one base material. Additionally, we made side panels which acted as more of an aesthetic piece and made the design look clean and cohesive. The panes themselves were 4x13 with an angled slant of 45 degrees to line up with the front angled face.

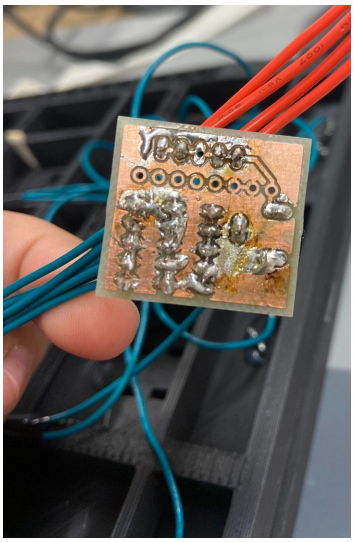


Figure 4

We were able to keep our circuit as streamlined as it could be and all that was needed from our PCB was 15 ground holes, 3 power holes, and a spot to hold our power jack. The Arduino itself was used to power the LED lights and the LED screen at the start but as we extended the lights from 9 to 18 and made the code more complicated the Arduino didn't have the capability to supply it all with enough voltage. Using a power jack that we plugged a cord from an outlet into we were able to power the LED screen and lights, as well as the Arduino without compromising the brightness of the display or lights. The 11 buttons that were used all took their voltage from the Arduino directly, and were then all wired to the ground rails of the PCB. The LED lights were soldered together with wires connecting the three copper pads (ground, power, data) on each side of the bulbs. At first the bulbs were soldered with enough spacing in mind for 1 per pair of X and O, this was later corrected as the final product has 1 bulb per X and 1 per O. The soldering portion was messy until the unspoken soldering rule of "Solder sticks to solder and not much else" was learned the amount of solder used was far too generous and caused more harm than good. The wires proved challenging to solder because of how small and precise it had to be. Brining the wire to the copper pad and then solder to the 2 touching was way harder than anticipated. This problem was solved by adding solder to the pads and tips of the wires individually, bringing them together and pressing down with the soldering iron. The solder on the wire would heat up and then heat the copper pad's solder, absorbing the wire and then cooling. This would have been super helpful to have known when the PCB was soldered.

The final touches we made to the outward physical appearance of our game was that we made it into a star wars theme. We laser engraved an R2D2 on one of the side wood panels as well as laser engraved and cut acrylic signs that had the name of our game and the start indicators. Additionally, we had the majority of the housing that wasn't wood to be all black so that the LEDs could really shine

through the clear acrylic we cut. We chose to make our player lights red and blue to also reference Star Wars.

Testing and Analysis

We tested multiple different people in our classroom. The goal was for them to break the code. This meant keep playing the game in multiple ways until we got an error. At this stage, our code was working well but it was still in its early stages. With having different people test our code, we were able to figure out loops in the code AND the circuit itself. At this stage, we also had our LCD screen working. This was important in figuring out if the circuit was working or if it was just the code. Because we had multiple people try to break our code in the beginning of the process, during the expo there was no problem with our project when it was being played by a lot of people over the duration of 2 hours.

Iteration and Improvements

After we began doing initial testing of the entire circuit on the breadboard it was obvious that something needed to be changed. With the Arduino plugged into my laptop the circuit worked great. However when we changed the Arduinos power supply to be a wall outlet barrel jack connector our game began to act unexpectedly. As the LED strip lit up more lights, both the LCD screen and the light strip began to dim. In order to troubleshoot this the team headed to the electronics center where we discovered we were trying to pull too much current through the Arduino. We were trying to power the screen and light strip which both needed 5V of power, and we were supplying the arduino with the 5V barrel jack connector. To fix this instead of plugging the power cable into the arduino we instead plugged it into a breadboard and later our PCB. This then allowed us to connect the LCD screen, LED light strip, and arduino directly to 5V. This fixed the dimming problems we were having and allowed the game to work properly. We then decided to make our PCB the power connections because we wanted to ensure they were very reliable.

PCB and PCB Process

The PCB design process ended up being a fairly basic PCB, but super helpful while we built the circuit. The PCB we designed encompassed the entire circuit and as the project progressed we knew that the PCB we needed was not going to be much more than a large amount of ground spots and voltage spots. We ran into a problem with the amount of voltage the arduino was able to supply, since it was not enough to power the LED lights and screen with enough brightness and speed we had to add in a power jack. We had been using the Arduino as a way of limiting the amount of current and voltage in the circuit, but now we had to find a way to power everything using the circuit. The power brick that we used for the power jack was rated with the perfect amount of resistance so that we could supply enough power to the circuit and not fry the Arduino. When we met with Lauren the PCB design still consisted of the entire circuit and over the span of about 5 minutes we deleted everything except the power jack and then added in a bunch of ground rails and a power rail (consisting of 5 holes each) that connected back to the power jack.

At the end the PCB was about 1 square inch but was incredibly useful in keeping the circuit neat and organized.

Conclusion

Tic Tac Go is a version of Tic Tac Toe that is meant to see how you perform under pressure. There are two versions of the game. You can play for 20 seconds or 10 seconds.

As a team, we went through a lot of troubleshooting with different aspects of the project. We had to be flexible because of the 3D printers. With the end of the year sneaking up on many different projects, 3D printers were limited. After brainstorming on items and how we could create pieces without 3D printing but still being able to maintain our goal, we decided to laser cut the outside bottom piece of the box.

Getting multiple people to try out the game was a crucial part in the troubleshooting process. It was important to be able to get an outside perspective and also see all the different possibilities of things that could go wrong.

If we were able to keep working on this project, we would incorporate a sound when someone wins. We would make it ring the Star Wars theme song in order to incorporate the overall Star Wars theme.

Throughout the duration of this project, we were able to work on the team and be open to each other's ideas which is a valuable skill moving forward. We were able to deliver a successful project that worked for Presentation Day.

Budget and Bill of Materials

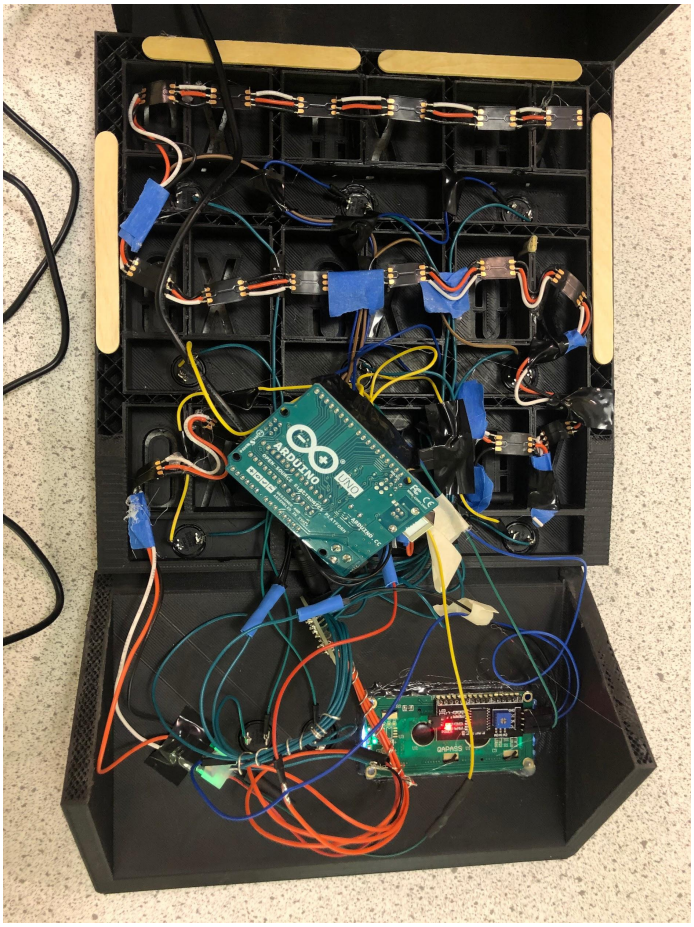
Item	Cost
Adafruit Neopixel LED strip	$33.19
Power Supply	
LCD Screen	$17.40
Push Buttons	$11.00
PLA	$25.00
Wooden Structure	$6.00
	TOTAL : $93.00

Appendix

Task	Due Date	Completed	Notes
Order the parts	End of 11/5 Week	☑	
start code, once compondnets arrive use breadboard	End of 11/12 Week	☑	
Prototyping outside boards aka the acrylic	End of 11/12 Week	☑	
CAD for Stands	End of 11/12 Week	☑	
CAD for housing	End of 11/12 Week	☑	** Switched to Laser Cutting**
Meeting with Lauren	End of 11/26	☑	
Create circuit on EAGLE	End of 11/27	☑	
3D Print - Individual Box	End of 11/28	☑	
3D Print - Scaled Down Version of Top of Game	End of 11/29	☑	
Final Print of Top of Game	End of 11/30	☑	
Final Print of LED holder	End of 11/31	☑	
Finish Putting It Together	12/1	☑	**3 people job to do the flipping and adjustments**

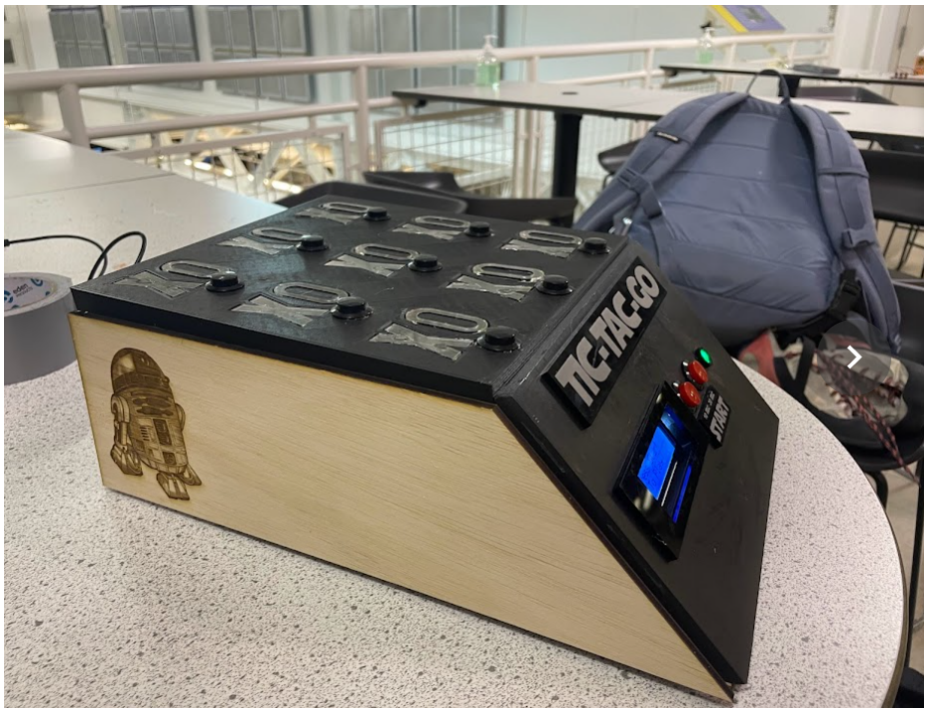
Timeline for Final Project

Picture 1 - Showing off Final Project at Expo

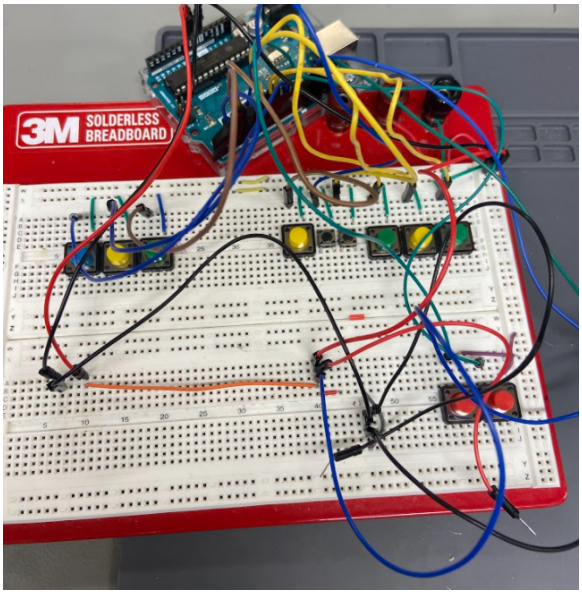


Picture 2 - Inside of Circuits

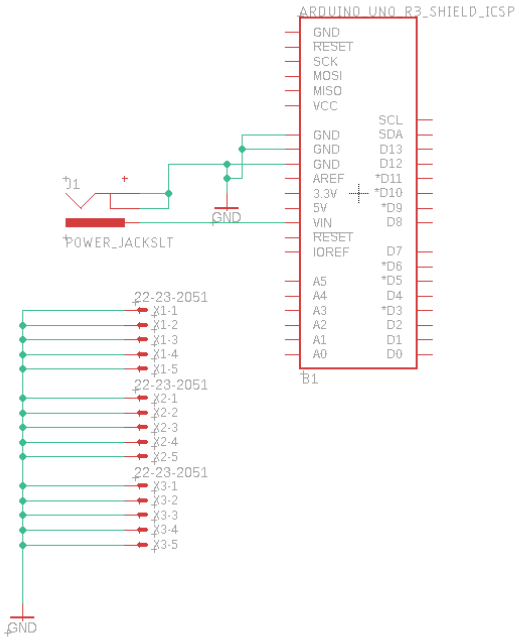
Picture 3 - Close Up of Circuit



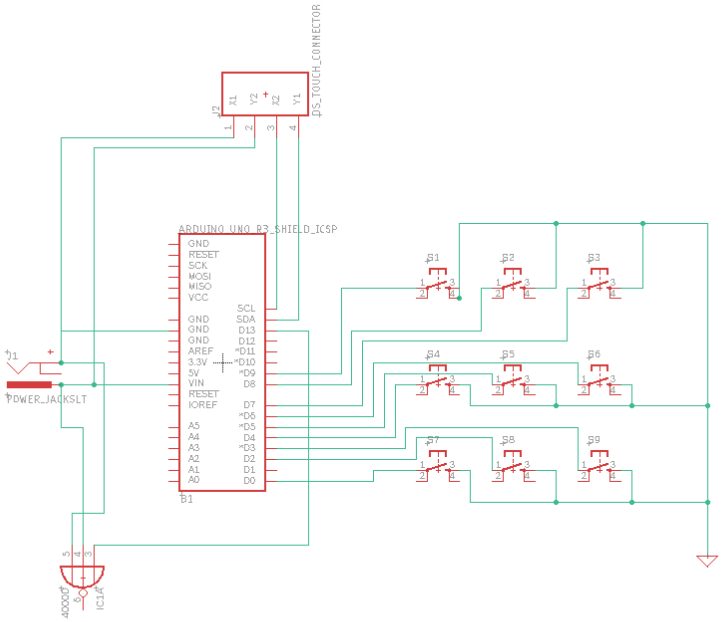
Picture 4 - Side View of Circuit



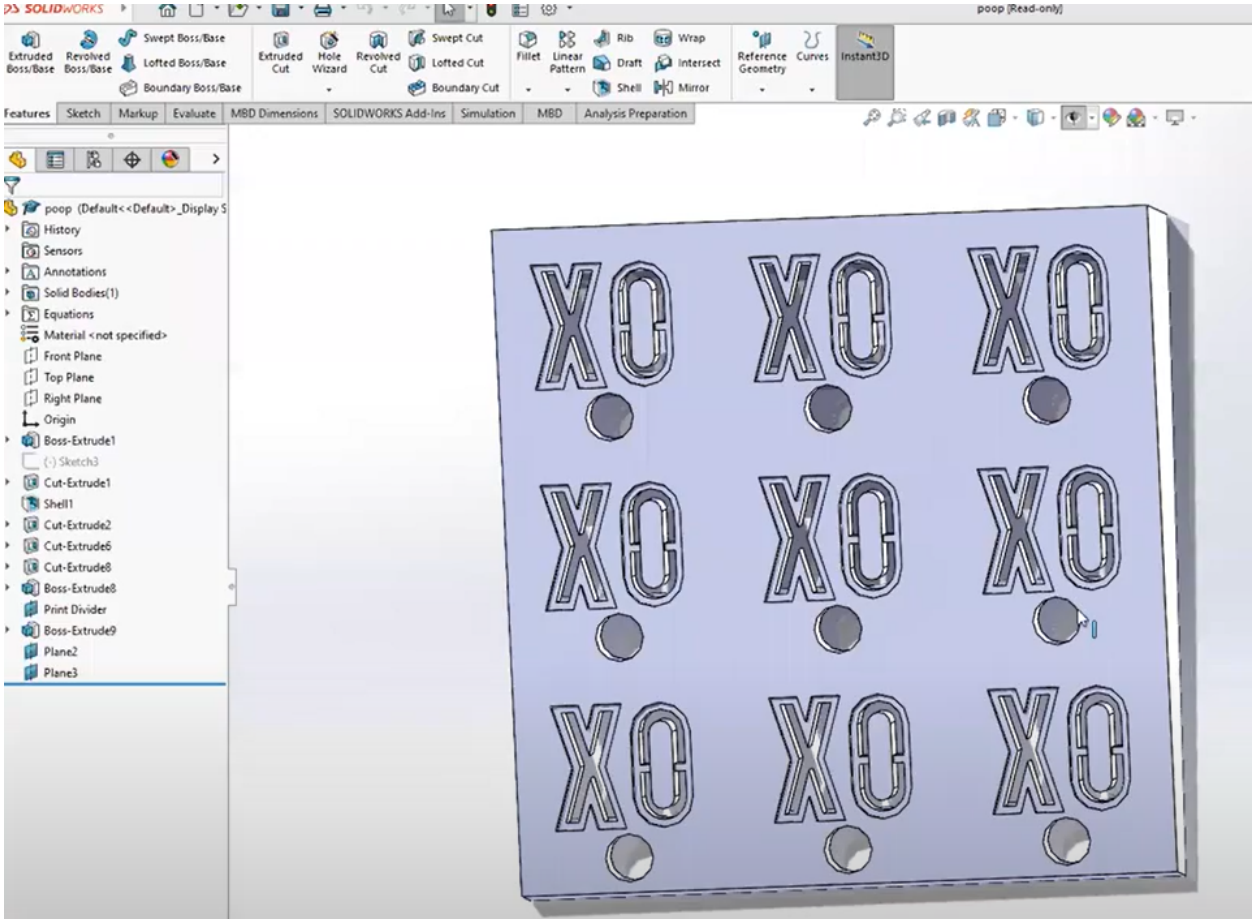
Picture 5 - Circuit Before PCB



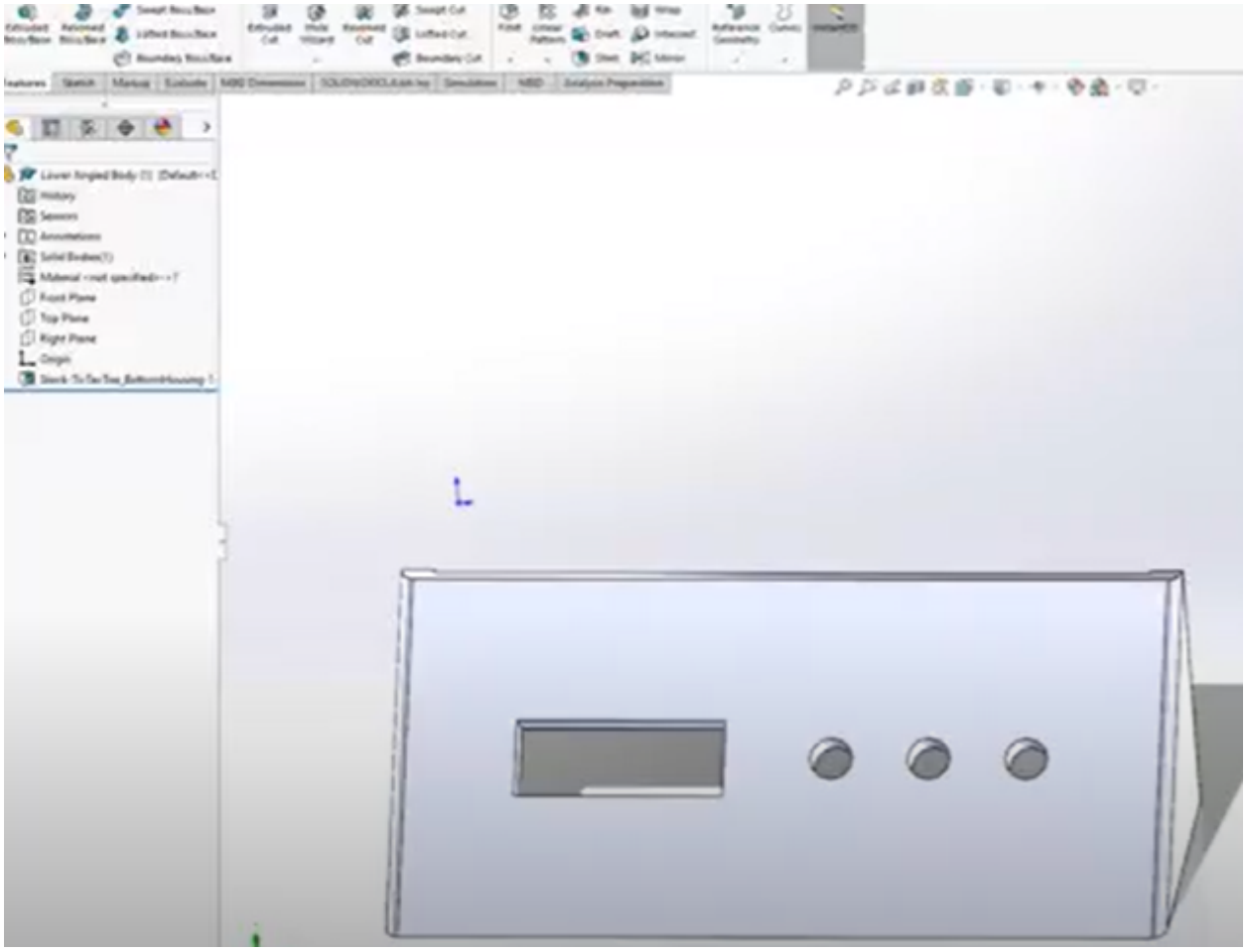
Picture 6 - Schematic of Circuit



Picture 7 - Schematic of Circuit



Picture 8 - CAD of Front Face



Picture 9 - Front Panel CAD

Code

```
//TIC TAC TOE, GEEN 3010, PROFESSOR ZARZKE
//BRYAN SIRNER, SYD KOBAK, DAVID PRICE, YAMI MEDINA-LOPEZ

//BUTTON VARIABLE SETUP (Buttons set up for pins 2-10) Pins 1 and 2 can not read buttons
int sqOne = 10;
int sqTwo = 2;
int sqThree = 3;
int sqFour = 4;
int sqFive = 5;
int sqSix = 6;
int sqSeven = 7;
int sqEight = 8;
int sqNine = 9;

//BUTTONS FOR GAME START
int startTen = 11; //This pin is for the button that when pressed stars a 10 second round of tic tac toe
int startTwenty = 12; // this pin is for the button to play a 20 second game

//LED SETUP
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif
// Which pin on the Arduino is connected to the NeoPixels?
#define PIN 13
// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS 19 // 18 FOR GAME + POWER INDICATOR
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

//TIMER DISPLAY
#include <LiquidCrystal_I2C.h>
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
// I2C address 0x27, 16 column and
2 rows

//TIMER COUNTDOWN SETUP
unsigned long currentTime;
unsigned long lastTime;
unsigned long delayTime = 20;


void setup() {
  // put your setup code here, to run
once:
  //SETTING ALL BUTTON
INPPUTS TO INPUT PULLUP.
  pinMode(sqOne,
INPUT_PULLUP); // // pullup sets
a base voltage which allows for the
button to read inputs better
  pinMode(sqTwo,
INPUT_PULLUP);
  pinMode(sqThree,
INPUT_PULLUP);
  pinMode(sqFour,
INPUT_PULLUP);
  pinMode(sqFive,
INPUT_PULLUP);
  pinMode(sqSix,
INPUT_PULLUP);
  pinMode(sqSeven,
INPUT_PULLUP);
  pinMode(sqEight,
INPUT_PULLUP);
  pinMode(sqNine,
INPUT_PULLUP);
  pinMode(startTen,
INPUT_PULLUP);
  pinMode(startTwenty,
INPUT_PULLUP);

  //LED SETUP
  pixels.begin(); // INITIALIZE
NeoPixel strip object (REQUIRED)
  for (int i = 18; i >= 0; i--) //TURN
ALL LEDS OFF
  {
    pixels.setPixelColor(i,
pixels.Color(0, 0, 0));

    pixels.show();
  }
  pinMode(A0, OUTPUT);


  //screen
  lcd.init(); // initialize the lcd
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("  Press Start  ");
  lcd.setCursor(1, 1);
  lcd.print("   To Play     ");

  //TIMER SETUP
  currentTime = millis();
  lastTime = currentTime;


  //Serial Monitor
  Serial.begin(9600);
}


//GAME TRACKING SETUP
//CREAING AN 9 BOX ARAY
FOR THE 9 suqares on board. if 0,
nothing has been selected, 1 = red, 2
= blue
int TicTac[9] = {0, 0, 0, 0, 0, 0, 0, 0,
0}; // initilizzed to zero at starty of
game because no squares have been
selected
bool player = 1; // This is the
boolian variable to switch between
the players // HIGH = RED = X //
LOW = BLUE = 0

//Timer setup
int second = 10;
int secondTwo = 20;
int Game = 0;
 int winner = 0;

void loop() {
  //HOUSEKEEPING
  int timerFlag = 0;
  digitalWrite(A0, HIGH);

  //POWER LIGHT
  pixels.setPixelColor(0,
pixels.Color(0, 200, 0)); // set color
  pixels.show();   // Send the updated
pixel colors to the hardware.

  //CODE FOR 10 SECOND
GAME TIMER
  currentTime = millis();
  if (digitalRead(startTen) != 1 &&
digitalRead(startTen) == 0) // if start
10 second game is activated
  {
    Serial.println("10 Second Game
Started");
    Game = 1;
  }
  if (currentTime - lastTime >= 1000
&& Game == 1)
  {
    lastTime = currentTime;
    lcd.setCursor(0, 0);
    lcd.print("Time:           ");
    lcd.setCursor(7, 0);
    lcd.print(second);
    lcd.setCursor(0, 1);
    lcd.print("                ");
    second = second - 1 ;
    Serial.println(second);
  }
  else if (second < -1)
  {
    lcd.setCursor(0, 0);
    lcd.print("  TIME RAN OUT");
    lcd.setCursor(2, 1);
    lcd.print(" GAME OVER  ");
    timerFlag = 0;
    delay(4000);
    lcd.setCursor(0, 0);
    lcd.print("  Press Start    ");
    lcd.setCursor(1, 1);
    lcd.print("To Play Again ");
    second = 10;
    Game = 0;
    winner = 0;
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
    {
      pixels.setPixelColor(i,
pixels.Color(255, 50, 255));
      pixels.show();
      delay(200);
    }
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS Off
    {
      pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
      pixels.show();
    }
    delay(100);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
    {
      pixels.setPixelColor(i,
pixels.Color(255, 50, 255));
      pixels.show();
    }
    delay(200);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS Off
    {
      pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
      pixels.show();
    }
    delay(100);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
    {
      pixels.setPixelColor(i,
pixels.Color(255, 50, 255));
      pixels.show();
    }
    delay(200);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
    {
```

```
    pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
    pixels.show();
   }
   for (int i = 0; i < 9; i++)
   {
    TicTac[i] = 0;
   }
  }

  //TWENTY SECOND TIMER
  if (digitalRead(startTwenty) != 1
&& digitalRead(startTwenty) == 0)
// if start 10 second game is
activated
  {
   Serial.println("20 Second Game
Started");
   Game = 2;
  }
  if (currentTime - lastTime >= 1000
&& Game == 2)
  {
   lastTime = currentTime;
   lcd.setCursor(0, 0);
   lcd.print("Time:          ");
   lcd.setCursor(7, 0);
   lcd.print(secondTwo);
   lcd.setCursor(0, 1);
   lcd.print("               ");
   secondTwo = secondTwo - 1 ;
   Serial.println(secondTwo);
  }
  else if (secondTwo < -1)
  {
   lcd.setCursor(0, 0);
   lcd.print("  TIME RAN OUT");
   lcd.setCursor(2, 1);
   lcd.print(" GAME OVER  ");
   timerFlag = 0;
   delay(4000);
   lcd.setCursor(0, 0);
   lcd.print("  Press Start    ");
   lcd.setCursor(1, 1);
   lcd.print("To Play Again ");
   secondTwo = 20;
   Game = 0;
   winner = 0;
   for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
   {
    pixels.setPixelColor(i,
pixels.Color(255, 50, 255));
    pixels.show();
    delay(200);
   }
   for (int i = 18; i > 0; i--) //TURN
ALL LEDS Off
   {
    pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
    pixels.show();
   }
   delay(100);
   for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
   {
    pixels.setPixelColor(i,
pixels.Color(255, 50, 255));
    pixels.show();
   }
   delay(200);
   for (int i = 18; i > 0; i--) //TURN
ALL LEDS Off
   {
    pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
    pixels.show();
   }
   delay(100);
   for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
   {
    pixels.setPixelColor(i,
pixels.Color(255, 50, 255));
    pixels.show();
   }
   delay(200);
   for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
   {
    pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
    pixels.show();
   }

   for (int i = 0; i < 9; i++)
   {
    TicTac[i] = 0;
   }
  }



//Serial.println(digitalRead(sqOne));
  //CODE FOR WHEN THE
SQUARE NUMEBER ONE IS
PRESSED!
  if (digitalRead(sqOne) != 1 &&
digitalRead(sqOne) == 0 &&
TicTac[0] == 0 && Game != 0) // if
squre one is selcted
  {
   Serial.println("Square 1
Pressed");

   if (player == 1)// if it is the red
players turn
   {
    Serial.println("1HIGH");
    TicTac[0] = 1; // updates game
array that red player selected square
1
    //TURN ON FIRST OF 18
LED'S RED
    pixels.setPixelColor(18,
pixels.Color(255, 0, 0)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   if (player == 0) // If it is the blue
players turn
   {
    Serial.println("1LOW");
    TicTac[0] = 2; // updates game
array so square 1 was selected by
blue player
    //TURN ON THE SECOND OF
18 LED'S BLUE
    pixels.setPixelColor(17,
pixels.Color(0, 0, 255)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   //  THIS CHANGES THE
PLAYERS TURN SO IT WILL
ALTERNATE. WHOEVERS
TURN IT WAS IT IS NOT NOT.
   if (player == 1)
   {
    player = 0;
   }
   else if (player == 0)
   {
    player = 1;
   }
   //Serial.println("player = ");
Serial.println(player);
  }


  //CODE FOR WHEN THE
SQUARE NUMEBER TWO IS
PRESSED!
  if (digitalRead(sqTwo) != 1 &&
digitalRead(sqTwo) == 0 &&
TicTac[sqTwo - 1] == 0 && Game
!= 0) // if squre one is selcted
  {
   Serial.println("Square 2
Pressed");

   if (player == 1)// if it is the red
players turn
   {
    Serial.println("2HIGH");
    TicTac[sqTwo - 1] = 1; //
updates game array that red player
selected square 1
    //TURN ON FIRST OF 18
LED'S RED
```

```
    pixels.setPixelColor(16,
pixels.Color(255, 0, 0)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   if (player == 0) // If it is the blue
players turn
   {
    Serial.println("2LOW");
    TicTac[sqTwo - 1] = 2; //
updates game array so square 1 was
selected by blue player
    //TURN ON THE SECOND OF
18 LED'S BLUE
    pixels.setPixelColor(15,
pixels.Color(0, 0, 255)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   //  THIS CHANGES THE
PLAYERS TURN SO IT WILL
ALTERNATE. WHOEVERS
TURN IT WAS IT IS NOT NOT.
   if (player == 1)
   {
    player = 0;
   }
   else if (player == 0)
   {
    player = 1;
   }
   //Serial.println("player = ");
Serial.println(player)
  }


  //CODE FOR WHEN THE
SQUARE NUMEBER THREE IS
PRESSED!
  if (digitalRead(sqThree) != 1 &&
digitalRead(sqThree) == 0 &&
TicTac[sqThree - 1] == 0 && Game
!= 0) // if squre one is selcted
  {
   Serial.println("Square 3
Pressed");

   if (player == 1)// if it is the red
players turn
   {
    Serial.println("3HIGH");
    TicTac[sqThree - 1] = 1; //
updates game array that red player
selected square 1
    //TURN ON FIRST OF 18
LED'S RED
    pixels.setPixelColor(14,
pixels.Color(255, 0, 0)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   if (player == 0) // If it is the blue
players turn
   {
    Serial.println("3LOW");
    TicTac[sqThree - 1] = 2; //
updates game array so square 1 was
selected by blue player
    //TURN ON THE SECOND OF
18 LED'S BLUE
    pixels.setPixelColor(13,
pixels.Color(0, 0, 255)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   //  THIS CHANGES THE
PLAYERS TURN SO IT WILL
ALTERNATE. WHOEVERS
TURN IT WAS IT IS NOT NOT.
   if (player == 1)
   {
    player = 0;
   }
   else if (player == 0)
   {
    player = 1;
   }
   //Serial.println("player = ");
Serial.println(player)
  }


  //CODE FOR WHEN THE
SQUARE NUMEBER FOUR IS
PRESSED!
  if (digitalRead(sqFour) != 1 &&
digitalRead(sqFour) == 0 &&
TicTac[sqFour - 1] == 0 && Game
!= 0) // if squre one is selcted
  {
   Serial.println("Square 4
Pressed");

   if (player == 1)// if it is the red
players turn
   {
    Serial.println("4HIGH");
    TicTac[sqFour - 1] = 1; //
updates game array that red player
selected square 1
    //TURN ON FIRST OF 18
LED'S RED
    pixels.setPixelColor(7,
pixels.Color(255, 0, 0)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   if (player == 0) // If it is the blue
players turn
   {
    Serial.println("4LOW");
    TicTac[sqFour - 1] = 2; //
updates game array so square 1 was
selected by blue player
    //TURN ON THE SECOND OF
18 LED'S BLUE
    pixels.setPixelColor(8,
pixels.Color(0, 0, 255)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   //  THIS CHANGES THE
PLAYERS TURN SO IT WILL
ALTERNATE. WHOEVERS
TURN IT WAS IT IS NOT NOT.
   if (player == 1)
   {
    player = 0;
   }
   else if (player == 0)
   {
    player = 1;
   }
   //Serial.println("player = ");
Serial.println(player)
  }


  //CODE FOR WHEN THE
SQUARE NUMEBER FIVE IS
PRESSED!
  if (digitalRead(sqFive) != 1 &&
digitalRead(sqFive) == 0 &&
TicTac[sqFive - 1] == 0 && Game
!= 0) // if squre one is selcted
  {
   Serial.println("Square 5
Pressed");

   if (player == 1)// if it is the red
players turn
   {
    Serial.println("5HIGH");
    TicTac[sqFive - 1] = 1; //
updates game array that red player
selected square 1
    //TURN ON FIRST OF 18
LED'S RED
    pixels.setPixelColor(9,
pixels.Color(255, 0, 0)); // set color
```

```
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   if (player == 0) // If it is the blue
players turn
   {
    Serial.println("5LOW");
    TicTac[sqFive - 1] = 2; //
updates game array so square 1 was
selected by blue player
    //TURN ON THE SECOND OF
18 LED'S BLUE
    pixels.setPixelColor(10,
pixels.Color(0, 0, 255)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   //  THIS CHANGES THE
PLAYERS TURN SO IT WILL
ALTERNATE. WHOEVERS
TURN IT WAS IT IS NOT NOT.
   if (player == 1)
   {
    player = 0;
   }
   else if (player == 0)
   {
    player = 1;
   }
   //Serial.println("player = ");
Serial.println(player)
  }


  //CODE FOR WHEN THE
SQUARE NUMEBER SIX IS
PRESSED!
  if (digitalRead(sqSix) != 1 &&
digitalRead(sqSix) == 0 &&
TicTac[sqSix - 1] == 0 && Game !=
0) // if squre one is selcted
  {
   Serial.println("Square 6
Pressed");

   if (player == 1)// if it is the red
players turn
   {
    Serial.println("6HIGH");
    TicTac[sqSix - 1] = 1; // updates
game array that red player selected
square 1
    //TURN ON FIRST OF 18
LED'S RED
    pixels.setPixelColor(11,
pixels.Color(255, 0, 0)); // set color
```

```
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   if (player == 0) // If it is the blue
players turn
   {
    Serial.println("6LOW");
    TicTac[sqSix - 1] = 2; // updates
game array so square 1 was selected
by blue player
    //TURN ON THE SECOND OF
18 LED'S BLUE
    pixels.setPixelColor(12,
pixels.Color(0, 0, 255)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   //  THIS CHANGES THE
PLAYERS TURN SO IT WILL
ALTERNATE. WHOEVERS
TURN IT WAS IT IS NOT NOT.
   if (player == 1)
   {
    player = 0;
   }
   else if (player == 0)
   {
    player = 1;
   }
   //Serial.println("player = ");
Serial.println(player)
  }


  //CODE FOR WHEN THE
SQUARE NUMEBER SEVEN IS
PRESSED!
  if (digitalRead(sqSeven) != 1 &&
digitalRead(sqSeven) == 0 &&
TicTac[sqSeven - 1] == 0 && Game
!= 0) // if squre one is selcted
  {
   Serial.println("Square 7
Pressed");

   if (player == 1)// if it is the red
players turn
   {
    Serial.println("7HIGH");
    TicTac[sqSeven - 1] = 1; //
updates game array that red player
selected square 1
    //TURN ON FIRST OF 18
LED'S RED
    pixels.setPixelColor(6,
pixels.Color(255, 0, 0)); // set color
```

```
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   if (player == 0) // If it is the blue
players turn
   {
    Serial.println("7LOW");
    TicTac[sqSeven - 1] = 2; //
updates game array so square 1 was
selected by blue player
    //TURN ON THE SECOND OF
18 LED'S BLUE
    pixels.setPixelColor(5,
pixels.Color(0, 0, 255)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   //  THIS CHANGES THE
PLAYERS TURN SO IT WILL
ALTERNATE. WHOEVERS
TURN IT WAS IT IS NOT NOT.
   if (player == 1)
   {
    player = 0;
   }
   else if (player == 0)
   {
    player = 1;
   }
   //Serial.println("player = ");
Serial.println(player)
  }


  //CODE FOR WHEN THE
SQUARE NUMEBER EIGHT IS
PRESSED!
  if (digitalRead(sqEight) != 1 &&
digitalRead(sqEight) == 0 &&
TicTac[sqEight - 1] == 0 && Game
!= 0) // if squre one is selcted
  {
   Serial.println("Square 8
Pressed");

   if (player == 1)// if it is the red
players turn
   {
    Serial.println("8HIGH");
    TicTac[sqEight - 1] = 1; //
updates game array that red player
selected square 1
    //TURN ON FIRST OF 18
LED'S RED
    pixels.setPixelColor(4,
pixels.Color(255, 0, 0)); // set color
```

```
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   if (player == 0) // If it is the blue
players turn
   {
    Serial.println("8LOW");
    TicTac[sqEight - 1] = 2; //
updates game array so square 1 was
selected by blue player
    //TURN ON THE SECOND OF
18 LED'S BLUE
    pixels.setPixelColor(3,
pixels.Color(0, 0, 255)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   //  THIS CHANGES THE
PLAYERS TURN SO IT WILL
ALTERNATE. WHOEVERS
TURN IT WAS IT IS NOT NOT.
   if (player == 1)
   {
    player = 0;
   }
   else if (player == 0)
   {
    player = 1;
   }
   //Serial.println("player = ");
Serial.println(player)
  }



  //CODE FOR WHEN THE
SQUARE NUMEBER NINE IS
PRESSED!
  if (digitalRead(sqNine) != 1 &&
digitalRead(sqNine) == 0 &&
TicTac[sqNine - 1] == 0 && Game
!= 0) // if squre one is selcted
  {
   Serial.println("Square 9
Pressed");

   if (player == 1)// if it is the red
players turn
   {
    Serial.println("9HIGH");
    TicTac[sqNine - 1] = 1; //
updates game array that red player
selected square 1
    //TURN ON FIRST OF 18
LED'S RED
    pixels.setPixelColor(2,
pixels.Color(255, 0, 0)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   if (player == 0) // If it is the blue
players turn
   {
    Serial.println("9LOW");
    TicTac[sqNine - 1] = 2; //
updates game array so square 1 was
selected by blue player
    //TURN ON THE SECOND OF
18 LED'S BLUE
    pixels.setPixelColor(1,
pixels.Color(0, 0, 255)); // set color
    pixels.show();   // Send the
updated pixel colors to the
hardware.
   }

   //  THIS CHANGES THE
PLAYERS TURN SO IT WILL
ALTERNATE. WHOEVERS
TURN IT WAS IT IS NOT NOT.
   if (player == 1)
   {
    player = 0;
   }
   else if (player == 0)
   {
    player = 1;
   }
   //Serial.println("player = ");
Serial.println(player)
  }



  //WIN CONDITIONS
  //CHECKING FOR GAME END
CONFDITONS



  //FIRST ROW
  if (TicTac[0] == TicTac[1] &&
TicTac[1] == TicTac[2] &&
TicTac[2] == TicTac[0]) // If the the
first row is the same
  {
   if (TicTac[0] == 1) // if they are
all = to one
   {
    winner = 1;
   }
   if (TicTac[0] == 2)
   {
    winner = 2;
   }
  }

  //SECOND ROW
  if (TicTac[3] == TicTac[4] &&
TicTac[4] == TicTac[5] &&
TicTac[5] == TicTac[3]) // If the the
second row is the same
  {
   if (TicTac[3] == 1) // if they are
all = to one
   {
    winner = 1;
   }
   if (TicTac[3] == 2)
   {
    winner = 2;
   }
  }

  //THIRD ROW
  if (TicTac[6] == TicTac[7] &&
TicTac[7] == TicTac[8] &&
TicTac[8] == TicTac[6]) // If the the
THIRD row is the same
  {
   if (TicTac[6] == 1) // if they are
all = to one
   {
    winner = 1;
   }
   if (TicTac[6] == 2)
   {
    winner = 2;
   }
  }

  //FIRST COLLUMN
  if (TicTac[0] == TicTac[3] &&
TicTac[3] == TicTac[6] &&
TicTac[6] == TicTac[0]) // If the the
THIRD row is the same
  {
   if (TicTac[0] == 1) // if they are
all = to one
   {
    winner = 1;
   }
   if (TicTac[0] == 2)
   {
    winner = 2;
```

```
    }
  }


  //SECOND COLUMN
  if (TicTac[1] == TicTac[4] &&
TicTac[4] == TicTac[7] &&
TicTac[7] == TicTac[1]) // If the the
THIRD row is the same
  {
    if (TicTac[1] == 1) // if they are
all = to one
    {
      winner = 1;

    }
    if (TicTac[1] == 2)
    {
      winner = 2;
    }
  }

  //THIRD COLUMN
  if (TicTac[2] == TicTac[5] &&
TicTac[5] == TicTac[8] &&
TicTac[8] == TicTac[2]) // If the the
THIRD row is the same
  {
    if (TicTac[2] == 1) // if they are
all = to one
    {
      winner = 1;
    }
    if (TicTac[2] == 2)
    {
      winner = 2;
    }
  }

  //negative diagnal
  if (TicTac[0] == TicTac[4] &&
TicTac[4] == TicTac[8] &&
TicTac[8] == TicTac[0]) // If the the
THIRD row is the same
  {
    if (TicTac[0] == 1) // if they are
all = to one
    {
      winner = 1;
    }
    if (TicTac[0] == 2)
    {
      winner = 2;
    }
  }

  //POSITIVE DIAGNAL
  if (TicTac[6] == TicTac[4] &&
TicTac[4] == TicTac[2] &&
TicTac[2] == TicTac[6]) // If the the
THIRD row is the same
  {
    if (TicTac[2] == 1) // if they are
all = to one
    {
      winner = 1;
    }
    if (TicTac[2] == 2)
    {
      winner = 2;
    }
  }


  //NOBODY WINS TIE
SCRATCH
  int j = 0;
  for (int i = 0; i < 9; i++)
  {
    if (TicTac[i] != 0)
    {
      j++;
    }
    if (j == 9 && winner == 0) // if
the board is full and nobody won
    {
      for (int i = 0; i < 9; i++) // reset
game data
      {
        TicTac[i] = 0;
      }
      Game = 0; //Cancel Timers

      winner = 0;
      second = 10;
      secondTwo = 20;
      lcd.setCursor(0, 0);
      lcd.print("      DRAW      ");
      lcd.setCursor(0, 1);
      lcd.print("                ");
      delay(2000);
      for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
      {
        pixels.setPixelColor(i,
pixels.Color(255, 50, 255));
        pixels.show();
        delay(200);
      }
      for (int i = 18; i > 0; i--) //TURN
ALL LEDS Off
      {
        pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
        pixels.show();
      }
      delay(100);
      for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
      {
        pixels.setPixelColor(i,
pixels.Color(255, 50, 255));
        pixels.show();
      }
      delay(200);
      for (int i = 18; i > 0; i--) //TURN
ALL LEDS Off
      {
        pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
        pixels.show();
      }
      delay(100);
      for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
      {
        pixels.setPixelColor(i,
pixels.Color(255, 50, 255));
        pixels.show();
      }
      delay(200);
      for (int i = 18; i > 0; i--) //TURN
ALL LEDS Off
      {
        pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
        pixels.show();
      }

      lcd.setCursor(0, 0);
      lcd.print(" To Play Again  ");
      lcd.setCursor(0, 1);
      lcd.print("  Press Start  ");
      delay(2000);
    }

  }



  //Reset Game Array and Data
  if (winner != 0)
  {
    for (int i = 0; i < 9; i++)
    {
      TicTac[i] = 0;
    }
  }


  //WINNNING ANNOUNCMETN
AND CELEBRATION EFFECTS.
  if (winner == 1)
  {
    second = 10;
    secondTwo = 20;
    Game = 0; //Cancel Timers
    winner = 0;
    lcd.setCursor(0, 0);
    lcd.print("RED PLAYER
WINS!");
    lcd.setCursor(0, 1);
    lcd.print("                ");
    delay(2000);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
```

```
    {
     pixels.setPixelColor(i,
pixels.Color(255, 0, 0));
     pixels.show();
     delay(200);
    }
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS Off
    {
     pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
     pixels.show();
    }
    delay(100);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
    {
     pixels.setPixelColor(i,
pixels.Color(255, 0, 0));
     pixels.show();
    }
    delay(200);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS Off
    {
     pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
     pixels.show();
    }
    delay(100);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
    {
     pixels.setPixelColor(i,
pixels.Color(255, 0, 0));
     pixels.show();
    }
    delay(200);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
    {
     pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
     pixels.show();
    }

    lcd.setCursor(0, 0);
    lcd.print(" To Play Again  ");
    lcd.setCursor(0, 1);
    lcd.print("  Press Start  ");
    delay(2000);
  }


  if (winner == 2)
  {
    second = 10;
    secondTwo = 20;
    Game = 0; //Cancel Timers
    winner = 0;
    lcd.setCursor(0, 0);
    lcd.print("BLUE PLAYER
WINS");
    lcd.setCursor(0, 1);
    lcd.print("                ");
    delay(2000);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
    {
     pixels.setPixelColor(i,
pixels.Color(0, 0, 255));
     pixels.show();
     delay(200);
    }
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS Off
    {
     pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
     pixels.show();
    }
    delay(100);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
    {
     pixels.setPixelColor(i,
pixels.Color(0, 0, 255));
     pixels.show();
    }
    delay(200);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS Off
    {
     pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
     pixels.show();
    }
    delay(100);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
    {
     pixels.setPixelColor(i,
pixels.Color(0, 0, 255));
     pixels.show();
    }
    delay(200);
    for (int i = 18; i > 0; i--) //TURN
ALL LEDS ON
    {
     pixels.setPixelColor(i,
pixels.Color(0, 0, 0));
     pixels.show();
    }

    lcd.setCursor(0, 0);
    lcd.print(" To Play Again  ");
    lcd.setCursor(0, 1);
    lcd.print("  Press Start  ");
    delay(2000);
    winner = 0;

  }
```