University of Oklahoma

Digital Design Lab

Homework 2

Aidan Ivy & Makya Stell

ECE 4273-010

Dr. Erik Petrich

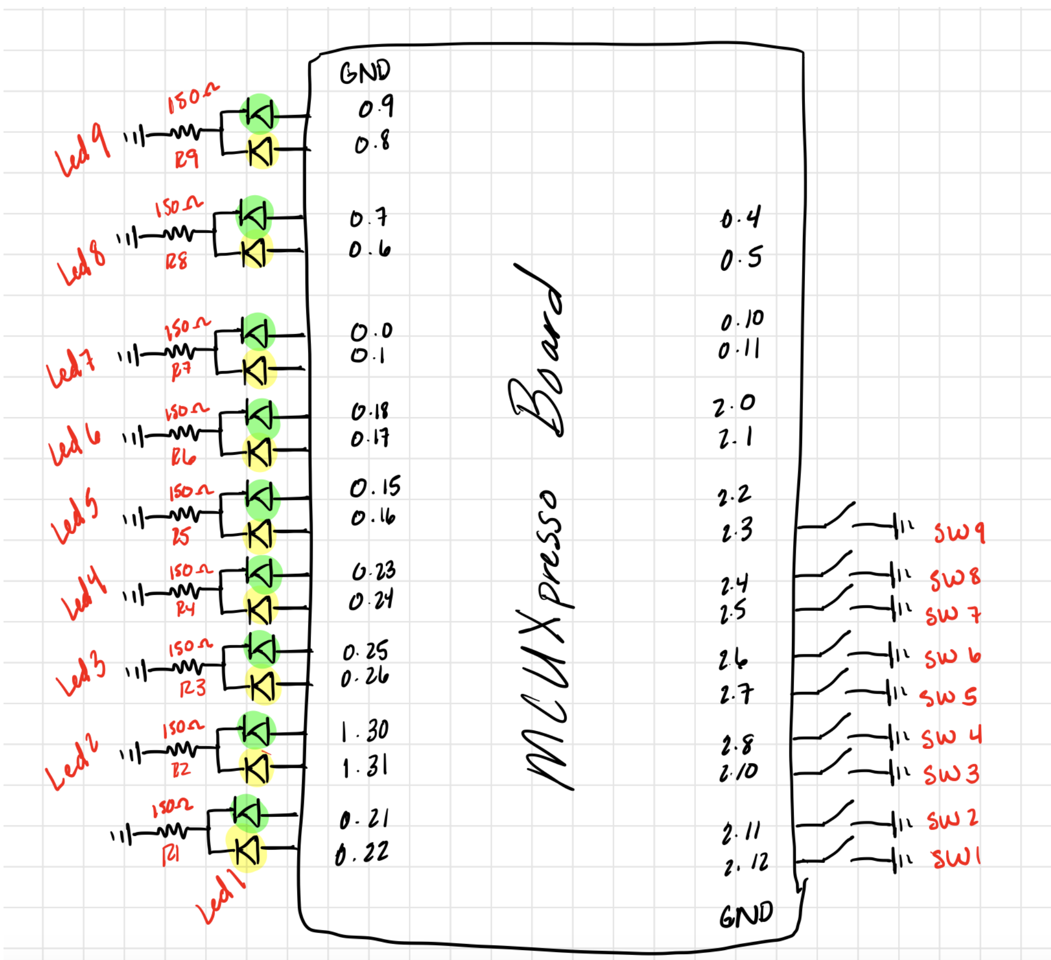
September 17, 2021

Overview

Within this project, we were able to utilize the LPCXpresso board to create a grid of 9 LEDs controlled by a matching grid of switches allowing two users to play a game of tic-tac-toe. The program should take the buttons as player inputs and automatically switch off turns between players, beginning with Player 1 (Yellow) followed by Player 2 (Green). The LEDs consist of 2 anodes that allow for the switching between either color depending on the desired input.

Design

The first step was creating our schematic. In order to make coding in the pin I/O cases more efficient and build the components into the board, we drew a schematic of the LPCXpresso board and matched their port and pin number to whatever LED/switch number we needed. To improve the cleanliness of the board we kept all of the LED pins on the left and the switch inputs on the right.



From a hardware perspective, we were able to use the documentation to provide insight in calculating the amount of resistance needed for each LED to keep the proper amount of current flow. We know that 100mA is provided with 20mA reserved from the board, leaving 80mA reserved for the LEDs with a 3.3V Vcc. We divide the 80mA across 9 LEDs to get 8.89mA each. We then find the difference of 3.3V with the 2.1V forward voltage of each LED and divide that by 8.89mA to get roughly 135 ohms needed. We were able to use 150 ohm resistors due to lab availability. We oriented the LEDs and switches in row format, starting with LED1 in the top left and ending with LED9 in the bottom right, and matched the orientation of the switches to this for ease of use.

$$\frac{80\,mA}{9} \approx 8.89\,mA$$

$$V_F = 2.1$$

$$R = \frac{3.3V - V_F}{.00889} = 134.98\,\Omega$$

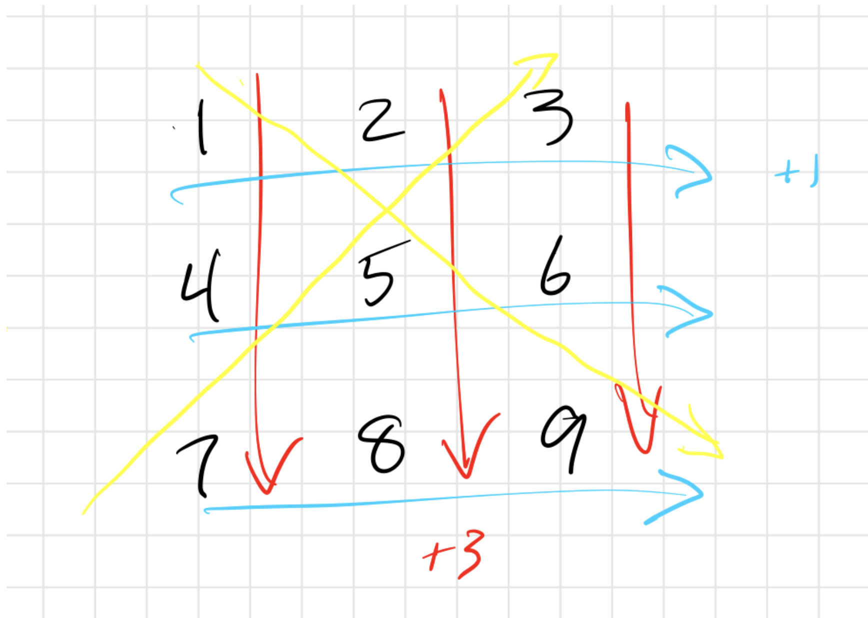
We then began to move into the coding portion. This program, at least with our current knowledge of the pinouts and the board functions, required a lot of declarations and pin mode settings. We first allocated register memory to each of the actions and their respective ports. Then we configured each of the LED pins as output, combining them in pairs in the code for each color anode and for ease of debugging. Then the switches were then configured as inputs.

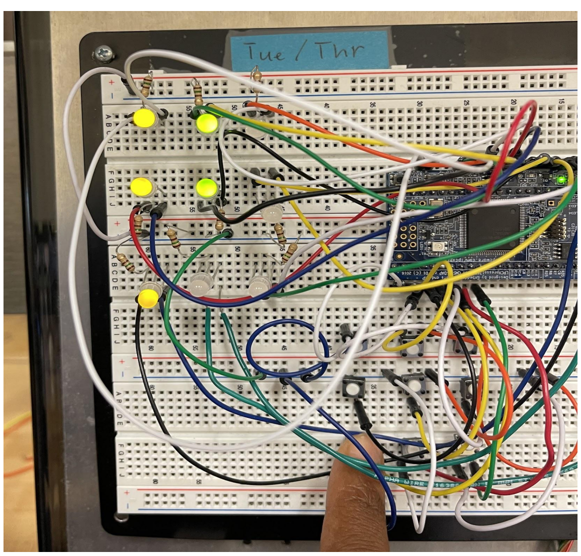
We used an int counter method to keep track of which player's turn it was. P1 and P2 both began valued at zero, and in *main()* the for loops compared their values as equal and incremented P1 within, while the second checked that P1 > P2 and incremented P2 within. Once inside, a function called *chooseLED()* in which the compiler searches for an input of a switch, and matches this input through if statements to the proper LED and color to be illuminated.

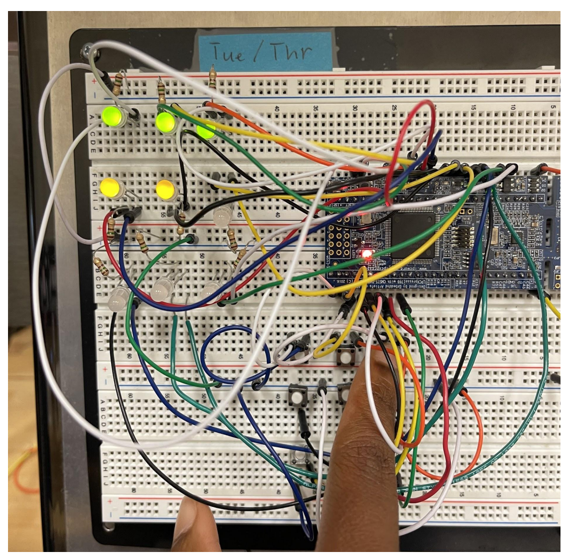
A *reset()* function was implemented that reset all LEDs to off whenever called in the event of either a winner or a full board draw. This is called before each iteration of the game to clear the board initially and when a winner is determined.

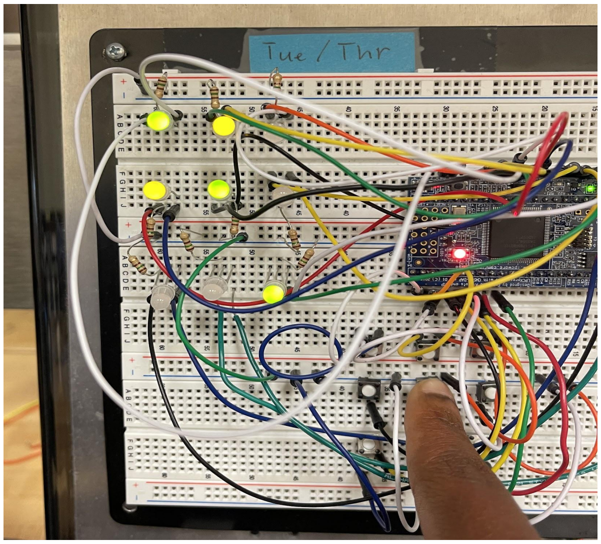
In order to determine the winner, we knew that there were 8 possible combinations of LEDs for a win: 3 row wins, 3 column wins, and 2 diagonal wins. When equating the LEDs to the numbered 2D matrix we created, a pattern was noticed between types of wins: row wins increment by 1 always (i.e. 1,2,3 or 6,7,8), column wins always increment by 3 (i.e. 1,4,7 or

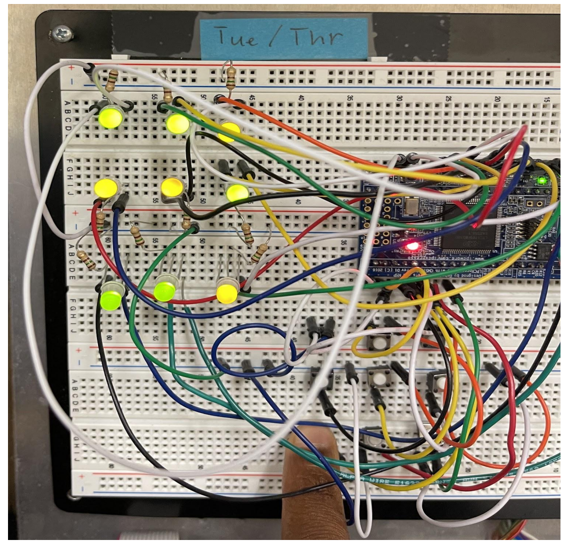
3,6,9). This allowed for us to constantly check and update arrays for each of the players then a sequence of 2 for loops is checking each time they are updated to find a winner. One for loop increments by 1 (rows), one by 3 (columns), and the two if statements check for each of the two diagonal wins.











Conclusion

These sequences of checks and functions allow us to move through the game and effectively execute the visuals across the board. A lot of hard coding and checks were necessary and the debugging took a bit of time, but incorporating the learned pinouts and experimenting with the functionality of the game was challenging but a lot of fun. We feel confident through our understanding of the game and how the board works through it.

ECE 4273

Lab Demonstration Sign-off

Assignment Number	2
Team Members Demoing	Makya Stell
	Aidan Ivy
Date	16 Sept 2021
Time	605pm
Witnessed by	Gopi

Were all objectives completed?

✓_____ Yes

_____ No

If "No", describe which objectives were completed or not completed (whichever is easiest):

Appendix

```
/*

================================================================================
==========
 Name        : HW2.c
 Author      : Aidan Ivy & Makya Stell
 Description : Assignment #2 - Designing with LEDs and Switches
            Yellow LEDs represent player 1 and Green LEDs represent
player 2

================================================================================
==========
 */

//TODO: DISPLAY THE NUMBER OF TIMES THE PLAYERS HAVE WON USING
LEDS???
//All Resistors are 150 ohms
#include <stdbool.h>
#define FIO0DIR (*(volatile unsigned int *) 0x2009c000)
#define FIO0PIN (*(volatile unsigned int *) 0x2009c014)

#define FIO1DIR (*(volatile unsigned int *) 0x2009c020)
#define FIO1PIN (*(volatile unsigned int *) 0x2009c034)

#define FIO2DIR (*(volatile unsigned int *) 0x2009c040)
#define FIO2PIN (*(volatile unsigned int *) 0x2009c054)

//Holds the amount of times the player has gone
volatile int P1 = 0;
volatile int P2 = 0;

//Booleans to prevent cheating
bool P1Turn = false;

const int Off = 0;
const int Y = 1;
const int G = 2;
int led_states[9] = {Off};

//TODO: Start with all of the LEDs off. FIND EASY WAY TO CLEAR THEM
void wait(int secs) {
```

```
	volatile int count;
	for (count = 0; count < (366666 * secs); count++) {
		//do nothing
	}
}

void chooseLED (void)
{
	//SW1
	if (((FIO2PIN >> 12) & 1) == 0)
	{
		if ((((FIO0PIN >> 22) & 1) == 0) && (((FIO0PIN >> 21) & 1)
== 0))
		{
			if (P1Turn == true)
			{
				FIO0PIN |= (1 << 22); //yellow
				FIO0PIN &= ~(1 << 21); //green
				wait(0.5);
				P1++;
				P1Turn = false;
				led_states[0] = Y;
				return;
			}

			else
			{
				FIO0PIN &= ~(1 << 22); //yellow
				FIO0PIN |= (1 << 21); //green
				wait(0.5);
				P2++;
				P1Turn = true;
				led_states[0] = G;
				return;
			}
		}
	}

	//SW2
	else if (((FIO2PIN >> 11) & 1) == 0)
	{
		if ((((FIO1PIN >> 31) & 1) == 0) && (((FIO1PIN >> 30) & 1)
== 0))
		{
			if (P1Turn == true)
```

```
                {
                    FIO1PIN |= (1 << 31); //yellow
                    FIO1PIN &= ~(1 << 30); //green
                    wait(0.5);
                    P1++;
                    P1Turn = false;
                    led_states[1] = Y;
                    return;
                }

                else
                {
                    FIO1PIN &= ~(1 << 31); //yellow
                    FIO1PIN |= (1 << 30); //green
                    wait(0.5);
                    P2++;
                    P1Turn = true;
                    led_states[1] = G;
                    return;
                }
            }
        }

        //SW3
        else if (((FIO2PIN >> 10) & 1) == 0)
        {
            if ((((FIO0PIN >> 26) & 1) == 0) && (((FIO0PIN >> 25) & 1)
    == 0))
            {
                if (P1Turn == true)
                {
                    FIO0PIN |= (1 << 26); //yellow
                    FIO0PIN &= ~(1 << 25); //green
                    wait(0.5);
                    P1++;
                    P1Turn = false;
                    led_states[2] = Y;
                    return;
                }

                else
                {
                    FIO0PIN &= ~(1 << 26); //yellow
                    FIO0PIN |= (1 << 25); //green
                    wait(0.5);
```

```
                P2++;
                P1Turn = true;
                led_states[2] = G;
                return;
            }
        }
    }

    //SW4
    else if (((FIO2PIN >> 8) & 1) == 0)
    {
        if ((((FIO0PIN >> 24) & 1) == 0) && (((FIO0PIN >> 23) & 1)
== 0))
        {
            if (P1Turn == true)
            {
                FIO0PIN |= (1 << 24); //yellow
                FIO0PIN &= ~(1 << 23); //green
                wait(0.5);
                P1++;
                P1Turn = false;
                led_states[3] = Y;
                return;
            }

            else
            {
                FIO0PIN &= ~(1 << 24); //yellow
                FIO0PIN |= (1 << 23); //green
                wait(0.5);
                P2++;
                P1Turn = true;
                led_states[3] = G;
                return;
            }
        }
    }

    //SW5
    else if (((FIO2PIN >> 7) & 1) == 0)
    {
        if ((((FIO0PIN >> 16) & 1) == 0) && (((FIO0PIN >> 15) & 1)
== 0))
        {
            if (P1Turn == true)
```

```
                {
                        FIO0PIN |= (1 << 16); //yellow
                        FIO0PIN &= ~(1 << 15); //green
                        wait(0.5);
                        P1++;
                        P1Turn = false;
                        led_states[4] = Y;
                        return;
                }

                else
                {
                        FIO0PIN &= ~(1 << 16); //yellow
                        FIO0PIN |= (1 << 15); //green
                        wait(0.5);
                        P2++;
                        P1Turn = true;
                        led_states[4] = G;
                        return;
                }
        }
}

//SW6
else if (((FIO2PIN >> 6) & 1) == 0)
{
        if ((((FIO0PIN >> 18) & 1) == 0) && (((FIO0PIN >> 17) & 1)
== 0))
        {
                if (P1Turn == true)
                {
                        FIO0PIN |= (1 << 17); //yellow
                        FIO0PIN &= ~(1 << 18); //green
                        wait(0.5);
                        P1++;
                        P1Turn = false;
                        led_states[5] = Y;
                        return;
                }

                else
                {
                        FIO0PIN &= ~(1 << 17); //yellow
                        FIO0PIN |= (1 << 18); //green
                        wait(0.5);
```

```
                    P2++;
                    P1Turn = true;
                    led_states[5] = G;
                    return;
               }
          }
     }

     //SW7
     else if (((FIO2PIN >> 5) & 1) == 0)
     {
          if ((((FIO0PIN >> 1) & 1) == 0) && (((FIO0PIN >> 0) & 1)
== 0))
          {
               if (P1Turn == true)
               {
                    FIO0PIN |= (1 << 1); //yellow
                    FIO0PIN &= ~(1 << 0); //green
                    wait(0.5);
                    P1++;
                    P1Turn = false;
                    led_states[6] = Y;
                    return;
               }

               else
               {
                    FIO0PIN &= ~(1 << 1); //yellow
                    FIO0PIN |= (1 << 0); //green
                    wait(0.5);
                    P2++;
                    P1Turn = true;
                    led_states[6] = G;
                    return;
               }
          }
     }

     //SW8
     else if (((FIO2PIN >> 4) & 1) == 0)
     {
          if ((((FIO0PIN >> 6) & 1) == 0) && (((FIO0PIN >> 7) & 1)
== 0))
          {
               if (P1Turn == true)
```

```
                {
                        FIO0PIN |= (1 << 6); //yellow
                        FIO0PIN &= ~(1 << 7); //green
                        wait(0.5);
                        P1++;
                        P1Turn = false;
                        led_states[7] = Y;
                        return;
                }

                else
                {
                        FIO0PIN &= ~(1 << 6); //yellow
                        FIO0PIN |= (1 << 7); //green
                        wait(0.5);
                        P2++;
                        P1Turn = true;
                        led_states[7] = G;
                        return;
                }
        }
}

//SW9
else if (((FIO2PIN >> 3) & 1) == 0)
{
        if ((((FIO0PIN >> 8) & 1) == 0) && (((FIO0PIN >> 9) & 1)
== 0))
        {
                if (P1Turn == true)
                {
                        FIO0PIN |= (1 << 8); //yellow
                        FIO0PIN &= ~(1 << 9); //green
                        wait(0.5);
                        P1++;
                        P1Turn = false;
                        led_states[8] = Y;
                        return;
                }

                else
                {
                        FIO0PIN &= ~(1 << 8); //yellow
                        FIO0PIN |= (1 << 9); //green
                        wait(0.5);
```

```
                    P2++;
                    P1Turn = true;
                    led_states[8] = G;
                    return;
                }
            }
        }

        else
        {
            //do nothing
        }
}

void reset (void)
{
        //TODO: Reset if all 9 switches have been pressed to make it
easy
        //TODO: Set noMoves bool to true for check in while

        //All switches have been pressed regardless of color or player
        //and no one has won

        //Turn off all LEDs.
        FIO0PIN &= ~(1 << 21); //green
        FIO0PIN &= ~(1 << 22); //yellow
        FIO1PIN &= ~(1 << 30); //green
        FIO1PIN &= ~(1 << 31); //yellow
        FIO0PIN &= ~(1 << 25); //green
        FIO0PIN &= ~(1 << 26); //yellow
        FIO0PIN &= ~(1 << 23); //green
        FIO0PIN &= ~(1 << 24); //yellow
        FIO0PIN &= ~(1 << 15); //green
        FIO0PIN &= ~(1 << 16); //yellow
        FIO0PIN &= ~(1 << 18); //green
        FIO0PIN &= ~(1 << 17); //yellow
        FIO0PIN &= ~(1 << 0); //green
        FIO0PIN &= ~(1 << 1); //yellow
        FIO0PIN &= ~(1 << 7); //green
        FIO0PIN &= ~(1 << 6); //yellow
        FIO0PIN &= ~(1 << 9); //green
        FIO0PIN &= ~(1 << 8); //yellow
        for (int h = 0; h<9 ; h++)
        {
            led_states[h] = Off;
```

```
	}
}

void winner (void)
{
	//TODO: If player 1 has 3 or if player 2 has 3 in a row then
they win
	//TODO: We need to think of all the combos of P1 and P2
	//TODO: Set win bool to true to test in main
	//TODO: Reset all LEDS
	//ROWS CHECK
	for (int i = 0; i <= 8; i = i + 3){
		if (led_states[i] == led_states[i+1] && led_states[i + 1]
== led_states[i+2] && led_states[i+2] != Off)
		{
			wait(1);
			reset();
			return;
		}
	}

	//COLUMNS CHECK
	for (int j = 0; j <= 2 ; j++){
		if(led_states[j] == led_states[j+3] && led_states [j+3] ==
led_states[j+6] && led_states [j+6] != Off)
		{
			wait(1);
			reset();
			return;
		}
	}

	//DIAGONAL CHECK
	if (led_states[0] == led_states[4] && led_states[4] ==
led_states[8] && led_states[8] != Off)
	{
		wait(1);
		reset();
		return;
	}

	if (led_states[2] == led_states[4] && led_states[4] ==
led_states[6] && led_states[6] != Off)
	{
		wait(1);
```

```
            reset();
            return;
      }

      for (int k = 0; k < 9; k++)
      {
            if (led_states[k] == Off)
            {
                  return;
            }
      }
      wait(1); reset(); return;
}



int main(void) {

      // Outputs (LEDs)

      // LED1
      FIO0DIR |= (1 << 22); FIO0DIR |= (1 << 21);
      // LED2
      FIO1DIR |= (1 << 31); FIO1DIR |= (1 << 30);
      //LED3
      FIO0DIR |= (1 << 26); FIO0DIR |= (1 << 25);
      //LED4
      FIO0DIR |= (1 << 24); FIO0DIR |= (1 << 23);
      //LED5
      FIO0DIR |= (1 << 16); FIO0DIR |= (1 << 15);
      //LED6
      FIO0DIR |= (1 << 17); FIO0DIR |= (1 << 18);
      //LED7
      FIO0DIR |= (1 << 1); FIO0DIR |= (1 << 0);
      //LED8
      FIO0DIR |= (1 << 6); FIO0DIR |= (1 << 7);
      //LED9
      FIO0DIR |= (1 << 8); FIO0DIR |= (1 << 9);

      //Inputs (buttons)

      // SW1
      FIO2DIR &= ~(1u << 12);
      // SW2
      FIO2DIR &= ~(1u << 11);
```

```
	// SW3
	FIO2DIR &= ~(1u << 10);
	// SW4
	FIO2DIR &= ~(1u << 8);
	// SW5
	FIO2DIR &= ~(1u << 7);
	// SW6
	FIO2DIR &= ~(1u << 6);
	// SW7
	FIO2DIR &= ~(1u << 5);
	// SW8
	FIO2DIR &= ~(1u << 4);
	// SW9
	FIO2DIR &= ~(1u << 3);
	reset();

	while (1) {
		//first checks to see if the program needs to reset
		winner();

		if (P1 == P2)
		{
			P1Turn = true;
			chooseLED();
			winner();
		}

		else
		{
			P1Turn = false;
			//do nothing
		}

		if (P1 > P2)
		{
			chooseLED();
			winner();
		}
		else
		{
			//do nothing
		}
	}
}
```