// tic tac toe game

#define CountLoop 5 // how many times to take a reading

#define Threshold 1010 // sensitivity threshold for analog inputs

#define LEDTime 500 // how long the LED stays on, in microseconds

#define ModePin 12 // mode selection pin

#define ModeLEDPin 13 // which mode: 0 = 2 player, 1 = 1 player vs uController

int RedLEDPins[] = {9, 10, 11};

int GreenLEDPins[] = {3, 5, 6};

int CathodePins[] = {4, 7, 8};

void setup()

{

 // setting up of mode selection pin

 pinMode(ModePin, INPUT);

 digitalWrite(ModePin, HIGH);

 pinMode(ModeLEDPin, OUTPUT);

 digitalWrite(ModeLEDPin, LOW);

}

// lightLED function lights up the LEDs

void lightLED(word LEDOnOff, word LEDColour)

{

 // shift the bits to the right, turning on the LEDs whenever

 // there is a 1, turning off whenever there is a 0 in LEDOnOff

 // If the LED is lit, LEDColour determines which LED is lit

 // 1 is red, 0 is green

 for (int j=0;j<3;j++)

 {

 pinMode(RedLEDPins[j], INPUT);

 digitalWrite(RedLEDPins[j], LOW);

 pinMode(GreenLEDPins[j], INPUT);

 digitalWrite(GreenLEDPins[j], LOW);

 pinMode(CathodePins[j], INPUT);

 digitalWrite(CathodePins[j], LOW);

 }

 for (int i=0;i<9;i++)

 {

 if (LEDOnOff & 1)

 {

 if (LEDColour & 1)

 {

 pinMode(RedLEDPins[i/3], OUTPUT);

 pinMode(CathodePins[i%3], OUTPUT);

 digitalWrite(RedLEDPins[i/3], HIGH);

 digitalWrite(CathodePins[i%3], LOW);

 delayMicroseconds(LEDTime);

 digitalWrite(RedLEDPins[i/3], LOW);

 pinMode(RedLEDPins[i/3], INPUT);

 pinMode(CathodePins[i%3], INPUT);

 } else

 {

 pinMode(GreenLEDPins[i/3], OUTPUT);

 pinMode(CathodePins[i%3], OUTPUT);

 digitalWrite(GreenLEDPins[i/3], HIGH);

 digitalWrite(CathodePins[i%3], LOW);

 delayMicroseconds(LEDTime);

 digitalWrite(GreenLEDPins[i/3], LOW);

 pinMode(GreenLEDPins[i/3], INPUT);

 pinMode(CathodePins[i%3], INPUT);

 }

 }

 LEDOnOff = LEDOnOff >> 1;

 LEDColour = LEDColour >> 1;

 }

}

// lightLED function lights up the LEDs

void lightLEDPWM(word LEDOnOff, word LEDColour, int level)

{

 // shift the bits to the right, turning on the LEDs whenever

 // there is a 1, turning off whenever there is a 0 in LEDOnOff

 // If the LED is lit, LEDColour determines which LED is lit

 // 1 is red, 0 is green

 for (int j=0;j<3;j++)

 {

 pinMode(RedLEDPins[j], INPUT);

 digitalWrite(RedLEDPins[j], LOW);

 pinMode(GreenLEDPins[j], INPUT);

 digitalWrite(GreenLEDPins[j], LOW);

 pinMode(CathodePins[j], INPUT);

 digitalWrite(CathodePins[j], LOW);

 }

 for (int i=0;i<9;i++)

 {

 if (LEDOnOff & 1)

 {

 if (LEDColour & 1)

 {

 pinMode(RedLEDPins[i/3], OUTPUT);

 pinMode(CathodePins[i%3], OUTPUT);

 analogWrite(RedLEDPins[i/3], level);

 digitalWrite(CathodePins[i%3], LOW);

 delayMicroseconds(LEDTime);

 digitalWrite(RedLEDPins[i/3], LOW);

 pinMode(RedLEDPins[i/3], INPUT);

 pinMode(CathodePins[i%3], INPUT);

 } else

 {

 pinMode(GreenLEDPins[i/3], OUTPUT);

 pinMode(CathodePins[i%3], OUTPUT);

 analogWrite(GreenLEDPins[i/3], level);

 digitalWrite(CathodePins[i%3], LOW);

 delayMicroseconds(LEDTime);

 digitalWrite(GreenLEDPins[i/3], LOW);

 pinMode(GreenLEDPins[i/3], INPUT);

 pinMode(CathodePins[i%3], INPUT);

 }

 }

 LEDOnOff = LEDOnOff >> 1;

 LEDColour = LEDColour >> 1;

 }

}

// readGrid function reads a press on the wire grid

word readGrid()

{

int PinRowVal, PinColVal;

int SelectRow = 0;

int SelectCol = 0;

// set columns to OUTPUT LOW, set rows to INPUT and enable pullups

for (int i=0;i<3;i++)

{

 pinMode(17+i, OUTPUT);

 digitalWrite(17+i, LOW);

 pinMode(14+i, INPUT);

 digitalWrite(14+i, HIGH);

}

// do an analog read of the row pins; stop when the first one

// drops below the threshold

 for (int RowCount=0;RowCount<3;RowCount++)

 {

 PinRowVal = 0;

 for (int i=0;i<CountLoop;i++)

 {

 PinRowVal += analogRead(RowCount);

 }

 if (PinRowVal < (CountLoop\*Threshold))

 {

 SelectRow = RowCount + 1;

 break;

 }

 }

// set rows to OUTPUT LOW, set columns to INPUT and enable pullups

for (int i=0;i<3;i++)

{

 pinMode(14+i, OUTPUT);

 digitalWrite(14+i, LOW);

 pinMode(17+i, INPUT);

 digitalWrite(17+i, HIGH);

}

 for (int ColCount=3;ColCount<6;ColCount++)

 {

 PinColVal = 0;

 for (int j=0;j<CountLoop;j++)

 {

 PinColVal += analogRead(ColCount);

 }

 if (PinColVal < (CountLoop\*Threshold))

 {

 SelectCol = ColCount-2;

 break;

 }

 }

 if ((SelectRow > 0) && (SelectCol > 0))

 {

 return (1 << (SelectRow+(SelectCol-1)\*3)-1);

 } else

 {

 return (0);

 }

}

word checkWin(word GridOnOff, word GridColour, boolean Turn)

{

 // there are 8 win conditions - three column wins, three row wins, two diagonal wins

 // this is put in an array, and the GridOnOff status is just compared again this

 // function then returns the win condition

word winArray[] = {7, 56, 73, 84, 146, 273, 292, 448};

if (Turn) // red's turn, check for green

 {

 for (int i=0;i<8;i++)

 {

 if ( ((GridOnOff & ~GridColour) & winArray[i]) == winArray[i])

 {

 return winArray[i];

 }

 }

 return 0;

 } else // green's turn, check for red

 {

 for (int i=0;i<8;i++)

 {

 if ( ((GridOnOff & GridColour) & winArray[i]) == winArray[i])

 {

 return winArray[i];

 }

 }

 return 0;

 }

}

void displayWin(word winCondition, boolean Turn)

{

 word winColour;

 if (Turn)

 {

 winColour = 0;

 } else

 {

 winColour = 511;

 }

 for (int i=256;i>=0;i--)

 {

 lightLED(winCondition, winColour); // light up the winning combo

 lightLEDPWM(~winCondition, ~winColour, i); // fade out the other colour

 }

 for (int i=0;i<10;i++) // blink winning combo a few times

 {

 lightLED(winCondition, winColour);

 delay(100);

 }

}

void loop()

{

 word Pressed;

 word WinCondition = 0;

 byte NoOfTurns = 0;

 word GridOnOff = 0;

 word GridColour = 0;

 boolean Turn = 1; // 1 = red's turn, 0 = green's turn

 boolean Mode;

 char RandomToss; // for playing against uC

 word PickedCell;

 // turn on the LED on pin 13 depending on the state of the ModePin

 Mode = digitalRead(ModePin);

 digitalWrite(ModeLEDPin, Mode);

 // begin the game loop

 // if one player, player starts first

 do

 {

 Pressed = readGrid(); // take a reading

 if (Pressed & ~GridOnOff) // if an empty space is selected

 {

 GridOnOff = GridOnOff | Pressed; // light up the space

 if (Turn) // set colour according to whose turn it is

 {

 GridColour = GridColour | Pressed;

 }

 Turn = !Turn; // Turn goes to opposite player

 NoOfTurns+=1;

 WinCondition = checkWin(GridOnOff, GridColour, Turn); // check if there's a win for player

 if ((Mode) && !(WinCondition)) // if uC's turn, play it now, unless player has won

 {

 do // randomly pick a blank cell

 {

 RandomToss = random(9);

 PickedCell = 1;

 for (int i=0;i<RandomToss;i++)

 {

 PickedCell = PickedCell << 1;

 }

 } while ((PickedCell & GridOnOff));

 GridOnOff = GridOnOff | PickedCell;

 Turn = !Turn; // Turn goes to opposite player

 NoOfTurns+=1;

 WinCondition = checkWin(GridOnOff, GridColour, Turn); // check if there's a win for uC

 }

 }

 lightLED(GridOnOff,GridColour); // light up the LED

 } while ((WinCondition == 0) && (NoOfTurns < 9));

 if (WinCondition > 0) // did anybody win?

 {

 displayWin(WinCondition, Turn);

 } else // it was a draw, fade out all lights

 {

 for (int i=512;i>=0;i--)

 {

 lightLEDPWM(GridOnOff, GridColour, i/2);

 }

 }

}