HW2 Simon Ng

Friday, April 10, 2020 13:09

1. It took me a while to figure out that I could use digitalRead() as a boolean value. After that I troubleshooted what needed to be within the if statement by printing the boolean value digitalRead() was being compared to as well as the count.

```
int count = 0;
bool previous = true;

void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP); // pullup resistor for button
  pinMode(3, INPUT);
}
void loop() {
  Serial.print(previous); Serial.print(" "); Serial.println(count);
  if ((digitalRead(3) != previous) && (digitalRead(3) == false)) {
    count++; // button's state has changed from before and is pressed
  }
  previous = digitalRead(3); // update comparison value
}
```

2. I set the LED to turn on between 0 and 5 seconds after both players press their buttons to confirm readiness, and also added an if() statement to prevent players from holding down the button before the LED turns on. I ran into a problem where after a while the LED would turn on immediately, and this was because I had my time variables as unsigned ints, only giving tPreLED about 1 minute before it was at its max value, and then it couldn't store the extra time I wanted to wait from random(). Using print statements, I realized that after tPreLED grew larger than ~65 000, it would reset millis() to 0. At the same time, millis(), which I compared to tPreLED to decide how long to delay the LED, kept growing. This meant that millis() would always be larger than tPreLED and the LED would instantly turn on. I fixed this by making all of my variables handling millis() unsigned longs.

```
unsigned long tStart;
unsigned long tPreLED;
unsigned long tElapsed;
#define btn1 3
#define btn2 6
#define LED 4

void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP); // pullup resistor for button 1
  pinMode(btn1, INPUT); // readout for button 1
  pinMode(LED, OUTPUT); // LED
  pinMode(5, INPUT_PULLUP); // pullup resistor for button 2
  pinMode(btn2, INPUT); // readout for button 2
  randomSeed(A0); // initialize random sequence
}
void loop() { // loop LED reflex game
  bool restart = false;
  Serial.println(); Serial.println("Both players press button to begin game.");
  while ((digitalRead(btn1) != false) || (digitalRead(btn2) != false)) {
    // wait until both of buttons are pressed (false)
  }
  while ((digitalRead(btn1) != true) || (digitalRead(btn2) != true)) {
    // wait until both of buttons are released (true)
  }
  Serial.println("Go!");
  tPreLED = millis() + random(0, 5000);
  while (millis() < tPreLED) { // wait random length of time 0-5 seconds
    if ((digitalRead(btn1) == false) || (digitalRead(btn2) == false)) {
      Serial.println("Too soon! The LED isn't on yet! Start Over.");
      restart = true; // button pressed too early, so restart game
      break; // don't print "Too soon..." more than once
    }
  }
  if (restart == false) {
    digitalWrite(LED, HIGH); // turn on LED
    tStart = millis(); // start timer
    while ((digitalRead(btn1) != false) && (digitalRead(btn2) != false)) {
      // wait until one of buttons is pressed (false)
    } // end of loop, so one of the buttons is pressed
    tElapsed = millis() - tStart; // time taken to press one of the buttons
    if (digitalRead(btn1) == false) { // button 1 was pressed
      Serial.print("Player w button on rows 24-26 pressed first with time of ");
      Serial.print(tElapsed); Serial.println(" ms");
    }
    else { // button 2 was pressed
      Serial.print("Player w button on rows 5-7 pressed first with time of ");
      Serial.print(tElapsed); Serial.println(" ms");
    }
    digitalWrite(LED, LOW); // turn off LED
    delay(500); // wait 0.5 seconds
  }
}
```

3. I hooked everything up and wrote some basic code, and the LEDs would turn on when desired but would not turn off when the joystick was replaced to its original position. I fixed this by adding a for loop to check for each LED whether the LED should be on or off.
   Then I realized my middle LED was on any time the joystick was directly on one of the axes, since I evaluated it for both the row and column LEDs. I added a few if statements to determine whether the joystick was in the middle of both LR and UD, which was the only case where I would turn on the middle LED. This is what I wanted to achieve, and it looked good when moving the joystick in one axis because the middle LED only turned on when the joystick was in the middle of both axes. However, when rotating the joystick in a circle in both axes, it looked slightly strange since the middle LED was "skipped". I suppose I could write more code so that the middle LED is dependent on which axes have been moving recently, but this seemed unnecessary for the homework.

```
#define midLED 2
int rowLED[] = {2, 3, 4, 5, 6}; // row LEDs are pins 2-6
int colLED[] = {7, 8, 4, 9, 10}; // column LEDs are pins 7-10 sharing 4
#define LR_pin A0 // row joystick is pin A0
#define UD_pin A1 // column joystick is pin A1
bool midLED_on = false;

void setup() {
  Serial.begin(9600);
  pinMode(rowLED, OUTPUT);
  pinMode(colLED, OUTPUT);
}

void loop() {
  for (int i = 0; i < 5; i++) {
    if (i == map(analogRead(LR_pin), 0, 1023, 0, 5)) {
      if (i == midLED) {
        midLED_on = true; // middle should be on according to row
      } // but don't turn it on yet in case the column doesn't agree
      else {
        digitalWrite(rowLED[i], HIGH); // not middle LED, turn on
        midLED_on = false; // middle shouldn't be on according to row
      }
    }
    else {
      digitalWrite(rowLED[i], LOW);
    }
    if ((i == map(analogRead(UD_pin), 0, 1023, 0, 5))) {
      if (i != midLED) { // not middle LED, turn on
        digitalWrite(colLED[i], HIGH);
        midLED_on = false; // middle shouldn't be on according to row
      }
      else if (midLED_on) { // middle should also be on according to col
        digitalWrite(colLED[midLED], HIGH);
      }
      else {
        digitalWrite(colLED[midLED], LOW);
      }
    }
    else {
      digitalWrite(colLED[i], LOW);
    }
  }
}
```