Skyler MacDougall, Matthew Gerace

Homework 6: Due 2/26/2020

1. Rewrite the reaction timer game program using only interrupts.
    - A pin change interrupt on PD7 is used to start the game. When the active-low button is pressed, the yellow LED turns on after a random delay. When the LED turns on, `TCNT1=0`.
    - `INT0` is attached to another active-low button. This is for the reaction button. if the button is pressed before a $\frac{1}{2}$ second has elapsed, the green LED should light.
    - A Timer1 interrupt should be used to count to $\frac{1}{2}$ second. If $\frac{1}{2}$ second passes before `INT0`, the red LED should be lit.

    - [x] Which pins will the 3 LEDs be connected to?

      Green = `PC0`. Yellow = `PC1`. Red = `PC2`.
    - [x] Initialize the pins for the LEDs.

      See line 8
    - [x] Initialize the two interrupt pins as inputs.

      See line 5
    - [x] What mode should Timer1 be in to create a $\frac{1}{2}$ second interrupt?

      CTC, although PWM is also possible.
    - [x] Initialize for `TCCR1A` and `TCCR1B` for the correct mode.

      See lines 11 and 12.
    - [x] Is a clock prescaler needed for $\frac{1}{2}$ second interrupt?

      Yes.
    - [x] If so, set it up in `TCCR1B`.

      See line 12.
    - [x] What value goes in `OCR1A` for $\frac{1}{2}$ second?

      See line 14.
    - [x] Enable the Timer1 interrupt.

      See line 18.
    - [x] Enable the pin change interrupt in the control register.

      See line 19.
    - [x] Enable the external interrupt in the control register.

      See line 21.
    - [x] Set the correct pin for the pin change interrupt in `PCMSK`

      See line 20.

- [x] Set the correct pin for the external interrupt in the `EIMSK`

  See line 22.

- [x] What are the names of the 3 interrupt vectors?

  `PCINT2_vect`, `INT0_vect`, and `TIMER1_COMPA_vect`.

```
1  // Homework 6, Problem 1
2
3  void setup()
4  {
5    DDRD &= ~(0x82); // make PD6 and PD7 inputs -- reaction switch and start
   switch
6    DDRC |= (1 << 0|1 << 1|1 << 2);//PC0, PC1, and PC2 outputs -- green,
   yellow, and red light
7
8    PORTC &= ~(1 << 0|1 << 1|1 << 2); //start all lights off
9
10   // CTC mode timer setups
11   TCCR1A = 0;
12   TCCR1B = 0x0C;
13   TCNT1  = 0;
14   OCR1A  = 31249;
15
16   cli(); // turn off global interrupts
17   // set the intertupt flag
18   TIMSK1 = 4;
19   PCICR  = 4;
20   PCMSK2 = 0x80;
21   EIMSK = 1;
22   EICRA = 3;
23   sei(); // turn on global interrupts
24 }
25
26 void loop(){while(1)}
27
28 ISR(PCINT2_vect){
29    // wait a random amount of time
30   delay(random());
31
32   // turn the yellow LED on
33   PORTC |= 0x02;
34   //reset the timer
35   TCNT1  = 0;
36 }
37
38 ISR(INT0_vect){
39   PORTC &= ~(0x02); // turn yellow LED off
40   PORTC |= 0x01;    // turn red LED on
41 }
42
43 ISR(TIMER1_COMPA_vect)
44 {
45   // code for servicing the interrupt
46   PORTC &= ~(0x02); // turn yellow LED off
47   PORTC |= 0x04;    // turn red LED on
48 }
```

2. In the previous homework, you determined the constants that control your servo motor to rotate 180°. Starting from that point, complete the following:
    1. Once you've determined the constants that will move the servo a full 180°, modify your program to create a windshield wiper controller. As you know, a windshield wiper moves smoothly across 180°. It doesn't "snap" back to position like the servo did in lab 4. You can achieve a smooth motion by incrementally increasing and decreasing your pulsewidth.
3. The last step is to add a pushbutton interrupt to start the windshield wiper. You can choose either a pin change or an external interrupt as a source.
    3. Submit a video of your working system (or bring it to me for a demonstration).

```
unsigned long servoValue=0, prevValue=0;

void setup()
{
pinMode(9,OUTPUT);
Serial.begin(9600);
TCCR1A=0xB2;
TCCR1B=0x1B;
ICR1=0x1387;
}

void loop()
{
delay(10);
prevValue = servoValue;
while ((servoValue>=525) || (prevValue > servoValue))
{
  if (servoValue > 0){
    servoValue--;
    OCR1A=144+servoValue;
    Serial.println(OCR1A);
    delay(10);
  }
  else{break;}
}
if (prevValue == servoValue) {servoValue++;}
OCR1A=144+servoValue;
Serial.println(OCR1A);
}
```

link to video

3. Rewrite Q1 of HW#4, using PWM mode instead of CTC mode.

Because the code is quite lengthy, code modifications can be seen in lines 35, 36, and 44.

```
//Note-name/constant conversion
#define LA   36363
#define LAS  34323
#define LB   32396
#define LC   30578
#define LCS  28861
#define LD   27242
```

```
#define LDS  25713
#define LE   24270
#define LF   22908
#define LFS  21622
#define LG   20408
#define LGS  19263
#define MA   18182
#define MAS  17161
#define MB   16198
#define MC   15289
#define MCS  14430
#define MD   13620
#define MDS  12856
#define ME   12135
#define MF   11454
#define MFS  10811
#define MG   10204
#define MGS   9631
#define HA    9091

//Note Length definitions
#define QUARTER 400
#define EIGHTH 200
#define DOTQUARTER 600
#define HALF 800

void setup(){
    TCCR1A=0b10100011;
    TCCR1B=0b00011001;
    TCCR1C=0;
    TCNT1=0;
    DDRB|=0x02;
}

void playNote(int note, int time){
    OCR1A = note;
    OCR1B = note / 2;
    delay(time);
    OCR1A = 0;
    delay(50);
}

void loop(){
    playNote(LD, EIGHTH);
    playNote(LD, EIGHTH);
    playNote(LG, QUARTER);
    playNote(LG, EIGHTH);
    playNote(MA, EIGHTH);
    playNote(MB, EIGHTH);
    playNote(LG, EIGHTH);
    playNote(MB, EIGHTH);
    playNote(MC, EIGHTH);
    playNote(MD, QUARTER);
    playNote(MD, EIGHTH);
    playNote(MC, EIGHTH);
    playNote(MB, QUARTER);
    playNote(MA, EIGHTH);
    playNote(LG, EIGHTH);
```

```
    playNote(MA, QUARTER);
    playNote(MA, QUARTER);
    playNote(MA, QUARTER);
    playNote(MA, QUARTER);
    playNote(MA, EIGHTH);
    playNote(MB, EIGHTH);
    playNote(MA, EIGHTH);
    playNote(LFS, EIGHTH);
    playNote(LD, QUARTER);
    playNote(LD, QUARTER);
    playNote(LG, EIGHTH);
    playNote(LFS, EIGHTH);
    playNote(LG, EIGHTH);
    playNote(MA, EIGHTH);
    playNote(MB, QUARTER);
    playNote(MA, EIGHTH);
    playNote(MA, EIGHTH);
    playNote(MB, QUARTER);
    playNote(MC, QUARTER);
    playNote(MD, DOTQUARTER);
    playNote(MD, EIGHTH);
    playNote(MA, QUARTER);
    playNote(MA, EIGHTH);
    playNote(MB, EIGHTH);
    playNote(MA, QUARTER);
    playNote(LG, EIGHTH);
    playNote(LG, EIGHTH);
    playNote(MA, QUARTER);
    playNote(LG, EIGHTH);
    playNote(LFS, EIGHTH);
    playNote(LG, HALF);
    playNote(MD, HALF);
    playNote(MB, DOTQUARTER);
    playNote(MA, EIGHTH);
    playNote(MB, EIGHTH);
    playNote(MC, EIGHTH);
    playNote(MB, EIGHTH);
    playNote(MA, EIGHTH);
    playNote(LG, DOTQUARTER);
    playNote(MA, EIGHTH);
    playNote(MB, QUARTER);
    playNote(MB, EIGHTH);
    playNote(MC, EIGHTH);
    playNote(MD, QUARTER);
    playNote(MB, QUARTER);
    playNote(MA, QUARTER);
    playNote(MA, EIGHTH);
    playNote(MB, EIGHTH);
    playNote(MA, DOTQUARTER);
    playNote(LD, EIGHTH);
    playNote(LG, HALF);
    playNote(LG, DOTQUARTER);
    playNote(MA, EIGHTH);
    playNote(MB, QUARTER);
    playNote(MB, EIGHTH);
    playNote(MC, EIGHTH);
    playNote(MD, QUARTER);
    playNote(MC, EIGHTH);
```

```
    playNote(MB, EIGHTH);
    playNote(MA, QUARTER);
    playNote(MA, QUARTER);
    playNote(MB, QUARTER);
    playNote(MB, EIGHTH);
    playNote(MA, EIGHTH);
    playNote(LG, QUARTER);
    playNote(LG, QUARTER);
    playNote(LG, HALF);
    delay(1000);
}
```