ECE 241 Lab 6 Report Spring 2018
Name: Graham Wood Lab Section A

Objective: Reading the Encoder Inputs Using Interrupts

Prelab: The code was written and uploaded to the website. The code written and seen below implements and attaches interrupts so that the encoder knob's position can be tracked accurately.

Part 1) In part one, the code was uploaded to the Arduino and verified that the program tracks the position of the encoder nob. The encoder counts 80 per revolution and about 4/5 per detent. The knob can be positive or negative.

Part 2) In part two the code was modified to add functionality when the button is clicked. When the button is clicked, the count will decrease by one. This was done by importing much of the code used in lab 5 and adding it to the existing code. Code ensuring that voltage bounce would not cause errors was also included. Code can be seen below.

Questions:

What was the count per rotation? 80

What was the count per detent? ~4 +- 1

Appendix A) Code for Part 1

```
 volatile int encoderPosition; //int the tracks the position of the encoder

#include <LiquidCrystal.h> //needed for LCD
LiquidCrystal LcdDriver(11, 9, 5, 6, 7, 8 ); // Must be in your code, exactly
like this
unsigned long timer; //timer to manage the refresh of the LCD


void MonitorA() { //Interrupt procedure for pin 2
    if (digitalRead(2) == digitalRead(3)) {
      encoderPosition++; //increments position of the pins read the same
value
    }
    else {
        encoderPosition--; //decrements otherwise
    }

}


void MonitorB() {
  if (digitalRead(2) == digitalRead(3)) {
    encoderPosition--; //decrements position if the read the same value
  }
  else {
    encoderPosition++; //increments otherwise
  }
}
```

```
void setup() {
  LcdDriver.begin(16,2); // begin
  LcdDriver.clear();// clear
  pinMode(2, INPUT_PULLUP); //sets pin two as an INPUT Pullup
  pinMode(3, INPUT_PULLUP);//sets pin three as an INPUT Pullup
  attachInterrupt(digitalPinToInterrupt(2), MonitorA, CHANGE); //attaches an
interrupt procedure (MOnitorA) to pin two and sets the mode to Change
  attachInterrupt(digitalPinToInterrupt(3), MonitorB, CHANGE); //attaches an
interrupt procedure (MOnitorB) to pin three and sets the mode to Change
  timer = millis(); //Sets the timer to the number of milliseconds since the
program began running

}

void loop() { //loops as long as the program is running
  if ((millis() - timer) >= 100) { //checks if 100 milliseconds has passed
    LcdDriver.clear(  ); //clears the screen
    LcdDriver.print(encoderPosition); //prints the new encoder position to
the LCD
    timer += 100; //increments the timer by 100 milliseconds
  }
  //80 per rotation
  //4/5 per click
}
```

Appendix B) Code for part 2 :

```
volatile int encoderPosition;
int ButtonState;
#include <LiquidCrystal.h>
LiquidCrystal LcdDriver(11, 9, 5, 6, 7, 8 ); // Must be in your code, exactly
like this
unsigned long timer;
unsigned long ButtonTime; //Time to maintain how many milliseconds since the
button was pushed


int ButtonNextState ( int input ) { //function whose purpose is to manage the
button press procedure on the arduino
  switch(ButtonState) { //switch case that's executed depending on the
current state the button is in
    case 1 : //Idle state
      if (input == LOW) {
         ButtonTime = millis();
         ButtonState = 2; //if input is LOW the state transitions to Wait
         digitalWrite(13, HIGH);//turn on LED turns on the LED
         }
    case 2:
      if (input == HIGH) {
        ButtonState = 1; //sets the state back to idle if input is HIGH
      }
        else if (millis() - ButtonTime >= 5) {
          ButtonState = 3;
          digitalWrite(13, LOW); //sets the LED back off
          return 1; //returns a one indicating that the button has been
pressed
        }
    case 3:
        if (input == HIGH) {
          ButtonState = 1; //sets the state to Idle
        }
            return 0; //returns a zero
        }
  }

void MonitorA() {
    if (digitalRead(2) == digitalRead(3)) {
      encoderPosition++;
    }
    else {
        encoderPosition--;
    }

}


void MonitorB() {
  if (digitalRead(2) == digitalRead(3)) {
    encoderPosition--;
  }
  else {
    encoderPosition++;
  }
}



void setup() {
  LcdDriver.begin(16,2); // begin
  LcdDriver.clear();// clear
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(2), MonitorA, CHANGE);
  attachInterrupt(digitalPinToInterrupt(3), MonitorB, CHANGE);
  timer = millis();
  pinMode(13,OUTPUT); //sets pin 13 (LED) to an output
  ButtonState = 1; //initial state is 1
}

void loop() {
  if ((millis() - timer) >= 100) {
    LcdDriver.clear(  );
    LcdDriver.print(encoderPosition); //prints the new count to the LCD
    timer += 100;
  }
  if (ButtonNextState(digitalRead(4) == 1)) { //if the buttonNExtState
returns a 1 then the program will print "Button Pressed"
      encoderPosition--;
  }
  //80 per rotation
  //4/5 per click
}
```