ECE 376 - Homework #3

Binary Inputs, Binary Outputs, & LEDs - Due Monday, September 16th

Binary Inputs

Assume a thermistor has a resistance-temperature relationship of

$$R = 1000 \cdot \exp\left(\frac{3905}{T+273} - \frac{3905}{298}\right)\Omega$$

1) Design a circuit which outputs

- 0V when T < 40C
- 5V when T > 40C

Step 1: Find the resistance at 40C

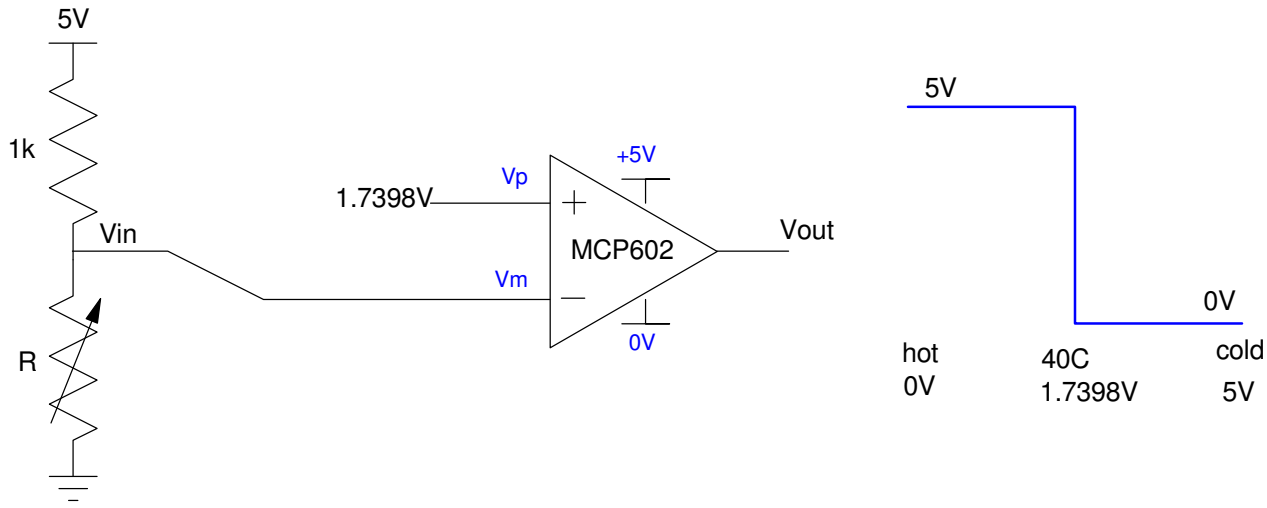
R = 533.6642 Ohms

Step 2: Convert to voltage. Assuming a 1k resistor and a voltage divider

$$V = \left(\frac{R}{R+1k}\right)5V = 1.7398V$$

Step 3: Use a comparitor to convert to 0V / 5V binary

- When T = 1000C (ish), R = 0 (ish), Vin = 0V (ish), Vout = 5V
- Connect to the minus input to get Vout = (T > 40C)



2) Design a circuit which outputs

- 0V when T < 40C
- 5V when T > 45C

Step 1: Find the resistance at temperature

- T = 40C R = 533.6642 Ohms
- T = 45C R = 438.6065 Ohms

Step 2: Convert to voltage. Assume a voltage divider with a 1k resistor

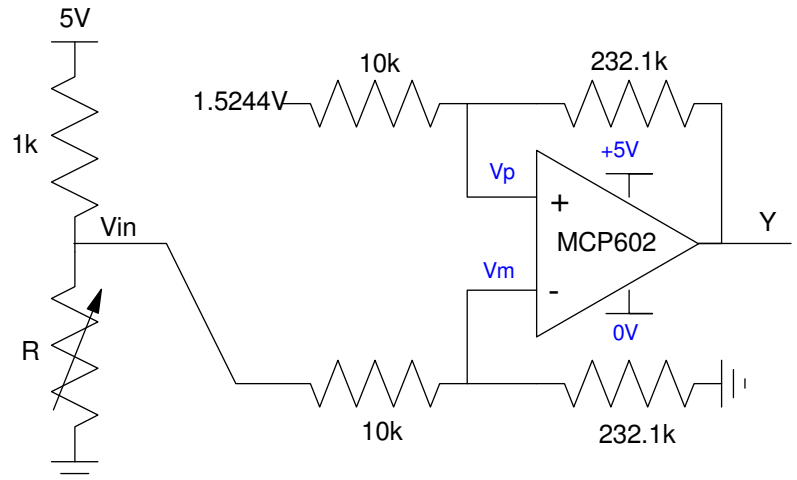
- T = 40C Vx = 1.7398V
- T = 45C Vx = 1.5244V

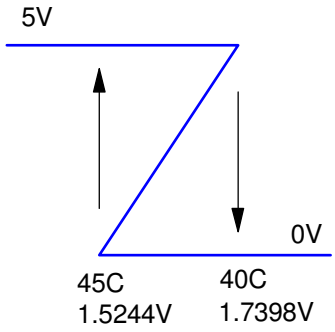
Step 3: Set up the Schmitt trigger

- The on voltage (45C) is less than the off voltage (40C). Connect to the minus input
- The on voltge is 1.5244V. Make the offset 1.5244V
- The gain needed is

$$gain = \left(\frac{\text{change in output}}{\text{change in input}}\right) = \left(\frac{5V-0V}{1.8398V-1.5244V}\right) = 23.21$$

Make the resistor ratio 23.21 : 1



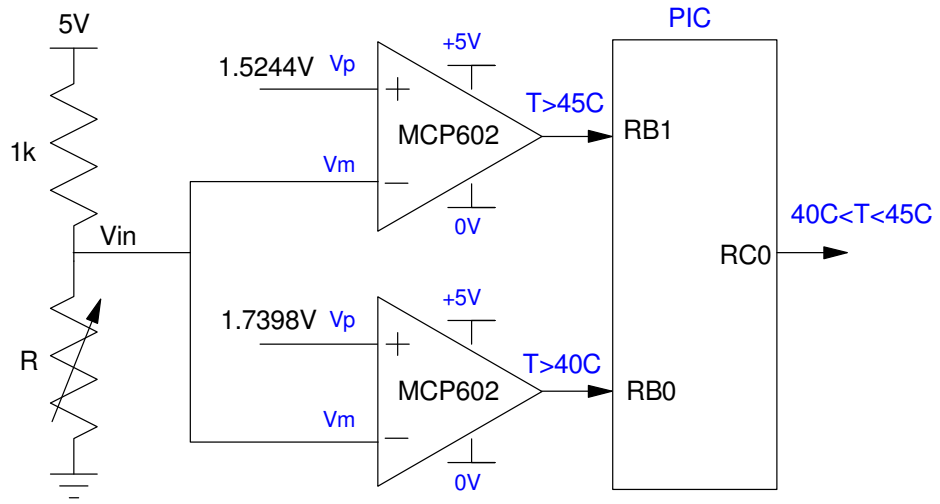


3) Design a circuit which outputs

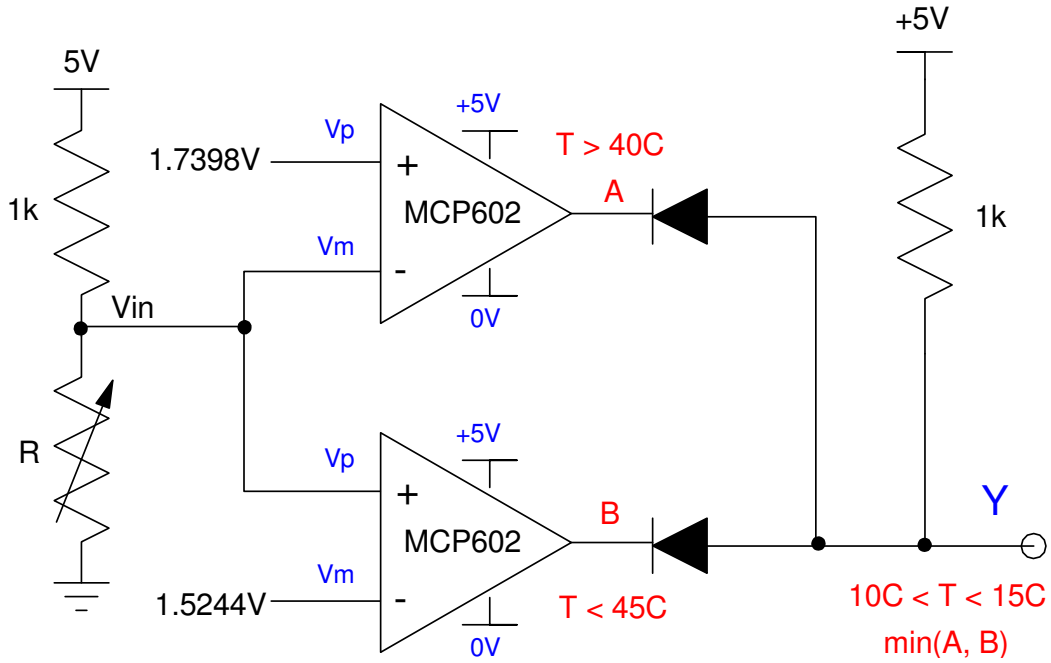
- 0V when 40C < T < 45C
- 5V otherwise

A little trickier. Option #1: Design two comparitors and compute the intersection in software

- V1: V > 40C
- V2: V > 45C



Option #2: Use a min-circuit from Electronics



Binary Outputs

4) Design a circuit which allows your PIC board to turn on and off an RGB Piranah LED at 0mA (off) and 15mA (on). Assume the specifications for the LEDs are:

Color	Vf @ 20mA	mcd @ 20mA
red	2.0V	10,000
green	3.2V	10,000
blue	3.2V	10,000

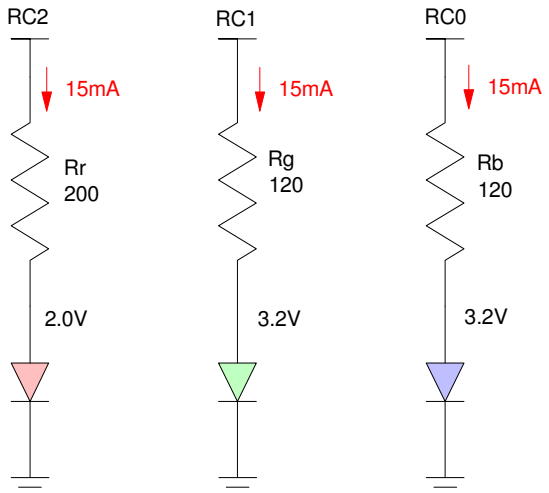
Since the voltage is less than 5V and the current is less than 25mA, the PIC can drive the LEDs directly. Just add a resistor to set the current.

Calculating the resistors:

$$R_r = \left(\frac{5V-2.0V}{15mA}\right) = 200\Omega$$

$$R_g = \left(\frac{5V-3.2V}{15mA}\right) = 120\Omega$$

$$R_b = \left(\frac{5V-3.2V}{15mA}\right) = 120\Omega$$



5) Design a circuit which allows your PIC board to turn on and off a 5W LED at 500mA. The specs for the LED are:

- Vf = 6.0-7.0V (assume 6.5V)
- Current = 700mA
- 500-600 Lumens (equivalent to a 60W light bulb).

https://www.ebay.com/itm/1W-3W-5W-10W-50W-100W-High-power-SMD-Chip-LED-COB-White-Blue-Red-Light-Beads/124011607823

Assume you have a 6144 NPN transistor:

- max continuous current = 3A
- current gain = 300
- Vbe = 0.7V, Vce(sat) = 0.2V

Since this needs more than 5V and more than 25mA, a PIC can't drive the LED directly. Add an NPN transistor as a switch. Assuming a 6144 NPN transistor and a 12V power supply:

$$R_c = \left(\frac{12V - 6.5V - 0.2V}{500mA}\right) = 10.6\Omega$$

In order to saturate the transistor, you need

$$\beta I_b > I_c$$

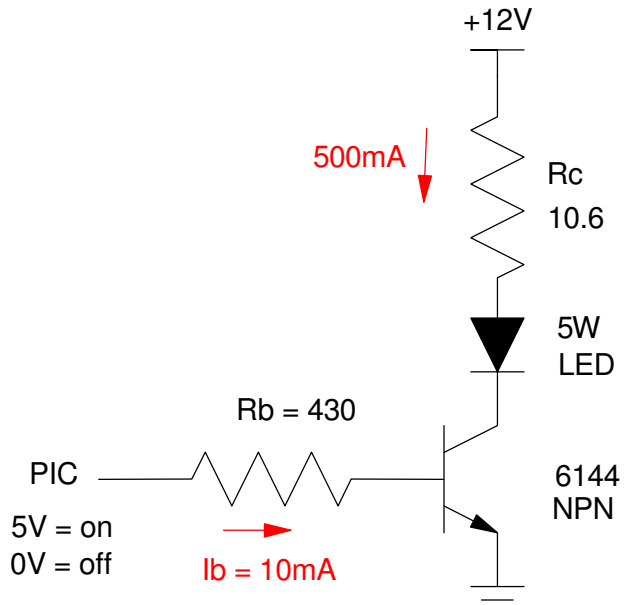
$$I_b > \left(\frac{500mA}{300}\right) = 1.67mA$$

Pick a number larger than 1.67mA and less than 25mA (the most a PIC can output). Let Ib = 10mA

$$I_b = 10mA$$

$$R_b = \left(\frac{5V - 0.7V}{10mA}\right) = 430\Omega$$

Rb doesn't have to be *exactly* 430 Ohms. 470 Ohms works. 330 Ohms works.



Timing:

6) Write a program which outputs the music note G#3 (207.652 Hz)

- Verify the frequency of the square wave you generate
- (Pano Tuner app on you cell phone works well for this)

The duration of the wait loop needs to be 24,078.747 clocks

$$N = \left(\frac{10{,}000{,}000}{2 \cdot Hz}\right) = 24{,}078.747$$

One way to do this is to have three nested wait loops:

```
Wait: movlw     A
; 4 clocks
                movwf     CNT0
W1:    movlw    B
; 5 clocks * A
                movwf     CNT1
W2:    nop
; 10 clocks * A * B
                nop
                nop
                nop
                nop
                nop
                nop
                decfsz CNT1,F
                goto W2
                decfsz CNT0,F
                goto W0
                return
```

The total time spend in the wait loop is

N = 10AB + 5A + 4

Come up with integers which are in the range of (1..255) and the product is close to 24,078,747. In Matlab, you can find the best combination (not necessary - just showing off)

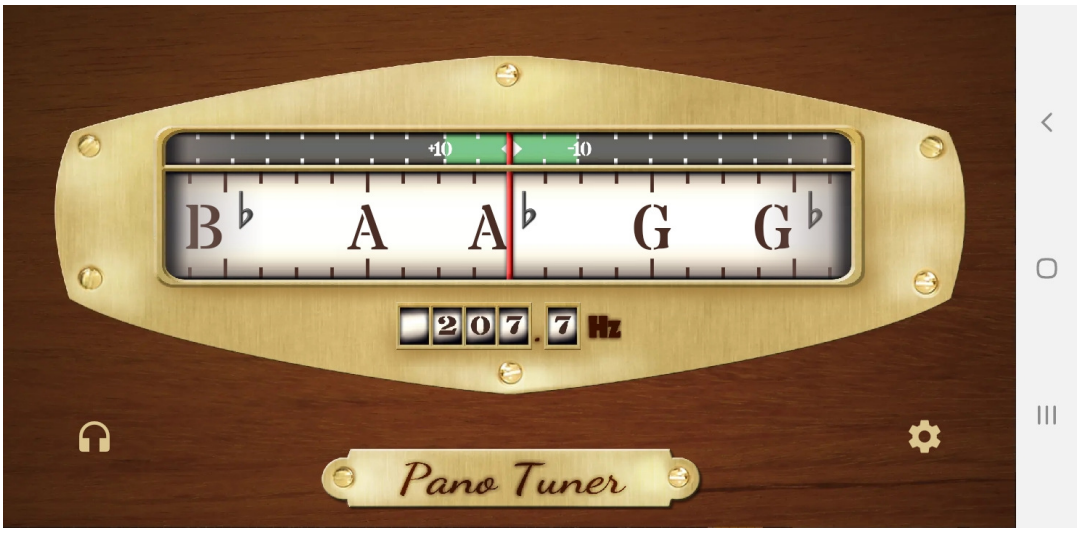
```
% Matlab Code
minE = 9999
for a = 1:255
    for b = 1:255
        N = 10*a*b + 5*a + 4;
        E = abs(24078.7 - N);
        if(E < minE)
            minE = E;
            A = a;
            B = b;
            [A,B,N]
         end
    end
end
```

This results in

A = 15, B = 160, N = 24,079

The resulting program is then

```
#include <p18f4620.inc>

; Variables
CNT0 EQU 1
CNT1 EQU 2

; Program
       org 0x800
       call Init
Loop:
       incf  PORTC,F
       call Wait
       goto Loop

; --- Subroutines ---

Init:
       clrf TRISA
       clrf TRISB
       clrf TRISC
       clrf TRISD
       clrf TRISE
       movlw 0x0F
       movwf ADCON1 ;everyone is binary
       return

Wait:  movlw     15
                 movwf     CNT0
W1:    movlw     160
                 movwf     CNT1
W2:    nop
                 nop
                 nop
                 nop
                 nop
                 nop
                 nop
                 decfsz CNT1,F
                 goto W2
                 decfsz CNT0,F
                 goto W0
                 return
```

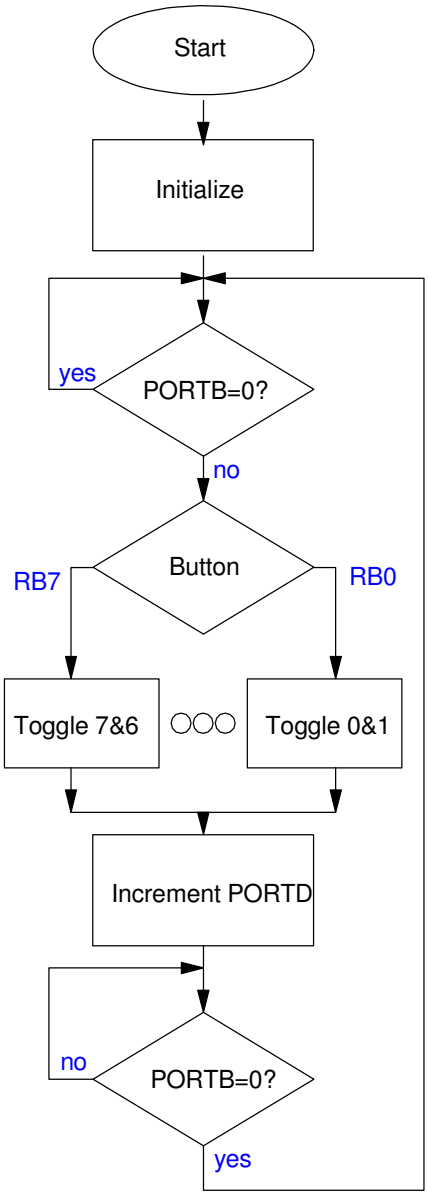


Lab: Lights-Out Game

7) Give the flow chart for a program to turn your PIC board into *Lights-Out* game

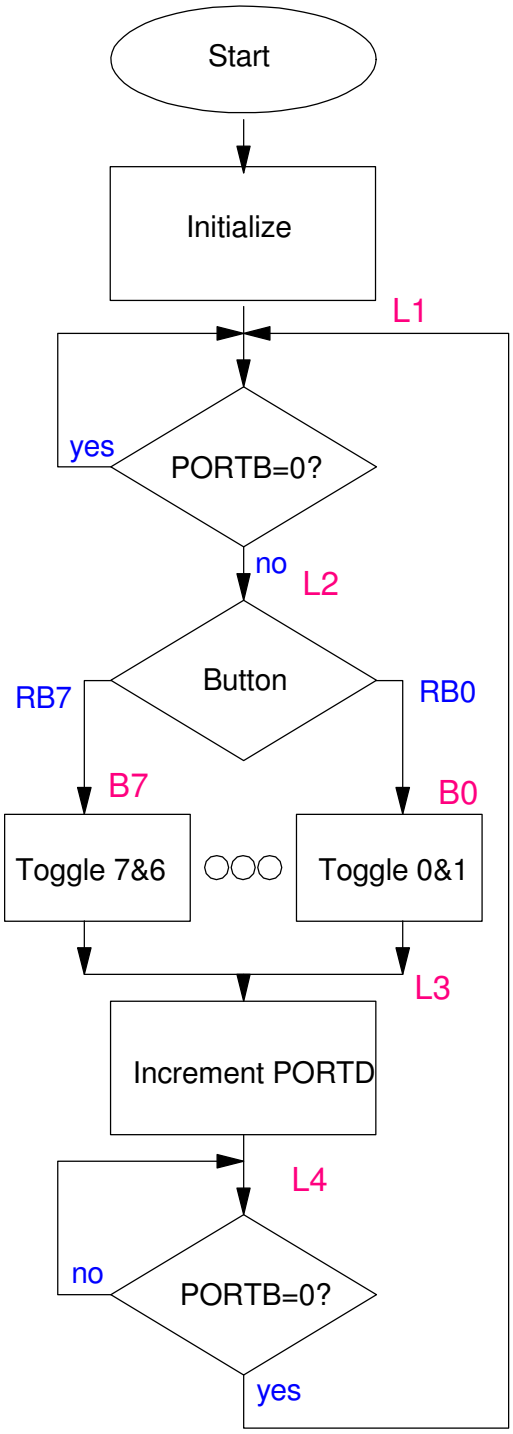
- On power up, PORTC = 0xFF and PORTD = 0x00
- When you press and release a button, the corresponding pin on PORTC and its neighbors are toggeled
  - RB0: Toggle pins RC0, RC1
  - RB1: Toggle pins RC0, RC1, RC2
  - etc.
- Each time you press and release a button, PORTD increments by one

The goal of the game is to turn off all of the lights on PORTC in the minimum number of moves



8) Write the corresponding assembler code

```
        org 0x800
        movlw     0xFF
        movwf     TRISB
        clrf      TRISC
        clrf      TRISD
        movlw     0xFF
        movwf     PORTC
        clrf      PORTD
        movlw     0x0F
        movwf     ADCON1

L1:     movlw     0
        cpfseq    PORTB
        goto      L2
        goto      L1

L2:     btfsc     PORTB,7
        goto      B7
        btfsc     PORTB,6
        goto      B6
        btfsc     PORTB,5
        goto      B5
        btfsc     PORTB,4
        goto      B4
        btfsc     PORTB,3
        goto      B3
        btfsc     PORTB,2
        goto      B2
        btfsc     PORTB,1
        goto      B1

B0:     movlw     0x03
        xorwf     PORTC,F
        goto      L3
B1:     movlw     0x07
        xorwf     PORTC,F
        goto      L3
B2:     movlw     0x0E
        xorwf     PORTC,F
        goto      L3
B3:     movlw     0x1C
        xorwf     PORTC,F
        goto      L3
B4:     movlw     0x38
        xorwf     PORTC,F
        goto      L3
B5:     movlw     0x70
        xorwf     PORTC,F
        goto      L3
B6:     movlw     0xE0
        xorwf     PORTC,F
        goto      L3
B7:     movlw     0xC0
        xorwf     PORTC,F
        goto      L3

L3:     incf      PORTD,F

L4:     movlw     0
        cpfseq    PORTB
        goto      L4
        goto      L1
        end
```



9) Test your code.

- Compile and program your PIC board
- Verify each button's operation

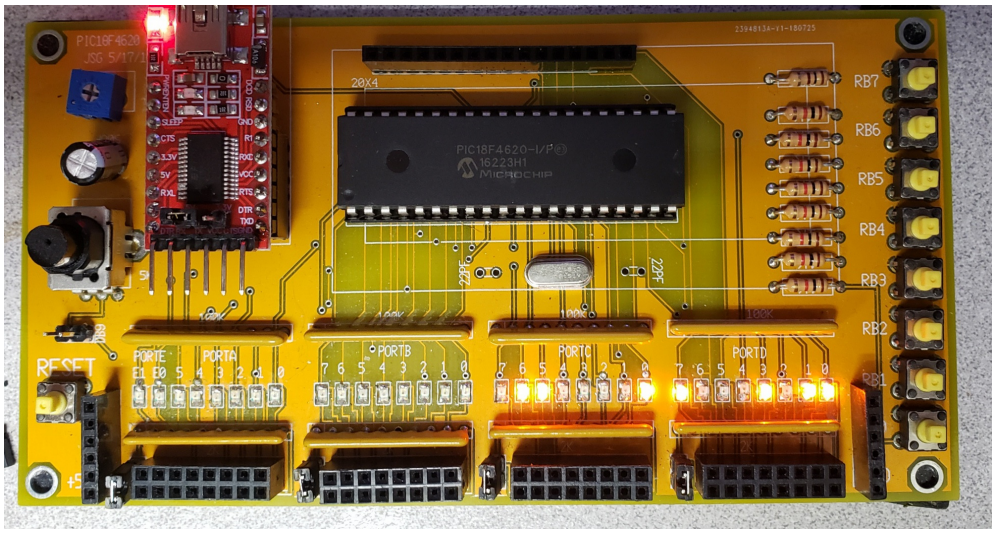
Checking:

- RB0 toggles lights 0,1 (check)
- RB1 toggles lights 0,1,2 (check)
- :
- RB6 toggles lights 5,6,7 (check)
- RB7 toggles lights 6,7 (check)
- PORTD counts with each button press (check)

There are some double counts, though. Adding debouncing would be good.

10) (20 points) Demonstration

- In-person of with a video



Lights-Out Game: As you press a button, that light and it's neighbors toggle on PORTC