**Final Project Proposal**

Bryan Greener

**Introduction**

I will be following the base design for the final project. I feel that this will be a good way for me to learn how to use multiple timers alongside each other.

**Components/Modules**

This design uses an RGB LED, the onboard button on P1.3, a potentiometer, and a 4 digit 7-seg. Please refer to the picture at the end of this document for my planned wiring and board layout.

Starting with the RGB LED, I decided to use pins 1.2, 1.6, and 2.1. This means that I will be using TA0CCR1 and TA1CCR1 to control the PWM for each of the lights. I had to split these into two different timers since no single timer on the board controls three physical pins. In the diagram, the wire colors are associated with the color of each LED and black is ground.

Next, the potentiometer is using P1.4 for its analog output and VCC for power. I am using P1.4 since I know that it works with the potentiometer and it doesn’t make the layout that much more confusing than it already is. Both the RBG LED and the potentiometer are using the same grounding pin on the board. In the diagram, red is VCC, yellow is data, and black is ground.

Finally, the 7-segment display is using wires all over the board. First off, to control two digits on the right of the display I have plugged P8 on the 7-seg into P2.4 on the PCB, P6 on the 7-seg into P2.5 on the PCB, and P2 on the 7-seg into P2.2 on the PCB. P2.4 and P2.5 on the PCB will be using TA1CCR2 and P2.2 will be using TA1CCR1. After that, each of the segments of the 7-seg fills any remaining unused pins on the board since nothing else needs to be plugged in. In the diagram P2, P6, and P8 on the 7-seg are purple and each of the segment pins are grey to avoid making the diagram more confusing.

**Pseudocode**

This program can all be run from a single C file.

int currentColor // Used in button interrupt to change color

int red, green, blue // Store intensity of color

int digit // Used to flip digits for PWM

Define constants for letters to display on 7seg

void main(void)

Enable Output Pins

Enable Interrupts

Initialize TA0 and TA1 timers as disabled

Initialize ADC10

Enable TA1

Set ADC10 result to TA1CCR1

select red pin

for(;;){ }

TA1CCTL1 Interrupt

int tempRed = red >> 2

Display ‘r’ in 2nd 7seg

IF digit = 0

Display first hex digit of red (TA1CCR1) value in 3rd 7seg

digit = 1

ELSE

Display second hex digit of red value in 4th 7seg

digit = 0

TA1CCTL2 Interrupt

IF currentColor = green

Display ‘g’ in 2nd 7seg

IF digit = 0 // Left digit

Display first hex digit of green (TA1CCR2) in 3rd 7seg

digit = 1

ELSE

Display second hex digit of green in 4th 7seg

digit = 0

ELSE IF currentColor = blue

Display ‘b’ in 2nd 7seg

IF digit = 0 // Left digit

Display first hex digit of blue in 3rd 7seg

digit = 1

ELSE

Display second hex digit of blue in 4th 7seg

digit = 0

P1.3 Interrupt

IF currentColor is red

red = TA1CCR1

Disable TA1CCTL1

Disable P2SEL

Enable TA0CCTL1

TA0CCR1 = ADC10 result

Select green pin

currentColor = green

IF currentColor is green

green = TA0CCR1

Select blue pin

currentColor = blue

IF currentColor is blue

blue = TA0CCR1

Disable TA0CCTL1

Disable P1SEL

Enable TA1CCTL1

Select red pin

currentColor = red

