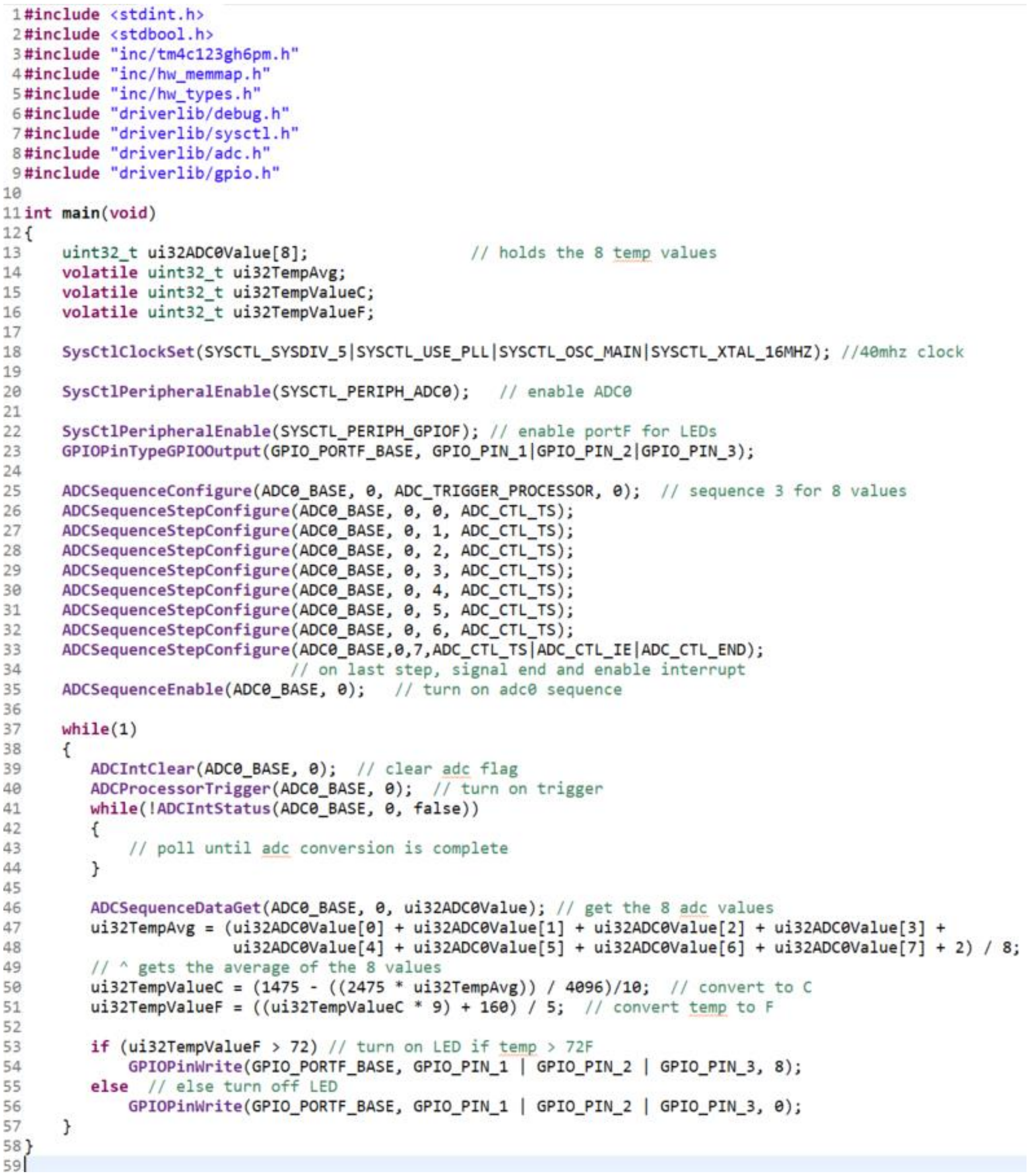
Brian Lopez
Github root directory: github.com/brian4280/Bed_Assign

Date Submitted: 10/8/18

Task 01:

Youtube Link: https://www.youtube.com/watch?v=-h3cZ64puYc

Modified Code:



```c
#include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"
#include "driverlib/gpio.h"

int main(void)
{
    uint32_t ui32ADC0Value[8];                  // holds the 8 temp values
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;

    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ); //40mhz clock

    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);    // enable ADC0

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); // enable portF for LEDs
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    ADCSequenceConfigure(ADC0_BASE, 0, ADC_TRIGGER_PROCESSOR, 0);  // sequence 3 for 8 values
    ADCSequenceStepConfigure(ADC0_BASE, 0, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 2, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 3, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 4, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 5, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 6, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE,0,7,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
                            // on last step, signal end and enable interrupt
    ADCSequenceEnable(ADC0_BASE, 0);   // turn on adc0 sequence

    while(1)
    {
       ADCIntClear(ADC0_BASE, 0);  // clear adc flag
       ADCProcessorTrigger(ADC0_BASE, 0);  // turn on trigger
       while(!ADCIntStatus(ADC0_BASE, 0, false))
       {
           // poll until adc conversion is complete
       }

       ADCSequenceDataGet(ADC0_BASE, 0, ui32ADC0Value); // get the 8 adc values
       ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] +
                      ui32ADC0Value[4] + ui32ADC0Value[5] + ui32ADC0Value[6] + ui32ADC0Value[7] + 2) / 8;
       // ^ gets the average of the 8 values
       ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;  // convert to C
       ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;  // convert temp to F

       if (ui32TempValueF > 72) // turn on LED if temp > 72F
           GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3, 8);
       else  // else turn off LED
           GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3, 0);
    }
}
```

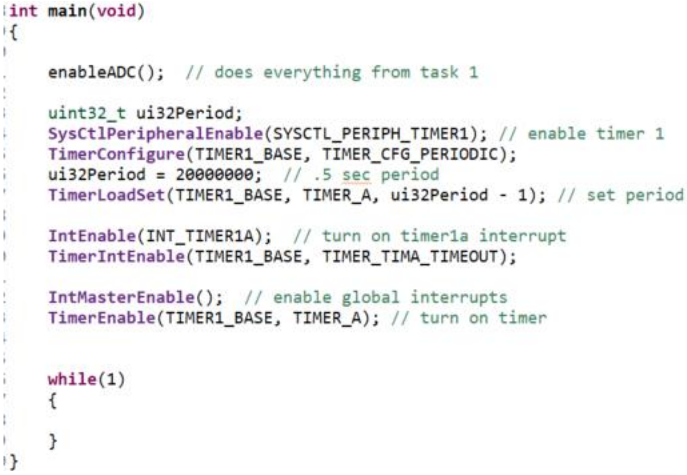
Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

Task 02:

Youtube Link: https://www.youtube.com/watch?v=W3BS9v3Dmpk

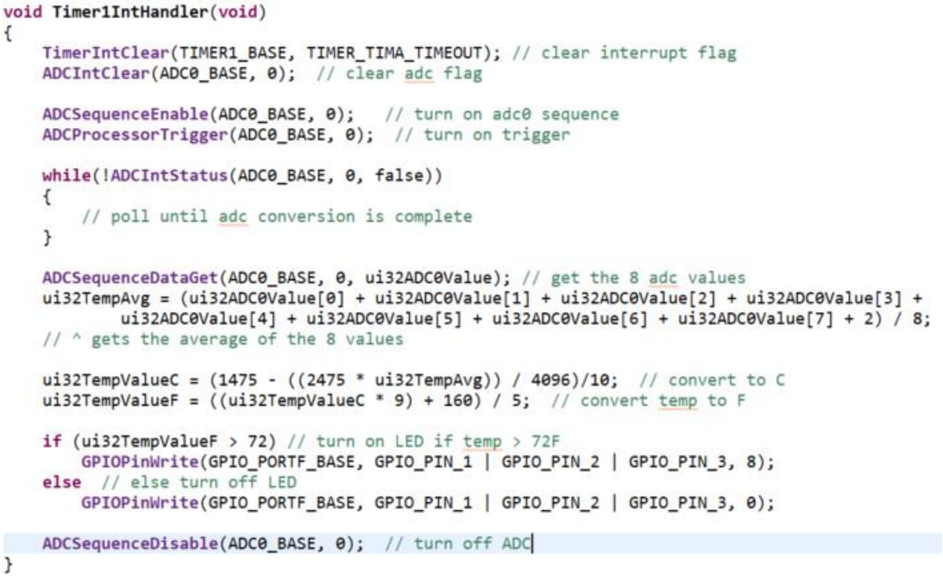
Modified Schematic (if applicable):

Modified Code:
Main function turns on ADC (I made into a function since it's the same as task 1), Then sets up timer1 for an interrupt with a period of .5 sec.



```
int main(void)
{

    enableADC();  // does everything from task 1

    uint32_t ui32Period;
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1); // enable timer 1
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
    ui32Period = 20000000;  // .5 sec period
    TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Period - 1); // set period

    IntEnable(INT_TIMER1A);  // turn on timer1a interrupt
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);

    IntMasterEnable();  // enable global interrupts
    TimerEnable(TIMER1_BASE, TIMER_A); // turn on timer


    while(1)
    {

    }
}
```

The Timer Interrupt function turns on ADC, polls until it's complete, then will calculate the temperatures. After that it will turn on the LED if the temp is over 72F and off otherwise. Then it will turn off ADC.



```
void Timer1IntHandler(void)
{
    TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT); // clear interrupt flag
    ADCIntClear(ADC0_BASE, 0);  // clear adc flag

    ADCSequenceEnable(ADC0_BASE, 0);   // turn on adc0 sequence
    ADCProcessorTrigger(ADC0_BASE, 0);  // turn on trigger

    while(!ADCIntStatus(ADC0_BASE, 0, false))
    {
        // poll until adc conversion is complete
    }

    ADCSequenceDataGet(ADC0_BASE, 0, ui32ADC0Value); // get the 8 adc values
    ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] +
            ui32ADC0Value[4] + ui32ADC0Value[5] + ui32ADC0Value[6] + ui32ADC0Value[7] + 2) / 8;
    // ^ gets the average of the 8 values

    ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;  // convert to C
    ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;  // convert temp to F

    if (ui32TempValueF > 72) // turn on LED if temp > 72F
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3, 8);
    else  // else turn off LED
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3, 0);

    ADCSequenceDisable(ADC0_BASE, 0);  // turn off ADC
}
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.