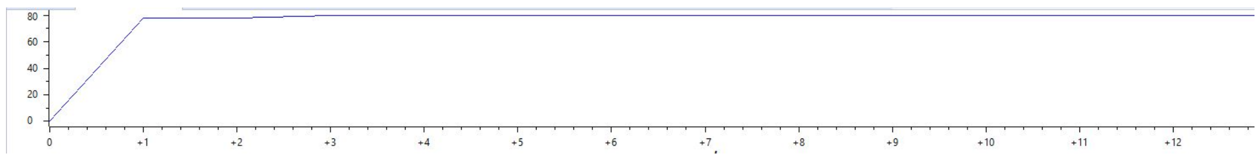
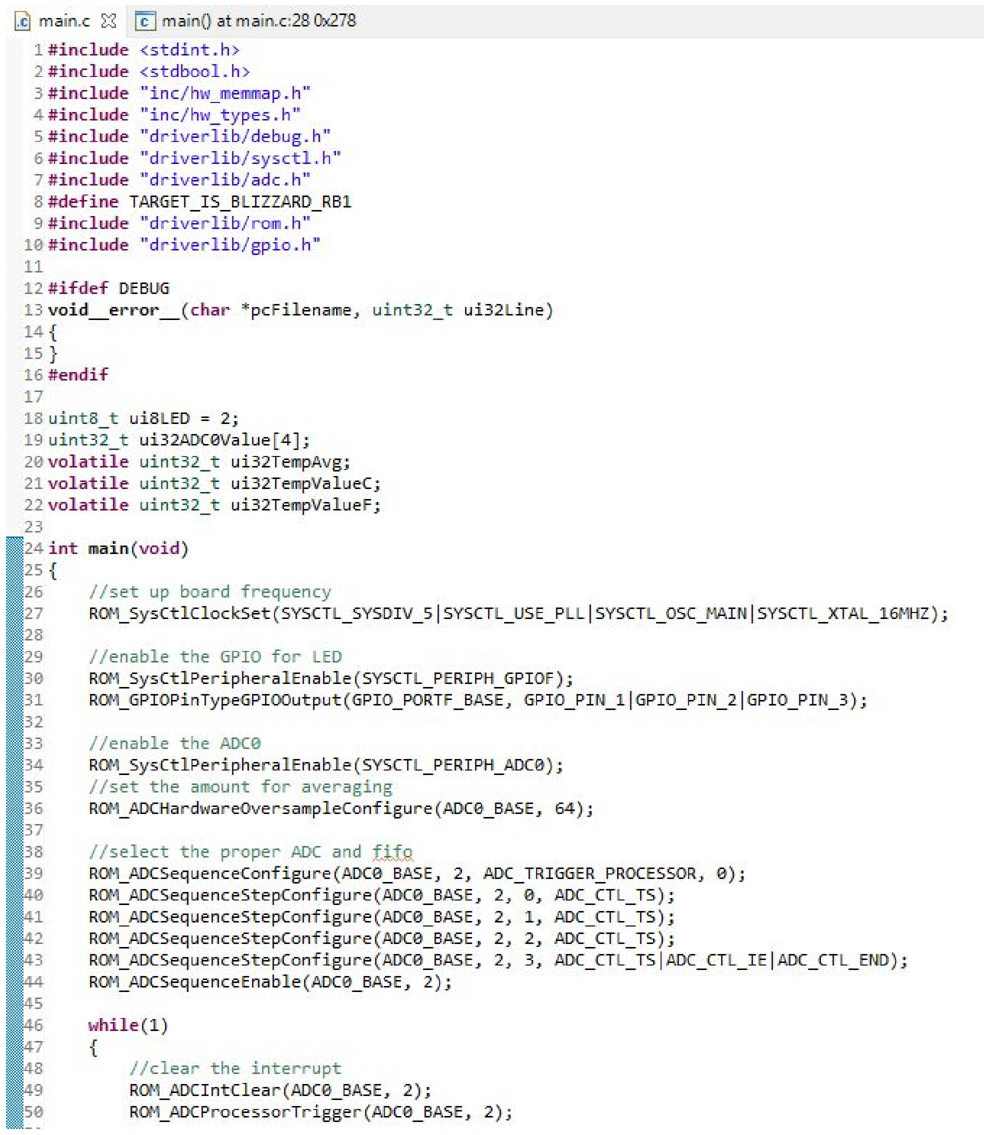
Joseph Sharp Halpin
CpE 403 Section 1001
10/9/2018

Task 01:
Youtube Link: https://youtu.be/cTRk370QT_M

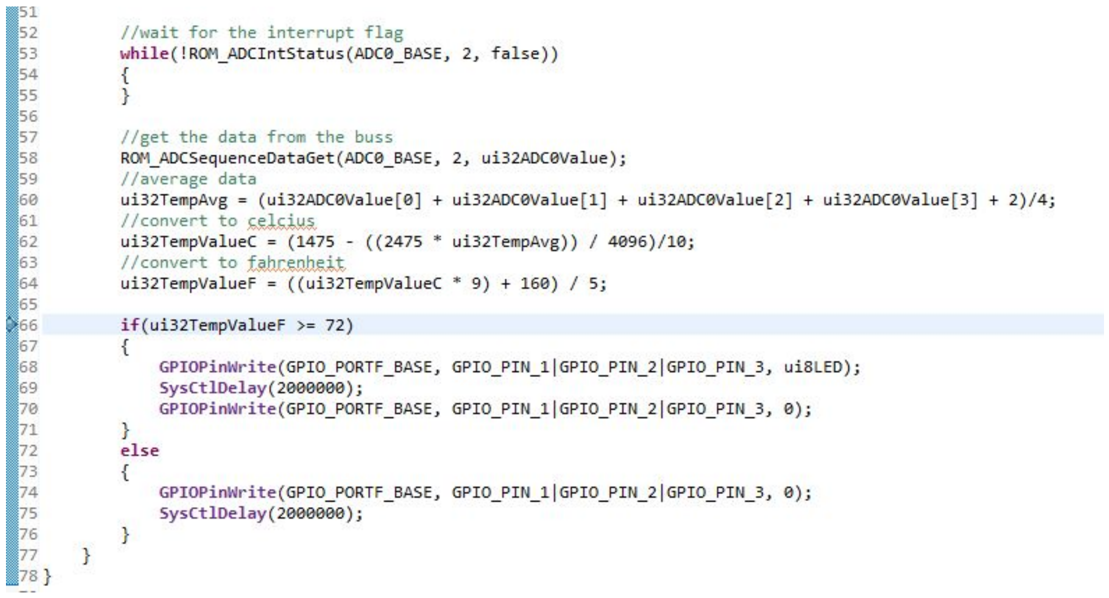
Graph:



Code:



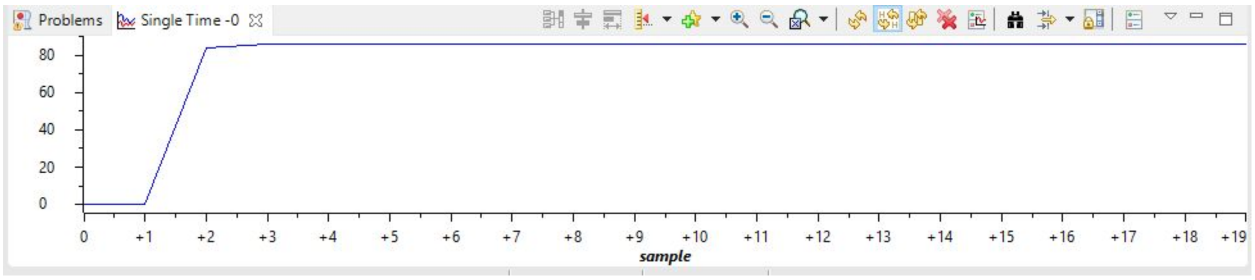
```c
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"
#define TARGET_IS_BLIZZARD_RB1
#include "driverlib/rom.h"
#include "driverlib/gpio.h"

#ifdef DEBUG
void__error__(char *pcFilename, uint32_t ui32Line)
{
}
#endif

uint8_t ui8LED = 2;
uint32_t ui32ADC0Value[4];
volatile uint32_t ui32TempAvg;
volatile uint32_t ui32TempValueC;
volatile uint32_t ui32TempValueF;

int main(void)
{
    //set up board frequency
    ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    //enable the GPIO for LED
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    ROM_GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    //enable the ADC0
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    //set the amount for averaging
    ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 64);

    //select the proper ADC and fifo
    ROM_ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 3, ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
    ROM_ADCSequenceEnable(ADC0_BASE, 2);

    while(1)
    {
        //clear the interrupt
        ROM_ADCIntClear(ADC0_BASE, 2);
        ROM_ADCProcessorTrigger(ADC0_BASE, 2);
```



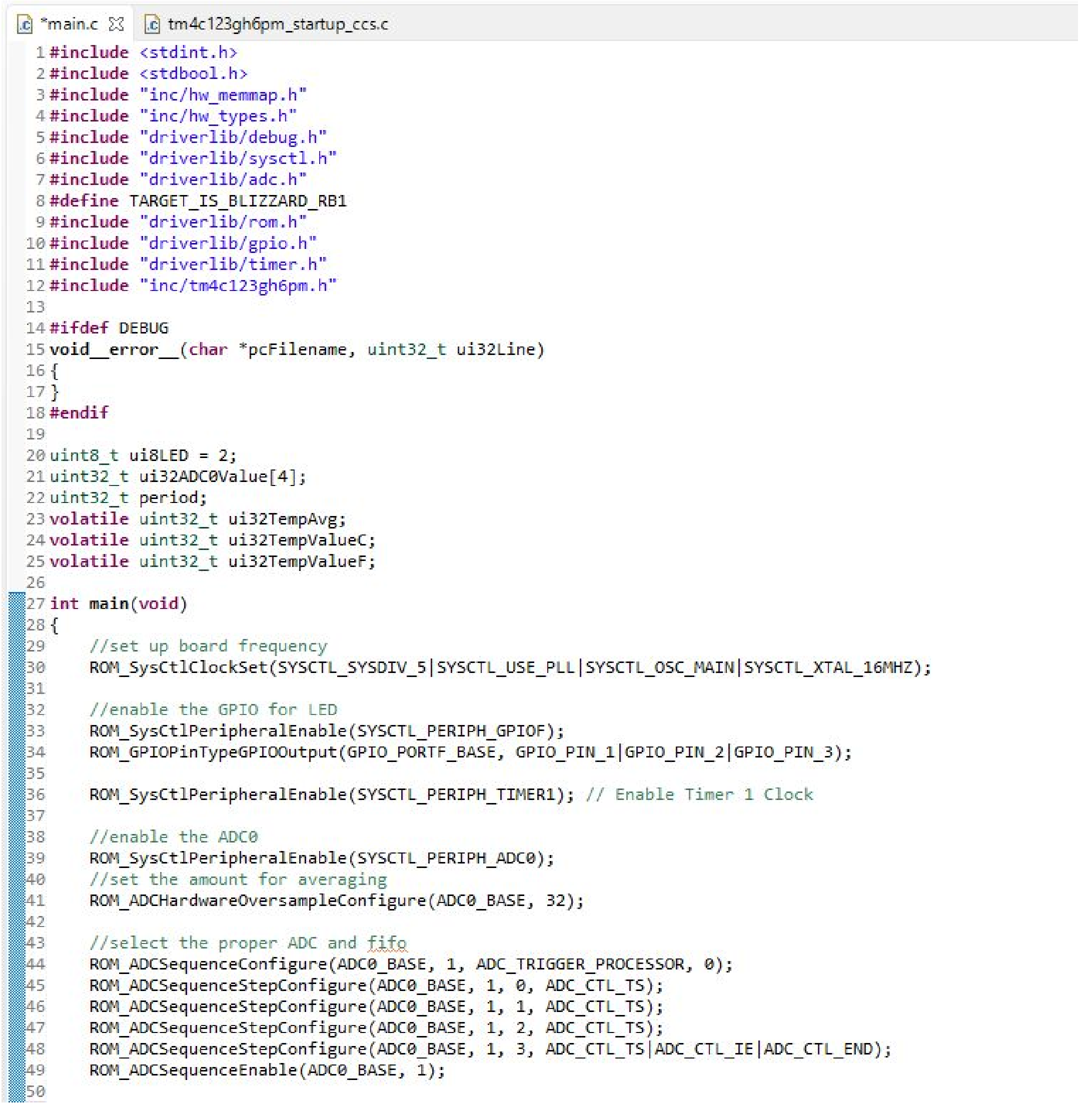
```c

        //wait for the interrupt flag
        while(!ROM_ADCIntStatus(ADC0_BASE, 2, false))
        {
        }

        //get the data from the buss
        ROM_ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);
        //average data
        ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
        //convert to celcius
        ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
        //convert to fahrenheit
        ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;

        if(ui32TempValueF >= 72)
        {
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, ui8LED);
            SysCtlDelay(2000000);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        }
        else
        {
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
            SysCtlDelay(2000000);
        }
    }
}
```

Task 02:
Youtube Link: https://youtu.be/14nXd1FUeyw

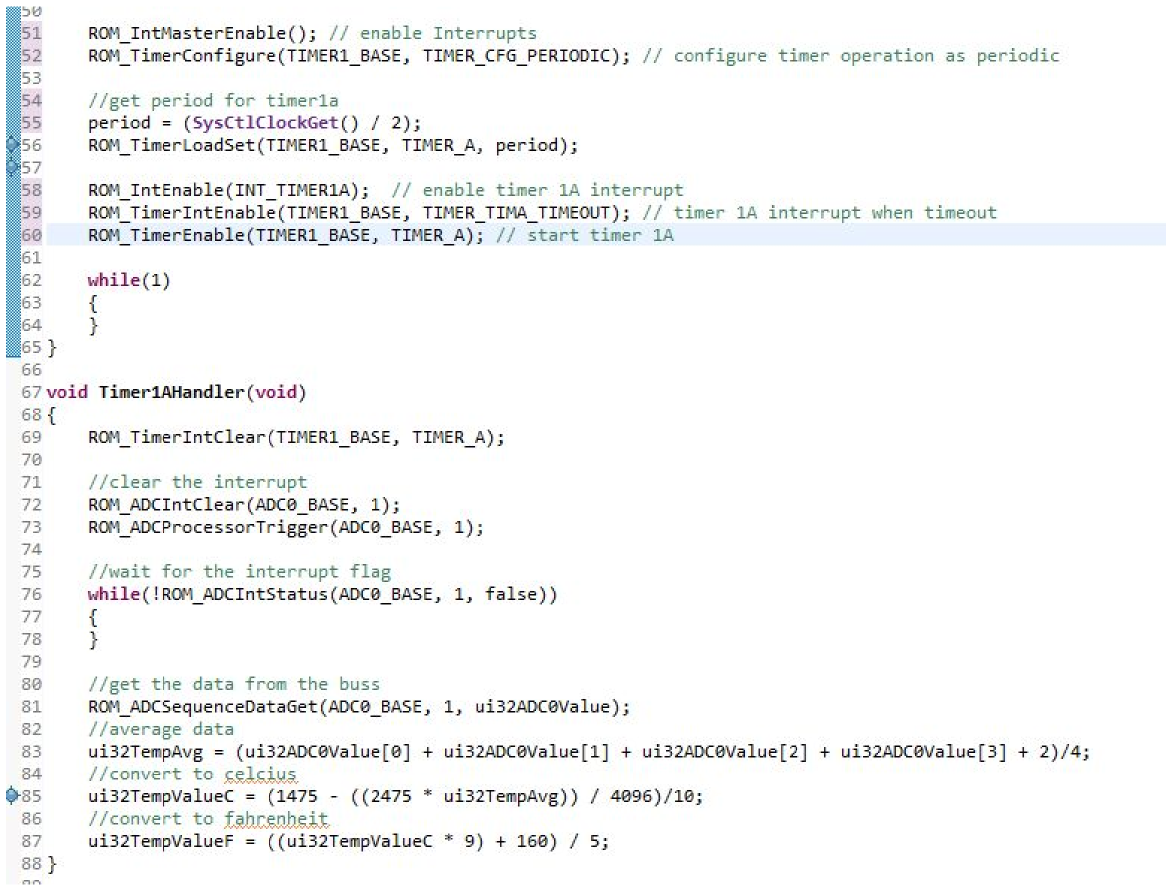
Graph:



Code:



```c
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"
#define TARGET_IS_BLIZZARD_RB1
#include "driverlib/rom.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "inc/tm4c123gh6pm.h"

#ifdef DEBUG
void__error__(char *pcFilename, uint32_t ui32Line)
{
}
#endif

uint8_t ui8LED = 2;
uint32_t ui32ADC0Value[4];
uint32_t period;
volatile uint32_t ui32TempAvg;
volatile uint32_t ui32TempValueC;
volatile uint32_t ui32TempValueF;

int main(void)
{
    //set up board frequency
    ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    //enable the GPIO for LED
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    ROM_GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1); // Enable Timer 1 Clock

    //enable the ADC0
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    //set the amount for averaging
    ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 32);

    //select the proper ADC and fifo
    ROM_ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 3, ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
    ROM_ADCSequenceEnable(ADC0_BASE, 1);

```



```c
    ROM_IntMasterEnable(); // enable Interrupts
    ROM_TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC); // configure timer operation as periodic

    //get period for timer1a
    period = (SysCtlClockGet() / 2);
    ROM_TimerLoadSet(TIMER1_BASE, TIMER_A, period);

    ROM_IntEnable(INT_TIMER1A);  // enable timer 1A interrupt
    ROM_TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT); // timer 1A interrupt when timeout
    ROM_TimerEnable(TIMER1_BASE, TIMER_A); // start timer 1A

    while(1)
    {
    }
}

void Timer1AHandler(void)
{
    ROM_TimerIntClear(TIMER1_BASE, TIMER_A);

    //clear the interrupt
    ROM_ADCIntClear(ADC0_BASE, 1);
    ROM_ADCProcessorTrigger(ADC0_BASE, 1);

    //wait for the interrupt flag
    while(!ROM_ADCIntStatus(ADC0_BASE, 1, false))
    {
    }

    //get the data from the buss
    ROM_ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
    //average data
    ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
    //convert to celcius
    ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
    //convert to fahrenheit
    ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
}
```