Nathan Hanuscin
Github root directory: https://github.com/nhanuscin/HappyFunStuff

Date Submitted: **10/01/19**

Task 01:

Youtube Link: https://www.youtube.com/watch?v=GM712P_yRxs

Modified Schematic (if applicable): N/A

Modified Code:

```
#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"
#include "driverlib/gpio.h"

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

uint8_t ui8PinData=4;

int main(void)
{
    uint32_t ui32ADC0Value[4];
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;

    //Set Clock to 40MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    //Enable ADC0 peripheral
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    //Enable PortF
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00); //All off

    //Configure ADC sequencer 1
    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
    //Sample Internal temp sensor with sequencer 1
    ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);
    //Sample temp sensor, set interrupt flag to end conversion, enable ADC
    ADCSequenceStepConfigure(ADC0_BASE,1,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceEnable(ADC0_BASE, 1);
```

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

```
    while(1)
    {
            //Clear interrupt flag
            ADCIntClear(ADC0_BASE, 1);
            //Trigger ADC conversion with software
            ADCProcessorTrigger(ADC0_BASE, 1);

            //wait for ADC conversion to finish
            while(!ADCIntStatus(ADC0_BASE, 1, false))
            {
            }
            //Get ADC values from SS1
            ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
            //Average and Calculate Temperature
            ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;
            ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
            ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
            //turn on Blue LED if temp is > 72F
            if(ui32TempValueF > 72)
                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
ui8PinData);
            else
                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x0);

    }
}
```

Nathan Hanuscin
Github root directory: https://github.com/nhanuscin/HappyFunStuff

Task 02:

Youtube Link: https://www.youtube.com/watch?v=kN5DgYvDVT0

Modified Schematic (if applicable):N/A

Modified Code:

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

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

uint8_t ui8PinData=4;
uint32_t ui32ADC0Value[4];
volatile uint32_t ui32TempAvg;
volatile uint32_t ui32TempValueC;
volatile uint32_t ui32TempValueF;

int main(void)
{

    uint32_t ui32Period;

    //Set Clock to 40MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    //Enable ADC0 peripheral and hardware averaging
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ADCHardwareOversampleConfigure(ADC0_BASE, 32);
    //Enable PortF
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00); //All off
    //Enable Timer1A
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

    //Configure ADC sequencer 1
    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
    //Sample Internal temp sensor with sequencer 1
    ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
```

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

```
    ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);
    //Sample temp sensor, set interrupt flag to end conversion, enable ADC
    ADCSequenceStepConfigure(ADC0_BASE,1,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceEnable(ADC0_BASE, 1);

    //Set timer to 2Hz
    ui32Period =  SysCtlClockGet()/ 2;
    TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Period -1);
    //Enable timer interrupt
    IntEnable(INT_TIMER1A);
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable();

    TimerEnable(TIMER1_BASE, TIMER_A);

    while(1)
    {
    //wait for timer interrupt every .5 seconds for ADC conversion
    }
}

void Timer1IntHandler(void)
{
    uint32_t ui32Period2;
    // Clear the timer interrupt
    TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);

    //Clear interrupt flag
    ADCIntClear(ADC0_BASE, 1);
    //Trigger ADC conversion with software
    ADCProcessorTrigger(ADC0_BASE, 1);

    //wait for ADC conversion to finish
    while(!ADCIntStatus(ADC0_BASE, 1, false))
    {
    }
    //Get ADC values from SS1
    ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
    //Average and Calculate Temperature
    ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;
    ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
    ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
    if(ui32TempValueF > 72)
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, ui8PinData);
    else
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x0);
    ui32Period2 =  SysCtlClockGet()/ 2;
    TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Period2 -1);

}
```

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