CPE403 – Advanced Embedded Systems

Design Assignment #3

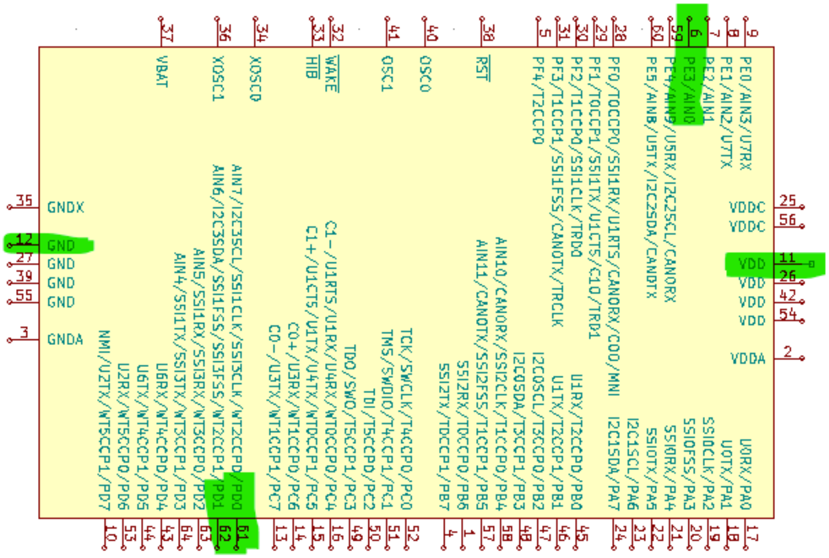
Name: Brian Wolak

Email: wolak@unlv.nevada.edu

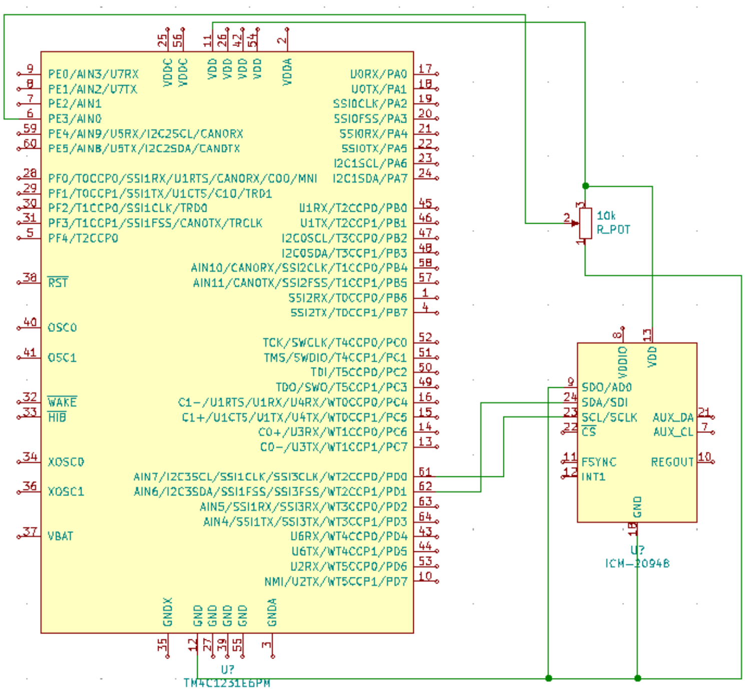
GitHub Repository link (root): https://github.com/brianwolak/advanced_submissions

YouTube Playlist (root): https://youtube.com/playlist?list=PLl6a3M--0IcYnZIMGyucOLe8c-16wAClN

Goal of this assignment is to create five tasks, 1) ADC task, 2) UART display task, 3) Switch Read task, 4) I2C Sensor read task, and 5) Heartbeat function (PF3). The heartbeat function is performed throughout the execution of the program. Each task will be executed in order specified above every 20 ms. Connect a potentiometer to the ADC pin. Use ADC0 CH4. Also initialize a PWM signal to a LED (PF1). Initial value of the PWM duty cycle is set to 0. Create a timer 0/1/2 HWI for every 1 ms, at 5th instance of HWI the task ADC is performed, at 10th instance of HWI the task UART displays the current value ADC in the terminal, at 15th instance of HWI the task Switch Read is performed to check the status of the SW1/SW2 to update the current value of duty cycle based on the ADC value, and at the 20th instance of HWI perform any I2C sensor read task. Note that the duty cycle of the PWM does not change unless the switch is pressed, even when the ADC value changes. However, the UART should display the dynamic and current active values of the ADC, and the I2C sensor data. Use semaphores to switch between the tasks.



Tiva-C TM4C123 Pins Used



Tiva-C TM4C123 & IMU-20948 Circuit

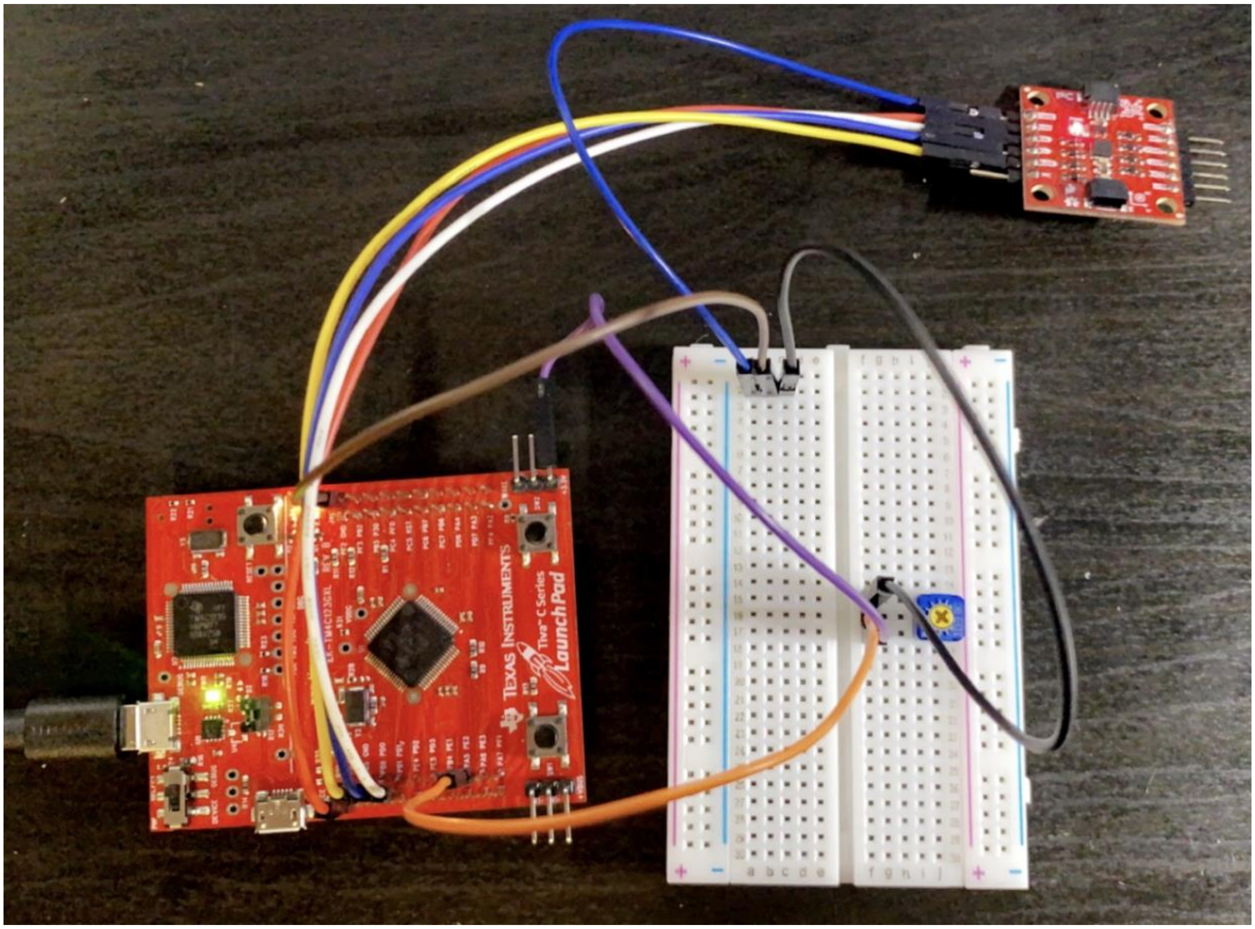
C Code:

```c
#include <xdc/std.h>
#include <xdc/runtime/System.h>
#include <xdc/runtime/Log.h>
#include <xdc/cfg/global.h>
#include <xdc/runtime/Diags.h>
#include <ti/sysbios/BIOS.h>
#include <ti/sysbios/knl/Clock.h>
#include <ti/sysbios/knl/Task.h>
#include <ti/sysbios/knl/Semaphore.h>
#include "Board.h"
#include <ti/drivers/GPIO.h>
#include <ti/drivers/I2C.h>
#include <ti/drivers/PWM.h>
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "inc/hw_i2c.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "inc/tm4c123gh6pm.h"
#include "driverlib/debug.h"
#include "driverlib/pin_map.h"
#include "driverlib/adc.h"
#include "driverlib/rom.h"
#include "driverlib/interrupt.h"
```

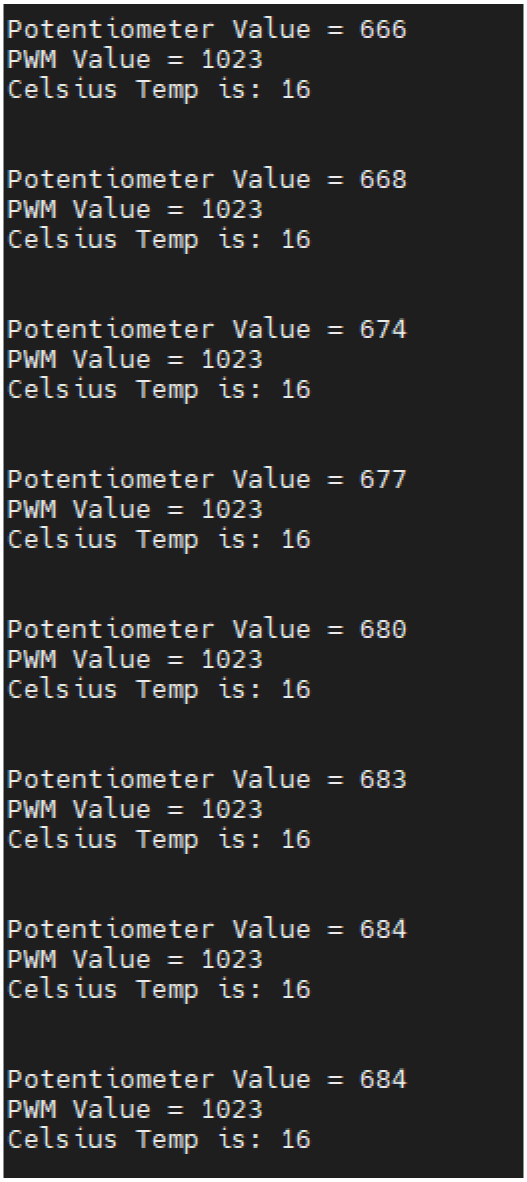
```c
#include "driverlib/timer.h"
#include <time.h>
#include <inc/hw_gpio.h>
#include "driverlib/uart.h"
#include "utils/uartstdio.h"
#include <string.h>


// global variables
uint32_t ADCread[4];
uint32_t ui32Period;
uint32_t PWMsave;
uint16_t output;                                // I2C read output
uint16_t pwmPeriod = 3000;                      // pwm period
uint16_t dutyCycle = 0;                         // initial duty cycle
volatile uint32_t ui32PWMClock;
volatile uint32_t ui32Load;
volatile uint32_t ui8Adjust;
volatile uint32_t ADCavg;
volatile uint32_t tickCount = 0;

void timer2Config(){
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER2);                     // timer2
enable
    TimerConfigure(TIMER2_BASE, TIMER_CFG_PERIODIC);                  // timer2
configure periodic mode
    ui32Period = (SysCtlClockGet()/500) / 4;
    TimerLoadSet(TIMER2_BASE, TIMER_A, ui32Period-1);                 // set timer2
period
    TimerIntEnable(TIMER2_BASE, TIMER_TIMA_TIMEOUT);                  // enable
timeout interrupt
    TimerEnable(TIMER2_BASE, TIMER_A);                                // enable
timer2
}

void adcConfig(){
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ADCHardwareOversampleConfigure(ADC0_BASE, 64);
    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_CH1);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_CH1);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_CH1);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 3, ADC_CTL_CH1|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceEnable(ADC0_BASE, 1);
}

void UART0Config(void){
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);                        // enable
UART0
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);                        // enable
GPIO PORTA
    GPIOPinConfigure(GPIO_PA0_U0RX);                                    // pin PA0 rx
    GPIOPinConfigure(GPIO_PA1_U0TX);                                    // pin PA1 tx
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
    UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);                  // set UART0
clock
    UARTStdioConfig(0, 115200, 16000000);                              // set baud
rate 115200
}

// ADC read function
void taskFunction1(void){
    while(1){
    Semaphore_pend(semaphore1, BIOS_WAIT_FOREVER);
    ADCIntClear(ADC0_BASE, 1);
    ADCProcessorTrigger(ADC0_BASE, 1);
    while(!ADCIntStatus(ADC0_BASE, 1, false)){}
    ADCSequenceDataGet(ADC0_BASE, 1, ADCread);
    ADCavg = ((ADCread[0] + ADCread[1] + ADCread[2] + ADCread[3])/16);
    }
}

// ADC & I2C print function
void taskFunction2(void){
    while(1){
        Semaphore_pend(semaphore2, BIOS_WAIT_FOREVER);
        UARTprintf("\nPotentiometer Value = %d \n", ADCavg);    // print pot value
        UARTprintf("PWM Value = %d\n", PWMsave);                // print PWM value
from pot
        UARTprintf("Celsius Temp is: %d \n\n", output);         // print celsius temp
from ICM20948
    }
}

// PWM adjust function
void taskFunction3(UArg arg0, UArg arg1){
    PWM_Handle PWM1;
    PWM_Params PWMparams;

    PWM_Params_init(&PWMparams);
    PWMparams.period = pwmPeriod;
    PWM1 = PWM_open(Board_PWM1, &PWMparams);

    if (PWM1 == NULL){
        System_abort("Warning! Board_PWM1 was unable to open!");
    }

    while(1){
        Semaphore_pend(semaphore3, BIOS_WAIT_FOREVER);
        if(GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_0)==0x00){
            PWMsave = ADCavg;
            dutyCycle = (ADCavg*3000) / 1023;                    // ADCavg
calculation for duty cycle
            PWM_setDuty(PWM1, dutyCycle);                        // set PWM1 duty
cycle
        }
    }
}
```

```
// ICM20948 I2C read function reading 0x39 temp sensor high value
void taskFunction4(void){
    while(1){
        Semaphore_pend(semaphore4, BIOS_WAIT_FOREVER);
        unsigned int    i;
        uint8_t         txBuffer[3] = {0x06, 0x01, 0x39};
        uint8_t         rxBuffer[2];
        I2C_Handle      i2c;
        I2C_Params      i2cParams;
        I2C_Transaction i2cTransaction;

        /* Create I2C for usage */
        I2C_Params_init(&i2cParams);
        i2cParams.bitRate = I2C_400kHz;
        i2c = I2C_open(Board_I2C_TMP, &i2cParams);

        /* Point to the T ambient register and read its 2 bytes */
        i2cTransaction.slaveAddress = 0x68;
        i2cTransaction.writeBuf = txBuffer;
        i2cTransaction.writeCount = 3;
        i2cTransaction.readBuf = rxBuffer;
        i2cTransaction.readCount = 2;

        /* Take 20 samples and print them out onto the console */
        for (i = 0; i < 1; i++) {
            if (I2C_transfer(i2c, &i2cTransaction)) {
            /* Extract degrees C from the received data; see TMP102 datasheet */
            output = (rxBuffer[0] << 6) | (rxBuffer[1] >> 2);
            /*
             * If the MSB is set '1', then we have a 2's complement
             * negative value which needs to be sign extended
            */
            if (rxBuffer[0] & 0x80) {
                output |= 0xF000;
            }
            /*
             * For simplicity, divide the temperature value by 32 to get rid of
             * the decimal precision; see TI's TMP006 datasheet
            */
            output /= 32;
            }
            else {
                System_printf("I2C Bus fault\n");
            }
        }
        /* Deinitialized I2C */
        I2C_close(i2c);
    }
}

// heartbeat function
void taskFunction5(UArg arg0, UArg arg1){
    while(1){
        Semaphore_pend(semaphore5, BIOS_WAIT_FOREVER);
        Task_sleep((UInt)arg0);
```

```c
        GPIO_toggle(Board_LED1);
    }
}

int main(){
    // set system clock to 40Mhz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    // enable peripherals
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    // unlock PF0 pin
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK)= GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK)= 0;
    GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_0);
    GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);

    // enable GPIO pin 1 & 2 as outputs
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_3);

    timer2Config();                 // timer2 setup
    adcConfig();                    // ADC setup
    UART0Config();                  // UART0 115200 baud rate setup
//initialize the IMU20948
    Board_initGeneral();            // board setup
    Board_initGPIO();
    Board_initI2C();
    Board_initPWM();
    Board_initUART();

    GPIO_write(Board_LED1, Board_LED_ON);
    BIOS_start();                   // start bios
    System_printf("Starting\n");
    System_flush();
}

void Timer_ISR(void){
    TimerIntClear(TIMER2_BASE, TIMER_TIMA_TIMEOUT);    // clear timeout flag
    tickCount++;                                       // increment count
    Semaphore_post(semaphore5);     // heartbeat every 1ms

    if(tickCount == 5){             // read ADC on 5th HWI
        Semaphore_post(semaphore1);
    }
    else if(tickCount == 10){       // display ADC on 10th HWI
        Semaphore_post(semaphore2);
    }
    else if(tickCount == 15){       // switch read / PWM update on 15th HWI
        Semaphore_post(semaphore3);
    }
    else if(tickCount == 20){       // I2C read on 20th HWI
        Semaphore_post(semaphore4);
        tickCount = 0;
    }
}
```



TM4C123GXL Board Connections to 10k Potentiometer and ICM20948



```
Potentiometer Value = 666
PWM Value = 1023
Celsius Temp is: 16


Potentiometer Value = 668
PWM Value = 1023
Celsius Temp is: 16


Potentiometer Value = 674
PWM Value = 1023
Celsius Temp is: 16


Potentiometer Value = 677
PWM Value = 1023
Celsius Temp is: 16


Potentiometer Value = 680
PWM Value = 1023
Celsius Temp is: 16


Potentiometer Value = 683
PWM Value = 1023
Celsius Temp is: 16


Potentiometer Value = 684
PWM Value = 1023
Celsius Temp is: 16


Potentiometer Value = 684
PWM Value = 1023
Celsius Temp is: 16
```

Sample Terminal Output Displaying PWM Value, POT Value, and Temp

GitHub: https://github.com/brianwolak/advanced_submissions/tree/main/DA_3

YouTube: **https://youtu.be/lVcywIlbP9I**

“This assignment submission is my own, original work”.
Brian Wolak