CPE403 – Advanced Embedded Systems

Design Assignment 2

Name: Elmer Mejia

Email: mejiae4@unlv.nevada.edu

Github Repository link (root): assignments

Youtube Playlist link (root): Tiva_C

1. Code for Tasks. for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the base code for the first task only. Use separate page for each task.

```
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include <stdint.h>
#include "inc/hw_i2c.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "driverlib/i2c.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "utils/uartstdio.h"
#include "driverlib/uart.h"
#include "utils/uartstdio.h"
#include "math.h"
#include "IQmath/IQmathlib.h"
```

```
#include <string.h>
#include "icm20948_def.h"




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

#define ACCELEROMETER_SENSITIVITY 8192.0
#define GYROSCOPE_SENSITIVITY 16384.0
#define SAMPLE_RATE 0.01
#define RATIO (180/3.14159265359)

float ACC_Data, ACC_Data2, ACC_Data3;
float GYRO_Data, GYRO_Data2, GYRO_Data3;          // raw values
_iq16 Pitch =0 ;
_iq16 Roll = 0;
```

```
_iq16 Yaw = 0;

void initI2C0(void)
{
    // Turn on I2C0
    SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);
    SysCtlDelay(3);
    // Reset I2C0
    SysCtlPeripheralReset(SYSCTL_PERIPH_I2C0);
    SysCtlDelay(3);
    // Enable GPIOB
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
    SysCtlDelay(3);

    // Configure GPIO SCL/SDA pins on PB2/PB3
    GPIOPinConfigure(GPIO_PB2_I2C0SCL);
    GPIOPinConfigure(GPIO_PB3_I2C0SDA);

    // Set pins to I2C function
    GPIOPinTypeI2CSCL(GPIO_PORTB_BASE, GPIO_PIN_2);
    GPIOPinTypeI2C(GPIO_PORTB_BASE, GPIO_PIN_3);

    // Enable and master I2C
    I2CMasterInitExpClk(I2C0_BASE, SysCtlClockGet(), false);

    // Clear I2C FIFOs
    HWREG(I2C0_BASE + I2C_O_FIFOCTL) = 80008000;
```

```
}



void I2C0Read(uint8_t slave_addr, uint8_t reg, uint8_t *data)
{
    I2CMasterSlaveAddrSet(I2C0_BASE, slave_addr, false);
    I2CMasterDataPut(I2C0_BASE, reg);
    I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START);
    while(I2CMasterBusy(I2C0_BASE));
    I2CMasterSlaveAddrSet(I2C0_BASE, slave_addr, true);
    I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_RECEIVE);
    while(I2CMasterBusy(I2C0_BASE));
    *data = I2CMasterDataGet(I2C0_BASE);
}

// This function has not been tested - for using 16bit read you can
// also use the I2C0Read twice if this does not work
void I2C0Read16(uint8_t slave_addr, uint8_t reg, uint16_t *data)
{
    uint8_t HByte , LByte=0;
    I2CMasterSlaveAddrSet(I2C0_BASE, slave_addr, false);
    I2CMasterDataPut(I2C0_BASE, reg);
    I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START);
    while(I2CMasterBusy(I2C0_BASE));
    I2CMasterSlaveAddrSet(I2C0_BASE, slave_addr, true);
    I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_START);
```

```
    while(I2CMasterBusy(I2C0_BASE));
    HByte = I2CMasterDataGet(I2C0_BASE);
    I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_CONT);
    while(I2CMasterBusy(I2C0_BASE));
    LByte = I2CMasterDataGet(I2C0_BASE);
    *data = (LByte <<8 | HByte);
    I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_FINISH);
    while(I2CMasterBusy(I2C0_BASE));
}
```

```
void I2C0Write(uint8_t slave_addr, uint8_t reg, uint8_t data)
{
    I2CMasterSlaveAddrSet(I2C0_BASE, slave_addr, false);
    I2CMasterDataPut(I2C0_BASE, reg);
    I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START);
    while(I2CMasterBusy(I2C0_BASE));
    I2CMasterDataPut(I2C0_BASE, data);
    I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH);
    while(I2CMasterBusy(I2C0_BASE));
}
```

```
/*reads the slave device*/
void ICM_get_whom_am_I()
{
    uint8_t WAI=0;
    I2C0Write(ICM20948_ADDRESS, ICM20948_REG_PWR_MGMT_1,
ICM20948_REG_LP_CONFIG);
    SysCtlDelay(3);
    I2C0Write(ICM20948_ADDRESS, ICM20948_REG_BANK_SEL,
ICM20948_BANK_0);
    SysCtlDelay(3);
    I2C0Read(ICM20948_ADDRESS, ICM20948_REG_WHO_AM_I, &WAI);
    if (WAI != ICM20948_DEVICE_ID)
    UARTprintf("Device Not Found\n");
    else
    UARTprintf("Device Found\n");
}

/*Initializes the ICM20948 device*/
void Init_ICM()
{
    UARTprintf("I2C Initialized\n");
    SysCtlDelay(3);
    //ICM_get_whom_am_I();
}
```

```
void ICM20948_config(void)
{
    I2C0Write(ICM20948_ADDRESS, ICM20948_REG_PWR_MGMT_1,
ICM20948_REG_LP_CONFIG); // power on
    SysCtlDelay(3);
    I2C0Write(ICM20948_ADDRESS, ICM20948_REG_BANK_SEL,
ICM20948_BANK_2); // Bank 2 select
    SysCtlDelay(3);
    I2C0Write(ICM20948_ADDRESS, ICM20948_REG_GYRO_CONFIG_1, 0x00); // gyro
config
    SysCtlDelay(3);
    I2C0Write(ICM20948_ADDRESS, ICM20948_SHIFT_GYRO_FS_SEL, 0x00); // gyro
config
    SysCtlDelay(3);
    I2C0Write(ICM20948_ADDRESS, ICM20948_REG_ACCEL_CONFIG, 0x00); // accel
config
    SysCtlDelay(3);
    I2C0Write(ICM20948_ADDRESS, ICM20948_ACCEL_FULLSCALE_4G, 0x00); //
accel config
    SysCtlDelay(3);
```

```
    I2C0Write(ICM20948_ADDRESS, ICM20948_REG_BANK_SEL,
ICM20948_BANK_0); // Bank 0 select
    SysCtlDelay(3);
}

void Comp_Filter(void)
{
    _iq16 ForceMagnitudeApprox, PitchAcc, RollAcc, YawAcc, Qsens, QRATIO;
    _iq16 Gyro[3], Acc[3];
    _iq16 val1, val2;
    Pitch = 0;
    Roll = 0;
    Yaw = 0;

    QRATIO = _IQ16((float)RATIO);
    val1 = _IQ16((float)0.98);
    val2 = _IQ16((float)0.02);
    Gyro[0] = _IQ16((float)GYRO_Data);
    Gyro[1] = _IQ16((float)GYRO_Data2);
    Gyro[2] = _IQ16((float)GYRO_Data3);
    Acc[0] = _IQ16((float)ACC_Data);
    Acc[1] = _IQ16((float)ACC_Data2);
    Acc[2] = _IQ16((float)ACC_Data3);
    Qsens = _IQ16((float)GYROSCOPE_SENSITIVITY);

    Pitch += _IQ16mpy(_IQ16div(Gyro[0],Qsens), _IQ16((float)SAMPLE_RATE));
    Roll -= _IQ16mpy(_IQ16div(Gyro[1],Qsens), _IQ16((float)SAMPLE_RATE));
```

```
    Yaw += _IQ16mpy(_IQ16div(Gyro[2],Qsens), _IQ16((float)SAMPLE_RATE));
    ForceMagnitudeApprox = _IQabs(Acc[0]) + _IQabs(Acc[1]) + _IQabs(Acc[2]);

    if(ForceMagnitudeApprox > 8192 && ForceMagnitudeApprox < 32768)
    {
        PitchAcc = _IQ16mpy(_IQ16atan2(Acc[1],Acc[2]), QRATIO);
        Pitch = _IQ16mpy(Pitch,val1) + _IQ16mpy(PitchAcc,val2);
        RollAcc = _IQ16mpy(_IQ16atan2(Acc[0],Acc[2]), QRATIO);
        Roll = _IQ16mpy(Roll,val1) + _IQ16mpy(RollAcc,val2);
        YawAcc = _IQ16mpy(_IQ16atan2(Acc[0],Acc[1]), QRATIO);
        Yaw = _IQ16mpy(Yaw,val1) + _IQ16mpy(YawAcc,val2);
    }

    UARTprintf("Pitch  : %d  | Roll  : %d | Yaw  : %d \n\n", (int)Pitch, (int)Roll, (int)Yaw);
}


int main(void)
{
    // Set clock to 40 MHz

SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYS
CTL_OSC_MAIN);

    // Enable UART peripheral
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
```

```
    // Configure UART GPIO
    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);
    UARTStdioConfig(0, 115200, 16000000);

    UARTprintf("UART Initialized\n");

    uint8_t HByte , LByte=0;

    //Init I2C
    initI2C0();
    Init_ICM();
    ICM20948_config();
    SysCtlDelay(3);
    while(1)
    {
        I2C0Read(ICM20948_ADDRESS, ICM20948_REG_ACCEL_XOUT_H_SH, &HByte);
        I2C0Read(ICM20948_ADDRESS, ICM20948_REG_ACCEL_XOUT_L_SH, &LByte);
        ACC_Data = (LByte <<8 | HByte);
        I2C0Read(ICM20948_ADDRESS, ICM20948_REG_ACCEL_YOUT_H_SH, &HByte);
```

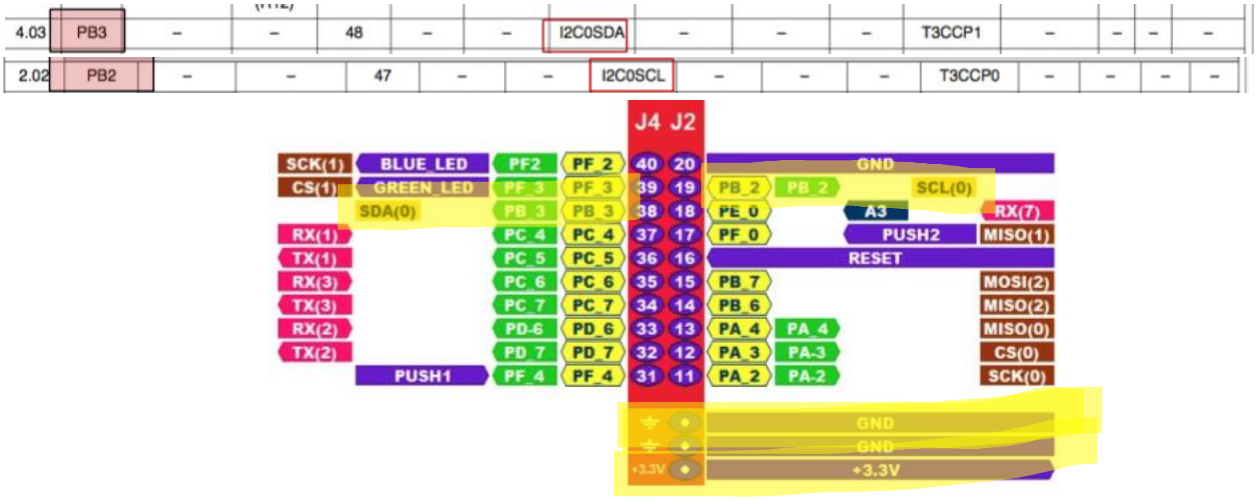
```
        I2C0Read(ICM20948_ADDRESS, ICM20948_REG_ACCEL_YOUT_L_SH,
&LByte);
        ACC_Data2 = (LByte <<8 | HByte);
        I2C0Read(ICM20948_ADDRESS, ICM20948_REG_ACCEL_ZOUT_H_SH,
&HByte);
        I2C0Read(ICM20948_ADDRESS, ICM20948_REG_ACCEL_ZOUT_L_SH,
&LByte);
        ACC_Data3 = (LByte <<8 | HByte);
        I2C0Read(ICM20948_ADDRESS, ICM20948_REG_GYRO_XOUT_H_SH,
&HByte);
        I2C0Read(ICM20948_ADDRESS, ICM20948_REG_GYRO_XOUT_L_SH, &LByte);
        GYRO_Data = (LByte <<8 | HByte);
        I2C0Read(ICM20948_ADDRESS, ICM20948_REG_GYRO_YOUT_H_SH,
&HByte);
        I2C0Read(ICM20948_ADDRESS, ICM20948_REG_GYRO_YOUT_L_SH, &LByte);
        GYRO_Data2 = (LByte <<8 | HByte);
        I2C0Read(ICM20948_ADDRESS, ICM20948_REG_GYRO_ZOUT_H_SH,
&HByte);
        I2C0Read(ICM20948_ADDRESS, ICM20948_REG_GYRO_ZOUT_L_SH, &LByte);
        GYRO_Data3 = (LByte <<8 | HByte);
        Comp_Filter();
        SysCtlDelay(15000000);
    }

}
```

2. Block diagram and/or Schematics showing the components, pins used, and interface.

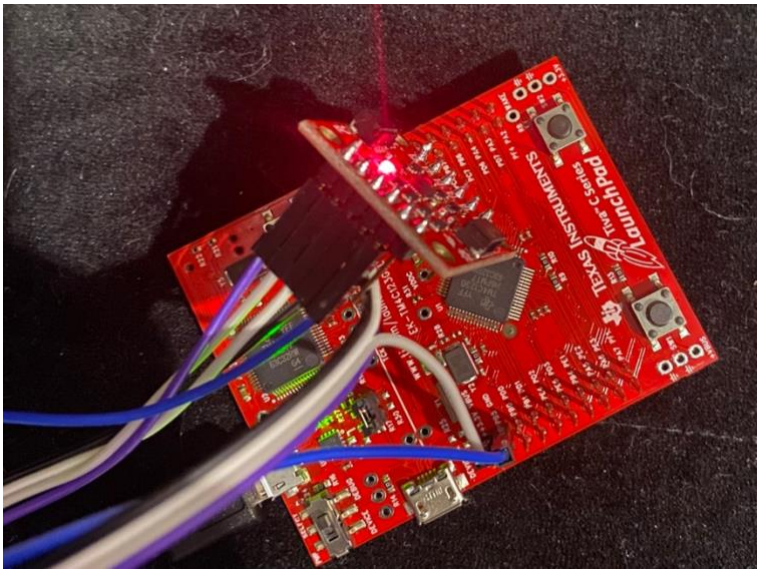
-ICM20948

-Tivac

-CCS

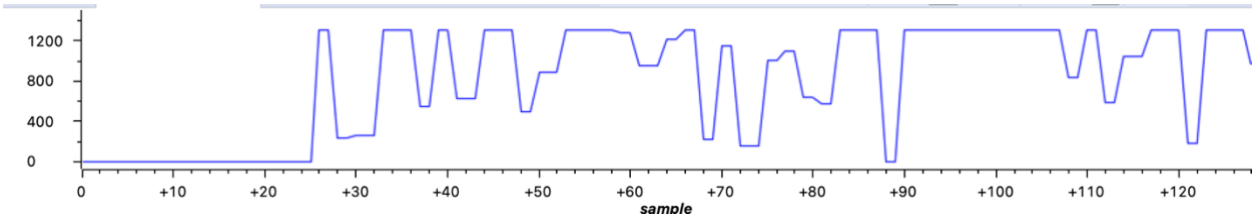


3. Screenshots of the IDE, physical setup, debugging process - Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.

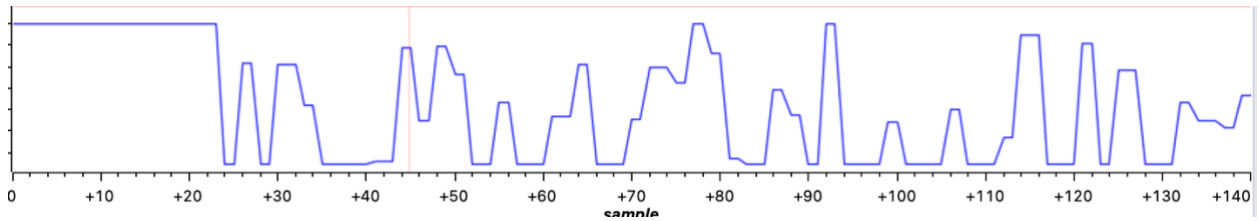



```
Pitch  : 1309  | Roll  : -1309 | Yaw  : 1309
Pitch  : 1309  | Roll  : -1309 | Yaw  : 961
Pitch  : 1309  | Roll  : -431 | Yaw  : 1309
Pitch  : 827  | Roll  : -223 | Yaw  : 1309
Pitch  : 521  | Roll  : -1259 | Yaw  : 757
Pitch  : 1309  | Roll  : -1309 | Yaw  : 1309
Pitch  : 1309  | Roll  : -727 | Yaw  : 1309
Pitch  : 1309  | Roll  : -1309 | Yaw  : 328
Pitch  : 1309  | Roll  : -1309 | Yaw  : 1309
Pitch  : 1053  | Roll  : -746 | Yaw  : 512
Pitch  : 1309  | Roll  : -1105 | Yaw  : 1309
```

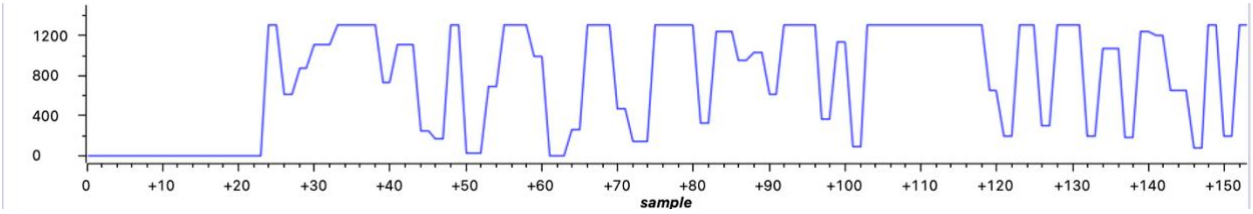
Pitch



Roll



Yaw



4. Declaration

I understand the Student Academic Misconduct Policy - http://studentconduct.unlv.edu/misconduct/policy.html

"This assignment submission is my own, original work".

-Elmer Mejia