CPE 403 ADV EMB SYS DES FALL 2019

TITLE: Midterm 1

AUTHOR: Steven Lee #5003887117

GOAL:

- Task 1:
  - Interface the given MPU6050 IMU using I2C protocol to TivaC.
  - Print all accelerometer and gyro values on to the serial terminal.
- Task 2:
  - Interface the given MPU6050 IMU using I2C protocol to TivaC.
  - Plot all accelerometer and gyro values on to a Graph
- Task 3:
  - Implement a complementary filter to filter the raw accelerometer and gyro values.
  - Print all raw and filtered accelerometer and gyro values on to the serial terminal.
  - Implement the filter using IQMath Library.
- Task 4:
  - Implement a complementary filter to filter the raw accelerometer and gyro values.
  - Plot all raw and filtered accelerometer and gyro values on to a Graph

DELIVERABLES:

For the midterm, I interfaced the MPU6050 with I2C to communicate with TM4C123GH6PM. To display the gyroscope and accelerometer values, they were sampled and printed through UART.

COMPONENTS:

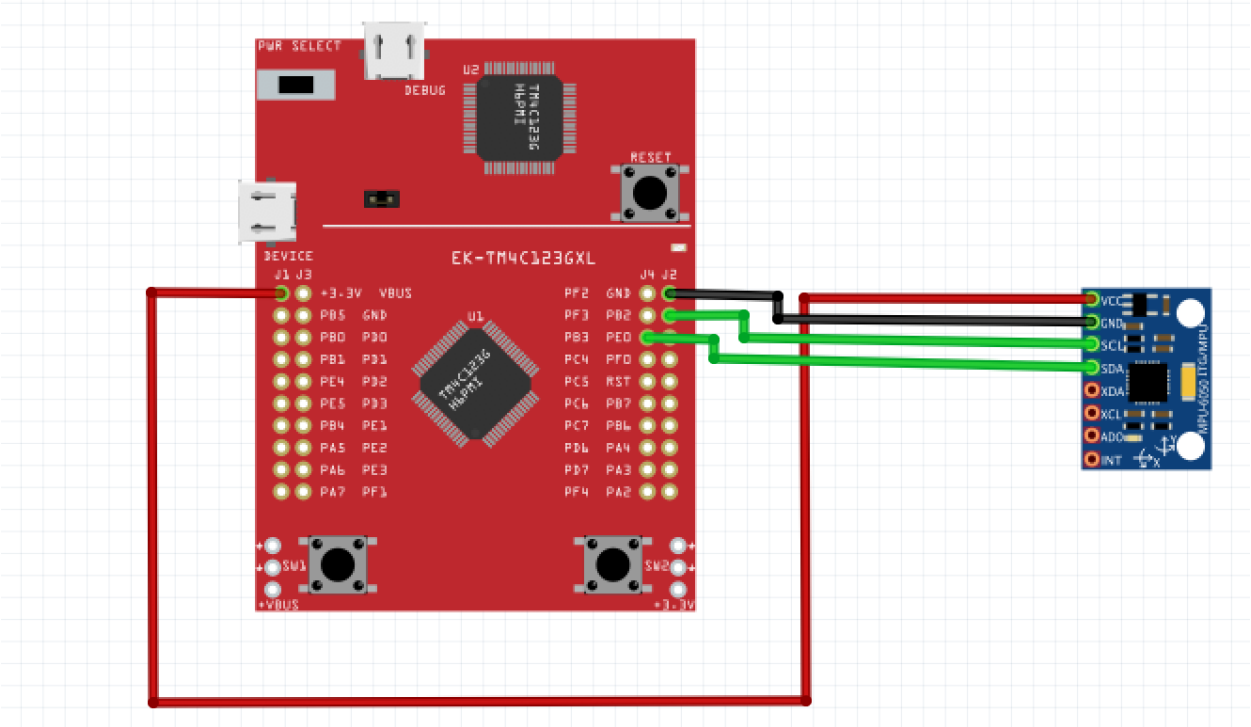
- I2C
  - Only two bus lines required (Serial data line SDA and Serial clock line SCL)
  - No Strict baud rate requirements.
  - Master/Slave relationships exist between all components.
  - One data bit transferred each clock pulse. The data on SDA must remain stable during high period of clock pulse.
  - Both SDA and SCL remain high when bus is not busy. High to Low indicate start condition while Low to High indicates stop condition

- UART
  - Clock generator.
  - Input and output shift registers.
  - Transmit and receive control.
  - Read and write logic.

- IQMath Library
    - Simplified porting from the floating-point simulations to a fixed point MCU
    - Math library for fixed-point processors speeds computation of floating-point values
    - Conversions from floating to fixed point are completed at the compile stage, so CPU cycles are not used.

- MPU6050
    - 3-axis Accelerometer and Gyroscope
    - Uses I2C protocol

SCHEMATICS:

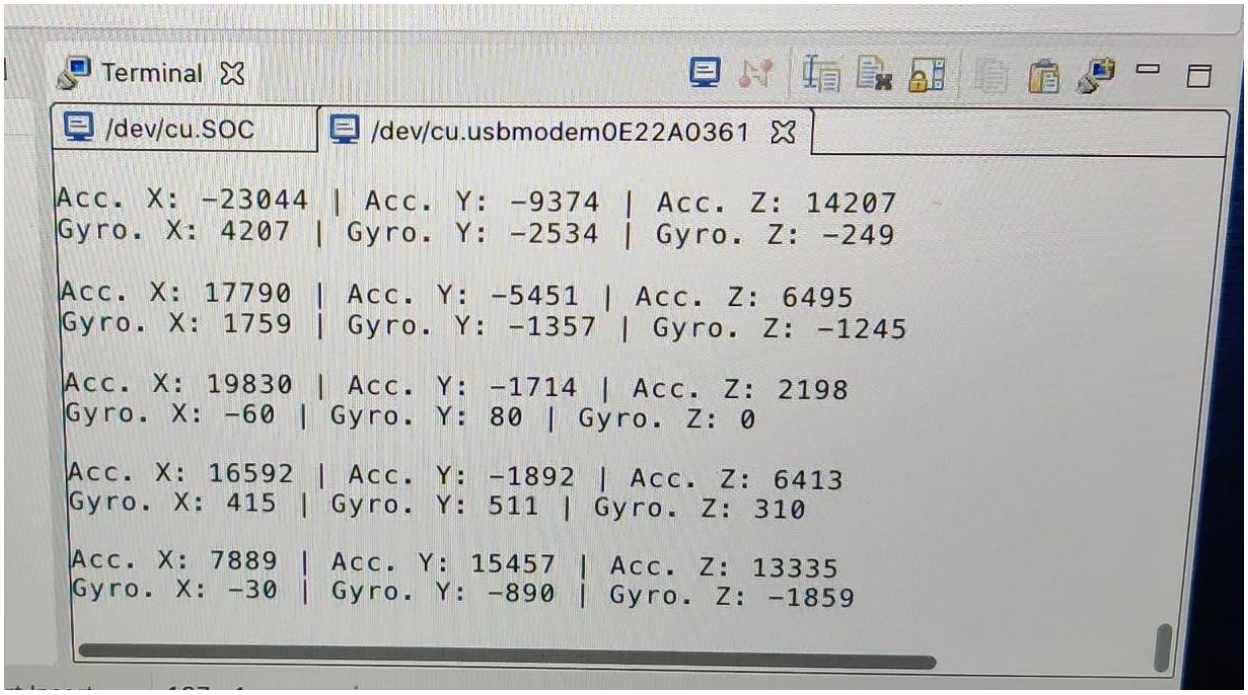


IMPLEMENTATION:

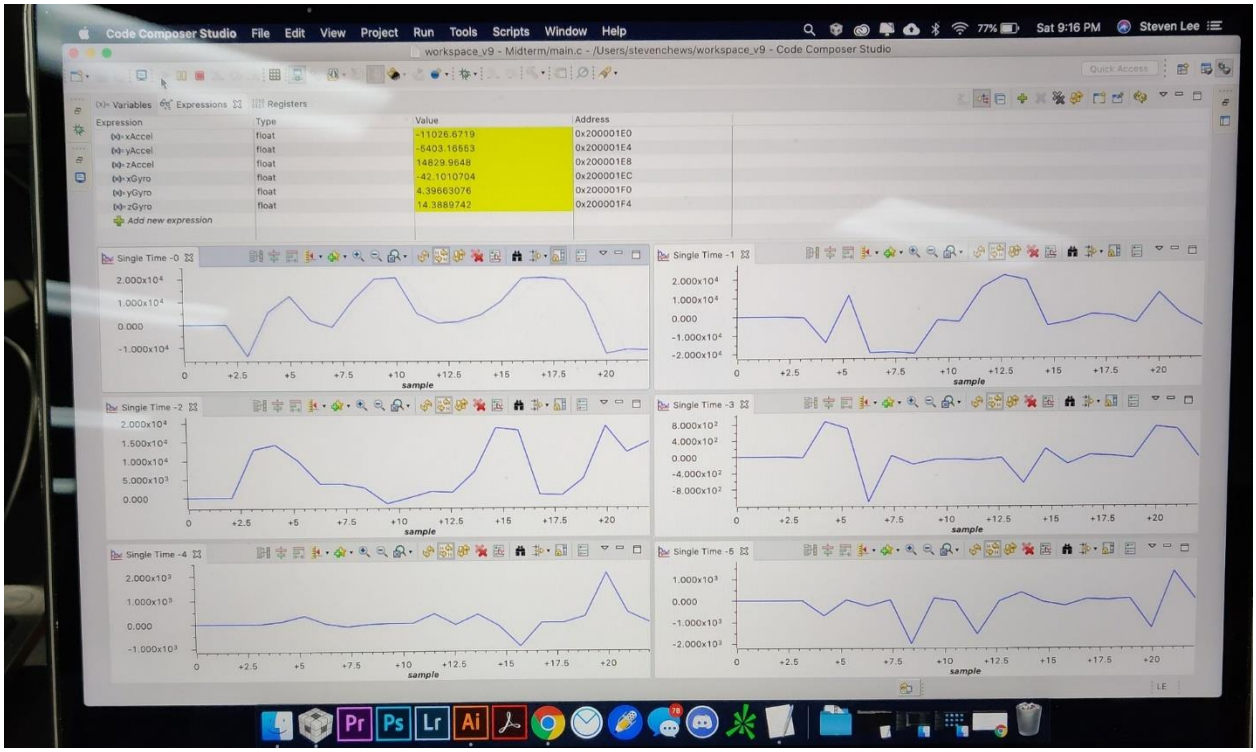
1. Initialize I2C
2. Initialize UART
3. Initialize MPU6050
4. Interface MPU6050 and TM4C123GH6PM with I2C
5. Read from MPU6050
6. Filter readings from gyroscope and accelerometer
7. Send serial data via UART

SCREENSHOTS:

Task 1:



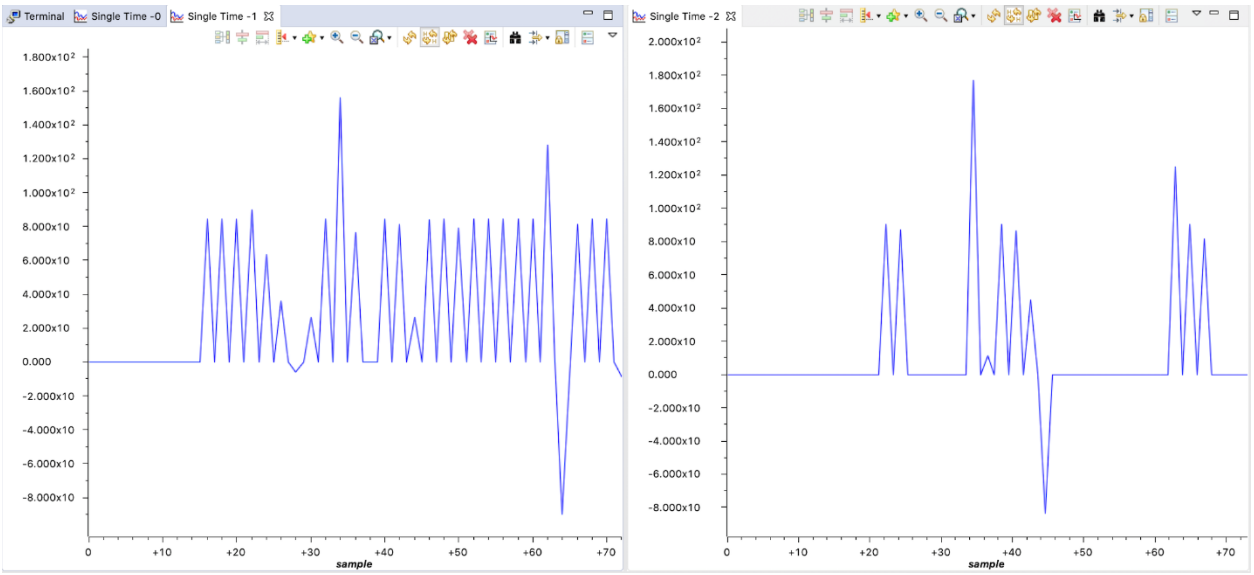
Task 2:



Task 3:

```
Acc. X: -110 | Acc. Y: 19916 | Acc. Z: 435
Gyro. X: -28 | Gyro. Y: -20 | Gyro. Z: -18
Pitch: 0Roll:  90
Acc. X: 10638 | Acc. Y: 17478 | Acc. Z: 1796
Gyro. X: -84 | Gyro. Y: -29 | Gyro. Z: 241
Pitch: 84Roll:  86
Acc. X: 19821 | Acc. Y: 3578 | Acc. Z: 3736
Gyro. X: 434 | Gyro. Y: 129 | Gyro. Z: 27
Pitch: 81Roll:  45
Acc. X: 1528 | Acc. Y: -18810 | Acc. Z: 2787
Gyro. X: -138 | Gyro. Y: 0 | Gyro. Z: 11
Pitch: 26Roll:  -83
Acc. X: 19969 | Acc. Y: -325 | Acc. Z: 2993
Gyro. X: 360 | Gyro. Y: 387 | Gyro. Z: -519
Pitch: 83Roll:  0
Acc. X: 20185 | Acc. Y: 574 | Acc. Z: 2778
Gyro. X: -27 | Gyro. Y: -19 | Gyro. Z: -21
Pitch: 84Roll:  0
Acc. X: 21938 | Acc. Y: -292 | Acc. Z: 4584
Gyro. X: -35 | Gyro. Y: 0 | Gyro. Z: -35
Pitch: 79Roll:  0
```

Task 4:



CODE:

```
#include <stdbool.h>
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
```

```c
#include <stdbool.h>

#include "sensorlib/i2cm_drv.h"
#include "sensorlib/hw_mpu6050.h"
#include "sensorlib/mpu6050.h"

#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_sysctl.h"
#include "inc/hw_types.h"
#include "inc/hw_i2c.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"

#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom.h"
#include "driverlib/rom_map.h"
#include "driverlib/debug.h"
#include "driverlib/interrupt.h"
#include "driverlib/i2c.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"

#include "uartstdio.h"
#include "IQmathLib.h"
#include "math.h"

#define ACCELEROMETER_SENSITIVITY 8192.0
#define GYROSCOPE_SENSITIVITY 65.536
#define SAMPLE_RATE 0.01
#define dt 0.01

volatile bool g_bMPU6050Done;
tI2CMInstance g_sI2CMSimpleInst;

void ComplementaryFilter(short [3], short [3], float *, float *);
void ConfigureUART(void);
void MPU6050Callback(void *, uint_fast8_t);
void InitI2C0(void);
void I2CMSimpleIntHandler(void);
void DelayInMs(int);

int main(void) {
    float pitch;
    float roll;
    float rawPitch;
    float rawRoll;

    float xA = 0;
    float yA = 0;
    float zA = 0;
    float xG = 0;
    float yG = 0;
    float zG = 0;

    float fAccel[3];
    float fGyro[3];
```

```
short fAccelShort[3];
short fGyroShort[3];

tMPU6050 sMPU6050;

SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
InitI2C0();
ConfigureUART();

g_bMPU6050Done = false;
MPU6050Init(&sMPU6050, &g_sI2CMSimpleInst, 0x68, MPU6050Callback, &sMPU6050);
while (!g_bMPU6050Done);

g_bMPU6050Done = false;
   MPU6050ReadModifyWrite(&sMPU6050,MPU6050_O_ACCEL_CONFIG,0xFF,MPU6050_ACCEL_CONFIG_
   AFS_SEL_4G,MPU6050Callback,&sMPU6050);
while (!g_bMPU6050Done);

g_bMPU6050Done = false;
   MPU6050ReadModifyWrite(&sMPU6050,MPU6050_O_GYRO_CONFIG,0xFF,MPU6050_GYRO_CONFIG_FS
   _SEL_250,MPU6050Callback,&sMPU6050);
while (!g_bMPU6050Done);

g_bMPU6050Done = false;
   MPU6050ReadModifyWrite(&sMPU6050,MPU6050_O_PWR_MGMT_1,
   0x00,0x00,MPU6050Callback,&sMPU6050);
while (!g_bMPU6050Done);

g_bMPU6050Done = false;
   MPU6050ReadModifyWrite(&sMPU6050,MPU6050_O_PWR_MGMT_2,0x00,0x00,MPU6050Callback,&s
   MPU6050);
while (!g_bMPU6050Done);

while (1)
{
    g_bMPU6050Done = false;
    MPU6050DataRead(&sMPU6050, MPU6050Callback, &sMPU6050);
    while (!g_bMPU6050Done) {        }

    MPU6050DataAccelGetFloat(&sMPU6050, &fAccel[0], &fAccel[1], &fAccel[2]);
    MPU6050DataGyroGetFloat(&sMPU6050, &fGyro[0], &fGyro[1], &fGyro[2]);

    //Set raw values to be printed
    xA = fAccel[0];
    yA = fAccel[1];
    zA = fAccel[2];
    xG = fGyro[0];
    yG = fGyro[1];
    zG = fGyro[2];

    //S fAccel and fGyro to short
    fAccelShort[0] = (short)fAccel[0];
    fAccelShort[1] = (short)fAccel[1];
    fAccelShort[2] = (short)fAccel[2];
    fGyroShort[0] = (short)fGyro[0];
    fGyroShort[1] = (short)fGyro[1];
    fGyroShort[2] = (short)fGyro[2];
```

```
        ComplementaryFilter(&fAccelShort[0], &fGyroShort[0], &pitch, &roll);

        // Raw values
        rawPitch = atan2f((float)fAccelShort[0], (float)fAccelShort[2]) * 180 / M_PI;
        rawRoll = atan2f((float)fAccelShort[1], (float)fAccelShort[2]) * 180 / M_PI;

        UARTprintf("Accel X: %d | Accel Y: %d | Accel Z: %d\n", (int)xA*1000,
       (int)yA*1000, (int)zA*1000);
        UARTprintf("Gyro X: %d | Gyro Y: %d | Gyro Z: %d\n", (int)xG*1000, (int)yG*1000,
       (int)zG*1000);
        UARTprintf("Raw Pitch: %d | Raw Roll: %d\n", (int)rawPitch, (int)rawRoll);

        // Set filtered values to be printed
        xA = (float)fAccelShort[0];
        yA = (float)fAccelShort[1];
        zA = (float)fAccelShort[2];
        xG = (float)fGyroShort[0];
        yG = (float)fGyroShort[1];
        zG = (float)fGyroShort[2];

        UARTprintf("Filtered Accel X: %d | Filtered Accel Y: %d | Filtered Accel Z:
        %d\n", (int)xA*1000, (int)yA*1000, (int)zA*1000);
        UARTprintf("Filtered Gyro X: %d | Filtered Gyro Y: %d | Filtered Gyro Z: %d\n",
        (int)xG*1000, (int)yG*1000, (int)zG*1000);
        UARTprintf("Filtered Pitch: %d | Filtered Roll: %d\n", (int)pitch, (int)roll);

        DelayInMs(1000);
    }
}

void ComplementaryFilter(short accData[3], short gyrData[3], float *pitch, float *roll) {
    float pitchAcc, rollAcc;

    *pitch += ((float)gyrData[0] / GYROSCOPE_SENSITIVITY) * dt;
    *roll += ((float)gyrData[1] / GYROSCOPE_SENSITIVITY) * dt;

    int forceMagnitudeApprox = abs(accData[0]) + abs(accData[1]) + abs(accData[2]);
    if (forceMagnitudeApprox > 8192 && forceMagnitudeApprox < 32768) {
        pitchAcc = atan2f((float)accData[1], (float)accData[2]) * 180 / M_PI;
        *pitch = *pitch * 0.98 + pitchAcc * 0.02;

        rollAcc = atan2f((float)accData[0], (float)accData[2]) * 180 / M_PI;
        *roll = *roll * 0.98 + rollAcc * 0.02;
    }
}

void ConfigureUART(void) {
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    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);
}

void MPU6050Callback(void *pvCallbackData, uint_fast8_t ui8Status) {
    if (ui8Status != I2CM_STATUS_SUCCESS) {}
```

```
    g_bMPU6050Done = true;
}

void InitI2C0(void) {
    SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);
    SysCtlPeripheralReset(SYSCTL_PERIPH_I2C0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);

    GPIOPinConfigure(GPIO_PB2_I2C0SCL);
    GPIOPinConfigure(GPIO_PB3_I2C0SDA);
    GPIOPinTypeI2CSCL(GPIO_PORTB_BASE, GPIO_PIN_2);
    GPIOPinTypeI2C(GPIO_PORTB_BASE, GPIO_PIN_3);

    I2CMasterInitExpClk(I2C0_BASE, SysCtlClockGet(), true);
    HWREG(I2C0_BASE + I2C_O_FIFOCTL) = 80008000;
    I2CMInit(&g_sI2CMSimpleInst, I2C0_BASE, INT_I2C0, 0xff, 0xff, SysCtlClockGet());
}

void I2CMSimpleIntHandler(void) {
    I2CMIntHandler(&g_sI2CMSimpleInst);
}

void DelayInMs(int ms) {
    SysCtlDelay((SysCtlClockGet() / (3 * 1000)) * ms) ;
}
```