TITLE: TivaC Midterm Project

GOAL:

- Interface with the MPU6050 IMU using I2C protocol to TivaC. Print all accelerometer and gyro values on to the serial terminal.
- Interface with the MPU6050 IMU using I2C protocol to TivaC. Plot all accelerometer and gyro values on to the CCS Graph Tool.
- 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.
- Implement a complementary filter to filter the raw accelerometer and gyro values. Plot all raw and filtered accelerometer and gyro values on to the CCS Graph Tool.

DELIVERABLES:

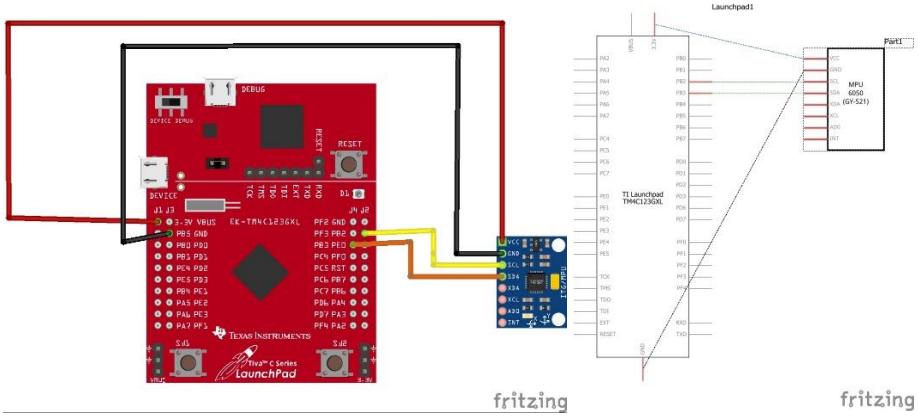
Was able to read and print raw/filter values of the MPU6050 IMU using I2C protocol.

COMPONENTS:

- TivaC TM4C123GXL
- MPU6050 IMU

Explain the main characteristics, interface, and limitation of the components used in the design, including the registered used and what was initialized? Why?

SCHEMATICS:



IIMPLEMENTATION:

Step implemented in the code - for example initialization of I2C, UART, start reading one set of data, print - explain each subroutine.

Before entering the main function, the libraries that are needed are for this project are the driverlib, sensorlib, and the IQmath. In the program we configure the UART, I2C, and the MPU6050. ConfiUART() configures two pins (PB 2 and PB3) that will be used to communicate. InitI2C0() Enables I2C module 0 and configures the pins PB2 and PB3 to be SCL and SDA respectively. In the main the MPU6050 is initialized in order to communicate with our microcontroller (TivaC TM4C123GXL). The MPU6050 is also configured (after initialization) using the sensor libraries functions such as MPU6050ReadModifyWrite() which allows us to rewrite the IMU(Inertial Measurement Unit) configuration settings, to gather data we use MPU6050DataRead(), in order to get the raw values of the accelerometer and gyro we use MPU6050DataAccelGetFloat() and MPU6050DataGyroGetFloat(). We are able to output the values from the MPU650 to the terminal using UARTprintf().

Name: Ricky Perez

CODE:

Task 1-2

```
// Ricky Perez
// CpE 403
// Midterm Project
// Task 1-2
#include <stdbool.h>
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#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 "utils/uartstdio.h"
#include "driverlib/uart.h"
#include <math.h>

volatile bool g_bMPU6050Done; // A boolean that is set when a MPU6050 command has
completed.

// I2C master instance
tI2CMInstance g_sI2CMSimpleInst;

// Modify startup file to initialize interrupt
void InitI2C0(void)
{

    SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);     // enable I2C module 0
    SysCtlPeripheralReset(SYSCTL_PERIPH_I2C0);      // reset module
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);    // enable GPIO peripheral that
contains I2C 0

    // Configure the pin muxing for I2C0 functions on port B2 and B3.
    GPIOPinConfigure(GPIO_PB2_I2C0SCL); // SCL on Port PB2
    GPIOPinConfigure(GPIO_PB3_I2C0SDA); // SDA on Port PB3

    // Select the I2C function for these pins.
    GPIOPinTypeI2CSCL(GPIO_PORTB_BASE, GPIO_PIN_2);
```

```
    GPIOPinTypeI2C(GPIO_PORTB_BASE, GPIO_PIN_3);

    // Enable and initialize the I2C0 master module.  Use the system clock for the
I2C0 module.
    I2CMasterInitExpClk(I2C0_BASE, SysCtlClockGet(), true); // Setting the last
parameter to true sets the I2C data trasfer rate to 400kbps


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


    I2CMInit(&g_sI2CMSimpleInst, I2C0_BASE, INT_I2C0, 0xff, 0xff, SysCtlClockGet());
// Initialize the I2C master driver.

}

void ConfigUART(void)
{
    // The following UART signals are configured only for displaying console
messages.
    // - UART0 peripheral
    // - GPIO Port A peripheral (for UART0 pins)
    // - UART0RX - PA0
    // - UART0TX - PA

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);  // Enable GPIO port A which is used
for UART0 pins.
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);// Enable UART0 so that we can
configure the clock.
    GPIOPinConfigure(GPIO_PA0_U0RX);  // Configure the pin muxing for UART0 functions
on port A0
    GPIOPinConfigure(GPIO_PA1_U0TX);  // Configure the pin muxing for UART0 functions
on port A1.
    UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);   // Use the internal 16MHz
oscillator as the UART clock source
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1); // Select the
alternate (UART) function for these pins.
    UARTStdioConfig(0, 115200, 16000000);  // Initialize the UART for console I/O.
}

// Check success status of MPU6050
void MPU6050Callback(void *pvCallbackData, uint_fast8_t ui8Status)
{
    //
    // See if an error occurred.
    //
    if (ui8Status != I2CM_STATUS_SUCCESS)
    {
            //
    }
    g_bMPU6050Done = true; // MPU6050 transaction has completed
}

// I2C interrupt handler
void I2CMSimpleIntHandler(void)
```

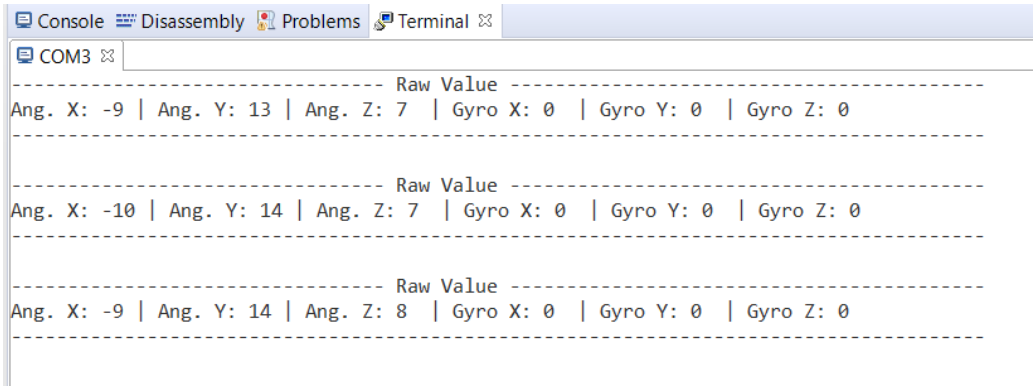

Name: Ricky Perez

```
{
    I2CMIntHandler(&g_sI2CMSimpleInst);  // Call the I2C master driver interrupt
handler.
}


int main()
{
    SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_PLL | SYSCTL_OSC_INT |
SYSCTL_XTAL_16MHZ);
    tMPU6050 sMPU6050;
    float fAccel[3], fGyro[3];

    InitI2C0();
    ConfigUART();

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

    // Configure the MPU6050 (MPU6050_ACCEL_CONFIG_AFS_SEL_4G) for +/- 4 g
accelerometer range.
    g_bMPU6050Done = false;
    MPU6050ReadModifyWrite(&sMPU6050, MPU6050_O_ACCEL_CONFIG,
~MPU6050_ACCEL_CONFIG_AFS_SEL_M,
                           MPU6050_ACCEL_CONFIG_AFS_SEL_4G, MPU6050Callback,
&sMPU6050);
    while (!g_bMPU6050Done)
    {
    }


    g_bMPU6050Done = false;
    MPU6050ReadModifyWrite(&sMPU6050, MPU6050_O_PWR_MGMT_1, 0x00, 0b00000010 &
MPU6050_PWR_MGMT_1_DEVICE_RESET, MPU6050Callback, &sMPU6050);
    while (!g_bMPU6050Done)
    {
    }

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

    while (1)
    {
        // Request another reading from the MPU6050.
        g_bMPU6050Done = false;
        MPU6050DataRead(&sMPU6050, MPU6050Callback, &sMPU6050);
        while (!g_bMPU6050Done)
        {
```


Name: Ricky Perez

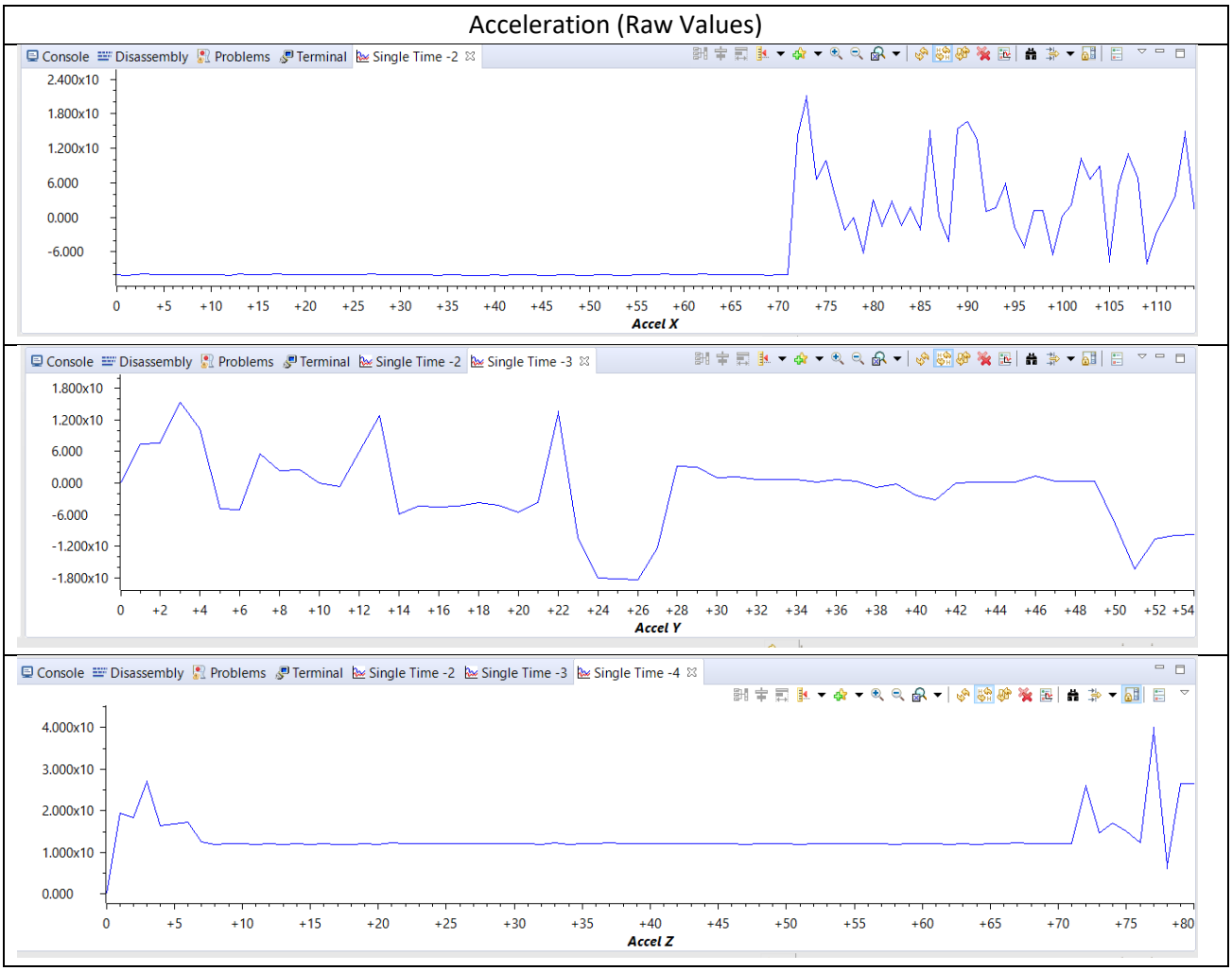
```
        }
        // Get the new accelerometer and gyroscope readings.
        MPU6050DataAccelGetFloat(&sMPU6050, &fAccel[0], &fAccel[1], &fAccel[2]);
        MPU6050DataGyroGetFloat(&sMPU6050, &fGyro[0], &fGyro[1], &fGyro[2]);

        UARTprintf("-------------------------------- Raw Value --------------------
--------------------\n");
        UARTprintf("Ang. X: %d | Ang. Y: %d | Ang. Z: %d  | Gyro X: %d  | Gyro Y: %d
| Gyro Z: %d  \n",(int)fAccel[0], (int)fAccel[1], (int)fAccel[2],(int)fGyro[0],
(int)fGyro[1], (int)fGyro[2]);
        UARTprintf("------------------------------------------------------------------
--------------------\n\n");
        SysCtlDelay( (SysCtlClockGet()/(3*1000))*1000 ) ;
    }


    return(0);
}
```

Screen Shot of terminal for Task 1

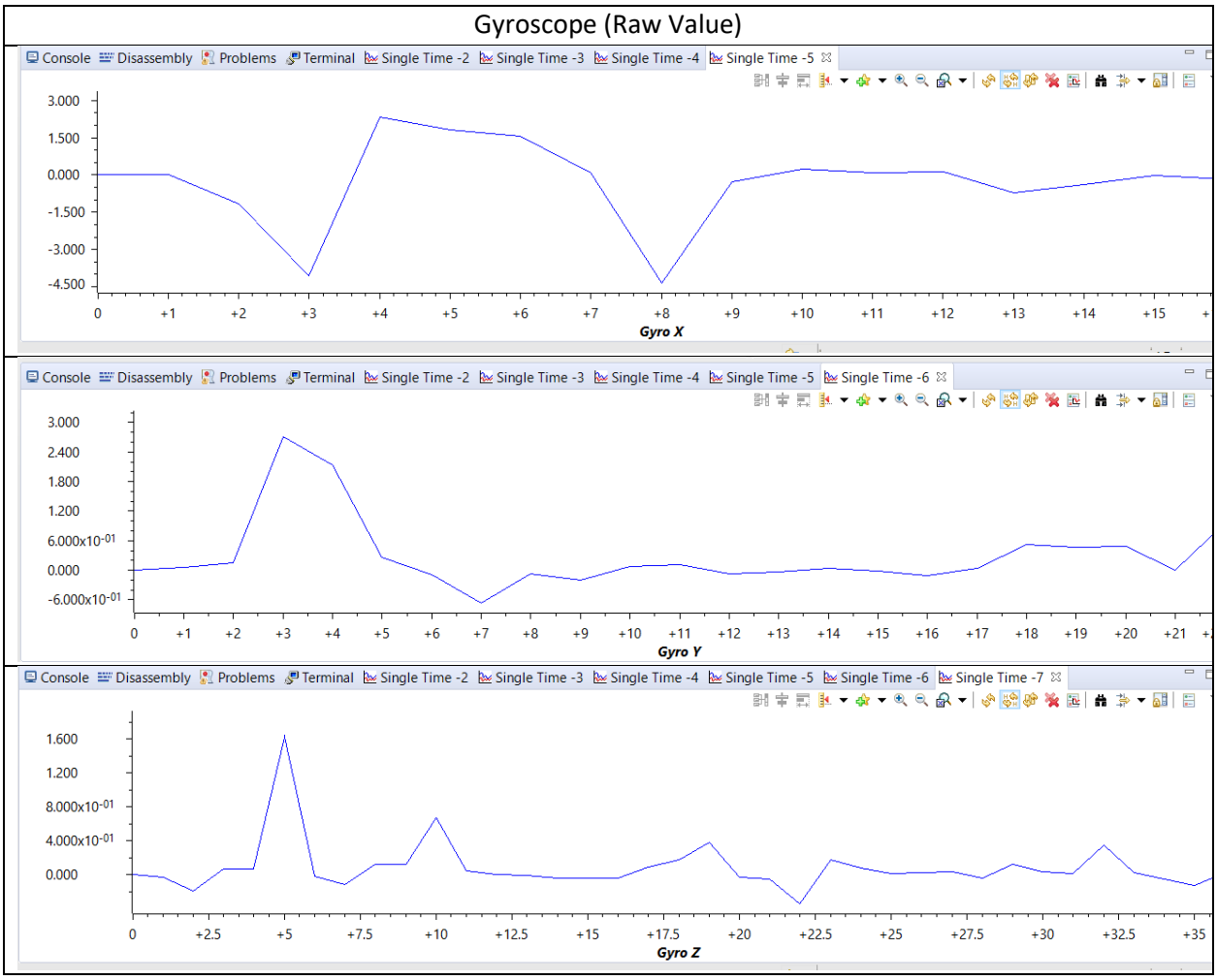


Name: Ricky Perez

Screen Shot of terminal for Task 2



Name: Ricky Perez



Name: Ricky Perez

Task 3-4

```
// Ricky Perez
// CpE 403
// Midterm Project
// Task 1-2
#include <stdbool.h>
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#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 "utils/uartstdio.h"
#include "driverlib/uart.h"
#include <math.h>
#include "IQmath/IQmathLib.h"
// Defined Variables

#define ACCELEROMETER_SENSITIVITY 8192.0
#define GYROSCOPE_SENSITIVITY 65.536
#define DT 0.01
#define RT (180/3.14159265359)
//
void Comp_Fil(float fAccel[], float fGyro[], _iq16 Pitch, _iq16 Roll);
//
// A boolean that is set when a MPU6050 command has completed.
//
volatile bool g_bMPU6050Done;

//
// I2C master instance
//
tI2CMInstance g_sI2CMSimpleInst;
```


Name: Ricky Perez

```
// IQmath global variables
_iq16 Pitch = 0;
_iq16 Roll = 0;

// Modify startup file to initialize interrupt
void InitI2C0(void)
{

    SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);     // enable I2C module 0
    SysCtlPeripheralReset(SYSCTL_PERIPH_I2C0);      // reset module
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);    // enable GPIO peripheral that
contains I2C 0

    // Configure the pin muxing for I2C0 functions on port B2 and B3.
    GPIOPinConfigure(GPIO_PB2_I2C0SCL); // SCL on Port PB2
    GPIOPinConfigure(GPIO_PB3_I2C0SDA); // SDA on Port PB3

    // Select the I2C function for these pins.
    GPIOPinTypeI2CSCL(GPIO_PORTB_BASE, GPIO_PIN_2);
    GPIOPinTypeI2C(GPIO_PORTB_BASE, GPIO_PIN_3);

    // Enable and initialize the I2C0 master module.  Use the system clock for the
I2C0 module.
    I2CMasterInitExpClk(I2C0_BASE, SysCtlClockGet(), true); // Setting the last
parameter to true sets the I2C data trasfer rate to 400kbps


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


    I2CMInit(&g_sI2CMSimpleInst, I2C0_BASE, INT_I2C0, 0xff, 0xff, SysCtlClockGet());
// Initialize the I2C master driver.

}

void ConfigUART(void)
{
    // The following UART signals are configured only for displaying console
messages.
    // - UART0 peripheral
    // - GPIO Port A peripheral (for UART0 pins)
    // - UART0RX - PA0
    // - UART0TX - PA

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);  // Enable GPIO port A which is used
for UART0 pins.
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);// Enable UART0 so that we can
configure the clock.
    GPIOPinConfigure(GPIO_PA0_U0RX);  // Configure the pin muxing for UART0 functions
on port A0
    GPIOPinConfigure(GPIO_PA1_U0TX);  // Configure the pin muxing for UART0 functions
on port A1.
    UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);   // Use the internal 16MHz
oscillator as the UART clock source
```

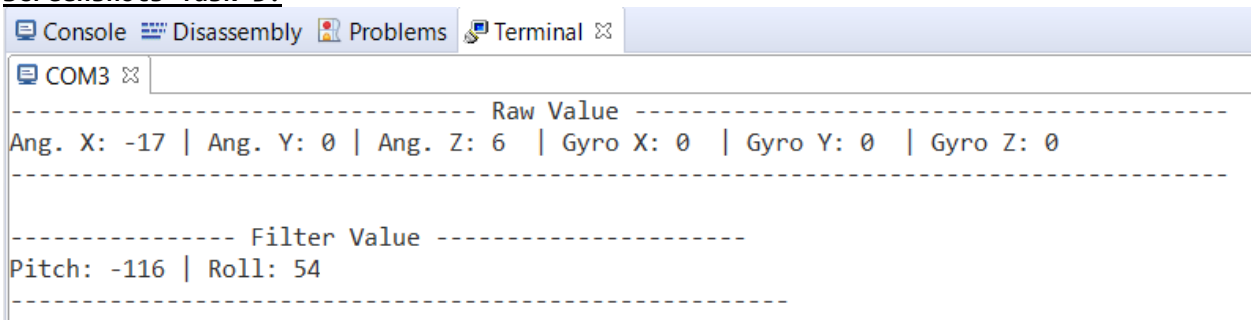
```
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1); // Select the
alternate (UART) function for these pins.
    UARTStdioConfig(0, 115200, 16000000);  // Initialize the UART for console I/O.
}

// Check success status of MPU6050
void MPU6050Callback(void *pvCallbackData, uint_fast8_t ui8Status)
{
    // See if an error occurred.
    if (ui8Status != I2CM_STATUS_SUCCESS)
    {
            //
    }
    // MPU6050 transaction has completed.
    g_bMPU6050Done = true;
}

// I2C interrupt handler
void I2CMSimpleIntHandler(void)
{
    // Call the I2C master driver interrupt handler.
    I2CMIntHandler(&g_sI2CMSimpleInst);
}


int main()
{
    SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_PLL | SYSCTL_OSC_INT |
SYSCTL_XTAL_16MHZ);
    tMPU6050 sMPU6050;
    float fAccel[3], fGyro[3];
    _iq16 GyroVal[3], Acc[3];
    _iq16 ForceMagApprx, g_sensitivity;
    _iq16 Accel_Pitch, Accel_Roll;


    InitI2C0();
    ConfigUART();

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

    //
    // Configure the MPU6050 for +/- 4 g accelerometer range.
    //
    g_bMPU6050Done = false;
    MPU6050ReadModifyWrite(&sMPU6050, MPU6050_O_ACCEL_CONFIG,
~MPU6050_ACCEL_CONFIG_AFS_SEL_M,
                           MPU6050_ACCEL_CONFIG_AFS_SEL_4G, MPU6050Callback,
&sMPU6050);
    while (!g_bMPU6050Done)
    {
```

```
    }


    g_bMPU6050Done = false;
    MPU6050ReadModifyWrite(&sMPU6050, MPU6050_O_PWR_MGMT_1, 0x00, 0b00000010 &
MPU6050_PWR_MGMT_1_DEVICE_RESET, MPU6050Callback, &sMPU6050);
    while (!g_bMPU6050Done)
    {
    }

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

    while (1)
    {
        // Request another reading from the MPU6050.
        g_bMPU6050Done = false;
        MPU6050DataRead(&sMPU6050, MPU6050Callback, &sMPU6050);
        while (!g_bMPU6050Done)
        {
        }
        // Get the new accelerometer and gyroscope readings.
        MPU6050DataAccelGetFloat(&sMPU6050, &fAccel[0], &fAccel[1], &fAccel[2]);
        MPU6050DataGyroGetFloat(&sMPU6050, &fGyro[0], &fGyro[1], &fGyro[2]);

        // Comp Filter

        GyroVal[0] = _IQ16(fGyro[0]);
        GyroVal[1] = _IQ16(fGyro[1]);
        GyroVal[2] = _IQ16(fGyro[2]);
        Acc[0] = _IQ16(fAccel[0]);
        Acc[1] = _IQ16(fAccel[1]);
        Acc[2] = _IQ16(fAccel[2]);
        g_sensitivity = _IQ16(GYROSCOPE_SENSITIVITY);

        Pitch += _IQ16mpy(_IQ16div(GyroVal[0],g_sensitivity), _IQ16(DT));
        Roll -= _IQ16mpy(_IQ16div(GyroVal[1],g_sensitivity), _IQ16(DT));


        ForceMagApprx = _IQ16abs(Acc[0]) + _IQ16abs(Acc[1]) + _IQ16abs(Acc[2]);

        if(ForceMagApprx > 8192 && ForceMagApprx < 32768)
        {

            Accel_Pitch = _IQ16mpy(_IQ16atan2(Acc[1],Acc[2]), _IQ16(RT));
            Pitch = _IQ16mpy(Pitch,_IQ16(0.98)) + _IQ16mpy(Accel_Pitch,_IQ16(0.02));
            Accel_Roll = _IQ16mpy(_IQ16atan2(Acc[0],Acc[2]), _IQ16(RT));
            Roll = _IQ16mpy(Roll,_IQ16(0.98)) + _IQ16mpy(Accel_Roll,_IQ16(0.02));
        }
        UARTprintf("-------------------------------- Raw Value --------------------
--------------------\n");
```


Name: Ricky Perez

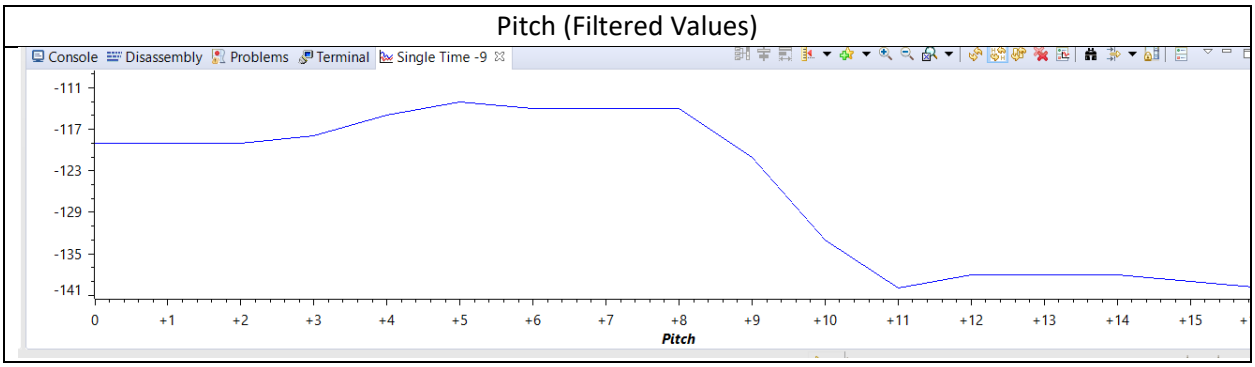
```
        UARTprintf("Ang. X: %d | Ang. Y: %d | Ang. Z: %d  | Gyro X: %d  | Gyro Y: %d
| Gyro Z: %d  \n",(int)fAccel[0], (int)fAccel[1], (int)fAccel[2],(int)fGyro[0],
(int)fGyro[1], (int)fGyro[2]);
        UARTprintf("------------------------------------------------------------------
--------------------\n\n");

        UARTprintf("---------------- Filter Value ----------------------\n");


        UARTprintf("Pitch: %d | Roll: %d\n",Pitch, Roll);
        UARTprintf("-----------------------------------------------------\n");
        SysCtlDelay( (SysCtlClockGet()/(3*1000))*1000 ) ;
    }


    return(0);
}
```

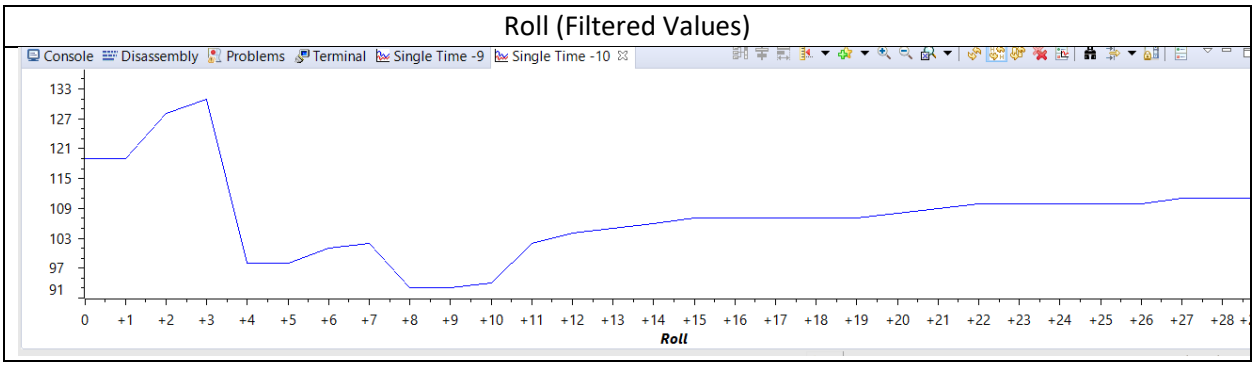
Screenshots Task 3:



Name: Ricky Perez

Screenshots Task 4:

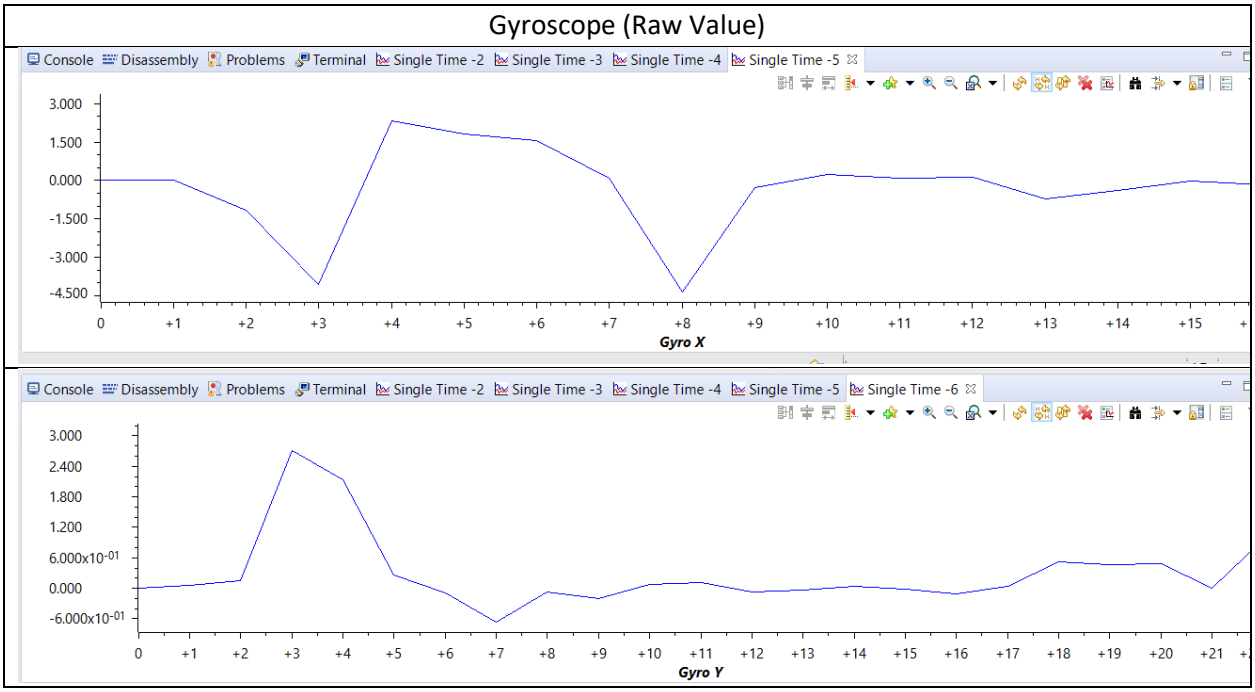




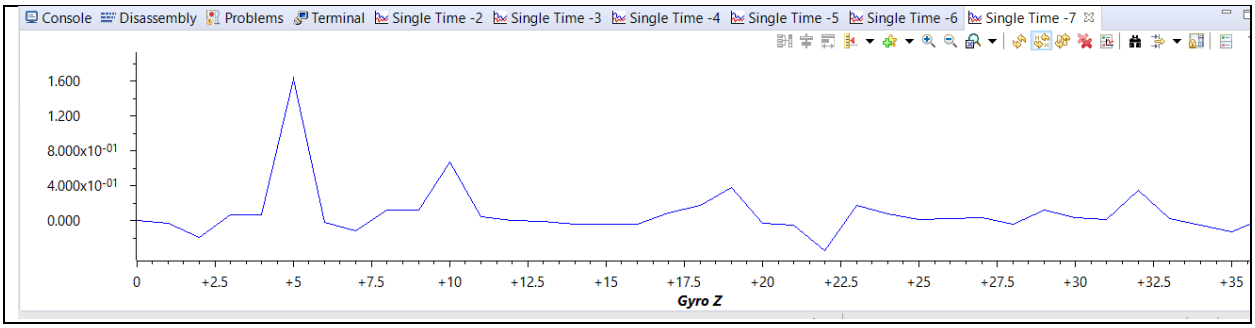
Name: Ricky Perez

Acceleration (Raw Values)

Gyroscope (Raw Value)



Name: Ricky Perez



YouTube Links:

Task 1 : https://youtu.be/WxbwTU2hgiI

Task 2 : https://youtu.be/_X8lBecQUEs

Task 3 : https://youtu.be/QpFtXIjsaEI

Task 4 : https://youtu.be/XVQFo0Mk2wM

Name: Ricky Perez