CPE 403 ADV EMB SYS DES F 2019

TITLE: Midterm

GOAL: The goal of this project is to familiarize oneself with using TI IDE tools to interface with an MPU6050 using a TIva C Device. We are to utilize the I2C communication protocol to create a master slave connection with the Tiva C and the MPU6050.

Make bullet points of the project goal(s).

* Initialize UART connection
* Initialize MPU and configure with 4g accelerometer
* Plot accelerometer and gyroscope values
* Implement Complementary filter using IQmath.
* Plot pitch and yaw values after filtering complete.

**DELIVERABLES**:

What is the intended project deliverables? What was completed?

The project deliverables are as follows

1. Interface the given MPU6050 IMU using I2C protocol to TivaC. Print all accelerometer and gyro values on to the serial terminal.
2. Interface the given MPU6050 IMU using I2C protocol to TivaC. Plot all accelerometer and gyro values on to a Graph (you can use any graphing tool).
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.
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 (you can use any graphing tool).

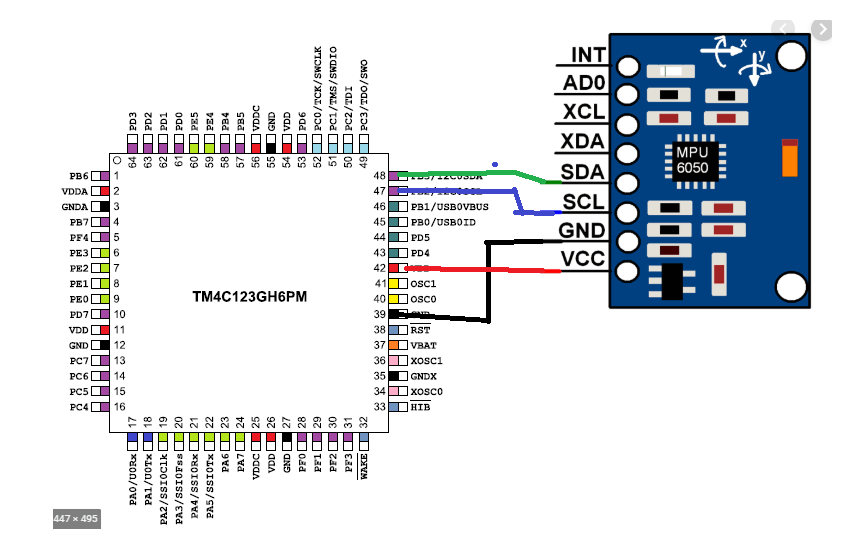
**COMPONENTS**:

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

The main characteristics are to start an I2C communication with the MPU6050 to be able to interface with the TIva C device. I used the I2C module 0 and enabled with to communicate. I also set the pins as SCL and SDA to specify between clock and data between Master slave connection. I also set up a FIFO at 400kbps. Then I cleared it to get rid of any unwanted values. Lastly, I initialized the I2C master driver.

After I2C is initialized, I configured the UART communication protocol to interface with the debugger and the TI CCS IDE. After that was complete, I Initialized my MPU device and began configuration of accelerometer and began reading for accelerometer and gyroscope values. Then I passed the values through the Complementary Filter, that was implemented in IQmath to improve calculation rate.

**SCHEMATICS**:



**IIMPLEMENTATION**:

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

Delay() – creates the desired delay in ms

InitI2C0() – Initializes the I2C communication between the MPU and Tiva C device

ConfigureUART() – configures the UART communication with the TI IDE

MPU6050Callback() – creates handshake with MPU and TivaC device

I2CMasterIntHandler()- Creates an interrupt when the Master has a message to distribute

Complementary \_Filter() – Filters the raw values and creates vectors for Pitch and Row.

MPU\_Init() – Initializes the Master (MPU)

MPURMW()-sets up for configuring the accelerometer as +-4g

MPU\_Reset()- resets masters previous configurations

MPU\_R()- calls Master to establish connection

MPU\_ReadData()-scanns for Master connection

MPU\_GetAccFloat()- Read the accelerometer values from MPU

MPU\_GetGyroFloat()- read the Gyro values from MPU

CF()- calls the Complementary Filter function (just for ease of read of MPU example)

MPU6050Example()- This is where the MPU is configured, where the values are obtained for accelerometer and gyro and then are filtered.

Main()- Here is where we initialize I2C module 0, where we configure UART, and where MPU6050example is implemented.

**YOUTUBE\_LINKS:**

Task1: 🡪 <https://www.youtube.com/watch?v=SdjTqF8gKiQ>

* <https://www.youtube.com/watch?v=lUcXiUnWqhQ>

Task3 🡪 <https://www.youtube.com/watch?v=OBeyzBrCuAI>

* <https://www.youtube.com/watch?v=y2yk5oS0Fpo>

**CODE**:

---------------------------------------------Task 1 code----------------------------------------------------------

/\*

\* main.c

\*

\* Created on: Oct 25, 2019

\* Author: rexaul

\*/

**#include** <stdbool.h>

**#include** <stdint.h>

**#include** <stdlib.h>

**#include** <stdio.h>

**#include** <stdarg.h>

**#include** <stdbool.h>

**#include** "sensorlib/i2cm\_drv.c"

**#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** <math.h>

//#include "uart.h"

**#include** "utils/uartstdio.h"

**#include** "driverlib/uart.h"

//

// A boolean that is set when a MPU6050 command has completed.

//

**volatile** bool g\_bMPU6050Done;

//

// I2C master instance

//

**tI2CMInstance** g\_sI2CMSimpleInst;

//

//Device frequency

//

**int** clockFreq;

**void** **InitI2C0**(**void**)

{

//enable I2C module 0

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_I2C0);

//reset module

**SysCtlPeripheralReset**(SYSCTL\_PERIPH\_I2C0);

//enable GPIO peripheral that contains I2C 0

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOB);

// Configure the pin muxing for I2C0 functions on port B2 and B3.

**GPIOPinConfigure**(GPIO\_PB2\_I2C0SCL);

**GPIOPinConfigure**(GPIO\_PB3\_I2C0SDA);

// 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.

// I2C data transfer rate set to 400kbps.

**I2CMasterInitExpClk**(I2C0\_BASE, **SysCtlClockGet**(), true);

//clear I2C FIFOs

HWREG(I2C0\_BASE + I2C\_O\_FIFOCTL) = 80008000;

// Initialize the I2C master driver.

**I2CMInit**(&g\_sI2CMSimpleInst, I2C0\_BASE, INT\_I2C0, 0xff, 0xff, **SysCtlClockGet**());

}

**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** **delayMS**(**int** ms) {

//ROM\_SysCtlDelay( (ROM\_SysCtlClockGet()/(3\*1000))\*ms ) ; // more accurate

**SysCtlDelay**( (**SysCtlClockGet**()/(3\*1000))\*ms ) ; // less accurate

}

//

// The function that is provided by this example as a callback when MPU6050

// transactions have completed.

//

**void** **MPU6050Callback**(**void** \*pvCallbackData, **uint\_fast8\_t** ui8Status)

{

//

// See if an error occurred.

//

**if** (ui8Status != I2CM\_STATUS\_SUCCESS)

{

//

// An error occurred, so handle it here if required.

//

}

//

// Indicate that the MPU6050 transaction has completed.

//

g\_bMPU6050Done = true;

}

//

// The interrupt handler for the I2C module.

//

**void** **I2CMSimpleIntHandler**(**void**)

{

//

// Call the I2C master driver interrupt handler.

//

**I2CMIntHandler**(&g\_sI2CMSimpleInst);

}

//

// The MPU6050 example.

//

**void** **MPU6050Example**(**void**)

{

**float** fAccel[3], fGyro[3];

**tMPU6050** sMPU6050;

**float** x = 0, y = 0, z = 0;

**float** xx = 0, yy = 0, zz = 0;

//

// Initialize the MPU6050. This code assumes that the I2C master instance

// has already been initialized.

//

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)

{

}

//

// Loop forever reading data from the MPU6050. Typically, this process

// would be done in the background, but for the purposes of this example,

// it is shown in an infinite loop.

//

**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]);

//

// Do something with the new accelerometer and gyroscope readings.

//

xx = fGyro[0]\*10000;

yy = fGyro[1]\*10000;

zz = fGyro[2]\*10000;

x = (**atan2**(fAccel[0], **sqrt** (fAccel[1] \* fAccel[1] + fAccel[2] \* fAccel[2]))\*180.0)/3.14;

y = (**atan2**(fAccel[1], **sqrt** (fAccel[0] \* fAccel[0] + fAccel[2] \* fAccel[2]))\*180.0)/3.14;

z = (**atan2**(fAccel[2], **sqrt** (fAccel[1] \* fAccel[1] + fAccel[2] \* fAccel[2]))\*180.0)/3.14;

**UARTprintf**("Acc. X: %d | Acc. Y: %d | Acc. Z: %d\n", (**int**)x, (**int**)y, (**int**)z);

**UARTprintf**("Gyro. XX: %d | Gyro. YY: %d | Gyro. ZZ: %d\n", (**int**)xx, (**int**)yy, (**int**)zz);

**delayMS**(1000);

}

}

**int** **main**()

{

//clockFreq = SysCtlClockFreqSet(SYSCTL\_OSC\_INT | SYSCTL\_USE\_PLL | SYSCTL\_CFG\_VCO\_480, 16000000);

**SysCtlClockSet**(SYSCTL\_SYSDIV\_1 | SYSCTL\_USE\_PLL | SYSCTL\_OSC\_INT | SYSCTL\_XTAL\_16MHZ);

**InitI2C0**();

**ConfigureUART**();

**MPU6050Example**();

**return**(0);

}

---------------------------Task 3 Code-----------------------------------

**#include** <stdbool.h>

**#include** <stdint.h>

**#include** <stdlib.h>

**#include** <stdio.h>

**#include** <stdarg.h>

**#include** <stdbool.h>

**#include** "sensorlib/i2cm\_drv.c"

**#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** "IQmath/IQmathLib.h"

**#include** <math.h>

**#include** "utils/uartstdio.h"

**#include** "driverlib/uart.h"

**volatile** bool MPU6050\_DONE;

**#define** **ACCELEROMETER\_SENSITIVITY** 16384

**#define** **GYROSCOPE\_SENSITIVITY** 131

**#define** **SAMPLE\_RATE** 0.01

**#define** **RATIO** (180/3.14)

**tMPU6050** sMPU6050;

**tI2CMInstance** I2CMaster; // I2C Master global instantiation

**\_iq16** Pitch = \_IQ(0);

**\_iq16** Roll = \_IQ(0);

**int** clkFreq;

**void** **delayMS**(**int** ms) {

**SysCtlDelay**( (**SysCtlClockGet**()/(3\*1000))\*ms ) ; // less accurate

}

**void** **InitI2C0**(**void**){

//enable I2C module 0

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_I2C0);

//reset module

**SysCtlPeripheralReset**(SYSCTL\_PERIPH\_I2C0);

//enable GPIO peripheral that contains I2C 0

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOB);

// Configure the pin muxing for I2C0 functions on port B2 and B3.

**GPIOPinConfigure**(GPIO\_PB2\_I2C0SCL);

**GPIOPinConfigure**(GPIO\_PB3\_I2C0SDA);

// 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.

// I2C data transfer rate set to 400kbps.

**I2CMasterInitExpClk**(I2C0\_BASE, **SysCtlClockGet**(), true);

//clear I2C FIFOs

HWREG(I2C0\_BASE + I2C\_O\_FIFOCTL) = 80008000;

// Initialize the I2C master driver.

**I2CMInit**(&I2CMaster, I2C0\_BASE, INT\_I2C0, 0xff, 0xff, **SysCtlClockGet**());

}

**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){

}

MPU6050\_DONE = true;

}

**void** **I2CMasterIntHandler**(**void**){

**I2CMIntHandler**(&I2CMaster);

}

**void** **Complementary\_Filter**(**float** \*fAccel, **float** \*fGyro)

{

**\_iq16** ForceMagApprx, PitchAcc, RollAcc, sensitivity, IQratio, NineEight, ohhTwo;

**\_iq16** GyroVal[3], Acc[3];

IQratio = \_IQ16(RATIO);

NineEight = \_IQ16(0.98);

ohhTwo = \_IQ16(0.02);

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]);

sensitivity = \_IQ16(GYROSCOPE\_SENSITIVITY);

Pitch += **\_IQ16mpy**(**\_IQ16div**(GyroVal[0],sensitivity), \_IQ16(SAMPLE\_RATE));

Roll -= **\_IQ16mpy**(**\_IQ16div**(GyroVal[1],sensitivity), \_IQ16(SAMPLE\_RATE));

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

**UARTprintf**("\nForce Mag Apprx in CF: %d\n\n",(**int**)(ForceMagApprx));

**if**(ForceMagApprx > 1940371 && ForceMagApprx < 4940371){

PitchAcc = **\_IQ16mpy**(**\_IQ16atan2**(Acc[1],Acc[2]), Rat);

//UARTprintf("PitchAcc bet 2g and 4g: %d\n",PitchAcc);

Pitch = **\_IQ16mpy**(Pitch,NineEight) + **\_IQ16mpy**(PitchAcc,ohhTwo);

**UARTprintf**("Pitch bet 2g and 4g: %d\n",(**int**)Pitch);

RollAcc = **\_IQ16mpy**(**\_IQ16atan2**(Acc[0],Acc[2]), Rat);

//UARTprintf("RollAcc bet 2g and 4g: %d\n",RollAcc);

Roll = **\_IQ16mpy**(Roll,NineEight) + **\_IQ16mpy**(RollAcc,ohhTwo);

**UARTprintf**("Roll bet 2g and 4g: %d\n",Roll);

}

}

**void** **MPU\_Init**(**void**){

MPU6050\_DONE = false;

**MPU6050Init**(&sMPU6050, &I2CMaster, 0x68, **MPU6050Callback**, &sMPU6050);

**while** (!MPU6050\_DONE){}

}

**void** **MPURMW**(**void**){

MPU6050\_DONE = false;

**MPU6050ReadModifyWrite**(&sMPU6050, MPU6050\_O\_ACCEL\_CONFIG, ~MPU6050\_ACCEL\_CONFIG\_AFS\_SEL\_M, MPU6050\_ACCEL\_CONFIG\_AFS\_SEL\_4G, **MPU6050Callback**, &sMPU6050);

**while** (!MPU6050\_DONE){}

}

**void** **MPU\_Reset**(**void**){

MPU6050\_DONE = false;

**MPU6050ReadModifyWrite**(&sMPU6050, MPU6050\_O\_PWR\_MGMT\_1, 0x00, 0b00000010 & MPU6050\_PWR\_MGMT\_1\_DEVICE\_RESET, **MPU6050Callback**, &sMPU6050);

**while** (!MPU6050\_DONE){}

}

**void** **MPU\_R**(**void**){

MPU6050\_DONE = false;

**MPU6050ReadModifyWrite**(&sMPU6050, MPU6050\_O\_PWR\_MGMT\_2, 0x00, 0x00, **MPU6050Callback**, &sMPU6050);

**while** (!MPU6050\_DONE){}

}

**void** **MPU\_ReadData**(**void**){

MPU6050\_DONE = false;

**MPU6050DataRead**(&sMPU6050, **MPU6050Callback**, &sMPU6050);

**while** (!MPU6050\_DONE){}

}

**void** **MPU\_GetAccFloat**(**float** \*fAccel, **float** \*fGyro){

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

}

**void** **MPU\_GetGyroFloat**(**float** \*fAccel, **float** \*fGyro){

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

}

**void** **CF**(**float** \*fAccel, **float** \*fGyro){

**Complementary\_Filter**(fAccel, fGyro);

}

**void** **MPU6050Example**(**void**)

{

**float** fAccel[3], fGyro[3];

**MPU\_Init**();

// configure to get 4g on accelerometer

**MPURMW**();

// here we reset previouse device settings

**MPU\_Reset**();

**MPU\_R**();

**while** (1){

**MPU\_ReadData**();

// Get the new accelerometer and gyroscope readings.

**MPU\_GetAccFloat**(fAccel,fGyro);

**MPU\_GetGyroFloat**(fAccel,fGyro);

**CF**(fAccel,fGyro);

**delayMS**(1000);

}

}

**int** **main**()

{

**SysCtlClockSet**(SYSCTL\_SYSDIV\_1 | SYSCTL\_USE\_PLL | SYSCTL\_OSC\_INT | SYSCTL\_XTAL\_16MHZ);

**InitI2C0**();

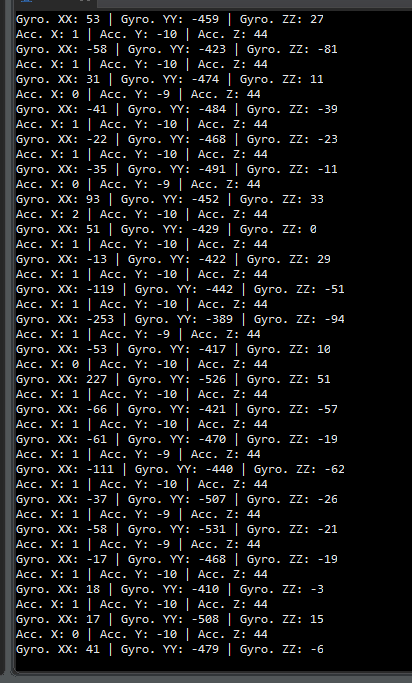
**ConfigureUART**();

**MPU6050Example**();

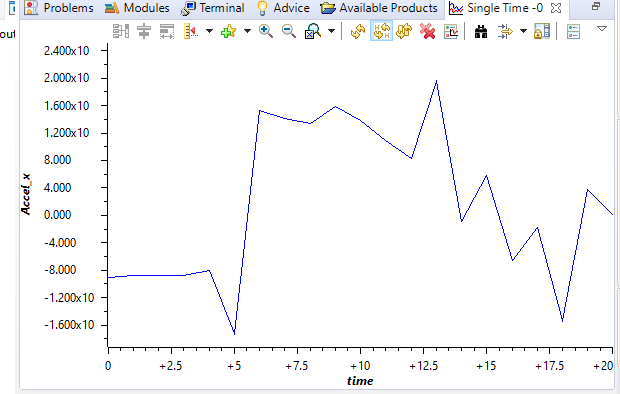
**return**(0);

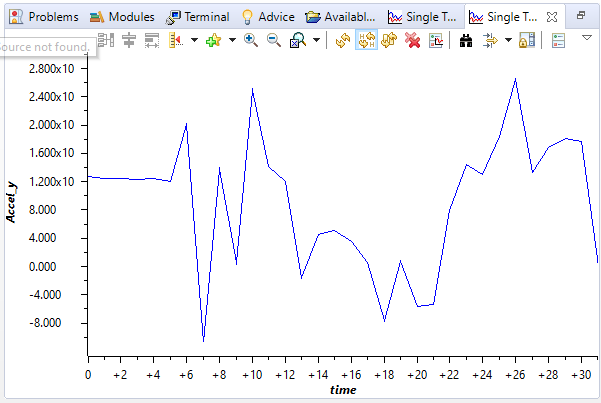
}

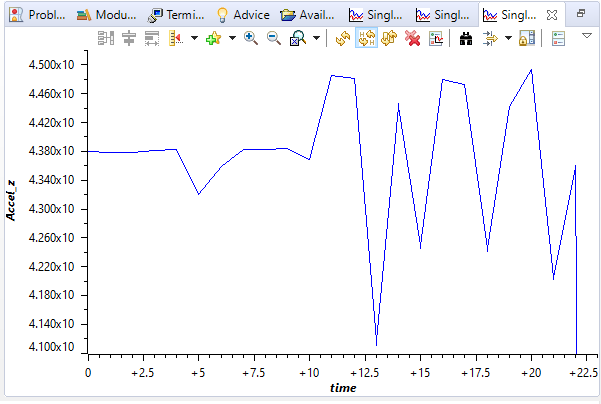
**Task1:**

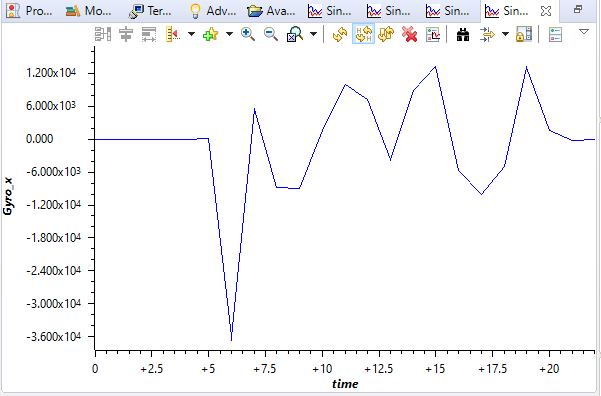


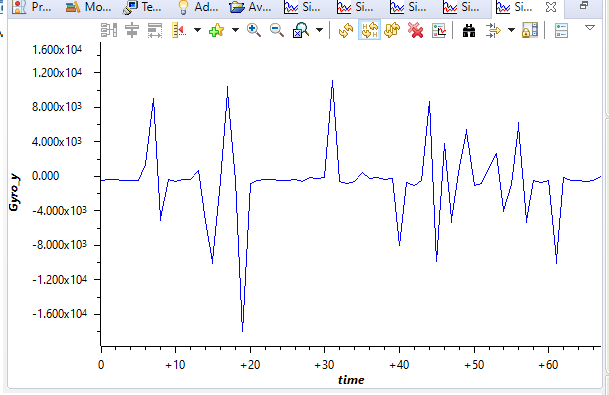
**Task 2 Snips of graphs:**

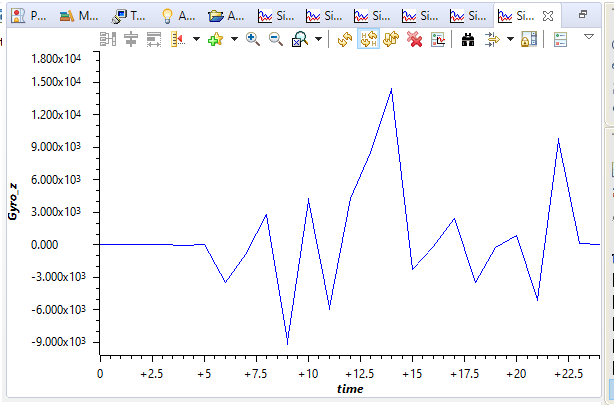




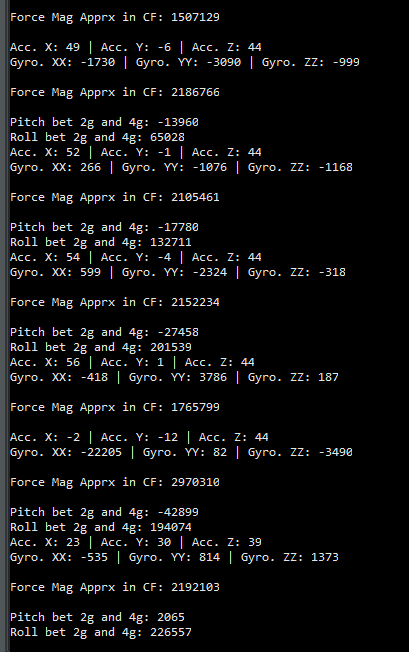








**Task 3**



**Task 4 snips of Complementary filter:**

