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).

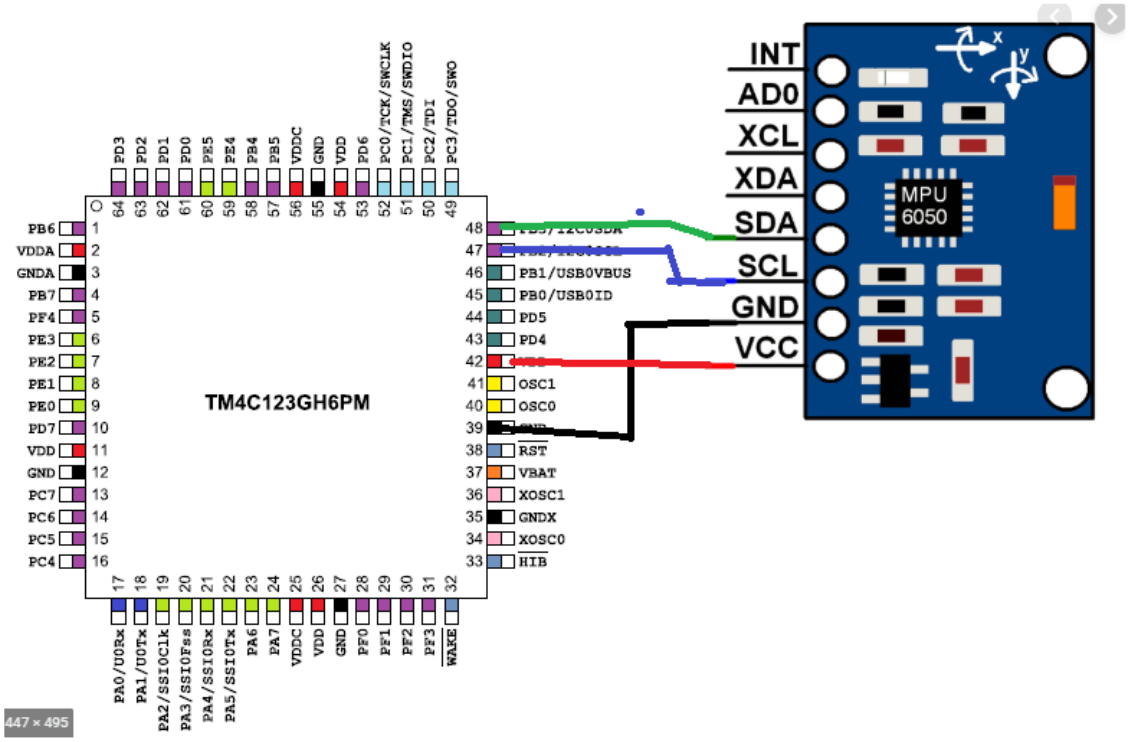
COMPONENTS:

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>
```

```c
#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);
```

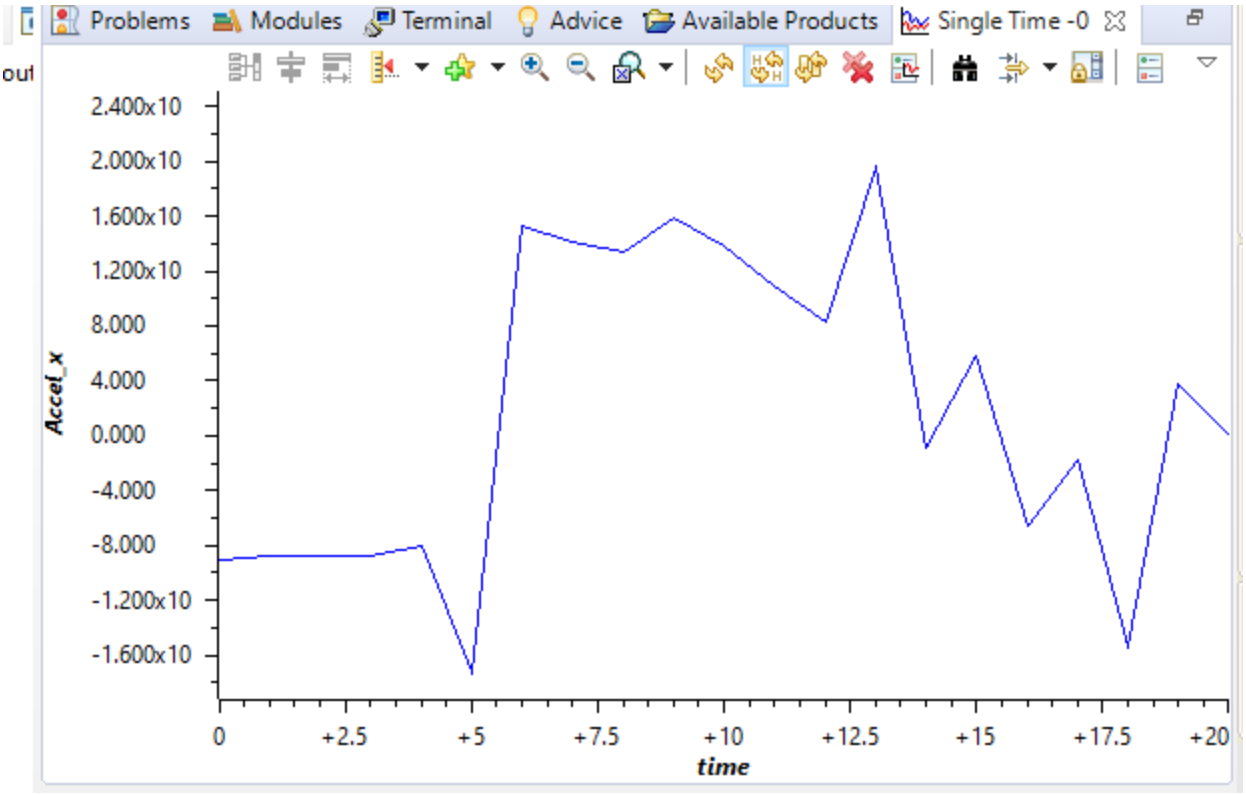
```
    Pitch += _IQ16mpy(_IQ16div(GyroVal[0],sensitivity), _IQ16(SAMPLE_RATE));
    Roll -= _IQ16mpy(_IQ16div(GyroVal[1],sensitivity), _IQ16(SAMPLE_RATE));

    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);
```

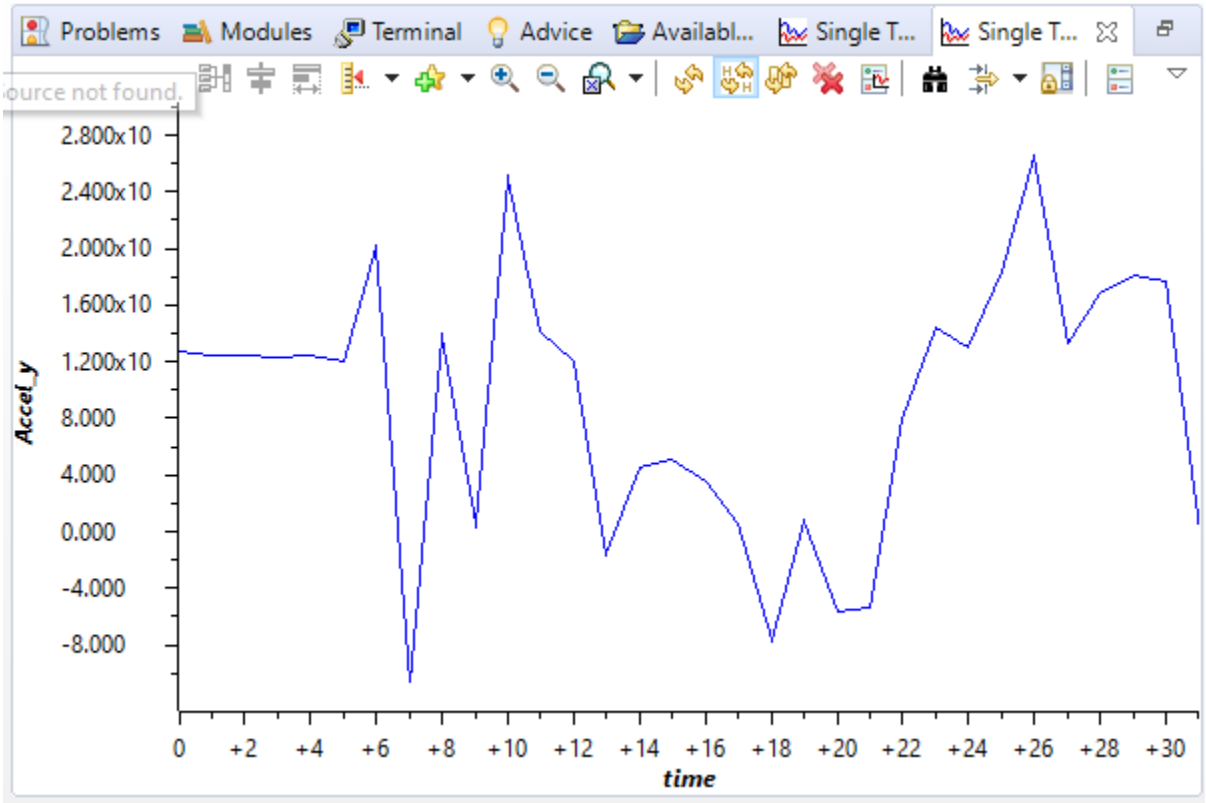
```
    while (!MPU6050_DONE){}
}

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;


}
```

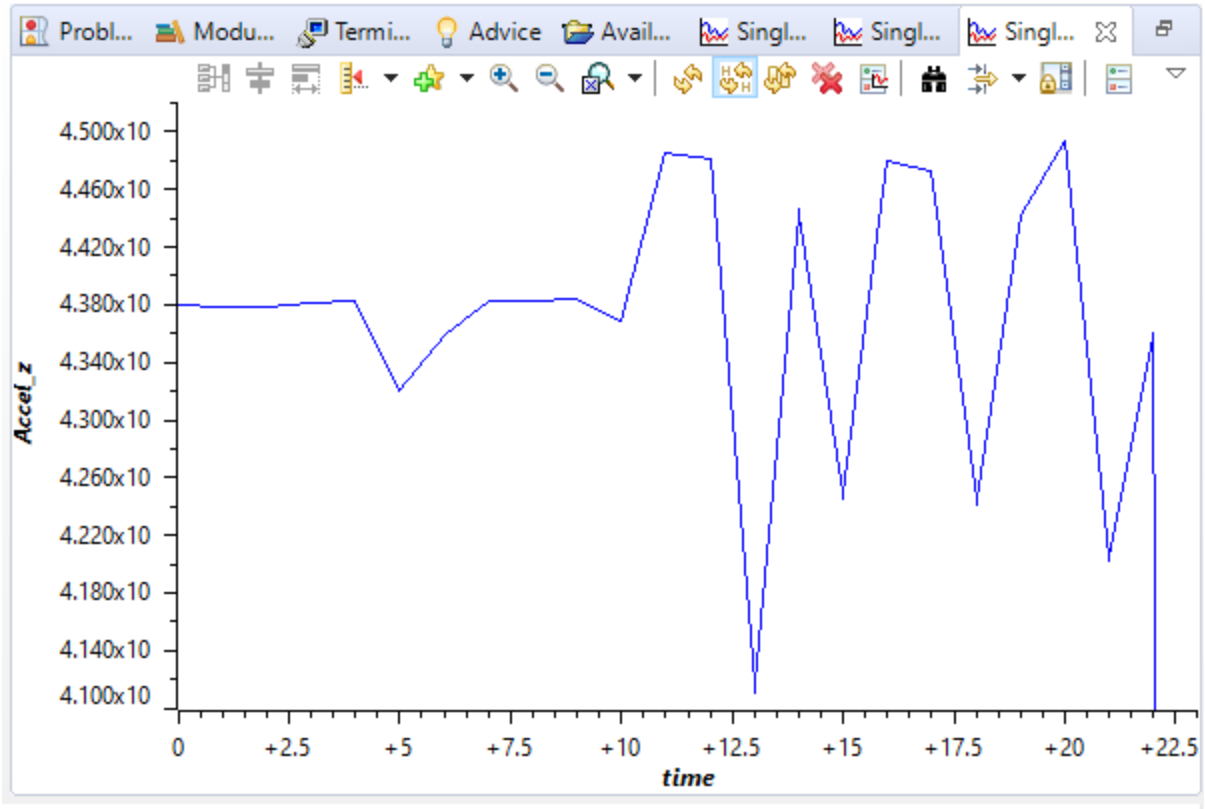
Task1:

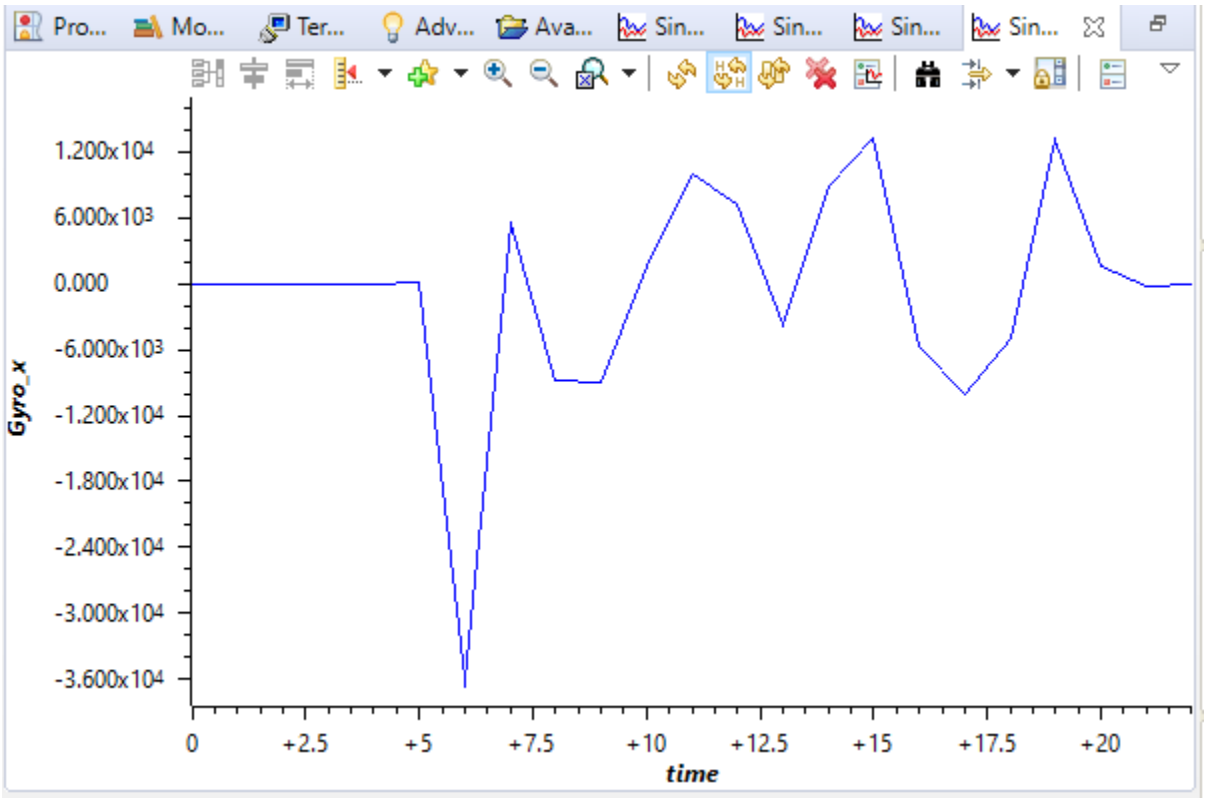
```
Gyro. XX: 53 | Gyro. YY: -459 | Gyro. ZZ: 27
Acc. X: 1 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: -58 | Gyro. YY: -423 | Gyro. ZZ: -81
Acc. X: 1 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: 31 | Gyro. YY: -474 | Gyro. ZZ: 11
Acc. X: 0 | Acc. Y: -9 | Acc. Z: 44
Gyro. XX: -41 | Gyro. YY: -484 | Gyro. ZZ: -39
Acc. X: 1 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: -22 | Gyro. YY: -468 | Gyro. ZZ: -23
Acc. X: 1 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: -35 | Gyro. YY: -491 | Gyro. ZZ: -11
Acc. X: 0 | Acc. Y: -9 | Acc. Z: 44
Gyro. XX: 93 | Gyro. YY: -452 | Gyro. ZZ: 33
Acc. X: 2 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: 51 | Gyro. YY: -429 | Gyro. ZZ: 0
Acc. X: 1 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: -13 | Gyro. YY: -422 | Gyro. ZZ: 29
Acc. X: 1 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: -119 | Gyro. YY: -442 | Gyro. ZZ: -51
Acc. X: 1 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: -253 | Gyro. YY: -389 | Gyro. ZZ: -94
Acc. X: 1 | Acc. Y: -9 | Acc. Z: 44
Gyro. XX: -53 | Gyro. YY: -417 | Gyro. ZZ: 10
Acc. X: 0 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: 227 | Gyro. YY: -526 | Gyro. ZZ: 51
Acc. X: 1 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: -66 | Gyro. YY: -421 | Gyro. ZZ: -57
Acc. X: 1 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: -61 | Gyro. YY: -470 | Gyro. ZZ: -19
Acc. X: 1 | Acc. Y: -9 | Acc. Z: 44
Gyro. XX: -111 | Gyro. YY: -440 | Gyro. ZZ: -62
Acc. X: 1 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: -37 | Gyro. YY: -507 | Gyro. ZZ: -26
Acc. X: 1 | Acc. Y: -9 | Acc. Z: 44
Gyro. XX: -58 | Gyro. YY: -531 | Gyro. ZZ: -21
Acc. X: 1 | Acc. Y: -9 | Acc. Z: 44
Gyro. XX: -17 | Gyro. YY: -468 | Gyro. ZZ: -19
Acc. X: 1 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: 18 | Gyro. YY: -410 | Gyro. ZZ: -3
Acc. X: 1 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: 17 | Gyro. YY: -508 | Gyro. ZZ: 15
Acc. X: 0 | Acc. Y: -10 | Acc. Z: 44
Gyro. XX: 41 | Gyro. YY: -479 | Gyro. ZZ: -6
```

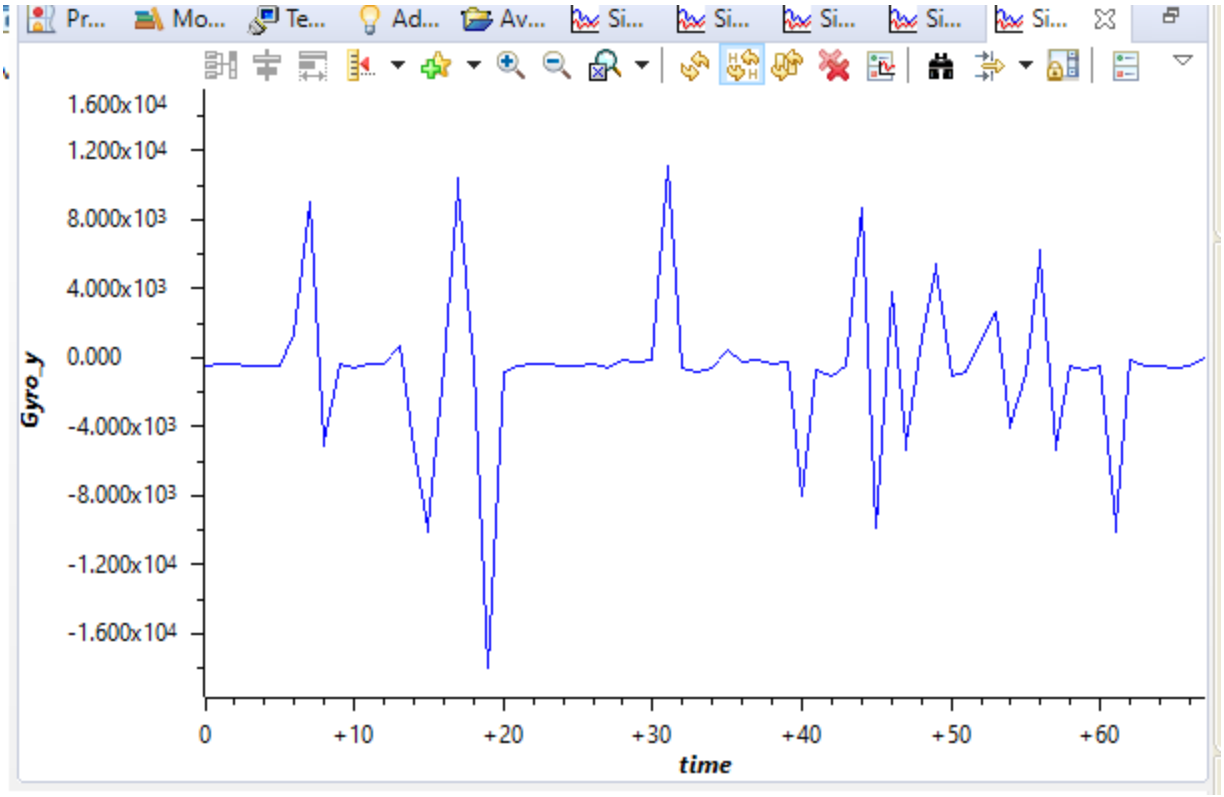
Task 2 Snips of graphs:

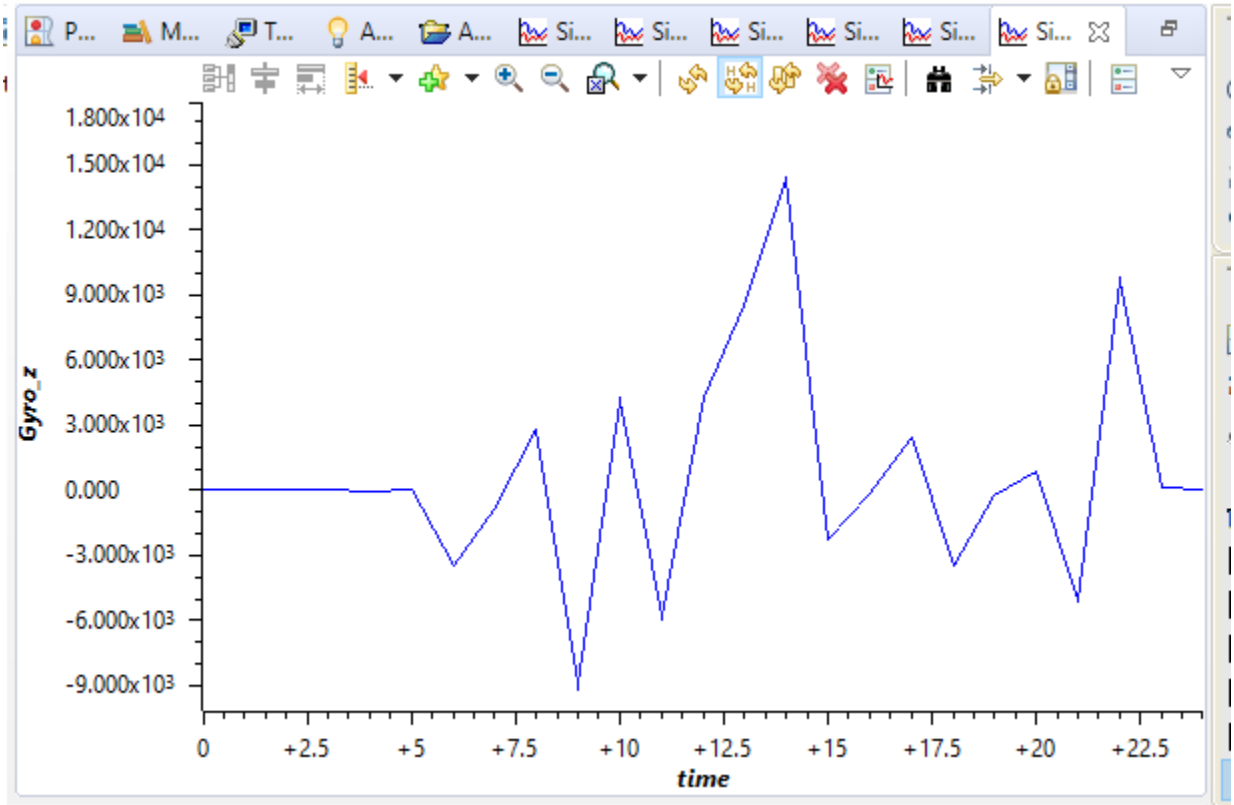

Problems
Modules
Terminal
Advice
Available Products
Single Time -0
2.400x10
2.000x10
1.600x10
1.200x10
8.000
4.000
0.000
-4.000
-8.000
-1.200x10
-1.600x10
Accel_x
0
+2.5
+5
+7.5
+10
+12.5
+15
+17.5
+20
time


Problems
Modules
Terminal
Advice
Availabl...
Single T...
Single T...
ource not found.
2.800x10
2.400x10
2.000x10
1.600x10
1.200x10
8.000
4.000
0.000
-4.000
-8.000
Accel_y
0
+2
+4
+6
+8
+10
+12
+14
+16
+18
+20
+22
+24
+26
+28
+30
time


Probl...
Modu...
Termi...
Advice
Avail...
Singl...
Singl...
Singl...
4.500x10
4.460x10
4.420x10
4.380x10
4.340x10
4.300x10
4.260x10
4.220x10
4.180x10
4.140x10
4.100x10
Accel_z
0
+2.5
+5
+7.5
+10
+12.5
+15
+17.5
+20
+22.5
time


Pro...
Mo...
Ter...
Adv...
Ava...
Sin...
Sin...
Sin...
Sin...
1.200x10⁴
6.000x10³
0.000
-6.000x10³
-1.200x10⁴
-1.800x10⁴
-2.400x10⁴
-3.000x10⁴
-3.600x10⁴
Gyro_x
0
+2.5
+5
+7.5
+10
+12.5
+15
+17.5
+20
time

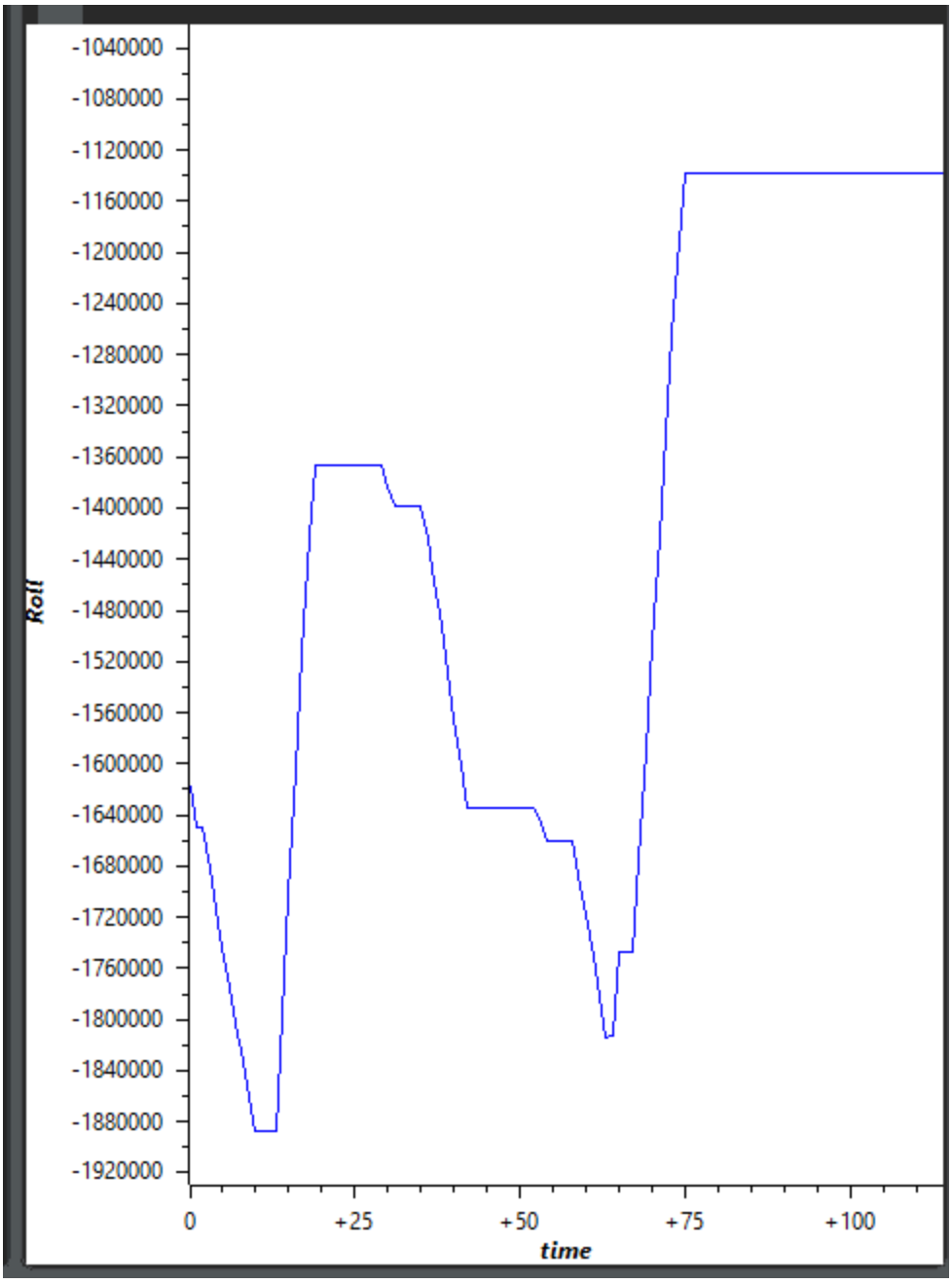

1.600x10⁴
1.200x10⁴
8.000x10³
4.000x10³
0.000
-4.000x10³
-8.000x10³
-1.200x10⁴
-1.600x10⁴
Gyro_y
0
+10
+20
+30
+40
+50
+60
time

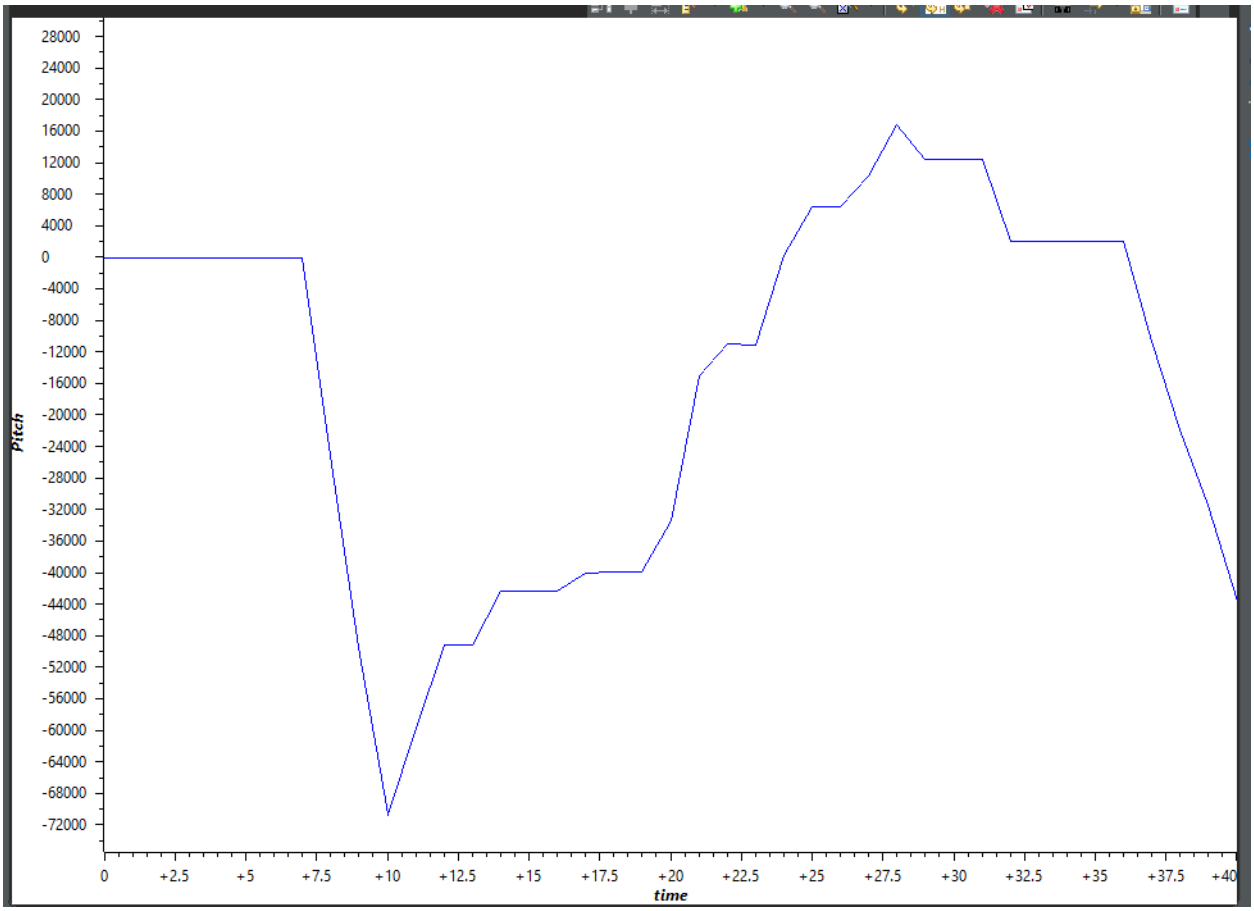

1.800x10⁴
1.500x10⁴
1.200x10⁴
9.000x10³
6.000x10³
3.000x10³
0.000
-3.000x10³
-6.000x10³
-9.000x10³
Gyro_z
0
+2.5
+5
+7.5
+10
+12.5
+15
+17.5
+20
+22.5
time

Task 3

```
Force Mag Apprx in CF: 1507129

Acc. X: 49 | Acc. Y: -6 | Acc. Z: 44
Gyro. XX: -1730 | Gyro. YY: -3090 | Gyro. ZZ: -999

Force Mag Apprx in CF: 2186766

Pitch bet 2g and 4g: -13960
Roll bet 2g and 4g: 65028
Acc. X: 52 | Acc. Y: -1 | Acc. Z: 44
Gyro. XX: 266 | Gyro. YY: -1076 | Gyro. ZZ: -1168

Force Mag Apprx in CF: 2105461

Pitch bet 2g and 4g: -17780
Roll bet 2g and 4g: 132711
Acc. X: 54 | Acc. Y: -4 | Acc. Z: 44
Gyro. XX: 599 | Gyro. YY: -2324 | Gyro. ZZ: -318

Force Mag Apprx in CF: 2152234

Pitch bet 2g and 4g: -27458
Roll bet 2g and 4g: 201539
Acc. X: 56 | Acc. Y: 1 | Acc. Z: 44
Gyro. XX: -418 | Gyro. YY: 3786 | Gyro. ZZ: 187

Force Mag Apprx in CF: 1765799

Acc. X: -2 | Acc. Y: -12 | Acc. Z: 44
Gyro. XX: -22205 | Gyro. YY: 82 | Gyro. ZZ: -3490

Force Mag Apprx in CF: 2970310

Pitch bet 2g and 4g: -42899
Roll bet 2g and 4g: 194074
Acc. X: 23 | Acc. Y: 30 | Acc. Z: 39
Gyro. XX: -535 | Gyro. YY: 814 | Gyro. ZZ: 1373

Force Mag Apprx in CF: 2192103

Pitch bet 2g and 4g: 2065
Roll bet 2g and 4g: 226557
```

Task 4 snips of Complementary filter:


-1040000
-1080000
-1120000
-1160000
-1200000
-1240000
-1280000
-1320000
-1360000
-1400000
-1440000
-1480000
-1520000
-1560000
-1600000
-1640000
-1680000
-1720000
-1760000
-1800000
-1840000
-1880000
-1920000
Roll
0
+25
+50
+75
+100
time


28000
24000
20000
16000
12000
8000
4000
0
-4000
-8000
-12000
-16000
-20000
-24000
-28000
-32000
-36000
-40000
-44000
-48000
-52000
-56000
-60000
-64000
-68000
-72000
Pitch
0
+2.5
+5
+7.5
+10
+12.5
+15
+17.5
+20
+22.5
+25
+27.5
+30
+32.5
+35
+37.5
+40
time