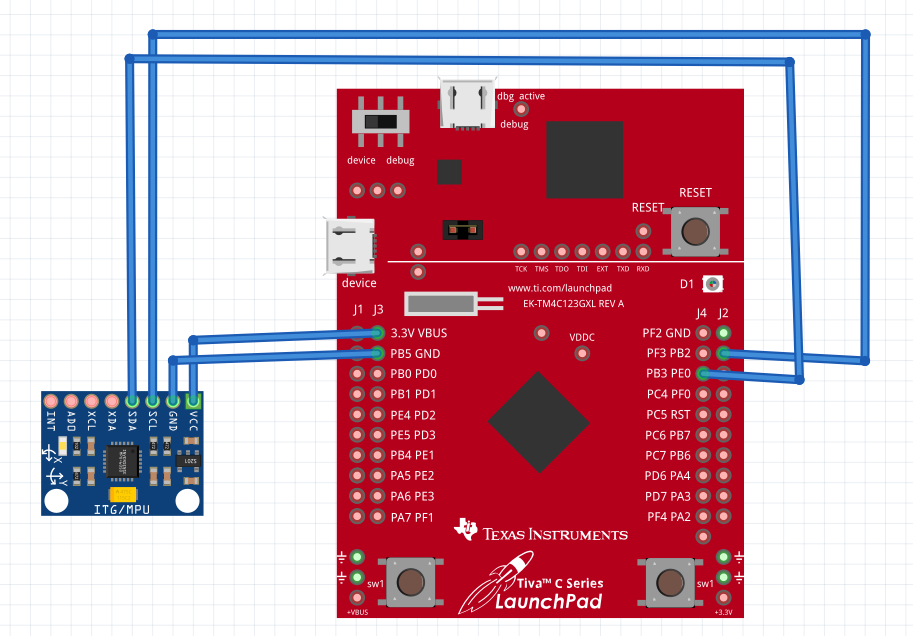
**Date Submitted: 11/14/2019**

**------------------------------------------------------------------------------------**

**Task 01:**

Youtube Link: <https://www.youtube.com/watch?v=TiD0jiqBB-o>

**Modified Schematic (if applicable):**



**Modified Code:**

**#include** <stdarg.h>

**#include** <stdbool.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** "utils/uartstdio.c"

**#include** "math.h"

/\* --------------------------------------------------- \*/

//-------- MPU6050 Register Addresses -----------------//

**#define** MPU\_ADDRESS 0x68

**#define** WHO\_AM\_I 0x75

**#define** PWR\_MGMT\_1 0x6B

**#define** SMPRT\_DIV 0x19

**#define** CONFIG 0x1A

**#define** GYRO\_CONFIG 0x1B

**#define** ACC\_CONFIG 0x1C

**#define** INT\_PIN\_CFG 0x37

**#define** INT\_ENABLE 0x38

**#define** ACCEL\_XOUT\_H 0x3B

**#define** ACCEL\_XOUT\_L 0x3C

**#define** ACCEL\_YOUT\_H 0x3D

**#define** ACCEL\_YOUT\_L 0x3E

**#define** ACCEL\_ZOUT\_H 0x3F

**#define** ACCEL\_ZOUT\_L 0x40

**#define** GYRO\_XOUT\_H 0x43

**#define** GYRO\_XOUT\_L 0x44

**#define** GYRO\_YOUT\_H 0x45

**#define** GYRO\_YOUT\_L 0x46

**#define** GYRO\_ZOUT\_H 0x47

**#define** GYRO\_ZOUT\_L 0x48

//---------------------------------------//

/\* --------------------------------------\*/

// ------- Variable Declarations --------//

**int** WhoAmI, RegReset;

**int** accXout\_L, accXout\_H, accXout;

**int** accYout\_L, accYout\_H, accYout;

**int** accZout\_L, accZout\_H, accZout;

**float** accXos[3] = {0,0,0};

**float** accYos[3]= {0,0,0};

**int** gyroXout\_L, gyroXout\_H, gyroXout;

**int** gyroYout\_L, gyroYout\_H, gyroYout;

**int** gyroZout\_L, gyroZout\_H, gyroZout;

**float** gyroXos[3] = {0,0,0};

**float** gyroYos[3] = {0,0,0};

**float** gyroZos;

**float** fiXos[3]={0,0,0};

**float** fiYos[3]= {0,0,0};

**float** kotXos;

//------------------------------------//

**#define** tip 0.001

**#define** tau 1.6

**#define** ett 0.999375195

**int** data\_ready;

**float** Itemp;

**#define** ACCELEROMETER\_SENSITIVITY 8192.0

**#define** GYROSCOPE\_SENSITIVITY 65.536

**#define** M\_PI 3.14159265359

**#define** dt 0.01 // 10 ms sample rate!

**void** **initMPU6050**()

{

writeI2C(MPU\_ADDRESS, PWR\_MGMT\_1, (1 << 3) || 0x03 ); // power managment setup, temp sensor OFF, sleep mode OFF ...

writeI2C(MPU\_ADDRESS, SMPRT\_DIV, 0x01); // sample rate 1kHz

writeI2C(MPU\_ADDRESS, CONFIG, 0x03); // disable FSYNC, 41 Hz gyro filtering, 1 kHz sampling ??????????

writeI2C(MPU\_ADDRESS, GYRO\_CONFIG, (3 << 3)); // gyro full scale range --> 2000 deg/s (3 << 3)

writeI2C(MPU\_ADDRESS, ACC\_CONFIG, (2 << 3)); // acc full scale range --> 8g (2 << 3)

//writeI2C(MPU\_ADDRESS, INT\_PIN\_CFG, 0x30); // Configure INT pin or 0011 0000 ??? 0x30

//writeI2C(MPU\_ADDRESS, INT\_ENABLE, 0x01); // Enable interrupt DATA READY bit

//SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOE);

//SysCtlDelay(3);

//GPIOPinTypeGPIOInput(GPIO\_PORTE\_BASE, GPIO\_PIN\_2); // Set as input

// GPIOIntTypeSet(GPIO\_PORTE\_BASE, GPIO\_PIN\_2, GPIO\_RISING\_EDGE);

// GPIOIntRegister(GPIO\_PORTE\_BASE, readMPU);

// IntPrioritySet(INT\_GPIOE, 0);

// GPIOIntEnable(GPIO\_PORTE\_BASE, GPIO\_INT\_PIN\_2);

}

//Configures the UART to run at 115200 baud rate

**void** **ConfigureUART**(**void**)

{

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0); //enables UART module 0

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOA); //enables GPIO port a

**GPIOPinConfigure**(GPIO\_PA1\_U0TX); //configures PA1 as TX pin

**GPIOPinConfigure**(GPIO\_PA0\_U0RX); //configures PA0 as RX pin

**GPIOPinTypeUART**(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1); //sets the UART pin type

**UARTClockSourceSet**(UART0\_BASE, UART\_CLOCK\_PIOSC); //sets the clock source

**UARTStdioConfig**(0, 115200, 16000000); //enables UARTstdio baud rate, clock, and which UART to use

}

//Configure/initialize the I2C0

**void** **I2C0\_Init**(**void**)

{

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_I2C0); //enables I2C0

**SysCtlPeripheralReset**(SYSCTL\_PERIPH\_I2C0); //reset module

**SysCtlPeripheralEnable** (SYSCTL\_PERIPH\_GPIOB); //enable PORTB as peripheral

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

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

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

//Select the I2C function for these pins

**GPIOPinTypeI2CSCL** (GPIO\_PORTB\_BASE, GPIO\_PIN\_2); //set I2C PB2 as SCLK

**GPIOPinTypeI2C** (GPIO\_PORTB\_BASE, GPIO\_PIN\_3); //set I2C PB3 as SDA

**I2CMasterInitExpClk** (I2C0\_BASE, **SysCtlClockGet**(), true); //Set the clock of the I2C to ensure proper connection

//while (I2CMasterBusy (I2C0\_BASE)); //wait while the master SDA is busy

}

**void** **readI2C**(uint8\_t slave\_addr, uint8\_t reg, **int** \*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);

}

// Sends 1 byte over i2c

**void** **writeI2C**(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));

}

**void** **readMPU**()

{

readI2C(MPU\_ADDRESS, ACCEL\_XOUT\_H, &accXout\_H);

readI2C(MPU\_ADDRESS, ACCEL\_XOUT\_L, &accXout\_L);

readI2C(MPU\_ADDRESS, ACCEL\_YOUT\_H, &accYout\_H);

readI2C(MPU\_ADDRESS, ACCEL\_YOUT\_L, &accYout\_L);

readI2C(MPU\_ADDRESS, ACCEL\_ZOUT\_H, &accZout\_H);

readI2C(MPU\_ADDRESS, ACCEL\_ZOUT\_L, &accZout\_L);

readI2C(MPU\_ADDRESS, GYRO\_XOUT\_H, &gyroXout\_H);

readI2C(MPU\_ADDRESS, GYRO\_XOUT\_L, &gyroXout\_L);

readI2C(MPU\_ADDRESS, GYRO\_YOUT\_H, &gyroYout\_H);

readI2C(MPU\_ADDRESS, GYRO\_YOUT\_L, &gyroYout\_L);

readI2C(MPU\_ADDRESS, GYRO\_ZOUT\_H, &gyroZout\_H);

readI2C(MPU\_ADDRESS, GYRO\_ZOUT\_L, &gyroZout\_L);

accXout = ((accXout\_H << 8) | accXout\_L);

accYout = ((accYout\_H << 8) | accYout\_L);

accZout = ((accZout\_H << 8) | accZout\_L);

gyroXout = ((gyroXout\_H << 8) | gyroXout\_L);

gyroYout = ((gyroYout\_H << 8) | gyroYout\_L);

gyroZout = ((gyroZout\_H << 8) | gyroZout\_L);

**if**(accXout&0x8000) accXout|=0xFFFF0000;

**if**(accYout&0x8000) accYout|=0xFFFF0000;

**if**(accZout&0x8000) accZout|=0xFFFF0000;

**if**(gyroXout&0x8000) gyroXout|=0xFFFF0000;

**if**(gyroYout&0x8000) gyroYout|=0xFFFF0000;

**if**(gyroZout&0x8000) gyroZout|=0xFFFF0000;

accXos[1]=accXos[0];

accYos[1]=accYos[0];

gyroXos[1]=gyroXos[0];

gyroYos[1]=gyroYos[0];

fiXos[2]=fiXos[1];

fiXos[1]=fiXos[0];

fiYos[2]=fiYos[1];

fiYos[1]=fiYos[0];

**float** a = 2\*ett,

b = -ett\*ett,

c = tip\*ett/tau-ett+1,

d = ett\*ett-tip\*ett/tau-ett,

e = tip\*ett;

accXos[0] = -**atan2**(accXout, accZout);

accYos[0] = -**atan2**(accYout, accZout);

gyroXos[0] = (**float**)gyroYout \* 0.00106422515365507901031932363932f; // pi/(180\*16.4)

gyroYos[0] = -(**float**)gyroXout \* 0.00106422515365507901031932363932f;

gyroZos = (**float**)gyroZout \* 0.06097478f; //degree / second

fiXos[0] = a\*fiXos[1] + b\*fiXos[2] + c\*accXos[0] + d\*accXos[1] + e\*(gyroXos[0]-gyroXos[1]);

fiYos[0] = a\*fiYos[1] + b\*fiYos[2] + c\*accYos[0] + d\*accYos[1] + e\*(gyroYos[0]-gyroYos[1]);

}

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

{

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_XTAL\_16MHZ|SYSCTL\_OSC\_MAIN); //set the main clock to run at 40MHz

ConfigureUART();

**UARTprintf**("Uart Configured\r\n");

**SysCtlDelay**(13500000); // ~1sec delay

I2C0\_Init();

**UARTprintf**("IC2 initialized\r\n");

**SysCtlDelay**(13500000); // ~1sec delay

initMPU6050();

**UARTprintf**("MPU6050 initialized\r\n");

**UARTprintf**("Starting data collection\r\n");

**SysCtlDelay**(13500000); // ~1sec delay

**while**(1)

{

readMPU();

**UARTprintf**("AX: %d\r\n", accXout);

**UARTprintf**("AY: %d\r\n", accYout);

**UARTprintf**("AZ: %d\r\n", accZout);

**UARTprintf**("GX: %d\r\n", gyroXout);

**UARTprintf**("GY: %d\r\n", gyroYout);

**UARTprintf**("GZ: %d\r\n", gyroZout);

**SysCtlDelay**(13500000); // ~1sec delay

}

}

**------------------------------------------------------------------------------------**

**Task 02:**

Youtube Link: <https://www.youtube.com/watch?v=6Bf9JkX-s5Q>

**Modified Schematic (if applicable): Same as Task 1**

**Modified Code:**

**Same code as Task 01**

**------------------------------------------------------------------------------------**

**Task 03:**

Youtube Link: <https://www.youtube.com/watch?v=BBN4dFjWfVM>

**Modified Schematic (if applicable): Same as Task 1**

**Modified Code:**

**#include** <stdarg.h>

**#include** <stdbool.h>

**#include** <stdint.h>

**#include** <stdio.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** "utils/uartstdio.c"

**#include** "math.h"

/\* --------------------------------------------------- \*/

//-------- MPU6050 Register Addresses -----------------//

**#define** MPU\_ADDRESS 0x68

**#define** WHO\_AM\_I 0x75

**#define** PWR\_MGMT\_1 0x6B

**#define** SMPRT\_DIV 0x19

**#define** CONFIG 0x1A

**#define** GYRO\_CONFIG 0x1B

**#define** ACC\_CONFIG 0x1C

**#define** INT\_PIN\_CFG 0x37

**#define** INT\_ENABLE 0x38

**#define** ACCEL\_XOUT\_H 0x3B

**#define** ACCEL\_XOUT\_L 0x3C

**#define** ACCEL\_YOUT\_H 0x3D

**#define** ACCEL\_YOUT\_L 0x3E

**#define** ACCEL\_ZOUT\_H 0x3F

**#define** ACCEL\_ZOUT\_L 0x40

**#define** GYRO\_XOUT\_H 0x43

**#define** GYRO\_XOUT\_L 0x44

**#define** GYRO\_YOUT\_H 0x45

**#define** GYRO\_YOUT\_L 0x46

**#define** GYRO\_ZOUT\_H 0x47

**#define** GYRO\_ZOUT\_L 0x48

//---------------------------------------//

/\* --------------------------------------\*/

// ------- Variable Declarations --------//

**int** WhoAmI, RegReset;

**int** accXout\_L, accXout\_H, accXout;

**int** accYout\_L, accYout\_H, accYout;

**int** accZout\_L, accZout\_H, accZout;

**int** pitchp, rollp; //used for printing floats

**int** pitchsign, rollsign; //used for printing negatives

//int fractional;

**float** accXos[3] = {0,0,0};

**float** accYos[3]= {0,0,0};

**int** gyroXout\_L, gyroXout\_H, gyroXout;

**int** gyroYout\_L, gyroYout\_H, gyroYout;

**int** gyroZout\_L, gyroZout\_H, gyroZout;

**float** gyroXos[3] = {0,0,0};

**float** gyroYos[3] = {0,0,0};

**float** gyroZos;

**float** fiXos[3]={0,0,0};

**float** fiYos[3]= {0,0,0};

**float** pitch = 0;

**float** roll = 0;

**float** kotXos;

//------------------------------------//

**#define** tip 0.001

**#define** tau 1.6

**#define** ett 0.999375195

**int** data\_ready;

**float** Itemp;

**#define** ACCELEROMETER\_SENSITIVITY 8192.0

**#define** GYROSCOPE\_SENSITIVITY 65.536

**#define** M\_PI 3.14159265359

**#define** dt 0.01 // 10 ms sample rate!

**void** **initMPU6050**()

{

writeI2C(MPU\_ADDRESS, PWR\_MGMT\_1, (1 << 3) || 0x03 ); // power managment setup, temp sensor OFF, sleep mode OFF ...

writeI2C(MPU\_ADDRESS, SMPRT\_DIV, 0x01); // sample rate 1kHz

writeI2C(MPU\_ADDRESS, CONFIG, 0x03); // disable FSYNC, 41 Hz gyro filtering, 1 kHz sampling ??????????

writeI2C(MPU\_ADDRESS, GYRO\_CONFIG, (3 << 3)); // gyro full scale range --> 2000 deg/s (3 << 3)

writeI2C(MPU\_ADDRESS, ACC\_CONFIG, (2 << 3)); // acc full scale range --> 8g (2 << 3)

//writeI2C(MPU\_ADDRESS, INT\_PIN\_CFG, 0x30); // Configure INT pin or 0011 0000 ??? 0x30

//writeI2C(MPU\_ADDRESS, INT\_ENABLE, 0x01); // Enable interrupt DATA READY bit

//SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOE);

//SysCtlDelay(3);

//GPIOPinTypeGPIOInput(GPIO\_PORTE\_BASE, GPIO\_PIN\_2); // Set as input

// GPIOIntTypeSet(GPIO\_PORTE\_BASE, GPIO\_PIN\_2, GPIO\_RISING\_EDGE);

// GPIOIntRegister(GPIO\_PORTE\_BASE, readMPU);

// IntPrioritySet(INT\_GPIOE, 0);

// GPIOIntEnable(GPIO\_PORTE\_BASE, GPIO\_INT\_PIN\_2);

}

//Configures the UART to run at 115200 baud rate

**void** **ConfigureUART**(**void**)

{

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0); //enables UART module 0

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOA); //enables GPIO port a

**GPIOPinConfigure**(GPIO\_PA1\_U0TX); //configures PA1 as TX pin

**GPIOPinConfigure**(GPIO\_PA0\_U0RX); //configures PA0 as RX pin

**GPIOPinTypeUART**(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1); //sets the UART pin type

**UARTClockSourceSet**(UART0\_BASE, UART\_CLOCK\_PIOSC); //sets the clock source

**UARTStdioConfig**(0, 115200, 16000000); //enables UARTstdio baud rate, clock, and which UART to use

}

//Configure/initialize the I2C0

**void** **I2C0\_Init**(**void**)

{

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_I2C0); //enables I2C0

**SysCtlPeripheralReset**(SYSCTL\_PERIPH\_I2C0); //reset module

**SysCtlPeripheralEnable** (SYSCTL\_PERIPH\_GPIOB); //enable PORTB as peripheral

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

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

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

//Select the I2C function for these pins

**GPIOPinTypeI2CSCL** (GPIO\_PORTB\_BASE, GPIO\_PIN\_2); //set I2C PB2 as SCLK

**GPIOPinTypeI2C** (GPIO\_PORTB\_BASE, GPIO\_PIN\_3); //set I2C PB3 as SDA

**I2CMasterInitExpClk** (I2C0\_BASE, **SysCtlClockGet**(), true); //Set the clock of the I2C to ensure proper connection

//while (I2CMasterBusy (I2C0\_BASE)); //wait while the master SDA is busy

}

**void** **readI2C**(uint8\_t slave\_addr, uint8\_t reg, **int** \*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);

}

// Sends 1 byte over i2c

**void** **writeI2C**(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));

}

**void** **readMPU**()

{

readI2C(MPU\_ADDRESS, ACCEL\_XOUT\_H, &accXout\_H);

readI2C(MPU\_ADDRESS, ACCEL\_XOUT\_L, &accXout\_L);

readI2C(MPU\_ADDRESS, ACCEL\_YOUT\_H, &accYout\_H);

readI2C(MPU\_ADDRESS, ACCEL\_YOUT\_L, &accYout\_L);

readI2C(MPU\_ADDRESS, ACCEL\_ZOUT\_H, &accZout\_H);

readI2C(MPU\_ADDRESS, ACCEL\_ZOUT\_L, &accZout\_L);

readI2C(MPU\_ADDRESS, GYRO\_XOUT\_H, &gyroXout\_H);

readI2C(MPU\_ADDRESS, GYRO\_XOUT\_L, &gyroXout\_L);

readI2C(MPU\_ADDRESS, GYRO\_YOUT\_H, &gyroYout\_H);

readI2C(MPU\_ADDRESS, GYRO\_YOUT\_L, &gyroYout\_L);

readI2C(MPU\_ADDRESS, GYRO\_ZOUT\_H, &gyroZout\_H);

readI2C(MPU\_ADDRESS, GYRO\_ZOUT\_L, &gyroZout\_L);

accXout = ((accXout\_H << 8) | accXout\_L);

accYout = ((accYout\_H << 8) | accYout\_L);

accZout = ((accZout\_H << 8) | accZout\_L);

gyroXout = ((gyroXout\_H << 8) | gyroXout\_L);

gyroYout = ((gyroYout\_H << 8) | gyroYout\_L);

gyroZout = ((gyroZout\_H << 8) | gyroZout\_L);

**if**(accXout&0x8000) accXout|=0xFFFF0000;

**if**(accYout&0x8000) accYout|=0xFFFF0000;

**if**(accZout&0x8000) accZout|=0xFFFF0000;

**if**(gyroXout&0x8000) gyroXout|=0xFFFF0000;

**if**(gyroYout&0x8000) gyroYout|=0xFFFF0000;

**if**(gyroZout&0x8000) gyroZout|=0xFFFF0000;

accXos[1]=accXos[0];

accYos[1]=accYos[0];

gyroXos[1]=gyroXos[0];

gyroYos[1]=gyroYos[0];

fiXos[2]=fiXos[1];

fiXos[1]=fiXos[0];

fiYos[2]=fiYos[1];

fiYos[1]=fiYos[0];

**float** a = 2\*ett,

b = -ett\*ett,

c = tip\*ett/tau-ett+1,

d = ett\*ett-tip\*ett/tau-ett,

e = tip\*ett;

accXos[0] = -**atan2**(accXout, accZout);

accYos[0] = -**atan2**(accYout, accZout);

gyroXos[0] = (**float**)gyroYout \* 0.00106422515365507901031932363932f; // pi/(180\*16.4)

gyroYos[0] = -(**float**)gyroXout \* 0.00106422515365507901031932363932f;

gyroZos = (**float**)gyroZout \* 0.06097478f; //degree / second

fiXos[0] = a\*fiXos[1] + b\*fiXos[2] + c\*accXos[0] + d\*accXos[1] + e\*(gyroXos[0]-gyroXos[1]);

fiYos[0] = a\*fiYos[1] + b\*fiYos[2] + c\*accYos[0] + d\*accYos[1] + e\*(gyroYos[0]-gyroYos[1]);

}

**void** **ComplementaryFilter**(**void**)

{

**float** pitchAcc, rollAcc;

// Integrate the gyroscope data -> int(angularSpeed) = angle

// Angle around the X-axis

pitch += ((**float**)gyroXout / GYROSCOPE\_SENSITIVITY) \* dt;

// Angle around the Y-axis

roll -= ((**float**)gyroYout / GYROSCOPE\_SENSITIVITY) \* dt;

// Compensate for drift with accelerometer data

// Sensitivity = -2 to 2 G at 16Bit -> 2G = 32768 && 0.5G = 8192

**int** forceMagnitudeApprox = **abs**(accXout) + **abs**(accYout) + **abs**(accZout);

**if** (forceMagnitudeApprox > 8192 && forceMagnitudeApprox < 32768)

{

// Turning around the X axis results in a vector on the Y-axis

pitchAcc = **atan2f**((**float**)accYout, (**float**)accZout) \* 180 / M\_PI;

pitch = pitch \* 0.98 + pitchAcc \* 0.02;

// Turning around the Y axis results in a vector on the X-axis

rollAcc = **atan2f**((**float**)accXout, (**float**)accZout) \* 180 / M\_PI;

roll = roll \* 0.98 + rollAcc \* 0.02;

}

rollp = roll \* 10000;

pitchp = pitch \* 10000;

**if**(rollp < 0)

{

rollp = rollp \*-1;

rollsign = 1;

}

**else**

rollsign = 0;

**if**(pitchp < 0)

{

pitchp = pitchp \*-1;

pitchsign = 1;

}

**else**

pitchsign = 0;

}

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

{

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_XTAL\_16MHZ|SYSCTL\_OSC\_MAIN); //set the main clock to run at 40MHz

ConfigureUART();

**UARTprintf**("Uart Configured\r\n");

**SysCtlDelay**(13500000); // ~1sec delay

I2C0\_Init();

**UARTprintf**("IC2 initialized\r\n");

**SysCtlDelay**(13500000); // ~1sec delay

initMPU6050();

**UARTprintf**("MPU6050 initialized\r\n");

**UARTprintf**("Starting data collection\r\n");

**SysCtlDelay**(13500000); // ~1sec delay

**while**(1)

{

readMPU();

ComplementaryFilter();

**UARTprintf**("AX: %d\r\n", accXout);

**UARTprintf**("AY: %d\r\n", accYout);

**UARTprintf**("AZ: %d\r\n", accZout);

**UARTprintf**("GX: %d\r\n", gyroXout);

**UARTprintf**("GY: %d\r\n", gyroYout);

**UARTprintf**("GZ: %d\r\n", gyroZout);

**if**(rollsign == 1)

{

**if**(rollp > 1000 && rollp < 9999)

**UARTprintf**("Roll: -0.%d\r\n", rollp);

**else** **if**(rollp < 1000 && rollp > 100)

**UARTprintf**("Roll: -0.0%d\r\n", rollp);

**else** **if**(rollp < 100)

**UARTprintf**("Roll: -0.00%d\r\n", rollp);

**else**

**UARTprintf**("Roll: -%d.%d\r\n", rollp/10000, (((roll\*-1.0) - rollp/10000)\*1000));

}

**else**

{

**if**(rollp > 1000 && rollp < 9999)

**UARTprintf**("Roll: 0.%d\r\n", rollp);

**else** **if**(rollp < 1000 && rollp > 100)

**UARTprintf**("Roll: 0.0%d\r\n", rollp);

**else** **if**(rollp < 100)

**UARTprintf**("Roll: 0.00%d\r\n", rollp);

**else**

**UARTprintf**("Roll: %d.%d\r\n", rollp/10000, (roll - (rollp/10000))\*1000);

}

**if**(pitchsign == 1)

{

**if**(pitchp > 1000 && pitchp < 9999)

**UARTprintf**("Pitch: -0.%d\r\n", pitchp);

**else** **if**(pitchp < 1000 && pitchp > 100)

**UARTprintf**("Pitch: -0.0%d\r\n", pitchp);

**else** **if**(pitchp < 100)

**UARTprintf**("Pitch: -0.00%d\r\n", pitchp);

**else**

**UARTprintf**("Pitch: -%d.%d\r\n", pitchp/10000, ((pitch\*-1.0) - (pitchp/10000))\*1000);

}

**else**

{

**if**(pitchp > 1000 && pitchp < 9999)

**UARTprintf**("Pitch: 0.%d\r\n", pitchp);

**else** **if**(pitchp < 1000 && pitchp > 100)

**UARTprintf**("Pitch: 0.0%d\r\n", pitchp);

**else** **if**(pitchp < 100)

**UARTprintf**("Pitch: 0.00%d\r\n", pitchp);

**else**

**UARTprintf**("Pitch: %d.%d\r\n", pitchp/10000, (pitch - (pitchp/10000))\*1000);

}

**SysCtlDelay**(13500000); // ~1sec delay

}

}

**------------------------------------------------------------------------------------**

**Task 04:**

Youtube Link: <https://www.youtube.com/watch?v=WdjwwapB-7Y>

**Modified Schematic (if applicable): Same as Task 1**

**Modified Code:**

**Same code as Task 03**

**------------------------------------------------------------------------------------**