CPE301 – SPRING 2019

MIDTERM 3

Student Name: Elmer Mejia

Student #: 5003824808

Student Email: mejiae4@unlv.nevada.edu

Primary Github address: https://github.com/cpemejia/design\_assignments.git

Directory: design\_assignments

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

-Atmega328pb -ICM20948

-Microchip Studio -breadboard

Table

Description automatically generated with medium confidence



1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A**

/\*

\* DA\_6.c

\*

\* Created: 5/5/2021 11:07:44 PM

\* Author : ElmerOMejia

\*/

#define F\_CPU 16000000UL /\* Define CPU clock Frequency e.g. here its 8MHz \*/

#include <avr/io.h> /\* Include AVR std. library file \*/

#include <util/delay.h> /\* Include delay header file \*/

#include <inttypes.h> /\* Include integer type header file \*/

#include <stdlib.h> /\* Include standard library file \*/

#include <stdio.h> /\* Include standard library file \*/

#include <avr/interrupt.h> /\* Include avr interrupt header file \*/

#include <util/twi.h>

#include "i2cmaster.h"

#include "uart.h" /\* Include USART header file \*/

#include "ICM20948.h"

*uint8\_t* ret; // return value

*uint16\_t* raw; // raw sensor value

*uint16\_t* ACC\_Data[3], GRYRO\_Data[3]; // raw values

void ICM20948\_writereg(*uint8\_t* reg, *uint8\_t* val)

{

i2c\_start(ICM20948+I2C\_WRITE);

i2c\_write(reg); // go to register e.g. 106 user control

i2c\_write(val); // set value e.g. to 0100 0000 FIFO enable

i2c\_stop(); // set stop condition = release bus

}

*uint8\_t* ICM20948\_readreg(*uint8\_t* reg)

{

i2c\_start(ICM20948+I2C\_WRITE); // set device address and write mode

i2c\_write(reg); // ACCEL\_XOUT

i2c\_stop(); // set stop condition = release bus

*\_delay\_ms*(100);

i2c\_start(ICM20948+I2C\_READ); // set device address and read mode

raw = i2c\_readNak(); // read one intermediate byte

i2c\_stop();

return raw;

}

*uint16\_t* ICM20948\_readreg16(*uint8\_t* reg)

{

i2c\_start(ICM20948+I2C\_WRITE); // set device address and write mode

i2c\_write(reg); // ACCEL\_XOUT

i2c\_start(ICM20948+I2C\_READ); // set device address and read mode

raw = i2c\_readNak(); // read one intermediate byte

raw = (raw<<8) | i2c\_readNak(); // read last byte

i2c\_stop();

return raw;

}

void ICM20948\_get\_whom\_am\_I()

{

*uint8\_t* data[1];

ICM20948\_writereg(ICM20948\_REG\_PWR\_MGMT\_1, 0x80);

*\_delay\_ms*(10);

ICM20948\_writereg(ICM20948\_REG\_PWR\_MGMT\_1, 0x01);

*\_delay\_ms*(10); // Wait for 200 ms.

data[0]=ICM20948\_readreg(ICM20948\_REG\_WHO\_AM\_I);

if (data[0] != ICM20948\_DEVICE\_ID)

USART\_SendString("Device Not Found\n");

else

USART\_SendString("Device Found\n");

}

void Init\_ICM20948()

{

i2c\_init(); // init I2C interface

USART\_SendString("I2C Initialized\n");

*\_delay\_ms*(2); // Wait for 200 ms.

ICM20948\_get\_whom\_am\_I();

}

void ICM20948\_config(void)

{

ICM20948\_writereg(ICM20948\_REG\_PWR\_MGMT\_1, 0x80); // reset device

*\_delay\_ms*(10); // Wait for 10 ms.

ICM20948\_writereg(ICM20948\_REG\_PWR\_MGMT\_1, 0x01); // power on

*\_delay\_ms*(10);

ICM20948\_writereg(ICM20948\_REG\_BANK\_SEL, 0x20); // Bank 2 select

*\_delay\_ms*(10);

ICM20948\_writereg(ICM20948\_REG\_GYRO\_SMPLRT\_DIV, 0x00); // gyro sample rate

*\_delay\_ms*(10);

ICM20948\_writereg(ICM20948\_REG\_ACCEL\_SMPLRT\_DIV\_1,0x00); // accel sample rate

*\_delay\_ms*(10);

ICM20948\_writereg(ICM20948\_REG\_GYRO\_CONFIG\_1, 0x00); // gyro config

*\_delay\_ms*(10);

ICM20948\_writereg(ICM20948\_REG\_ACCEL\_CONFIG, 0x00); // accel config

*\_delay\_ms*(10);

ICM20948\_writereg(ICM20948\_REG\_BANK\_SEL, 0x00); // bank 0 select

}

int main(void)

{

char buffer[30], float\_[5];

float pitchAcc, rollAcc, yawAcc, pitch, roll, yaw;

*uint8\_t* p,r,y;

USART\_Init(9600);

USART\_SendString("UART Initialized\n");

*\_delay\_ms*(150);

Init\_ICM20948(); // Sensor init

ICM20948\_config();

*\_delay\_ms*(200); // Wait for 200 ms.

while (1)

{

ACC\_Data[0]=ICM20948\_readreg16(ICM20948\_REG\_ACCEL\_XOUT\_H\_SH);

ACC\_Data[1]=ICM20948\_readreg16(ICM20948\_REG\_ACCEL\_YOUT\_H\_SH);

ACC\_Data[2]=ICM20948\_readreg16(ICM20948\_REG\_ACCEL\_ZOUT\_H\_SH);

GRYRO\_Data[0]=ICM20948\_readreg16(ICM20948\_REG\_GYRO\_XOUT\_H\_SH);

GRYRO\_Data[1]=ICM20948\_readreg16(ICM20948\_REG\_GYRO\_YOUT\_H\_SH);

GRYRO\_Data[2]=ICM20948\_readreg16(ICM20948\_REG\_GYRO\_ZOUT\_H\_SH);

pitch += ((float)GRYRO\_Data[0]/ 16384.0) \* 0.01; // angle around x axis

roll -= ((float)GRYRO\_Data[1]/ 16384.0) \* 0.01; // angle around y axis

yaw += ((float)GRYRO\_Data[2]/ 16384.0) \* 0.01; // angle around z axis

int forceMagnitudeApprox = *abs*(ACC\_Data[0]) + *abs*(ACC\_Data[1]) + *abs*(ACC\_Data[2]);

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

// calculate pitch acceleration

pitchAcc = *atan2f*((float)ACC\_Data[1], (float)ACC\_Data[2]) \*180 / 3.14;

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

// calculate roll accelaration

rollAcc = *atan2f*((float)ACC\_Data[0], (float)ACC\_Data[2]) \*180 / 3.14;

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

// calculate yaw acceleration

yawAcc = *atan2f*((float)ACC\_Data[0], (float)ACC\_Data[1]) \*180 / 3.14;

yaw = yaw \* 0.98 + yawAcc \* 0.02;

}

// send pitch, roll, yaw to uart for serial plotter

p = pitch;

r = roll;

y = yaw;

USART\_TxChar(p);

USART\_TxChar(r);

USART\_TxChar(y);

}

}

/\*

\* uart.h, uart.c, i2cmaster.c, and i2cmaster.h, ICM20948.h

\* are not included to save space but were

\* taken from cpe301 github.

\* They are included in my github.

\*/

1. **SCREENSHOTS OF EACH TASK OUTPUT**

Graphical user interface, text

Description automatically generated

1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

A close-up of a syringe

Description automatically generated with medium confidence

1. **VIDEO LINKS OF EACH DEMO**

[MID3](https://youtu.be/TqQJn5NA0RQ)

1. **GITHUB LINK OF THIS DA**

[Design\_assign](https://github.com/cpemejia/design_assignments.git)

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

-Elmer Mejia