CPE301 – SPRING 2021

Midterm III

Student Name: Brian Wolak

Student #: 2000509437

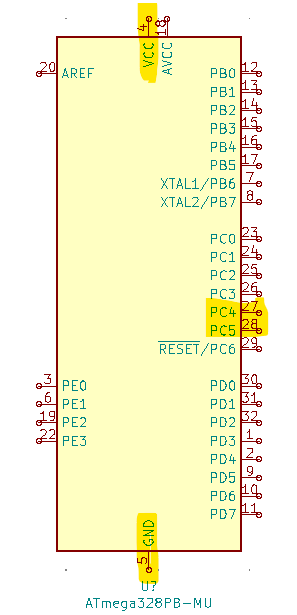
Student Email: wolak@unlv.nevada.edu

Primary Github address: <https://github.com/brianwolak/submission_da.git>

Directory: [submission\_da/MIDTERM\_3 at main · brianwolak/submission\_da (github.com)](https://github.com/brianwolak/submission_da/tree/main/MIDTERM_3)

**Task 1:**

In this design assignment a C code is written to interface a 9-DOF ICM-20948 sensor with the ATmega328pb using i2C. The goal was to gather information from the accelerometer and gyro for each axis (x,y,z). A complimentary filter is then applied to the sensor data and the data is then displayed using a SerialPlot program. Below you can see the code, demonstration video link, and outputs verifying the proper operation.



*Midterm III Ports Used*

**Video Link:**

<https://youtu.be/VXbHa1FhJZA>

**C Code:**

#define *F\_CPU* 16000000UL //CPU clock speed

#define BAUD 9600 //Baud rate is 9600

#include <avr/common.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/setbaud.h>

#include <stdio.h>

#include <util/delay.h>

#include <inttypes.h>

#include <util/twi.h>

#include <math.h>

#define SCL\_CLOCK 100000L

#define TW\_STATUS\_PB (TWSR0 & 0xF8)

#define ACCELEROMETER\_SENSITIVITY 16384.0

#define GYROSCOPE\_SENSITIVITY 131.0

#define dt 0.01// 10 mssample rate!

char display[20], display2[20];

*uint8\_t* who\_am\_i; //device address

*int16\_t* Xacc, Yacc, Zacc, Xgyr, Ygyr, Zgyr;

float Xaccf, Yaccf, Zaccf, Xgyrf, Ygyrf, Zgyrf;

float roll, pitch, yaw;

//UART initialize

void USART\_init(void)

{

UBRR0H = *UBRRH\_VALUE*;

UBRR0L = *UBRRL\_VALUE*;

UCSR0C = \_BV(UCSZ01) | \_BV(UCSZ00);

UCSR0B = \_BV(RXEN0) | \_BV(TXEN0);

}

//UART print function

void USART\_tx\_string(char \*data){

while((\*data != '\0')){ //while the data string is not empty

while(!(UCSR0A & (1 << UDRE0))); //while the data regsiter is not empty

UDR0 = \*data; //UDR0 register receives data

data++; //next data value

}

}

void i2c\_init(void)

{/\* initialize TWI clock: 100 kHz clock, TWPS = 0 => prescaler = 1 \*/

TWSR0 = 0; /\* no prescaler \*/

TWBR0 = ((*F\_CPU*/SCL\_CLOCK)-16)/2; /\* must be > 10 for stable operation 12\*/

}/\* i2c\_init \*/

unsigned char i2c\_start(unsigned char address)

{

*uint8\_t* twst;

// send START condition

TWCR0 = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);

// wait until transmission completed

while(!(TWCR0 & (1<<TWINT)));

// check value of TWI Status Register. Mask prescaler bits.

twst = TW\_STATUS\_PB & 0xF8;

if ( (twst != *TW\_START*) && (twst != *TW\_REP\_START*)) return 1;

// send device address

TWDR0 = address;

TWCR0 = (1<<TWINT) | (1<<TWEN);

// wail until transmission completed and ACK/NACK has been received

while(!(TWCR0 & (1<<TWINT)));

// check value of TWI Status Register. Mask prescaler bits.

twst = TW\_STATUS\_PB & 0xF8;

if ( (twst != *TW\_MT\_SLA\_ACK*) && (twst != *TW\_MR\_SLA\_ACK*) ) return 1;

return 0;

}/\* i2c\_start \*/

unsigned char i2c\_rep\_start(unsigned char address)

{

return i2c\_start( address );

}/\* i2c\_rep\_start \*/

void i2c\_stop(void)

{

/\* send stop condition \*/

TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);

// wait until stop condition is executed and bus released

while(TWCR0 & (1<<TWSTO));

}/\* i2c\_stop \*/

unsigned char i2c\_write( unsigned char data )

{

*uint8\_t* twst;

// send data to the previously addressed device

TWDR0 = data;

TWCR0 = (1<<TWINT) | (1<<TWEN);

// wait until transmission completed

while(!(TWCR0 & (1<<TWINT)));

// check value of TWI Status Register. Mask prescaler bits

twst = TW\_STATUS\_PB & 0xF8;

if( twst != *TW\_MT\_DATA\_ACK*) return 1;

return 0;

}//\* i2c\_write \*

unsigned char i2c\_readAck(void)

{

TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);

while(!(TWCR0 & (1<<TWINT)));

return TWDR0;

}/\* i2c\_readAck \*/

unsigned char i2c\_readNak(void)

{

TWCR0 = (1<<TWINT) | (1<<TWEN);

while(!(TWCR0 & (1<<TWINT)));

return TWDR0;

}/\* i2c\_readNak \*/

void i2c\_start\_wait(unsigned char address)

{

*uint8\_t* twst;

while ( 1 )

{

// send START condition

TWCR0 = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);

// wait until transmission completed

while(!(TWCR0 & (1<<TWINT)));

// check value of TWI Status Register. Mask prescaler bits.

twst = TW\_STATUS\_PB & 0xF8;

if ( (twst != *TW\_START*) && (twst != *TW\_REP\_START*)) continue;

// send device address

TWDR0 = address;

TWCR0 = (1<<TWINT) | (1<<TWEN);

// wail until transmission completed

while(!(TWCR0 & (1<<TWINT)));

// check value of TWI Status Register. Mask prescaler bits.

twst = TW\_STATUS\_PB & 0xF8;

if ( (twst == *TW\_MT\_SLA\_NACK* )||(twst ==*TW\_MR\_DATA\_NACK*) )

{/\* device busy, send stop condition to terminate write operation \*/

TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);

// wait until stop condition is executed and bus released

while(TWCR0 & (1<<TWSTO));

continue;

}

//if( twst != TW\_MT\_SLA\_ACK) return 1;

break;

}

}/\* i2c\_start\_wait \*/

//who\_am\_i function

*uint16\_t* who\_am\_i\_func(void){

*uint16\_t* data;

i2c\_start\_wait(0xD0); //open write communication with 20948

i2c\_write(0x00); //select register address

i2c\_start\_wait(0xD1); //open communication again

data = i2c\_readNak(); //read data value

i2c\_stop(); //stop transmission

return data;

}

//get 16-bit data function

int get\_data(*uint16\_t* addressL, *uint16\_t* addressH){

*int16\_t* data;

i2c\_start\_wait(0xD0); //start communication with sensor

i2c\_write(addressH); //send high register address

i2c\_start\_wait(0xD1); //open communication

data = i2c\_readNak(); //read high register

i2c\_stop(); //stop transmission

i2c\_start\_wait(0xD0); //open write

i2c\_write(addressL); //send low address

i2c\_start\_wait(0xD1); //open communication

data = (data << 8) | i2c\_readNak(); //read values

i2c\_stop(); //stop transmission

return data;

}

//complimentery filter function

void ComplementaryFilter()

{

float pitchAcc, rollAcc, yawAcc;

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

pitch += (Xgyrf / GYROSCOPE\_SENSITIVITY) \* dt; // Angle around the X-axis

roll -= (Ygyrf / GYROSCOPE\_SENSITIVITY) \* dt; // Angle around the Y-axis

yaw += (Zgyrf / GYROSCOPE\_SENSITIVITY) \* dt; // yaw calculation

// Compensate for drift with accelerometer data if !bullshit

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

int forceMagnitudeApprox = *abs*(Xacc) + *abs*(Yacc) + *abs*(Zacc);

if (forceMagnitudeApprox > 8192 && forceMagnitudeApprox < 32768)

{

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

pitchAcc = *atan2f*(Yaccf, Zaccf) \* 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*(Xaccf, Zaccf) \* 180 / *M\_PI*;

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

// Yaw

yawAcc = *atan2f*(Zaccf, Xaccf) \* 180 / *M\_PI*;

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

UDR0 = pitch;

*\_delay\_ms*(1);

UDR0 = roll;

*\_delay\_ms*(1);

UDR0 = yaw;

}

}

int main(void)

{

int device; //device read address

USART\_init(); //function call

i2c\_init(); //initialize i2c

i2c\_start\_wait(0xD0); //open communication with sensor

i2c\_write(0x06); //select register 6

i2c\_write(0x00); //write value to the register location

i2c\_stop(); //stop i2c

//who\_am\_i

who\_am\_i = who\_am\_i\_func();

device = who\_am\_i;

while (1){

//gyro X

Xgyr = get\_data(0x33, 0x34);

Xgyrf = Xgyr / 131.0; //scale with sensitivity factor

//gyro Y

Ygyr = get\_data(0x35, 0x36);

Ygyrf = Ygyr / 131.0; //scale with sensitivity factor

//gyro Z

Zgyr = get\_data(0x37, 0x38);

Zgyrf = Zgyr / 131.0; //scale with sensitivity factor

//accelerometer X

Xacc = get\_data(0x2D, 0x2E);

Xaccf = Xacc / 16384.0; //scale with sensitivity factor

//accelerometer Y

Yacc = get\_data(0x2F, 0x30);

Yaccf = Yacc / 16384.0; //scale with sensitivity factor

//Accelerometer Z

Zacc = get\_data(0x31, 0x32);

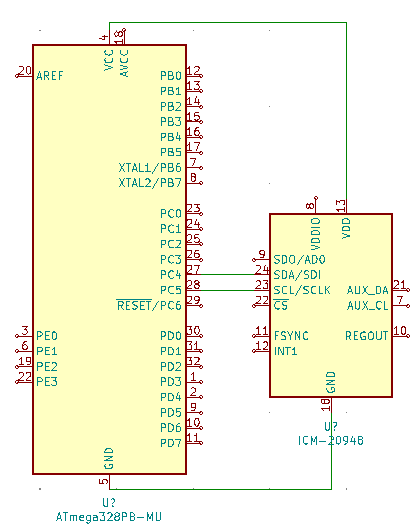
Zaccf = Zacc / 16384.0; //scale with sensitivity factor

//applying complementary filter and print

ComplementaryFilter();

}

}



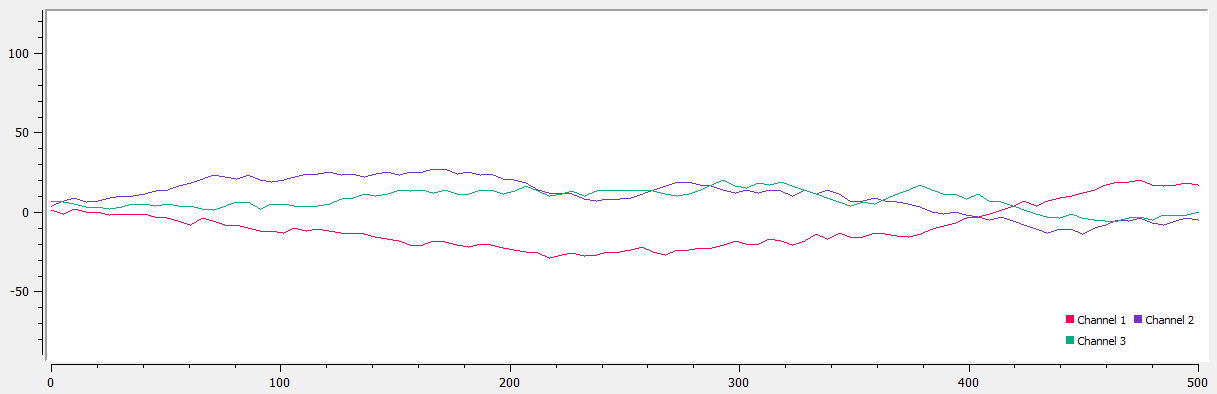
*Midterm III Circuit*

A picture containing text

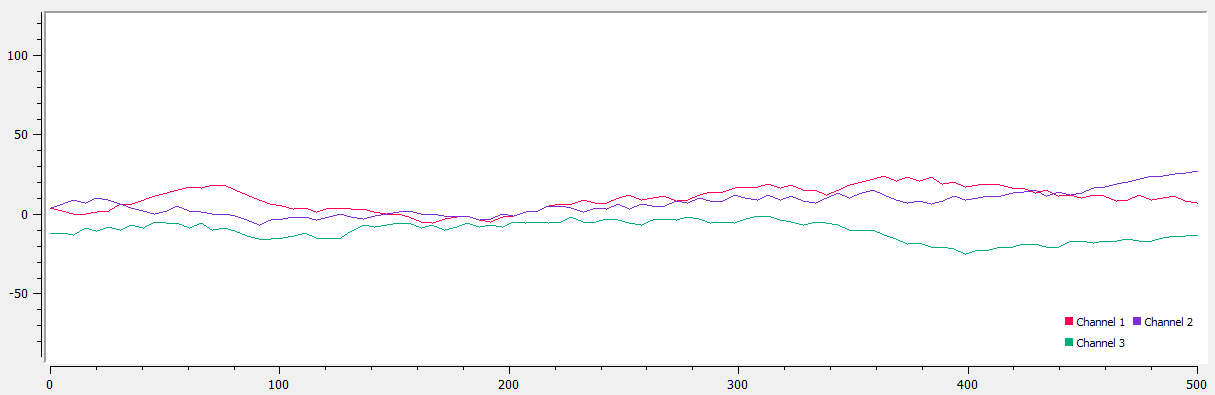
Description automatically generated

*Midterm III Board Setup*

**Serial Plot Outputs**



*Serial Plot Sample Output #1*



*Serial Plot Sample Output #2*

**GitHub Link:**

[submission\_da/MIDTERM\_3 at main · brianwolak/submission\_da (github.com)](https://github.com/brianwolak/submission_da/tree/main/MIDTERM_3)

**Video Link:**

<https://youtu.be/VXbHa1FhJZA>