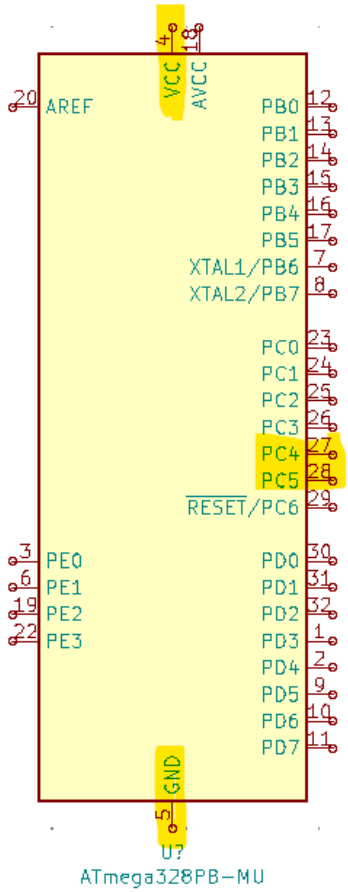
CPE301 – SPRING 2021

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)

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:

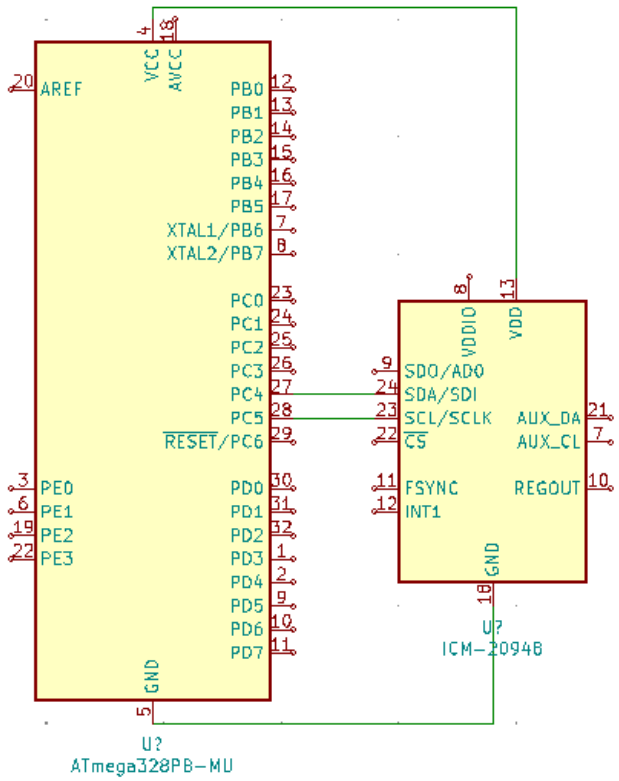
```
#define F_CPU 16000000UL //CPU clock speed
#define BAUD 9600 //Baud rate is 9600
#include <avr/common.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/setbaud.h>
#include <stdio.h>
#include <util/delay.h>
#include <inttypes.h>
#include <util/twi.h>
#include <math.h>
#define SCL_CLOCK 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;
```

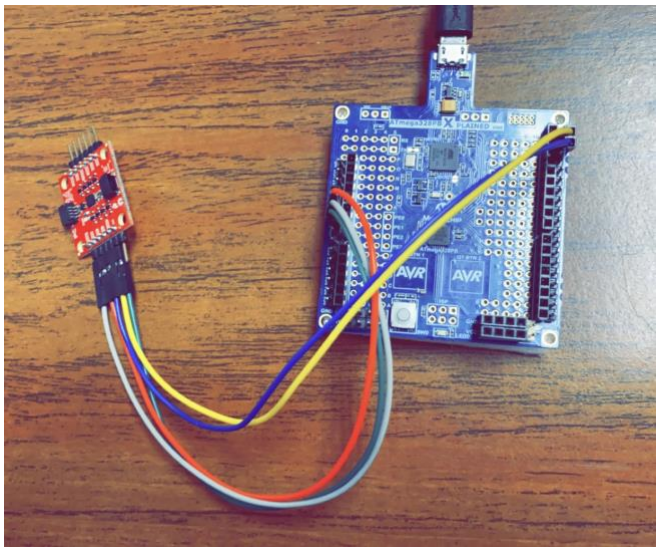
```
        while ( 1 )
        {
        // send START condition
        TWCR0 = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
        // wait until transmission completed
        while(!(TWCR0 & (1<<TWINT)));
        // check value of TWI Status Register. Mask prescaler bits.
        twst = TW_STATUS_PB & 0xF8;
        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();

                }
                                }
```

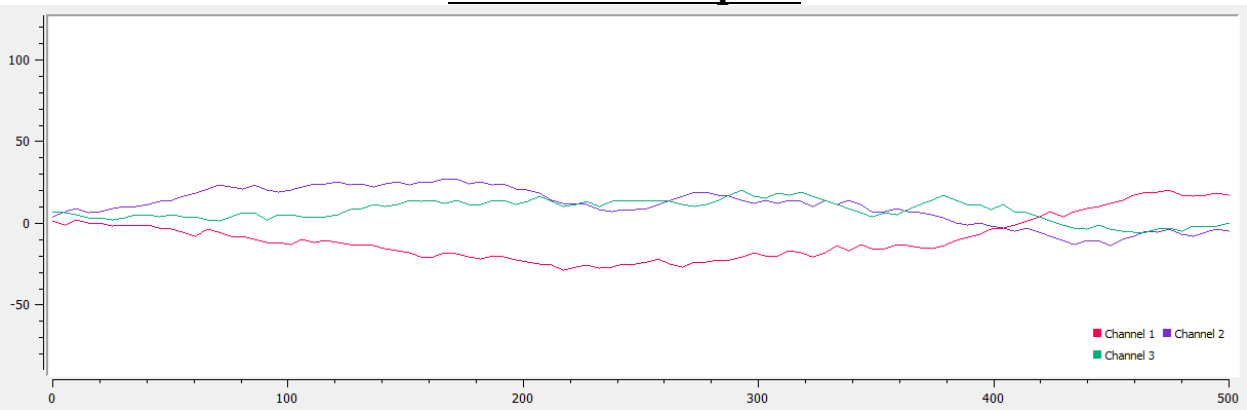


Midterm III Circuit

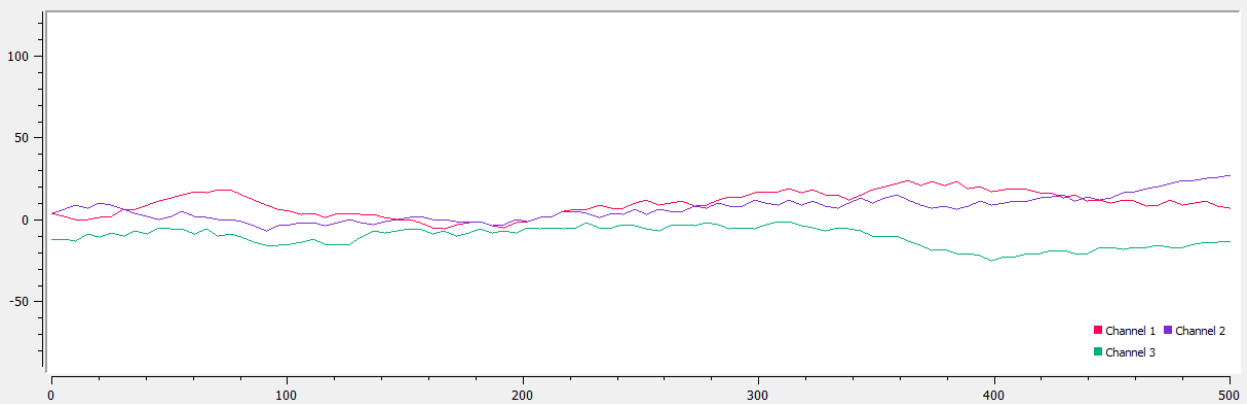


Midterm III Board Setup

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)

Video Link:
https://youtu.be/VXbHa1FhJZA