[redacted]

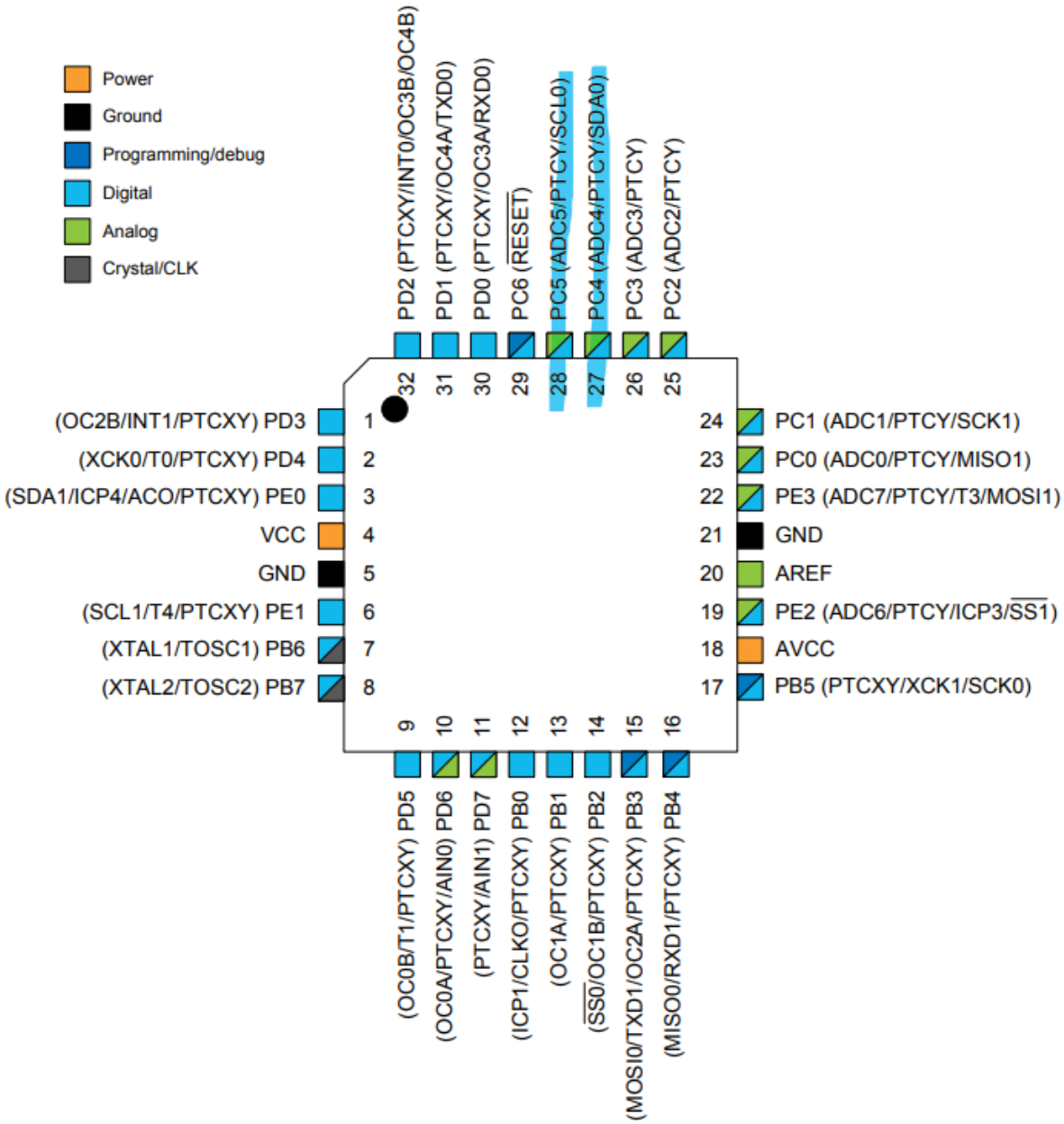
06 ICM20498 Motion Tracking

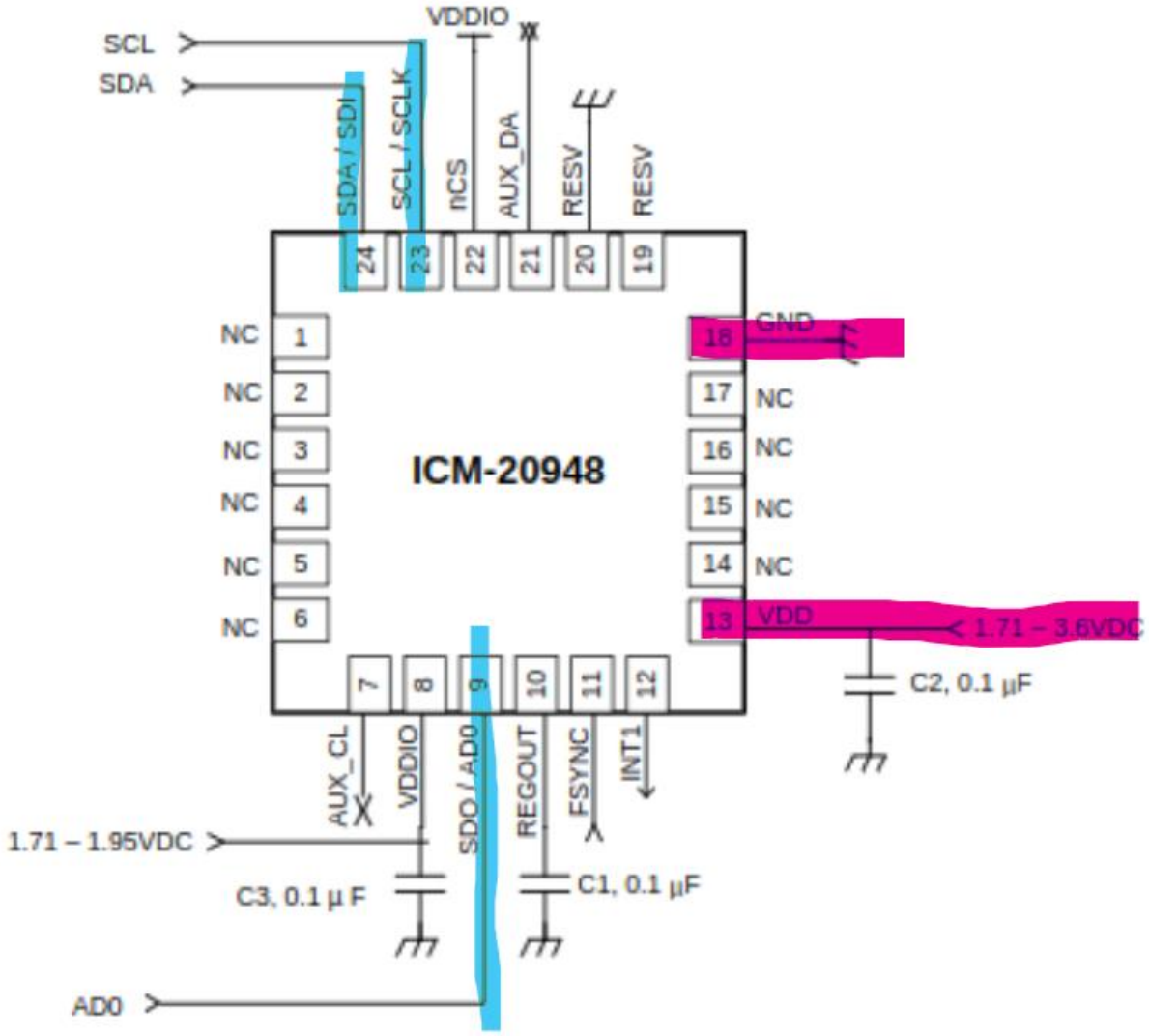
Student Name: [redacted]
Student #: [redacted]
Student Email: [redacted]
Primary Github address: https://github.com/DylanCaz/Submission_DA.git
Directory:
https://github.com/DylanCaz/Submission_DA/tree/main/Design_Assignments_sub/DA_6_sub

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS





2. DEVELOPED CODE OF TASK 1

```
/*************************
Design Assignment 6 Task 1
By: Dylan Cazares
*************************/

#define F_CPU 16000000UL

#include <avr/io.h>
#include <stdio.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <util/twi.h>
```

```
#include "i2cmaster.h"

#define BAUDRATE 9600
#define BAUD_PRESCALLER (((F_CPU / (BAUDRATE * 16UL))) -1)

#define ICM20948     (0x68<<1)     // (1101001 << 1) I2C slave address when AD0=1
#define WHO_AM_I      0x00
#define PWR_MGMT_1    0x06
#define ACCEL_XOUT_H 0x2D
#define ACCEL_YOUT_H 0x2F
#define ACCEL_ZOUT_H 0x31
#define GYRO_XOUT_H   0x33
#define GYRO_YOUT_H   0x35
#define GYRO_ZOUT_H   0x37

/*Magnitometer*/
#define MAGNE_XOUT_H 0x39
#define MAGNE_YOUT_H 0x3B
#define MAGNE_ZOUT_H 0x3D

#define DEVICE_ID     0xEA

uint8_t raw;
uint16_t bigraw;

void USART_init(unsigned int ubrr);
void USART_tx_string(char * data);

//void icm_start_loc(void);
//void read_display_icm(void);

void ICM20948_writereg(uint8_t reg, uint8_t val);

/* functions i modified*/
uint16_t ICM20948_readreg16(uint8_t reg);
void ICM20948_Init(void);
void ICM20948_verify_whoami(void);
void ICM_write(uint8_t reg, uint8_t data);

char buffer[30], myfloat[5];

uint16_t accel_x;
uint16_t accel_y;
uint16_t accel_z;

uint16_t gyro_x;
uint16_t gyro_y;
uint16_t gyro_z;

uint16_t magne_x;
uint16_t magne_y;
uint16_t magne_z;

int main(void)
{
	PORTC |= (1<<5) | (1<<4);                // enable pull ups for TWI pins

	i2c_init();                                          // initialize TWI
```

```
USART_init(BAUD_PRESCALLER);              // initialize USART
USART_tx_string("UART Connected!\r\n");

ICM20948_Init();                          // change clksel on icm
ICM20948_verify_whoami();                 // verify we are connected

_delay_ms(200);

while(1){
        accel_x = ICM20948_readreg16(ACCEL_XOUT_H);
        accel_y = ICM20948_readreg16(ACCEL_YOUT_H);
        accel_z = ICM20948_readreg16(ACCEL_ZOUT_H);

        gyro_x = ICM20948_readreg16(GYRO_XOUT_H);
        gyro_y = ICM20948_readreg16(GYRO_YOUT_H);
        gyro_z = ICM20948_readreg16(GYRO_ZOUT_H);

        magne_x = ICM20948_readreg16(MAGNE_XOUT_H);
        magne_y = ICM20948_readreg16(MAGNE_YOUT_H);
        magne_z = ICM20948_readreg16(MAGNE_ZOUT_H);

        snprintf(buffer,sizeof(buffer),"ACCEL X: %d\r\n",accel_x);
        USART_tx_string(buffer);

        snprintf(buffer,sizeof(buffer),"ACCEL Y: %d\r\n",accel_y);
        USART_tx_string(buffer);

        snprintf(buffer,sizeof(buffer),"ACCEL Z: %d\r\n",accel_z);
        USART_tx_string(buffer);

        USART_tx_string("\r\n");
        _delay_ms(500);

        snprintf(buffer,sizeof(buffer),"GYRO X: %d\r\n",gyro_x);
        USART_tx_string(buffer);

        snprintf(buffer,sizeof(buffer),"GYRO Y: %d\r\n",gyro_y);
        USART_tx_string(buffer);

        snprintf(buffer,sizeof(buffer),"GYRO Z: %d\r\n",gyro_z);
        USART_tx_string(buffer);

        USART_tx_string("\r\n");
        _delay_ms(500);

        snprintf(buffer,sizeof(buffer),"MAGNE X: %d\r\n", magne_x); //
        USART_tx_string(buffer);

        snprintf(buffer,sizeof(buffer),"MAGNE Y: %d\r\n", magne_y);//
        USART_tx_string(buffer);

        snprintf(buffer,sizeof(buffer),"MAGNE Z: %d\r\n", magne_z);//
        USART_tx_string(buffer);

        USART_tx_string("\r\n");
        //read_display_icm();
        _delay_ms(1000);
}
```

```
}

void USART_init(unsigned int ubrr){
	UBRR0H = (unsigned char)(ubrr>>8);
	UBRR0L = (unsigned char)ubrr;
	UCSR0B = (1<<TXEN0);
	UCSR0C = (3<<UCSZ00);
}

void USART_tx_string(char * data){
	while((*data != '\0')){
		while(!(UCSR0A & (1<<UDRE0)));
		UDR0 = *data;
		data++;
	}
}

/* Ensure we are talking to ICM, print WHO_AM_I to terminal, should be 0xEA or 234 */
void ICM20948_verify_whoami(void){
	uint8_t who_am_i = 0;
	i2c_start(ICM20948+I2C_WRITE);		// 68 << 1 = D0
	i2c_write(WHO_AM_I);			// select who_am_i
	i2c_stop();					// halt i2c
	i2c_start(ICM20948+I2C_READ);		// D0 + 1(TWI READ)
	who_am_i = i2c_readNak();		// save to variable
	snprintf(buffer,sizeof(buffer),"DEVICE ID: %02x\r\n",who_am_i);
	USART_tx_string(buffer);
	i2c_stop();
}

/* Change clksel to use best available clock source  */
void ICM20948_Init(void){
	i2c_start(0xD0);				// select ICM20948 (0x68<<1)+0
	i2c_write(0x06);				// select pwr_mgmt_1
	i2c_write(0x01);				// set bit 1
	i2c_stop();


}

/* modified to writes to and reads from reg+1 */
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
	i2c_start(ICM20948+I2C_WRITE);
	i2c_write(reg + 1);
	i2c_start(ICM20948+I2C_READ);
	bigraw = (raw<<8) | i2c_readNak();  // read last byte
	i2c_stop();
	return bigraw;
}

void ICM_write(uint8_t reg, uint8_t data){
	i2c_start(ICM20948+I2C_WRITE);    // start I2C for writing 0xD0
	i2c_write(reg);        // write register address to read
```

```
        i2c_write(data);    // write data to be saved to register
        i2c_stop();            // stop I2C

        ICM_write(0x06, 0x01);    // exit sleep mode, set clk to auto
        ICM_write(0x7F, 0x20);    // select User Bank 2
        ICM_write(0x01, 0x29);    // set gyro rate for 250 with LPF of 17Hz
        ICM_write(0x00, 0x0A);    // set gyroscope sample rate for 100Hz
        ICM_write(0x14, 0x15);    // set accelerometer low pass filter to 136Hz and the
rate to 8G
        ICM_write(0x11, 0x0A);    // set accelerometer rate to 100hz
        ICM_write(0x7F, 0x00);    // select User Bank 0

              /*Initializing Magnetometer */
        ICM_write(0x03, 0x20);        // Enable I2C master
        ICM_write(0x7F, 0x30);        // Select User Bank 3
        ICM_write(0x01, 0x07);        // Set Master Clk speed to 400k
}
```

3. DEVELOPED CODE OF TASK 2

```
/*************************
Design Assignment 6 Task 2
By: Dylan Cazares
*************************/

#define F_CPU 16000000UL

#include <avr/io.h>
#include <stdio.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <util/twi.h>
#include "i2cmaster.h"

#define BAUDRATE 9600
#define BAUD_PRESCALLER (((F_CPU / (BAUDRATE * 16UL))) -1)

#define ICM20948 (0x68<<1)     // (1101001 << 1) I2C slave address when AD0=1
#define WHO_AM_I  0x00
#define PWR_MGMT_1       0x06
#define ACCEL_XOUT_H 0x2D
#define ACCEL_YOUT_H 0x2F
#define ACCEL_ZOUT_H 0x31
#define GYRO_XOUT_H      0x33
#define GYRO_YOUT_H      0x35
#define GYRO_ZOUT_H      0x37

/*Magnitometer*/
#define MAGNE_XOUT_H 0x39
#define MAGNE_YOUT_H 0x3B
#define MAGNE_ZOUT_H 0x3D

#define DEVICE_ID 0xEA

uint8_t raw;
uint16_t bigraw;

void USART_init(unsigned int ubrr);
void USART_tx_string(char * data);
```

```
//void icm_start_loc(void);
//void read_display_icm(void);

void ICM20948_writereg(uint8_t reg, uint8_t val);

/* functions i modified*/
uint16_t ICM20948_readreg16(uint8_t reg);
void ICM20948_Init(void);
void ICM20948_verify_whoami(void);
void ICM_write(uint8_t reg, uint8_t data);

char buffer[30], myfloat[5];

uint16_t accel_x;
uint16_t accel_y;
uint16_t accel_z;

uint16_t gyro_x;
uint16_t gyro_y;
uint16_t gyro_z;

uint16_t magne_x;
uint16_t magne_y;
uint16_t magne_z;

int main(void)
{
    PORTC |= (1<<5) | (1<<4);                // enable pull ups for TWI pins

    i2c_init();                                              // initialize TWI
    USART_init(BAUD_PRESCALLER);             // initialize USART
    USART_tx_string("UART Connected!\r\n");

    ICM20948_Init();                         // change clksel on icm
    ICM20948_verify_whoami();                // verify we are connected

    _delay_ms(200);

    while(1){
           accel_x = ICM20948_readreg16(ACCEL_XOUT_H);
           accel_y = ICM20948_readreg16(ACCEL_YOUT_H);
           accel_z = ICM20948_readreg16(ACCEL_ZOUT_H);

           gyro_x = ICM20948_readreg16(GYRO_XOUT_H);
           gyro_y = ICM20948_readreg16(GYRO_YOUT_H);
           gyro_z = ICM20948_readreg16(GYRO_ZOUT_H);

           magne_x = ICM20948_readreg16(MAGNE_XOUT_H);
           magne_y = ICM20948_readreg16(MAGNE_YOUT_H);
           magne_z = ICM20948_readreg16(MAGNE_ZOUT_H);

           snprintf(buffer,sizeof(buffer),"ACCEL X: %d\r\n",accel_x);
           USART_tx_string(buffer);

           snprintf(buffer,sizeof(buffer),"ACCEL Y: %d\r\n",accel_y);
           USART_tx_string(buffer);
```

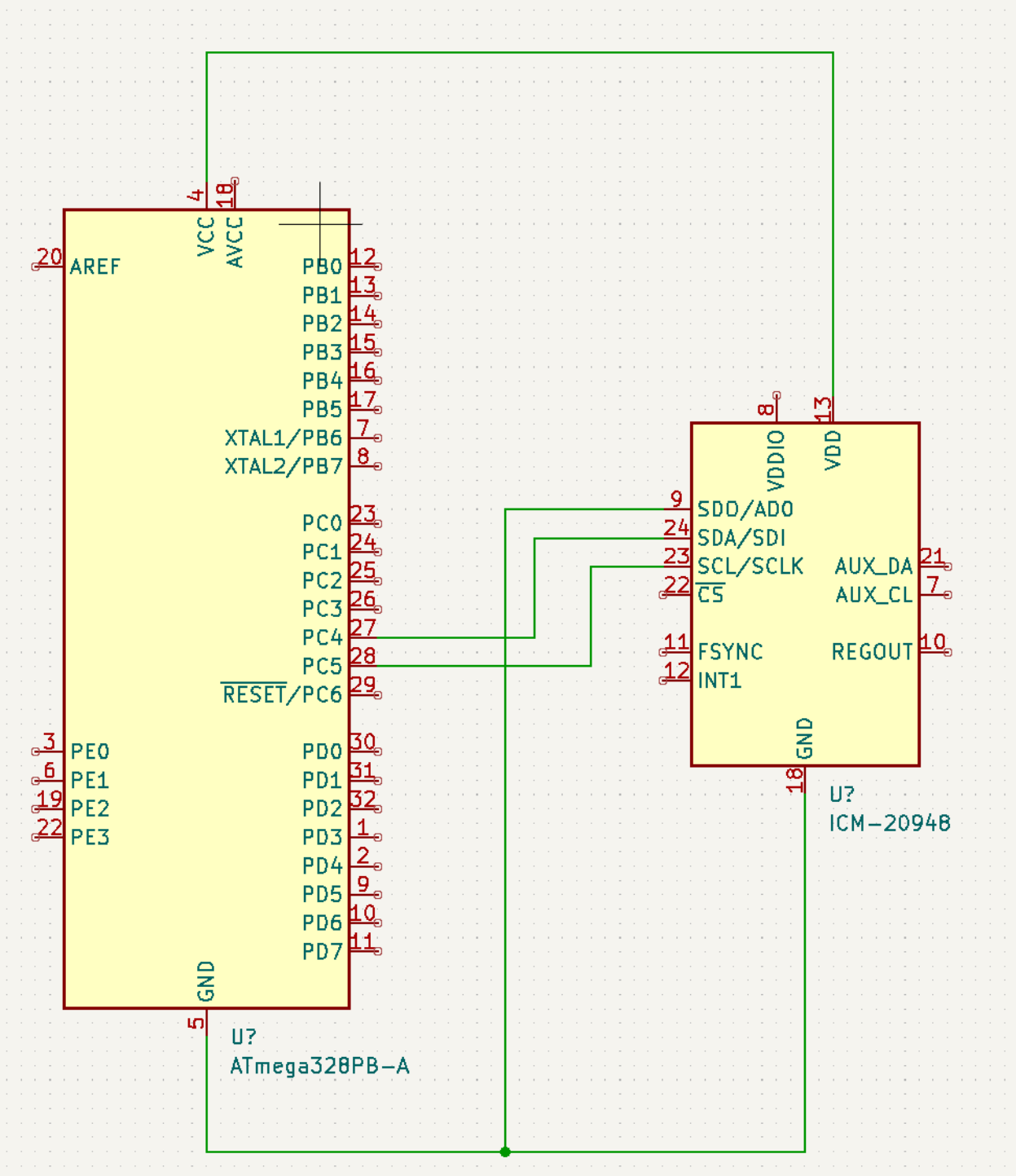
```
            snprintf(buffer,sizeof(buffer),"ACCEL Z: %d\r\n",accel_z);
            USART_tx_string(buffer);

            USART_tx_string("\r\n");
            _delay_ms(500);

            snprintf(buffer,sizeof(buffer),"GYRO X: %d\r\n",gyro_x);
            USART_tx_string(buffer);

            snprintf(buffer,sizeof(buffer),"GYRO Y: %d\r\n",gyro_y);
            USART_tx_string(buffer);

            snprintf(buffer,sizeof(buffer),"GYRO Z: %d\r\n",gyro_z);
            USART_tx_string(buffer);

            USART_tx_string("\r\n");
            _delay_ms(500);

            snprintf(buffer,sizeof(buffer),"MAGNE X: %d\r\n", magne_x); //
            USART_tx_string(buffer);

            snprintf(buffer,sizeof(buffer),"MAGNE Y: %d\r\n", magne_y);//
            USART_tx_string(buffer);

            snprintf(buffer,sizeof(buffer),"MAGNE Z: %d\r\n", magne_z);//
            USART_tx_string(buffer);

            USART_tx_string("\r\n");
            //read_display_icm();
            _delay_ms(1000);
    }
}

void USART_init(unsigned int ubrr){
    UBRR0H = (unsigned char)(ubrr>>8);
    UBRR0L = (unsigned char)ubrr;
    UCSR0B = (1<<TXEN0);
    UCSR0C = (3<<UCSZ00);
}

void USART_tx_string(char * data){
    while((*data != '\0')){
            while(!(UCSR0A & (1<<UDRE0)));
            UDR0 = *data;
            data++;
    }
}

/* Ensure we are talking to ICM, print WHO_AM_I to terminal, should be 0xEA or 234 */
void ICM20948_verify_whoami(void){
    uint8_t who_am_i = 0;
    i2c_start(ICM20948+I2C_WRITE);           // 68 << 1 = D0
    i2c_write(WHO_AM_I);                     // select who_am_i
    i2c_stop();                                          // halt i2c
    i2c_start(ICM20948+I2C_READ);            // D0 + 1(TWI READ)
    who_am_i = i2c_readNak();                // save to variable
    snprintf(buffer,sizeof(buffer),"DEVICE ID: %02x\r\n",who_am_i);
    USART_tx_string(buffer);
```

```
    i2c_stop();
}

/* Change clksel to use best available clock source  */
void ICM20948_Init(void){
    i2c_start(0xD0);                                    // select ICM20948 (0x68<<1)+0
    i2c_write(0x06);                                    // select pwr_mgmt_1
    i2c_write(0x01);                                    // set bit 1
    i2c_stop();


}

/* modified to writes to and reads from reg+1 */
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
    i2c_start(ICM20948+I2C_WRITE);
    i2c_write(reg + 1);
    i2c_start(ICM20948+I2C_READ);
    bigraw = (raw<<8) | i2c_readNak();  // read last byte
    i2c_stop();
    return bigraw;
}

void ICM_write(uint8_t reg, uint8_t data){
    i2c_start(ICM20948+I2C_WRITE);    // start I2C for writing 0xD0
    i2c_write(reg);        // write register address to read
    i2c_write(data);    // write data to be saved to register
    i2c_stop();            // stop I2C

    ICM_write(0x06, 0x01);    // exit sleep mode, set clk to auto
    ICM_write(0x7F, 0x20);    // select User Bank 2
    ICM_write(0x01, 0x29);    // set gyro rate for 250 with LPF of 17Hz
    ICM_write(0x00, 0x0A);    // set gyroscope sample rate for 100Hz
    ICM_write(0x14, 0x15);    // set accelerometer low pass filter to 136Hz and the
rate to 8G
    ICM_write(0x11, 0x0A);    // set accelerometer rate to 100hz
    ICM_write(0x7F, 0x00);    // select User Bank 0

          /*Initializing Magnetometer */
    ICM_write(0x03, 0x20);       // Enable I2C master
    ICM_write(0x7F, 0x30);       // Select User Bank 3
    ICM_write(0x01, 0x07);       // Set Master Clk speed to 400k
}
```

4. DEVELOPED CODE OF TASK 3

Insert only the modified sections here

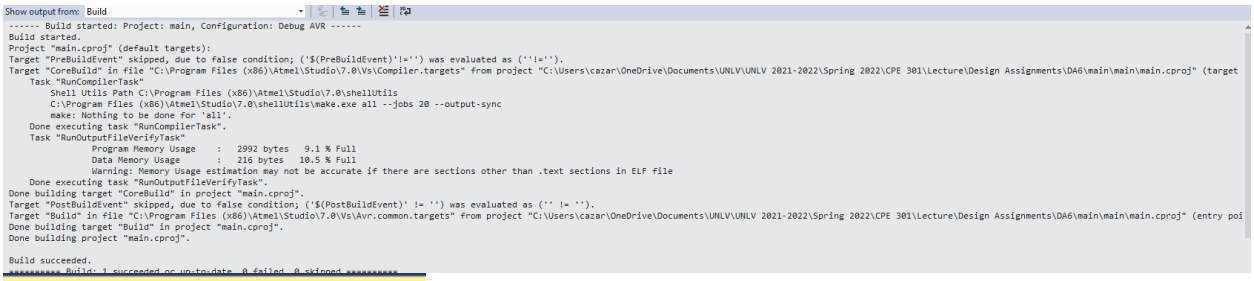
5. SCHEMATICS

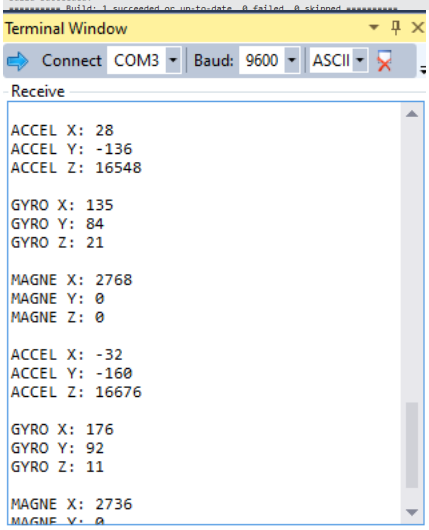

4
18
VCC
AVCC
20
AREF
PB0
PB1
PB2
PB3
PB4
PB5
XTAL1/PB6
XTAL2/PB7
12
13
14
15
16
17
7
8
PC0
PC1
PC2
PC3
PC4
PC5
RESET/PC6
23
24
25
26
27
28
29
3
6
19
22
PE0
PE1
PE2
PE3
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
30
31
32
1
2
9
10
11
GND
5
U?
ATmega328PB-A
8
13
VDDIO
VDD
9
24
23
22
SDO/ADO
SDA/SDI
SCL/SCLK
CS
AUX_DA
AUX_CL
21
7
11
12
FSYNC
INT1
REGOUT
10
GND
18
U?
ICM-20948

6. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

Task 1

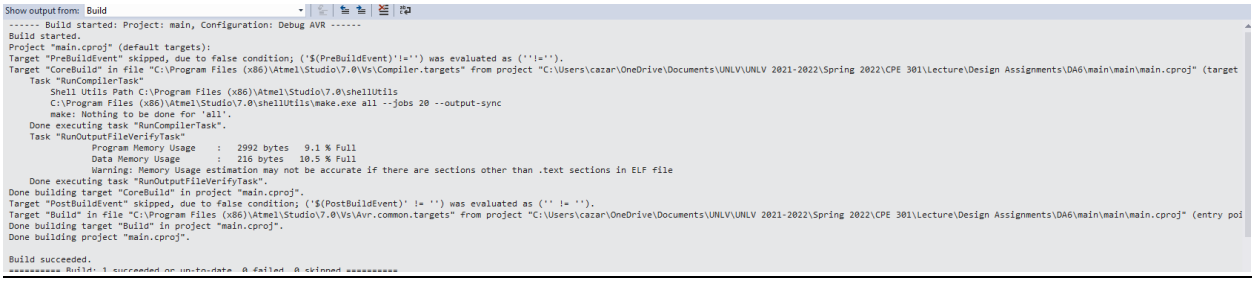
Successful Build



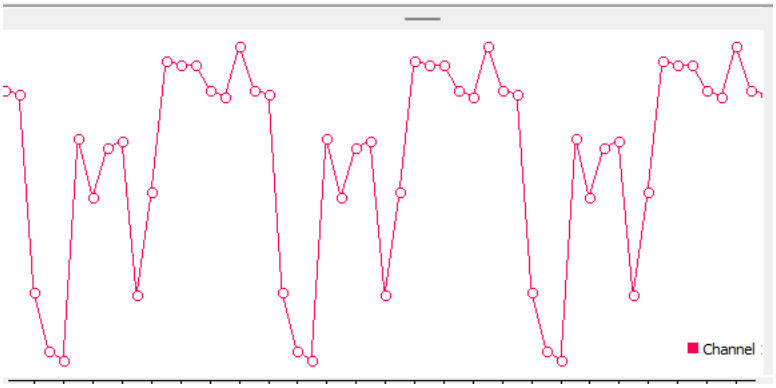


Task 2

Successful Build

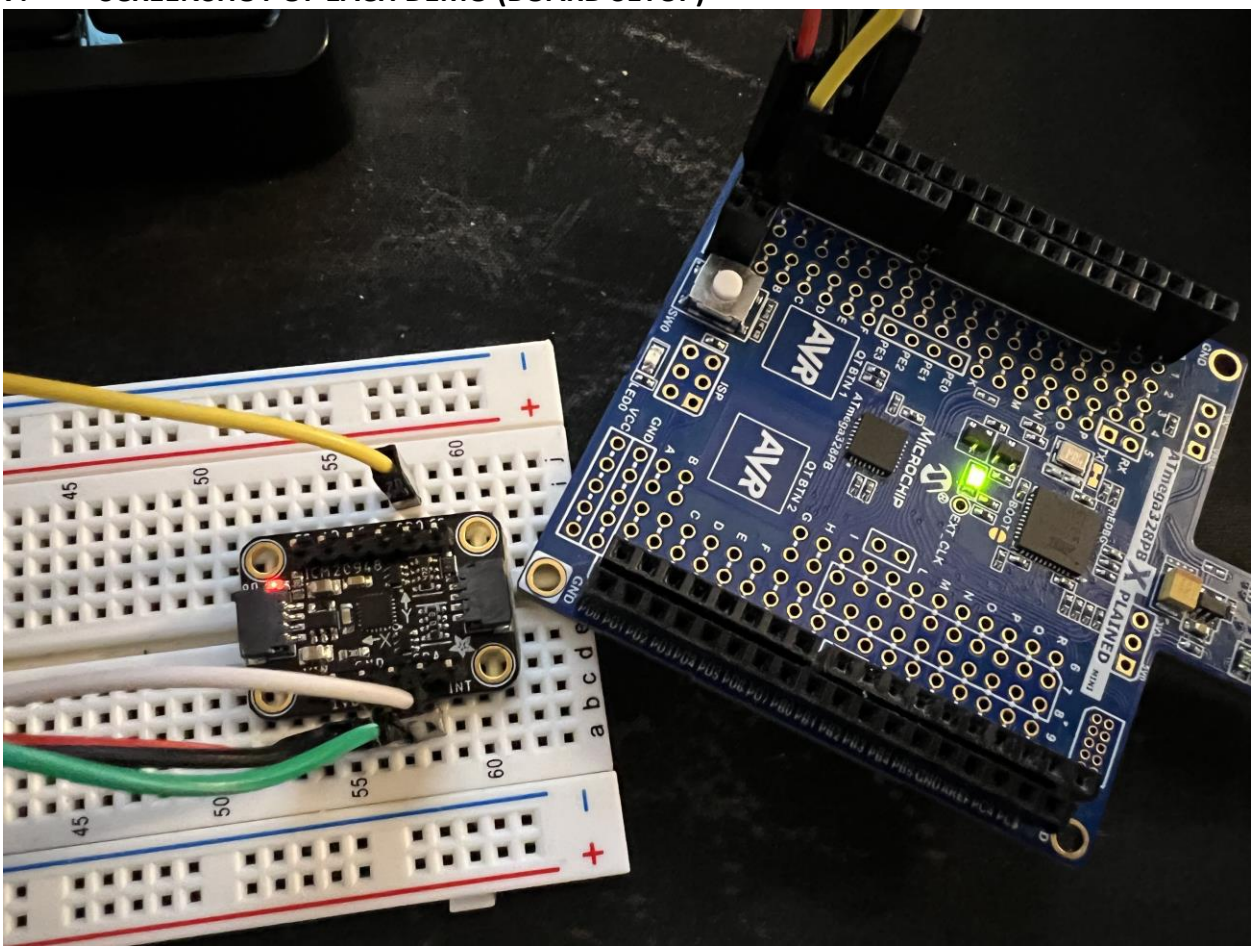


Serial Plot



Task 3

7. SCREENSHOT OF EACH DEMO (BOARD SETUP)



8. VIDEO LINKS OF EACH DEMO

Video Link for Task 1
Video Link for Task 2
Video Link for Task 3

9. GITHUB LINK OF THIS DA

https://github.com/DylanCaz/Submission_DA/tree/main/Design_Assignments_sub/DA_6_sub

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"This assignment submission is my own, original work". [redacted]