CPE301 – SPRING 2025

Design Assignment 7

Student Name: Ryan Sewell

Student #: 8000473785

Student Email: sewelr2@unlv.nevada.edu

Primary Github address: https://github.com/sewelr2

Directory: DA-Submissions/DA7

Video Playlist: <https://youtube.com/playlist?list=PLt45mEFhRV6ffOYRcGHhoI5aDeP3Zgqt5&feature=shared>

Objective I. I²C Interface & UART Streaming

Task: Read raw accel (X/Y/Z) and gyro (X/Y/Z) from the BMI160 over TWI (100 kHz) and transmit via UART.

Deliverables: Compact C code + terminal/Serial-Plotter outputs.

Objective II. Sensor Fusion

Task: Fuse accel and gyro with a complementary filter to compute roll, pitch, yaw at a fixed sample rate.

Deliverables: Filter code + three time-series plots (accel, gyro, angles).

Objective III. OLED Display

Task: Drive a 128×64 I²C OLED to show live roll, pitch, and yaw.

Deliverables: OLED-interface code + image of real-time angle display.

Objective IV. Schematics & Results

Task: Document wiring (MCU ↔ BMI160, MCU ↔ OLED, power/GND, pull-ups) and collate plots.

Deliverables: Annotated schematic(s) + consolidated plots with brief captions.

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

Microchip Studio Atmega328PB-Xmini PC Breadboard Shield Tauno Serial Plotter

- Assembler - BMI160 - USART

- Simulator - OLED Display

- Debugger

A computer circuit board with many letters and numbers

AI-generated content may be incorrect.

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

#define F\_CPU 16000000 //Change to UL if not working

#include <stdio.h>

#include <stdlib.h>

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define USART\_BAUDRATE 9600

#define UBRR\_VALUE (((F\_CPU / (USART\_BAUDRATE \* 16UL))) - 1)

#define SCL\_CLK 10000L/\* Define SCL clock frequency \*/

#define BMI160\_ADDR 0x69

#define BMI160\_CHIP\_ID 0x00

#define BMI160\_CMD\_REG 0x7E

#define BMI160\_ACC\_DATA 0x12

#define BMI160\_GYR\_DATA 0x0C

#define ACC\_SCALE 16384.0

#define GYRO\_SCALE 262.4

void UART\_init( void )

{

UBRR0H = (uint8\_t)(UBRR\_VALUE >> 8); //Enable baud rate to 9600

UBRR0L = (uint8\_t)UBRR\_VALUE; //Enable baud rate to 9600

UCSR0C = (1 << UCSZ01) | (1 << UCSZ00); //Data is 8 bit size, and there is 1 stop bit

UCSR0B = (1 << TXEN0); //Enable transmitter

}

//This function transmits the data, each char is entered into UDR0 register and tramsitted to the /\* serial monitor

void UART\_Transmit(char data){

while (!(UCSR0A & (1 << UDRE0))); // Cycles through each char until UDRE0 is 1

UDR0 = data;

}

//Sends data to UART\_Transmit

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

while (\*data != '\0') { //Cycles through each char in data

UART\_Transmit(\*data); //Sends

data++;

}

}

void i2c\_init() {

TWSR0 = 0x00;

TWBR0 = 0x48;

TWCR0 = (1 << TWEN);

}

void i2c\_start() {

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

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

}

void i2c\_stop() {

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

}

void i2c\_write(uint8\_t data) {

TWDR0 = data;

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

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

}

uint8\_t i2c\_read\_ack() {

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

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

return TWDR0;

}

uint8\_t i2c\_read\_nack() {

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

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

return TWDR0;

}

void bmi160\_write(uint8\_t reg, uint8\_t data) {

i2c\_start();

i2c\_write((BMI160\_ADDR << 1) | 0);

i2c\_write(reg);

i2c\_write(data);

i2c\_stop();

}

void bmi160\_read\_bytes(uint8\_t reg, uint8\_t \*buf, uint8\_t len) {

i2c\_start();

i2c\_write((BMI160\_ADDR << 1) | 0);

i2c\_write(reg);

\_delay\_us(10);

i2c\_start();

i2c\_write((BMI160\_ADDR << 1) | 1);

for (uint8\_t i = 0; i < len; i++) {

buf[i] = (i == len - 1) ? i2c\_read\_nack() : i2c\_read\_ack();

}

i2c\_stop();

}

void bmi160\_init() {

// Check Chip ID

i2c\_start();

i2c\_write((BMI160\_ADDR << 1) | 0);

i2c\_write(BMI160\_CHIP\_ID);

i2c\_start();

i2c\_write((BMI160\_ADDR << 1) | 1);

uint8\_t id = i2c\_read\_nack();

i2c\_stop();

if (id != 0xD1) return;

\_delay\_ms(100);

// Set accelerometer and gyroscope to normal mode

bmi160\_write(BMI160\_CMD\_REG, 0x11); // Accel Normal

\_delay\_ms(50);

bmi160\_write(BMI160\_CMD\_REG, 0x15); // Gyro Normal

\_delay\_ms(50);

}

void bmi160\_read\_accel(int16\_t \*x, int16\_t \*y, int16\_t \*z) {

uint8\_t data[6];

bmi160\_read\_bytes(BMI160\_ACC\_DATA, data, 6);

\*x = (int16\_t)((data[1] << 8) | data[0]);

\*y = (int16\_t)((data[3] << 8) | data[2]);

\*z = (int16\_t)((data[5] << 8) | data[4]);

}

void bmi160\_read\_gyro(int16\_t \*x, int16\_t \*y, int16\_t \*z) {

uint8\_t data[6];

bmi160\_read\_bytes(BMI160\_GYR\_DATA, data, 6);

\*x = (int16\_t)((data[1] << 8) | data[0]);

\*y = (int16\_t)((data[3] << 8) | data[2]);

\*z = (int16\_t)((data[5] << 8) | data[4]);

}

int main(void) {

i2c\_init();

UART\_init();

\_delay\_ms(1000);

bmi160\_init();

USART\_tx\_string("BMI160 Init");

int16\_t ax, ay, az, gx, gy, gz;

float fax, fay, faz, fgx, fgy, fgz;

char buffer[64];

while (1) {

bmi160\_read\_accel(&ax, &ay, &az);

bmi160\_read\_gyro(&gx, &gy, &gz);

fax = ax / ACC\_SCALE;

fay = ay / ACC\_SCALE;

faz = az / ACC\_SCALE;

fgx = gx / GYRO\_SCALE;

fgy = gy / GYRO\_SCALE;

fgz = gz / GYRO\_SCALE;

snprintf(buffer, sizeof(buffer), "ACC: %.2f, %.2f, %.2f, GYR: %.2f, %.2f, %.2f", fax, fay, faz, fgx, fgy, fgz);

USART\_tx\_string(buffer);

USART\_tx\_string("\n");

\_delay\_ms(100);

}

}

1. **DEVELOPED/MODIFIED CODE OF TASK 2/A from TASK 1/A**

#define F\_CPU 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <math.h>

#include <stdio.h>

#include <stdint.h>

#include "font.h" // your SSD1306 + font helper

/\* === UART === \*/

#define USART\_BAUDRATE 9600

#define UBRR\_VALUE (((F\_CPU/(USART\_BAUDRATE\*16UL)))-1)

void UART\_init(void) {

UBRR0H = (uint8\_t)(UBRR\_VALUE >> 8);

UBRR0L = (uint8\_t)UBRR\_VALUE;

UCSR0C = (1<<UCSZ01)|(1<<UCSZ00); // 8 data bits, 1 stop bit

UCSR0B = (1<<TXEN0); // enable transmitter

}

void UART\_tx(char c) {

while (!(UCSR0A & (1<<UDRE0)));

UDR0 = c;

}

void UART\_str(const char \*s) {

while (\*s) UART\_tx(\*s++);

}

/\* === I²C (TWI) === \*/

void i2c\_init(void) {

TWSR0 = 0x00; // prescaler = 1

TWBR0 = ((F\_CPU/100000UL) - 16) / 2; // SCL ≈ 100 kHz

TWCR0 = (1<<TWEN); // enable TWI

}

void i2c\_start(void) {

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

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

}

void i2c\_stop(void) {

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

\_delay\_us(10);

}

void i2c\_write(uint8\_t data) {

TWDR0 = data;

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

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

}

uint8\_t i2c\_read\_ack(void) {

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

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

return TWDR0;

}

uint8\_t i2c\_read\_nack(void) {

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

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

return TWDR0;

}

/\* === BMI160 === \*/

#define BMI160\_ADDR 0x68 // or 0x69 if your SDO pin is HIGH

#define BMI160\_CMD\_REG 0x7E

#define BMI160\_ACC\_REG 0x12

#define BMI160\_GYR\_REG 0x0C

void bmi160\_write(uint8\_t reg, uint8\_t val) {

i2c\_start();

i2c\_write((BMI160\_ADDR<<1)|0);

i2c\_write(reg);

i2c\_write(val);

i2c\_stop();

}

void bmi160\_read(uint8\_t reg, uint8\_t \*buf, uint8\_t len) {

i2c\_start();

i2c\_write((BMI160\_ADDR<<1)|0);

i2c\_write(reg);

\_delay\_us(10);

i2c\_start();

i2c\_write((BMI160\_ADDR<<1)|1);

for (uint8\_t i = 0; i < len-1; i++) {

buf[i] = i2c\_read\_ack();

}

buf[len-1] = i2c\_read\_nack();

i2c\_stop();

}

void bmi160\_init(void) {

// Soft-reset

bmi160\_write(BMI160\_CMD\_REG, 0xB6);

\_delay\_ms(100);

// Normal accel mode

bmi160\_write(BMI160\_CMD\_REG, 0x11);

\_delay\_ms(50);

// Normal gyro mode

bmi160\_write(BMI160\_CMD\_REG, 0x15);

\_delay\_ms(50);

}

void bmi160\_accel(int16\_t \*x, int16\_t \*y, int16\_t \*z) {

uint8\_t d[6];

bmi160\_read(BMI160\_ACC\_REG, d, 6);

\*x = (int16\_t)((d[1]<<8) | d[0]);

\*y = (int16\_t)((d[3]<<8) | d[2]);

\*z = (int16\_t)((d[5]<<8) | d[4]);

}

void bmi160\_gyro(int16\_t \*x, int16\_t \*y, int16\_t \*z) {

uint8\_t d[6];

bmi160\_read(BMI160\_GYR\_REG, d, 6);

\*x = (int16\_t)((d[1]<<8) | d[0]);

\*y = (int16\_t)((d[3]<<8) | d[2]);

\*z = (int16\_t)((d[5]<<8) | d[4]);

}

/\* === OLED via font.h === \*/

void oled\_init(void) {

ssd1306\_init();

ssd1306\_clear();

}

void oled\_display\_angles(float roll, float pitch, float yaw) {

char buf[32];

ssd1306\_clear();

// Line 0: Roll

ssd1306\_set\_cursor(0, 0);

snprintf(buf, sizeof(buf), "Roll: %+5.1f", roll);

ssd1306\_write\_string(buf);

// Line 1: Pitch

ssd1306\_set\_cursor(0, 1);

snprintf(buf, sizeof(buf), "Pitch:%+5.1f", pitch);

ssd1306\_write\_string(buf);

// Line 2: Yaw

ssd1306\_set\_cursor(0, 2);

snprintf(buf, sizeof(buf), "Yaw: %+5.1f", yaw);

ssd1306\_write\_string(buf);

ssd1306\_update();

}

/\* === Main Loop === \*/

int main(void) {

// Initialize peripherals

i2c\_init();

UART\_init();

oled\_init();

\_delay\_ms(100);

bmi160\_init();

UART\_str("BMI160 + OLED Ready\r\n");

// Sensor and filter variables

int16\_t ax, ay, az, gx, gy, gz;

float f\_ax, f\_ay, f\_az;

float roll = 0, pitch = 0, yaw = 0;

const float dt = 0.1f; // 100 ms loop

const float R2D = 57.2957795f; // rad → deg

char uart\_buf[32];

while (1) {

// 1) Read raw sensor data

bmi160\_accel(&ax, &ay, &az);

bmi160\_gyro (&gx, &gy, &gz);

// 2) Convert and compute angles

f\_ax = ax/16384.0f; f\_ay = ay/16384.0f; f\_az = az/16384.0f;

roll = atan2f( f\_ay, f\_az) \* R2D;

pitch = atan2f(-f\_ax, sqrtf(f\_ay\*f\_ay + f\_az\*f\_az)) \* R2D;

yaw += (gz/262.4f) \* dt;

if (yaw > 180) yaw -= 360;

if (yaw < -180) yaw += 360;

// 3) UART output

snprintf(uart\_buf, sizeof(uart\_buf),

"R:%+5.1f P:%+5.1f Y:%+5.1f\r\n",

roll, pitch, yaw);

UART\_str(uart\_buf);

// 4) OLED update

oled\_display\_angles(roll, pitch, yaw);

\_delay\_ms(100);

}

return 0;

}

1. **SCHEMATICS**

A diagram of a rectangular object with a circuit board and a red and blue text

AI-generated content may be incorrect.

1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

A screen shot of a computer

AI-generated content may be incorrect.

1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**
2. **VIDEO LINKS OF EACH DEMO**

Task 1:

Task 2:

…

1. **GITHUB LINK OF THIS DA**

**Student Academic Misconduct Policy**

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

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

Ryan Sewell