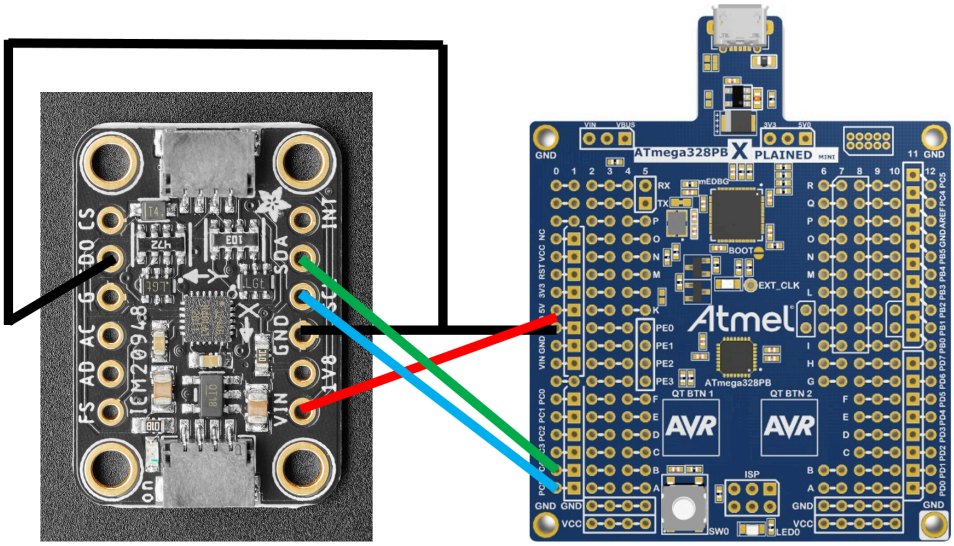
**CPE301 – SPRING 2022**

# Design Assignment 6

Student Name: David Nakasone
Student #: 2001646072
Student Email: nakasd3@unlv.nevada.edu
Primary Github address: https://github.com/davenakasone
Directory: https://github.com/davenakasone/cpe301_David_Nakasone

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

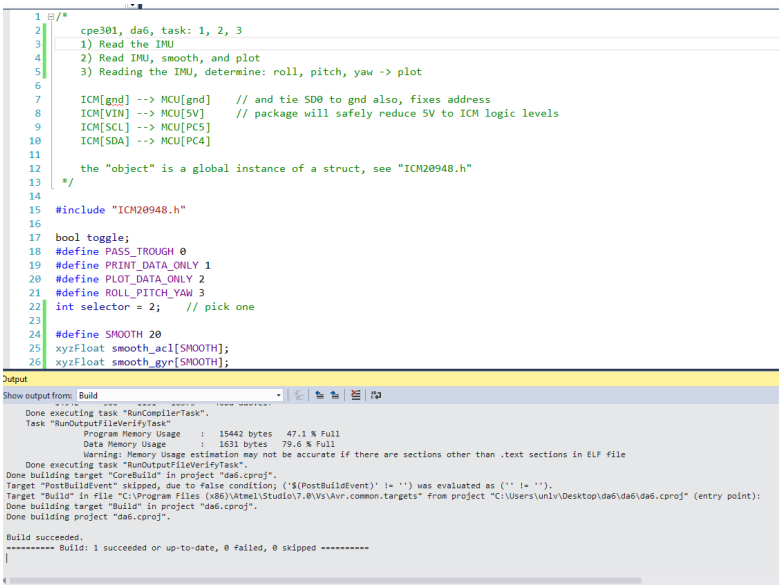
The ICM20948 has a built-in magnetometer.
Otherwise, just ground the address, 5V power, common ground, SCL, and SDA:




The sensor and MCU are shown without soldered header pins for simplicity.

## 2. INITIAL/MODIFIED/DEVELOPED CODE, all tasks

This program builds successfully:




C code:

```
/*
    cpe301, da6, task: 1, 2, 3
    1) Read the IMU
    2) Read IMU, smooth, and plot
    3) Reading the IMU, determine: roll, pitch, yaw -> plot

    ICM[gnd] --> MCU[gnd]    // and tie SD0 to gnd also, fixes address
    ICM[VIN] --> MCU[5V]     // package will safely reduce 5V to ICM logic levels
    ICM[SCL] --> MCU[PC5]
    ICM[SDA] --> MCU[PC4]

    the "object" is a global instance of a struct, see "ICM20948.h"
 */

#include "ICM20948.h"

bool toggle;
#define PASS_TROUGH 0
#define PRINT_DATA_ONLY 1
```

```
#define PLOT_DATA_ONLY 2
#define ROLL_PITCH_YAW 3
int selector = 2;    // pick one

#define SMOOTH 20
xyzFloat smooth_acl[SMOOTH];
xyzFloat smooth_gyr[SMOOTH];
xyzFloat smooth_mag[SMOOTH];

void obj_init();
void print_data_max();
void send_9();
void send_rpy();


int main(void)
{
    toggle = true;
    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");
    _delay_ms(PAUSE_RESET_MS);

    // initialize the "object" , calibrates, whoAmI checks
    obj_init();

    while(1)
    {
        hold :
        if (toggle == false) {goto hold;}

        switch (selector)
        {
            case (PRINT_DATA_ONLY) :
                print_data_max();
```

```
                _delay_ms(1000);
                break;
            case (PLOT_DATA_ONLY) :
                send_9();
                break;
            case (ROLL_PITCH_YAW) :
                send_rpy();
                break;
            default :
                toggle = false;
                break;
        }
    }
}


////~~~~


void obj_init()
{
    _delay_ms(1000);
    c_init();                                                   // the old constructor
and initializer
    c_initMagnetometer();                                       // old magnetometer
initializer
    //c_setAccOffsets(-16330.0, 16450.0, -16600.0, 16180.0, -16520.0, 16690.0);    // calibrates
    _delay_ms(1000);                                            // wait to compare new
measurement
    c_autoOffsets();                                            // user holds keeps
device flat and still
    c_setAccRange(ICM20948_ACC_RANGE_2G);                       // the most reliable,
also default
    c_setAccDLPF(ICM20948_DLPF_6);                              // moderate noise
filter
    c_setGyrDLPF(ICM20948_DLPF_6);                              // same filter for
```

```
gyro
    c_setTempDLPF(ICM20948_DLPF_6);                                          // same filter for
temperature
    c_setMagOpMode(AK09916_CONT_MODE_100HZ);                                 // set the
magnetometer
    _delay_ms(1000);
    USART_tx_string("all initialization and calibration is complete\r\n");       // device is ready,
automatic averaging
    USART_tx_string("\r\n");
}


////~~~~


void print_data_max()
{
    c_readSensor();
    vars.acl = c_getGValues();
    vars.gyr = c_getGyrValues();
    vars.mag = c_getMagValues();
    vars.celsius = c_getTemperature();
    vars.g_result = c_getResultantG(vars.acl);
    c_getOrientationAsString();
    snprintf(vars.helper, BUF_SIZ-1, "acceleration (g)  [x, y, z]  :  %0.3f , %0.3f , %0.3f  ,
resultant:  %0.3f m/s^2\r\n",
        vars.acl.x, vars.acl.y, vars.acl.z, 9.8*vars.g_result);
    USART_tx_string(vars.helper);
    snprintf(vars.helper, BUF_SIZ-1, "gyroscope (deg)   [x, y, z]  :  %0.3f , %0.3f , %0.3f\r\n",
vars.gyr.x, vars.gyr.y, vars.gyr.z);
    USART_tx_string(vars.helper);
    snprintf(vars.helper, BUF_SIZ-1, "magnetometer (uT) [x, y, z]  :  %0.3f , %0.3f , %0.3f\r\n",
vars.mag.x, vars.mag.y, vars.mag.z);
    USART_tx_string(vars.helper);
    snprintf(vars.helper, BUF_SIZ-1, "temperature (celsius)        :  %0.3f\r\n", vars.celsius);
    USART_tx_string(vars.helper);
```

```
    snprintf(vars.helper, BUF_SIZ-1, "orientation                 :  %s\r\n", vars.str_orientation);
    USART_tx_string(vars.helper);
    USART_tx_string("\r\n");
}



////~~~



void send_9()
{
    vars.acl.x = 0;
    vars.acl.y = 0;
    vars.acl.z = 0;
    vars.gyr.x = 0;
    vars.gyr.y = 0;
    vars.gyr.z = 0;
    vars.mag.x = 0;
    vars.mag.y = 0;
    vars.mag.z = 0;
    for (int ii = 0; ii < SMOOTH; ii++)
    {
        c_readSensor();
        smooth_acl[ii] = c_getGValues();
        smooth_gyr[ii] = c_getGyrValues();
        smooth_mag[ii] = c_getMagValues();
        vars.acl.x = vars.acl.x + smooth_acl[ii].x;
        vars.acl.y = vars.acl.y + smooth_acl[ii].y;
        vars.acl.z = vars.acl.z + smooth_acl[ii].z;
        vars.gyr.x = vars.gyr.x + smooth_gyr[ii].x;
        vars.gyr.y = vars.gyr.y + smooth_gyr[ii].y;
        vars.gyr.z = vars.gyr.z + smooth_gyr[ii].z;
        vars.mag.x = vars.mag.x + smooth_mag[ii].x;
        vars.mag.y = vars.mag.y + smooth_mag[ii].y;
        vars.mag.z = vars.mag.z + smooth_mag[ii].z;
    }
```

```
    vars.acl.x = vars.acl.x / SMOOTH;
    vars.acl.y = vars.acl.y / SMOOTH;
    vars.acl.z = vars.acl.z / SMOOTH;
    vars.gyr.x = vars.gyr.x / SMOOTH;
    vars.gyr.y = vars.gyr.y / SMOOTH;
    vars.gyr.z = vars.gyr.z / SMOOTH;
    vars.mag.x = vars.mag.x / SMOOTH;
    vars.mag.y = vars.mag.y / SMOOTH;
    vars.mag.z = vars.mag.z / SMOOTH;

    snprintf(vars.helper, BUF_SIZ-1, "%f,%f,%f,%f,%f,%f,%f,%f,%f\r\n",
        vars.acl.x, vars.acl.y, vars.acl.z,
        vars.gyr.x, vars.gyr.y, vars.gyr.z,
        vars.mag.x, vars.mag.y, vars.mag.z);
    USART_tx_string(vars.helper);
}


////~~~~


void send_rpy()
{
    float catch_roll[SMOOTH];
    float catch_pitch[SMOOTH];
    float catch_yaw[SMOOTH];
    float h_x;
    float h_y;
    float sum_roll = 0;
    float sum_pitch = 0;
    float sum_yaw = 0;

    for (int ii = 0; ii < SMOOTH; ii++)
    {
        c_readSensor();
        catch_roll[ii] = c_getRoll();
```

```
        catch_pitch[ii] = c_getPitch();
        smooth_mag[ii] = c_getMagValues();

        h_y = (smooth_mag[ii].y * cos(catch_roll[ii]) -
            smooth_mag[ii].z * sin(catch_roll[ii]));

        h_x = (smooth_mag[ii].x * cos(catch_pitch[ii]) +
            smooth_mag[ii].y * sin(catch_roll[ii]) * sin(catch_pitch[ii]) +
            smooth_mag[ii].z * cos(catch_roll[ii]) * sin(catch_pitch[ii]));

        catch_yaw[ii] = 57.3 * atan2(h_y, h_x);
        sum_roll = sum_roll + catch_roll[ii];
        sum_pitch = sum_pitch + catch_pitch[ii];
        sum_yaw = sum_yaw + catch_yaw[ii];
    }
    vars.roll = sum_roll / SMOOTH;
    vars.pitch = sum_pitch / SMOOTH;
    vars.yaw = sum_yaw / SMOOTH;

    snprintf(vars.helper, BUF_SIZ-1, "%f,%f,%f\r\n",
    vars.roll, vars.pitch, vars.yaw);
    USART_tx_string(vars.helper);
}


////////~~~~~~~~END>  main.c
```

See the github link for all source files

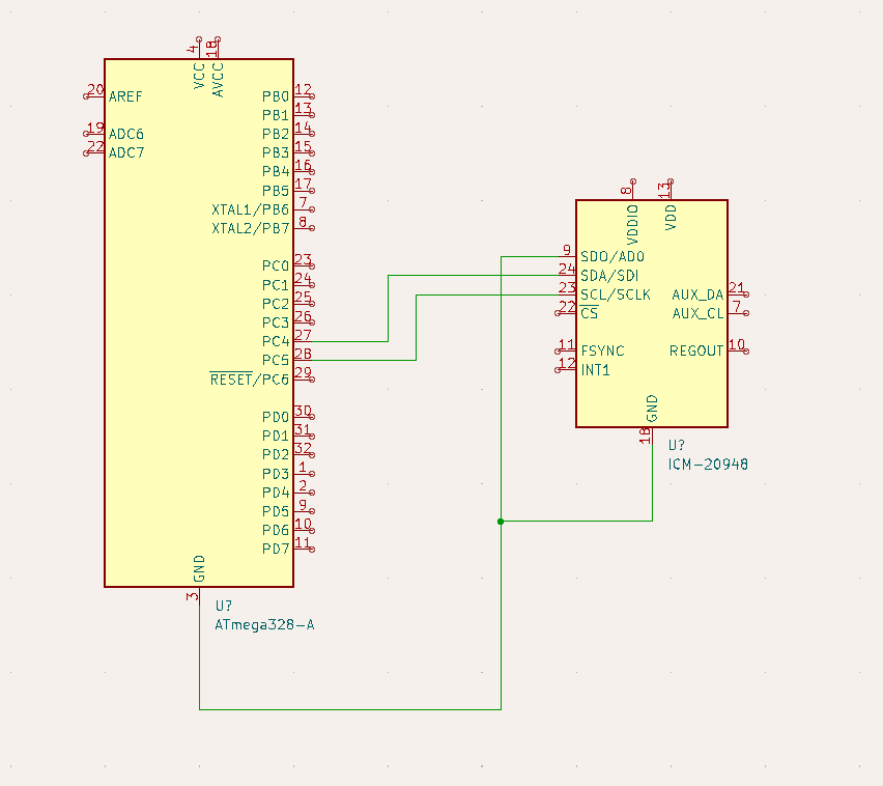
Additional helping functions:

```
uint8_t c_readRegister8(uint8_t bank, uint8_t reg)
{
    #ifdef DEBUG_CLI
    snprintf(vars.helper, BUF_SIZ-1, "<%d> %s/%s()\r\n", __LINE__, __FILE__, __func__);
    USART_tx_string(vars.helper);
    #endif
    c_switchBank(bank);
    uint8_t regValue = 0;
    i2c_start((vars.i2cAddress << 1) + I2C_WRITE);    //_wire->beginTransmission(vars.i2cAddress);
    i2c_write(reg);                                   //_wire->write(reg);
    i2c_start((vars.i2cAddress << 1) + I2C_READ);     // read mode
    regValue = i2c_readNak();                         // stop condition is sent
    return regValue;
}
```

Additional helping functions:

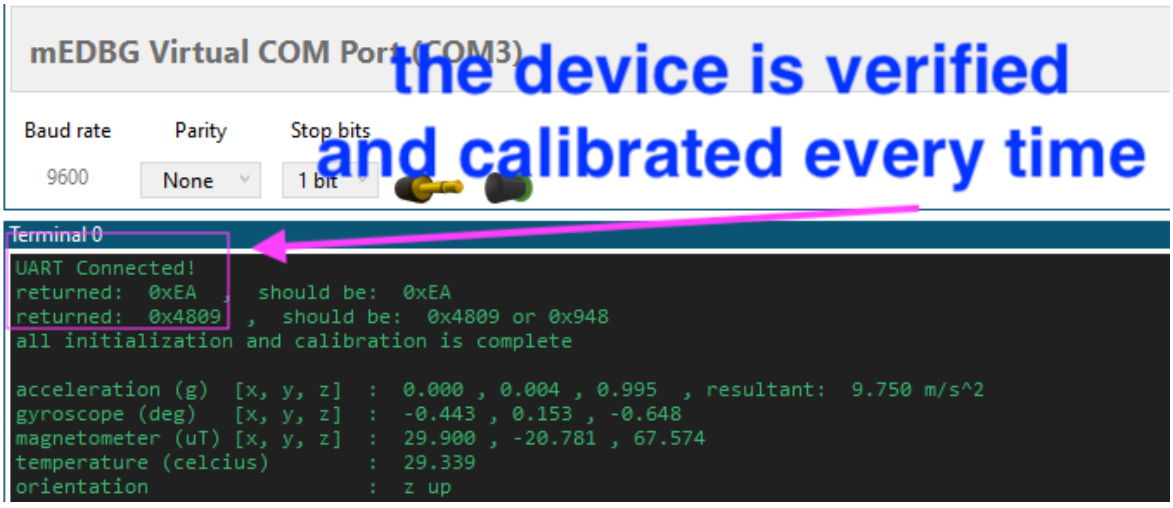
```
float c_getPitch()
{
    xyzFloat angleVal = c_getAngles();
    float pitch = (atan2(angleVal.x, sqrt(abs((angleVal.x*angleVal.y +
angleVal.z*angleVal.z))))*180.0)/M_PI;
    return pitch;
}



////~~~~



float c_getRoll()
{
    xyzFloat angleVal = c_getAngles();
    float roll = (atan2(angleVal.y, angleVal.z)*180.0)/M_PI;
    return roll;
}
```

## 3. SCHEMATICS

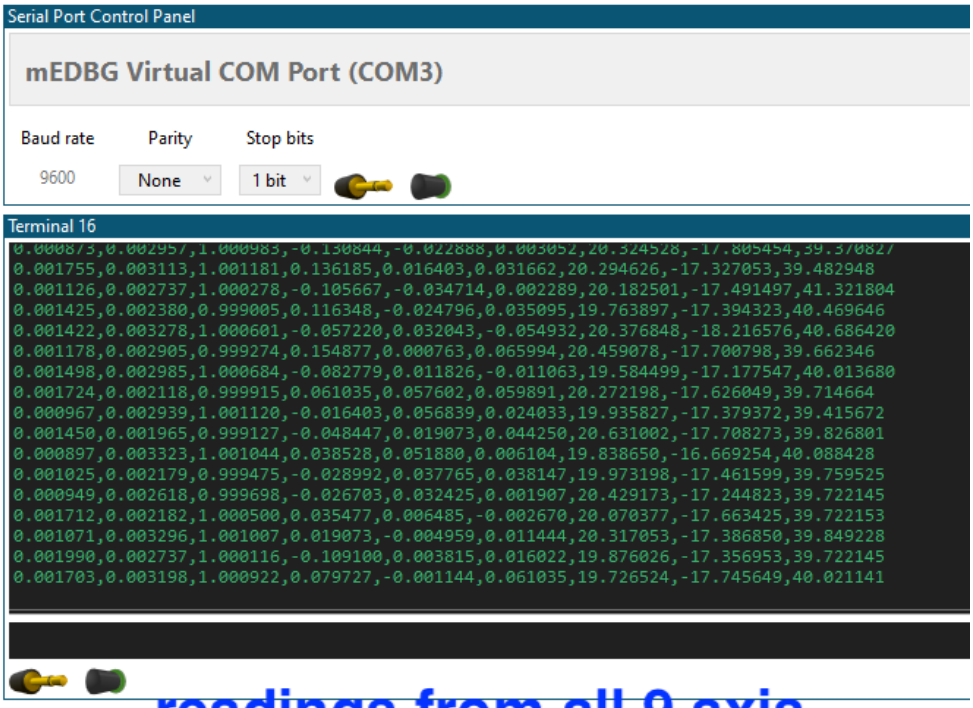



## 4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

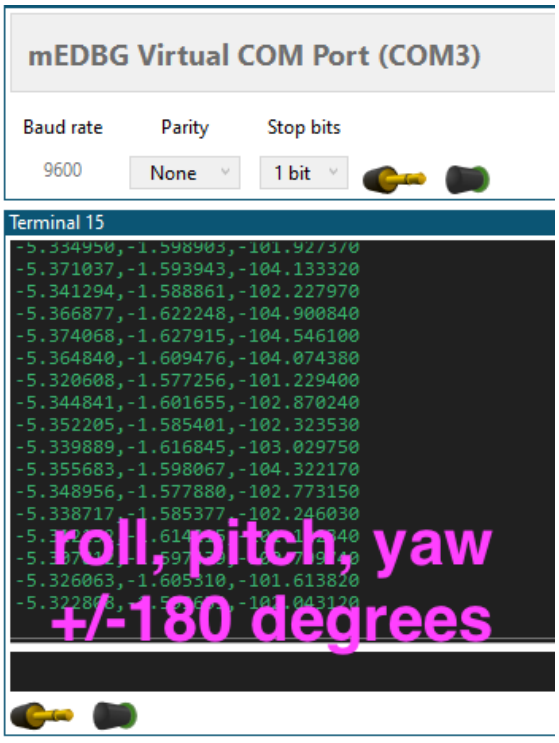
Task 1:



Task 2:

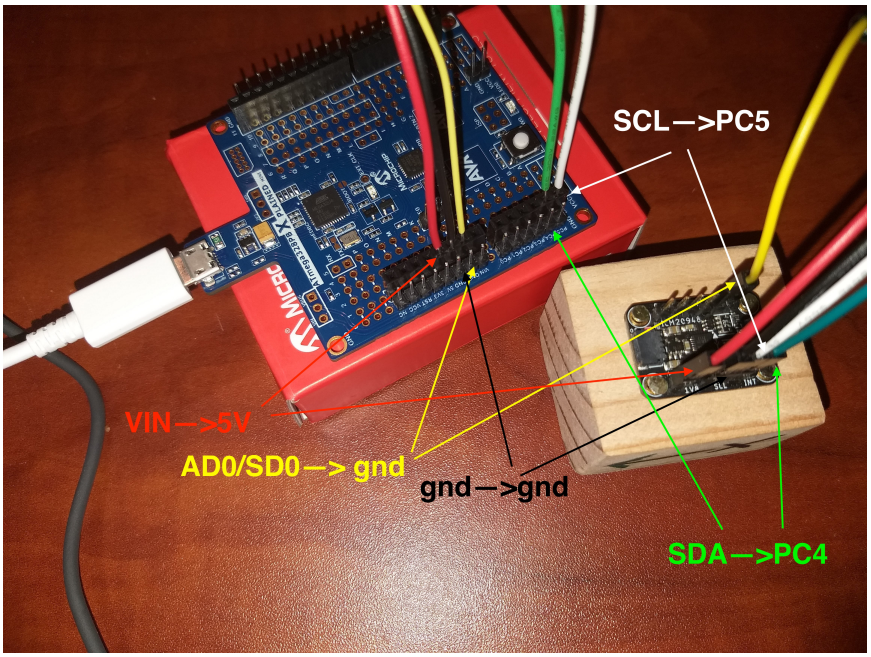

Serial Port Control Panel
mEDBG Virtual COM Port (COM3)
Baud rate
Parity
Stop bits
9600
None
1 bit
Terminal 16
0.000873,0.002957,1.000983,-0.130844,-0.022888,0.003052,20.324528,-17.805454,39.370827
0.001755,0.003113,1.001181,0.136185,0.016403,0.031662,20.294626,-17.327053,39.482948
0.001126,0.002737,1.000278,-0.105667,-0.034714,0.002289,20.182501,-17.491497,41.321804
0.001425,0.002380,0.999005,0.116348,-0.024796,0.035095,19.763897,-17.394323,40.469646
0.001422,0.003278,1.000601,-0.057220,0.032043,-0.054932,20.376848,-18.216576,40.686420
0.001178,0.002905,0.999274,0.154877,0.000763,0.065994,20.459078,-17.700798,39.662346
0.001498,0.002985,1.000684,-0.082779,0.011826,-0.011063,19.584499,-17.177547,40.013680
0.001724,0.002118,0.999915,0.061035,0.057602,0.059891,20.272198,-17.626049,39.714664
0.000967,0.002939,1.001120,-0.016403,0.056839,0.024033,19.935827,-17.379372,39.415672
0.001450,0.001965,0.999127,-0.048447,0.019073,0.044250,20.631002,-17.708273,39.826801
0.000897,0.003323,1.001044,0.038528,0.051880,0.006104,19.838650,-16.669254,40.088428
0.001025,0.002179,0.999475,-0.028992,0.037765,0.038147,19.973198,-17.461599,39.759525
0.000949,0.002618,0.999698,-0.026703,0.032425,0.001907,20.429173,-17.244823,39.722145
0.001712,0.002182,1.000500,0.035477,0.006485,-0.002670,20.070377,-17.663425,39.722153
0.001071,0.003296,1.001007,0.019073,-0.004959,0.011444,20.317053,-17.386850,39.849228
0.001990,0.002737,1.000116,-0.109100,0.003815,0.016022,19.876026,-17.356953,39.722145
0.001703,0.003198,1.000922,0.079727,-0.001144,0.061035,19.726524,-17.745649,40.021141
readings from all 9 axis

Task 3:

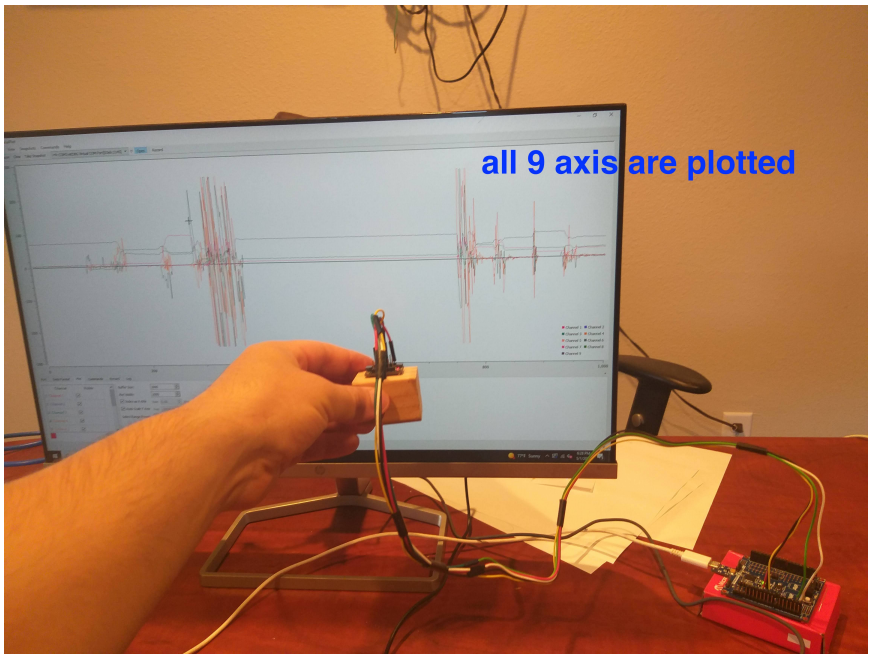

mEDBG Virtual COM Port (COM3)
Baud rate
Parity
Stop bits
9600
None
1 bit
Terminal 15
-5.334950,-1.598903,-101.927370
-5.371037,-1.593943,-104.133320
-5.341294,-1.588861,-102.227970
-5.366877,-1.622248,-104.900840
-5.374068,-1.627915,-104.546100
-5.364840,-1.609476,-104.074380
-5.320608,-1.577256,-101.229400
-5.344841,-1.601655,-102.870240
-5.352205,-1.585401,-102.323530
-5.339889,-1.616845,-103.029750
-5.355683,-1.598067,-104.322170
-5.348956,-1.577880,-102.773150
-5.338717,-1.585377,-102.246030
-5.326063,-1.605310,-101.613820
roll, pitch, yaw
+/-180 degrees

## 5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

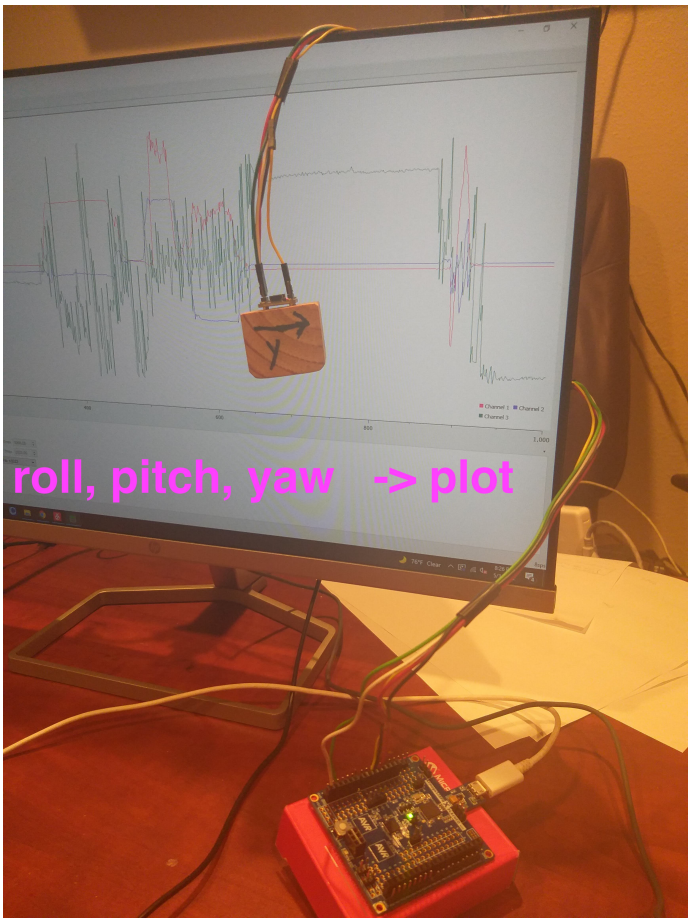
No configuration changes occur, all tasks use the same setup.




Task 2:



Task 3:




## 6. VIDEO LINKS OF EACH DEMO

task 1, print to UART: https://youtu.be/so3LgRhPMWA

task2, filter and plot: https://youtu.be/02JxrRWnF2k

task3, calculate roll, pitch, yaw --> plot: https://youtu.be/0dkAURsSJgw

## 7. GITHUB LINK OF THIS DA

https://github.com/davenakasone/cpe301_David_Nakasone/tree/main/Design_Assignmentz/DA6

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

"*This assignment submission is my own, original work*".
David Nakasone