Exercise Manual For SC2104/CE3002 Sensors, Interfacing and Digital Control

Practical Exercise #2: Operating Characteristics of IMU's Accelerator and Gyroscope

Venue: SCSE Labs

COMPUTER ENGINEERING COURSE

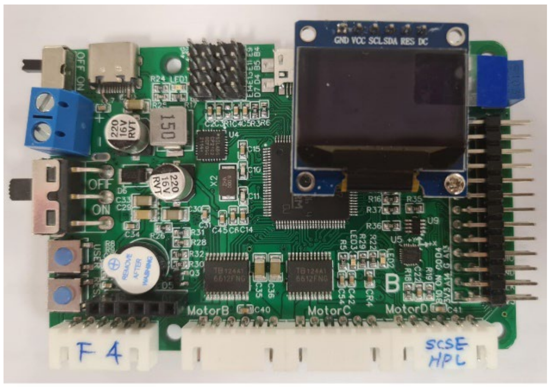
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

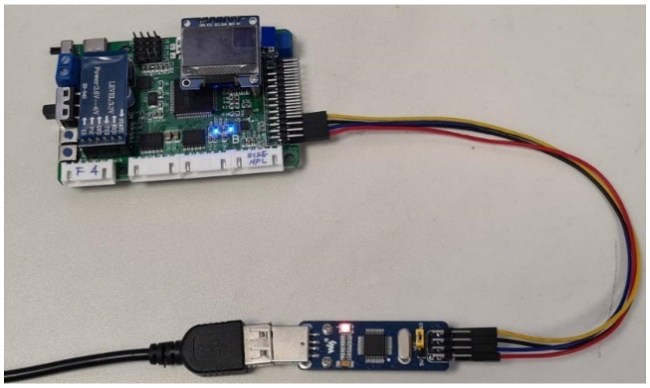
Learning Objectives

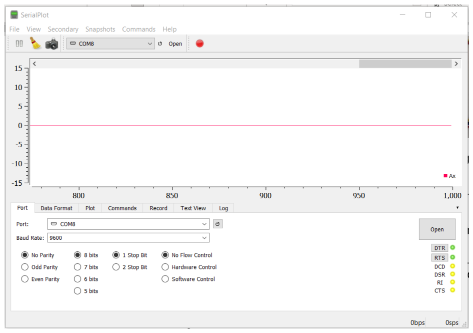
Continue from Exercise #1, you will now develop programs to access the on-board IMU's Accelerator and Gyroscope devices on the STM32F4 platform, and to examine and understand the characteristics of these devices during operations.

Equipment and accessories required

i) One desktop computer installed with STM32CubeIDE, PuTTY and SerialPlot software.
ii) One STM32F4 board
iii) ONE ST-LINK SWD board


STM32F4 board


ST-Link for downloading and debugging code


SerialPlot
File View Secondary Snapshots Commands Help
Open
Ax
Port Data Format Plot Commands Record Text View Log
Open
DTR RTS DCD DSR RI CTS
0bps 0sps

Port: COM8
Baud Rate: 9600
[✓ No Parity] Odd Parity Even Parity
[✓ 8 bits] 7 bits 6 bits 5 bits
[✓ 1 Stop Bit] 2 Stop Bit
[✓ No Flow Control] Hardware Control Software Control

SerialPlot's UI

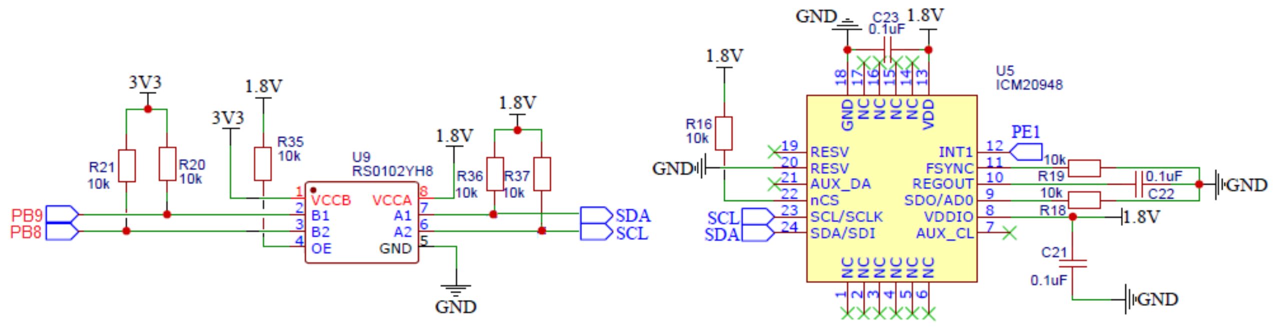
Introduction

Typical IMU contains devices such as Accelerator, Gyroscope and Magnetometer that are used widely in many equipment and machines, especially those that are designed to operate autonomously such as drones and vehicles for navigation purpose. However, to use these devices effectively, it is importance to first understand the characteristics and limitation of these devices.

In the following exercises, you will first learn how to access an IMU that are connected to the microcontroller through an I^{2}C interface. You will then examine the devices' real time operating characteristics, by collecting and sending their measurements through the STM32F4's serial port and display them on the computer using the "SerialPort" software.

1. I²C interface of the IMU

The STM32F4 board contains a 9-axis IMU device, ICM20948, which is connected to the microcontroller based on the I²C interfaceas shown in the following schematic diagram (The interface is connect through a 2-bit bidirectional voltage translator IC, RS01012YH8 – google this device to find out the reason for using it in this design).



As such, you will need to first configure the appropriate microcontroller's I/Os to function as the I²C interface pins in order to access the IMU.

1.1. Configure the I²C interface STM32CubeIDE

Study the schematic diagram above and configure the appropriate I/O pins for the IMU interface using the STM32CubeIDE's GUI based configuration feature. (Hint: you will find that the relevant I/O pins to be used have built-in support for I²C functions.)

If this is done correctly, you would notice that a function `MX_I2C1_Init();` is generated and added into the main.c file. You will also notice the following declarations been generated and stored in the file (which you need to use when calling the IMU related functions.)

```
/* Private variables ---------------------------------------------*/
I2C_HandleTypeDef  hi2c1;   // this is for the I²C
UART_HandleTypeDef huart3;  // this is for ??
```

1.2. IMU Driver code

The code that are used to access and to obtain the IMU's measurements are available in NTULearn course-site, which consists of two files: `"IMU.c"` and its header file `"ICM20948.h"`. The following are brief descriptions of some relevant information in these two files that can be used to operate the IMU.

(i) You will need to declare a variable with the structure type **`ICM20948`**; E.g., `ICM20948 imu;`
Declaration of the `ICM20948` structure is in the `"ICM20948.h"` file which is as shown below:

```
typedef struct {
    I2C_HandleTypeDef *i2cHandle;
    UART_HandleTypeDef *uart;
    float acc[3];   // store the 3 axes Accelerator measured values
    float gyro[3];  // store the 3 axes Gyroscope measured values
    float temp_C;   // store the IMU on-chip temperature sensor output value
    } ICM20948;
```

(ii) The following function is to be called to initialize the IMU device before its operation.

```
uint8_t* IMU_Initialise(ICM20948 *dev, I2C_HandleTypeDef *i2cHandle, UART_HandleTypeDef *uart)
```

E.g., `uint8_t status = IMU_Initialise(&imu, &hi2c1, &huart3);`

The return value `status` is a code that indicates the execution status of the function (0 = no error). (You can study the code of the function in the `"IMU.c"` file to find out all the possible return values).

(iii) The function `IMU_AccelRead(ICM20948 *dev)` can be used to read the accelerator values.

(iv) The function `IMU_GyroRead(ICM20948 *dev)` can be used to read the gyroscope values

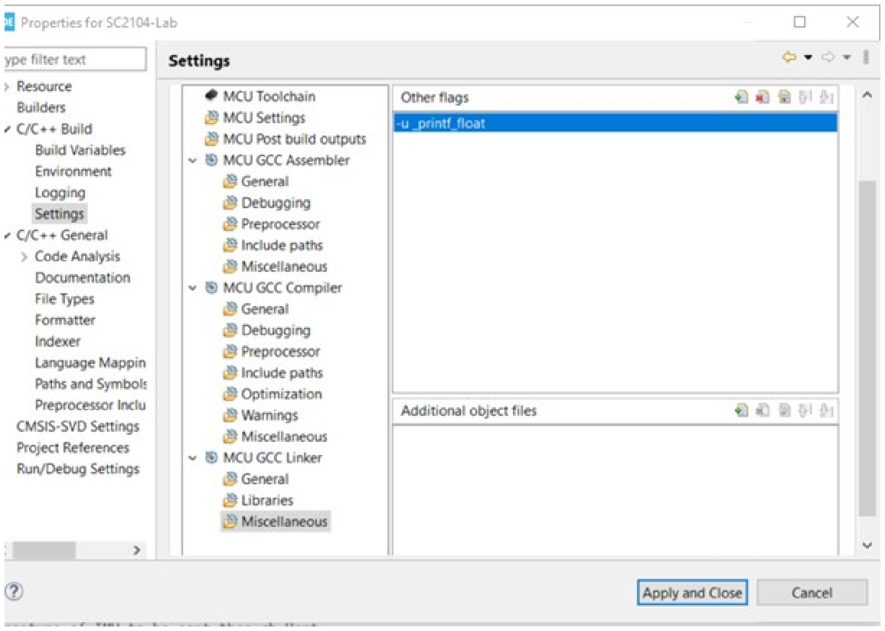
1.3. Other relevant information

The UART function that is used in Exercise #1, `HAL_UART_Transmit()` can only transmit unsigned 8-bit integer data type. On the other hand, the data values from the IMU are of the **float** data type. As such, you will need to convert the float data type to uint8_t (i.e., char), before calling the UART function.

One way to do so is to use the sprintf() function, such as the following:

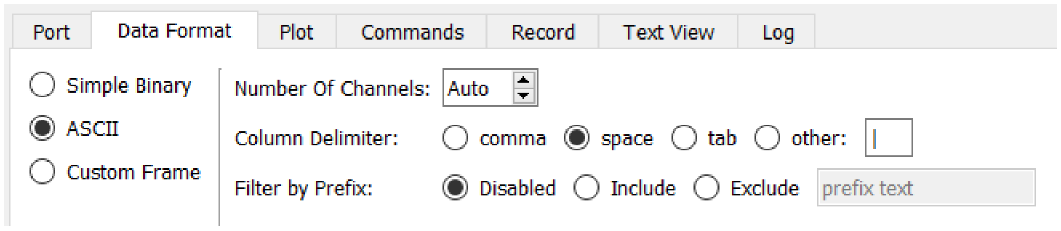
```
sprintf(sbuf[i], "%5.2f ", imu.acc[i]);
HAL_UART_Transmit(&huart3, sbuf[i], 6, HAL_MAX_DELAY);
```

However, the floating-point support for printf is not enabled by default in STM32CubeIDE. To do so, we need to add the flag `-u _printf_float` to its linker option. (-u is to force linking and loading of library module that is undefined, in this case, `_printf_float`)



Other flags
-u _printf_float

The corresponding program that is used to display the data (sent by the microcontroller) on the computer is the SerialPort, which need to be configured with the following settings.



Simple Binary [✓ ASCII] Custom Frame

Number Of Channels: Auto

Column Delimiter: comma [✓ space] tab other:

Filter by Prefix: [✓ Disabled] Include Exclude prefix text

2. Practical Exercises

Based on the above information, you will now develop the code to perform the various operations as described below.

2.1. Configure the I^{2}C interface

Configure the I^{2}C interface for the microcontroller to access the IMU (as described in 1.1 above).

2.2. Accelerator

(a) Code the routine to access the IMU's Accelerator and transmit its values to the SerialPort program and display them in real time on the computer.

Check that your code is working correctly - by rotating the STM32F4 board in different directions and observe the changes in the output values (as displayed in the SerialPlot program).

(b) Derive the Pitch and Roll angles by using the appropriate data from the accelerator output (as discussed in the lecture). Transmit the values to the SerialPort program for display. Ascertain that the program is behaving accordingly when you pitch and roll the STM32F4 board at different angles.

(c) Observe the change in the displayed values when you move and shake the board. Add appropriate code to your program to reduce/overcome the effect observed.

2.2. Gyroscope

Repeat the above procedure for the Gyroscope.

3. Summary of observations

Compare and comment on the accuracy and behavior of the Pitch and Roll angles derived from the Accelerator and Gyroscope respectively.