

# PCI-3 [Week 5] : Sensor & Device Interfacing Using GPIO & I2C

## Resources

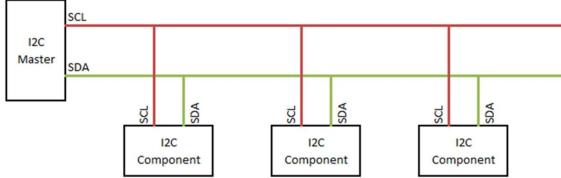
- [Demo code: Blink by HAL](https://canvas.nus.edu.sg/courses/85090/files/8295827?wrap=1) (<https://canvas.nus.edu.sg/courses/85090/files/8295827?wrap=1>)

## Objectives

- To be familiar with using the HAL and BSP libraries to configure pins, initialise, and read various sensors.
- To be able to configure and use GPIO input and output.
- To read the data from the various sensors on the board through polling.
- To print messages using UART.

## 1. Background: I<sup>2</sup>C (Inter-Integrated Circuit) Devices

I<sup>2</sup>C is a serial communication protocol, so data is transferred bit by bit through a single wire - SDA (Serial Data). It is synchronised with the timing information on another wire - SCL (Serial Clock). SCL and SDA are bidirectional open-drain lines with pull-up resistors.



This is a diagram showing SCL and SDA lines connection between PC Master and Slave devices.

## 2. Understand the Project Libraries

- Analyse the existing main.c code (Core) as well as the CMSIS, HAL, and BSP libraries to understand the various definitions and functions.
  - Identify the location of each function (Core / CMSIS / HAL / BSP). You can do this by pressing F3 when the cursor is on the function, which will open the file where the function is defined.
  - Explore the parameters passed to each function.
  - Go into the function and see how it is implemented. Go deeper into sub-functions to get a sense of the function hierarchy. This can be challenging when function pointers are used (which is the case with many functions in HAL / BSP), but running in debug mode and stepping into the function will help.
  - Look at the various definitions and macros used and passed to the functions, such as the association between pin names and pin numbers.
  - You can learn a lot by running the program in debug mode, setting a breakpoint at the function, running until there, and stepping into it.
  - You can also refer to the HAL and BSP manuals, which can be found in the software section of Datasheets and Downloads, though this is not absolutely necessary.
  - How do you see the raw value of x-, y-, and z-axes readings given by the accelerometer (the BSP function gives you a processed form)?

## 3. Exercise 1: Interfacing to peripherals using the HAL library

1. Run [Blink\\_by\\_HAL.zip](https://canvas.nus.edu.sg/courses/85090/files/8295827?wrap=1) (<https://canvas.nus.edu.sg/courses/85090/files/8295827?wrap=1>)

2. Observe how LED toggling is achieved using the HAL library.

## 4. Exercise 2: Interfacing to peripherals using the BSP library

1. Go to Drivers > BSP library:

- The BSP library contains functions to access and control peripherals and devices more conveniently.
- There are two levels of interface
  - Device level
  - Component level (Part No.)
- If any components do not have a specific library file named "stm32475e\_iot01\_SensorName.c", look into "stm32475e\_iot01.c"

2. Relate the function `BSP_LED_Init()` to `MX_GPIO_Init()`. Use the BSP library, refer to `stm32475e_iot01.c` and write code to toggle the LED every 1 second.

## 5. Getting to Know Various Sensors

In this lab, you will learn how to use the I<sup>2</sup>C protocol for the interfacing between I<sup>2</sup>C devices and the STM32L4S5. In Lab2.zip, you used two I<sup>2</sup>C sensors - a temperature sensor and an accelerometer.

- Identify other I<sup>2</sup>C devices and their locations on STM32L4S5.
- Which I<sup>2</sup>C interface on STM32L4S5 is used by those sensors for data transfer? I<sup>2</sup>C1 / I<sup>2</sup>C2 ? Which STM32L4S5 pin(s) are used for the I<sup>2</sup>C interface? Are some of the STM32L4S5 pins used by these sensors the same? Why?
- To find more information about the sensors, you can Google and search for the datasheet of the specific I<sup>2</sup>C sensors. It is good to understand the detailed features and specifications of the devices that you are going to use.

Many of the sensors on STM32L4S5 can be found on your smartphones too.

- Try installing Sensor Multitool or a similar application on your phone.
- Once you open the app, you will be able to see the various sensors available on your phone.
- Compare the names/IDs of sensors on your phone and the ones on STM32L4S5. What do you observe?
- Try to open one sensor at a time in the app and observe the change in their readings.
- Why does the magnetometer change value as you rotate the phone? What does the change in reading signify? What does the gyroscope measure?

## 6. Exercise 3: UART Configuration

UART is a much faster way to transmit and receive messages as compared to printf(). The recommended terminal program is [Tera Term](https://github.com/TeraTermProject/teraterm/releases/tag/v5.3) (<https://github.com/TeraTermProject/teraterm/releases/tag/v5.3>). A more powerful alternative is [RealTerm](https://realterm.mechzoo.com/) (<https://realterm.mechzoo.com/>), but it has a lot more options which a beginner will not need. Other options include PuTTY (multi-platform), GTKTerm (Linux), etc. Select the corresponding port (COM port in Windows, ttyXXX in MacOS/ Linux) after opening Tera Term. Do not forget to verify the UART configuration.

### UART on Windows

The UART port may be determined through the computer's device manager by looking for the USB Serial Port as shown in Figure 1 below. If you are unsure, see which device under Ports get removed when you unplug.



The settings for the Tera Term program are as depicted in Figure 2 and Figure 3. UART port setup in Tera Term is available from the menu : Setup -> Serial.



### UART on Mac OS

For Mac user, open the system terminal and find the device with the following command:

```
ls /dev/tty.*
```

An example output will be:



Identify the board connection tty name and use the following command to establish a serial port connection:

```
sudo chmod 600 /dev/tty.*
```

You should replace the `/dev/tty.*` with the device path you identified earlier.

This connection may behave slightly differently from Tera Term on a Windows PC, but it is good enough for you to debug and test out the code.

## 5. Reading More Sensors Using the BSP Library (Optional, Self-study)

Now, we will read the magnetometer on the STM32L4S5 using I<sup>2</sup>C protocol with the help of the BSP driver library.

- Open the Driver > BSP > BL475E-IOT01 > stm32475e\_iot01\_magno.c in the project explorer.
- How many functions do you see? What do they do? (Note: Functions in the libraries are usually explained above them with brief comments)
- Which functions will you be using to initialize the magnetometer and read its data? What are the required input/output types of those functions?
- To use Magnetometer, we need to do three things in main.c - (1) include header, (2) Sensor Initialization, and (3) Sensor reading. We did these three for the accelerometer in the previous lab. Do the same for the Magnetometer.
- What are the parameters to be passed to the function that reads the magnetometer? How will you retrieve the magnetic field values in the X, Y, and Z directions?
- What is the unit of the magnetometer reading? (Note: read from the magnetometer sensor datasheet)
- How is the magnitude of that you obtain related to the raw value returned by the magnetometer? You might wish to go through slides 19-21 of Topic 8A.
- Debug the code (use single-stepping, breakpoints, etc.) and observe the output. How do you interpret the values? Are you receiving the correct magnetic field values? Try to rotate/move the board and observe the change in the output. Why/how does the value change?
- You can now read the rest of the I<sup>2</sup>C devices such as the gyroscope and the pressure sensor using the BSP library file. Be able to answer the above questions in the context of these devices as well.