Report

Homework #2 Socket Programming and Data Visualization

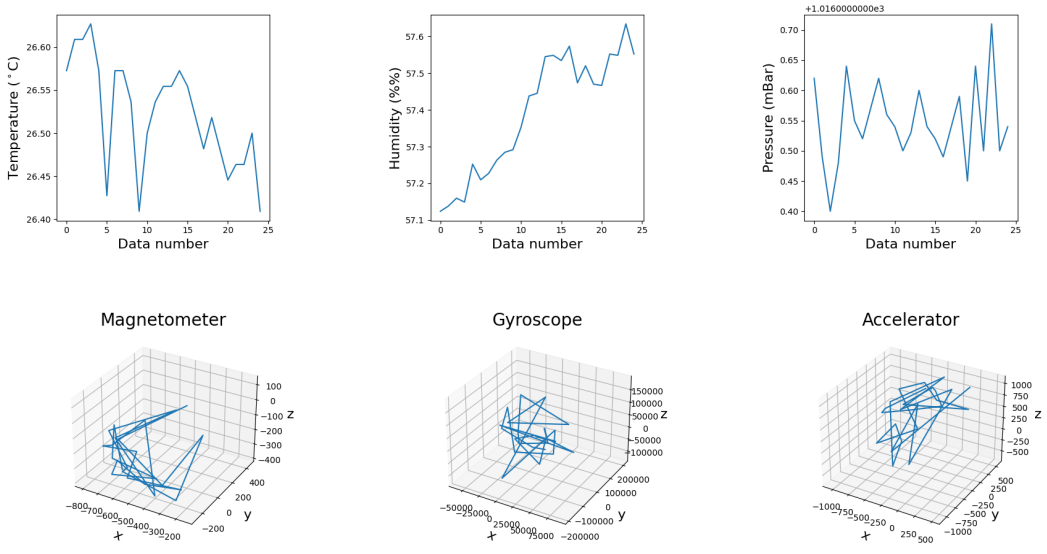
In this homework, we use the B-L4S5I-IOT01A discovery board as an IOT node sensor device. With BSP driver libraries, we could retrieve the sensor values and transmit to the host server via TCP sockets. The data is then being visualized using plotting tools. This project is built in Mbed OS environment.

Through the lab in the lectures, I managed to get the data from the sensors and print it in the terminal. Also with the help of TAs and putting in a lot of problem solving effort, I could finally connect the system board onto WiFi and connect to a host via self-resolving host..

However, I faced a problem that was by the time remained unsolved. I could not connect to the host server that should supposedly be the computer that we are using. The socket program written in python works perfectly with P2P connection, however it could not be connected by the board. I encountered several returning error codes from the Mbed OS TCP socket API, which is unclear and not documented. Even though I even tried to disable the default firewalls on Windows, nothing progressive happened.

After spending lots of time debugging, I finally found the solution to the problem. The network interface of the board is not defined in the configurations, and the variables for the interface should be declared in the [‘*’] part where it would be executed regardless of the type of the operating board.

With the connection problem solved, finally I could retrieve the data and successfully visualize the sensor values by using python module *matplotlib*.



The figures above show the visualization of sensor values.

(Github repo: https://github.com/dWKZ5jSZ8F/ESLab-2022-Fall)

Questions

1. **What are the units used for the sensor values?**

The model we are using here is **B-L4S5I-IOT01A** and the sensors integrated are 3D gyroscope and accelerometer **LSM6DSL**, temperature and relative humidity sensor **HTS221**, magnetic sensor **LIS3MDL**, nano-pressure sensor **LPS22HB** and ToF sensor **VL53L0X**.

The units used are listed as below:

LSM6DSL: Input signals ranging ±2/±4/±8/±16 g full scale and ±125/±250/±500/±1000/±2000 dps full scale for accelerometer and gyroscope respectively.

HTS221: 16-bit output data with range of rH sensitivity: 0.004% rH/LSB; Humidity accuracy: ± 3.5% rH, +20 to +80% rH; Temperature accuracy: ± 0.5 °C, +15 to +40 °C.

LIS3MDL: Full scales of ±4/±8/±12/±16 gauss and 16-bit output data.

LPS22HB: Absolute pressure range: 260 to 1260 hPa, 24-bit pressure output data and 16-bit temperature output data.

VL53L0X: Absolute measure range up to 2m with 940 nm laser VCSEL.

2. **What is I2C read address and I2C write address allocated for the LSM6DSL 3D accelerometer and 3D gyroscope sensors in the IoT node (B-L475E-IOT01A or B-L4S5I-IOT01A)?**

By inspecting the manual, the I2C read address and write address are 0xD4 and 0xD5 respectively.

6.13 I²C addresses of modules used on MB1297

Table 5 displays the I²C read and write addresses for the modules that are connected to the I2C2 bus.

Table 5. I²C addresses for each module

Modules	Description	SAD[6:0] + R/W	I2C write address	I2C read address
HTS221	Capacitive digital sensor for relative humidity and temperature	1011111x	0xBE	0xBF
LIS3MDL	3-axis magnetometer	0011110x	0x3C	0x3D
LPS22HB	MEMS nano pressure sensor	1011101x	0xBA	0xBB
LSM6DSL	3D accelerometer and 3D gyroscope	1101010x	0xD4	0xD5

3. What are the main differences I2C between SMbus (System Management Bus)?

I2C protocol is dedicated for low bandwidth infrequent communication needs between multiple types of devices and widely used in embedded systems. SMBus (System Management Bus) is based on the I2C protocol, however some of the protocols introduced are semantically beyond the original I2C protocol. Despite the difference, most I2C devices would still work on an SMBus and mainly modern PC mainboards rely on SMBus. One of the examples is RAM modules configured using I2C EEPROMs and hardware monitoring chips connecting through SMBus protocol.

Ref: https://www.kernel.org/doc/html/latest/i2c/summary.html

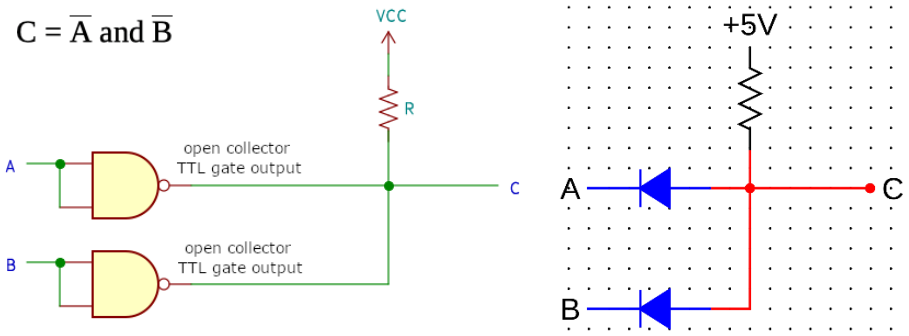
4. What is the I2C address of ADXL 345, if ALT ADDRESS is connected to HIGH?

With the **ALT ADDRESS** pin **HIGH**, the 7-bit I2C address for the device is 0x1D, followed by the R/W bit. This translates to 0x3A for a write and 0x3B for a read. An alternate I2C address of 0x53 (followed by the R/W bit) can be chosen by grounding the SDO/ALT ADDRESS pin (Pin 12). This translates to 0xA6 for a write and 0xA7 for a read.

Ref: https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf

5. How to connect two signal lines to achieve the wired-AND logic?

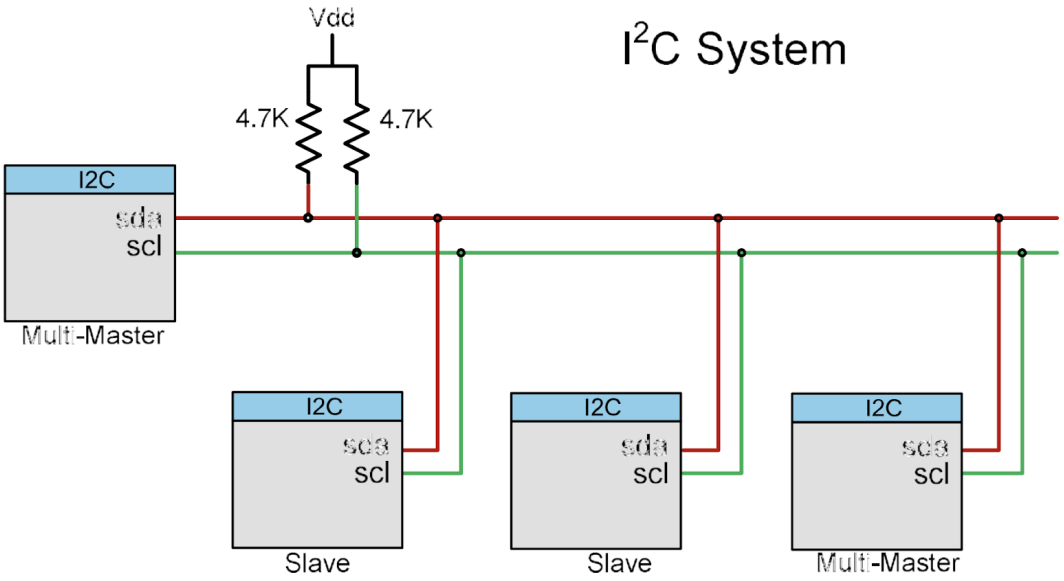
In order to achieve wired-AND logic, we need a pull up resistor with the help of one diode per input. For input sources A and B, the positive voltage from the output C “propagates” toward input sources. When the positive voltage is equal to or greater than all input sources, the source voltage is directed to the output. In this sense the input can be extended and conjunctive logic could be exhibited. The figure below shows how it works.



Ref: https://en.wikipedia.org/wiki/Wired_logic_connection

6. **What is the main difference between the bus master and the bus slave?**

In a system network of multiple devices, it can be designed with a combination of masters and slaves. A bus master is the program that is usually in a processing unit or I/O controller. It is used to direct traffic on the computer bus or I/O paths. In this case, the I/O devices are then so called "slaves" as their data flows are well regulated and controlled by the "master". Once the mechanism is being set up, the data bits flow directly and organized between I/O devices and processors.



Ref:
https://www.infineon.com/dgdl/Infineon-Component_I2C_V3.0-Software%20Module%20Datasheets-v03_05-EN.pdf?fileId=8ac78c8c7d0d8da4017d0e952b3f1fbe