ELEC5514 Lab1: Serial Communication between Board and PC

During the lab period of this semester, the students will get access to the state-of-the-art IoT technology and the related hardware. The whole content is divided into three parts:

The first one is BLE, which is short for Bluetooth low energy version. The other one is Lora-WAN, which is short for long range wide area network. At the first two parts, the students will study the protocol of BLE and Lora-WAN through some programming tasks. At the third part, every group needs to complete one or more practical application projects based on these two technologies.

At the beginning, we will arrange some quick start labs:

Outline (5 sections):

1. Introduce Bluetooth (general back ground, stack architecture).
2. Introduce the hardware (STM32 Nucleo pack and BLE, Bluetooth low energy, expansion board).
3. Get familiar with the working environment (Keil uVision5 and Putty) and run the Bluetooth Demo.
4. Modify the demo script (Rename the Bluetooth name and IP address).
5. Using Putty Console to realize the communication between the STM32 Nucleo pack and the PC.

Section 1

Introduction of Bluetooth low energy technology

Bluetooth low energy (BLE) wireless technology has been developed by the Bluetooth. Special Interest Group (SIG) in order to achieve a very low power standard operating with a coin cell battery for several years. Classic Bluetooth technology was developed as a wireless standard allowing replacing cables connecting portable and/or fixed electronic devices, but it cannot achieve an extreme level of battery life because of its fast hopping, connection-oriented behaviour, and relatively complex connection procedures. Bluetooth low energy devices consume only a fraction of the power of standard Bluetooth products and enable devices with coin cell batteries to be wirelessly connected to standard Bluetooth enabled devices.

(The detail related to the BLE protocol stack, please follow the document “BLE_stacks programming guidelines”. This document explains the whole structure of the BLE standard ***pp7 - 31***. Please read it.)

Section 2

Introduction of the STM32 Nucleo pack

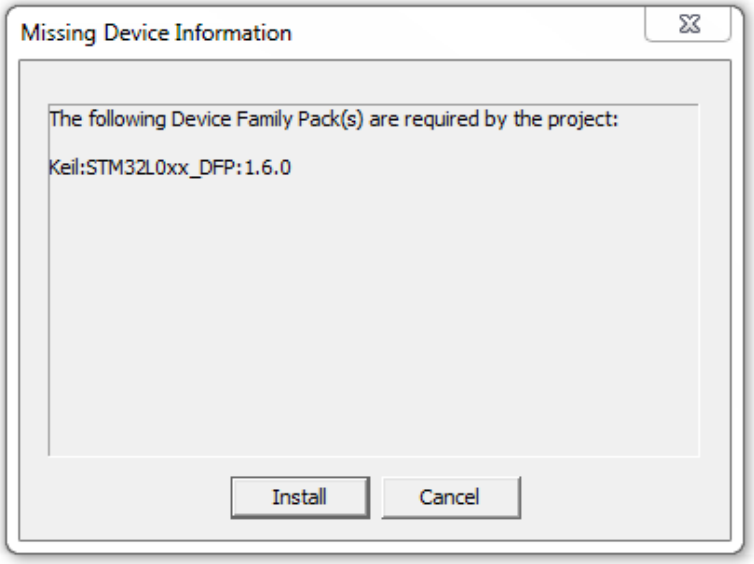
The evaluation board we used is embedded with the STM32L073R8 MCU provided by ST. This kind of MCU mainly focuses on ultralow power application. (Please follow the attached files, “STM32_MCU_datasheet”, “STM32_nucleo_board_user_manual”).

Introduction of the Bluetooth expansion board (X-NUCLEO-IDB05A1). This expansion board is designed to achieve BLE application. (Please follow the attached file "BLE_IDB05A1_datasheet").

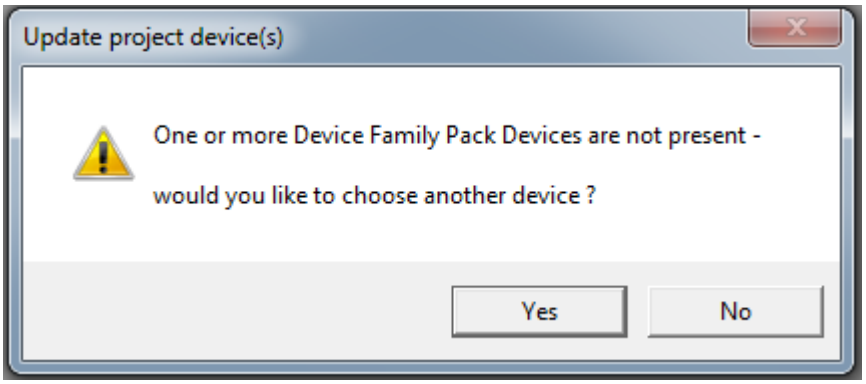
Section 3

Initialize the development environment

1. Connect the board with the PC.
2. Please download the "Lab1. zip" folder into the desktop. Open the folder: *SourceCode>Lab1_Demo_Code>Projects>Multi>Applications>SensorDemo>MDK-ARM>STEM32L053R8-Nucleo>SensorDemoProject*
3. Open the Sensor Demo Project via the **Keil uVision5**.

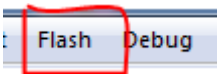


When you see this figure, click "Install". (If this window does not show, it is fine and feels free to go ahead.)

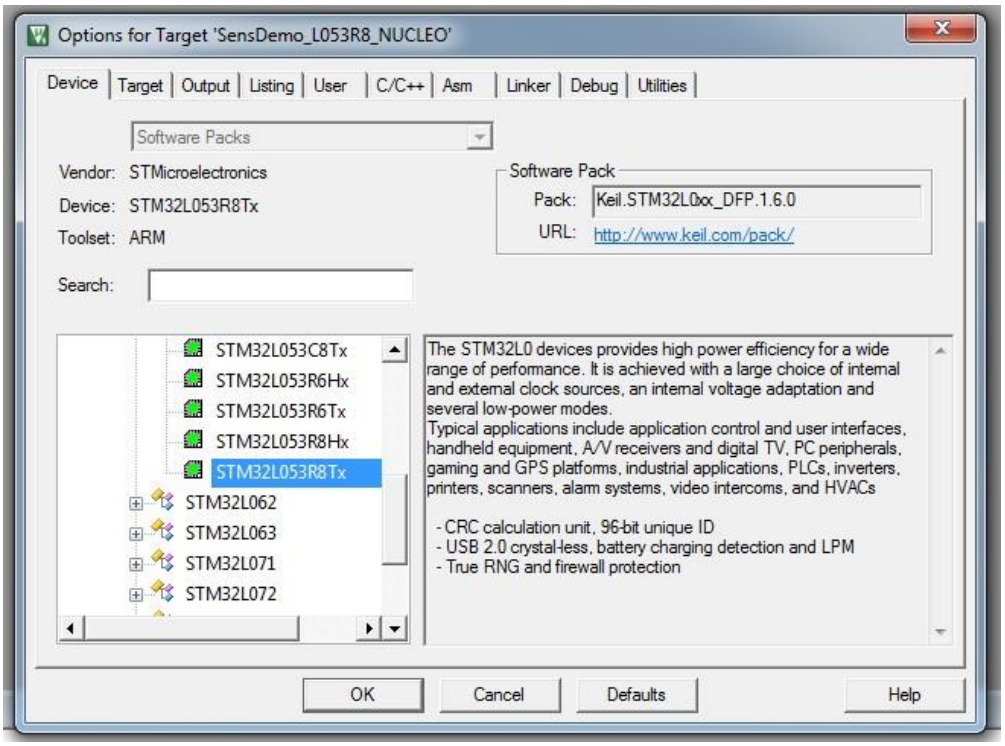


Click "No".

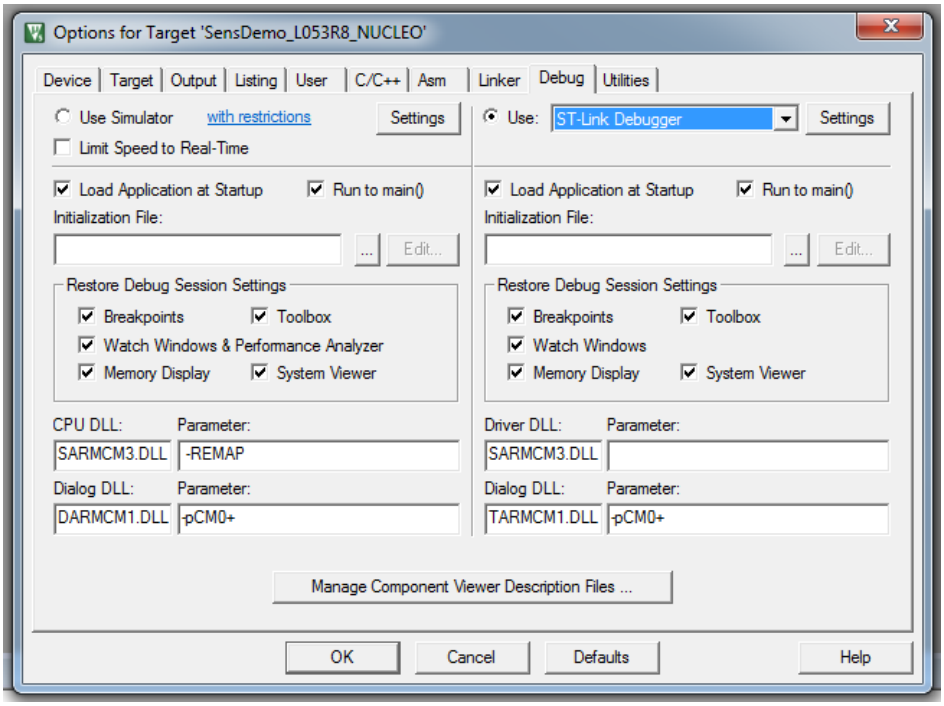
4. Now, you can try to build the project first to make sure the file is correct. Click "F7" or click the button on the top left of the window.
5. Configure the STEM32 Nucleo pack and the Bluetooth board.



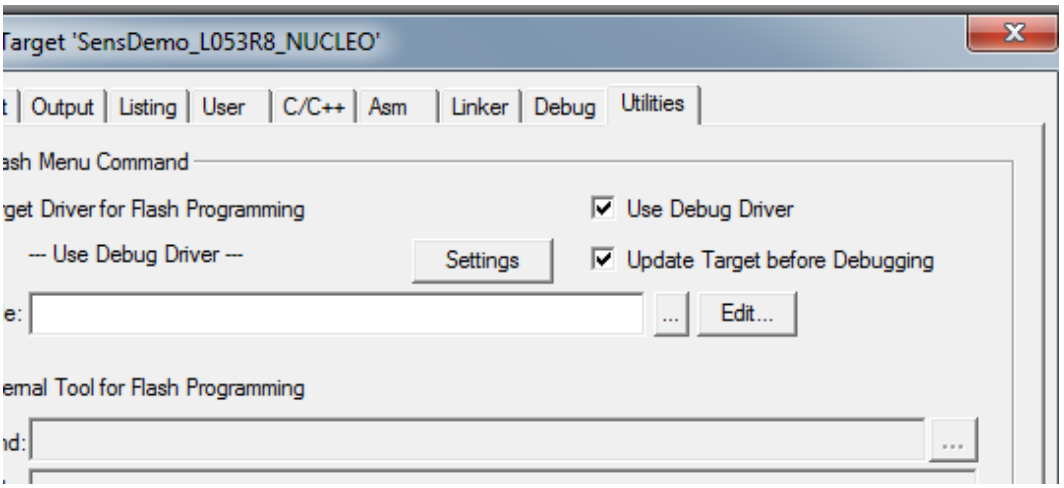
Click Flash button on the top left and open the configure flash tools.



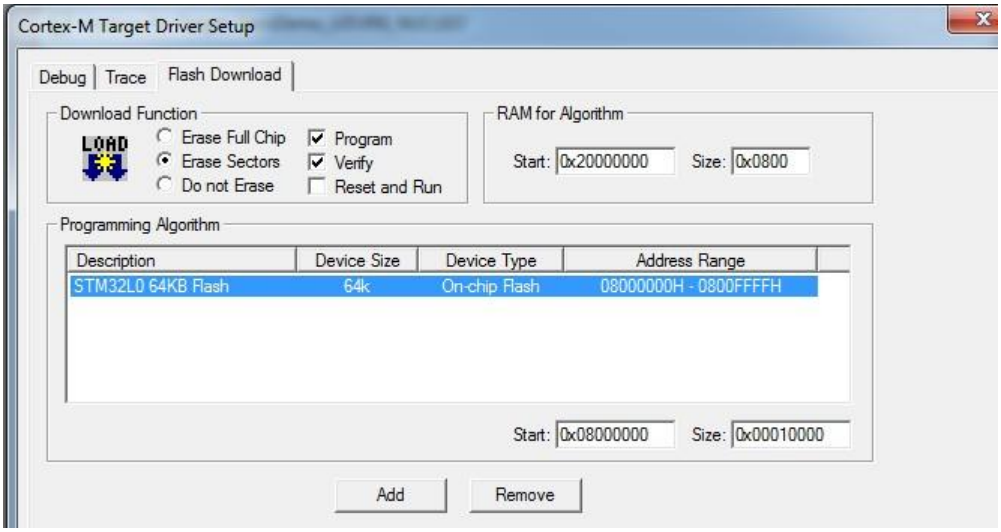
In the “Device” group and select the “STM32L053R8Tx” and click “OK”.



In the “Debug” group and select “ST-Link Debugger”.



In the “Utilities” group and select “Use Debug Driver”.



On the same page, in the “Settings” group, select “STEM32LO 64KB Flash”.

6. After configuring the flash, you can build the code. After build without error, you can download the program into the board, press “F8” or click the button “Load” on the top left.

When it finishes, you can call the tutors to show how the demo works.

Section 4

Rename device information

In this section, you are asked to rename your Bluetooth:

1) Device name
2) Local name
3) MAC address

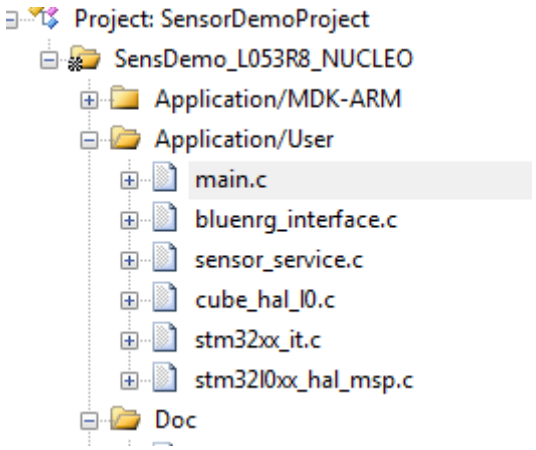
To avoid device connection conflicting, each BLE device should have a specific identification. To identify a device, the designer should define three kinds of IDs. They are **device name**, **local name** and the **MAC address**. The local name and the MAC address can be found on the scanning list of the smart terminal (e.g., mobile phone), which means they

can be got before building the pairing. Until the paring is established, the terminal can get the BLE device's device name. To avoid the connecting conflicting among groups, we will allocate each group the specific addresses. Please following the form below:

Before editing them, you need to read the BLE stack first, "BLE_stacks programming guidelines.pdf"

Group N0.	Local name	Device name	MAC address
1	BLE_LOC_G1	Dev1	0xF1, 0x33, 0x01, 0xE1, 0x80, 0x03
2	BLE_LOC_G2	Dev2	0xF2, 0x33, 0x01, 0xE1, 0x80, 0x03
3	BLE_LOC_G3	Dev3	0xF3, 0x33, 0x01, 0xE1, 0x80, 0x03
4	BLE_LOC_G4	Dev4	0xF4, 0x33, 0x01, 0xE1, 0x80, 0x03
5	BLE_LOC_G5	Dev5	0xF5, 0x33, 0x01, 0xE1, 0x80, 0x03
6	BLE_LOC_G6	Dev6	0xF6, 0x33, 0x01, 0xE1, 0x80, 0x03
7	BLE_LOC_G7	Dev7	0xF7, 0x33, 0x01, 0xE1, 0x80, 0x03
8	BLE_LOC_G8	Dev8	0xF8, 0x33, 0x01, 0xE1, 0x80, 0x03
9	BLE_LOC_G9	Dev9	0xF9, 0x33, 0x01, 0xE1, 0x80, 0x03
10	BLE_LOC_G10	Dev10	0x10, 0x33, 0x01, 0xE1, 0x80, 0x03
11	BLE_LOC_G11	Dev11	0x11, 0x33, 0x01, 0xE1, 0x80, 0x03
12	BLE_LOC_G12	Dev12	0x12, 0x33, 0x01, 0xE1, 0x80, 0x03
13	BLE_LOC_G13	Dev13	0x13, 0x33, 0x01, 0xE1, 0x80, 0x03

The tips are to go through the code in the folder Application/User>**main.c** and **sensor_service.c**.



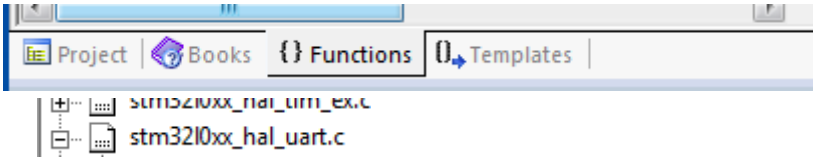
When you finished, please raise your hand up to call the tutor.

Section 5

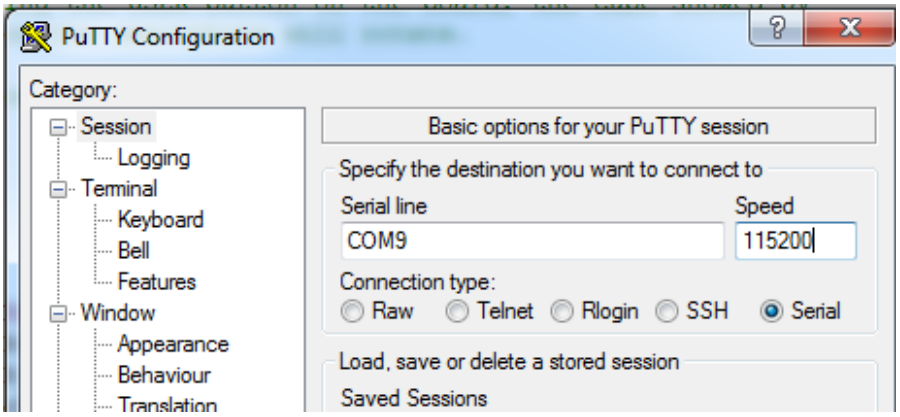
Simple serial communication between the board and the PC

In the section, you are asked to realize the communication between the board and the PC. The communication between the BLE device and PC can be achieved though the UART (Universal Asynchronous Receiver / Transmitter) .

1. Open the folder:
   Source Code>Lab1_UART_Tx>MDK_ARM>Lab1_UART_Tx
   If the software asks to configure the board again, do the same as the section 3 done.
2. You can modify the "**main.c**" script and try to realize communications between the two devices.
   For example, you can ask the board to keep transmitting "The University of Sydney" to the PC via the serial port, and you can read this information via PUTTY on PC. You can look the **stm3210xx_hal_uart.c** functions for help.



   When you use the PUTTY, you can find this software in the "start" menu. Please set the parameters as the following figure:



   The Serial line is the USP port number, which is different in different PCs and you can find the port number in the device manager. The speed is set to **115200**.

When you finish the labs, please record the content and the results in the lab, and write your lab report. The requirement of Lab Report will be released later today.

In this section, we need the screenshot of the PUTTY and the code you added to the source file.