Carleton UNIVERSITY

Department of Electronics

ELEC 4601: Laboratory 5
Wireless Networking Using ZigBee Protocols

Introduction

In this lab, you will use the Keil µVision IDE (and the included mbed library) to program the Arm microcontroller to read raw data from a digital temperature sensor. The sensor is connected to the board by an Inter-Integrated Circuit (I2C) serial bus—a protocol often used to connect low-speed peripherals to processors and microcontrollers. For this lab, all you need to know is that I2C has two (bidirectional) lines—data (SDA) and clock (SCL)—and is an included interface in the mbed API (see I2C.h). You will need to write some basic C++ code to convert the raw temperature data into decimal values that makes sense to the user.

This lab also provides you with valuable experience in using Universal Asynchronous Receiver-Transmitter (UART) serial communication and ZigBee wireless personal area networking. Using UART, you will continuously send temperature values to a ZigBee-compatible transmitter device (connected through an application shield). For this lab, all you need to know is that UART also has two lines of interest—transmit (TX) and receive (RX)—and is also an included interface in the mbed API (see Serial.h). A second ZigBee-compatible device on the same shield will receive all of the transmitted data, which will be displayed on TeraTerm by the USB communications port.

Hardware & Software

There are ten tools used in this lab:

1. STM32 Nucleo F401RE development board (Arm Cortex-M4)
2. Keil µVision IDE
3. mbed SDK
4. Tera Term terminal emulator
5. **2 Digi XBee Pro Series 2 ZigBee modules**
6. **XBee application shield**
7. **Digi XCTU configuration software**
8. **Analog Devices AD7416 digital temperature sensor**
9. **FTDI breakout device**
10. **Rohde & Schwarz RTM 2034 Oscilloscope**

Digi XBee Pro Series 2 ZigBee module

XBee is an implementation of the ZigBee wireless networking protocol by the company Digi. This small device, shown in Fig. 1, is suitable for creating complex mesh networks or simple point-to-point embedded communications. Each development board in this lab comes with two XBee modules: one module will be configured as a transmitter, and the other will be configured as a receiver.

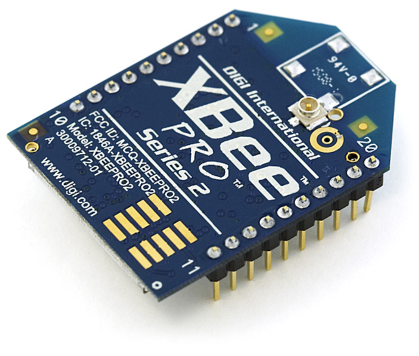
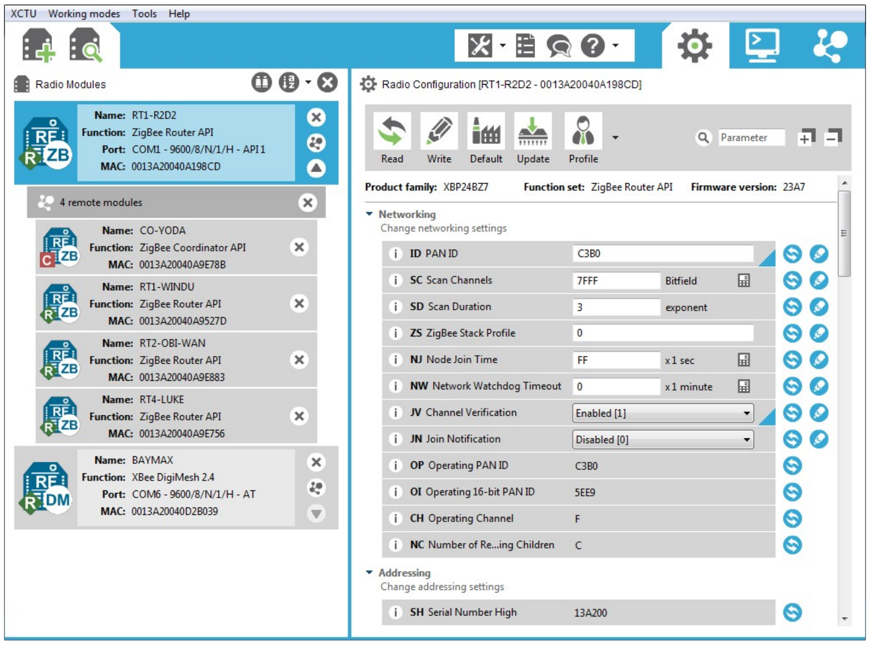


Figure 1 – Digi XBee Pro Series 2 ZigBee Module

Digi XCTU configuration software

XCTU is a free, multiplatform software provided by Digi to set up, configure, and test XBee wireless modules. It has useful embedded tools—like the data console, graphical network view, and signal-range test—but we will only use it to set up our simple, 2-node network. The graphical interface of this software is shown in Fig. 2.



Name: RT1-R2D2
Function: ZigBee Router API
Port: COM1 - 9600/8/N/1/H - API 1
MAC: 0013A20040A198CD

4 remote modules

Name: CO-YODA
Function: ZigBee Coordinator API
MAC: 0013A20040A9E78B

Name: RT1-WINDU
Function: ZigBee Router API
MAC: 0013A20040A9527D

Name: RT2-OBI-WAN
Function: ZigBee Router API
MAC: 0013A20040A9E883

Name: RT4-LUKE
Function: ZigBee Router API
MAC: 0013A20040A9E756

Name: BAYMAX
Function: XBee DigiMesh 2.4
Port: COM6 - 9600/8/N/1/H - AT
MAC: 0013A20040D2B039

Radio Configuration [RT1-R2D2 - 0013A20040A198CD]

Product family: XBP24BZ7 Function set: ZigBee Router API Firmware version: 23A7

Networking
Change networking settings

ID PAN ID	C3B0	
SC Scan Channels	7FFF	Bitfield
SD Scan Duration	3	exponent
ZS ZigBee Stack Profile	0	
NJ Node Join Time	FF	x 1 sec
NW Network Watchdog Timeout	0	x 1 minute
JV Channel Verification	Enabled [1]	
JN Join Notification	Disabled [0]	
OP Operating PAN ID	C3B0	
OI Operating 16-bit PAN ID	5EE9	
CH Operating Channel	F	
NC Number of Re...ing Children	C	

Addressing
Change addressing settings

SH Serial Number High	13A200

Figure 2 – Digi XCTU Screenshot

Analog Devices AD7416 digital temperature sensor

The AD7416 digital temperature sensor from Analog Devices is a 10-bit analog-to-digital converter that simply monitors the temperature (from -40°C to 125°C). It is I2C-compatible, meaning that we can easily connect it to the STM development board through the mbed I2C interface. In this lab, you will have to refer to the provided datasheet in order to properly access the temperature sensor.

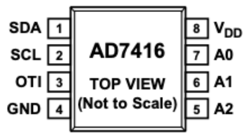


Figure 3 – AD7416 Pin Configuration

FDTI breakout device

The FTDI breakout is a simple device that allows you to communicate serially with the XBee modules through USB. You will use this port to configure the XBee modules and to read the data transmitted between them. This device is connected through the XBee top shield and is shown in Fig. 4 below.

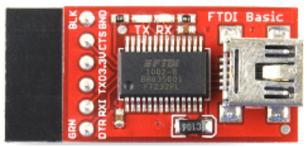


Figure 4 – FTDI Breakout Device

Part A: Setting up the ZigBee Network

In this part of the lab, you will use the Digi XCTU software to configure the two XBee modules into a wireless personal area network. There are no deliverables here but be sure to carefully follow each step in the process, as you will need a working network in order to properly debug/test your software.

Procedure

The XBee modules in the lab have two operating modes:

- AT (Transparent) mode:
  - In this mode, the device acts as a simple wireless modem: any data that is transmitted to the device via UART will be transmitted over the wireless link to the destination node. Also, any data that is received from the network will be made available by the device on the UART RX line
- API mode
  - When operating in this mode, data is transmitted in packet form. This mode provides more information about the network (e.g., was the packet received?) and easier addressing (the destination address of the packet is included in the packet, much like internet TCP/IP packets) at the expense of extra software complexity
  - A network can contain both API and AT mode devices. Data transmitted by a device operating in AT mode can be received by a device operating in API mode. The reverse is not possible

Xbee modules can also be configured with different roles in the ZigBee network, which you can change by changing the firmware inside the module:

- ZigBee Coordinator, or ZC (there needs to be at least one in the network)
- ZigBee Router, or ZR
- ZigBee End Device, or ZED

For this lab, each station will have two XBee modules. One module will be configured in AT mode with the Router AT firmware. The other module will be configured in API mode with the Coordinator API firmware.

1. Open the **XCTU** software

2. Click on **Add devices** and select **USB Serial Port** in the list of available ports. Click **Finish**

3. Once the module has been added successfully, you should be able to see it in the list of **Radio Modules** on the left. Check to make sure its **Function** is set to **ZigBee Coordinator API**; if so, click on the module and see the list of configuration parameters that appear on the right

4. Set the **Personal Area Network (PAN) ID** for your network: use one of your student numbers

5. Write down the **Serial Number High (SH)** and **Serial Number Low (SL)** parameters of this module: we will need to know where to send data to when configuring the next device

6. You have successfully configured **BEE1** (written on the shield underneath the module) as the **ZigBee** Coordinator (i.e., the module that receives data). We will now configure the other module (labeled as **BEE2** on the shield) as the router that transmits the data. To do so, we must first make some important circuitry changes:
   i. Unplug the two USB cables
   ii. Change the position of the two jumpers to connect **BEE2**. The pin layout is shown in Tab. 1
   iii. Set the position of the switch at the bottom of the shield to the middle
   iv. Plug the USB cables back in

Table 1 – Pin Selection for XBee Shield

BEE2_RX	TX	TX	BEE1_RX
BEE2_TX	RX	RX	BEE1_TX

7. Click the **X** button to remove the module we just added

8. Click on **Add devices** and select **USB Serial Port** in the list of available ports. Click **Finish**

9. Once the module has been added successfully, you should be able to see it in the list of **Radio Modules** on the left. Check to make sure its **Function** is set to **ZigBee Router AT**; if so, click on the module and see the list of configuration parameters that appear on the right

10. Set the **Personal Area Network (PAN) ID** for your network: use one of your student numbers

11. Set **Destination Address High (DH)** and **Destination Address Low (DL)** to the **SH** and **SL** values you wrote down in Step 5

12. We will now connect the two modules. To do so, we must first make some important circuitry changes:
    i. Unplug the two USB cables
    ii. Change the position of the two jumpers back to reconnect **BEE1**
    iii. Set the position of the switch at the bottom of the shield back to the right
    iv. Plug the USB cables back in

13. Click the **X** button on the module we just added

14. Click on **Add devices** and select **USB Serial Port** in the list of available ports. Click **Finish**

15. Click on the **Discover radio node in the same network** icon under the **X**

16. If you can see the **Router AT XBee** listed under the discovered devices, congratulations, you have successfully setup the ZigBee network. Click to add the module to the network

Part B: Acquiring and Transmitting Data from the Temperature Sensor

In this part of the lab, you will write the C++ code to read data from the temperature sensor and write it to the XBee module. TeraTerm will be used to view the serial data being received by the ZigBee Coordinator.

Procedure

You will need to finish the project, test it, and check with the TA before checking out. The following steps will guide you along the process:

1. Download and open the project "I2C.uvprojx" in **Keil µVision**

2. Familiarize yourself with "main.cpp" by reading the comments and C++ code already included. You will need to fill in code wherever you see the comment //Write your code here. The comments already in the file should be sufficient in guiding you along this process

3. Open **Tera Term** and select the USB communications port

4. Check if your code works (i.e., does Tera Term continuously display the temperature readings?). If it does, put your finger on the sensor and see how it responds to temperature changes. Demo your program to the TA before moving on

5. Use the **oscilloscope** to capture the **I2C** waveform of one temperature reading. Wire connections have already been made to the SCL and SDA pins, you just have to connect them to the oscilloscope and capture the waveform. There is a sticker on top of the oscilloscope with an address you can type into your web browser; you can capture a screenshot of your waveform from there

Part B Deliverables

Before moving on, make sure you have the following:

- Save the entire project for later reference (i.e., writing your lab report)
- One screenshot of the I2C temperature reading

Lab report

Answer the questions in the **Google Form** posted on **cuLearn**.