CS 498 Internet of Things
Lab 2

Team Ace
Alexey Burlakov (NetID: alexeyb2)
Christopher Lewis (NetID: calewis4)
Li Yi (NetID: liyi2)
Pui Sze Pansy Ng (NetID: ppn2)
Zhijie Wang (NetID: zhijiew2)

Table of contents

1. Introduction

In this lab, our goal is to build an IoT monitoring infrastructure for an enterprise IoT environment. We will monitor the sensors and gather the data of MAC addresses via Wireless protocols.

Here is the GitHub links: https://github.com/alexeyb2-illinois/CS498_IoT_Lab2 and https://github.com/yilimail/CS498_IOT_Lab2
Part I and Part II videos shared via Google Drive.

Here is the division of our works:
Each team member has completed circuit connection, task 1, task 2, and task 3. Some of the code is shared by Alexey Burlakov and Li Yi. Task 1 report writing, Question 2, and Question 3 are completed by Pui Sze Ng. Task 2 report writing is completed by Zhijie Wang. Task 3 report writing is completed by Alexey Burlakov. Question 1 and Question 4 are completed by Li Yi.

2. Task 1: Setting up the Sensing Device (Sensor)

First, we connected the ESP8266 with the Arduino and followed the ESP-8266 WiFi Sniffer code written by Ray Burnette. Through this code, we can get a list of MAC addresses of router and devices.
After completing the setup of the MAC address scanner, we connected the Ardunio with an air quality sensor, which shown in Figure 2.1

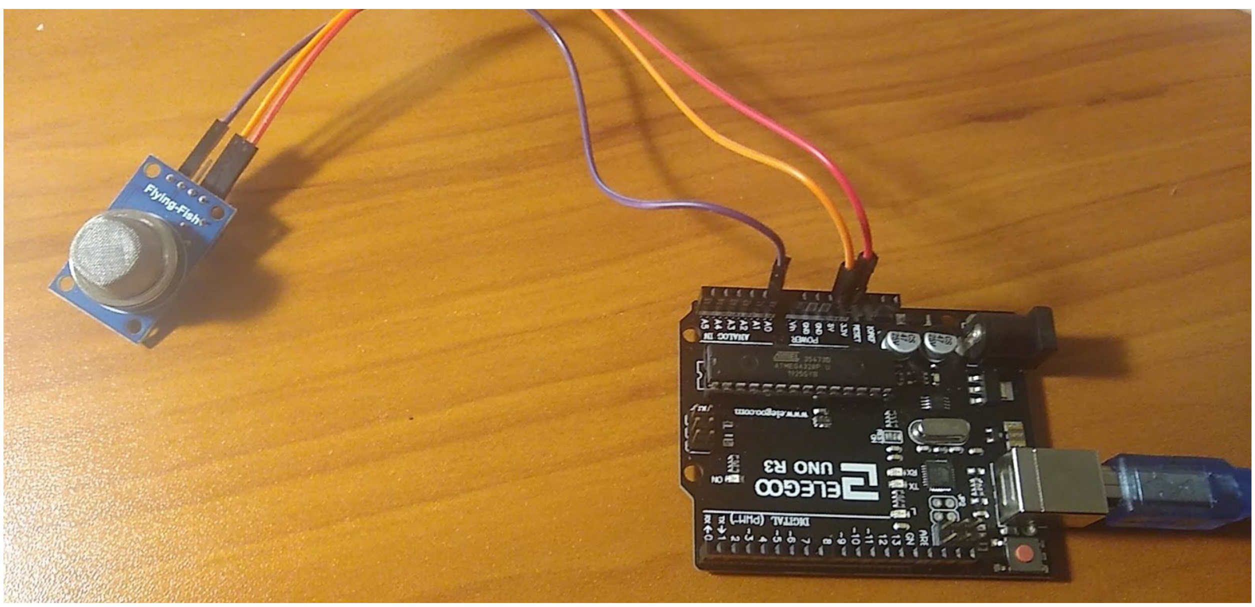


Figure 2.1 Air Quality Sensor and Arduino Uno

3. Task 2: Set up Xbee Mesh Network

In this task, we used XBee and XCTU to create a Xbee Mesh Network We connect 3 XBee to arduino with air quality sensor(MQ-135), raspberry pi, and ESP-8266 Node MCU respectively.

The ESP-8266 MCU connects with XBee in Edge mode, periodically publishing the count of MAC addresses detected to router Xbee.

The Arduino connects with air quality sensor and XBee in Router mode. The Arduino reads the count of Mac addresses from the Serial and appends the air quality info before relaying the information back to Coordinator.

The Raspberry Pi connects to the XBee in Coordinator mode through RX/TX UART serial. We need to first do raspi-config with “sudo” to enable Serial interface. Then every time the Raspberry Pi receives any information relayed from Router, it marks the timestamp for recording purpose.

Common Setting
- PAN ID:1234
- Function 802.14.5
- Channel: C
- Baudrate:9600

Device Specific:
- Xbee 1 -- Coordinator 0013A20041A31D76
    - DH:0
    - DL:FFFF
    - CE:1 (coordinator)
- XBee 2 -- Router 0013A20041A31FF6
    - DH : 0
    - DL : 0
    - SM : 0
- XBee 3 -- Edge 0013A20041A31FC3
    - DH: 13A200
    - DL: 41A31FF6
    - SM: 4

Wiring -- Edge Xbee using USB adapter board + Arduino

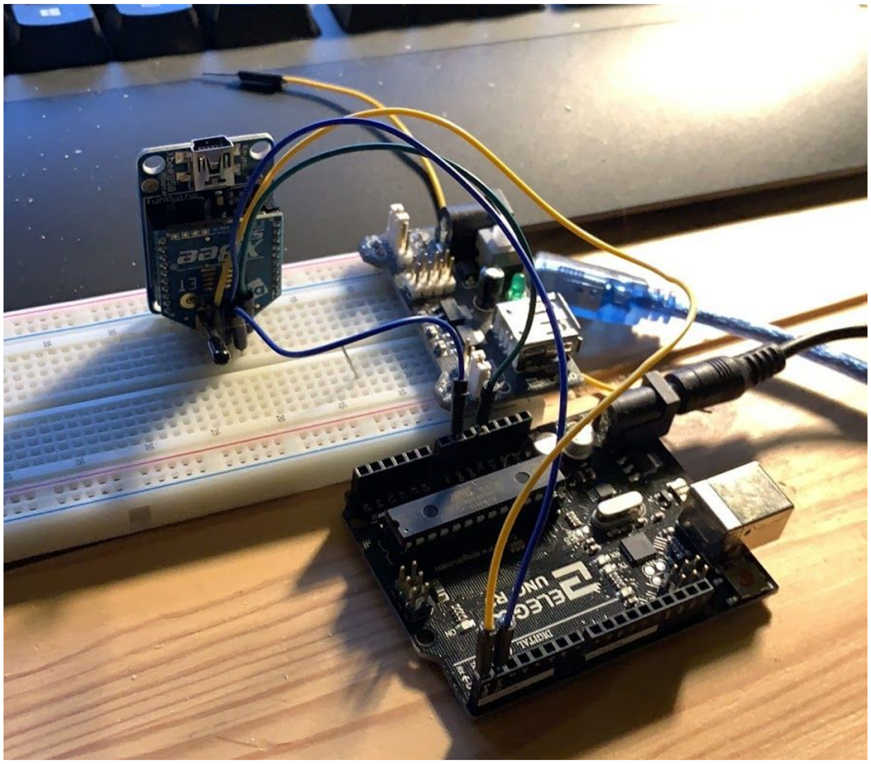
XBee	Arduino
3.3 V	3.3 V

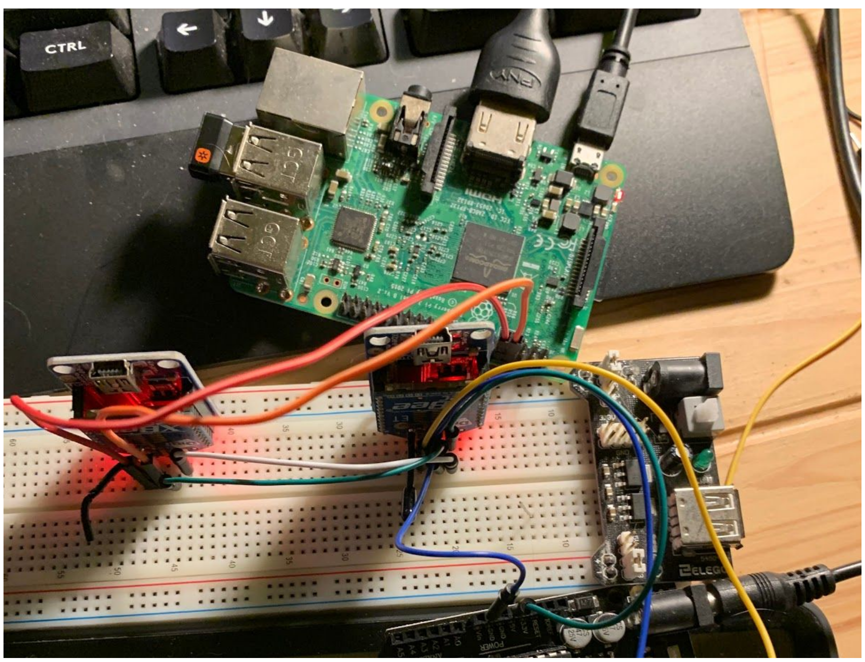
GND	GND
DIN	TX
DOUT	RX

Arduino code utilizes the default Serial command, set baud rate 9600.

Wiring -- Raspberry Pi and XBee using USB adapter board

XBee	Raspberry Pi
3.3 V	3.3 V
GND	GND
DIN	TX
DOUT	RX





4. Task 3: Setting up the Web Server on Raspberry Pi

XBee Coordinator is Raspberry Pi connected to XBee module so that XBee's DOUT (TX) is connected to RPi's GPIO 15 (RXD).

Raspberry Pi will continuously read data received from XBee, parse it into number of active WiFi devices and gas sensor data (CO2 level). The time of reception (in UNIX timestamp format) will also be recorded. If a path to SQLite database file is provided (by default it is **./sensor_data.db**, will be created if not exists) the sensor data will be locally stored in an SQLite table **sensor_data(mac INTEGER, air INTEGER, timestamp INTEGER)**.

RPi will also launch a simple HTTP server on port 8080 that will respond to HTTP GET requests with the latest received sensor data in JSON format:

```
{
   "mac": 12, // number of unique MAC addresses
   "air": 34, // MQ-135 reading
   "timestamp": 1584194172 // UNIX timestamp (seconds)
}
```

To make the device work as intended it is necessary to enable UART and disable Linux's use of console UART (can be done by configuring **Interfacing options -> Serial** in **raspi-config**) and reboot.

To start the routine the following commands should be issued:

```
git clone https://github.com/alexeyb2-illinois/CS498_IoT_Lab2.git
cd CS498_IoT_Lab2/coordinator
chmod +x ./launch.sh
./launch.sh
```

The data, served by Raspberry Pi, is visualized on an HTML page (**monitor.html**). For the page to work as intended the monitoring device (PC, smartphone etc.) must be in the same network as Raspberry Pi and port 8080 on RPi must be open. The file **layout.png** must be in the same folder as **monitor.html**. The page **monitor.html** should be opened in any modern browser (tested on Google Chrome). The value in the **Address** field of the page should be Raspberry Pi's IP address (default value is 192.168.0.10 - the IP of RPi in my network).

The number of unique MAC addresses and the level of CO2 will be updated periodically (default period is 1 second) if everything is set up correctly.

5. Questions

5.1. Compare the advantages and disadvantages of using Zigbee, Bluetooth(Classic & BLE), and WiFi wireless technologies in terms of power consumption, range, frequency, bandwidth, and future feasibility. Also write one paragraph on what each wireless technology is best suitable for in a smart home environment

Power Consumption: BLE is the most power saving option compared with Zigbee, WiFi and Bluetooth. BLE consumes very low energy as it has sleep functionality. Zigbee does not consume much power either, but it consumes more power than BLE. Bluetooth consumes relatively high power compared with BLE and Zigbee. WiFi has the highest power consumption compared with Bluetooth, BLE, and Zigbee.

Range: For Zigbee, each node can reach short-range. But it can use device to device communication creating a mesh network. The mesh network allows users to create a huge network of interconnected devices that can cover a large area. Bluetooth Low Energy also supports mesh Network, so it allows connecting a large number of BLE devices over a wider area. Range of Bluetooth is dependent on its class. Class 1 transmitting at 100 mW with a range of 100 meters or 328 feet. Class 2 transmitting at 2.5 mW with a range of 10 meters or 33 feet (most Bluetooth headsets and headphones are common Class 2 devices). Class 3 transmitting at 1 mW with a range of fewer than 10 meters. WiFi operating on 2.4 GHz band can reach up to 150 ft. For 5 GHz band, because it uses narrower wavelengths, it is more susceptible to obstructions than 2.4 GHz, so it has a slightly shorter effective range.

Frequency: Bluetooth and Bluetooth Low Energy Zigbee uses 2.4 GHz carrier frequency, WiFi uses 2.4GHz and 5GHz depends on different WiFi standards.

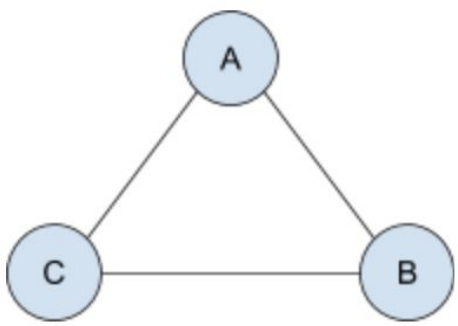
Bandwidth: WiFi has the highest bandwidth. The 802.11 Wi-Fi standards specify a bandwidth of 22 MHz and channels are on a 5 MHz incremental step. Bluetooth has a low bandwidth (1 MHz). Bluetooth Low Energy has a higher bandwidth (2 MHz) than Bluetooth. Zigbee has a low-channel bandwidth of 1 MHz.

Future Feasibility: WiFi is used by very many devices from smartphones to laptops and tablets to remote sensors, actuators televisions and many more. It is used as the main wireless communications bearer in wireless LANs as well as for small home WLANs as well. Bluetooth and BLE are widely used in all industries. BLE has many applications but one of the most common is transmitting sensor data. Zigbee is the low-cost and low-powered mesh network that is widely deployed for controlling and monitoring applications. It is less expensive as well as simpler than the other proprietary short-range wireless sensor networks such as Bluetooth and Wi-Fi connectivity.

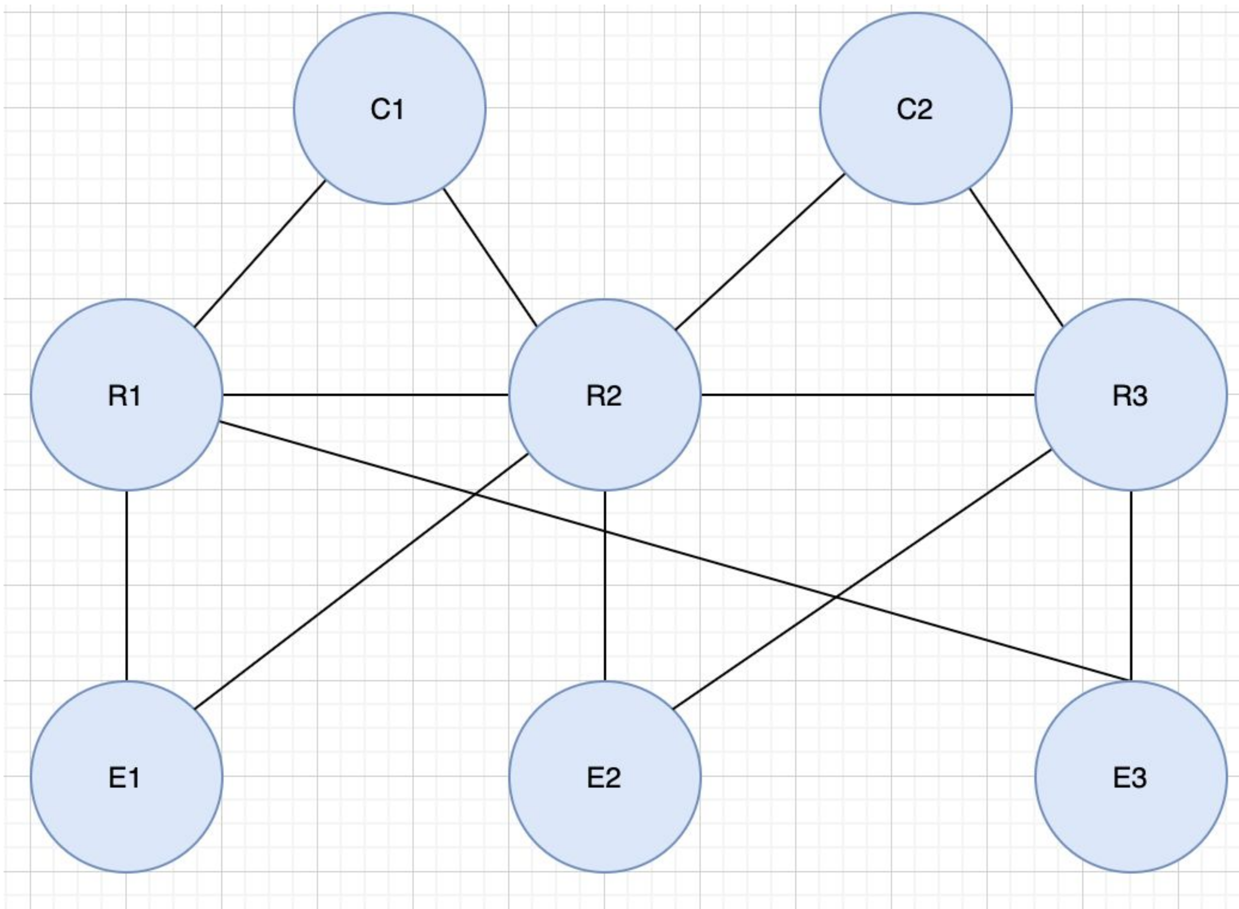
Since ZigBee is a low-energy, low-data network. It is best suitable for a Smart home environment. ZigBee is a local area network (LAN). It connects to devices that need a wider range. Because of this, it’s an ideal protocol for home automation and smart lighting. It is also commonly used in industrial automation, smart meters, and security systems.

5.2. What is the difference between a router and an end device for XBee and which saves more power?

Router is responsible for routing networks among different nodes. It also acts as a gate keepers of the networks that are responsible for allowing new nodes to join the network. Routers run for 24/7 and do not sleep. The power consumption of routers is very high. On the other hand, an end device does not route traffic and it has a sleep mode which saves more power and is ideal for battery operated devices.



5.3. If you have 8 nodes (i.e. 3 groups working together, 2 coordinator nodes, 3 router nodes & 3 end nodes), please design & draw a network configuration which can tolerate (i.e. other nodes that do not fail still have connectivity to a live coordinator) at least 1 node or edge failure. Example shown above is a graph of 3 nodes that could tolerate one edge or node failure. (For example, assume Edge AC fails, A can still send a message to C via edge AB and BC.)

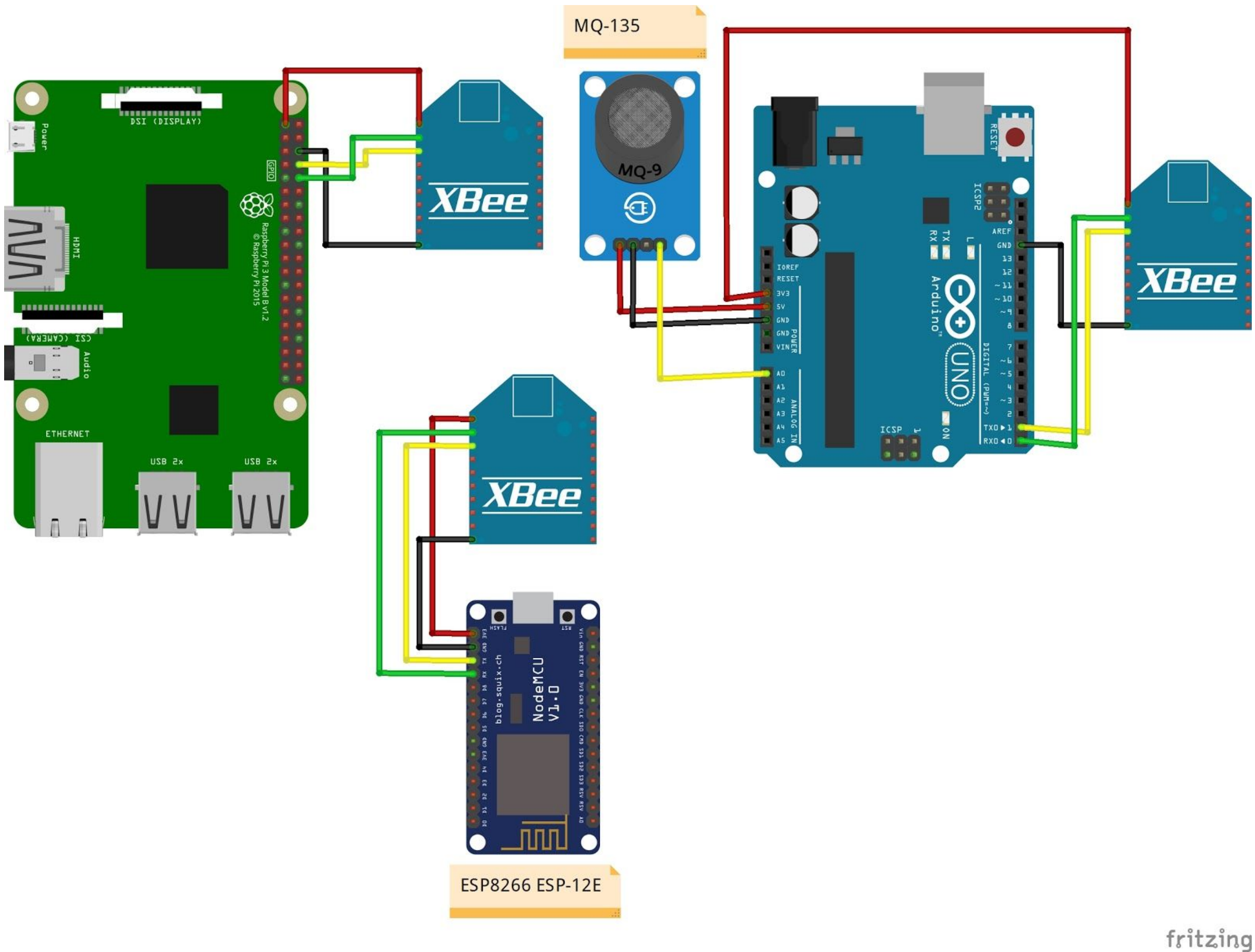


C1 and C2 represent the coordinator nodes. R1, R2, and R3 represent the router nodes. E1, E2, and E3 represent the end nodes. For example, we would like to connect C2 to E3. In normal situations, it would go to the edge C2-R3, then R3-E3. If Edge C2-R3 or node R3 fails, it still can send message to E3 via edge C2-R2, R2-R1, and R1-R3

5.4. What other real-world applications can you think of for the kind of data collected in this lab?

The air quality data and mac addresses collected can be used for an application of COVID-19 warning. People are carrying cell phones everywhere. If a large amount of people gathered in a public space, then the number of MAC addresses is high. In addition, if PPM (parts per million) is also high in the air, that means people in the area have a high risk of COVID-19 infection. In this scenario, a warning notice should be displayed on the screen.

6. Topology


MQ-135
MQ-9
XBee
XBee
XBee
ESP8266 ESP-12E
NodeMCU
V1.0
Arduino
UNO
fritzing