

**Intelligent and Communicating Systems, ICS**  
2<sup>nd</sup> Year Specialty SIQ G02, 2CS SIQ2

## Bonus-HomeWork

Title:

### Cloud/Standalone Platforms and **ZigBee** Simulation

Studied by:

**First Name:** Nour el Imane

**Last Name:** HEDDADJI

**E-mail:** jn\_heddadji@esi.dz

# A. Theory

## 1. Zigbee Protocol in IoT

ZigBee is a popular wireless communication protocol that is designed for low-power and low-data-rate IoT applications. Developed under the IEEE 802.15.4 standard by The Zigbee Alliance, this protocol introduces a range of features tailored for efficient communication in IoT scenarios.

### 1.1. Key Zigbee Features

- Support for multiple network topologies (point-to-point, point-to-multipoint, and **mesh networks**).
- Low duty cycle, ensuring **extended battery life** for devices.
- Low latency for responsive communication.
- Up to 65,000 nodes per network, supporting scalability.
- 128-bit **AES encryption** ensures secure data connections.
- **Collision avoidance**, retries, and **acknowledgements** for robust communication.

The catch with these networks is that they offer a low data rate as shown in the next table :

Network	Supported Data Rates (bps)
Wi-Fi	1.73 Gbps, 866.7 Mbps, 450 Mbps, 54 Mbps
Bluetooth	24 Mbps, 3 Mbps, 700 Kbps
<b>Zigbee</b>	<b>250 Kbps</b> , 100 Kbps, 40 Kbps, 20 Kbps

Table 1: Supported Data Rates of Different Networks

The Zigbee protocol utilizes the **IEEE 802.15.4** standard for the first two layers and adds the **network, security, and framework** protocols on top of it. The network layer ensures **reliable and secure** transmission among devices.

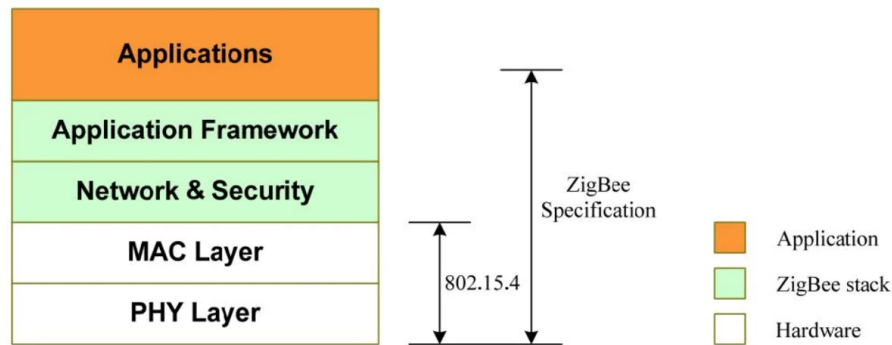


Figure 1: Zigbee Protocol Architecture.

The Zigbee network defines three types of devices:

1. **ZigBee Coordinator**: responsible for initializing, maintaining and controlling the network. There is one and only one per network.
2. **ZigBee Router**: connected to the coordinator or other routers. It can have zero or more children nodes. Participate in multi hop routing
3. **ZigBee End Devices**: does not participate in routing

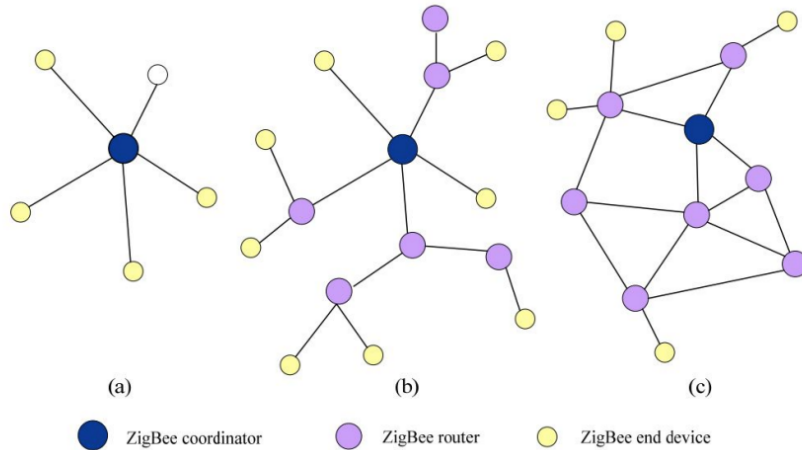


Figure 2: Types of Zigbee Devices.

## 1.2. Zigbee's concurrent place in the IoT world.

Zigbee finds extensive use in smart home and building automation applications, including lighting control, HVAC systems, and security sensors.

Despite facing competition from other wireless protocols such as Wi-Fi, Bluetooth, and Z-Wave, Zigbee remains a **preferred choice** for industrial and commercial IoT applications, where reliability and security are paramount.

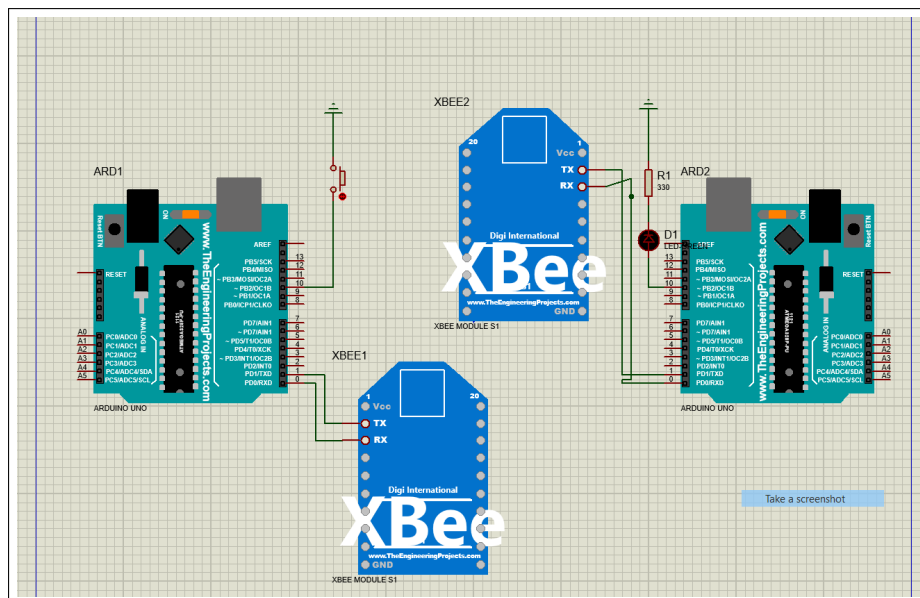
## B. Activity

### 1. ZigBee simulation

#### Hardware Setup

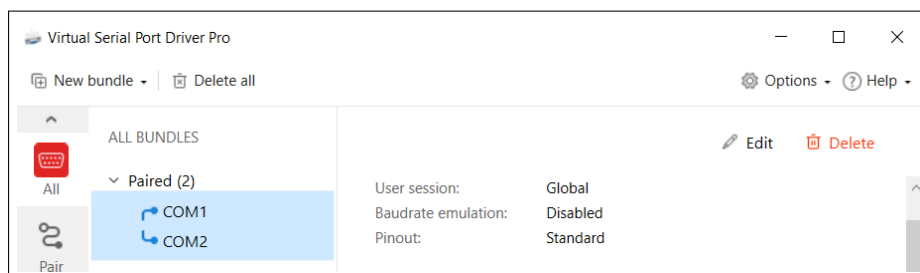
Connect the Arduino boards to the XBee modules using the following connection:

1. Arduino TX -> XBee TX
2. Arduino RX -> XBee RX

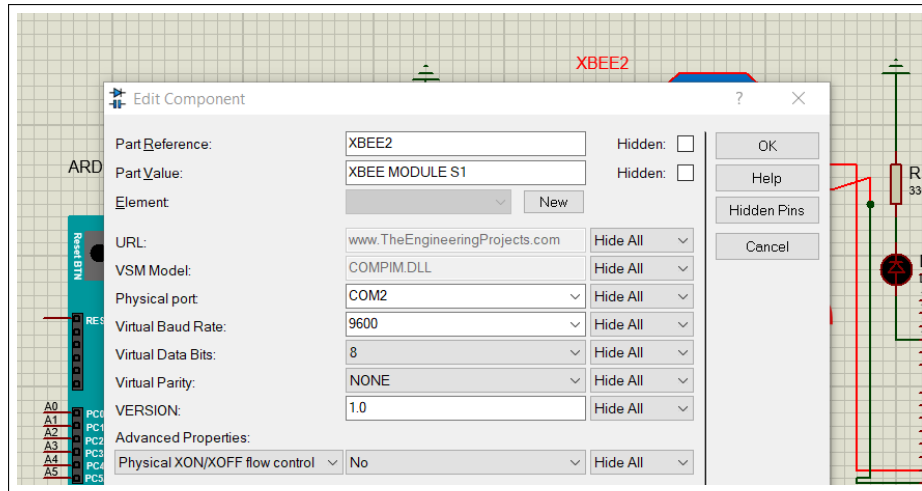


#### 1.1. Port assignments using Virtual Serial Port Driver

1. Pair creation in VSPD



2. Assign each port to a Zigbee Xbee



## Software

```

1 void setup() {
2   pinMode(10, INPUT_PULLUP);
3   // Enable the internal pull-up resistor
4   Serial.begin(9600);
5   // Start serial communication at 9600 baud
6 }
7
8 void loop() {
9   int buttonState = digitalRead(10);
10  // Read the button state
11  if (buttonState == LOW) { // Button is pressed
12    Serial.println('1');
13    // Send a message to the second Arduino
14  } else {
15    Serial.println('0'); // Button is released
16    // Send a message to the second Arduino
17  }

```

Listing 1: the code to handle the pushbutton

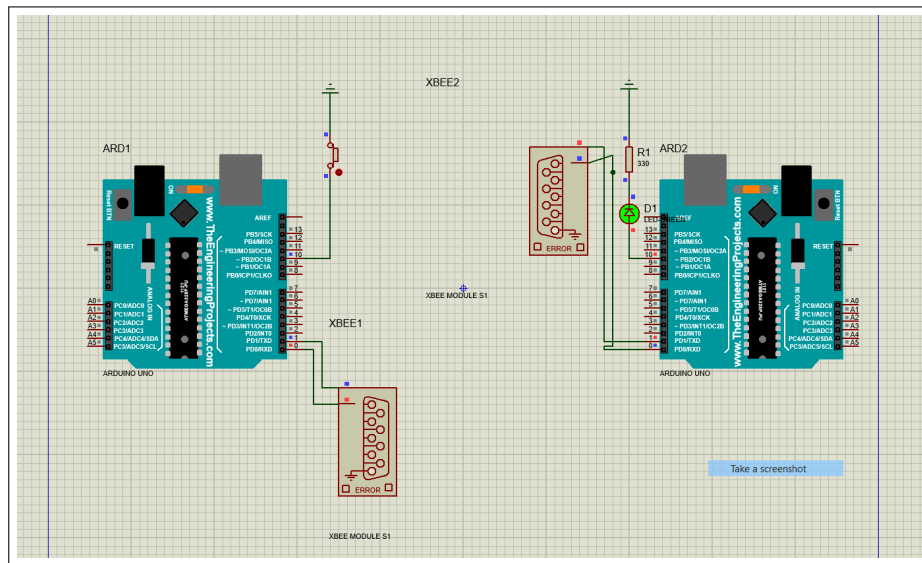
```

1 char Signal;
2 const int LEDpin = 10; // Assign the LED pin number
3
4 void setup() {
5   pinMode(LEDpin, OUTPUT); // Set the digital pin connected to the LED as
6   // output
7   Serial.begin(9600); // Start serial communication at 9600 baud
8 }
9
10 void loop() {
11   Signal = Serial.read(); // Read the incoming signal
12   if (Signal == '1') {
13     digitalWrite(LEDpin, HIGH); // Turn on the LED
14   } else if (Signal == '0') {
15     digitalWrite(LEDpin, LOW); // Turn off the LED
16   }

```

Listing 2: the code to handle the LED

## 1.2. Results



## 2. Experimenting another IoT platform

Having already worked with Arduino IoT Cloud , Blynk, Home-Assistant in LAB08 and LAB09 . I decided to experiment with openHab for the bonus homework.


### 2.1. OpenHab

#### 1. Install openHAB on raspberry

```
1 curl -fsSL "https://openhab.jfrog.io/artifactory/openhab-linuxpkg/stable/main" | sudo gpg --dearmor -o /usr/share/keyrings/org.openhab.gpg
2
3 sudo mkdir -p /usr/share/keyrings
4 sudo mv org.openhab.gpg /usr/share/keyrings/
5 sudo apt-get install apt-transport-https
6 sudo apt-get update && sudo apt-get install openhab
7 sudo apt-get install openhab-addons
8 sudo systemctl enable openhab
9 sudo systemctl start openhab
```

Listing 3: Install and Configure openHAB on Raspberry Pi

#### 2. Create an account



Create a first administrator account to continue.

User Name

Password

Confirm New Password

Create Account

### 3. MQTT binding

```
1 sudo openhab-cli console
2 feature:install openhab-binding-mqtt
3 logout
4 sudo systemctl restart openhab
5
```

Listing 4: Install and Configure MQTT binding

4. Next, we create MQTT Broker.

### New MQTT Broker

Unique ID	42e0524e6e <small>Note: cannot be changed after the creation</small>
Identifier	mqtt.broker.42e0524e6e
Label	MQTT Broker
Location	e.g. Kitchen

#### MQTT Broker

A connection to a MQTT broker

Show advanced ☒

Broker Hostname/IP  
192.168.43.204

**Required** The IP/Hostname of the MQTT broker

Broker Port  
1883

The port is optional, if none is provided, the typical ports 1883 and 8883 (SSL) are used.

Secure Connection ☐

**Required** Uses TLS/SSL to establish a secure connection to the broker.

Hostname Validated ☒

Validate hostname from certificate against server hostname for secure connection.

Protocol  
☒ TCP  
☐ WebSockets

The protocol used for communicating with the broker.

MQTT Version  
☒ Version 3  
☐ Version 5

The MQTT version used for communicating with the broker.

### Things

1 things

Alphabetical

By binding

M

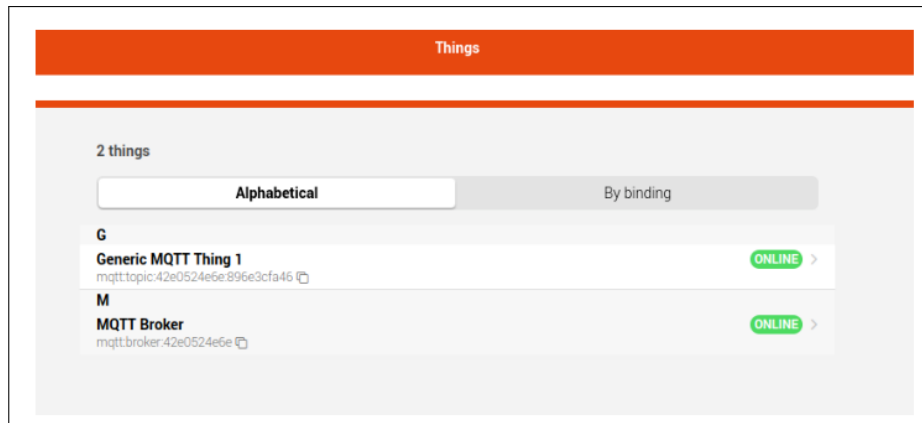
**MQTT Broker**

mqtt.broker.42e0524e6e

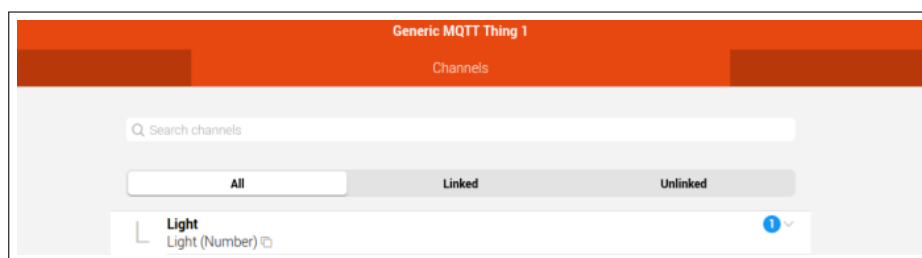
ONLINE >

5. We create a generic mqtt broker.

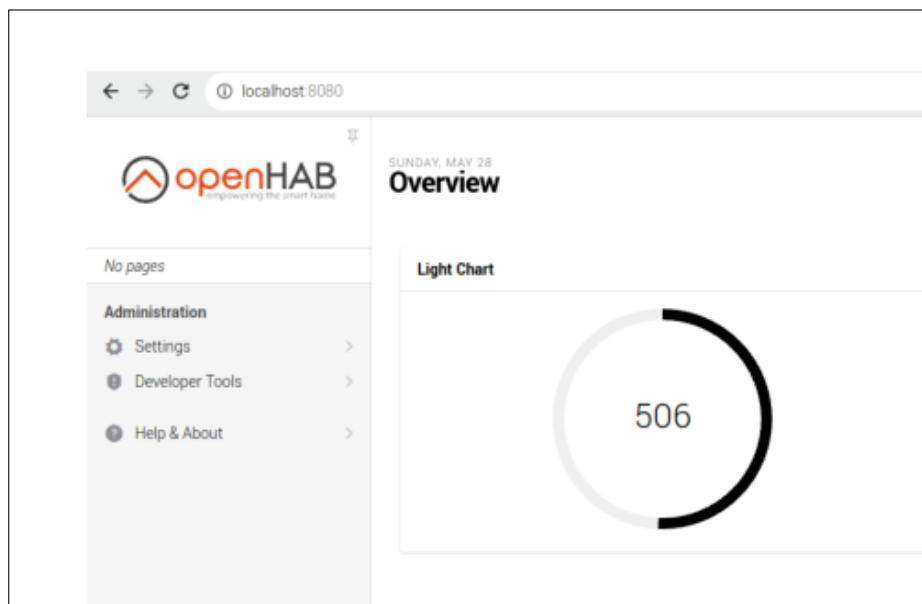




6. we add channels related to the topic LIGHT



7. We display the Light in the dashboard :



## C. Conclusion

This bonus homework provided valuable hands-on experience with both ZigBee simulation and the openHAB IoT platform. The hardware simulation allowed for a practical understanding of ZigBee protocols, showcasing the seamless communication between devices facilitated by the Arduino boards and XBee modules.

On the software side, the exploration of openHAB demonstrated its versatility in orchestrating IoT ecosystems. The setup of MQTT binding showcased openHAB's role as a robust middleware, enabling the integration and management of diverse smart devices. This is particularly crucial in the context of home automation, where centralized control and device interoperability are essential for creating a user-friendly and efficient smart home environment.

The synergy between ZigBee simulation and openHAB experimentation highlights the holistic nature of IoT solutions. ZigBee's low-power, cost-effective communication capabilities are fundamental for establishing efficient wireless networks, while openHAB provides a unified interface for seamless management and control of these networks. This combination not only enhances understanding of wireless communication but also emphasizes the practical significance of platforms like openHAB in simplifying the complexity of IoT deployments. In essence, the convergence of ZigBee and openHAB underscores their collective importance in shaping the landscape of modern smart technologies.