# Zigbee IoT technology



## Introduction

In biology, “zigbee” refers to the “honey bee zig-zag dance”, performed by honey bees when they return to the beehive and need to inform their fellow bees of the path to a new source of sugar.

Zigbee is a very simple and inexpensive IoT communication technology developed to be a complete solution for both the automated home and for business and public sectors. Zigbee can be everything from the lightswitch in your home to the traffic light in the streets. Zigbee is developed to be a platform for controlling smart and simple devices and therefore aims to be cost-effective by being very low-powered and using low-data rates. As Zigbee puts it: “Wireless Control That Simply Works”.

You may already find Zigbee in many of your home products and Zigbee aims to be a great tool to optimize your daily routines. For example, Phillips Hue is built on the Zigbee/802.15.4 standard. Via a phone app you can control the light or heating in your house. Because Zigbee is able to connect to wireless networks you can use Zigbee to control your devices, even if you are not at home.

In contrast to other IoT platforms, Zigbee devices are able to act as links for other devices, thus making a meshed network of devices that can carry and transmit information across long distances without the need for extra network connectivity. This is especially applicable in large scale environments, for instance in factories, public infrastructure, sensory driven workspaces, collection services or with agricultural uses, etc.

# Utilization and comparison

## - What are (some of) the uses and applications of Zigbee?

Zigbee is designed for devices that only require low power and low data transmission and makes networked homes and businesses easy to control and optimize. The low power consumption and low data transmission allows the devices to have long battery life.

Zigbee provides network, security and application services to be able to operate on the top of the IEEE 802.15.4 stack and can be used for both small and large networks The ability to auto-scale as a business or home expands is also a key feature.

## - Case/examples

**Smart home**

Smart home is a concept of building automation for a home. It involves the control and automation of lightning, heating, ventilation, air conditioning and refrigerators. The modern systems consist of devices and sensors connected to a central hub - “gateway” which allows the system to be controlled with a user interface that is integrated either with a wall-mounted terminal or an application software.

**Amazon echo plus**

The Amazon echo plus is a device built on Zigbee architecture thereby allowing for direct setup and control of Zigbee compatible devices.
Echo will connect the user to a cloud-based voice service enabling them to control the smart home via voice commands.

**Philips Hue**

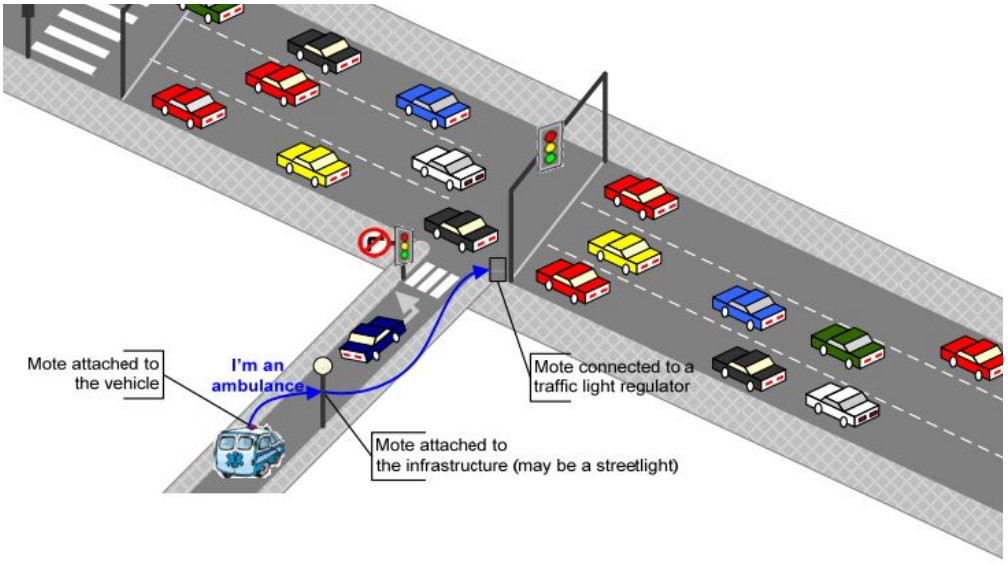
Philips is using Zigbee to control the lights. In a Hue architecture you are using a bridge (FFD) to talk to all your bulbs and light strips. Hue is its own ecosystem that contains the bridge and the bulb/strip. You can control the bulbs and strips from an app on your mobile device. From the app you can turn the bulb on or off and even dim the light. They have also made a out of home connection so that you can turn the lights on and off even when you're not home.


- ## ITS (Intelligent Transport Systems)

Wireless communication in the automotive industry is currently being developed and is expected to be widely employed for various purposes. Zigbee can be used for in-vehicle, inter-vehicle and between vehicle and transport infrastructure installations.

There is no limit to the purposes it could employ. Imagine a traffic light intersection where Zigbee protocols can be used to communicate and control the traffic light sequence and provide a more agile and fast intersection. It could also be used to register passing vehicles for control, security or share information with the vehicle about road temperatures, wind-speeds or anything that might be valuable to the driver.

Example of ITS system:



- How does Zigbee compare to other IoT technologies?

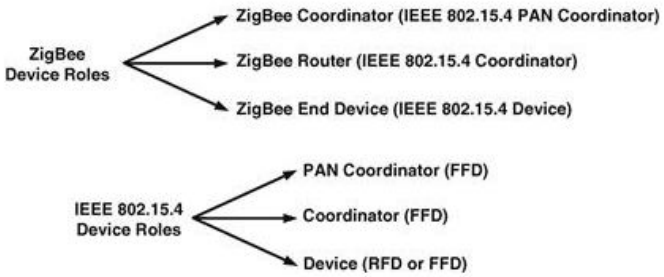
  There are many emerging IoT technologies on the market today, from standards like Zigbee or 6LowPan to brands like Amazon AWS IoT, Microsoft Azure IoT or Cisco Cloud IoT. Zigbee has been around for quite some time and has been created to take off from the 802.15.4 IEEE standard. It has already achieved recognition by several major brands and is in the race to become one of the more popular standards that companies choose to base their technology on. IoT standards work from different aspects and sometimes it makes little sense to compare because they are developed for different reasons. In the tables below we try and compare Zigbee to a few other standards to illustrate the different key functionalities. The most important difference is in the data transfer rates.

| Variable | Wi-Fi | Z-Wave | ZigBee | Thread | BLE |
|---|---|---|---|---|---|
| Year first launched in Market | 1997 | 2003 | 2003 | 2015 | 2010 |
| PHY/MAC Standard | IEEE 802.11.1 | ITU-T G.9959 | IEEE 802.15.4 | IEEE 802.15.4 | IEEE 802.15.1 |
| Frequency Band | 2.4 GHz | 900 MHz* | 2.4 GHz | 2.4 GHz | 2.4 GHz |
| Nominal Range (0 dBm) | 100 m | 30 – 100 m | 10 – 100 m | 10 – 100 m | 30 m |
| Maximum Data Rate | 54 Mbit/s | 40-100 kbit/s | 250 kbit/s | 250 kbit/s | 1 Mbit/s |
| Topology | Star | Mesh | Mesh | Mesh | Scatternet |
| Power Usage | High | Low | Low | Low | Low |
| Alliance | Wi-Fi Alliance | Z-Wave Alliance | ZigBee Alliance | Thread Group | Bluetooth SIG |

# The technology behind

- What are some of the essential characteristics of Zigbee?
  - IEEE 802.15.4 General Characteristics:
    - 2.400-2.4835 GHz (sub 1ghz in China+USA)
    - 250 Kbps bit rate
    - Direct Sequence Spread Spectrum (DSSS)
    - 16 non-overlapping channels (5 MHz separation)
    - 2 MHz channel bandwidth
    - O-QPSK Modulation
    - Receiver sensitivity -85 dBm or better
    - Transmit power +6 dB
    - AES Encryption
    - +65000 devices per network

- Zigbee networking




In IEEE 802.15.4 networks we have two types of devices, FFD's and RFD's. Full-Function Devices (FFD's) are able to perform all tasks that are described in the IEEE standard and can assume any role in the network. Reduced-Function Devices (RFD's) has limited functionality. In Zigbee FFD's (ZC or ZR) can communicate with any device on the network, while RFDs (ZEDs) can only communicate through an FFD. RFD devices are meant to be very simple devices such as power switches or sensors and they typically have less processing power and memory than an FFD.

## Zigbee Coordinator (ZC):

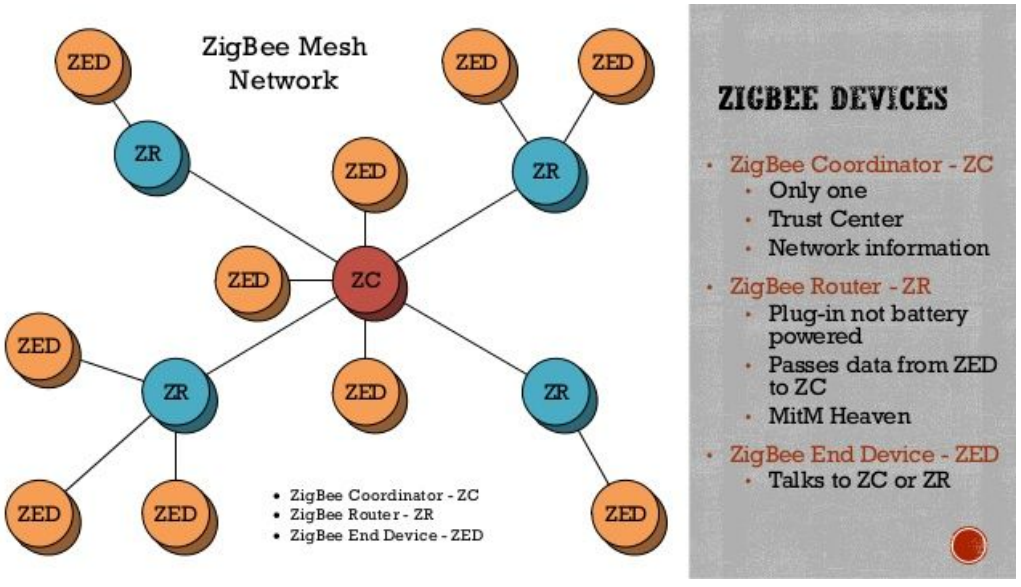
The ZC is unique to other Zigbee devices, in the way that it acts sort of like a normal router. The first device on the WPAN must be an FFD and that FFD will be chosen as the coordinator. It is also the device that forms the network and subsequently, there can only be one ZC on a Zigbee network. A ZC will typically be more powerful than the other Zigbee devices and run constantly with a power supply.

## Zigbee Router (ZR):

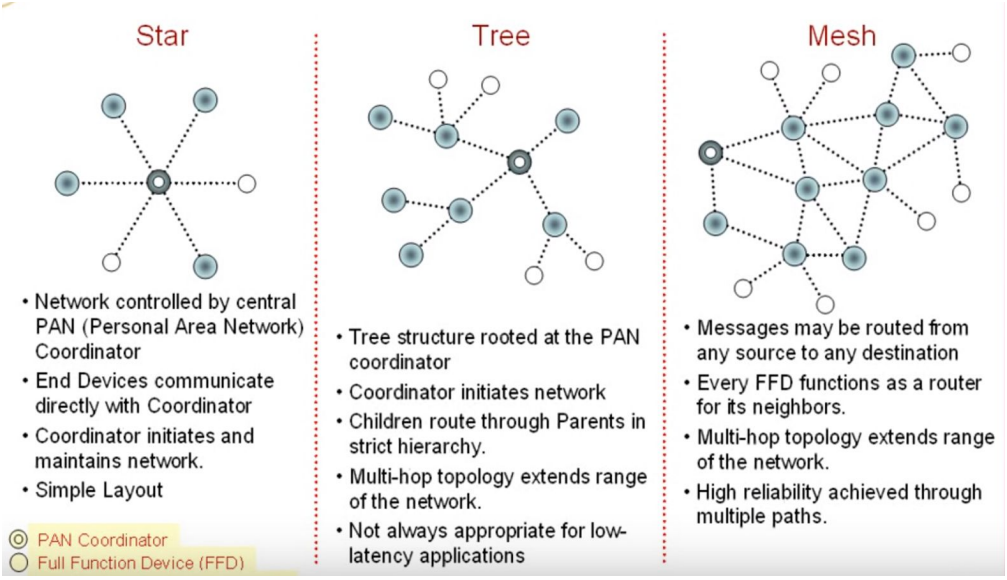
ZR's have the ability to send and receive their own packets but can also relay packets for other devices on the Zigbee network and in consequence the ZR's cannot perform duty cycling (sleep/wake mode) because they have to be available for routing packets in the network. ZR will always be FFD's.

## Zigbee End Device (ZED):

ZED's does not participate in routing because they are exactly what the name suggests: End devices. They can perform their tasks but are reliant on a ZC or ZR to handle the communication to and from the ZED. They may use sleep mode but will require a "parent" or a route in order to communicate on the network.



## Topology




In order to understand what Zigbee really is, we need to look at the topologies we can create for a network. Zigbee can run “Star”, “Tree” but also in “Mesh” topology. With Star-topology all devices must communicate directly with the coordinator.

Tree and Mesh are considered to be peer-to-peer networks where all devices that are able to retransmit (relay) messages from other devices are FFD’s. RFD’s are only able to communicate with one other device in the network, this device will be a ZR or ZC. With Mesh networking you get extended range by “multi-hopping” from device to device and ad-hoc/self-forming formation of the network. The network is no longer restricted, meaning that devices doesn’t have to be directly connected to a specific coordinator.

The most important thing about mesh is that it is self healing by using automatic route discovery. This means that Zigbee is able to figure out - without any manual network administration - the best path from one node to another, even if the two nodes are not directly connected and furthermore, if the used ZR path suddenly becomes unavailable, Zigbee will try and find another route. A derived ability of the mesh topology, is that the WPAN can become a very long ranged network. In fact you could make the network more than 4000 km in range and with 64-addressing in theory it could be infinite.

## Basic Communication

When a PAN and a FFD tries to connect to each other there is a set of rules that apply. The standards IEEE 802.15.4 have a very simple way to connect multiple devices together. Zigbee uses channel access mode called Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) or Time Slots. With CSMA-CA if the device want to start a transmission it first have to send an Clear Channel Assessment(CCA). It has to do this because it needs to clear all traffic to ensure that no traffic is present when new traffic is established. When the PAN is sure there is no other transmissions it will start the new transmission of its own. The CCA is also a method that is called measuring spectral energy in a frequency channel of choice. This Method can then detect if any signal is active.

When the device tries to send a signal for the first time it has to change mode from CCA to Receive mode so that it can measure the energy level and estimate the prefered channel. This method is called Energy Detection (ED). The ID is not a decoder of the signal itself. It is more like a decoder of the

energy level of the signal. It will then try to see if it is an IEEE 802.15.4 signal. If it's already an IEEE 802.15.4 signal it will not do anything.

There is an alternative way to declare a channel free of transmission. It is called Carrier Sense (CS). This method is a bit more advanced and can declare a channel in use even if it is not in use. This is used to force the PAN to take the desired channel.

## Beacon and Non-Beacon

When the devices wants to access a channel there are two different ways to do that. There are contention-based(Non-beacon) and contention-free(Beacon) networks.
When devices in a contention based network tries to transmit messages in the same channel frequency it uses the CSMA-CA to evaluate if the channel is clear and the device will then transmit. That simply means first come - first served. If the channel is clear the device will start transmitting.

When the contention free protocol is trying to connect to other devices it uses a more strict method called GTS (Guaranteed time slot). This function is to insure that all devices will get their own time to “speak”. Meaning that the PAN will allocate time slots and decide who gets to speak at a certain time. This method doesn´t use the CSMA-CA by default, because of GTS. However, if a device in a contention-free network needs to speak outside its GTS it will use CSMA-CA.

| **Beacon** | **Non-Beacon** |
|---|---|
| Contention-free (does not have to listen in allocated time slot) | Contention based (first come - first serve, must listen before talking) |
| Uses beacons to synchronize clocks and allocate time slots (GTS) | Devices use CSMA-CA to know if they can transmit without disrupting channels. |
| Uses CSMA-CA (channel listening) if transmission needs to happen outside the Guaranteed Time Slot (GTS) | Does not synchronize or use time slots |
| Devices must wake up often and synchronize | Only needs to send when it needs to |
| Less battery life uses more data | Longer battery life and less data |
| Beacons can be used to indicate that messages are ready for a device, the device can then request the data if it’s ready. | Devices need to request data themselves, hence the coordinator may have data ready before the device requests it. |
| Devices are kept alive or awoken more often but in turn are faster to update. | Not as agile / responsive as Beacon-enabled |

- **Extended information**

  - **Device range**
    Devices in the IEEE 802.15.4 standard are limited to a maximum communication range of a 100 meters within line of sight. Typically vendors will advertise maximum lengths as 10-50 meters.

  - **Power consumption/battery life**
    Zigbee FFDs require the most power, and some of these will make sense to place in power plug whereas some will be battery-driven. RFD's (end-devices) are typically the least power hungry devices and will often be in sleep mode. Some RFDs will be able to harvest power from inertia created when flipping a switch or general movement of the device or near the device. This gives life to devices that can operate infinitely (until they break), without ever having to replace batteries or power them.

  - **Data transmission**
    Zigbee can do the data transmission on 10–100 meters line-of-sight, depending on power output and environmental characteristics.

  - **Security**
    Zigbee security is provided with tons of secure features. From secure key establishment to secure device management. The encryption/decryption feature of zigbee can be optionally protected with the security to provide data confidentiality, data authentication and data integrity. Its encryption method is an minor variation of AES (Advanced Encryption Standard) with a modified CCM mode (Counter with CBC-MAC). Even with its many secure features, it also have vulnerabilities in the implementation and protocol stages. Such like: Sending security headers in clear text, lack of DDoS Protection Mechanisms and privacy issues.

  - **Network Layer**
    In Zigbee, the network layer deals with formation of the network, allocation of addresses, adding/removing devices and routing of data packets. The network layer does not rely on a pre-existing infrastructure as in other, more common network types. It uses the AODV (**A**d-hoc **O**n-demand **D**istance **V**ector) routing protocol. Zigbee nodes participates in routing by forwarding packets for each other.

    If the destination is already in the source device's routing table or neighbor table it will forward the packet to the destination because it already knows it. In case it does not know the destination, the source device will broadcast "route requests" to its neighbors to find the destination device. Once the destination device is found, the destination device will send a unicast "route reply" following the lowest cost path back to the source.

# API Implementation

API implementation explains how a programmer can build, configure and implemented Zigbee in their code. In this case, these examples only shows their level of abstract to prove that Zigbee exists on any plane of abstracts in programming. The first example is arduino code which is mainly C language and its abstract level is high.

The second example is python that proves that you can program zigbee in a low abstract level and more human-readable code than arduino code. Zigbee is mainly focused on a mix of hardware and software programming, since it's a protocol that don't "normally" exists on every computer, hence one would have to require the hardware before started on programming.

- **arduino example**

```arduino
const int ledPin = 13; // the pin that the LED is attached to
int incomingByte;      // a variable to read incoming serial data into

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // see if there's incoming serial data:
  if (Serial.available() > 0) {
    // read the oldest byte in the serial buffer:
    incomingByte = Serial.read();
    // if it's a capital H (ASCII 72), turn on the LED:
    if (incomingByte == 'H') {
      digitalWrite(ledPin, HIGH);
    }
    // if it's an L (ASCII 76) turn off the LED:
    if (incomingByte == 'L') {
      digitalWrite(ledPin, LOW);
    }
  }
}
```

- **python example**

```python
#! /usr/bin/python

from xbee.thread import XBee
import serial

def main():
    """
    Sends an API AT command to read the lower-order address bits from
    an XBee Series 1 and looks for a response
    """
    try:

        # Open serial port
        ser = serial.Serial('/dev/ttyUSB0', 9600)

        # Create XBee Series 1 object
        xbee = XBee(ser)

        # Send AT packet
        xbee.send('at', frame_id='A', command='DH')

        # Wait for response
        response = xbee.wait_read_frame()
        print response

        # Send AT packet
        xbee.send('at', frame_id='B', command='DL')

        # Wait for response
        response = xbee.wait_read_frame()
        print response

        # Send AT packet
        xbee.send('at', frame_id='C', command='MY')

        # Wait for response
        response = xbee.wait_read_frame()
        print response

        # Send AT packet
        xbee.send('at', frame_id='D', command='CE')

        # Wait for response
        response = xbee.wait_read_frame()
        print response
    except KeyboardInterrupt:
        pass
    finally:
        ser.close()

if __name__ == '__main__':
    main()
```

# Source documentation/Links

- http://www.springer.com/cda/content/document/cda_downloaddocument/9783319478050-c2.pdf?SGWID=0-0-45-1593332-p180328094
- https://www.pocket-lint.com/smart-home/news/129857-what-is-zigbee-and-why-is-it-important-for-your-smart-home
- https://www.arduino.cc/en/Guide/ArduinoXbeeShield
- http://docs.smartthings.com/en/latest/device-type-developers-guide/zigbee-example.html
- http://www.zigbee.org/zigbee-for-developers/zigbee-3-0/
- https://research.kudelskisecurity.com/2017/11/08/zigbee-security-basics-part-2/
- https://research.kudelskisecurity.com/2017/11/21/zigbee-security-basics-part-3/
- https://www.norwegiancreations.com/2013/10/arduino-tutorial-1-lets-make-xbee-talk/
- http://www.tmrfindia.org/ijcsa/v10i22.pdf
- https://www.safaribooksonline.com/library/view/zigbee-wireless-networks/9780080558479/