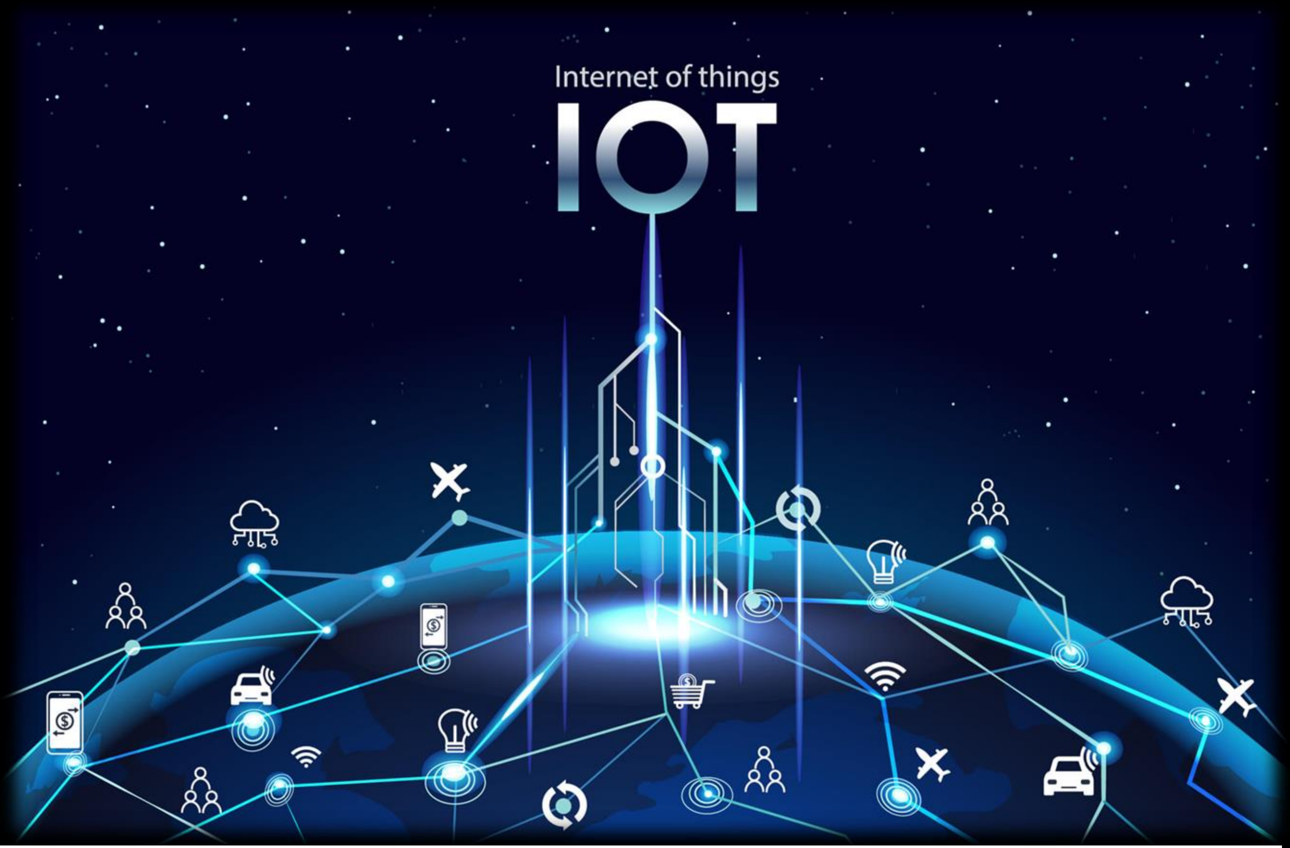


Learn Fastly the IoT

Cellular and Non Cellular IoT technology

FIRST EDITION

YEAR : 2019

About me

Hassan SALLOUM, I has been fascinated by computers and electronics from my very first programming project, which it was a real electro-mechanic billiard table for handicapped peoples.

I has master's degree in real time embedded system and master degree in computer science and telecommunication.

I also has 2 years of experience in the telecommunications domain (RAN), and I am now a platform IoT device tester and I has professional skills on Arduino, PI, and FPGA and am working now on the Nucleo board.

Also I have two great publish on scribd one about Decoding radio message of the layer 3 in module phone and another publish in the utility of the AMDEC tools.

Resume

In this research we will explain the different communication technology that we can be used to build our IoT network, and more focusing on the ZigBee, 6LowPAN, LoRaWAN and SigFox technology.

The first chapter of the research contains an overview about the IoT and its different domain application and characteristics of the IoT embedded device and the weak security points of IoT.

In chapter 2 we describe and explain the short range technology and in chapter 3 we explain the long range IoT technology also we will discuss for both the architecture and system functionality to their protocols stack.

Table of content

Chapter 1. Introduction to IoT

The Internet of Things (IoT) is the network of physical objects (Devices) or "things" embedded with electronics, software, sensors, and network connectivity, which enables these objects to collect and exchange data across existing network infrastructure, creating opportunities for more direct integration between the physical world and computer-based systems, and resulting in improved efficiency, accuracy and economic benefit.

Practically any object around you could be part of the Internet of Things, if you give it sensors, actuators, and a little bit of intelligence. Things in the Internet of Things include elements of two kinds: software and hardware. Software is a generic name given to the programs run by computers in all the Internet of Things devices.

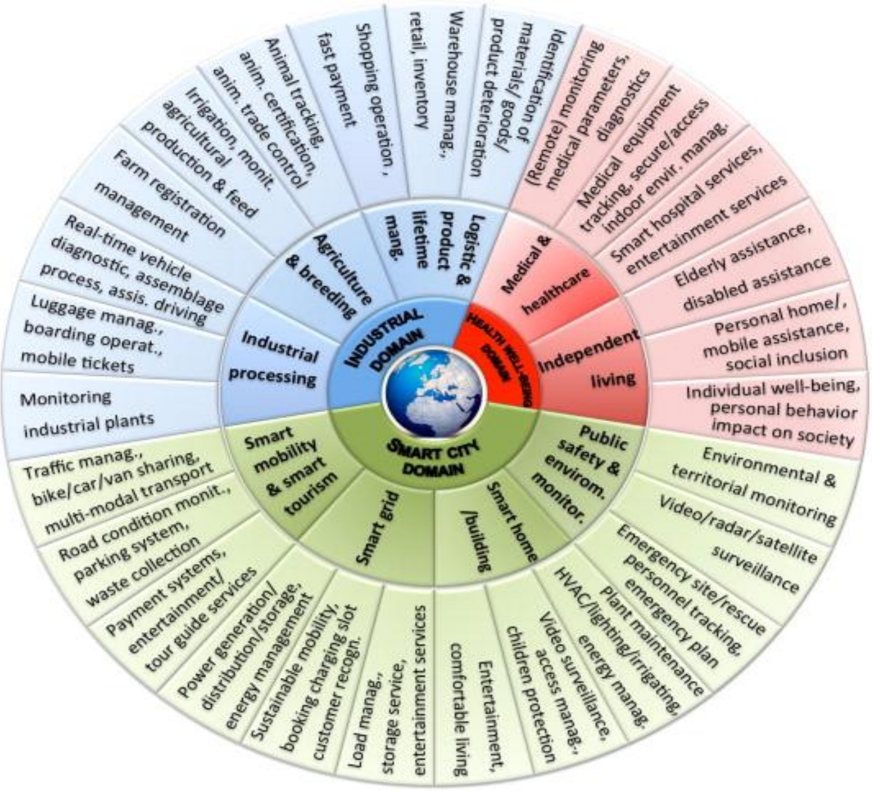

INDUSTRIAL DOMAIN
HEALTH WELL-BEING DOMAIN
SMART CITY DOMAIN
Agriculture & breeding
Logistic & product lifetime mang.
Industrial processing
Medical & healthcare
Independent living
Public safety & envirom. monitor.
Smart home /building
Smart grid
Smart mobility & smart tourism
Shopping operation, fast payment
Warehouse manag., retail, inventory
Identification of materials/ goods/ product deterioration
(Remote) monitoring medical parameters, diagnostics
Medical equipment tracking, secure/access indoor envir. manag.
Smart hospital services, entertainment services
Elderly assistance, disabled assistance
Personal home/, mobile assistance, social inclusion
Individual well-being, personal behavior impact on society
Environmental & territorial monitoring
Video/radar/satellite surveillance
Emergency site/rescue personnel tracking, emergency plan
Plant maintenance
HVAC/lighting/irrigating, energy manag.
Video surveillance, access manag., children protection
Entertainment, comfortable living
entertainment services
Load manag., storage service,
Sustainable mobility, booking charging slot customer recogn.
Power generation/ distribution/storage, energy management
Payment systems, entertainment/ tour guide services
Road condition monit., parking system, waste collection
Traffic manag., bike/car/van sharing, multi-modal transport
Monitoring industrial plants
Luggage manag., boarding operat., mobile tickets
Real-time vehicle diagnostic, assemblage process, assis. driving
Farm registration management
Irrigation, monit. agricultural production & feed
Animal tracking, anim. certification, anim. trade control

Figure 1: IoT domains and sectors

1.1 IoT technology

Many communication technologies are well known such as WiFi, Bluetooth, ZigBee and 2G/3G/4G cellular, but there are also several new emerging networking options such as Thread as an alternative for home automation applications, and Whitespace TV technologies being implemented in major cities for wider area IoT-based use cases. Depending on the application, factors such as range, data requirements, security and power demands and battery life will dictate the choice of one or some form of combination of technologies. These are some of the major communication technologies on offer to developers.

Short range communication technology	Long Range communication technology (LPWAN)	
E.g. Bluetooth, NFC, ZigBee, WIFI, 6LowPAN, RFID	Non Cellular LPWAN	Cellular LPWAN
	LoRaWAN, SigFox, Ingenue, Nwave, Weightless	Cat-1, Cat-0, Cat-M1, Cat-M2, EC GSM

Table 1: Different IoT technology

1.2 IoT device architecture

The IoT is expected be a worldwide network comprising, by 2020, billions of devices. This gigantic number of devices, pervasively deployed, will be characterized by their heterogeneity in terms of software and, in particular, hardware.

The main hardware components in a smart object are:

- **Communication module:** This gives the smart object its communication capabilities. It is typically either a radio transceiver with an antenna or a wired connection.
- **Microcontroller:** This gives the smart object its behavior. It is a small microprocessor that runs the software of the smart object.
- **Sensors or actuators:** These give the smart object a way to sense and interact with the physical world.
- **Power source:** This is needed because the smart object contains electrical circuits. The most common power source is a battery, but there are other examples as well, such as piezoelectric power sources, that provide power when a physical force is applied, or small solar cells that provide power when light shines on them.

1.3 IoT device OS

There is a lot of operating system used for the embedded devices like OpenWSN, TinyOS, FreeRTOS, TI-RTOS, RIOT and Contiki OS, ARM Mbed OS. But in this book we are highlighted only on two of them:

- **OpenWSN**

The OpenWSN project is an open-source implementation of a fully standards-based protocol stack for IoT networks. It was based on the new IEEE802.15.4e time-slotted channel-hopping standard.

IEEE802.15.4e, coupled with IoT standards such as 6LoWPAN, RPL and CoAP, enables ultra-low-power and highly reliable mesh networks that are fully integrated into the Internet.

OpenWSN has been ported to numerous commercial available platforms 16-bit micro-controllers and 32-bitCortex-M architectures. Also it offers a free open-source implementation of a protocol stack and the surrounding debugging and integration tools, thereby contributing to the overall goal of promoting the use of low-power wireless mesh networks.

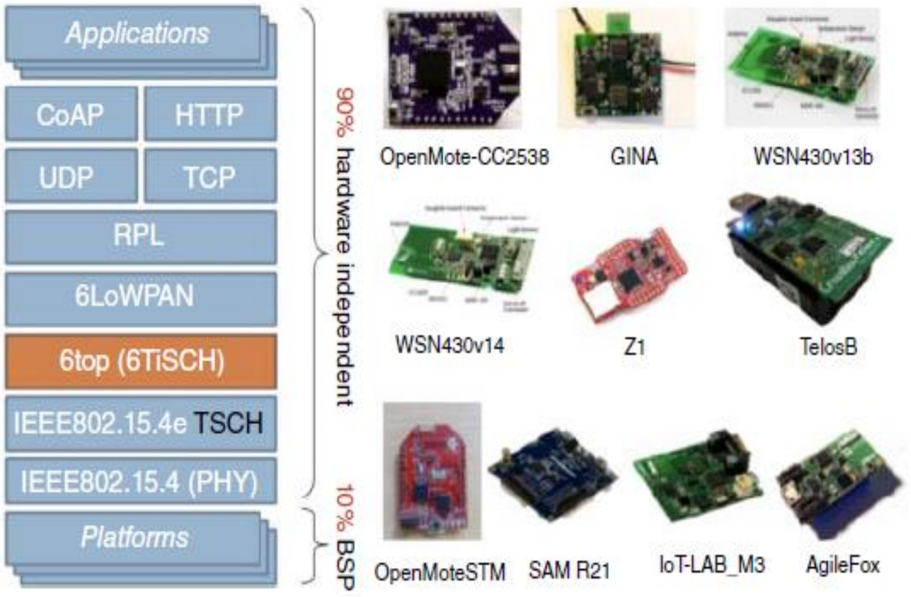


Figure 2: OpenWSN protocol stack highlighting hardware-independent modules and supported hardware platforms

- **FreeRTOS**

FreeRTOS is a real-time operating system kernel for embedded devices designed to be small and simple. It been ported to 35 micro-controllers and it is distributed under the GPL with an optional exception. The exception permits users' proprietary code to remain closed source while maintaining the kernel itself as open source, thereby facilitating the use of FreeRTOS in proprietary applications.

In order to make the code readable, easy to port, and maintainable, it is written mostly in C, (but some assembly functions have been included to support architecture-specific scheduler routines). It provides methods for multiple threads or tasks, mutexes, semaphores and software timers.

1.4 IoT platform

An IoT platforms originated in the form of IoT middleware, which purpose was to function as a mediator between the hardware and application layers.

- Its primary tasks included data collection from the devices over different protocols and network topologies,
- Remote device configuration and control,
- Device management, and over-the-air firmware updates,
- Modern IoT platforms provide components for frontend and analytics, on-device data processing, and cloud-based deployment.

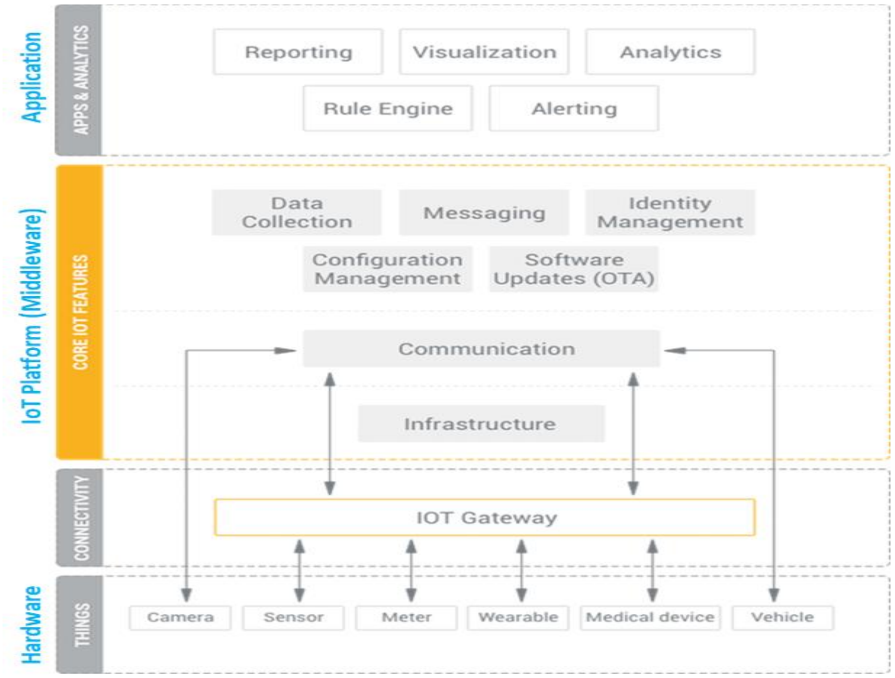


Figure 3: IoT platform layers

In the four typical layers of the IoT stack, which are:

- Things,
- Connectivity,
- Core IoT features,
- Applications & analytics,

Your devices connect to the platform, which sits in the cloud or in your on-premises data center, either directly or by using an IoT gateway. A gateway comes useful whenever your endpoints aren't capable of direct cloud communication IoT device architecture.

1.5 Security challenges

The security challenges derive from the very nature of smart objects and the use of standard protocols. Garcia-Morchon *et al*. summarize security threats in the IoT as follows:

1. Cloning of smart objects by unauthorized manufacturers.
2. Malicious substitution of smart things during installation firmware replacement attacks.
3. Extraction of security parameters (smart things may be physically unprotected.
4. Eavesdropping attacks if communication channels are not adequately protected.
5. Man-in-the-middle attacks during key exchange.
6. Routing attacks.
7. Denial-of-service attacks.
8. Privacy threats.

Threats 1–4 are related to the physical nature of smart objects, which are typically deployed in public areas and cannot be constantly supervised, thus leading to potential security problems.

Threats 5–8 are examples of security issues arising from the need for objects to communicate with each other. Finally, Threat 5.1 is related to the fact that smart objects might deal with personal or sensitive data, which, if intercepted by unauthorized parties, might create ethical and privacy problems.

While it is possible to cope with issues arising from the physical nature of objects only by adopting safe supply and installation measures, such as avoiding untrusted manufacturers and installers, and by trying to protect smart objects in safe places, all other security threats can be tackled by adopting means such as secure communication protocols and cryptographic algorithms. These measures enforce the following basic security properties:

- **Confidentiality:** transmitted data can be read only by the communication endpoints.
- **Availability:** the communication endpoints can always be reached and cannot be made inaccessible.
- **Integrity:** received data are not tampered with during transmission, if this does not happen, then any change can be detected.
- **Authenticity:** data senders can always be verified and data receivers cannot be spoofed.

1.6 Note before we start

1.6.1 Modulation

- **Why we need modulation**

The modulation is used because some data signals are not always suitable for direct transmission, but the modulated signal may be more suitable.

For example, let's consider a channel that essentially acts like a bandpass filter where both the lowest frequency and the highest frequency components are attenuated, with transmission only being practical over some intermediate frequency range. If we can't send low-frequency signals, then we need to shift our signal up the frequency ladder.

Another reason to modulate a signal is to allow the use of a smaller antenna. A baseband (low frequency) signal would need a huge antenna because in order to be efficient, the antenna needs to be about 1/10th the length of the wavelength. Modulation shifts the baseband signal up to a much higher frequency, which has much smaller wavelengths and allows the use of a much smaller antenna.

- **What is modulation**

So, the **Modulation** is a process of mixing a data signal with a sinusoid (**carrier signal: F**) to produce a new signal. This new signal, conceivably, will have certain benefits over an un-modulated signal. Mixing of low frequency signal with high frequency carrier signal is called modulation.

$F(t) = A \sin(\omega t + \varphi)$

We can see that this sinusoid has 3 parameters that can be altered, to affect the shape of the graph.

- The first term, A, is called the magnitude, or amplitude of the sinusoid.
- The next term, ω is known as the frequency,
- And the last term, φ is known as the phase angle. All 3 parameters can be altered to transmit data.

- **Modulation types**

- **Amplitude modulation:** the amplitude of the carrier signal is modulated (changed) in proportion to the message signal while the frequency and phase are kept constant (Application of ASK: used to transmit digital data over optical fiber).
- **Frequency modulation:** the frequency of the carrier signal is modulated (changed) in proportion to the message signal while the amplitude and phase are kept constant. (Application FSK: over voice lines, in high-freq. radio transmission, etc…).
- **Phase modulation:** the phase of the carrier signal is varied accordance to the low frequency of the message signal is known as phase modulation.

- **Advantage and disadvantage of modulation**

	FSK	ASK	PSK
Advantage	less susceptible to errors than ASK receiver looks for specific frequency changes over a number of intervals, so voltage (noise) spikes can be ignored	simplicity	less susceptible to errors than ASK, while it requires/occupies the same bandwidth as ASK, more efficient use of bandwidth (higher data-rate) are possible, compared to FSK !!!
Disadvantage	spectrum is 2 x ASK spectrum	is very susceptible to noise interference noise usually (only) affects the amplitude, therefore ASK is the modulation technique most affected by noise	more complex signal detection / recovery process, than in ASK and FSK

Table 2: advantage and disadvantage of different modulation technic

1.6.2 Start vs. Mesh topology

- **Start topology**

A star network is typically a central coordinator or concentrator acts as the conduit for all network traffic. All network transmissions are routed via the central coordinator:

- Most common form of network topology for power constrained end-point nodes.
- Is relatively simple to implement.
- Minimize the amount of network traffic. For a network that is not link constrained only 3 devices and two links are involved in any communications between two nodes.
- A disadvantage of this topology is that failure of the coordinator will disable all network communications.

- **Mesh topology**

Mesh networks are typically implemented in circumstances where the nodes are not power constrained.

- In a mesh network data propagates through the network via every node.
- Networks typically employ look-up tables or are self-routing.
- The ability to "self-heal" and reconfigure themselves in the event of a loss of connectivity to a node or group of nodes.
- A disadvantage of this topology is the relatively increased complexity (Increase traffic) over traditional star networks and an increase in network traffic due to the inherent in-built redundancy of the network.

Chapter 2. Short range technology

2.1 Bluetooth

An important short-range communications technology is of course Bluetooth, which is expected to be key for wearable products in particular, again connecting to the IoT albeit probably via a smartphone in many cases. The new Bluetooth Low-Energy (BLE) – or Bluetooth Smart, as it is now branded – is a significant protocol for IoT applications.

However, Smart/BLE is not really designed for file transfer and is more suitable for small chunks of data.

Devices that employ Bluetooth Smart features incorporate the Bluetooth Core Specification Version 4.0 (or higher – the latest is version 4.2 announced in late 2014) with a combined basic-data-rate and low-energy core configuration for a RF transceiver, baseband and protocol stack.

Importantly, version 4.2 via its Internet Protocol Support Profile will allow Bluetooth Smart sensors to access the Internet directly via 6LoWPAN connectivity (more on this below). This IP connectivity makes it possible to use existing IP infrastructure to manage Bluetooth Smart 'edge' devices. More information on Bluetooth 4.2 is available here and a wide range of Bluetooth modules are available from RS.

- Standard: Bluetooth 4.2 core specification.
- Frequency: 2.4GHz (ISM).
- Range: 50-150m (Smart/BLE).
- Data Rates: 1Mbps (Smart/BLE).

2.2 RFID

Radio-Frequency Identification (RFID) is the use of radio waves to read and capture information stored on a tag attached to an object. A tag can be read from up to several feet away and does not need to be within direct line-of-sight of the reader to be tracked.

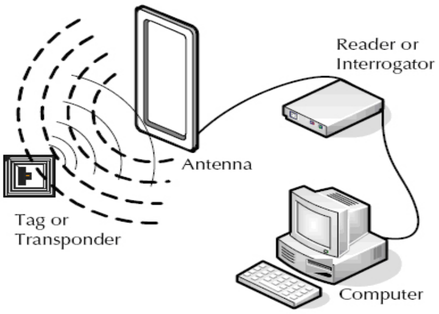


Figure 4: RFID system connection

- **How RFID system work?**

A RFID system is made up of two parts: a tag and a reader.

The RFID tags have two parts: a microchip that stores and processes information, and an antenna to receive and transmit a signal. The tag contains the specific serial number for one specific object.

There are two types of RFID tags: passive and battery powered. A passive RFID tag will use the interrogator's radio wave energy to relay its stored information back to the interrogator. A batter powered RFID tag is embedded with a small battery that powers the relay of information.

To read the information encoded on a tag, a two-way radio transmitter-receiver called an interrogator or reader emits a signal to the tag using an antenna. The tag responds with the information written in its memory bank. The interrogator will then transmit the read results to an RFID computer program.

2.3 ZigBee

ZigBee, has a large installed base of operation, although perhaps traditionally more in industrial settings. ZigBee PRO and ZigBee Remote Control (RF4CE), among other available ZigBee profiles, are based on the IEEE802.15.4 protocol.

ZigBee/RF4CE has some significant advantages in complex systems offering low-power operation, high security, robustness and high scalability with high node counts and is well positioned to take advantage of wireless control and sensor networks in M2M and IoT applications.

Zigbee supports several network topologies, however, the most commonly used configurations are star, mesh and cluster tree topologies.

In general and the most use in ZigBee is a mesh network, so each node in a ZigBee system can act as a wireless data endpoint or a repeater. Data travels from node to node until it reaches the router. It is designed for relatively low data-rate applications over a restricted area and within a 100m range such as in a home or building.

- Standard: ZigBee 3.0 based on IEEE802.15.4
- Frequency: 2.4GHz
- Range: 10-100m
- Data Rates: 250kbps

2.3.1 ZigBee architecture

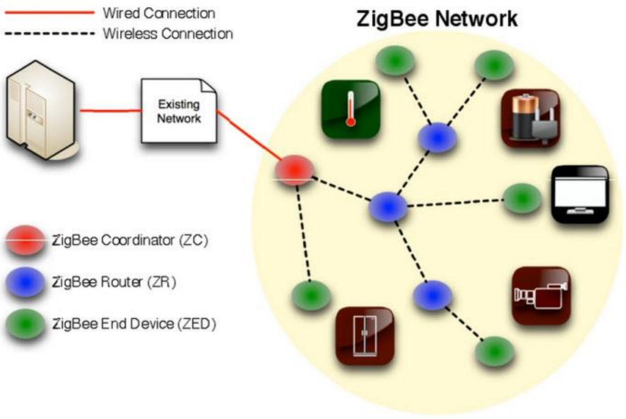


Figure 5: ZigBee system architecture

2.3.2 ZigBee protocol stack

The following figure depicts ZigBee protocol stack, which consists of four layers viz. PHY, MAC, network & security and application layer. The first two are covered in IEEE 802.15.4 WPAN standard and the latter two are covered in documents published by ZigBee alliance.

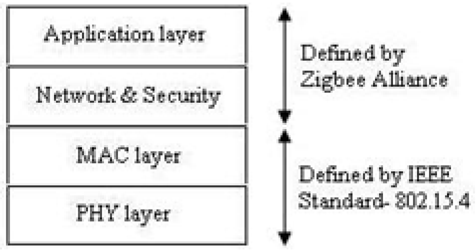

Application layer
Network & Security
MAC layer
PHY layer
Defined by Zigbee Alliance
Defined by IEEE Standard- 802.15.4

Figure 6: ZigBee protocol stack

2.3.2.1 Application Layer

The application support sublayer (APS) provides the services necessary for application objects (endpoints) and the ZigBee device object (ZDO) to interface with the network layer for data and management services. Some of the services provided by the APS to the application objects for data transfer are request, confirm, and response. Furthermore, the APS provides communication for applications by defining a unified communication structure (for example, a profile, cluster, or endpoint).

- **Application object (endpoint):** An application object defines input and output to the APS. For example, a switch that controls a light is the input from the application object, and the output is the light bulb condition. Each node can have 240 separate application objects. An application object may also be referred to as an endpoint (EP). an example of home control lighting.

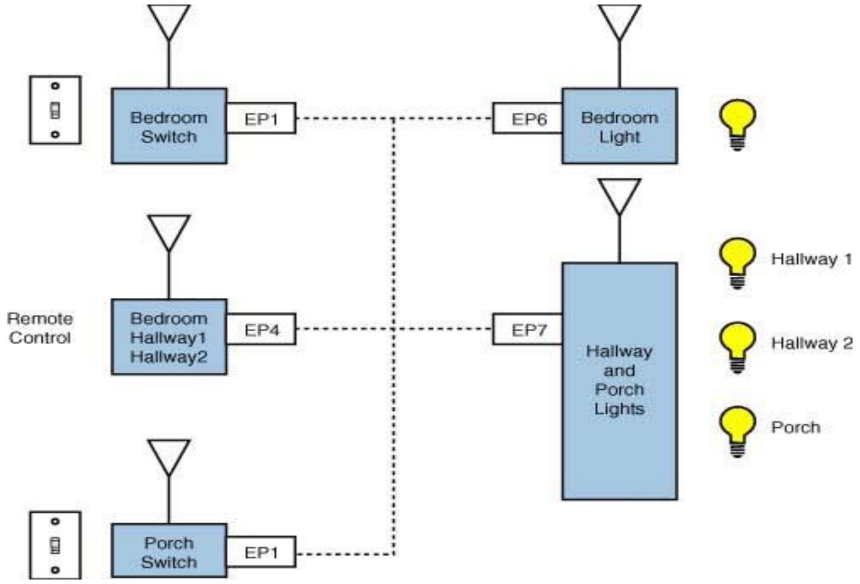


Figure 7: Simple demo explain the ZigBee functionality

- **ZigBee device object (ZDO):** A ZigBee device object performs control and management of application objects. The ZDO performs the overall device management tasks:
  - Determines the type of device in a network (for example, end device, router, or coordinator).
  - Initializes the APS, network layer, and security service provider.
  - Performs device and service discovery.
  - Initializes coordinator for establishing a network.
  - Security management.
  - Network management.
  - Binding management.
- **End node:** Each end node or end device can have multiple EPs. Each EP contains an application profile, such as home automation, and can be used to control multiple devices or a single device. More to the point, each EP defines the communication functions within a device. As shown in the up figure, the bedroom switch controls the bedroom light, and the remote control is used to control three lights: bedroom, hallway1, and hallway2.
- **ZigBee addressing mode:** ZigBee uses direct, group, and broadcast addressing for transmission of information. In direct addressing, two devices communicate directly with each other. This requires that the source device has both the address and endpoint of the destination device. Group addressing requires that the application assign a group membership to one or more devices. A packet is then transmitted to the group address in which the destination device lies. The broadcast address is used to send a packet to all devices in the network.

2.3.2.2 Network Layer

The main functions of the network layer are to enable the correct use of the MAC sublayer and provide a suitable interface for use by the next upper layer, namely the application layer.

The Network layer it deals with network functions such as connecting, disconnecting, and setting up networks. It will add a network, allocate addresses, and add/remove certain devices. This layer makes use of star, mesh and tree topologies. It adds an interface to the application layer.

The routing protocol used by the network layer is AODV to find the destination device.

- **Ad hoc On Demand Distance Vector (AODV)**

AODV broadcasts out a route request to all of its neighbors. The neighbors then broadcast the request to their neighbors and onward until the destination is reached. Once the destination is reached, it sends its route reply via unicast transmission following the lowest cost path back to the source. Once the source receives the reply, it will update its routing table for the destination address of the next hop in the path and the path cost.

- **Security Layer**

If security is enabled, ZigBee device will start up using a 128 bit AES encryption key. Devices having same security key can communicate on PAN.

How to obtain this key?
1. Pre-installation.
2. Key is received over the air during joining.

2.3.2.3 MAC Layer

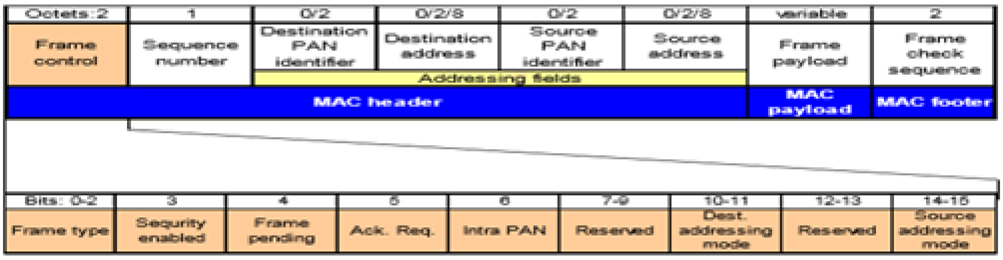
The Mac layer defined by 802.15.4, the functions of the MAC layer is:

- Access the network by using carrier-sense multiple access to avoid collision avoidance (CSMA/CA), to transmit beacon frames for synchronization, and to provide reliable transmission.
- Defines a frame format with things like MAC addresses etc.
- Defines network topologies which ZigBee builds upon and enhances at higher levels of the stack.

- **CSMA/CA**

The CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) issued by most wireless LANs in the ISM bands. The basic principle of CSMA/CA is listening before talking and Argument. This is an asynchronous message passing mechanism (connectionless), delivering the best energy service lacking bandwidth and latency guarantee. Its main advantage is that it is suited for network protocols such as TCP/IP and adapts quite well with the variable condition of traffic and is quite robust against interferences. But the CSMA/CA protocol can't directly detect collisions like Ethernet and only tries to avoid them.

The MAC frames are divided into following four major categories, which is used by ZigBee devices to establish connection to the PAN by exchanging system information: Beacon, Data frame, Acknowledgement, MAC command and Reserved.



Octets:2	1	0/2	0/2/8	0/2	0/2/8	variable	2
Frame control	Sequence number	Destination PAN identifier	Destination address	Source PAN identifier	Source address	Frame payload	Frame check sequence
		Addressing fields					
MAC header						MAC payload	MAC footer

Bits: 0-2	3	4	5	6	7-9	10-11	12-13	14-15
Frame type	Sequrity enabled	Frame pending	Ack. Req.	Intra PAN	Reserved	Dest. addressing mode	Reserved	Source addressing mode

Figure 8: Frame control field

2.3.2.4 Physical layer

Defined by 802.15.4 the PHY layer is responsible for the modulation, demodulation and physical transmission of packets over the air and handles various things needed for robust radio transmission in noisy, interference prone environments.

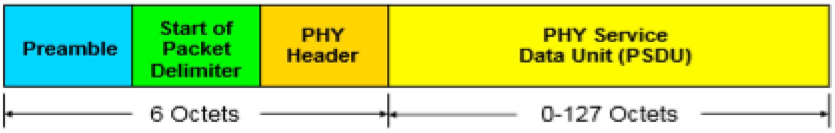


Figure 9: ZigBee packet

- **There are two physical layers in ZigBee:**

- 868/915 MHz uses BPSK modulation, raised cosine pulse shaping.
- And 2450 MHz, O-QPSK modulation.

- **The PHY layer is also responsible for the following tasks:**

- Enable/disable the radio transceiver.
- Link quality indication (LQI) for received packets.
- Energy detection (ED) within the current channel.
- Clear channel assessment (CCA).

- **BPSK modulation**

BPSK (also sometimes called PRK, phase reversal keying, or 2PSK) is the simplest form of phase shift keying (PSK), so in PSK the phase of carrier signal is varied to represent binary 1 or 0 where binary 1 = 0º phase, binary 0 = 180º (πrad) phase.

⇒ PSK is equivalent to multiplying carrier signal by +1 when the information is 1, and by -1 when the information is 0.

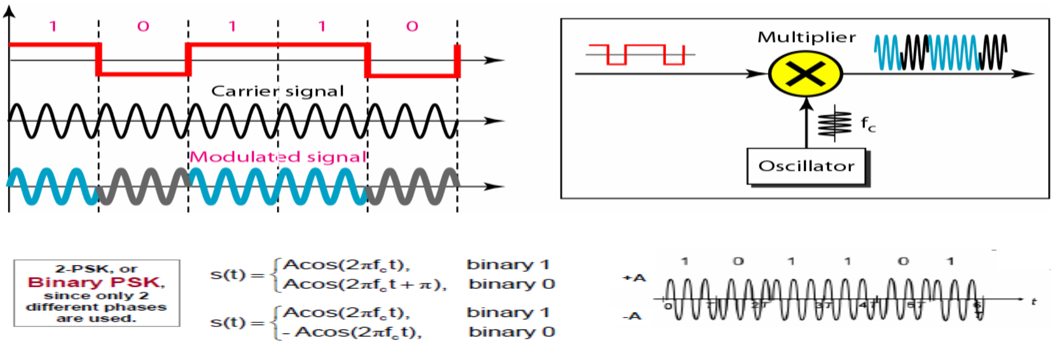

1
0
1
1
0
Carrier signal
Modulated signal
Multiplier
f_c
Oscillator
2-PSK, or
Binary PSK,
since only 2
different phases
are used.
$s(t) = \begin{cases} A\cos(2\pi f_c t), & \text{binary 1} \\ A\cos(2\pi f_c t + \pi), & \text{binary 0} \end{cases}$
$s(t) = \begin{cases} A\cos(2\pi f_c t), & \text{binary 1} \\ -A\cos(2\pi f_c t), & \text{binary 0} \end{cases}$
+A
-A
1 0 1 1 0 1
t

Figure 10: BPSK modulation scheme

- **O-QPSK modulation**

To understand the O-QPSK, we need to understand first the QPSK first,

So the Quadrature phase shift keying (QPSK) is type of phase shift keying and it’s a particularly interesting one because it actually transmits two bits per symbol and not one bit like BPSK. In other words, a QPSK symbol doesn’t represent 0 or 1—it represents 00, 01, 10, or 11 (to get High data rate and high Bandwidth).

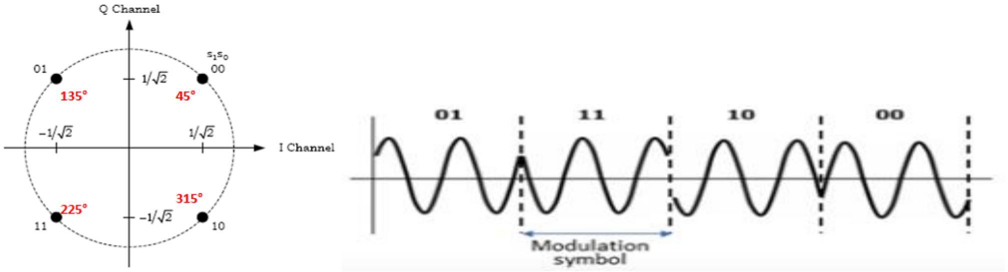


Figure 11: O_QPSK modulation scheme

In QPSK, the carrier varies in terms of phase, and we have 360° of phase to work with and four phase states, and thus the separation should be 360°/4 = 90°. So our four QPSK phase shifts are 45°, 135°, 225°, and 315°. And that it makes sense to seek maximum separation between the four phase options, so that the receiver has less difficulty distinguishing one state from another.

- **What is the O-QPSK?**

The offset quadrature phase-shift keying (OQPSK) also known as Staggered quadrature phase-shift keying (SQPSK), is a method of phase-shift keying (PSK) in which the signal carrier-wave phase transition is always 90 degrees or 1/4 cycle at a time. A phase shift of 90 degrees is known as phase quadrature.

In SQPSK there are four possible states: 0, +90, -90 and 180 degrees. The average magnitude of the phase transitions is smaller with SQPSK than with conventional QPSK. The result of the smaller average phase "jump" is an improved signal-to-noise ratio (SNR) and a reduced error rate.

2.4 6LowPAN

The short-range, low power networks, sometimes called ‘last 100 meters of connectivity’ represent a large fraction of the potential number of things, e.g. IPv6 over low-power wireless personal area networks (6LoWPAN).

Frequency band	2400-2483.5Mhz (Worldwide)	900-929 Mhz (North America)	868-868.3Mhz (Europe)
Number of channels	16 channels for 2.4 Ghz band	10 channels for 915Mhz band	1 channel for 868.3 band
Channel bandwidth	5Mhz for 2.4Ghz band	2Mhz for 915Mhz band	
Maximum data rate	250kbps for 2.4 Ghz band	40kbps for 915Mhz	20kbps for 868.3 band
Protocol data unit	6 bytes header + 127 bytes SDU		
Channel coding	Direct sequence spread spectrum (DSSS)		
Channel modulation	O-QPSK for 2.4 Ghz band	BPSK for 915Mhz and 868.6Mhz band	
Receiver sensitivity	-85dBm for 2.4Ghz band	-92dBm for 915Mhz and 868.3Mhz band	
Transmission range	10-100m		
Battery lifetime	1-2 years		

Table 3: 6LowPAN Features

2.4.1 6LowPAN architecture

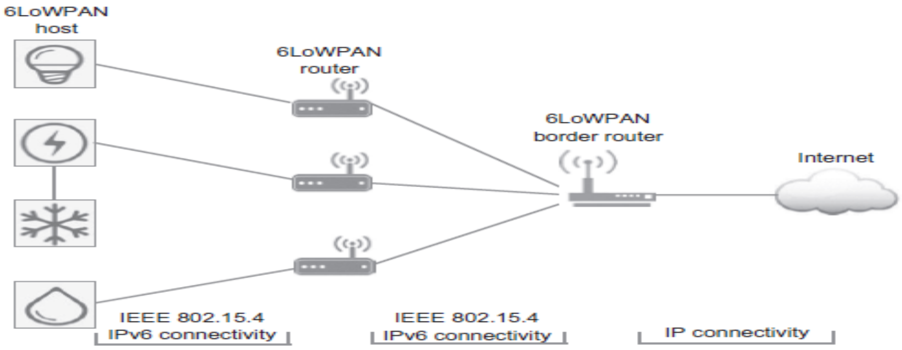


Figure 12: 6LowPAN system communication

The 6LoWPAN network architecture contains three elements: host node, router node and edge router.

- The hosts can sense the physical environment and actuate devices.
- The routers are intermediate nodes that forward data packets from the hosts to the edge routers or to a destination inside the 6LoWPAN network. The connection among 6LoWPAN elements is implemented via IPv6 over IEEE 802.15.4.
- The edge routers provide interconnection and traffic management (e.g. Neighbor Discovery (ND) and handling IPv4 interconnectivity) between 6LoWPAN network and other IP networks (typically the Internet).

Sending and receiving packets between 6LoWPAN elements and IP nodes in other networks occur in an end-to-end scheme similar to any IP nodes where each 6LoWPAN element is identified by a unique IPv6 address.

- In 6LoWPAN networks, host (end) node can communicate with other host nodes directly, also, this eliminates the need for a routing topology as each end node knows where to forward its packets directly. Whereas, in LoRaWAN and SigFox, end nodes can communicate only with gateways and they cannot send packets directly to other end nodes.
- In 6LoWPAN networks, the distance is 10 –100 m distances between end nodes and intermediate nodes, whereas for LoRaWAN and SigFox, it is 5–50 km. This means that many 6LoWPAN nodes have to be distributed to cover a small area (i.e. high node density).

2.4.2 6LowPAN protocol stack

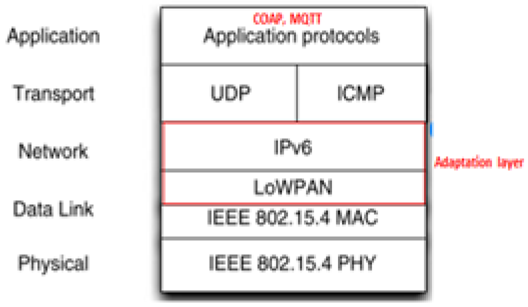


Figure 13 : 6LowPAN protocol stack

2.4.2.1 Application layer

The IoT needs standard protocols. Two of the most promising for small devices are MQTT and CoAP.

Both MQTT and CoAP: Are open standards, Are better suited to constrained environments than HTTP, Provide mechanisms for asynchronous communication, Run on IP, Have a range of implementations, MQTT gives flexibility in communication patterns and acts purely as a pipe for binary data and CoAP is designed for interoperability with the web.

- **COAP**

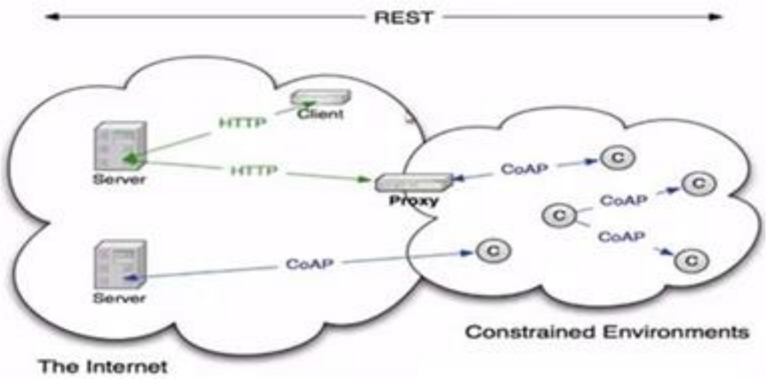


Figure 14 : COAP system interaction

- Like HTTP, CoAP is a document transfer protocol. Unlike HTTP, CoAP is designed for the needs of constrained devices.
- CoAP is designed to: interoperate with HTTP and the RESTful web at large through simple proxies.
- CoAP follows a client/server model: Clients make requests to servers, servers send back responses. Clients may GET, PUT, POST and DELETE resources.
- CoAP runs over UDP, not TCP: Clients and servers communicate through connectionless datagrams. Retries and reordering are implemented in the application stack. Removing the need for TCP may allow full IP networking in small microcontrollers. CoAP allows UDP broadcast and multicast to be used for addressing.
- In CoAP, a sensor node is typically a server, not a client (though it may be both): The sensor (or actuator) provides resources which can be accessed by clients to read or alter the state of the sensor.
- As CoAP sensors are servers: they must be able to receive inbound packets. To function properly behind NAT, a device may first send a request out to the server, as is done in LWM2M, allowing the router to associate the two. Although CoAP does not require IPv6, it is easiest used in IP environments where devices are directly routable.

- CoAP packets: are much smaller than HTTP TCP flows. Bitfields and mappings from strings to integers are used extensively to save space. Packets are simple to generate and can be parsed in place without consuming extra RAM in constrained devices.
- CoAP message format: consists of 4 bytes header followed by token value (from 0 to 8 bytes). The table below mentions header which consists of 4 bytes i.e. 32 bits.

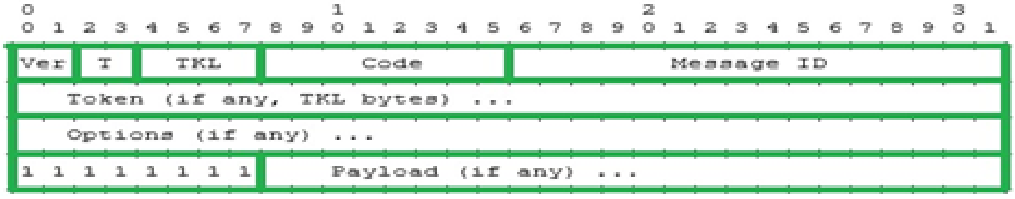


Figure 15 : COAP message format

CoAP message header	Description
Ver	2 bit unsigned integer. It mentions CoAP versions number. Set to one
T	2 bit unsigned integer. Indicates message type: confirmable(0), non-confirmable(1), ACk(2) or RESET(3)
TKL	4 bit unsigned integer. Indicates length of token (0 to 8 bytes)
Code	8 bit unsigned integer, its used split into two parts viz. 3 bit class (MSBs) and 5bits detail (LSBs)
Message ID	16 bits unsigned integer. Used for matching responses. Used to detect message duplication

Table 4: COAP message header description

Code-Indicates: **0 GET** (Retrieves the information corresponding to the resource in request URI), **1 POST** (Similar to GET, if additional information is sent in body of request, rather than URI -POST is used), **2 PUT** (Resource identified by the request URI be updated), **3 DELETE** (Delete a resource on the server).

- **MQTT**

- MQTT is a publish/subscribe messaging protocol designed for lightweight M2M communications. It was originally developed by IBM and is now an open standard.
- MQTT has a client/server model, where every sensor is a client and connects to a server, known as a broker, over TCP.
- MQTT brokers may require username and password authentication from clients to connect. To ensure privacy, the TCP connection may be encrypted with SSL/TLS.
- MQTT is message oriented. Every message is a discrete chunk of data, opaque to the broker.
- Every message is published to an address, known as a topic. Clients may subscribe to multiple topics. Every client subscribed to a topic receives every message published to the topic.

For example, imagine a simple network with three clients and a central broker.

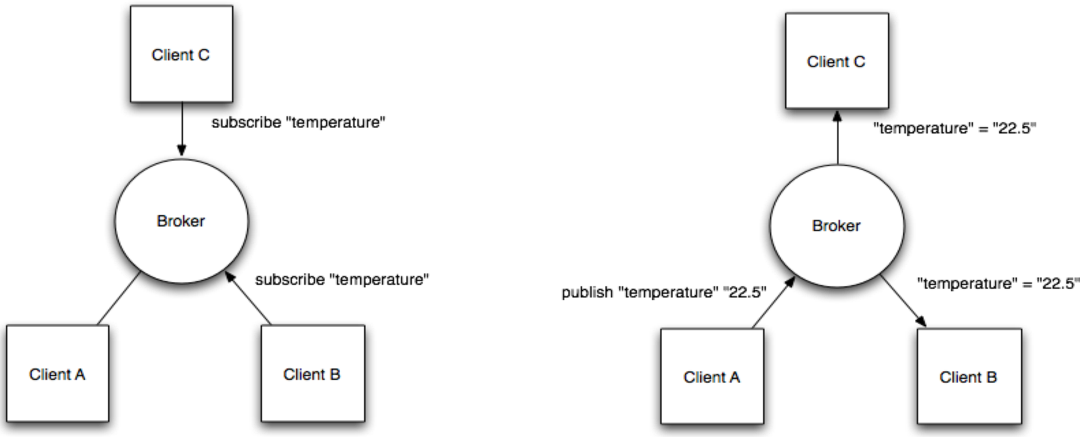


Figure 16: MQTT system communication

- All three clients open TCP connections with the broker. Clients B and C subscribe to the topic temperature. At a later time, Client A publishes a value of 22.5 for topic temperature. The broker forwards the message to all subscribed clients.
- The publisher subscriber model allows MQTT clients to communicate one-to-one, one-to-many and many-to-one.

2.4.2.2 Transport layer

- **UDP**

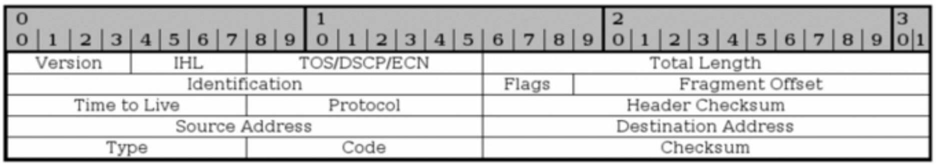
User datagram protocol is an open systems interconnection (OSI) transport layer protocol for client- server network applications. UDP uses a simple transmission model but does not employ handshaking dialogs for reliability, ordering and data integrity. The protocol assumes that error-checking and correction is not required, thus avoiding processing at the network interface level.

UDP is widely used in video conferencing and real-time computer games. UDP network traffic is organized in the form of datagrams, which comprise one message units. The first eight bytes of a datagram contain header information (source, destination, length, checksum), while the remaining bytes contain message data. A UDP datagram header contains four fields of two bytes each: Source port number, Destination port number, Datagram size and Checksum.

- **ICMP**

The **Internet Control Message Protocol** (**ICMP**) is a supporting protocol in the Internet protocol suite. It is used by network devices, including routers, to send error messages and operational information indicating, for example, that a requested service is not available or that a host or router could not be reached.

ICMP differs from transport protocols such as TCP and UDP in that it is not typically used to exchange data between systems, nor is it regularly employed by end-user network applications (with the exception of some diagnostic tools like ping and traceroute).



<table>
<tr><td colspan="10">0</td><td colspan="10">1</td><td colspan="10">2</td><td colspan="2">3</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td></tr>
<tr><td colspan="4">Version</td><td colspan="4">IHL</td><td colspan="8">TOS/DSCP/ECN</td><td colspan="16">Total Length</td></tr>
<tr><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment Offset</td></tr>
<tr><td colspan="8">Time to Live</td><td colspan="8">Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><td colspan="16">Source Address</td><td colspan="16">Destination Address</td></tr>
<tr><td colspan="8">Type</td><td colspan="8">Code</td><td colspan="16">Checksum</td></tr>
</table>

Figure 17: ICMP message header

2.4.2.3 Network layer

The main considerations of this layer are addressing and routing protocols.

Ipv6 header is added in this field, route over routing decision is done in network layer.

RPL routing protocol for low power and lossy network can be used.IPv6 extension header will have routing details.

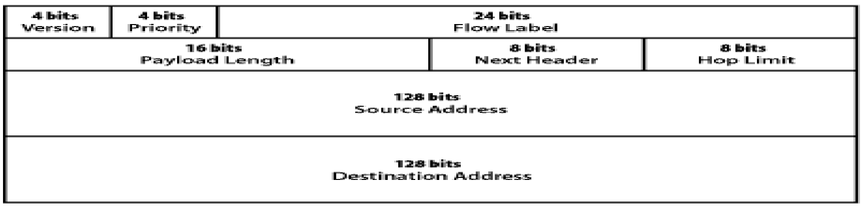


Figure 18: RPL message header

- **RPL protocol**

RPL it is a routing protocol for wireless networks with low power consumption and generally susceptible to packet loss. It is a proactive protocol based on distance vectors and opera on IEEE 802.15.4, optimized for multi-hop and many-to-one communication, but also supports one-to-one messages.

RPL creates a topology similar to a tree (DAG or directed acyclic graph). Each node within the network has an assigned rank (Rank), which increases as the teams move away from the root node (DODAG). The nodes resend packets using the lowest range as the route selection criteria.

Three types of packages are defined ICMPv6:

- DIS (information request DODAG): Used to request information from nearby DODAG, analogous to router request messages used to discover existing networks.
- DIO (object of information of the DAG): Message that shares information from the DAG, sent in response to DIS messages, as well as used periodically to refresh the information of the nodes on the topology of the network.
- DAO (object of update to the destination): Sent in the direction of the DODAG, it is a message sent by the teams to update the information of their "parent" nodes throughout the DAG.

2.4.2.4 Adaption layer

6LoWPAN makes use of IPv6 address compression. The usage of IPv6 in transmission of packets over LoWPAN (IEEE standard 802.15.4) is not a natural fit. Hence an adaptation layer is proposed by IETF to make IPv6 and 802.15.4 compatible with each other. This layer is placed between network layer and data link layer in 6LoWPAN protocol stack.

There are three main functions of the adaptation layer:

1. Fragmentation and reassembly of packets
2. Header Compression and decompression:
3. Routing

RFC4944 Features	RFC6282 Features
- Basic LoWPAN header format - HC1 (IPv6 header) and HC2 (UDP header) compression formats - Fragmentation & reassembly - Mesh header feature (depreciation planned) - Multicast mapping to 16-bit address space	- New HC (IPv6 header) and NHC (Next-header) compression - Support for global address compression (with contexts) - Support for IPv6 extension header compression - Support for UDP - Support for compact multicast address compression

Table 5: Different LowPAN release

- **IPV6 address**

The IPv6 Addressing: 128-bit IPv6 address Interface ID (IID) = 64-bit prefix + 64-bit

The 64-bit prefix is hierarchical Identifies the network you are on and where it is globally
The 64-bit IID identifies the network interface Must be unique for that network
Typically is formed statelessly from the interface MAC address: Called Stateless Address Auto configuration (RFC4862)

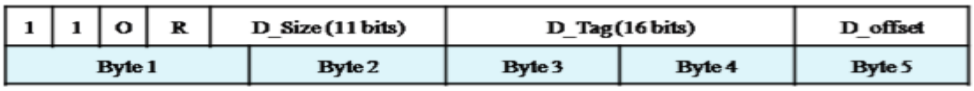
And there are different kinds of IPv6 addresses:

- Loopback (0::1) and Unspecified (0::0).
- Unicast with global (e.g. 2001::) or link-local (FE80::) scope.
- Multicast addresses (starts with FF::).
- Anycast addresses (special-purpose unicast address).

1. Fragmentation and reassembly of packets

Fragmentation is only required when the entire IPv6 packet cannot fit in a single IEEE 802.15.4 frame. The size of the fragments will be according to the maximum frame size at the data link layer. If datagram is small and a single frame is sufficient enough to carry the payload than there is no need to perform fragmentation. In that case no fragment header is attached with the packet.

Each fragment will then include a fragment header before it is transmitted. The fields of fragment header are Datagram Size, Offset and Tag.



<table>
<tr><td>1</td><td>1</td><td>O</td><td>R</td><td colspan="2">D_Size (11 bits)</td><td colspan="2">D_Tag (16 bits)</td><td>D_offset</td></tr>
<tr><td colspan="5">Byte 1</td><td>Byte 2</td><td>Byte 3</td><td>Byte 4</td><td>Byte 5</td></tr>
</table>

Figure 19: The fields of fragment header are Datagram Size, Offset and Tag.

The first two bit of byte 1 are ‘11’: that means this header is a fragment header.

- D_size: indicates the size of datagram before fragmentation. The size of un-fragmented datagram is send with every fragment so that sufficient buffer is allocated on receiver side as datagram may reach out of order.
- D_Tag: is a unique number which is attached with each fragment belonging to same datagram.
- D_offset: indicates the placing of the fragment in an un-fragmented datagram. It is useful for arranging the fragments in order during reassembly of datagram.
- Bit number 3 of byte 1: represents offset (O) whose value could be 0 or 1. The offset value of first fragment will be 0 and for rest of the fragments it will be 1. The fragment header of first fragment is of 4 bytes as there is no need to send the offset value in it. For rest of the fragments, the fragment header is of 5 bytes.
- Bit numbers 4 and 5: are reserved bits (R) for future use.

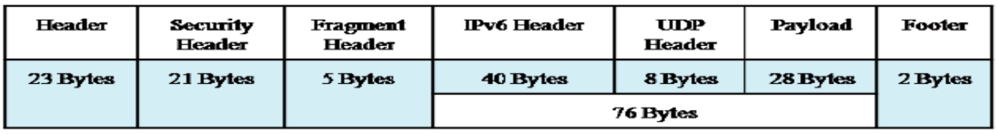
- **Reassembly**

- Reassembly process takes place at adaptation layer of receiver node.
- There is no memory crunch during reassembly of complete packet because each fragment carries information (D_Tag, D_offset) about how much buffer need to be reserved at receiver node.
- The Fragment reassembly time is usually 60 sec and if all fragments belonging to same packet does not arrive and misses then the reassembly buffer is drained. And in this case all the fragments need to be transmitted again by the sender.
- When fragments reach the adaptation layer of receiver then they are reassembled to form a complete packet and then passed to the upper layer.

2. Header Compression and decompression

Hence compression of IPv6 and UDP/TCP header is needed so as to increase the bytes available for payload. IPv6 packet size is of 1280 bytes whereas in 6LoWPAN the Maximum Transmission Unit (MTU) of a packet is 127 bytes. So the IPv6 addresses are compressed in 6LoWPAN.

The frame format of 802.15.4 is shown in this figure. This leaves only 76 bytes for upper layer headers and payload, as illustrated in Figure 20.



Header	Security Header	Fragment Header	IPv6 Header	UDP Header	Payload	Footer
23 Bytes	21 Bytes	5 Bytes	40 Bytes	8 Bytes	28 Bytes	2 Bytes
			76 Bytes			

Figure 20: Frame format of 802.15.4

3. Routing

Usually routing is considered as the main task of network layer but it can also be handled by adaptation layer. When routing decision takes place at adaptation layer it is called **mesh under routing** and when routing is done at network layer it is route over routing.

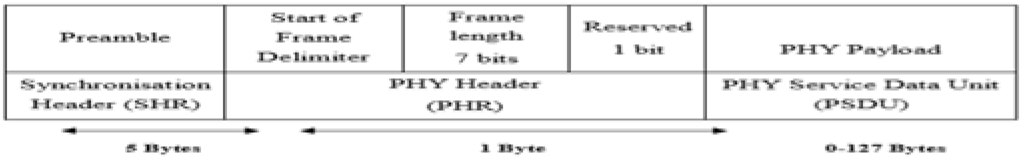
2.4.2.5 Mac layer

The task of Data link layer is to detect and correct the errors which may occur during transmission of data bits. Medium access layer or MAC layer is present in data link layer. This layer senses the medium for collision free transmission of frames using various protocols like CSMA/CD or CSMA/CA.

2.4.2.6 Physical layer

6LoWPAN PHY layer provides two services namely the PHY data service and PHY management service. The key function of PHY data services is to provide transmission and reception of data packets between MAC and PHY through the physical radio channel.

The PHY management service interface, offers access to every layer management function and maintains a database of information on related personal area networks. It is based on IEEE 802.15.4 standard which operates at the frequency of 2400 – 2483.5MHz offering a data rate of 250 kbps. The protocol data unit is IEEE 802.15.4 compliant with a maximum payload of 127 bytes.



Preamble	Start of Frame Delimiter	Frame length 7 bits	Reserved 1 bit	PHY Payload
Synchronisation Header (SHR)	PHY Header (PHR)			PHY Service Data Unit (PSDU)
5 Bytes	1 Byte			0-127 Bytes

Figure 21: IEEE 802.16.4 PHY Packet structure

2.4.3 6lowPAN security

As the number of devices connected to the IoT world is increasing day by day, security becomes one of the biggest issues in the IoT.

In 6LoWPAN networks, security can be handled at three layers: MAC, network and application (Hennebert and Dos Santos 2014).

IEEE 802.15.4 MAC layer implements security services to achieve data encryption and authentication.

IEEE 802.15.4 defines different security suites which can be mainly classified into:

- Null (no security), encryption only (AES-CTR)
- Authentication only (AES-CBC)
- Encryption and authentication (AES-CCM) (LAN/MAN-Standards-Committee et al. 2003).

At the network layer, IPsec protocol suite can be used to authenticate and encrypt each IP packet.

Also, at the application layer, datagram transport layer security (DTLS) can be used to secure the network traffic.

In SigFox networks, the radio protocol provides a mechanism for message authentication and integrity using a unique device ID and a message authentication code (Zuniga and Ponsard 2016).

Each message is signed using a key (device ID) which is flashed inside the device by a manufacturer. This ensures that the message has been generated and sent by the device with its ID in the message (VT-Networks 2016).

However, SigFox does not provide an encryption mechanism to encrypt data sent in the message payload where an attacker in the middle can read the messages.

2.4.4 Building a Nucleo Powered 6LowPAN Gateway

This tutorial will walk you through setting up a quick 6LoWPAN (pronounced "six-lo-pan") network to send data from a node to a gateway connected to a Wi-Fi network.

For this demo, we'll use a node, and a Wi-Fi Bridge that acts as a gateway to the Internet.

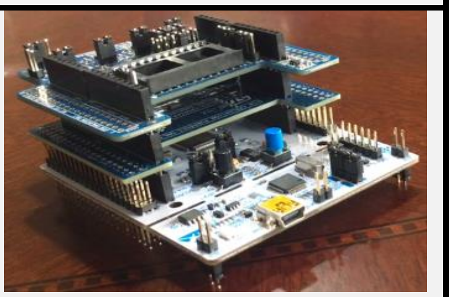
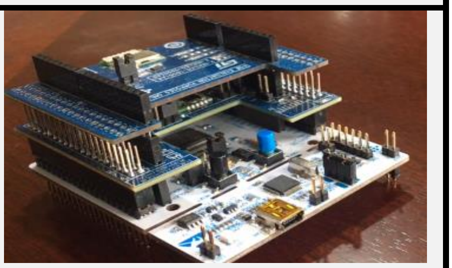
Node component	**Schematic**
Nucleo-F401RE or L152RE boards, which will serve as the base for the node.	
X-Nucleo IDS01A5 or IDS01A4, which are sub-gigahertz expansion boards.	
X-Nucleo IKS01A2, which contains multiple sensors.	
Gateway components	**Schematic**
Nucleo-F401RE or L152RE boards, which will serve as the base for the gateway.	
X-Nucleo IDW01M1 board to connect the gateway to a Wi-Fi access point.	
X-Nucleo IDS01A5 or IDS01A4, which are sub-gigahertz expansion boards.	

Table 6: open source Node and Gateway components for 6LowPAN

- **Downloading**

- The STSW-LINK009**,** which are USB drivers for the Nucleo-F401RE boards.
- Download and install the STSW-LINK007 firmware to update your Nucleo-F401RE.
- Download the source code and application examples for the node in FP-SNS-6LPNODE1, and those for the GTW in FP-NET-6LPWIFI1.
- Download a terminal emulator, like Tera Term Pro on Windows, or CoolTerm on macOS.

- **Checking and updating the Wi-Fi Module's firmware for the X-Nucleo IDW01M1**

- Place the X-Nucleo IDW01M1 onto the Nucleo F401RE, and plug the latter to your PC, then locate the storage volume NODE_F401RE.
- At this stage, we'll use an application that will help us determine the firmware version of the X-Nucleo IDW01M1. Just download the X-CUBE-WIFI1 software stack, and then open the following folders:
  STM32CubeExpansionWIFI1_V2.1.1 -> Multi -> Applications -> WiFi_VCOM -> Binary -> STM32F401RE-Nucleo.
- The only step left is to drag the file Project.bin onto the NODE_F401RE volume, and wait for the lights to blink.
- We must now open the terminal emulator ad establish a connection with the Nucleo Board (usually COM3 or COM4 on Windows, if unsure, check Device Manager) using the following settings :
  Baud: 115200, Data: 8 bit, Parity: None; Stop Bit: 1 bit, Flow Ctrl: None and Localecho: on.
- Once the connection is open, press the reset button on the F401RE.
- When the program is available, and it appears on your terminal window, type AT+S.STS, and press Enter.
- A list of parameters will be printed. Scroll at the top to find "version". If the value starts with 160129, you have the latest firmware, and you can proceed to the next step (Moving a 0-ohm resistance on the Wi-Fi Module). If it's not, you absolutely need to update before proceeding.

- **Moving the 0-ohm resistance to R34**

  At the back of the X-Nucleo IDW01M1, you will notice a 0-ohm resistance on path R4. And after you checked your firmware version, and upgraded if necessary, move that resistance to path R34. Once this is done, you're ready to build your first 6LoWPAN network.

- **Creating and setting up the Gateway**

  Assemble your module by putting the X-Nucleo IDW01M1 on top of the Nucleo-F401RE, then the X-Nucleo IDS01A5 on top of the X-Nucleo IDW01M1.

- Connect the module to your PC, and locate the demo program from FP-NET-6LPWIFI1 in: STM32CubeFunctionPack_6LPWIFI1_V1.1.0 -> Projects -> Multi -> Applications -> WiFi-Bridge -> Binary -> STM32F401RE-Nucleo -> STM32F4xx-Nucleo-WiFi-Bridge-IDS01A5.bin. Then Move it to your Nucleo F401RE volume, and wait for the lights to blink.
- Open a connection to the serial line with your Terminal Emulator (Baud Rate = 115200, Parity = None, Bits = 8, Stopbits = 1), then press the reset button on your Nucleo board.
- The program will ask if you want to enter your Wi-Fi network's settings. If you do nothing – or press "n" -, it will default to a Wi-Fi network with SSID "STM", and a WPA2 password "STMdemoPWD".
- To enter your settings, press "y" in your Terminal Emulator. At this point, the module will scan the surrounding Wi-Fi networks and print a list of all the SSIDs it was able to scan. If you don't see yours, although you're supposed to, something may be wrong with your Wi-Fi Access Point. Make sure it is a 2.4 GHz network, and that the SSID is "broadcasted", by checking your router's setup. If you are sure that your Wi-Fi network is OK, or if you see your Wi-Fi network in the printed list, enter the name of your SSID**.**
- Sometimes, entering the SSID, even if it's not found on the list, will also work. The next step is to key-in your password. If your network doesn't use a password just enter a letter anyway. No matter what, the system will not allow you to leave that field blank. Finally, choose the correct encryption standard. If your network doesn't require a password, type "0" and the system will ignore the password. Otherwise, type "1" for WEP, or "2" for WPA/WPA2.
- If everything went well, you will see the message "Successfully configured WiFi module and connected to the selected AP" and a list of values.

- **Tips and tricks for an Autonomous Gateway**

However, if you wish to use the demo program and be able to unplug your module from the power source, the software was designed to automatically connect to a default Wi-Fi network (SSID: “STM”, WPA2 Password: “STMdemoPWD”) after a period of inactivity**.** Hence, there are two ways one could easily create a gateway that can function without necessarily being setup beforehand:

You could import the demo’s source files to your toolchain, update the default Wi-Fi network values to match your access point, and load the application onto your board, but this is outside of scope of this demo.

You could also create a network that matches the default values used by the program, which will then connect to it automatically.

- **Creating and setting up the Node**

Assemble your module by putting the X-Nucleo IDS01A5 on top of the Nucleo-F401RE, then place the X-Nucleo IKS01A2 on top of the IDS01A5**.**

- Connect the module to your PC, and locate the program (from FP-SNS-6LPNODE1) in STM32CubeFunctionPack_6LPNODE1_V1.1.0 -> Projects -> Multi -> Applications -> ipso-mems -> Binary -> STM32F401RE-Nucleo -> ipso-mems-IKS01A2-IDS01A5_f401.bin, then move it to the Nucleo F401RE volume, and wait for the LEDs to blink.
- Open a connection to the serial line with your Terminal Emulator (Baud Rate = 115200, Parity = None, Bits = 8, Stopbits = 1), then press the reset button on your Nucleo board.
- If everything went well, you should see a line that says, “RD Client process started”.
- After a few seconds, you’ll see a message such as “RD Client started with endpoint ‘?ep=STF4-mems-[string_of_numbers]’”. In our case, it was “STF4-mems-34328034A034”. It is important to note that identifier as we’ll need it later**.**
- The string “STF4-mems-[string_of_numbers]” is unique to all devices and set in the firmware. You can therefore reinstall the demo program, or lose power, and that identifier will remain the same.

- **Working with Your Node**

- The demo program automatically sends the sensors’ data to a cloud server. To access it simply go to: http://leshan.eclipse.org/#/clients and locate your module’s ID under “Client Endpoint”**.**
- Clicking on your identifier will open a window with a list of values. To monitor the movements of your X-Nucleo IKS01A2 sensor board, go to the section called “Accelerometer”, and click “Observe” next to “X Value”, “Y Value”, and Z” Value”**.** You will see numbers appearing. Moving your node will automatically update them, showing that the system can now track of your movements using your sub-gigahertz network.

- **Tips and tricks for an Autonomous Gateway**

Leshan is not designed for production code and mass market applications. However, it is possible to use their API to automatically get the values from the node to your application, so you can start developing a very simple program, thanks to a very well thought-out URL scheme, designed by the OMA LightweightM2M (LwM2M) standard, which takes the form of “/Object ID/Instance Number/Resource ID”.

In Leshan, the page used to monitor the values from the node’s gyroscope is found at http://leshan.eclipse.org/#/clients/STF4-mems-[string_of_numbers], or in our case http://leshan.eclipse.org/#/clients/STF4-mems-34328034A034. Next to the X Value in the Accelerometer section, is a path called “/3313/0/5702”, which will remain the same, no matter your ID.

Hence, if you replace the “#” with “api” in the url, and add the path for the X Value, you can get the string containing the measurement.

In our example, typing:

“http://leshan.eclipse.org/api/clients/STF4-mems-34328034A034/3313/0/5702” in a browser will return the string: “{“status”:”CONTENT”,”content” :{“id”:5702,”value”:-0.1025390625}}”, the number after “value” being the sensor’s data.

As are result, one could create a program with a simple loop to get the X (/3313/0/5702), Y (/3313/0/5703), and Z (/3313/0/5704) values from the Leshan servers. by using their API and URL scheme, which means, you have not only setup your sub-gigahertz network, but have created the foundation for your first project.

Chapter 3. LPWAN technology

A **low-power wide-area network** (**LPWAN**) or **low-power wide-area** (**LPWA**) network or **low-power network** (**LPN**) is a type of wireless telecommunication wide area network designed to **allow long range communications at a low bit rate among things** (connected objects), such as sensors operated on a battery.

LPWAN offers multi-year battery lifetime and is designed for sensors and applications that need to send small amounts of data over long distances a few times per hour from varying environments.

3.1 Cellular LPWAN

For LTE-based IoT networks to succeed, they need to have the following characteristics: Long battery life**, low** cost, support for a high volume of devices**, Enhanced** coverage (better signal penetration through walls) **and** Long range/wide spectrums.

3.1.1 Cat_1

It is the **only fully-available** cellular IoT option at the moment, while **the performance is inferior to 3G networks,** it's an excellent option for IoT applications that require a browser interface or voice.

The major attraction is that it's **already standardized**, Experts predict that**, Cat-1 (and Cat-M1) networks will take place of 3G and eventually 4G technologies.**

3.1.2 Cat_0

- Cat-0 **optimizes for cost**.
- It eliminated features that supported high data rate.
- Requirements for Cat-1 (dual receiver chain, duplex filter).
- While Cat-1 is replacing 3G, Cat-0 is the protocol that set the groundwork for Cat-M replacing 2G as the cheaper option.

3.1.3 Cat-M1/Cat-M/LTE-M

- Cat-M viewed as the second generation of LTE chips built for IoT applications.
- It completes the cost and power consumption reduction for which Cat-0 originally set the stage.
- By capping the maximum system bandwidth at 1.4 MHz (as opposed to Cat-0's 20 MHz),
- Cat-M has specific use cases for LPWAN applications like smart metering, in which only small amount of data transfer is required.
- But the real advantage of Cat-M over other options lies in this: **Cat-M is compatible with the existing LTE network**.
- For carriers such as Verizon and AT&T, this is great news as they don't have to spend money to build new antennas, although meshing Cat-M into LTE networks requires a software patch. T
- Lastly, it's almost certain that 5G and LTE technologies will coexist well into the 2020s, so the backward-compatibility of Cat-M is a bonus.

3.1.4 NB-IoT/Cat-M2

- NB-IoT (also called Cat-M2) has a goal similar to that of Cat-M (low cost, long battery life, and high connection density).
- NB-IoT focuses specifically on indoor coverage
- Limits the bandwidth to a single narrow-band of 200 kHz.

- It uses **DSSS modulation** instead of LTE radios.
- Therefore, **NB-IoT doesn't operate in the LTE band**, which means that providers **have a higher upfront cost** to deploy NB-IoT.
- NB-IoT is being touted as the potentially less expensive option**,** because it **eliminates the need for a gateway.**
- **With NB-IoT, however, sensor data is sent directly to the primary server.** For that reason**, Huawei, Ericsson, Qualcomm, and Vodafone** are actively investing in commercial applications of NB-IoT.

- NB-IoT works best in sophisticated urban (work well indoors) locations but its performance is not up to the mark in suburban or rural areas (practically any place that does not have strong, glitch-free 4G coverage). Where the LoRaWAN, its coverage remains relatively steady across all types of locations.
- Because NB-IoT chips are more complex: That means users get the high performance level associated with cellular connections. But at the cost of more complexity and greater power consumption then LoRaWAN.
- It has faster response times than LoRa and can guarantee a better quality of service.
- NB-IoT, the devices have to be synced with the network at regular (and relatively frequent) intervals, where no such network synchronization is required in the ALOHA-based LoRa architecture.

3.1.5 EC-GSM (formerly EC-EGPRS)

EC stands for Extended Coverage. **EC-GSM is the IoT-optimized GSM network**, the wireless protocol 80 percent of the world's smartphones use.

As the name suggests, **EC-GSM can be deployed in existing GSM networks**—a **huge advantage in terms of practicality and modularity**, since a simple piece of software enables EC-GSM connectivity within 2G, 3G, and 4G networks.

EC-GSM also has specific use cases in non-Western regions **such as Malaysia, and African** and Middle-Eastern countries, where 2G remains a popular standard. **Ericsson, Intel, and Orange** are said to have completed live trials of EC-GSM earlier this year. EC-GSM, however, isn't generating as much buzz as Cat-M or NB-IoT.

3.1.6 5G Cellular IoT

Unlike the cellular IoT options above, 5G has yet to be officially defined. Next Generation Mobile Networks Alliance (NGMN) is pushing for specs for it to be 40 times faster than 4G while supporting up to 1 million connections per square kilometer.

5G is already enabling high-bandwidth, high-speed applications for Ultra-HD (4k) streaming, self-driving car connectivity, or VR/AR applications, such as Verizon and Samsung showcased at the Superbowl and Olympics respectively. What will the future hold?

	Max.system bandwidth	**Downlink peak rate**	**Uplink peak rate**	**Duplex**	**Number of antennas**	**Transmit power (UE)**	**Estimated modem complexity**
Release 8: Cat-4	20Mhz	150Mbit/s	50Mbit/s	Full duplex	2	23dBm	100%
Release 12: Cat-0	20Mhz	1Mbit/s	1Mbit/s	Half duplex	1	23dBm	40%
Release 13: Cat-M	1.4Mhz	1Mbit/s	1Mbit/s	Half duplex	1	20dBm	20%
Release 13: NB-IoT	200Khz	~200Kbit/s	~200Kbit/s	Half duplex	1	23dBm	<15%

Table 7: Features of different 5G release

3.2 Non Cellular LPWAN

There is a lot of Non cellular LPWAN technology but the most three important technology is LoRaWAN, SigFox.

3.2.1 LoRaWAN

LoRaWAN (Long Range Wide Area Network), built upon the underlying LoRa physical layer (defined by the LoRa Alliance consortium), an open source communication protocol.

LoRaWAN define the communication protocol and system architecture for the network.

LoRaWAN communication protocol ensures: Data rates are defined that range from 300bps to 5.5kbps, with two high-speed channels at 11kbps and 50kbps (FSK modulation)!

Supports: secure bi-directional communication, mobility and localization and reliable communication, and adds additional headers to the data packets.

Frequency band	902-928 Mhz (North America)	863-870 Mhz and 434 Mhz (Europe)	779-787Mhz (China)
Number of channels	80 channels for 915 Mhz band	10 channels for 868 Mhz band and 780 Mhz band	
Channel bandwidth	125 KHz and 500 KHz for 915 Mhz band	125 KHz and 250 KHz for 868 Mhz band and 780 Mhz band	
Maximum data rate	980 bps -21.9 Kbps	250bps-50Kbps	980 bps -21.9 Kbps
Protocol data unit	Variable number of bytes header + (19 to 250) bytes		
Channel coding	Chirp spread spectrum (CSS)		
Channel modulation	LoRa for 915Mhz band		
	LoRa and GFSK for 868 Mhz band and 780 Mhz band		
Receiver sensitivity	-137dBm		
Transmission range	5-15 Km		
Battery lifetime	<10 years		

Table 8: LoRaWAN features

3.2.1.1 LoRaWAN architecture

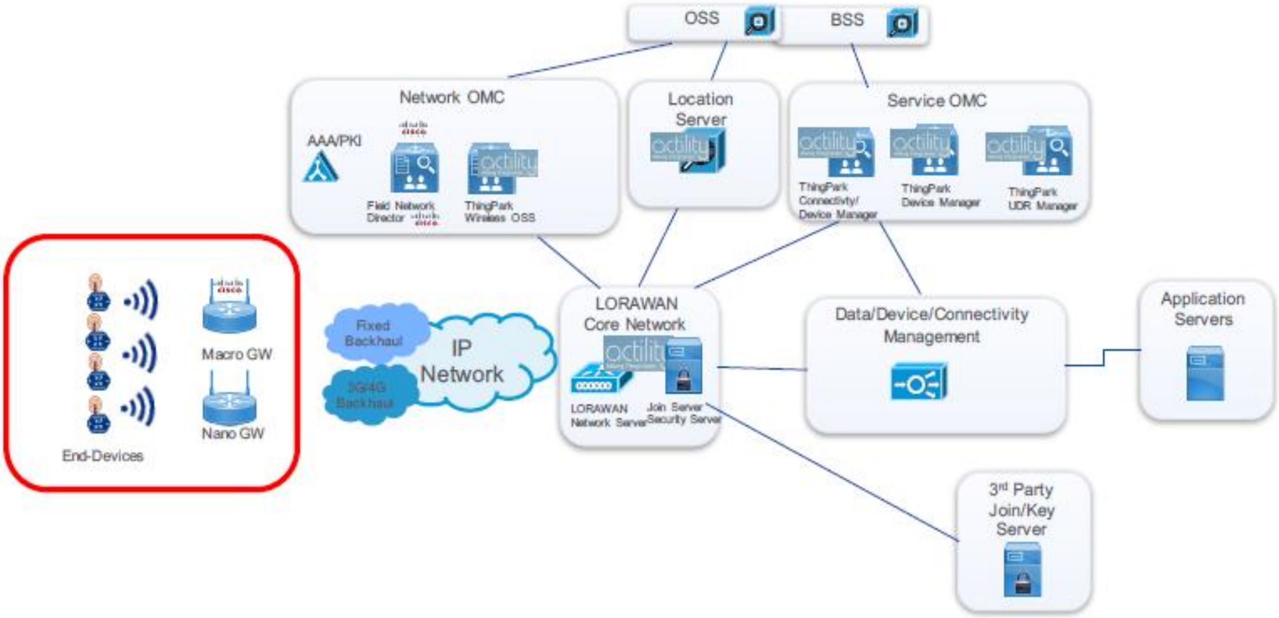


Figure 22: LoRaWAN system communication architecture

LoRaWAN network architecture is deployed in a **star-of-stars topology** (vs. mesh topology e.g. ZigBee).

The LoRaWAN networks laid out in a star-of-stars topology have base stations relaying the data between the sensor nodes and the network server.

The Communication between the sensor nodes and the base stations goes over the wireless channel utilizing the LoRa physical layer, whilst the connection between the gateways and the central server are handled over a backbone IP-based network.

- **End Nodes** transmit directly to all gateways within range, using LoRa.
- **Gateways** relay messages between end-devices and a central network server using IP.

- **End Nodes**

The End Nodes are LoRa embedded sensors. The nodes typically have:

- Sensors (used to detect the changing parameter eg. temperature, humidity, accelerometer, and gps).
- LoRa transponder to transmit signals over LoRa patented radio transmission method, and optionally a micro-controller (with on board Memory).

  The sensors may connect to the LoRa transponder chip, or the sensor may be an integrated unit with the LoRa transponder chip embedded.

  The LoRaWAN end nodes (sensors) typically use Low Power and are battery powered (Class A and Class B).

  LoRa embedded sensors that run on batteries that can typically last from 2–5 years.

  The LoRa sensors can transmit signals over distances from 1km — 10km.

- **Gateway**

The LoRa sensors transmit data to the LoRa gateways. The LoRa gateways connect to the internet via the standard IP protocol and transmit the data received from the LoRa embedded sensors to the Internet i.e. a network, server or cloud simply converting RF packets to IP packets and vice versa.

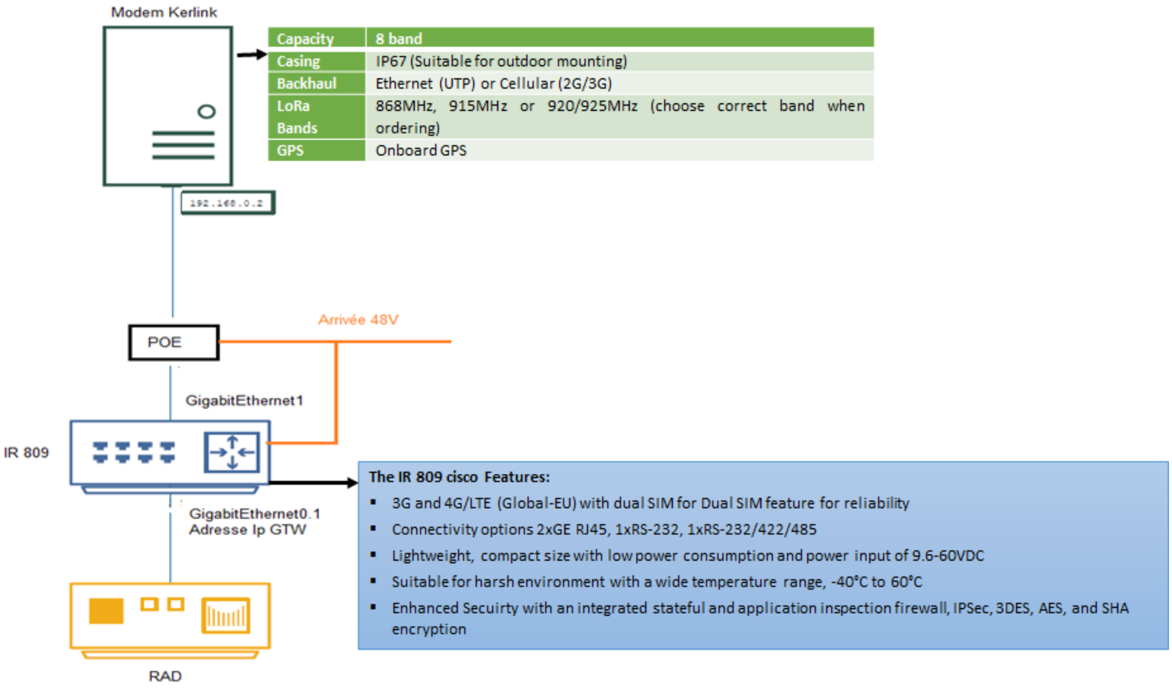
Gateways do require accurate time synchronization, this is currently achieved with GPS at the gateways (or any means available to synchronize gateway clocks to within a few tens of nanoseconds).

There is two type of gateways:

- **The Macro-GW (Cisco or Kerlink):** deployed outdoor (i.e. cell tower, rooftops) to cover large rural or urban areas) is composed of 2 main components:
  - A cell-site Gateway: providing connectivity, routing, and security functions
  - LoRa Modem: providing RF functions and hosting the LRR Software (implementing the LORAWAN PHY/MAC functions).
- **The Nano-GW (Multitech):** Indoor radio equipment transferring frames between LoRa radio network and a central LoRa Network Server using Ethernet technology for backhauling. Used to improve indoor coverage and for small indoor private networks.

  The Multitech Gateway is very easy to use, which we can connect directly to any internet backhaul like Home modem etc.…

There is 4 scenario of Macro-GW which you can found in page (), and here two of them

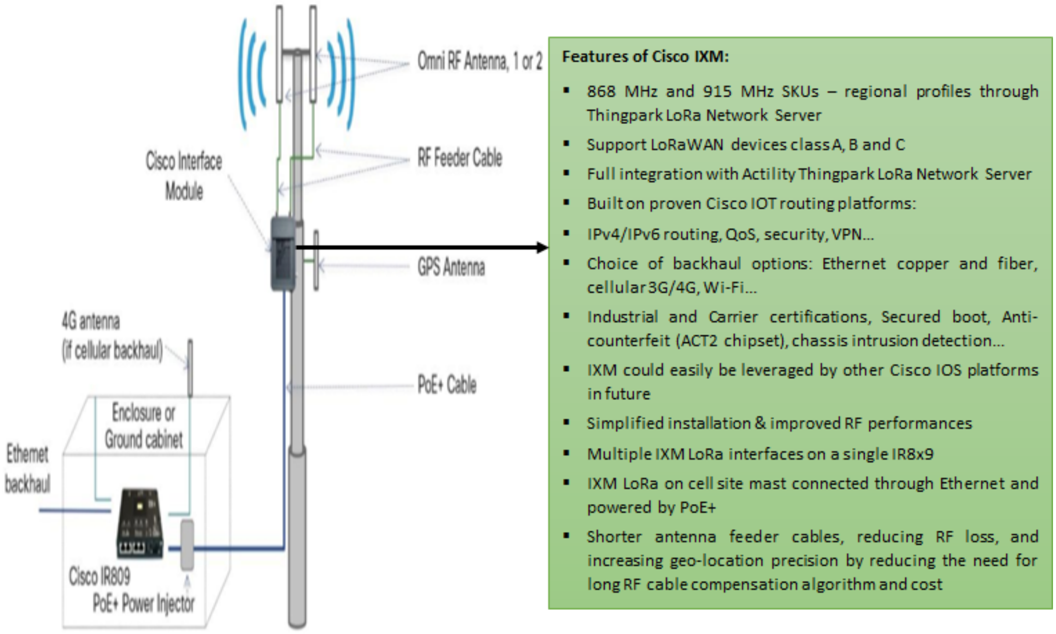


Capacity	8 band
Casing	IP67 (Suitable for outdoor mounting)
Backhaul	Ethernet (UTP) or Cellular (2G/3G)
LoRa Bands	868MHz, 915MHz or 920/925MHz (choose correct band when ordering)
GPS	Onboard GPS

The IR 809 cisco Features:

- 3G and 4G/LTE (Global-EU) with dual SIM for Dual SIM feature for reliability
- Connectivity options 2xGE RJ45, 1xRS-232, 1xRS-232/422/485
- Lightweight, compact size with low power consumption and power input of 9.6-60VDC
- Suitable for harsh environment with a wide temperature range, -40°C to 60°C
- Enhanced Secuirty with an integrated stateful and application inspection firewall, IPSec, 3DES, AES, and SHA encryption

Figure 23: Modem Kerlink_ Gateway Cisco (IR809) and Ethernet backhaul connections



Features of Cisco IXM:

- 868 MHz and 915 MHz SKUs – regional profiles through Thingpark LoRa Network Server
- Support LoRaWAN devices class A, B and C
- Full integration with Actility Thingpark LoRa Network Server
- Built on proven Cisco IOT routing platforms:
- IPv4/IPv6 routing, QoS, security, VPN…
- Choice of backhaul options: Ethernet copper and fiber, cellular 3G/4G, Wi-Fi…
- Industrial and Carrier certifications, Secured boot, Anti-counterfeit (ACT2 chipset), chassis intrusion detection…
- IXM could easily be leveraged by other Cisco IOS platforms in future
- Simplified installation & improved RF performances
- Multiple IXM LoRa interfaces on a single IR8x9
- IXM LoRa on cell site mast connected through Ethernet and powered by PoE+
- Shorter antenna feeder cables, reducing RF loss, and increasing geo-location precision by reducing the need for long RF cable compensation algorithm and cost

Figure 24: Modem Cisco_ Gateway Cisco (IR809) and 4G backhaul connections

- **Network server**

- The Network servers can be cloud based platform solutions like The Things Network (TTN) or LoRIOT.
- The network servers connect to the gateways and de-dup data packets, and then routes it to the relevant application.
- The network servers can be used for both uplink (i.e. sensor to application) and downlink (i.e. application to sensor) communication.
- The Things Network server has a Router, Broker and Handler, which processes the data packets from the LoRaWAN gateway. It also has an AWS Bridge that connects TTN to the AWS IOT platform.

- **Application server**

The Application can typically be built over IoT platforms like AWS IoT using Lambda, DynamoDb or S3 services.

3.2.1.2 LoRaWAN Geolocation

- **End device level**

- A LoRaWAN end-device can be located if uplink transmissions from the device are received by three or more gateways. *Note, these uplink transmissions need not be specific transmissions for geolocation, they may be typical LoRaWAN application data frames.*

- **Gateway level**

- Several gateways simultaneously receive the same uplink message, and the end-device location is determined using multilateration techniques. The multilateration process is shown in the figure below:

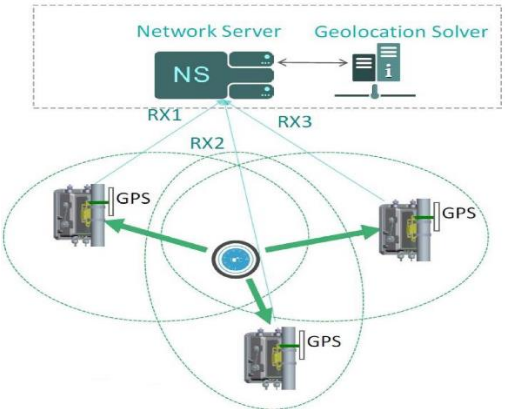


Figure 25: LoRaWAN multilateration process

- Each received uplink frame is accurately time-stamped by the gateway. This time stamp is forwarded to the network server as part of a frame’s metadata, which also includes signal level, signal-to-noise ratio and frequency error.

- **Network server level**

- The network server sorts multiple receptions of the same frame, groups all the metadata including the timestamps for this frame, and requests a geolocation computation from the geolocation solver.

- **Geolocation solver level**

- The elementary geolocation solver function is to compute, for a given frame, the difference in time of reception seen by pairs of gateways. This time difference measures proximity of the end-device to one gateway of the pair compared with the other.
- When the TDOA (Time Difference Of Arrival) is known for a pair of gateways, the end-device can be placed on a hyperbola. With several such time differences, the end-device can be placed on several hyperbolae.
- The end-device is positioned at the intersection of these hyperbolae.

3.2.1.3 LoRaWAN protocols stack

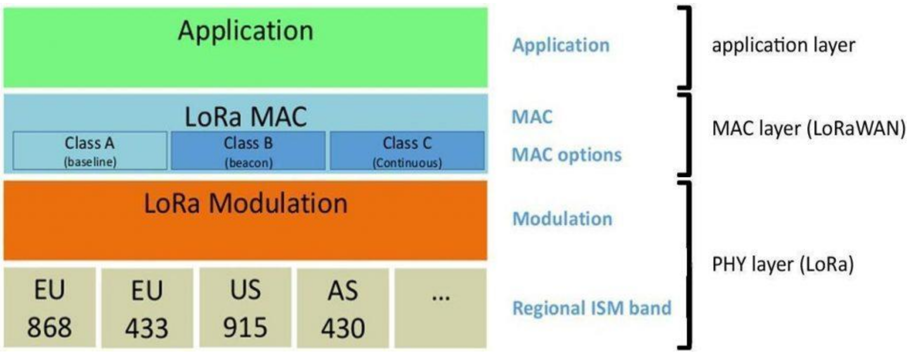


Figure 26: LoRaWAN protocols layers

3.2.1.3.1 Application layer

- High level application layer that handles business logic.
- Application layer implements open source LoRaMAC library and communication layer with LoRa transceiver.
- Encryption is carried out at the network and application layers.
- Data from the application layer is mapped into the MAC Payload

3.2.1.3.2 LoRa MAC layer

Note some people describe the functionality of the MAC layer as a general functionality of the LoRaWAN protocol.

- The MAC layer constructs the MAC frame using the MAC payload. The MAC payload contains:
  - Frame header (holding both the source and destination addresses plus a frame counter),
  - Frame port: used to determine if the frame contains MAC commands alone (when it is set to 0) or application specific data.
  - Frame payload (holding application data).
- Following MAC messages are used in LoRa for establishing communication between end device and server.
  - Join request (From End device to Server)
  - Join accept (from network server to End device)
  - Beacon frame (from gateway to End device) for scheduling slot for reception by End devices.
  - Confirmed Data Up/Down (This messages are to be acknowledged by LoRa receiver)
  - Unconfirmed Data Up/Down (This messages do not require any ack).

- **LoRa class**

Class A	Class B	Class C
Battery powered (Sensors sleep most of the time and life devices is typically 2–5 years)	Low latency (Battery powered sensors, the sleep is controlled by the network, and network can instruct the device when to sleep when to wake up)	No latency (Always connected to the power source, have a continuous open receive widow, except when transmitting)
Bidirectional communications	Bidirectional with scheduled receive slots	Bidirectional communications
Unicast messages	Unicast and Multicast messages	Unicast and Multicast messages
Small payloads, long intervals	Small payloads, long intervals, Periodic beacon from gateway	Small payloads
End-device initiates communication (Uplink)	Extra receive window (ping slot)	Server can initiate transmission at any time
Server communicates with end-device (downlink) during predetermined response windows	Server can initiate transmission at fixed intervals	End-device is constantly receiving

Table 9: Difference between LoRa class A, B and C

Server communicates with end-device (downlink) during predetermined response windows: Devices transmit data and then wake up after a receive delay period to receive data (Each device's uplink transmission is followed by two short downlink receive windows). This means that the acknowledgement for the data transmitted by Class A devices must come immediately).

3.2.1.3.3 Physical layer

- **LoRa** is the physical layer (enables the long-range communication link) is a patented digital wireless data communication IoT technology developed by Cycleo of Grenoble, France. It was acquired by Semtech in 2012, which holds the IP for LoRa transmission methodology.
- **LoRa** enables very-long-range transmissions (more than 10 km in rural areas) with low power consumption.
- Wireless modulation technology (uses Spreading Factors to set the modulation rate: SF7 to SF12).
- Low bandwidth! Low battery usage! Sensitivity: -142 dBm! Link budget (EU): 156 dB and Data Rates: 0.3-50 kbps
- Operates in the license-free ISM bands all around the world! 433, 915 MHz and 868 Mhz in EU band is mostly limited to 14 dBm (25 mW)
- Regulated (power, duty-cycle, and bandwidth). In EU: 0.1% or 1% per sub-band duty-cycle limitation (per hour).
- Robust to interference, multipath and fading
- IP requires less than 50k gates
- The physical layer also contains gray indexing, data whitening, interleaving, and forward error correction to reduce effects of interference and poor radio conditions. An exact implementation is proprietary.

- **LoRa modulation**

The LoRa modulation is a derivative of Chirp Spread Spectrum modulation (CSS). The modulation has constant amplitude and it sweeps across the entire bandwidth.

The CSS modulation has few selectable parameters to tune its performance: modulation bandwidth, code rate, and spreading factor.

CSS also has 11 relatively low transmission power and it is very resistant against jamming, multi-path propagation, and unwanted Doppler effects.

- **CSS modulation**

LoRa modulates information into chirps (Data is modulated into chirps by instantaneous frequency changes). Chirps are constantly varying frequency signals. Rising frequency chirps are upchirps and decreasing frequency chirps are downchirps. The frequency bandwidth of a chirp is equal to the used channel bandwidth, meaning that a single chirp uses the entire bandwidth.

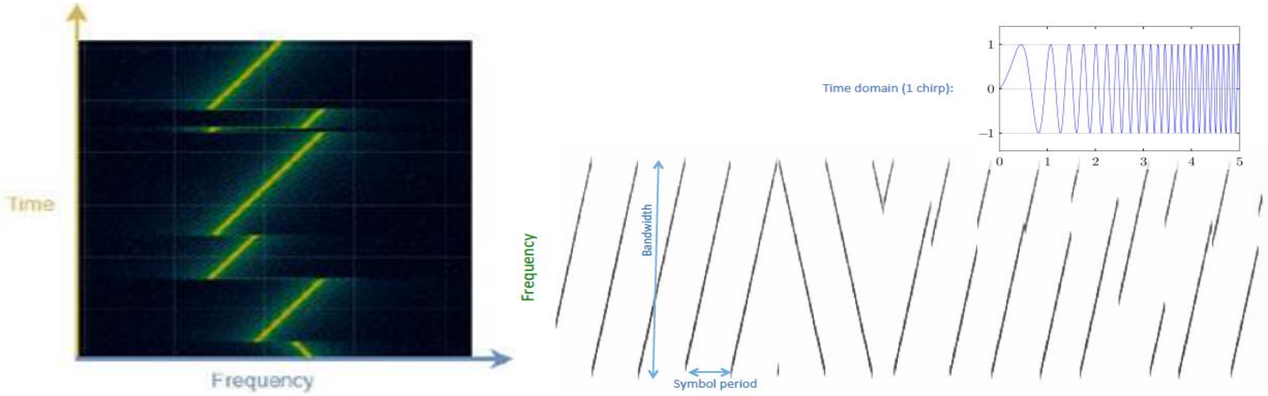


Figure 27: LoRa modulated signal received with a software defined radio RTL SDR

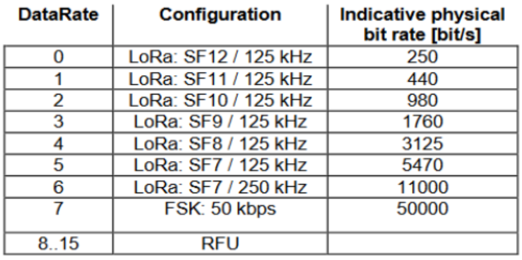
Figure on the left corner shows the actual LoRa modulated signal received with a software defined radio. Modulation has also a preamble which consists of repeated upchirps in the be-ginning of a frame. The receiver will use this known sequence to adjust its receiver before demodulation.

- **Spreading Factor**

Another feature very specific to this network is the Spreading Factor. The spreading in spread spectrum systems is generally achieved by multiplying the original data signal with a spreading code or chip sequence that is at a much faster rate than the data signal and therefore spreads the resulting signal bandwidth beyond that of the original.

The principle is simple, when a device is close by a Gateway, the risk of radio signal perturbation is low, allowing to have less redundancy and gain speed. Inversely, when a device is far from the Gateway, radio signal perturbations may occur. So, we can transmit further but the speed is slower.

There are 7 different spreading factors call **FSK** (Frequency-shift keying) are used to adjust the radio signal speed depending on the distance.



DataRate	Configuration	Indicative physical bit rate [bit/s]
0	LoRa: SF12 / 125 kHz	250
1	LoRa: SF11 / 125 kHz	440
2	LoRa: SF10 / 125 kHz	980
3	LoRa: SF9 / 125 kHz	1760
4	LoRa: SF8 / 125 kHz	3125
5	LoRa: SF7 / 125 kHz	5470
6	LoRa: SF7 / 250 kHz	11000
7	FSK: 50 kbps	50000
8..15	RFU	

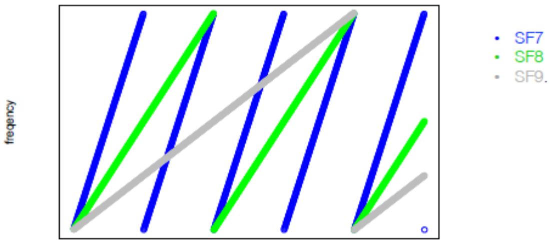
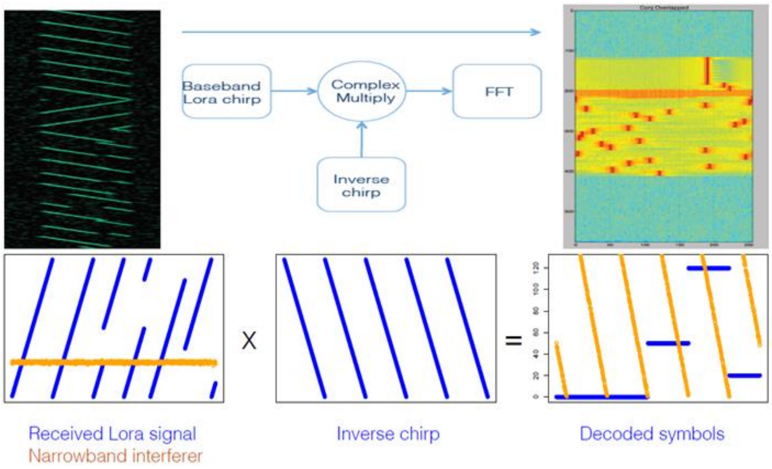


Figure 28: different spreading factors

- **CSS demodulation**

Demodulating a spread spectrum signal requires the receiver to know how the expected signal has been spread across the spectrum. The receiver can then generate upchirp and downchirp signal patterns and then multiply the received signal with generated patterns to extract symbols.



SpreadingFactor (RegModulationCfg)	Spreading Factor (Chips / symbol)	LoRa Demodulator SNR
6	64	-5 dB
7	128	-7.5 dB
8	256	-10 dB
9	512	-12.5 dB
10	1024	-15 dB
11	2048	-17.5 dB
12	4096	-20 dB

Figure 29: Demodulating a spread spectrum signal

- **LoRa message format**

LoRa frame consists of uplink messages and downlink messages. There are three type of classes supported in LoRa system. Based on these classes, LoRa frame structure varies. Uplink messages are transmitted from end devices to the server using one or more gateways.

Downlink message is transmitted from server to only one LoRa end device. This is done using single gateway connected with network server.

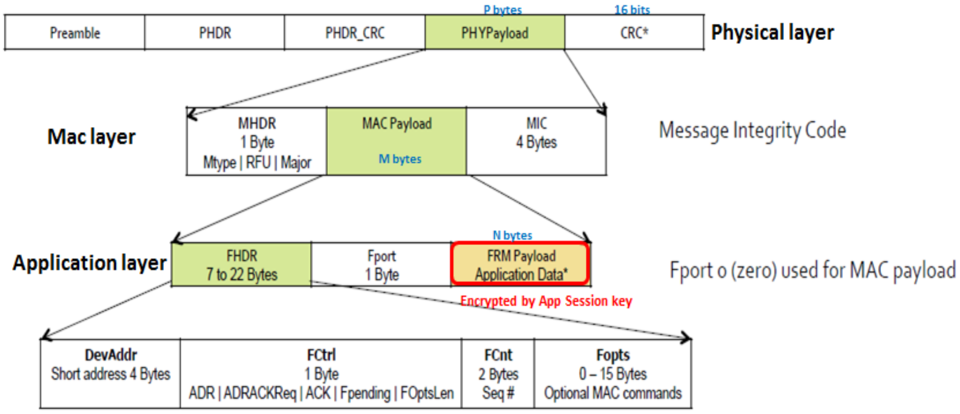


Figure 30: LoRa frame format

- Preamble: defines the packet modulation scheme, being modulated with the same spreading factor as the rest of the packet. Typically, the preamble duration is 12.25 Ts.
- MAC header: defines protocol version and message type, i.e., whether it is a data or a management frame, whether it is transmitted in uplink or downlink, whether it shall be acknowledged. MAC Header can also notify that this is a vendor specific message.
- MAC Payload: can be replaced by join request or join accept messages, in a join procedure for end node activation.
- MIC: The entire MAC Header and MAC Payload portion is used to compute the MIC value with a network session key (NwkSkey). The MIC value is used to prevent the forgery of messages and authenticate the end node.
- Frame Port: value is determined depending on the application type.
- Frame Payload: value is encrypted with an application session key (AppSKey). This encryption is based on the AES 128 algorithm.
- Device address: The first 8 bits identify the network, other bits are assigned dynamically during joining the network and identify the device in a network.

- Frame Control: used for network control information, such as whether to use the data rate specified by the gateway for uplink transmission, whether this message acknowledges the reception of the previous message, whether the gateway has more data for the mote.
- Frame counter: used for sequence numbering
- Frame options: for commands used to change data rate, transmission power and connection validation etc.

3.2.1.4 LoRaWAN Security

LoRaWAN utilizes two layers of security:

- **One for the network**, to ensure authenticity of the node in the network
- **Second for the application layer** to ensure the network operator does not have access to the end user's application data

An End Device (Node) must be activated before it can communicate on the LoRaWAN network. In LoRaWAN networks, two activation methods are available: OTAA und ABP.

3.2.1.4.1 OTAA, Over-the-Air-Activation

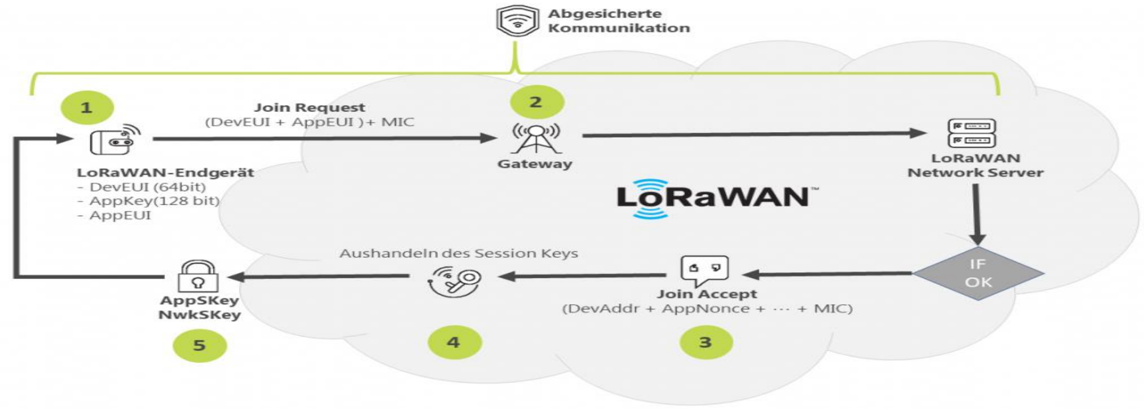


Figure 31: Different phases of the OTAA system security

This method is based on over-the-air Join Requests and Join Accept messages working hand in hand.

- **Join Requests scenario**

Each End Device (Node) is deployed with a **64-bit DevEUI**, a **64-bit AppEUI**, and a **128-bit AppKey**.

- **DevEUI:** is a globally unique identifier for the device which has a 64-bit address comparable with the MAC address for a TCP/IP device.
- **AppEUI**:
- **AppKey**: is used to cryptographically sign the Join Request.
- Additionally sends a **DevNonce**: which is a unique, randomly generated, two-byte value used for preventing replay attacks.
- These three values (DevEUI, AppEUI, AppKey) are signed with a **4-byte MIC** (Message Integrity Code) using the device's AppKey.

- **IF OK scenario**

The server accepts Join Requests only from devices with known DevEUI and AppEUI values while validating the MIC using the AppKey.

- **Join Accept scenario**

If the server accepts the Join Request, it responds to the device with a Join Accept message.

The application and network servers calculate the node's two 128-bit keys: the **Application Session Key (AppSKey)** and the **Network Session Key (NwkSKey)**, respectively. These are calculated based on the values sent in the Join Request message from the node.

- **What is Network Session Key (NwkSKey)?** This is a network layer security mechanism. This key is unique to each end device and shared between the end device and the network server. The network session key provides message integrity during communication and security for end device to network server communication.
- **What is Application Session Key (AppSKey)?** This key is responsible for end-to-end (application to application) ciphering of the payload. This is also an AES 128-bit key, unique to each end device. It is shared between the end device and application server. The application session key encrypts and decrypts application data messages and provides security for application payloads.

- The Join Accept reply includes the **AppNonce**, a **NetID**, and an end **device address** (DevAddr) along with configuration data for **RF delays (RxDelay)** and **channels to use (CFList)**.
  - **What is AppNonce?** The application server generates its own nonce value, this is another unique, randomly generated value.
  - **What is NetID?**
  - **What is DevAddr?** 32-bit identifier which is unique within the network, used to differentiate between end devices which have already joined the network. This allows the network and application servers to use correct encryption keys and to properly interpret the data.
  - When receiving data back, the data is encrypted with the AppKey.
- **Join accept received scenario**

  The node then uses the AppKey to decrypt the data and derives the AppSKey and the NwkSKey using the AppNonce value received in the Join Accept reply.

3.2.1.4.2 ABP, Activation by Personalization

This method differs from OTAA as the nodes are shipped with the **DevAddr** and both session keys (**NwkSKey and AppSKey**), which should be unique to the node.

Because the nodes already have the information and keys they need, they can begin communicating with the network server without the need for the join message exchange.

3.2.1.4.3 After Connection success

Once a node has joined a LoRaWAN network – either through OTAA or ABP – all future messages will be encrypted and signed using a combination of specific keys: (NwkSKey and AppSKey)

3.2.1.5 Advantages of LoRaWAN

- Open: an open alliance and an open standard. Open technology compared to competitor SigFox
- No restriction in maximum number of daily messages (compared to SigFox limitation of 140/day)
- Operates on free(unlicensed) frequencies, no upfront licensing cost to use the technology
- Low power means long battery life for devices, can last for 2–5 years (Class A and Class B), and wide coverage area measured in kilometers
- Single LoRa Gateway device is designed to take care of thousands of end devices or nodes
- It is easy to deploy due to its simple architecture (Wireless, easy to set up and fast deployment and Low(er) connectivity costs)
- It is widely used for M2M/IoT applications
- Better payload size (100 bytes), compared to SigFox which is 12 bytes
- Low bandwidth makes it ideal for practical IoT deployments with less data and/or with data transmissions which aren’t constant.
- Security: a layer of security for the network and one for the application with AES encryption.
- Fully bi-directional communication.

3.2.1.6 Disadvantages of LoRaWAN

- Not for large data payloads, payload limited to 100 bytes.
- Not for continuous monitoring (except Class C devices).
- Not ideal candidate for real time applications requiring lower latency and bounded jitter requirements.
- The proliferation of LPWAN technologies (SigFox and particularly LoRaWAN), poses co-existence challenges as the deployment of gateways populate urban areas.
- Disadvantage of open frequency is that you may get interference on that frequency and the data rate may be low.4

3.2.1.7 Building a LoRaWAN Gateway

3.2.1.7.1 Open source LoRa Gateway

There is a lot of embedded Gateway board (Open source) that we can use to build an IoT gateway, in this research we will focus on the IC880A-SPI.

- **What is the IC880A-SPI**

This integrates two Semtech SX1257 transceiver ICs plus an SX1301 baseband processor. A combination that is able to emulate 49x LoRa demodulators, with 10 parallel demodulation paths, in order to receive up to 8 LoRa packets simultaneously sent with different spreading factors on different channels.

- **Before we start we need to do these steps**

This guide focuses on using Wifi as backhaul, but you could also use Ethernet or 3G/4G (the less noisy, the better). But if you choose WiFi instead of Ethernet, then try to use a dongle with external antenna and move the antenna outside the enclosure to have less noise inside the box.

- Then, we should install the Raspbian Stretch Lite OS on the RPi.
- Then enables SSH on the RPi upon startup by creating an empty file named "ssh" (without extension) onto the boot partition of the SD card.
- Now turn off power on the PI then connect the WiFi dongle to PI and connect the pigtail to iC880a (never power up without the antenna!).
- Connect the jumper wires between the two boards using the following table:

iC880a pin	Description	RPi physical pin
21	Supply 5V	2
22	GND	6
13	Reset	22
14	SPI CLK	23
15	MISO	21
16	MOSI	19
17	NSS	24

Table 10: Connection between Rpi and IC880a

- **Setting up the Getaway**

- Plug the power supply of the PI which will also power the concentrator board
- From a computer in the same LAN, ssh into the RPi using the default hostname:

```
local $ ssh pi@raspberrypi.local
```

- Default password of a plain-vanilla RASPBIAN install for user pi is `raspberry`.
- Use `raspi-config` utility to enable SPI and also to expand the filesystem:

```
[5] Interfacing options -> P4 SPI
[7] Advanced options -> A1 Expand filesystem
```

- Reboot (it will ask on exit, but you can do it manually with `sudo reboot`)
- Configure locales and time zone:

```
$ sudo dpkg-reconfigure locales
$ sudo dpkg-reconfigure tzdata
```

- Make sure you have an updated installation and install git:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install git
```

- Create new user for TTN and add it to sudoers

```
$ sudo adduser ttn
$ sudo adduser ttn sudo
```

- To prevent the system asking root password regularly, add TTN user in sudoers file
  `$ sudo visudo`
  Add the line `ttn ALL=(ALL) NOPASSWD: ALL`

⚠Beware this allows a connected console with the ttn user to issue any commands on your system, without any password control. This step is completely optional and remains your decision.

- Logout and login as ttn and remove the default pi user:
  `$ sudo userdel -rf pi`

- Configure the wifi credentials :
  `$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf`

  Then add the following block at the end of the file, replacing SSID and password to match your network:

```
network={
ssid="The_SSID_of_your_wifi"
psk="Your_wifi_password"
        }
```

- Clone the installer and start the installation

```
$ git clone -b spi https://github.com/ttn-zh/ic880a-gateway.git ~/ic880a-gateway
$ cd ~/ic880a-gateway
$ sudo ./install.sh spi
```

The installation step will ask you if you want to enable remote configuration. Type 'y' or 'yes' and continue with the installation.

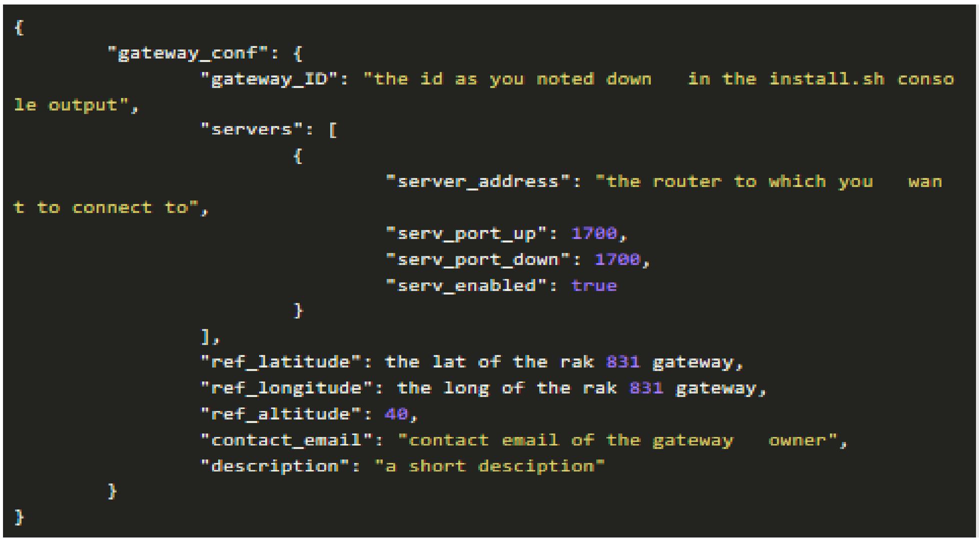
At the start of the command line install, the script would show you the gateway EUI which is important for the next steps.

If you want to use the remote configuration option, please make sure you have created a JSON file named as your gateway EUI (e.g. B827EBFFFE7B80CD.json) in the Gateway Remote Config repository here: https://github.com/ttn-zh/gateway-remote-config.

Fork the repo, add your <EUI>.json file with the proper configuration and then commit the forked repo.

Once done, send a pull request to the master repo and the file should show up in the repo the next day.

A simple json is show below:



```
{
        "gateway_conf": {
                "gateway_ID": "the id as you noted down   in the install.sh conso
le output",
                "servers": [
                        {
                                "server_address": "the router to which you   wan
t to connect to",
                                "serv_port_up": 1700,
                                "serv_port_down": 1700,
                                "serv_enabled": true
                        }
                ],
                "ref_latitude": the lat of the rak 831 gateway,
                "ref_longitude": the long of the rak 831 gateway,
                "ref_altitude": 40,
                "contact_email": "contact email of the gateway   owner",
                "description": "a short desciption"
        }
}
```

Figure 32: Content of Json file need for remotely access GTW

For a list of valid routers check the link here: https://www.thethingsnetwork.org/wiki/Backend/Connect/Gateway

By default, the installer changes the hostname of your Raspeberry Pi to ttn-gateway (to prevent collisions with other Raspberry Pis in your network). You can override this in non-remote configuration mode.

HURRAY your gateway should now work. Make sure you restart the gateway the next day for your json file to be downloaded properly to the RPi3.

Note that the global_config.json in needs to be adjusted as per: https://github.com/TheThingsNetwork/gateway-conf/blob/master/US-global_conf.json

For those looking to use the mp_pkt_fwd instead of the old poly packet forwarder heard over here and install the same with the instruction provided: https://github.com/kersing/packet_forwarder/tree/master/mp_pkt_fwd.

Again you can see the global_conf.json file in the root of the project just make sure you edit the file (imp sections described below) and copy the same to the bin folder after compilation.

- **Some configurable entities in the global_conf.json**

The global_conf.json file can be found in ./bin/global_conf.json from the base of your project directory after the install script has run. Here is a list of some entities that you might want to edit in the global_conf.json file for your particular gateway configuration:

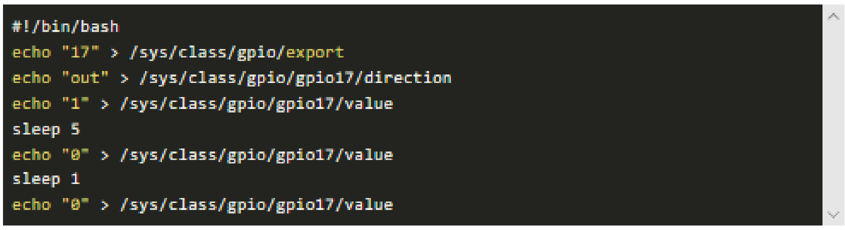
1. "radio_0" or "radio_1" configuration, especially the Frequency parameter and the min and max frequency sweep parameters.
2. "gateway_conf" section. Especially the gateway ID or the EUI of your gateway.
3. Server up and down port in the same gateway_conf object along with your TTN server address of the address of your own application server if it is available.

- **Resetting the board**

Whenever we start the raspberry pi, it is a good practice to reset the attached LoRa module. There are two ways to do it:

- **Via Shell script**

A small shell script can be written to reset the LoRa GW before the LoRa driver can access the hardware. The content of the shell script can look like the following example (which assumes that the GPIO 17 of the Raspberry Pi is connected to the reset pin of the IC880a):



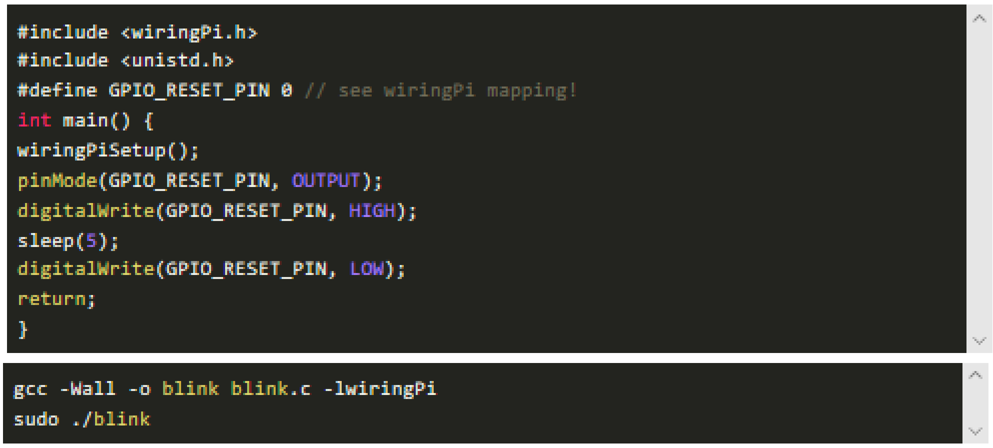
```
#!/bin/bash
echo "17" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio17/direction
echo "1" > /sys/class/gpio/gpio17/value
sleep 5
echo "0" > /sys/class/gpio/gpio17/value
sleep 1
echo "0" > /sys/class/gpio/gpio17/value
```

Figure 33: Reset GTW by using bash script

These lines can be stored in a file called “ic880a_reset.sh”. The user must call this script once after every boot up in order to get the concentrator IC in a clean state.

- **Via Wiring pi**

If the host system is a Raspberry Pi the user can write a small C-Tool to reset set the IC. In order to access the GPIO pins of the Raspberry Pi there is a library called “wiring Pi” that takes care of the low level details. The library can be downloaded from http://wiringpi.com. Please refer to this site to get installation and usage instructions. The content of the RAK831_reset.c file can look like the following:



```
#include <wiringPi.h>
#include <unistd.h>
#define GPIO_RESET_PIN 0 // see wiringPi mapping!
int main() {
wiringPiSetup();
pinMode(GPIO_RESET_PIN, OUTPUT);
digitalWrite(GPIO_RESET_PIN, HIGH);
sleep(5);
digitalWrite(GPIO_RESET_PIN, LOW);
return;
}
```

```
gcc -Wall -o blink blink.c -lwiringPi
sudo ./blink
```

Figure 34: Reset GTW using C code

The user must call this tool once after every boot up in order to get the concentrator IC in a clean state.

- **Register the gateway to the TTN network**

To get your nodes to send data to the cloud TheThingsNetwork provides a cloud service to parse and store the data sent by lora nodes via a lora gateway. You need to register yourself with the thethingsnetwork.org and follow the instructions to register your gateway:

https://www.thethingsnetwork.org/docs/gateways/registration.html

- **Troubleshooting the gateway**

If you are not able to see the RX light on the IC880a turn red check the following:

- Make sure your <EUI>.json file is committed in the GitHub repo https://github.com/ttn-zh/gateway-remote-config
- If you enter the following commands. Sudo tail /var/log/syslog -- when it is online, you will see messages from the TTN-Gateway (might need to look beyond the last few lines).
- Make sure your connections are good and that the IC880a power supply LED is bright RED. <DIG>
- On the gateway UI page in the TTN website make sure te gateway shows up as connected.
- Global Config: The global_config.json in needs to be adjusted as per: https://github.com/TheThingsNetwork/gateway-conf/blob/master/US-global_conf.json

For some advanced troubleshooting visit: https://www.thethingsnetwork.org/docs/gateways/troubleshooting/

- **Checking connection to TTN**

- The best way to check if the gateway is working is registering it on the TTN Console.
- Login to the thingsnetwork.org Console
- Click on Gateways -> register gateway
- Enable checkbox I'm using the legacy packet forwarder
- Enter your Gateway EUI (if is printed on start and end of the installer)
- Enter any description
- Select Europe 868Mhz as frequency plan
- Select the correct antenna placement according to your plans
- Confirm clicking Register gateway

3.2.2 SigFox

Sigfox was created by the French company by the same name. One of the major differences between it and LoRaWAN is that Sigfox owns all of its technology from the edge to the server and endpoint, and it effectively functions as the supplier of the entire ecosystem or, in some cases, as the network operator itself. However, the company allows its endpoint technology to be used free of charge by any organization that agrees to its terms, so it has been able to establish relationships with major IoT device suppliers and even some wireless carriers. Along with LoRaWAN, Sigfox continues to gain in market share, especially in Europe where its transmission length adheres to European Union guidelines. The version used in the U.S. is significantly different in order to meet Federal Communication Commission (FCC) rules. The only drawback of Sigfox is its proprietary nature.

Frequency band	902Mhz (America)
	868Mhz (Europe)
Number of channels	360 channels for 2.4 Ghz band
Channel bandwidth	100hz-1.2Khz band
Maximum data rate	100bps-600bps
Protocol data unit	12 bytes header + (0 to 12) bytes
Channel coding	Ultra-narrow band coding
Channel modulation	BPSK and GFSK
Receiver sensitivity	-137dBm
Transmission range	10-50Km
Battery lifetime	<10 years

Table 35: SigFox Features

- **Drawbacks of SigFox**

- It is not deployed everywhere, so it won't work for a large number of use cases currently.
- Communication is better headed up from the endpoint to the base station.
- It has bidirectional functionality, but its capacity from the base station back to the endpoint is constrained, and you'll have less link budget going down than going up.
- Mobility is difficult with Sigfox devices.
- Sigfox is uplink only. Though limited downlink is possible, it has a different link budget and is very restricted.

3.2.2.1 SigFox architecture

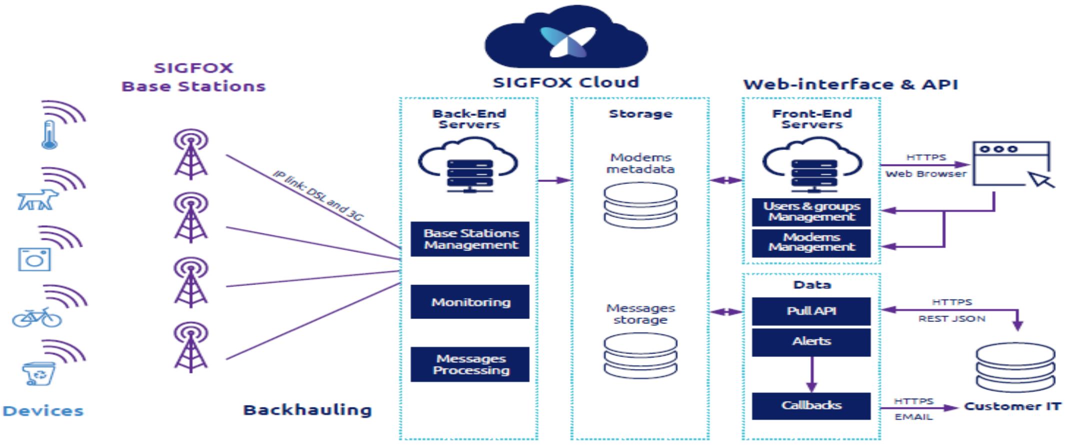


Figure 36: SigFox system connection

The Sigfox network consists of: objects (end user devices), SigFox gateway or base stations, and SigFox cloud and application servers.

Sigfox objects are connected with Gateway using **star topology**.

There is direct secure point to point link between SigFox gateways and SigFox cloud.

The cloud interfaces with servers using different protocols such as SNMP, MQTT, HTTP, IPv6 etc. as per end applications.

- **SigFox functionality**

The data is sent over the air to the base stations, then goes through the backhaul. The backhaul generally uses DSL connectivity and 3G or 4G as a back-up. When one of the two is not available, satellite connectivity can be used as an alternative back-up technology.

The back-end handles message processing. There are potentially lots of replicates of the same message that arrive on the core network but only one should be stored. The core network servers also monitor the status of the network and manages the base stations globally.

The network infrastructure also stores the messages in two locations: the metadata can be used for building services on one hand and the customers' messages so that customers can retrieve them later on the other hand.

Finally, the web interface and the API allow customers to access their messages. They can either access the platform through their web browser or use a REST API to synchronize them with their IT system and push downlink messages to the device.

3.2.2.2 SigFox protocol stack

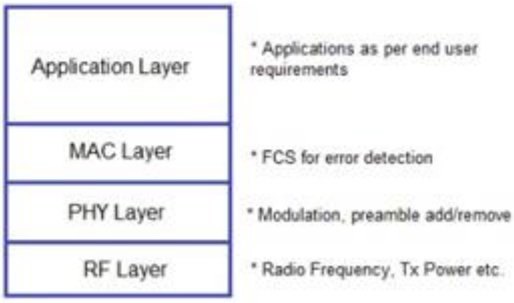

Application Layer
* Applications as per end user requirements
MAC Layer
* FCS for error detection
PHY Layer
* Modulation, preamble add/remove
RF Layer
* Radio Frequency, Tx Power etc.

Figure 37: SigFox protocol layers

3.2.2.2.1 Application Layer

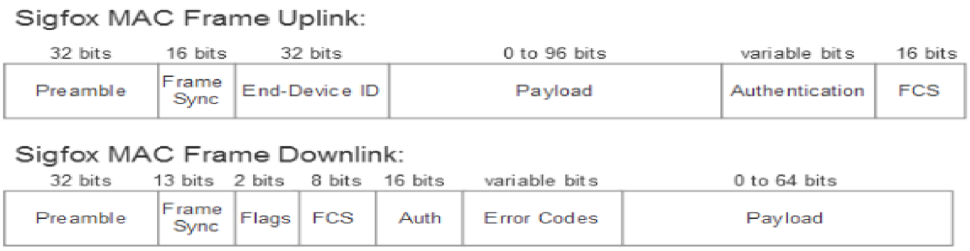
The different applications are supported in this LTN technology. There are various interfaces/protocols between WAN (i.e. cloud) and servers to support the same e.g. SNMP, HTTP, MQTT, IPv6 etc.

3.2.2.2.2 MAC Layer

Each device in a Sigfox network has a unique Sigfox ID:

- The ID is used for the routing and signing of messages.
- The ID is used to authenticate the Sigfox device.

Another characteristic of Sigfox communication is that it uses **fire and forget**. Messages are not acknowledged by the receiver. Rather, a message is sent three times on three different frequencies at three different times by the node. This ensures the integrity of transmission of a message. The fire and forget model has no way of ensuring that the message was actually received, so it's up to the transmitter to do as much as possible to ensure accurate transmission:



Sigfox MAC Frame Uplink:

32 bits	16 bits	32 bits	0 to 96 bits	variable bits	16 bits
Preamble	Frame Sync	End-Device ID	Payload	Authentication	FCS

Sigfox MAC Frame Downlink:

32 bits	13 bits	2 bits	8 bits	16 bits	variable bits	0 to 64 bits
Preamble	Frame Sync	Flags	FCS	Auth	Error Codes	Payload

Figure 38: SigFox uplink and downlink MAC Frame

- It takes care of authentication of end users and it takes care of error detection using FCS.

- **FCS error detection**

A **frame check sequence** (**FCS**) refers to an error-detecting code added to a frame in a communications protocol. All frames and the bits, bytes, and fields contained within them, are susceptible to errors from a variety of sources. The FCS field contains a number that is calculated by the source node based on the data in the frame. This number is added to the end of a frame that is sent. When the destination node receives the frame the FCS number is recalculated and compared with the FCS number included in the frame. If the two numbers are different, an error is assumed and the frame is discarded.

3.2.2.2.3 PHY Layer

- It handles MAC frame during transmission and during reception.
- It uses BPSK modulation in the uplink and GFSK in the downlink (in Ultra Narrow Band implementation technology). Messages are very short. User has a maximum of 12 bytes of info available.
- It use direct-sequence spread spectrum (DSSS) with orthogonal signaling (in Orthogonal Sequence Spread Spectrum implementations).
- The SigFox end device adds preamble during transmit (i.e. in uplink from end device to Gateway) and removes it during receive operation (at Gateway receiver part). The process is followed in Gateway transmit and end device receive link i.e. in the downlink direction. Preamble is used for synchronization purpose.
- SIGFOX currently has a tiered option plan for how many uplink transmissions you are allocated per day, as well as how many downlink transmissions you get from the main network station to your device (which is a different signal, using GFSK at 600 Bd).

  Platinum: 101 to 140 uplink messages + 4 downlink
  Gold: 51 to 100 uplink messages + 2 downlink
  Silver: 3 to 50 uplink messages + 1 downlink
  One: 1 to 2 uplink messages + no downlink

- **GFSK modulation**

Frequency of carrier signal is varied to represent binary 1 or 0, peak amplitude & phase remain constant during each bit interval.

$$s(t) = \begin{cases} A\cos(2\pi f_1 t), & \text{binary } 0 \\ A\cos(2\pi f_2 t), & \text{binary } 1 \end{cases}$$

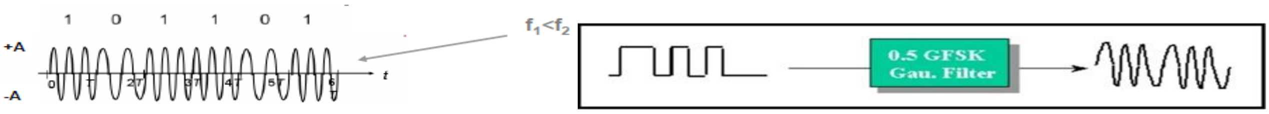


Figure 39: GFSK system modulation technic

In Gaussian FSK modulation (GFSK) data is encoded in the form of variations of frequency in a carrier in a similar manner to FSK. Therefore, the modulator used can be the same that is used for FSK modulation.

However, the impulses pass through a Gaussian filter before entering the pulse modulator to decrease the spectral width of the same. The Gaussian filter is a kind of pulse formatter used to smooth the transition between the values of the impulses.

GFSK modulation is used in Bluetooth systems, since it provides a better spectral efficiency compared to FSK.

- **DSSS modulation**

D*irect*-**s***equence* **s***pread* **s***pectrum (DSSS)* is a spread spectrum radio.

A key element in spread spectrum communication is the use of a spreading sequence. The spreading sequence, c(t), is a sequence of binary digits shared by the transmitter and receiver. In the common scheme known as direct-sequence spread spectrum (DSSS), spreading consists of multiplying (XOR) the input data by the spreading sequence or (chipping code), where the bit rate of the spreading sequence is higher than that of the input data. When the signal is received, the spreading is removed by multiplying with the same spreading code, exactly synchronized with the received signal.

The chipping code is a redundant bit pattern for each bit that is transmitted, which increases the signal's resistance to interference.

The resulting data rate is consequently that of the spreading sequence. This increases the transmitted data rate and therefore increases the required bandwidth. The redundancy of the system is also increased (If one or more bits in the pattern are damaged during transmission, the original data can be recovered due to the redundancy of the transmission).

The spreading codes are chosen so that the resulting signal is noise-like; therefore, there should be an approximately equal number of ones and zeros in the spreading code and few or no repeated patterns.

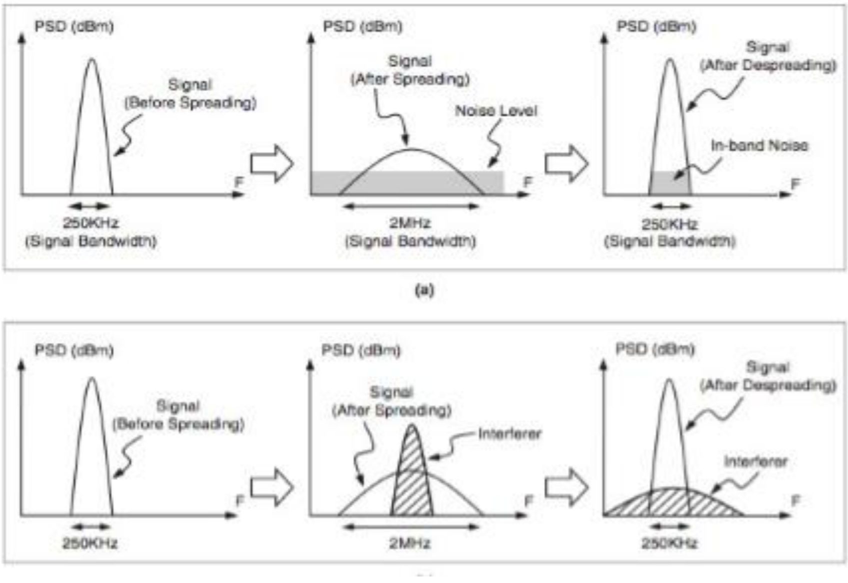


Figure 40: DSSS modulation technic

3.2.2.2.4 RF Layer

This layer takes care of frequency assignment and transmit/receive power requirements at Sigfox end points and base stations.

- There are two implementations of the radio layer in SigFox system viz. UNB (Ultra Narrow Band) and OSSS (Orthogonal Sequence Spread Spectrum).
- There are different frequency spectrum allocated in USA (915 MHz), Europe (868 MHz), China (433 MHz) and Japan.
- The max. Transmit power of 25 mW is specified for UNB uplink transmissions for 868 MHz Band.
- Each packet transmits in about 2 seconds, and each transmission from the IoT SIGFOX device to the main station consists of 3 of these packets transmitted on 3 pseudorandom frequencies, each transmission offset by about 45 ms. this is done as a redundancy measure.
- The Sigfox receiver sensitivity should be better than -135 dBm.
- Sigfox is using 192 KHz of the publicly available band to exchange messages over the air. The modulation is Ultra-Narrow band. Each message is 100 Hz wide and transferred with a data rate of 100 or 600 bits per second depending on the region.

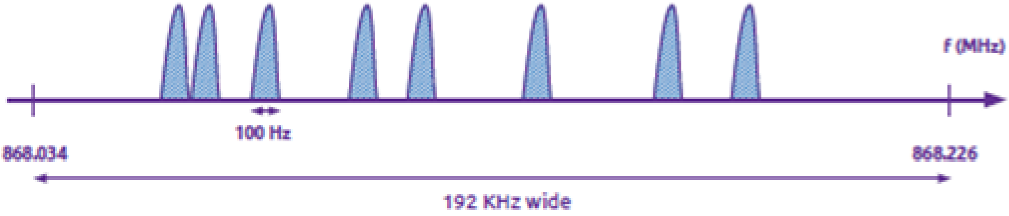


Figure 41: SigFox technology based in Ultra-Narrow Band

- **Ultra-Narrow band (UNB) transmission**

In communication engineering, **Ultra NarrowBand** (UNB) systems are those in which the channel has a very narrow bandwidth.

UNB technology is often used where links from very high numbers of devices are needed, with relatively small amounts of data being exchanged on each link. Some such applications can be found in the Internet of things, with UNB being one of the technologies that have been used to implement Low-Power Wide Area Networks. Short, infrequent transmissions with low transmit power can enable long-life, battery-powered operation of UNB devices connected in a LPWAN.

Narrower band = lower data rate = lower noise floor = higher sensitivity = longer distance

Ultra-narrow band modulation has difficulties in the receiver side that Sigfox solved using modern cognitive radio techniques.

- **Cognitive radio technic**

C*ognitive* **R***adio* (CR) is an adaptive, intelligent radio and network technology that can automatically detect available channels in a wireless spectrum and change transmission parameters enabling more communications to run concurrently and also improve radio operating behavior.

Cognitive radio uses a number of technologies including Adaptive Radio (where the communications system monitors and modifies its own performance) and Software Defined Radio (SDR) where traditional hardware components including mixers, modulators and amplifies have been replaced with intelligent software.

3.2.2.3 SigFox security

The Sigfox ecosystem integrates the security by-default:

- Authentication + integrity + anti-replay on messages propagated on the network.
- Cryptography based on Advanced Encryption Standard (AES) with no key OTA transmission.
- Payload encryption as an option to ensure the confidentiality of the data.
- Isolation of each part of the network and assess the risks so that in case of a hack only a minor segment of the network is impacted.

▪ On the device side, Sigfox had defined three different levels of security. Depending on the use case and its sensitivity, the device maker or the application provider will decide which level to implement:

- Medium level: the security credentials are stored in the device;
- High level: the security credentials are stored in a S/W based protected area;
- Very high level: the security credentials are stored in a secure element.

The secure element also helps to encrypt the data that is transferred over the network. Only the device and the end-customer know the secret key. The algorithm does not impact the size of the payload. While the message is encrypted, the payload is still 12-bytes long.

Throughout the path of the message, the Sigfox network makes sure that the device ID has not been duplicated. In the case of a corrupted device, a blacklist list mechanism will prevent the communication of this device.

From the start, Sigfox has designed the network with security in mind, separating functions onto several servers. For instance, the server generating IDs has a reinforced security.

List of figures

List of tables

References

[1] "ZigBee / IEEE 802.15.4", URL:
https://fenix.tecnico.ulisboa.pt/downloadFile/563568428736043/Zigbee-sintese.pdf

[2] Matt Hillman, "An Overview of ZigBee N etworks,A guide for implementers and security testers", URL:
https://www.mwrinfosecurity.com/assets/Whitepapers/mwri-zigbee-overview-finalv2.pdf

[3] Stefan Schmidt, "6LoWPAN: An Open IoT, Networking Protocol», OpenIoT Summit 2016, San Diego, URL:
http://events17.linuxfoundation.org/sites/events/files/slides/6lowpan-openiot-2016.pdf

[5] Prof. Dr. Mesut Güneş, "Embedded Internet and the Internet of Things WS 12/13",Freie Universität Berlin, URL:
http://docplayer.net/8829168-Embedded-internet-and-the-internet-of-things-ws-12-13.html

[6] "6LoWPAN CHAPTER 3", URL:
https://shodhganga.inflibnet.ac.in/bitstream/10603/36007/3/chapter3.pdf

[7] Ruchi Garg and Sanjay Sharma, "A study on Need of Adaptation Layer in 6LoWPAN Protocol Stack", I.J. Wireless and Microwave Technologies, 2017, 3, 49-57, URL:
http://www.mecs-press.org/ijwmt/ijwmt-v7-n3/IJWMT-V7-N3-5.pdf

[8] Hayder A. A. Al-Kashoash and Andrew H. Kemp, "Comparison of 6LoWPAN and LPWAN for the Internet of Things", Article in Australian Journal of Electrical and Electronics Engineering, December 2017, URL:
https://www.researchgate.net/publication/316236998_Comparison_of_6LoWPAN_and_LPWAN_for_the_Internet_of_Things

[9] Politecnico di Milano, Advanced Network Technologies Laboratory, "The Long Range Wide Area Network - LoraWAN", URL:
http://home.deib.polimi.it/cesana/teaching/IoT/2017/classes/4.LoraWAN.pdf

[10] Ukko-Pekka Peura, "LORAWAN OPTIMIZATION FOR A BATTERY POWERED SEN-SOR NETWORK", OULU UNIVERSITY OF APPLIED SCIENCES, spring 2018, URL:
https://www.theseus.fi/bitstream/handle/10024/146549/Peura_Ukko-Pekka.pdf?sequence=1&isAllowed=y

[11] SEMTECH AN1200.22, "LoRa™ Modulation Basics", Revision 2, May 2015, URL:
https://www.semtech.com/uploads/documents/an1200.22.pdf

[12] Jonathan de Carvalho Silva, Joel J. P. C. Rodrigues, Antonio M. Alberti, Petar Solic, Andre L. L. Aquino, "LoRaWAN - A Low Power WAN Protocol for Internet of Things: a Review and Opportunities", 2ND INTERNATIONAL MULTIDISCIPLINARY CONFERENCE ON COMPUTER AND ENERGY SCIENCE (SPLITTECH 2017) Split, Croatia July 12–14, 2017, URL:
https://www.researchgate.net/publication/318866065_LoRaWAN_-_A_Low_Power_WAN_Protocol_for_Internet_of_Things_a_Review_and_Opportunities

[13] "LoRa Specification", LoRa Alliance, Inc. 2400 Camino Ramon, Suite 375, San Ramon, CA 94583, URL:
https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf

[14] "LoRa APPLICATIONS", METOVA, URL:
https://cdn2.hubspot.net/hubfs/2098550/Downloadable_PDFs/LoRa/LoRa%20Applications.pdf

[15] Raj Jain,"Low Power WAN Protocols for IoT: IEEE 802.11ah, LoRaWAN", Washington University in Saint Louis Saint Louis, MO 63130, URL:
https://www.cse.wustl.edu/~jain/cse574-16/ftp/j_14ahl4.pdf

[16] Hadi Jamali-Rad, Johannes van den Brand, Xander Campman, "IoT-based wireless seismic quality control", Article in The Leading Edge · March 2018, URL:
https://www.researchgate.net/publication/323620460_IoT-based_wireless_seismic_quality_control

[17] Ismail Butun, Nuno Pereira and Mikael Gidlund, "Demystifying the Security of LoRaWAN v1.1", 21 November 2018, URL:
https://www.preprints.org/manuscript/201811.0531/v1

[18] Thomas Telkamp, "LoRa, LoRaWAN, and the challenges of long-range networking in shared spectrum", Cogni&ve Radio Pla/orm NL, december 2015, URL:
https://www.kivi.nl/uploads/media/584e9180f3822/cr-platform-lora-workshop-shared.pdf

[19] LoRa Alliance™ Strategy Committee, "GEOLOCATION WHITEPAPER", January 2018, URL:
https://lora-alliance.org/resource-hub/lora-alliance-geolocation-whitepaper

[20] "Analog Transmission of Digital Data: ASK, FSK, PSK, QAM", CSE 3213, Fall 2010, Instructor: N. Vlajic, URL:
https://www.eecs.yorku.ca/course_archive/2015-16/F/3213/CSE3213_10_ShiftKeying_F2015_posted_part1.pdf

[21] SIMONE CIRANI, GIANLUIGI FERRARI, MARCO PICONE, LUCA VELTRI, " INTERNET OF THINGS ARCHITECTURES, PROTOCOLS AND STANDARDS", URL:
https://www.amazon.com/Internet-Things-Architectures-Protocols-Standards/dp/1119359678

[22] Sigfox, "Technical Overview", May 2017, URL:
https://www.disk91.com/wp-content/uploads/2017/05/4967675830228422064.pdf

[23] Laurent Andriantsiferana, Henri Parfait, "LORA Training Module 1: End-to-End Architecture Functional and System Architecture", CISCO Version 5

[24] Laurent Andriantsiferana,"LORA Training: Module 2 LORAWAN: Architecture, PHY, MAC, Flows, Security", CISCO Version 2

[25] "Cisco Gateway Management, Training on Field Network Director Concepts & Use", CISCO Version 1