

# QuakeSense

## DESCRIZIONE DEL PROGETTO, OBIETTIVI E FINALITÀ

QuakeSense è un progetto che ha come obiettivi principali la rilevazione e il monitoraggio di fenomeni sismici attraverso dei nodi LoRa e la visualizzazione dei dati raccolti su una piattaforma Cloud.

Il sistema consente anche il monitoraggio di parametri ambientali come pressione, temperatura e umidità relativa dell'aria.

Il sistema è costituito dai seguenti componenti:

- Dei nodi LoRa realizzati con delle schede di sviluppo dotate di microcontrollore (MCU) collegato ai seguenti componenti:
  - un accelerometro a tre assi per rilevare l'attività sismica
  - un modulo radio basato sullo standard LoRa per trasmettere i dati
  - un modulo GPS per ottenere la posizione geografica del nodo LoRa
  - una batteria LiPo per alimentare il nodo LoRa

In presenza di un fenomeno sismico, il MCU della scheda elabora i dati provenienti dall'accelerometro e calcola alcuni parametri importanti che caratterizzano il moto sismico come:

- l'accelerazione di picco al suolo considerando le due componenti orizzontali x e y (PGHA - Peak Ground Horizontal Acceleration)
- l'accelerazione di picco al suolo considerando la componente verticale z (PGVA - Peak Ground Vertical Acceleration)
- la durata del terremoto intesa come l'intervallo di tempo compreso tra il primo e l'ultimo superamento di una soglia di accelerazione fissata a 80 mg.

Ciascun nodo LoRa riceve ed elabora anche i dati provenienti da un modulo GPS ovvero i valori di latitudine, longitudine, altitudine, data e orario. Tali dati vengono trasmessi insieme a quelli relativi all'attività sismica utilizzando il protocollo LoRa.

Attraverso l'aggiunta di sensori di temperatura, umidità e pressione è possibile monitorare anche alcuni parametri ambientali della zona i cui è presente il dispositivo. Tali parametri ambientali sono trasmessi ad un gateway utilizzando sempre il protocollo LoRa.

- Un gateway che ha il compito di raccogliere i dati provenienti dai nodi della rete LoRa, elaborarli e trasmetterli ad una piattaforma Cloud. Il gateway può essere implementato attraverso una scheda di sviluppo basata su microcontrollore oppure attraverso un computer a singola scheda (SBC - Single Board Computer). Per trasmettere i dati ricevuti alla piattaforma Cloud, il gateway utilizza il protocollo applicativo MQTT basato sullo stack TCP/IP attraverso un canale di comunicazione backhaul di tipo wireless (WiFi, 3G, 4G, ...) oppure wired (Ethernet).

- Una piattaforma IoT basata sulla tecnologia Cloud che consente di:
  - visualizzare dei dati raccolti in tempo reale (real-time) attraverso una dashboard realizzata attraverso una interfaccia utente semplice ed intuitiva.
  - elaborare i dati attraverso dei tools in modo da realizzare delle applicazioni interattive e dei meccanismi di notifica basati sui dati ricevuti.

## Implementazione del sistema

- **LoRa Nodes:**
  - MCU → STM32 Nucleo F401RE
  - LoRa & GPS module → Dragino LoRa/GPS shield
  - 3-axis accelerometer & environmental sensors → X-NUCLEO-IKS01A2
  - Power → Adafruit PowerBoost 500 shield + LiPo battery
  - Software → QuakeSense\_Node.ino (Arduino sketch)
  - LoRa Mode 3: BW = 125 kHz, CR = 4/5, SF = 10
- **LoRa Gateway:**
  - MCU & WiFi module → B-L475E-IOT01A2 STM32L4 Discovery kit
  - LoRa module → Dragino LoRa/GPS Hat
  - Software → QuakeSense\_Gateway.ino (Arduino sketch)
  - LoRa Mode 3: BW = 125 kHz, CR = 4/5, SF = 10
- **Cloud Platform:**
  - Adafruit IO
  - Protocols: MQTT, WiFi

# DESCRIZIONE DEI COMPONENTI UTILIZZATI

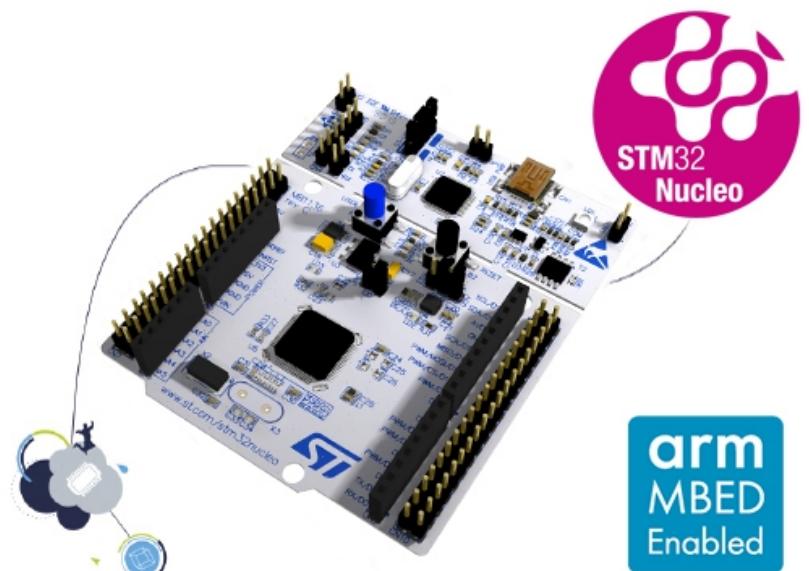
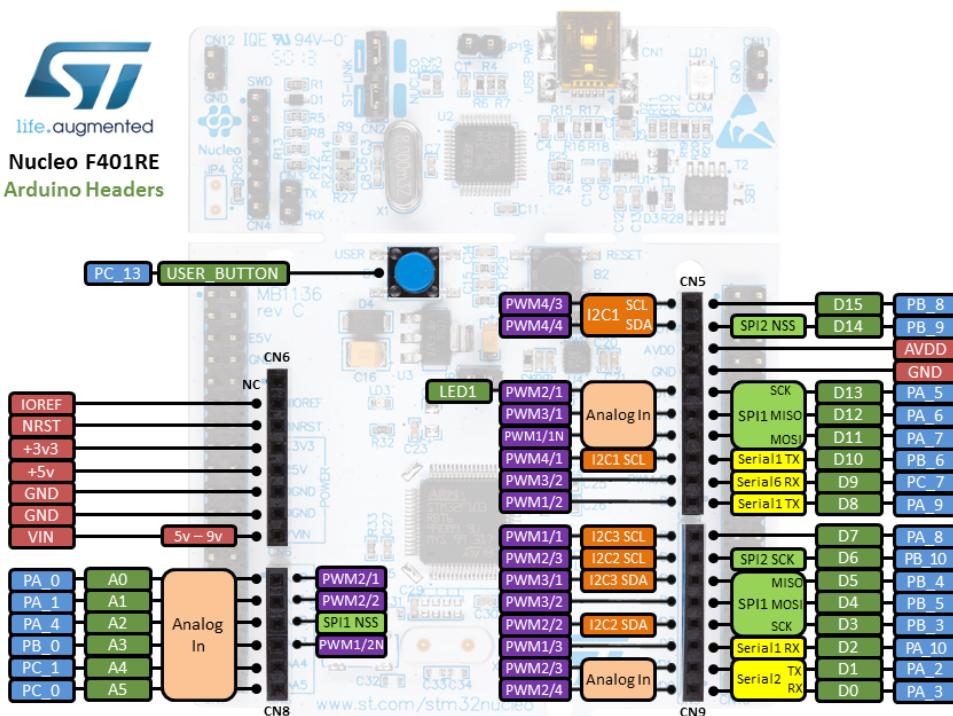
## REALIZZAZIONE DEI NODI DELLA RETE LoRa

I nodi della rete LoRa sono stati implementati utilizzando i seguenti componenti hardware:

### STM32 Nucleo-64 development board

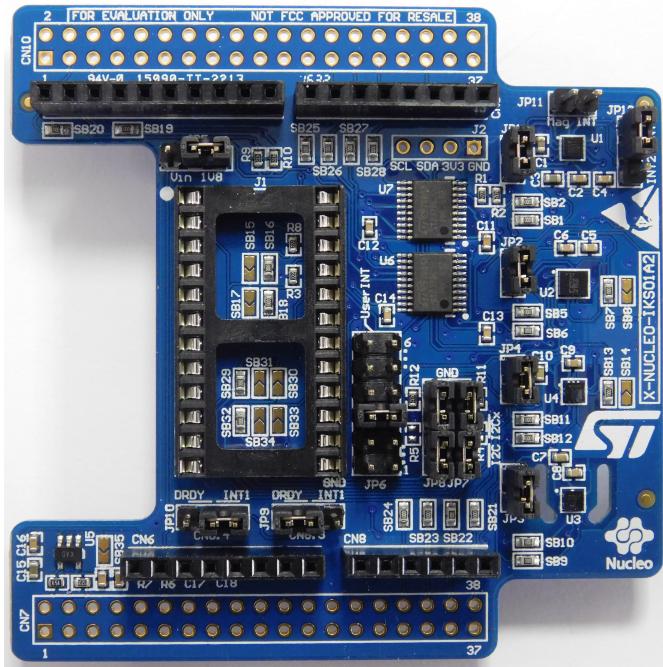
- STM32F401RE ARM® 32-bit Cortex®-M4 MCU(84 MHz) with FPU
- Memory: Flash memory → 512 KB, SRAM → 96 KB
- 1 user LED + 1 reset and user button
- 12-bit ADC with 16 channels, USART/UART (3), I2C (3), SPI (3)
- Arduino™ Uno V3 pinout + ST Morpho pin headers
- On-board ST-LINK/V2-1 debugger/programmer
- Include il supporto per diversi ambienti di sviluppo integrato (IDE - Integrated Development Environment) come Arduino™ (attraverso il progetto STM32duino), IAR™, Keil®, GCC-based IDEs, Arm® Mbed™

Per ulteriori informazioni si può consultare la pagina web ufficiale della scheda: [STM32 Nucleo F401RE ST.com](http://www.st.com/stm32nucleo)



## X-NUCLEO-IKS01A2 - Scheda di espansione per le boards STM32 Nucleo comprendente diversi sensori:

- LSM6DSL: sensore MEMS che integra un accelerometro 3D e un giroscopio 3D.
- LSM303AGR: sensore MEMS costituito da un accelerometro 3D e un magnetometro.
- LPS22HB: sensore di pressione basato su tecnologia MEMS.
- HTS221: sensore di temperatura e umidità relativa di tipo capacitivo.
- Per maggiori informazioni: [X-NUCLEO-IKS01A2 ST.com](http://X-NUCLEO-IKS01A2 ST.com)



## Adafruit PowerBoost 500 Shield + LiPo battery (3.7V - 1200 mAh)

La shield PowerBoost 500 e la batteria LiPo sono utilizzate per alimentare i nodi LoRa.

### Specifiche [batteria LiPo](#):

- Tensione in uscita compresa tra 4.2 V (quando completamente carica) e 3.7 V
- Capacità totale: 1200mAh per un totale di 4.5 Wh
- Connettore JST per collegarla alla scheda di espansione PowerBoost
- Include un circuito di protezione per evitare sovraccarichi o sottocarichi di tensione.

### Specifiche [PowerBoost 500 Shield](#):

- Il convertitore boost incorporato può fornire un valore minimo di corrente pari a 500 mA di corrente e un valore picco pari a 1 A.
- È presente anche un fusibile di per la protezione da correnti più elevate che potrebbero danneggiare il convertitore boost o la batteria.
- Dotata di una porta micro USB per ricaricare la batteria LiPo.

## **Dragino LoRa/GPS Shield - Scheda di espansione costituita da un modulo LoRa (SX1276) e un modulo GPS (MT3339)**

### **Specifiche principali modulo LoRa**

- Based on SX1276 which is a low-power and long-range transceiver.
- Frequency Band: 868 MHz / 433 MHz / 915 MHz.
- 168dB maximum link budget.
- +14 dBm high efficiency PA.
- High sensitivity: down to -148 dBm.
- Low RX current of 10.3 mA, 200 nA register retention.
- Modulations: FSK, GFSK, MSK, GMSK, LoRa™ and OOK.
- Preamble detection.
- 127 dB Dynamic Range RSSI.

### **Specifiche principali modulo GPS**

- Based on Mediatek MT3339, an All-in-One GPS system on a chip (SoC).
- Compliant with GPS, QZSS, SBAS(WAAS/EGNOS/MSAS/GAGAN).
- Serial Interfaces (UART) with adjustable bitrate 4800~115200 bps, default: 9600bps.
- Update rate: up to 10 Hz, default 1 Hz.
- Protocols: NMEA 0183 standard V3.01, PMTK.
- GPS automatic switching between internal patch antenna and external active antenna.
- EASY™ technolory: advanced AGPS technology without external memory.
- AlwaysLocate™: an intelligent controller of periodic mode.
- Timing accuracy: 1PPS out 10ns, Reacquisition Time < 1s.
- Velocity accuracy without aid < 0.1m/s.
- Acceleration accuracy without aid 0.1m/s<sup>2</sup>.
- Sensitivity acquisition -148dBm, Tracking -165dBm, Reacquisition -160dBm.

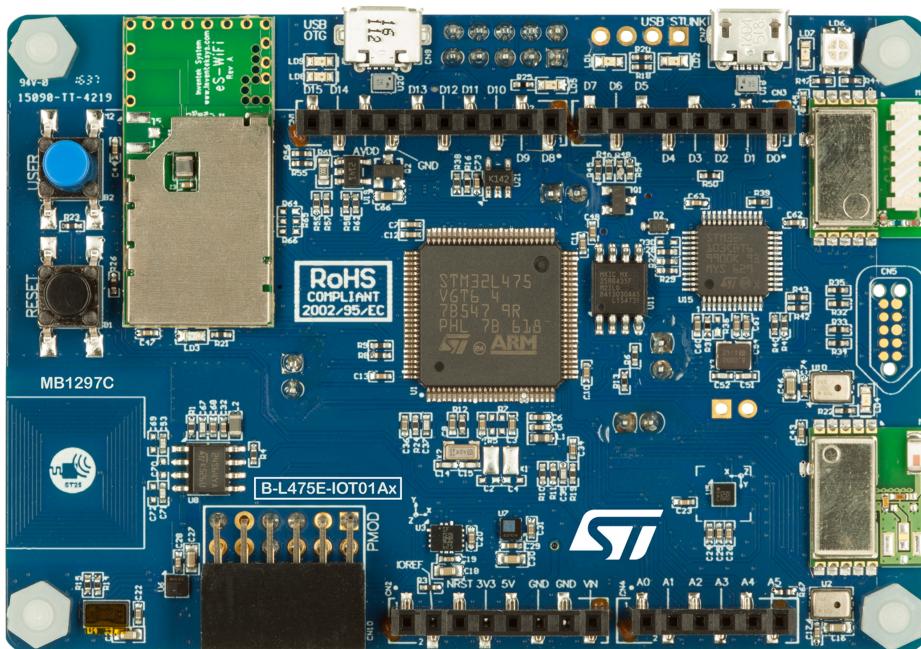


## REALIZZAZIONE DEL GATEWAY

Per la realizzazione del gateway sono stati utilizzati i seguenti componenti:

### B-L475E-IOT01A2 - STM32L4 Discovery kit

- Based on the ultra low-power STM32L475VG MCU (ARM® Cortex-M4).
- 1 Mbyte of Flash memory and 128 kbytes of SRAM.
- Connectivity:
  - Bluetooth® V4.1 module (SPBTLE-RF);
  - Sub-GHz (868 MHz) low-power programmable RF module;
  - Wi-Fi® module Inventek ISM43362-M3G-L44 (802.11 b/g/n);
  - Dynamic NFC tag based on M24SR with printed NFC antenna;
- Integrated sensors:
  - HTS221: capacitive digital sensor for relative humidity and temperature;
  - LIS3MDL: high-performance 3-axis magnetometer;
  - LSM6DSL: 3D accelerometer and 3D gyroscope;
  - LPS22HB: 260-1260 hPa absolute digital output barometer.
  - VL53L0X: Time-of-Flight and gesture-detection sensor;
- Arduino™ expansion connectors.
- On-board ST-LINK/V2-1 debugger/programmer.
- Support of a wide choice of Integrated Development Environments (IDEs) including Arduino (STM32duino project), IAR™, ARM® mbed.



**Dragino LoRa/GPS Shield oppure Dragino LoRa/GPS Hat**  
**Schede di espansione che integra un modulo LoRa (SX1276) e un**  
**modulo GPS (MT3339)**

Stesse specifiche tecniche descritte in precedenza.



## SOFTWARE UTILIZZATO

### Arduino IDE

- [Arduino](#) è un sistema di sviluppo integrato Open Source che consente di scrivere dei programmi detti “sketches” in linguaggio C/C++ e di effettuare l’upload su una scheda supportata.
- Scritto in Java e basato sull’ambiente di sviluppo Processing.
- Può essere eseguito su diversi sistemi operativi: Windows, Mac OS X e Linux.
- Ampia [documentazione disponibile](#).

Per poter eseguire gli sketch del progetto QuakeSense è necessario installare le seguenti librerie attraverso il [Library Manager](#) dell’IDE Arduino:

- [STM32duino LSM6DSL](#): libreria per gestire l’accelerometro e il giroscopio del sensore LSM6DSL presente sulla scheda di espansione X-NUCLEO-IKS01A2.
- [STM32duino LPS22HB](#): libreria per gestire il sensore di pressione LPS22HB presente sulla scheda di espansione X-NUCLEO-IKS01A2.
- [STM32duino HTS221](#): libreria per utilizzare il sensore di temperatura e umidità HTS221 della scheda di espansione X-NUCLEO-IKS01A2
- [LoRa](#): libreria per utilizzare il modulo LoRa SX1276
- [STM32duino ISM43362-M3G-L44](#): libreria per utilizzare il modulo WiFi presente sulla scheda B-L475E-IOT01A2 Discovery Kit
- [Adafruit MQTT Library](#): libreria per implementare la comunicazione tra il gateway e la piattaforma Cloud Adafruit IO
- [Adafruit GPS Library](#): libreria per comunicare con il modulo GPS e decodificare i messaggi provenienti dall’interfaccia seriale del modulo GPS.

### Adafruit IO

Per visualizzare e gestire i dati ambientali e quelli associati ad eventi sismici, si utilizza la piattaforma Adafruit IO.

Le caratteristiche principali di Adafruit IO sono:

- Facilità di utilizzo
- Numerose librerie client disponibili per interfacciarsi alla piattaforma (Arduino, Ruby, Python, Node.js)
- Controllo completo sui dati da parte dell’utente:  
l’utente può visualizzare e scaricare in qualunque momento i dati memorizzati sulla piattaforma.
- Sicurezza e privacy:
  - Per accedere alla piattaforma si utilizza username e password
  - Per la trasmissione dei dati e per effettuare qualunque operazione su dati, Feeds e Dashboards è necessaria una chiave d’accesso (AIO key). Tale chiave d’accesso può essere utilizzata sia con le API REST, sia per connettersi utilizzando MQTT.
- Ampia documentazione disponibile riguardo le API REST e MQTT.  
Sono disponibili anche numerosi esempi per le diverse schede di sviluppo supportate.

- Organizzazione efficiente dei dati attraverso numerosi **blocchi** che possono essere aggiunti ad una Dashboard.  
I blocchi principali sono: [Gauge](#), [Number Slider](#), [Momentary Button](#), [Toggle Button](#), [Color Picker](#), [Line Graph](#), [Text Box](#), [Image](#), [Stream](#).

Gli elementi fondamentali del sistema Adafruit IO sono i **Feeds** e le **Dashboards**.

A ciascun Feed sono associati dei metadati che comprendono:

- i dati relativi ad un sensore o dispositivo collegato al Feed.
- la licenza associata ai dati raccolti.
- il tipo di accesso ai dati:
  - public (dati condivisibili)
  - private (dati privati ovvero visualizzabili solo dall'utente)
- descrizione generale dei dati che indica ciò che essi rappresentano o a cosa sono associati.
- data di trasmissione dei dati.

Per ulteriori informazioni sui Feeds si può consultare la seguente [guida](#).

Una Dashboard è un insieme di blocchi implementati attraverso dei widgets. Ciascun blocco può essere associato ad un Feed in modo da gestire i dati relativi del Feed stesso.

Attraverso una Dashboard è possibile:

- visualizzare i dati relativi ai blocchi.
- creare, eliminare e modificare dei blocchi.
- gestire i dati dei Feeds attraverso widgets interattivi.

Ulteriori informazioni relative alle Dashboards di Adafruit IO sono disponibili alla seguente [guida](#).

Una guida ben documentata sulla piattaforma Adafruit IO è disponibile al [seguente indirizzo](#).

Per interfacciarsi con la piattaforma AdafruitIO e per gestire Feeds e Dashboards attraverso l'IDE Arduino sono disponibili due librerie Client:

## 1) Adafruit IO Arduino library

Libreria basata sulle API REST che consentono di gestire i Feeds e le Dashboards utilizzando il protocollo HTTP oppure HTTPS ovvero utilizzando i metodi HTTP per i servizi RESTful:

- GET (Read)
- POST (Create)
- PUT (Update/Replace)
- DELETE (Delete)

Alcune funzionalità offerte dalla libreria Adafruit IO sono:

- Creare un Feed o una Dashboard
- Inviare dei dati ad un Feed attraverso il metodo save()
- Impostare un message handler ovvero una callback per gestire i dati relativi ad un Feed che vengono modificati e inviati dalla piattaforma Adafruit IO.

Per ulteriori informazioni riguardo le REST API si può consultare la seguente [documentazione sulle API REST](#).

## 2) Adafruit MQTT library

Libreria per l'IDE Arduino che consente di gestire i servizi e gli elementi della piattaforma Adafruit IO attraverso il protocollo MQTT.

Alcune funzionalità offerte da questa libreria sono:

- Gestire i dati dei Feed attraverso i metodi:
    - publish() per inviare i dati ad un Feed presente sulla piattaforma.
    - subscribe() per ricevere i dati relativi ad un Feed quando vengono modificati attraverso la piattaforma Adafruit IO.
  - Inizializzare la comunicazione con il broker MQTT presente sulla piattaforma Adafruit IO
  - Gestire le funzionalità di QoS e il topic Will che caratterizzano il protocollo MQTT.
- La libreria Adafruit MQTT supporta solo i livelli 0 e 1 di QoS.

Per ulteriori informazioni riguardo la libreria Adafruit MQTT si può consultare la seguente [guida](#).

# TECNOLOGIE E PROTOCOLLI UTILIZZATI

## LA TECNOLOGIA LoRa

Negli ultimi anni si sta verificando un notevole sviluppo delle Low Power Wide Area Networks (LPWANs) in molti settori dell'IoT.

Le caratteristiche principali delle LPWANs sono:

- ampia copertura (10-15 km per aree rurali, e 2-5 km in aree urbane)
- basso data rate (valori medi compresi tra 100 bps a 100 kbps)
- trasmissione di pacchetti di piccole dimensioni (10 – 1000 bytes)
- license-free frequency bands (2.4 GHz, 868 MHz, 915 MHz, 433 MHz)
- ottimizzazione dei consumi energetici

Name of Standard	Weightless			SigFox	LoRaWAN	LTE-Cat M	IEEE P802.11ah (low power WiFi)	Dash7 Alliance Protocol 1.0	Ingenu RPMA	nWave
	-W	-N	-P							
Frequency Band	TV whitespace (400-800 MHz)	Sub-GHz ISM	Sub-GHz ISM	868 MHz/902 MHz ISM	433/868/780/915 MHz ISM	Cellular	License-exempt bands below 1 GHz, excluding the TV White Spaces	433, 868, 915 MHz ISM/SDR	2.4 GHz ISM	Sub-GHz ISM
Channel Width	5MHz	Ultra narrow band (200Hz)	12.5 kHz	Ultra narrow band	EU: 8x125kHz, US 64x125kHz/8x125kHz, Modulation: Chirp Spread Spectrum	1.4MHz	1/2/4/8/16 MHz	25 KHz or 200 KHz	1 MHz (40 channels available)	Ultra narrow band
Range	5km (urban)	3km (urban)	2km (urban)	30-50km (rural), 3-10km (urban), 1000km LoS	2-5k (urban), 15k (rural)	2.5- 5km	Up to 1Km (outdoor)	0 – 5 km	>500 km LoS	10km (urban), 20-30km (rural)
End Node Transmit Power	17 dBm	17 dBm	17 dBm	10µW to 100 mW	EU:<+14dBm, US:<+27dBm	100 mW	Dependent on Regional Regulations (from 1 mW to 1 W)	Depending on FCC/ETSI regulations	to 20 dBm	25-100 mW
Packet Size	10 byte min.	Up to 20 bytes	10 byte min.	12 bytes	Defined by User	~100 ~1000 bytes typical	Up to 7,991 Bytes (w/o Aggregation), up to 65,535 Bytes (with Aggregation)	256 bytes max / packet	Flexible (6 bytes to 10 kbytes)	12 byte header, 2-20 byte payload
Uplink Data Rate	1 kbps to 10 Mbps	100bps	200 bps to 100 kbps	100 bps to 140 messages/day	EU: 300 bps to 50 kbps, US:900-100kbps	~200kbps	150 Kbps ~ 346.666 Mbps	9.6 kb/s, 55.55 kbps or 166.667 kb/s	AP aggregates to 624 kbps per Sector (Assumes 8 channel Access Point)	100 bps
Downlink Data Rate	1 kbps to 10 Mbps	No downlink	200 bps to 100 kbps	Max 4 messages of 8 bytes/day	EU: 300 bps to 50 kbps, US:900-100kbps	~200kbps	150 Kbps ~ 346.666 Mbps	9.6 kb/s, 55.55 kbps or 166.667 kb/s	AP aggregates to 156 kbps per Sector (Assumes 8 channel Access Point)	–
Devices per Access Point	Unlimited	Unlimited	Unlimited	1M	Uplink:>1M, Downlink:<100k	20k+	8191	NA (connectionless communication)	Up to 384,000 per sector	1M
Topology	Star	Star	Star	Star	Star on Star	Star	Star, Tree	Node-to-node, Star, Tree	Typically Star, Tree supported with an RPMA extender	Star
End node roaming allowed	Yes	Yes	Yes	Yes	Yes	Yes	Allowed by other IEEE 802.11 amendments (e.g., IEEE 802.11r)	Yes	Yes	Yes
Governing Body	<a href="#">Weightless SIG</a>			Sigfox	<a href="#">LoRa Alliance</a>	<a href="#">3GPP</a>	IEEE 802.11 working group	<a href="#">Dash7 Alliance</a>	<a href="#">Ingenu (formerly OnRamp)</a>	<a href="#">Weightless SIG</a>
Status	Limited deployment awaiting spectrum availability	Deployment beginning	Standard in development. Scheduled release 4Q 2015	In deployment	Spec released June 2015, in deployment	Release 13 expected 2016	Targeting 2016 release	Released May 2015	In Deployment	In Deployment

La tecnologia LoRa (Long Range) appartiene alla classe LPWANs ed è stata progettata per essere integrata in numerose applicazioni e settori: monitoraggio di parametri ambientali, smart agriculture, asset tracking, smart parking, monitoraggio dell'inquinamento e dei consumi e in ambito healthcare.

## Lo stack LoRa si suddivide in due livelli:

- **LoRa PHY:**

rappresenta il livello fisico del protocollo LoRa e consiste in una tecnica di modulazione a spettro espanso che offre un'ampia copertura.

Il protocollo LoRa PHY è proprietario ed è stato inizialmente sviluppato da Cycléo nel 2010 la quale è stata in seguito acquisita da Semtech.

### Caratteristiche principali

- **Modulazione a spettro espanso basata su un segnale Chirp (Chirp Spread Spectrum - CSS)** che integra anche un **meccanismo di correzione automatica degli errori** chiamato **Forward Error Correction (FEC)**
- LoRa migliora significativamente la sensibility del ricevitore e utilizza l'intera larghezza di banda del canale per trasmettere un segnale, rendendolo robusto al rumore del canale e insensibile agli offset di frequenza causati dall'impiego di oscillatori low-cost.
- I parametri principali come la banda di frequenze utilizzata, la potenza trasmessa, l'ampiezza di banda dei singoli canali e il numero dei canali dipendono dall'area geografica di appartenenza:
  - **Uplink and Downlink Data Rate:**  
EU: 250 bps to 50 kbps; US: 980 bps to 21.9 kbps
  - **Node transmit power:**  
EU: <= +14 dBm (PA\_BOOST mode: 20 dBm); US: <= +27 dBm
  - **Frequency Bands:**  
sub-1GHz ISM bands subdivided in this way:
    - Europe: 868 MHz (863-870 MHz) and 433 MHz
    - North America: 915 MHz (902 - 928 MHz)
    - Asia: 470-510 MHz and 779-787 MHz[List of frequency plan and regulations by country \(TTN website\)](#)
  - **Bandwidth:**  
the most used bandwidths are 500 kHz, 250 kHz and 125 kHz
  - **Channels:**
    - EU863-870: 8 channels for uplink and downlink  
1 channel with FSK modulation.
    - US902-928: 9 channels for uplink (903.9 - 904.6 MHz)  
9 channels for downlink (923.3 - 927.5 MHz)[Complete list of LoRa and LoRaWAN frequencies and channels](#)

- **Payload size:**  
maximum size between 51 bytes and 222 bytes, depending on the Spreading Factor (SF).
- **Throughput, data rate e range:**  
Throughput, data rate e range dipendono da 3 parametri importanti:
  1. **Bandwidth** (larghezza di banda - BW).
  2. **Coding Rate** (CR): è il rapporto (minore di 1) fra i bit del messaggio (cioè l'informazione) e la lunghezza totale della parola di codice. Il coding rate è un parametro associato alla codifica del segnale da trasmettere che viene utilizzata per la rilevazione e correzione degli errori (Forward Error Correction).
  3. **Spreading Factor** (SF): è il rapporto tra la frequenza del segnale di chip (utilizzato nella modulazione a spettro espanso) e la frequenza di simbolo: SF = chip\_rate / symbol\_rate.

**All'aumentare della larghezza di banda aumenta il data rate e quindi si riduce il tempo di trasmissione ma d'altra parte viene ridotto il valore di sensitivity.**

**All'aumentare dello Spreading Factor (SF) diminuisce il data rate ma aumenta la robustezza del segnale garantendo una copertura (range) maggiore.**

**Per calcolare il valore di sensitivity** ovvero l'ampiezza minima che può essere ricevuta dal ricevitore in modo che sia possibile elaborare il segnale ricevuto fissato un livello di SNR (Signal to Noise Ratio), **si può utilizzare la seguente formula:**

$$S = -174 + 10 \cdot \log_{10}(BW) + NF + SNR$$

**Il data rate può essere calcolato attraverso la seguente formula:**

LoRa Data Rate (Rb) Formula : -

$$R_b = SF * \frac{\left[ \frac{4}{4+CR} \right]}{\left[ \frac{2^SF}{BW} \right]} * 1000$$

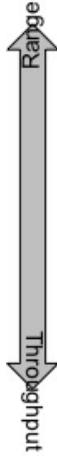
SF = Spreading Factor (6,7,8,9,10,11,12)

CR = Code Rate (1,2,3,4)

BW = Bandwidth in KHz  
(10.4,15.6,20.8,31.25,41.7,62.5,125,250,500)

Rb = Data rate or Bit Rate in bps

## Esistono delle combinazioni predefinite dei 3 parametri descritti in precedenza detti LoRa Modes:



LoRa mode	BW	CR	SF	time on air in second for payload size of					
				5 bytes	55 bytes	105 bytes	155 Bytes	205 Bytes	255 Bytes
1	125	4/5	12	0.95846	2.59686	4.23526	5.87366	7.51206	9.15046
2	250	4/5	12	0.47923	1.21651	1.87187	2.52723	3.26451	3.91987
3	125	4/5	10	0.28058	0.69018	1.09978	1.50938	1.91898	2.32858
4	500	4/5	12	0.23962	0.60826	0.93594	1.26362	1.63226	1.95994
5	250	4/5	10	0.14029	0.34509	0.54989	0.75469	0.95949	1.16429
6	500	4/5	11	0.11981	0.30413	0.50893	0.69325	0.87757	1.06189
7	250	4/5	9	0.07014	0.18278	0.29542	0.40806	0.5207	0.63334
8	500	4/5	9	0.03507	0.09139	0.14771	0.20403	0.26035	0.31667
9	500	4/5	8	0.01754	0.05082	0.08154	0.11482	0.14554	0.17882
10	500	4/5	7	0.00877	0.02797	0.04589	0.06381	0.08301	0.10093

- **LoRaWAN:**

LoRaWAN rappresenta il livello MAC e quello di rete dello standard LoRa. L'architettura di una rete LoRaWAN prevede una **topologia a stella di stelle** ed è costituita da **tre elementi fondamentali**:

- (a) **End-devices** (LoRaWAN nodes)
- (b) **Gateway**
- (c) **Network Server**

Ogni nodo è connesso ad uno o più gateway attraverso un collegamento single-hop secondo lo standard LoRa.

I gateways si connettono a turno ad un Network Server (NetServer) comune attraverso un backhaul channel utilizzando lo stack TCP/IP ed hanno il compito di ritrasmettere i messaggi ricevuti (aggiungendo informazioni riguardanti la qualità del segnale ricevuto (RSSI) o il numero di messaggi ricevuti) implementando la comunicazione tra end-devices e Network Server.

Un nodo può essere associato a più di un gateway ma può essere associato ad un unico Network Server. Questo implica che diversi gateways possono ricevere uno stesso messaggio inviato da un nodo e ritrasmetterlo ad uno stesso Network Server il quale deve occuparsi di filtrare i messaggi duplicati.

Le comunicazioni sono nella maggior parte dei casi bidirezionali fornendo trasmissioni in uplink e downlink, anche se la comunicazione in uplink è predominante rispetto a quella in downlink.

L'infrastruttura di una rete LoRaWAN ha la capacità di adattare il data rate e le frequenze utilizzate, attraverso un meccanismo chiamato **Adaptive Data Rate** (ADR) che permette di ottimizzare la durata delle batterie e il data rate a seconda delle condizioni del canale.

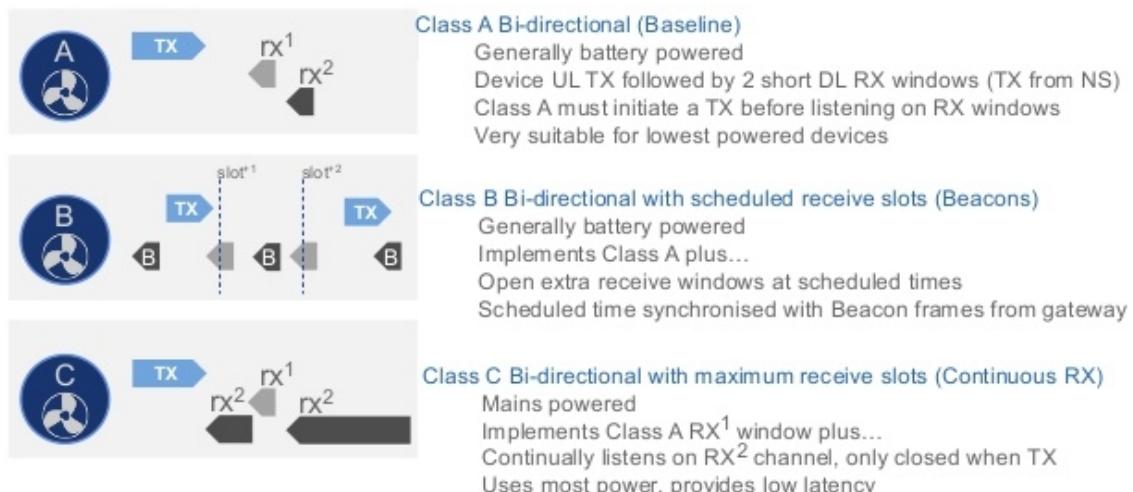
La scelta del canale da parte di un end-device e del rispettivo data rate per la trasmissione deve soddisfare i seguenti requisiti:

- Gli end-devices cambiano canale in modo pseudo-random per ogni trasmissione in modo da garantire una maggiore robustezza del segnale alle interferenze.
- Devono essere rispettati il massimo duty-cycle e durata della trasmissione relativi alla sub-band in base alla regolamentazione locale consentita.  
Le regolamentazioni (ETSI EN300-220-1 e ERC/REC 70-03) che riguardano **le trasmissioni nella banda 863-870 MHz**, definiscono un **duty cycle pari a 0.1 % ovvero 3.6 s**; tuttavia **se si sceglie un canale compreso nell'intervallo 865-868 MHz** allora si può utilizzare un **duty cycle pari a 1 % che corrisponde a circa 36 s**.

Il protocollo LoRaWAN prevede 3 classi di end-devices:

- **Class A: Bi-directional end-devices**  
Possono implementare comunicazioni bidirezionali in cui dopo ogni messaggio trasmesso in uplink, vengono aperte due finestre (RX1 e RX2) in cui il dispositivo è in ascolto e può ricevere messaggi in downlink. La seconda finestra (RX2) può essere aperta su un canale differente rispetto a quello utilizzato nella prima finestra (RX1).
- **Class B: Bi-directional end-device with scheduled receive slots**  
Consentono la comunicazione bidirezionale utilizzando oltre alle finestre RX1 e RX2, utilizzate per classe A, anche un ulteriore finestra di ricezione che viene attivata dal Network Server attraverso l'invio di un Beacon di sincronizzazione. In questo modo il Network Server sa quando un end-device è in ascolto.
- **Class C: Bi-directional end-devices with maximum receive slot**  
Prevedono una finestra di ricezione costantemente aperta tranne durante la trasmissione ovvero un end-device di classe C è sempre in modalità di ricezione tranne quando deve trasmettere.

## LoRaWAN Device Classes



Per poter entrare a far parte di una rete LoRaWAN, un end-device deve essere attivato.

**L'attivazione di un end-device può essere effettuata in due modi distinti:**

- 1) **Over-The-Air Activation (OTAA).**
- 2) **Activation By Personalization (ABP).**

Dopo aver eseguito l'attivazione, ogni end-device possiede:

- un indirizzo IEEE EUI32 identificativo (**DevAddr**).
- un identificativo globale per l'applicazione (**AppEUI**) associata all'end-device.
- una chiave di sessione di rete (**NwkSKEY**) usata per garantire la sicurezza a livello di rete e per garantire l'integrità dei messaggi verificando il MIC (Message Integrity Code).
- una chiave di sessione per l'applicazione (**AppSKey**) specifica per ogni end-device che viene utilizzata per cifrare il payload dei messaggi inviati.

### **Over-The-Air Activation**

Per effettuare la OTAA, un end-device esegue la **procedura di join** attraverso i messaggi di **join request** e **join accept** per autenticarsi con la rete e per poter iniziare ad inviare dati al Network Server.

La procedura di join richiede che un end-device venga configurato definendo:

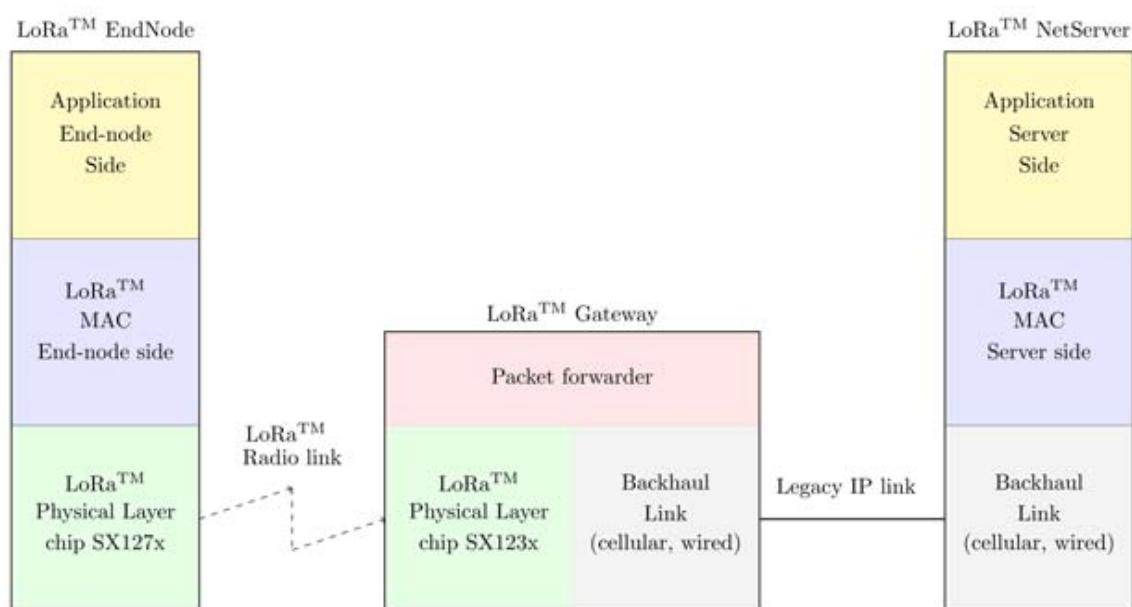
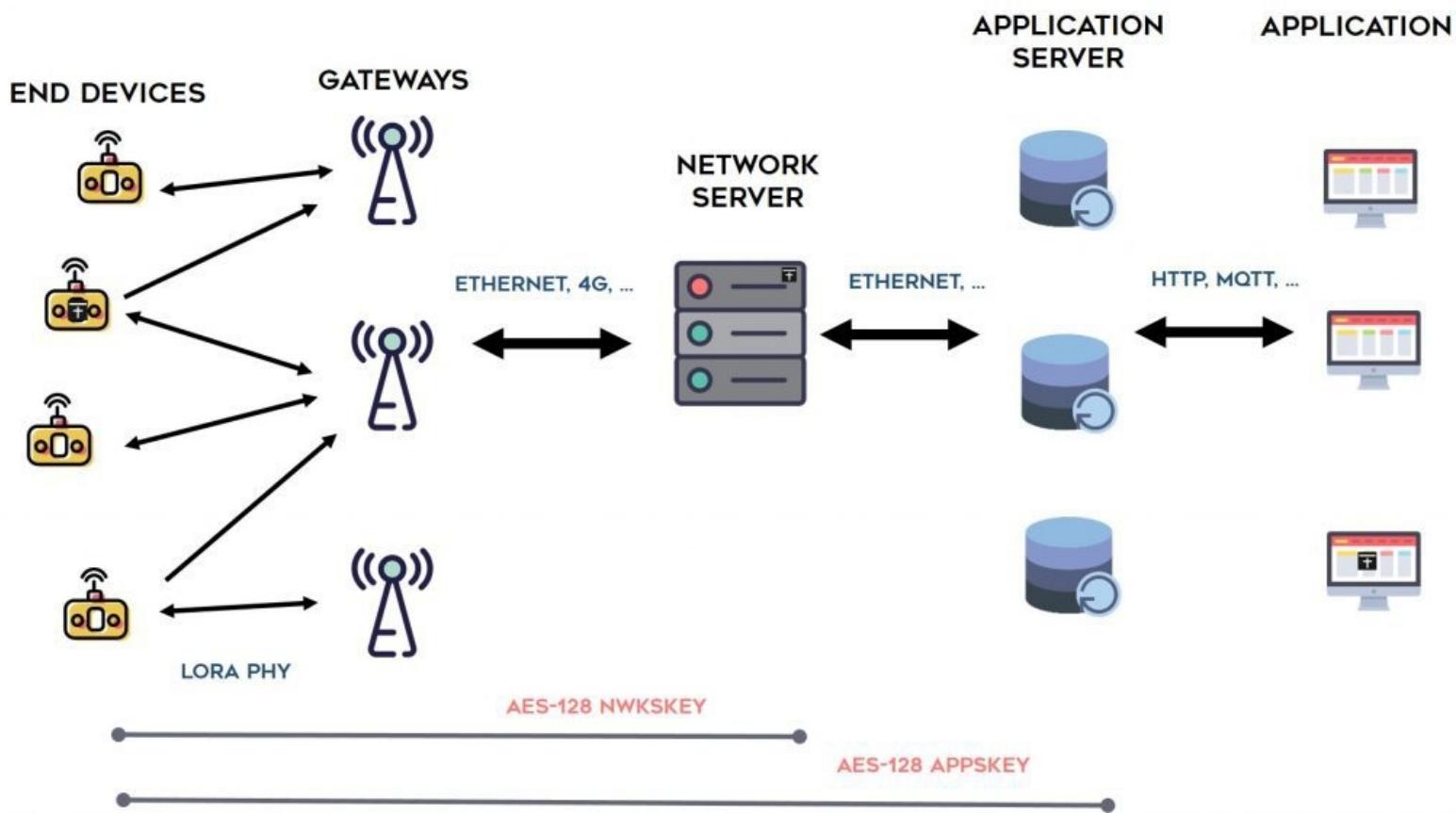
- **DevEUI**: identificativo globale univoco del nodo (IEEE EUI64)
- **AppEUI**: identificativo applicazione
- **AppKey**: chiave AES di 128 bit

Ad ogni procedura di join **l'AppKey viene utilizzata per generare le chiavi NwkSKey ed AppSKey** specifiche per il nodo, utilizzate per criptare/decriptare i messaggi che lo interessano.

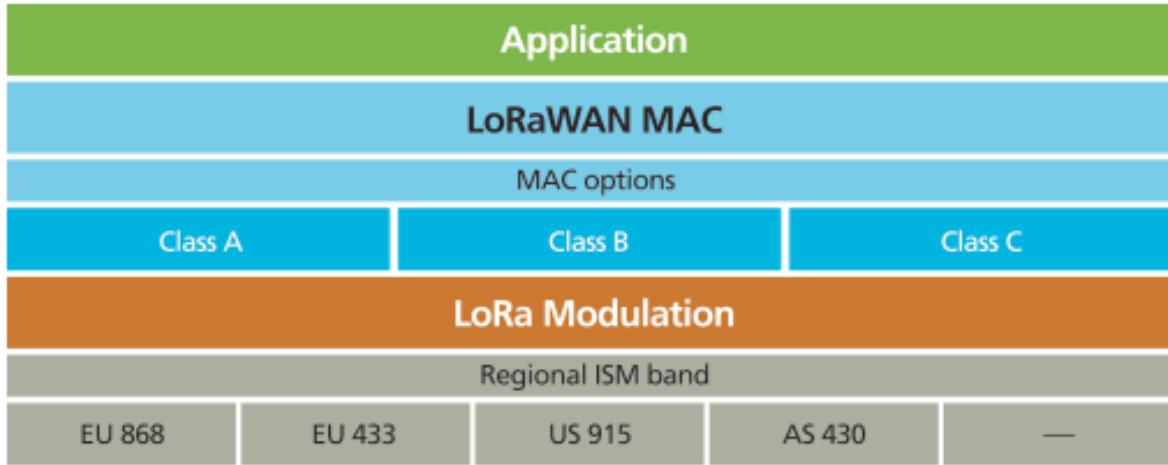
### **Activation By Personalization**

Per l'attivazione del nodo **non viene eseguita la procedura di join** ma **il DevAddr e le due chiavi di sessione NwkSKey e AppSKey sono memorizzate direttamente nel nodo** invece dei parametri DevEUI, AppEUI e AppKey.

# Architettura di una rete LoRaWAN



## LoRaWAN protocol stack



## LoRaWAN Frame Format

PHYPayload:	MHDR : 8	MACPayload	MIC : 32
MACPayload:	FHDR : 56..176	FPort : 8	FRMPayload (encrypted)
FHDR:	DevAddr : 32	FCtrl : 8	FCnt : 16
MHDR:	MType : 3	RFU : 3	Major : 2
FCtrl:	Uplink: ADR : 1   ADRAckReq : 1   ACK : 1 Downlink: ADR : 1   ADRAckReq : 1   ACK : 1   RFU : 1		
FOpts:	MACCommand_1 : 8..40	...	MACCommand_n : 8..40
MACCommand:	CID : 8	Args : 0..32	

# IL PROTOCOLLO MQTT

## - Introduzione a MQTT e specifiche generali

MQTT è un protocollo applicativo per la messaggistica basato sul paradigma pubblicazione/sottoscrizione (publish/subscribe).

È leggero, open, semplice e progettato in modo da essere facile da implementare. Queste caratteristiche lo rendono ideale per molte applicazioni e soprattutto quelle riguardanti il mondo dell'Internet of Things (IoT) e le comunicazioni di tipo Machine to Machine (M2M).

Le caratteristiche principali di MQTT sono:

- Semplice da implementare
- Fornisce il servizio di Quality of Service (QoS) dei dati trasportati
- Consente di minimizzare il consumo della banda ed è un protocollo lightweight
- Data Agnostic ovvero consente il trasferimento di dati di qualsiasi tipo
- Continuous Session Awareness
- Versione attuale: 3.1.1

## - Il modello PUBLISH / SUBSCRIBE

MQTT è basato sul modello publish/subscribe. Esso è un alternativa al modello client/server in cui i client inviano delle richieste ad un server e il server risponde ai client inviando loro delle risposte, quindi nel modello client-server quindi il client deve conoscere l'indirizzo IP e la porta su cui il server è in ascolto.

Il modello publish/subscribe, invece, è caratterizzato dalla presenza di clients che non sono direttamente collegati tra loro ma essi inviano i messaggi ad un entità chiamata broker, il quale si occupa di filtrare i messaggi ricevuti ed inviarli ai clients destinatari.

Un client è rappresentato da un generico dispositivo che:

- implementa uno stack TCP/IP
- possiede una libreria MQTT
- è collegato ad un broker MQTT attraverso una rete di telecomunicazione

Un broker è rappresentato da un'entità software che:

- deve gestire un numero elevato di connessioni con i clients
- deve occuparsi di filtrare i messaggi ricevuti in base al topic e inoltrarli ai subscribers
- deve essere in grado di gestire il servizio di QoS
- deve gestire l'autenticazione dei clients

I clients che inviano dei messaggi sono detti publishers.

I clients che ricevono i messaggi, ovvero quelli che si registrano (subscribe) con il broker come destinatari di alcuni topic, sono detti subscribers.

Un client può rivestire il ruolo sia di publisher, sia di subscriber.

Nel modello publish/subscribe e quindi anche in MQTT che si basa su tale paradigma, si parla di un triplice disaccoppiamento (decoupling):

### **1) Disaccoppiamento spaziale:**

publisher e subscriber non devono essere a conoscenza l'uno dell'altro (in termini ad esempio di indirizzo IP e porta) ma devono conoscere solo hostanme/IP e la porta del broker.

### **2) Disaccoppiamento temporale:**

publisher e subscriber non devono essere in esecuzione nello stesso momento poiché non viene stabilita una comunicazione diretta tra publisher e subscriber.

### **3) Disaccoppiamento di sincronizzazione:**

la comunicazione tra publisher e subscriber avviene in modo asincrono e spesso essa è implementata attraverso delle callback che consentono ai client di non rimanere in attesa di ricevere un messaggio in modo da poter eseguire altri tasks.

## **- Topics e filtraggio dei messaggi**

Il filtraggio dei messaggi da parte del broker MQTT è di tipo subject-based e per questo motivo qualunque messaggio trasmesso è associato ad un **topic** che specifica l'argomento associato al messaggio.

Un publisher invia un messaggio specificando il topic a cui è associato il messaggio.

Un subscriber effettua una subscription ad uno o più topic: in questo modo il broker MQTT inoltrerà al subscriber solo i messaggi che contengono il topic a cui il subscriber è associato.

Un topic è rappresentato da una stringa di caratteri con codifica UTF-8 ed è costituito da uno o più **"topic levels"**.

Ciascun topic level è separato dal precedente e dal successivo dal carattere '/'.

Un topic level può contenere anche spazi ma in generale è conveniente utilizzare delle stringhe senza spazi.

Ciascun topic deve essere costituito da almeno un carattere.

I nomi dei topic sono case-sensitive.

Struttura di un topic:



Esempio di un topic:

network/device1/sensors/temperature

network/device2/sensors/accelerometer/x-axis

Un client può effettuare la subscription a più di un topics utilizzando dei **wildcards**. I wildcards non possono essere utilizzati quando un client effettua l'invio (publish) di un messaggio.

Esistono 2 tipi di wildcards:

- **Single level wildcard: +**

Esso rappresenta una generica stringa infatti è utilizzato per indicare un generico topic level.



Per esempio, una subscription al seguente topic  
myhome/groundfloor/+/temperature  
corrisponde anche ai seguenti topics:

- ✓ myhome / groundfloor / livingroom / temperature
- ✓ myhome / groundfloor / kitchen / temperature
- ✗ myhome / groundfloor / kitchen / brightness
- ✗ myhome / firstfloor / kitchen / temperature
- ✗ myhome / groundfloor / kitchen / fridge / temperature

- **Multi level wildcard: #**

Esso è utilizzato per indicare un numero arbitrario di topic levels.

Un multi level wildcard può essere inserito solo come ultimo carattere del topic ed è preceduto sempre dal carattere '/'

Esempio:



- ✓ myhome / groundfloor / livingroom / temperature
- ✓ myhome / groundfloor / kitchen / temperature
- ✓ myhome / groundfloor / kitchen / brightness
- ✗ myhome / firstfloor / kitchen / temperature

I topics che iniziano con il carattere '\$' sono considerati topics speciali e sono associati a statistiche relative al broker MQTT.

Alcuni esempi di topics speciali sono quelli che iniziano per "\$SYS" come ad esempio:

\$SYS/broker/clients/connected  
\$SYS/broker/clients/disconnected  
\$SYS/broker/clients/total  
\$SYS/broker/messages/sent  
\$SYS/broker/uptime

Non è possibile effettuare l'invio di messaggi (publish) ai topics speciali.

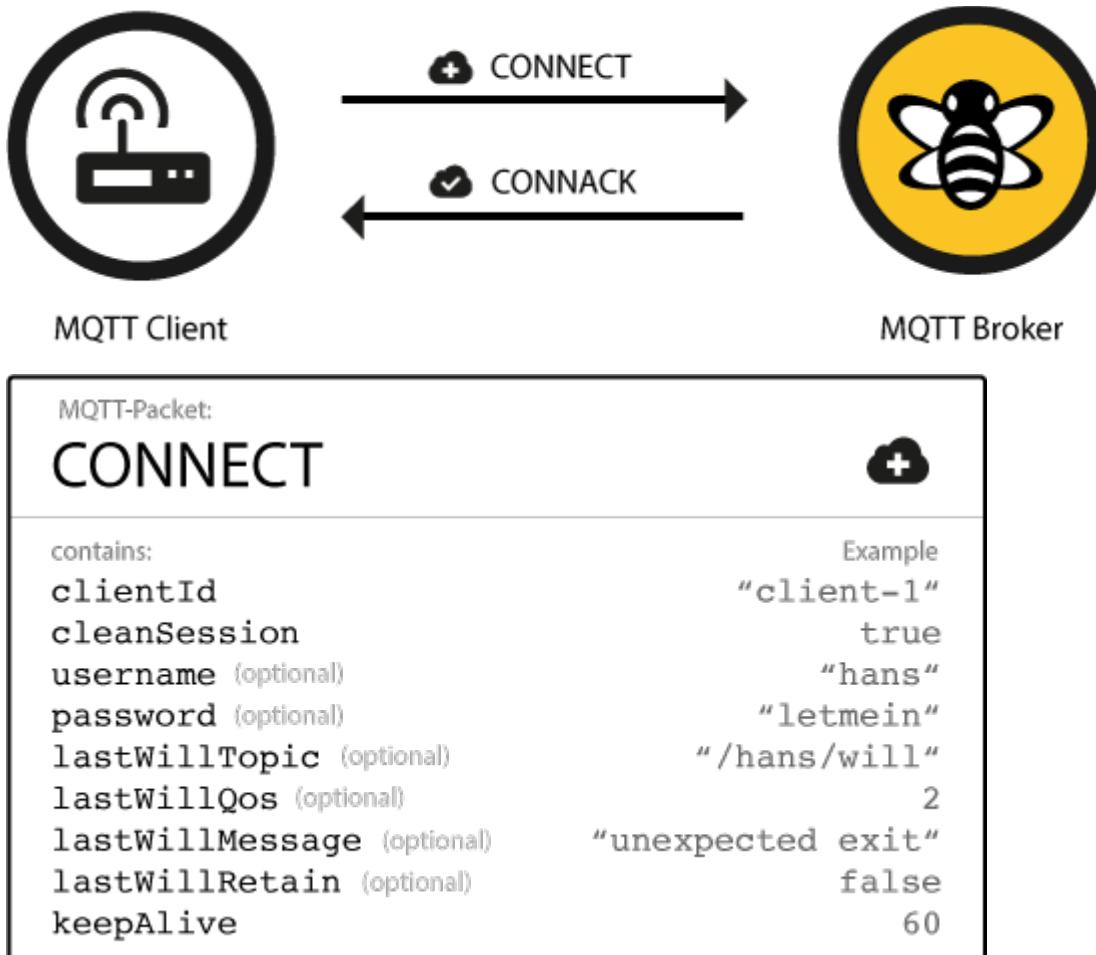
## - CONNESSIONE MQTT tra Client e Broker: CONNECT eCONNACK

Il protocollo MQTT è basato sullo stack TCP/IP il quale deve essere presente sia sui clients, sia sul broker.

La comunicazione tra client e broker è inizializzata quando il client invia un messaggio di **CONNECT** al broker.

Il broker risponde inviando un messaggio di **CONNACK** contenente un codice di stato che rappresenta lo stato della connessione.

Una volta che la connessione è inizializzata, il broker la manterrà aperta fino a quando il client non invia il messaggio di **DISCONNECT** oppure fino a quando la connessione non viene persa.



- **clientId**: identificatore del client.
- **cleanSession**: flag utilizzato per inizializzare una connessione persistente. Se cleanSession è impostato su false il broker memorizzerà tutte le subscriptions effettuate dal client.
- **username** e **password**: utilizzate per l'autenticazione del client.
- **will topic** e **will message**: utilizzati quando il client si disconnette in modo anomalo. In questo caso il messaggio e il topic di will sono inviati dal broker MQTT ai subscribers.
- **keep alive**: intervallo di tempo dopo il quale il client si impegna a inviare un messaggio di PING Request al broker che risponderà con un PING Response in modo da mantenere attiva la connessione.

MQTT-Packet:

## CONNACK



contains:

**sessionPresent**  
**returnCode**

Example

true

0

- **sessionPresent**: indica che esiste già una sessione attiva tra il client e il broker. Se il flag cleanSession è impostato su true allora il flag sessionPresent sarà impostato su false.
- **returnCode**: codice di stato che rappresenta lo stato della connessione:
  - 0 => Connection Accepted
  - 1 => Connection Refused, unacceptable protocol version
  - 2 => Connection Refused, identifier rejected
  - 3 => Connection Refused, server unavailable
  - 4 => Connection Refused, bad username or password
  - 5 => Connection Refused, not authorized

MQTT-Packet:

## DISCONNECT



no payload

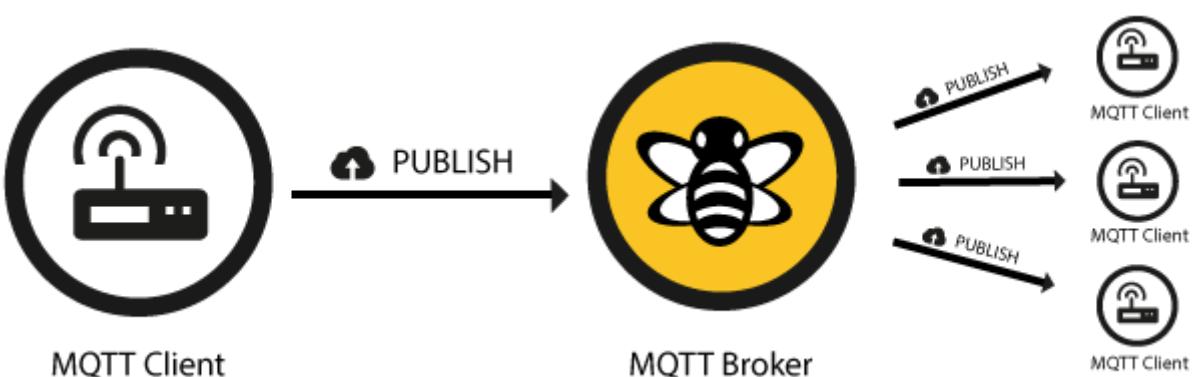
## - PUBLISH

Un client, dopo essersi connesso al broker MQTT, può effettuare l'invio (publish) dei messaggi al broker specificando il topic che verrà utilizzato dal broker per inoltrare il messaggio ai subscribers di tale topic.

Il messaggio di **PUBLISH** contiene anche il payload che rappresenta i dati da trasmettere. Il dati contenuti nel payload possono essere di qualsiasi tipo/formato (bytes, testo, JSON, ...).

MQTT-Packet:	
<b>PUBLISH</b>	
contains:	Example
<b>packetId</b> (always 0 for qos 0)	4314
<b>topicName</b>	"topic/1"
<b>qos</b>	1
<b>retainFlag</b>	false
<b>payload</b>	"temperature:32.5"
<b>dupFlag</b>	false

- **packetId**: identificatore del messaggio.  
Tale valore è diverso da zero solo se QoS > 0.
- **topicName**: nome del topic associato al messaggio.
- **qos**: valore del parametro Quality of Service (QoS).
- **retainFlag**: questo flag determina se il messaggio sarà salvato dal broker come "last known good value".
- **payload**: contiene i dati da trasmettere ai subscribers.
- **dupFlag**: indica che il messaggio è un duplicato di messaggi che sono stati inviati in precedenza dal client. È utilizzato quando il client non riceve il PUBACK relativo ad un messaggio inviato in precedenza dal client. Flag valido solo per messaggi con QoS > 0.



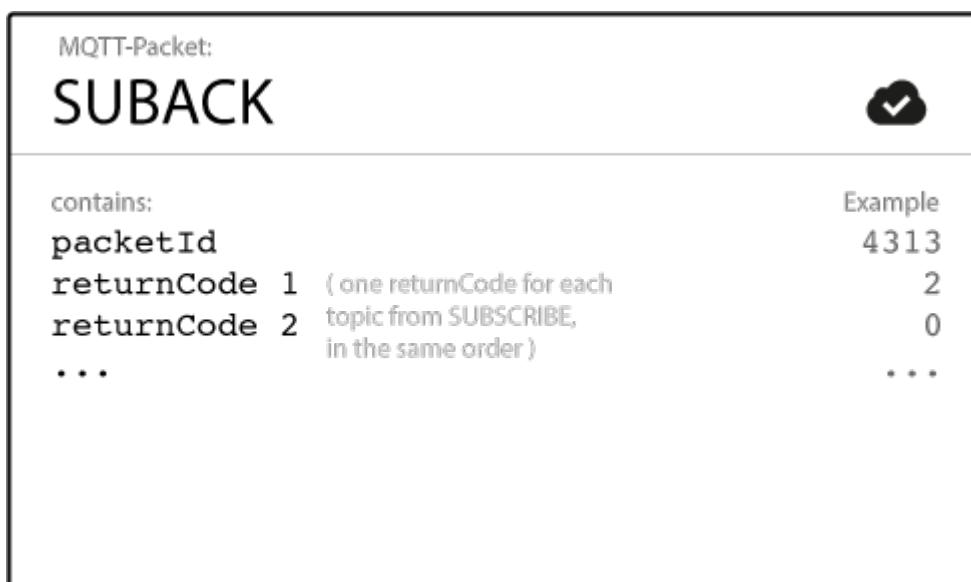
## - SUBSCRIBE

Quando un client vuole connettersi al broker MQTT come subscriber per uno o più topics invia un messaggio di **SUBSCRIBE**.

Ciascuna subscription viene confermata attraverso un messaggio di **SUBACK**.

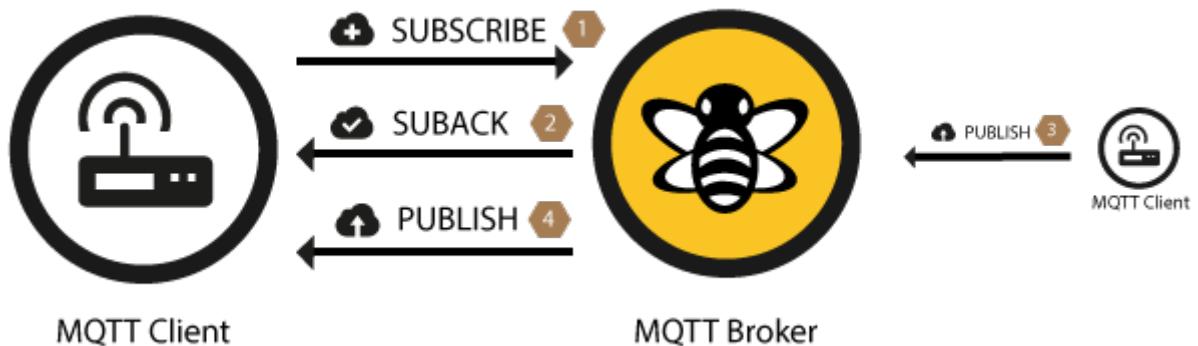


- **packetId**: identificatore univoco del messaggio.
- **quos(i)**: QoS associato al topic i-esimo.
- **topic(i)**: nome del topic i-esimo per cui il client effettua la subscription. Possono essere specificate diverse coppie (qos, topic) per ciascuno dei topics a cui il client effettua la subscription.



- **packetId**: identificatore del pacchetto. È lo stesso valore inserito nel campo packetId del messaggio di SUBSCRIBE.
- **returnCode list**: viene inserito un returnCode per ogni topic inserito nel messaggio di SUBSCRIBE (nello stesso ordine dei topic). Ciascun returnCode rappresenta il valore di QoS garantito dal broker e può assumere i seguenti valori:

- 0 => Success – Maximum QoS 0
- 1 => Success – Maximum QoS 1
- 2 => Success – Maximum QoS 2
- 128 => Failure

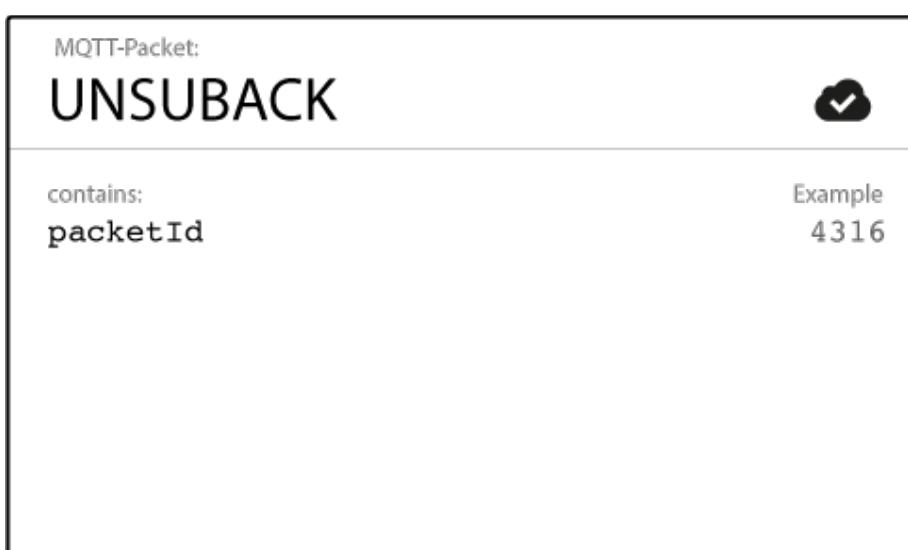
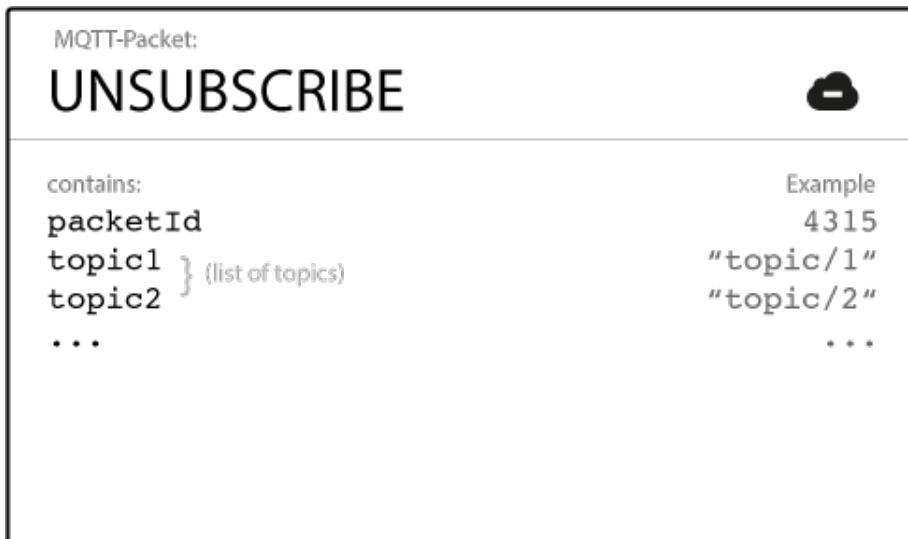


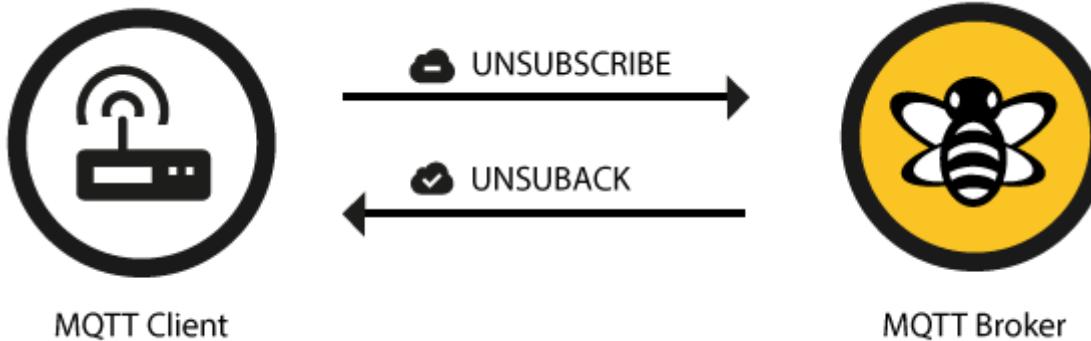
### - UNSUBSCRIBE

Un client può effettuare la “unsubscription” che consente di eliminare le subscription esistenti effettuate dal client.

Una unsubscription viene effettuata dal client attraverso il messaggio di **UNSUBSCRIBE**.

Il broker invierà l'acknowledgement attraverso il messaggio di **UNSUBACK**.





### - **QUALITY OF SERVICE (QoS)**

Il termine Quality of Service è utilizzato per indicare l'affidabilità e le garanzie di invio di un messaggio.

**In MQTT ci sono 3 livelli di QoS:**

- 0: "Al massimo una volta"
- 1: "Almeno una volta"
- 2: "Esattamente una volta"

È inoltre importante capire che l'invio di un messaggio è costituito da 2 fasi:

1) Il publisher invia il messaggio al broker: PUBLISHER → BROKER

Per l'invio del messaggio dal publisher al broker viene utilizzato il livello di QoS specificato dal client nel messaggio di PUBLISH.

2) Il broker invia il messaggio al subscriber: BROKER → SUBSCRIBER

Per l'invio del messaggio dal broker al subscriber viene utilizzato il livello di QoS specificato dal subscriber nel messaggio di SUBSCRIBE.

Questo significa che se QoS subscribe message < QoS publish message, le garanzie di invio del messaggio dal broker al subscriber possono diminuire.

QoS è una caratteristica importante di MQTT, poiché rende molto più semplice la comunicazione in reti inaffidabili infatti il protocollo MQTT gestisce la ritrasmissione e garantisce la consegna del messaggio, indipendentemente da quanto sia inaffidabile il livello trasporto sottostante.

Inoltre, consente al client di scegliere il livello di QoS in base all'affidabilità della rete e alla logica dell'applicazione.

## **QoS 0 - at most once: Best effort delivery**

Il destinatario non invia l'acknowledgement (ACK) relativo alla ricezione del messaggio ed il broker non memorizza il messaggio per poi ritrasmetterlo. Stesse garanzie del protocollo TCP.

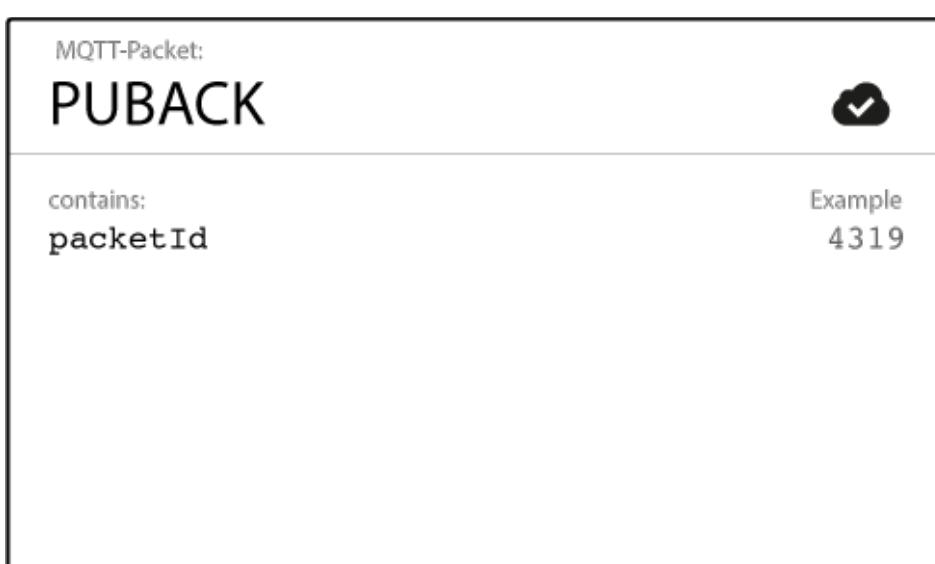
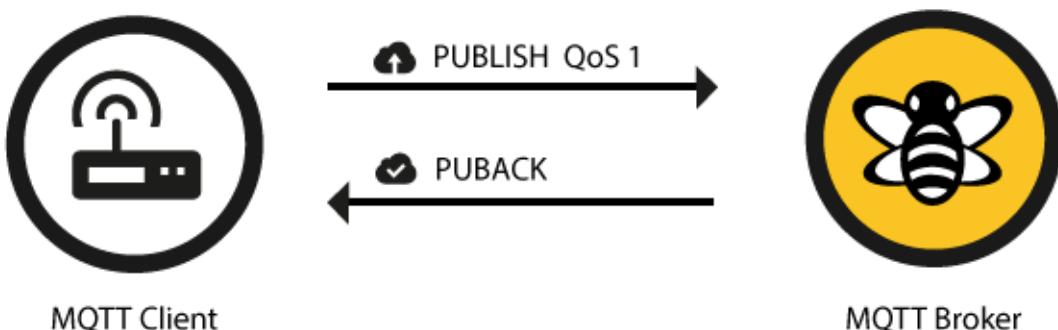


## **QoS 1 - at least once**

Quando si utilizza il livello 1 di QoS vi è garanzia che il messaggio sarà inviato almeno una volta.

Un messaggio con QoS 1 viene memorizzato dal client fino a quando esso non riceve il messaggio di PUBACK da parte del broker.

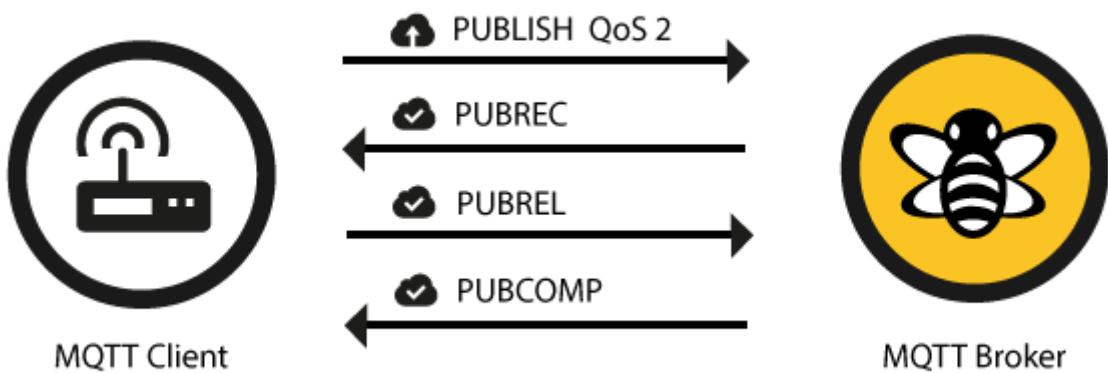
Se il messaggio di PUBACK non viene ricevuto dopo un certo intervallo di tempo, il client invia di nuovo il messaggio di PUBLISH impostando dupFlag su true.



## QoS 2 - exactly once

Il livello 2 di QoS garantisce che il messaggio sia ricevuto una sola volta dal destinatario attraverso i seguenti messaggi:

- **PUBREC**: messaggio di acknowledgement inviato dal destinatario (broker) quando riceve un messaggio di **PUBLISH con QoS 2**. Il destinatario memorizza un riferimento del packetId del messaggio di PUBLISH fino a quando non avrà inviato il messaggio di PUBCOMP in modo da riconoscere gli eventuali messaggi duplicati.
- **PUBREL**: messaggio di acknowledgement inviato dal mittente al destinatario dopo che il mittente ha ricevuto il messaggio di PUBREC che notifica l'avvenuta ricezione del messaggio di PUBLISH da parte del destinatario.
- **PUBCOMP**: messaggio finale di acknowledgement inviato dal destinatario al mittente.



MQTT-Packet:  
**PUBREC**

contains:  
**packetId**



Example  
4320

MQTT-Packet:  
**PUBREL**

contains:  
**packetId**



Example  
4320

MQTT-Packet:  
**PUBCOMP**

contains:  
**packetId**



Example  
4320

# FENOMENI SISMICI

## DEFINIZIONE E DESCRIZIONE DI UN FENOMENO SISMICO

I terremoti, o sismi, consistono in una serie di rapide oscillazioni del terreno causate da una brusca liberazione di energia elastica da una zona del sottosuolo definito come **ipocentro**.

L'ipocentro può essere situato a profondità comprese tra poche decine di metri fino ad alcune centinaia di chilometri e a partire da esso si propagano in tutte le direzioni le onde sismiche.

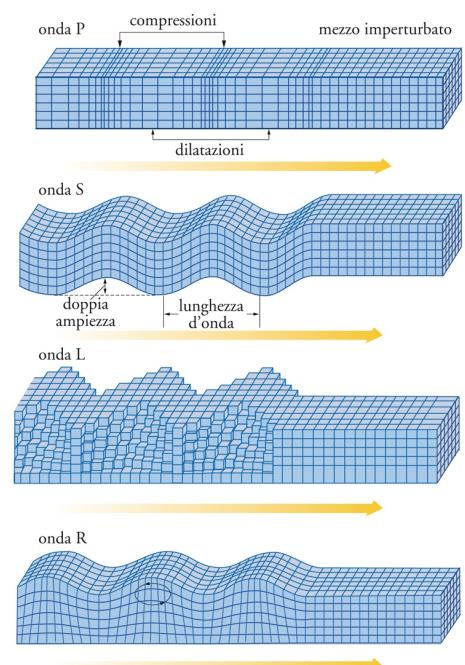
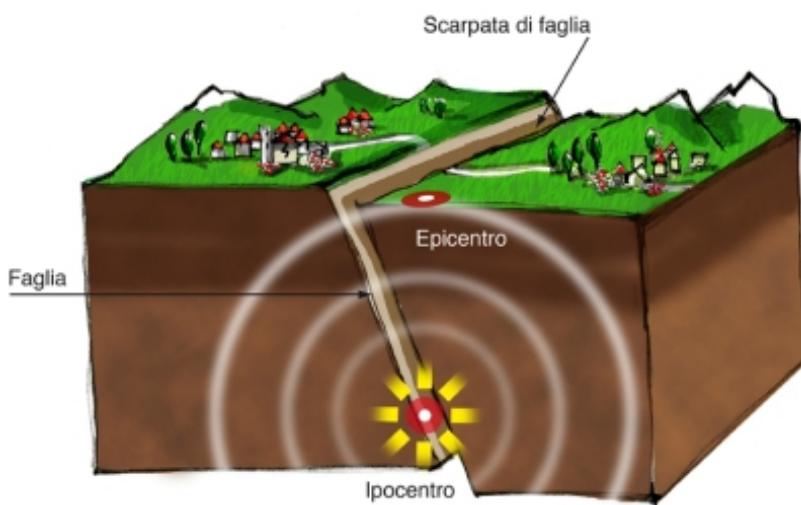
Il punto della superficie situato sulla verticale dell'ipocentro viene chiamato **epicentro**.

I movimenti del suolo sono spesso descritti come:

- ondulatori: se sono provocati da onde sismiche a bassa frequenza
- sussultori: se sono provocati da onde sismiche ad alta frequenza

Le onde sismiche possono essere di **3 tipi**:

- I. **ONDE P (onde primarie)** sono **onde longitudinali o di compressione** infatti determinano fenomeni di compressione e rarefazione delle rocce producendo una variazione di volume.
  - Caratteristiche principali:
    - a) Le particelle delle rocce oscillano nella stessa direzione di propagazione dell'onda P.
    - b) Le onde primarie sono quelle che si propagano più rapidamente e sono le prime ad essere rilevate dai sismometri e registrate dai sismografi.
    - c) Possono propagarsi sia nei solidi che nei fluidi.
- II. **Le ONDE S (onde secondarie)** sono **onde trasversali o di taglio** che si propagano con oscillazioni su un piano perpendicolare alla direzione di propagazione.
  - Caratteristiche principali:
    - a) L'effetto prodotto su una roccia è quello della distorsione mentre non si hanno variazioni di volume.
    - b) Le onde S sono più lente di quelle P.
    - c) Le onde S non si propagano nei fluidi.
- III. **Le ONDE R e L (onde superficiali)** sono il risultato della combinazione delle onde P con le onde S.  
Sono quelle che provocano i danni maggiori infatti possiedono una minore velocità ma anche una ampiezza elevata.  
Si propagano a partire dall'epicentro.



Nello studio dell'attività sismica, spesso si prende in considerazione l'**attività strong motion** caratterizzata da vibrazioni di ampiezza e periodo tali da produrre danni su ambiente e infrastrutture ed è rilevabile con i più comuni strumenti.

Per valutare gli effetti dell'attività strong motion su un determinato sito si considerano le 3 componenti ortogonali della traslazione, le quali possono essere registrate in termini di:

- **accelerazione** attraverso accelerometri e accelerogrammi
- **velocità** attraverso velocimetri
- **spostamenti** attraverso sismometri

Dalle registrazioni possono essere ottenuti diversi parametri rappresentativi del moto sismico.

**I parametri più importanti ai fini ingegneristici per rappresentare il moto sismico sono quelli che si ricavano dagli accelerometri ed essi sono:**

- **Aampiezza massima dell'accelerazione (PGA)**

- L'ampiezza massima dell'accelerazione è detta anche accelerazione di picco al suolo (Peak Ground Acceleration) e corrisponde al picco più alto in valore assoluto registrato dall'accelerometro.

Si possono misurare due tipi di PGA:

- **PGHA (Peak Ground Horizontal Acceleration):**

ampiezza massima dell'accelerazione ottenuta considerando una tra le due componenti orizzontali (x o y).

- **PGVA (Peak Ground Vertical Acceleration):**

ampiezza massima dell'accelerazione ottenuta considerando la componente verticale (z).

Per tener conto del potenziale di danneggiamento reale del terremoto spesso si fa riferimento ad un valore di accelerazione "efficace", ottenuto moltiplicando PGA per un fattore riduttivo (per alcune applicazioni ingegneristiche si assume ad es. 0.65)

- **Durata**

- Per calcolare la durata del moto sismico ( $T_d$ ) si fa riferimento alla durata dell'attività di strong motion ed essa può essere calcolata in diversi modi:

- **Durata "bracketed" (T<sub>b</sub>):**

intervallo di tempo compreso tra il primo e l'ultimo superamento di una soglia di accelerazione (di solito compresa tra 0.05 e 0.08 g)

- **Durata di "Trifunac" (T<sub>t</sub>):**

intervallo tempo in cui l'energia della registrazione è compresa tra il 5% e il 95% dell'energia totale.

- **Reciproco della frequenza fondamentale dello spettro del segnale ottenuto applicando la trasformata di Fourier**

- **Contenuto in frequenza**

- Descrive come varia l'ampiezza del moto sismico in relazione alle frequenze contenute nel segnale
- Un onda sismica può essere rappresentata da un segnale periodico e quindi è possibile effettuarne lo sviluppo in serie di Fourier.

Per il teorema di Fourier, una funzione periodica  $s(t)$  di periodo  $T$  si può esprimere come sommatoria infinita di funzioni armoniche (sinusoidi):

$$s(t) = C_0 + \sum_{n=1}^{\infty} [A_n \cos(n\omega t) + B_n \sin(n\omega t)]$$

$$C_0 = \frac{1}{T} \int_0^T s(t) dt$$

$$A_n = \frac{2}{T} \int_0^T s(t) \cos(n\omega t) dt$$

$$B_n = \frac{2}{T} \int_0^T s(t) \sin(n\omega t) dt$$

$$(\omega = 2\pi f = 2\pi/T)$$

- Un accelerogramma è generalmente rappresentato da una funzione discreta  $x(KT) = x_0, x_1, \dots, x_{N-1}$  e quindi per effettuare l'analisi in frequenza viene applicata la Trasformata Discreta di Fourier che consente di ottenere una successione di N numeri complessi  $X_0, X_1, \dots, X_{N-1}$ :

$$X_k = \sum_{n=0}^{N-1} x_n e^{-ik\frac{2\pi}{N}n} \quad k = 0, \dots, N-1$$

- Dallo spettro di Fourier si ricava il valore della frequenza (o del periodo) fondamentale o predominante, cioè quello a cui corrisponde il valore dell'ampiezza massima.

