UNIVERSITÀ DEGLI STUDI DI MILANO

MASTER IN DATA SCIENCE FOR ECONOMICS, BUSINESS AND FINANCE

Protocollo di comunicazione LoRaWAN e implementazione di un progetto IoT tramite la piattaforma di The Things Network

Dott. Marco Richard Morandi

Milano, 20/07/2021

INTRODUZIONE

Internet of Things (IoT) si può definire come un insieme di tecnologie che danno la possibilità di connettere a internet qualsiasi tipo di dispositivo rendendolo “smart”. Dietro questa definizione molto semplicistica esiste un’architettura molto complessa con molteplici sfaccettature e potenzialità.
È possibile provare a scendere un po' più nel dettaglio andando ad elencare sommariamente alcune caratteristiche che un dispositivo dovrebbe avere per far parte dell’Internet of Things:

- Una identità unica per distinguerlo da altri dispositivi simili
- Capacità di comunicazione
- Uno o più sensori
- Un terminale tramite il quale è possibile controllare il dispositivo da remoto

La grande evoluzione che in questo momento sta attraversando il mondo dell’IoT ha come protagonista principale la capacità di comunicazione. Nello specifico si tratta della possibilità di connettere alla rete dispositivi che si trovano a distanze ben maggiori di quella che può coprire un semplice Wi-Fi casalingo utilizzando inoltre un bassissimo dispendio energetico. La tecnologia che lo permette si chiama LoRaWAN ed è già ormai al centro dell’attenzione del mondo IoT da un po' di anni.
A supporto di essa è nato il progetto The Things Network che mira alla creazione e all’espansione di una rete libera e pubblica dedicata all’IoT utilizzando appunto LoRaWAN. Diventa quindi una occasione di studio, in particolar modo per il data scientist, il poter approfondire il funzionamento di questa nuova tecnologia assieme agli strumenti e le metodologie utilizzate per sfruttarla a pieno perché sarà al centro del progresso informatico per parecchi anni e , come stimato dagli analisti di IDC (International Data Corporation), i dati che verranno generati dai dispositivi IoT connessi raggiungeranno addirittura i 79,4 ZB (zettabyte, 1 ZB = 10^{21} byte) entro il 2025.

1. LORAWAN

Per capire LoRaWAN il primo passo è definirne l'acronimo, ovvero **Lo**ng **Ra**nge **W**ide **A**rea **N**etwork, che fornisce già una prima intuizione sulla effettiva capacità di questo tipo di trasmissione wireless.

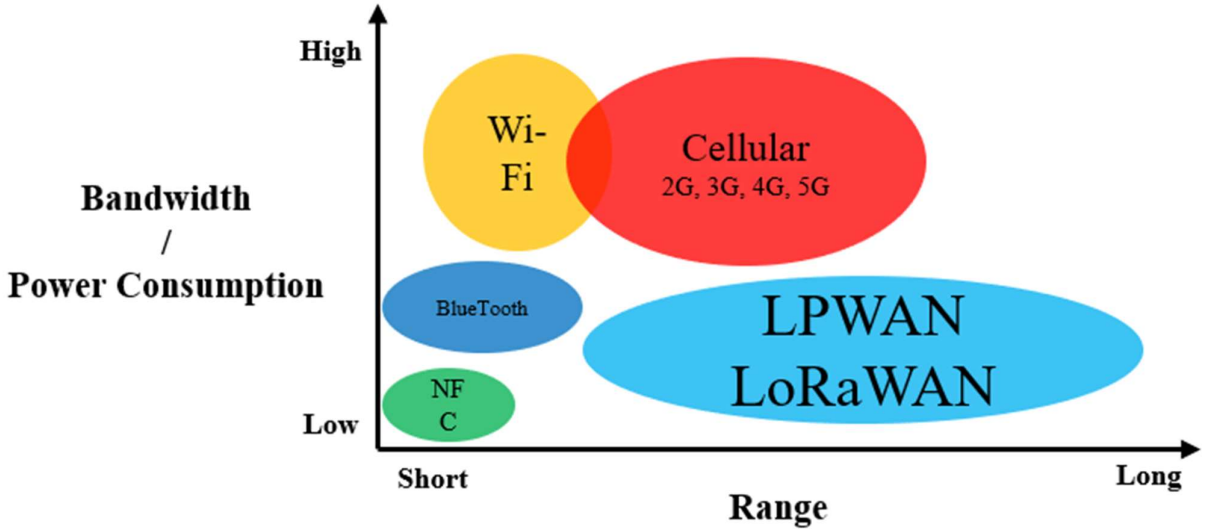
Oggigiorno sono molteplici le tecnologie di trasmissione wireless e di seguito definiremo le più conosciute in termini di capacità di banda, consumo di energia e raggio di azione per poter successivamente delineare il profilo di LoRaWAN.

Il Wi-Fi è la classica connessione utilizzata nelle case di ogni persona; il suo luogo di utilizzo ne sottolinea le sue caratteristiche: una grande capacità di trasmissione dati, ma anche un elevato consumo di energia assieme ad una copertura limitata.

Il BlueTooth e l'NFC sono invece tecnologie destinate ad essere utilizzate maggiormente da dispositivi a batterie in quanto, sebbene abbiano una bassa capacità di trasmissione ed un raggio di azione molto breve, hanno un dispendio energetico estremamente basso.

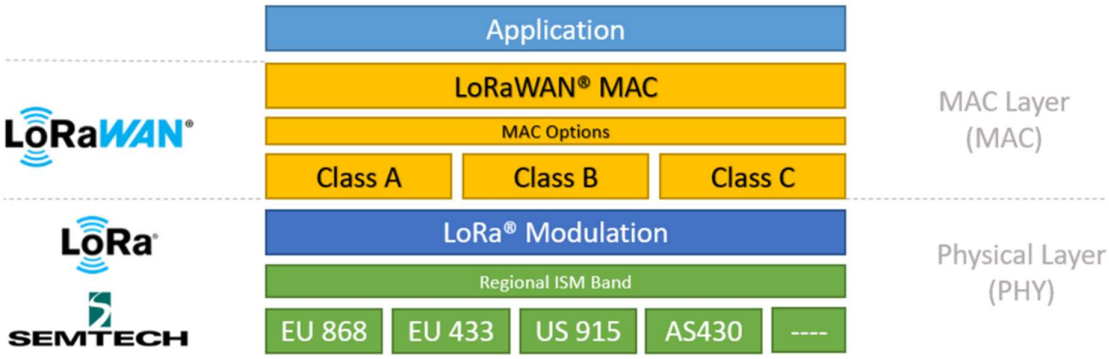
Spostando l'attenzione verso una più ampia copertura troviamo tutte le tecnologie utilizzate dai cellulari, 2G, 3G, 4G e 5G che sono in grado di raggiungere distanze molto elevate garantendo una alta capacità di trasmissione dati, ma con un elevato dispendio energetico.

In un panorama di questo tipo diventa evidente la necessità di poter trovare una tecnologia che sia in grado di coprire grandi distanze con un basso consumo energetico. Necessità che è stata soddisfatta dalla tecnologie LPWAN di cui LoRaWAN è uno tra i protocolli di comunicazione più conosciuti.



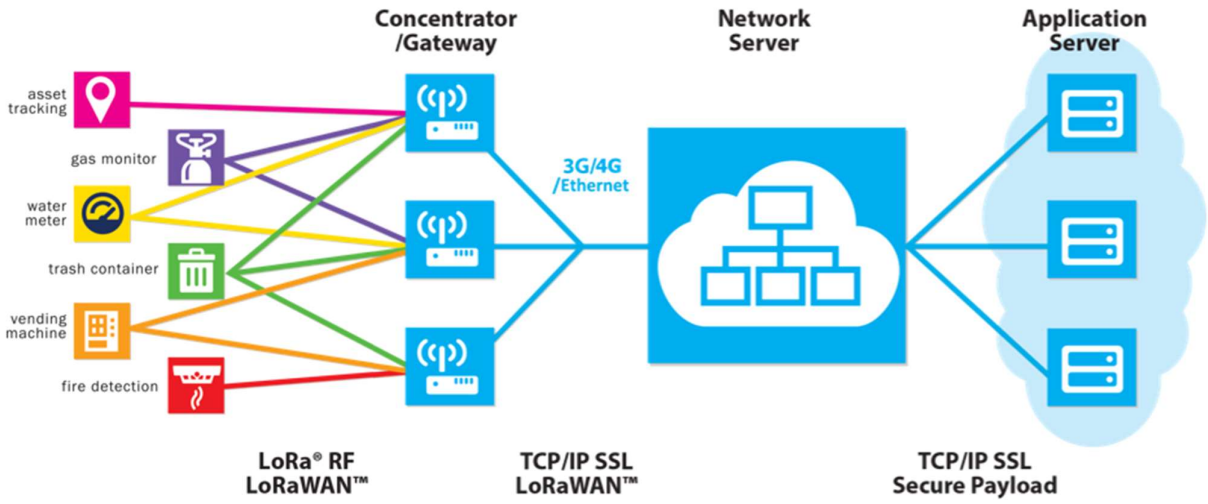
Il successo di LoRaWAN è in gran parte merito di LoRa Alliance, una associazione senza scopo di lucro che mira all'implementazione su larga scala delle tecnologie LPWAN tramite lo sviluppo e la promozione di LoRaWAN.

- **Specifiche**



Tramite le specifiche è possibile identificare meglio LoRaWAN e il suo livello fisico, LoRa.
LoRa (Long Range) è una tecnica di modulazione a lungo raggio, basata sulla frequenza radio e si posiziona a livello fisico. Sfrutta l'intera larghezza di banda del canale durante la trasmissione dei dati, perciò è più immune al rumore e alle fluttuazioni.
Utilizza le bande di frequenza ISM (Industrial, Scientific and Medical) riservate alle applicazioni di radiocomunicazioni per uso industriale, scientifico e medico. In particolare, a seconda dell'area geografica e delle relative regolamentazioni, le due frequenze più diffuse sono 868 Mhz in Europa e 915 MHz in Nord America.
LoRaWAN è invece un protocollo Data Link, quindi si posiziona ad un livello superiore (MAC). È stato concepito per un uso su una rete a lunga portata con l'obiettivo di coniugare alta capacità e bassa potenza. LoRaWAN utilizza tre classi differenti di device, Classe A, B e C che verranno approfonditi successivamente.

- **Struttura**



Di seguito verrà approfondita la struttura che sta alla base di tutte le soluzioni IoT che implementano la tecnologia LoRaWAN.
I dispositivi periferici della struttura sono conosciuti come end nodes o device e dialogano in modalità bidirezionale con dei concentratori, i gateway.
I gateway assieme ai nostri dispositivi periferici costituiscono la rete IoT LoRaWAN.
Esistono tre classi differenti di device: Classe A, Classe B e Classe C.
I dispositivi di classe A supportano la comunicazione bidirezionale tra un dispositivo e un gateway. I messaggi di uplink (dal nodo al gateway) possono essere inviati in qualsiasi momento. Per quanto riguarda i messaggi di downlink (dal gateway al nodo) il device apre due finestre di recezione a orari specificati dopo una trasmissione in uplink. Se non c'è alcuna risposta in nessuna di queste finestre di ricezione, la prossima opportunità sarà dopo la successiva trasmissione di uplink dal device. La risposta può utilizzare la prima finestra di ricezione oppure la seconda, ma non dovrebbe mai utilizzarle entrambe.
I dispositivi di classe B estendono la classe A aggiungendo finestre di ricezione pianificate per i messaggi di downlink dal server.
I dispositivi di classe C estendono a loro volta la classe A mantenendo sempre aperte le finestre di ricezione a meno che non stiano trasmettendo. Ciò consente una comunicazione a bassa latenza ma consuma molta più energia rispetto ai dispositivi di Classe A.

I dati raccolti dai dispositivi periferici vengono mandati in cloud al network server dell'operatore di rete tramite un'infrastruttura di backhauling, come ad esempio la fibra ottica. Il network server gestisce questi dati e li fornisce all'application server per renderli così a disposizione dell'utente finale tramite app o web.

In merito alla sicurezza esistono due tipi di chiavi simmetriche che sono uniche per ogni device LoRa, la NwkSkey e la AppSkey.
La NwkSkey (Network Session Key) è utilizzata per garantire l'integrità del messaggio dal device al Network Server.
L'AppSkey (Application Session Key) è utilizzata per la criptazione end-to-end in AES-128 dal device all'Application Server.
Per quanto riguarda il join con il network i device LoRaWAN hanno due metodi detti OTAA e ABP.
Il primo, Over-the-Air-Activation prevede che il device e il network si scambino una chiave a 128 bit chiamata AppKey. Quando il device spedisce la richiesta per effettuare il join, l'AppKey viene utilizzata per creare un Message Integrity Code (MIC), dopodiché il server controlla il MIC utilizzando l'AppKey. Se il controllo viene passato, il server crea due nuove chiavi a 128 bit, l'App Session Key (AppSkey) e la Network Session Key (NwkSkey). Queste chiavi sono spedite indietro al device, criptate utilizzando l'AppKey come chiave di criptazione. Quando le chiavi vengono ricevute, il device le decripta e le installa.
La seconda modalità, Activation by Personalization, prevede che le chiavi di sessioni vengano inserite manualmente dall'utente, tuttavia questo potrebbe causare problemi di sicurezza.

- **Vantaggi**

Alla luce di quanto visto è possibile riassumere i vantaggi ottenuti utilizzando questa tecnologia:
Lungo raggio: il raggio di azione può arrivare a coprire fino a 5 km in un'area ad alta densità e fino a 15 km in campo aperto.
Basso consumo energetico: i requisiti di alimentazione sono talmente bassi che consentono la creazione di dispositivi a batteria che possono durare fino a 10 anni.
Alta capacità: una rete di questo tipo può supportare milioni di messaggi, tuttavia, il numero di messaggi supportati in una determinata distribuzione dipende dal numero di

gateway installati; un singolo gateway a otto canali può supportare alcune centinaia di migliaia di messaggi nel corso di un periodo di 24 ore. Se è necessaria più capacità, è sufficiente aggiungere ulteriori gateway alla rete

Geolocalizzazione: non viene utilizzato alcun geo localizzatore GPS, ma sono disponibili due metodologie per la geolocalizzazione: un primo metodo si basa sul Received Signal Strength Indication (RSSI), ovvero sulla misurazione della potenza del segnale ricevuto, che fornisce un rilevamento della posizione con un'accuratezza compresa tra 1000-2000m; il secondo metodo si basa sul Time Difference Of Arrival (TDOA), ovvero sulla misurazione della differenza del tempo di arrivo del segnale, che fornisce un rilevamento della posizione con un'accuratezza di 20-200m. In ogni caso un nodo di una rete LoRaWAN può essere localizzato solo se le sue trasmissioni in uplink sono ricevute da tre o più gateway ed è possibile migliorare la precisione della geolocalizzazione aumentando il numero di gateway.

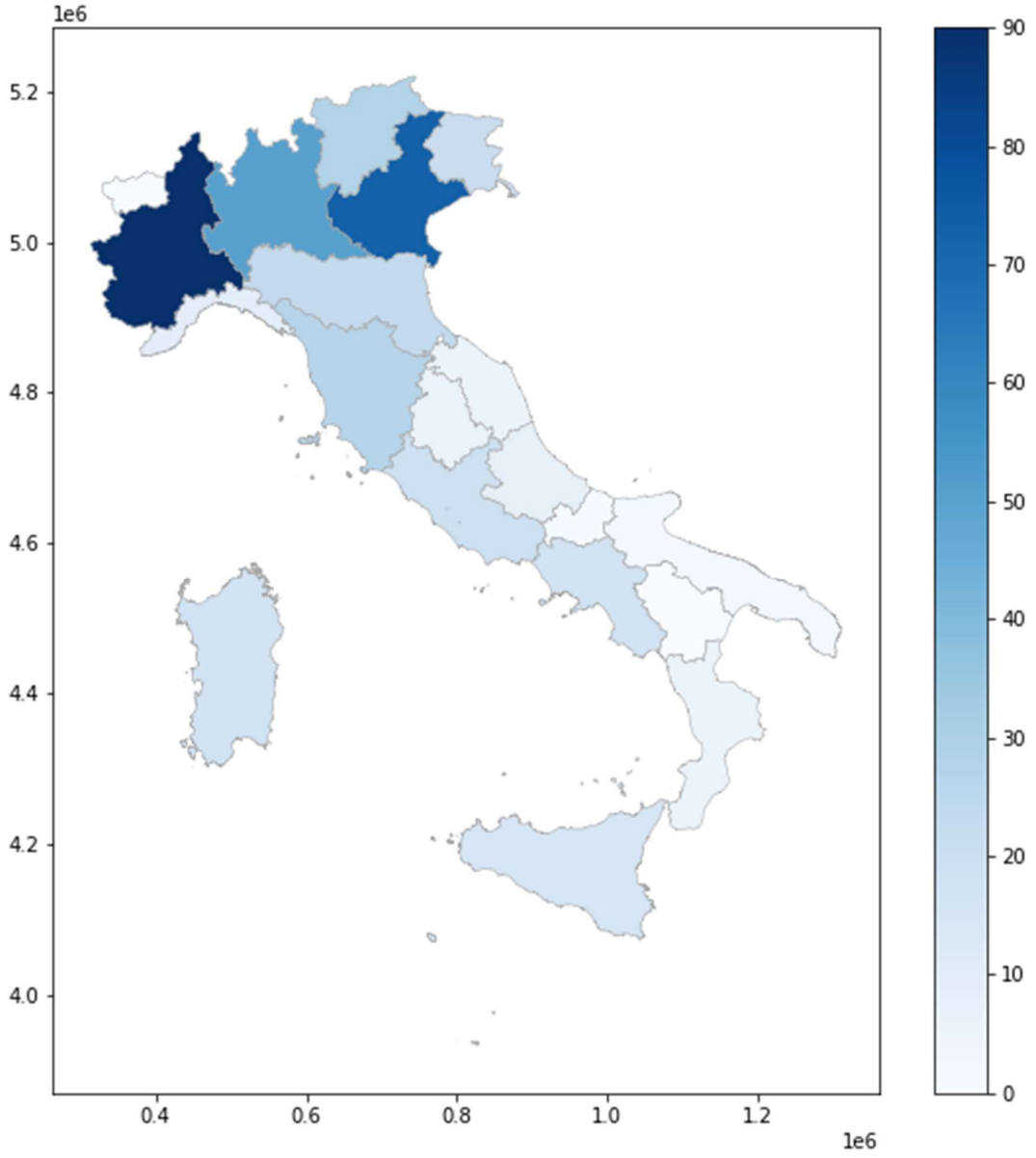
Standardizzato: LoRaWAN è uno standard aperto, dove l'utente finale può tranquillamente cambiare operatore, che sia IoT o di telecomunicazione, senza dover modificare i dispositivi o l'infrastruttura creata garantendo una importante interoperabilità.

Bassi costi: minor consumo di energia, quindi una maggiore durata di vita delle batterie e di conseguenza una diminuzione delle spese di manutenzione. A questo si aggiunge l'utilizzo di frequenze libere, che comportano un grande risparmio rispetto ad altre tecnologie che utilizzano, ad esempio, la rete cellulare tramite SIM che comporta il pagamento di canoni periodici.

Sicurezza: il sistema si basa sull'utilizzo di due chiavi simmetriche per garantire la sicurezza dei dati.

2. THE THINGS NETWORK

The Things Network nasce da una startup olandese che, come obiettivo, si prefigge di lanciare una rete IoT aperta, libera e decentralizzata. Il modello scelto per sviluppare questa piattaforma di IoT urbana è quello del crowdsourcing, aperto ad organizzazioni di comuni cittadini e alle aziende.



Come si può intuire facilmente dall'heatmap c'è una maggiore concentrazione di gateway al nord, ma questa è una situazione temporanea in quanto i dati utilizzati per creare il grafico sono in continuo aggiornamento grazie allo sviluppo continuo della rete tramite l'installazione di nuovi gateway unitamente alla nascita di comunità sparse per tutto il territorio nazionale.

The Things Network mette a disposizione dei propri utenti una piattaforma online, detta console, dove è possibile aggiungere dei propri gateway alla rete esistente, oppure sfruttare quelli già presenti sul territorio per connettere i propri device.

La console, a partire dal 2022, passerà totalmente ad una nuova versione, la numero 3.0, che vanterà una nuova struttura basata su micro-servizi per una migliore distribuzione dei servizi, una migliore scalabilità e interoperabilità con altre reti LoRaWAN garantendo inoltre il supporto per tutte le classi di device (A, B e C) assieme a maggiori integrazioni con servizi di terze parti.

La console permette di monitorare in real-time i propri gateway e device assieme a tutti i dati che generano. Il miglior modo per poter gestire e analizzare il flusso di dati è quello di appoggiarsi ai servizi integrati resi disponibili da the Things Network, i quali sono settabili anche dalla console; di seguito i servizi maggiormente utilizzati:

MQTT (Message Queue Telemetry Transport): protocollo applicativo di messaggistica basato sul paradigma publish / subscribe. The Things Network espone un server MQTT per lavorare con eventi di streaming al quale è possibile connettersi con un client MQTT.

HTTP Webhooks: dette anche API inverse, poiché forniscono in tempo reale ad applicazioni esterne ciò che equivale a una specifica API. La funzione HTTP Webhooks consente all'Application Server di inviare messaggi relativi all'applicazione a specifici endpoint HTTP(S).

Storage Integration: consente di memorizzare i messaggi ricevuti in un database persistente e di recuperarli in un secondo momento. È integrato come un pacchetto applicativo e può essere abilitato per applicazione o per device.

Cloud integrations: integrazione diretta con le più popolari piattaforme IoT, quali ad esempio AWS IoT, Microsoft Azure, ThingsBoard ecc.

3. PROGETTO IOT

Tramite la creazione di un device virtuale è stato possibile testare alcune integrazioni precedentemente elencate e quindi simulare un vero e proprio flusso di dati, sia in downlink che in uplink, dove è stata implementata una analisi in real-time sui flussi di dati ricevuti.

L'intero progetto è stato sviluppato utilizzando Python e la nuova versione 3.0 della console di the Things Network.
Si può suddividere in tre fasi:
downlink: normalmente il downlink è l'invio di dati dal terminale al device per aggiornare, impostare o modificare ogni aspetto del device; in questo caso verrà utilizzato per "caricare" all'interno del device virtuale dati che normalmente un device raccoglie attraverso i suoi sensori.
uplink: si tratta della "raccolta" dei dati generarti dal device da parte del terminale tramite servizi integrati.
analisi dati: è stata impostata una analisi per ogni flusso di dato scaricato dal device e una seconda analisi che compara tutti i flussi giornalieri raccolti fino a quel momento.

I dati che verranno inviati tramite downlink al device, e successivamente recuperati tramite uplink, sono dati reali generati da un device con sensori ambientali. Nello specifico verranno utilizzati i dati relativi all'umidità.
Il dataset originale contiene dati rilevati da tre device nel periodo che va dal 12/07/2020 al 19/07/2020 ed è recuperabile al seguente link:
https://www.kaggle.com/garystafford/environmental-sensor-data-132k

- **Downlink**

Del dataset originale vengono utilizzati i dati relativi ad un sensore (id: b8:27:eb:bf:9d:51) e a loro volta vengono suddivisi in quattro dataset ognuno che copre la fascia oraria che va dalle 00:00 alle 23:59 per i giorni 16, 17, 18 e 19 luglio. La suddivisione è stata fatta per poter simulare nella seconda fase quattro flussi differenti provenienti dallo stesso device che coprono, in giorni differenti, le medesime fasce orarie.

Una volta creati i quattro dataset e salvati come file .csv vengono inviati ciascuno mediante il servizio integrato HTTP webhooks, utilizzato mediante la libreria requests di Python.

Nello specifico il file .csv viene nuovamente caricato come un dataframe pandas e trasformato in un dizionario.

```
# Prepare csv for downlink and send
def downlink_http(app, webhook, dev, passw, d):

    x = base64.b64encode(str(d).encode("ascii")).decode("ascii")

    headers = {'Authorization': 'Bearer ' + passw + ''}
    data = '{"downlinks":[{"frm_payload":"' + x + '","f_port":15,
"priority":"NORMAL"}]}'

    r = requests.post(
        'https://eu1.cloud.thethings.network/api/v3/as/applications/' + app +
'/webhooks/' + webhook + '/devices/' + dev + '/down/push',
        headers=headers, data=data)

    print("URL: {}\n".format(r.url))
    print("Status: {} {}\n".format(r.status_code, r.reason))
    print("Datetime: {}\n".format(datetime.now().strftime('%Y%m%d%H%M')))
    print("Response: {}\n".format(r.text))
```

Successivamente, attraverso la funzione **downlink_http** il dizionario viene codificato e passato al parametro *data* del metodo *requests.post()*. L'URL utilizzato prevede l'inserimento dell'ID dell'application, dell'ID del device e del webhook utilizzati. Per quanto riguarda il parametro *headers* va inserita la API key che può essere generata all'interno della piattaforma web.

All'interno della documentation di The Things Network è presente una spiegazione dettagliata di come utilizzare le varie funzioni e a completamento è presente anche un forum estremamente attivo.

La funzione viene richiamata per ognuno dei quattro dataset creati fornendo per ciascuno di essi l'output relativo all'URL, allo Status, al Datetime e al Response.

Trattandosi di un device simulato all'interno della console di The Things Network i downlink non sono abilitati fino a che non viene riconosciuta una sessione valida, motivo per cui la procedura di downlink restituisce come risposta:

```
(no device session; check device activation)
```

Durante la fase di uplink vedremo che il flusso di dati viene comunque rilevato dal sistema rendendo così possibile il completamento del progetto.

- **Uplink**

La fase di uplink prevede il recupero dei dati che abbiamo precedentemente caricato attraverso il servizio HTTP webhook.
La problematica precedentemente descritta non permette il download attraverso il servizio di data storage, che sotto molti aspetti risulta il metodo più diretto e flessibile rispetto ad altri servizi, ma è possibile risolvere l'impedimento attraverso l'utilizzo del servizio MQTT server sottoscrivendosi, tramite un MQTT client, al topic di proprio interesse, che in questo caso è relativo ai downlink falliti (down/failed) .

```
def mqtt_sub(broker, port, appid, passw):
    def on_connect(client, userdata, flags, rc):
        print("Connected with result code " + str(rc))
        client.subscribe('v3/+/devices/+/down/failed')  # subscribe to failed
downlinks
        client.subscribe('v3/+/devices/+/down/sent')  # subscribe to uplinks
        client.subscribe('v3/+/devices/+/up')  # subscribe to uplinks

    def on_message(client, userdata, msg):
        print(msg.topic + " " + str(msg.payload))
        on_message.counter += 1
        date = datetime.today().strftime('%Y_%m_%d_%H_%M_%S')
        if '/failed' in msg.topic:
            a = [json.loads(msg.payload.decode('utf-8'))]
            df = pd.json_normalize(a)
            print(type(df), df)
            data = df['downlink_failed.downlink.frm_payload'][0]
            data = str(base64.b64decode(data))[14:-2]
            data = ast.literal_eval(data)
            df_data = pd.json_normalize(data)
            name_file = date + '_' + str(on_message.counter)
            with open(config['data_download']['path_output'] + name_file +
'.csv', 'a', newline='') as d:
                df_data.to_csv(d, header=d.tell() == 0)
                d.close()

            df = pd.read_csv(config['data_download']['path_output'] + name_file
+ '.csv', index_col='DateTime').drop(labels="Unnamed: 0", axis=1)
            df = df.sort_index()
            df.index = pd.to_datetime(df.index, infer_datetime_format=True)

            #functions for analysis
            describe_add(df=df, colname=config['variable_analyzed']['variable']
, name_file=name_file)
            hist_box(df=df, colname=config['variable_analyzed']['variable']
, name_file=name_file)
            top_v_range(df=df, colname=config['variable_analyzed']['variable']
```

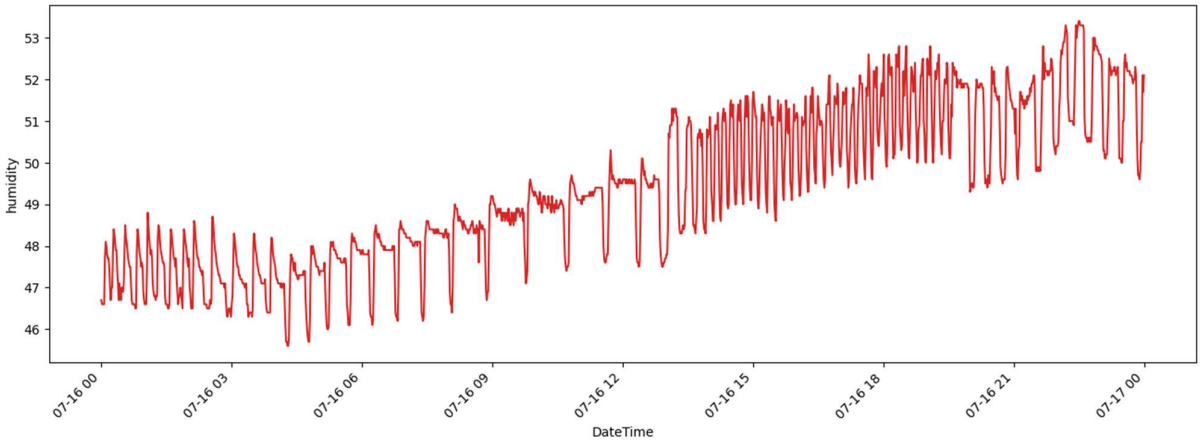
```
, name_file=name_file)
            plot_d(df=df, colname=config['variable_analyzed']['variable']
, name_file=name_file)

        elif '/up' in msg.topic:
            name_file = date + '_' + str(on_message.counter)
            a = [json.loads(msg.payload.decode('utf-8'))]
            df = pd.json_normalize(a)
            print(type(df), df)
            with open(config['data_download']['path_output'] + name_file +
'.csv', 'a', newline='') as f:
                df.to_csv(f, header=f.tell() == 0)
                f.close()

    def on_log(client, userdata, level, buf):
        print("LOG: message:" + str(buf))
        print("LOG: userdata:" + str(userdata))

    on_message.counter = 0

    client = mqtt.Client()
    client.on_log = on_log
    client.on_connect = on_connect
    client.on_message = on_message

    client.username_pw_set(appid, passw)
    client.connect(broker, port, 60)

    client.loop_forever()
```

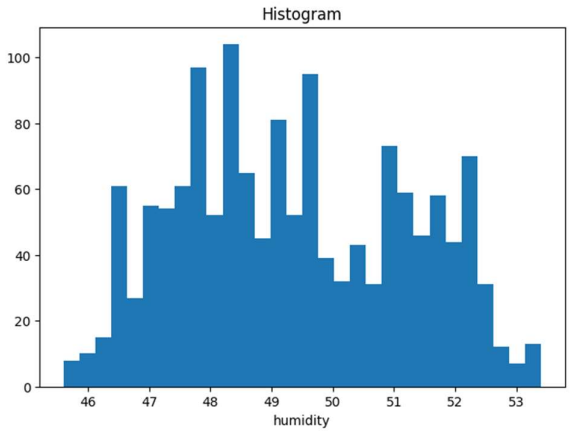
Per l'utilizzo di questo servizio è stata utilizzata la libreria *paho.mqtt* di Python che fornisce una classe client che consente la connessione ad un server(broker) MQTT per pubblicare messaggi, sottoscriversi a topic e ricevere messaggi pubblicati.
La maniera in cui la classe Client è stata progettata ricorda una piattaforma ad eventi dove ad ogni evento che si verifica, viene associata una funzione particolare che lo cattura e lo processa. La classe Client nativa ha definito dentro di sé una serie di metodi callback predefiniti che possiamo liberamente modificare a seconda dei nostri scopi.
Difatti le callback utilizzate e modificate sono:
*on_connect***,** attraverso cui possiamo avere un responso in merito alla richiesta di connessione e richiedere la sottoscrizione a determinati topic.
on_message alla quale in questo caso sono state dati istruzioni di decodificare il flusso dati, salvarlo in un file .csv e tramite alcune funzioni effettuare dei riepiloghi analitici che verranno salvati in formato .png(tabelle e grafici).
on_log, che permette di monitorare al meglio tutta la procedura tramite i log.

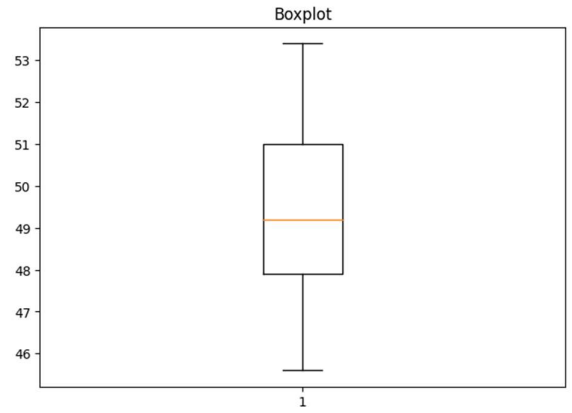
La funzione principale *mqtt_sub()* ha bisogno di ricevere il broker, la porta, l'ID dell'application alla quale sottoscriversi e la API key.
I dati passati vengono utilizzati per procedere con la connessione al server MQTT. Una

volta che si verificheranno determinati eventi verranno richiamate le callback precedentemente definite.
Molti dei parametri e delle funzionalità utilizzate vengono forniti e dettagliati nella documentation di The Things Network e di paho-mqtt.

Per quanto riguarda i riepiloghi analitici attraverso un diagramma cartesiano è stato rappresentato l'andamento dell'umidità durante la fascia oraria analizzata e attraverso l'istogramma e il boxplot viene evidenziata la distribuzione.







Per quanto riguarda le tabelle, la prima è stata costruita richiamando il metodo *.describe* a cui sono stati aggiunti un paio di parametri (mediana e moda); la successiva riporta il tempo di inizio e di fine della serie storica assieme alla durata; le ultime due tabelle riportano rispettivamente i valori più alti e più bassi rilevati assieme al momento in cui è stato rilevato il valore.

index	humidity
count	1440.0
mode	48.4
sum	71088.1
mean	49.36674
std	1.84165
median	49.2
min	45.6
25%	47.9
50%	49.2
75%	51.0
max	53.4

index	DateTime
start	2020-07-16 00:00:00
end	2020-07-16 23:59:00
range	0 days 23:59:00

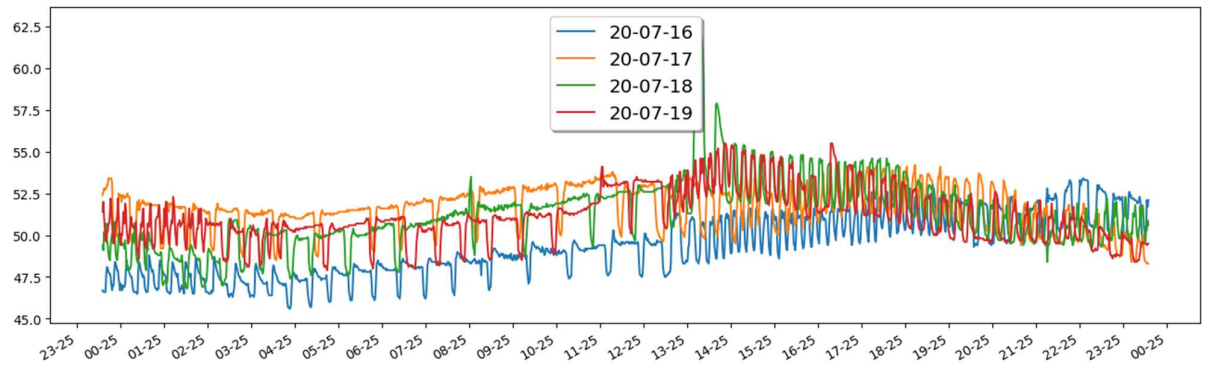
DateTime	humidity
2020-07-16 22:30:00	53.4
2020-07-16 22:29:00	53.4
2020-07-16 22:32:00	53.3
2020-07-16 22:25:00	53.3
2020-07-16 22:28:00	53.3

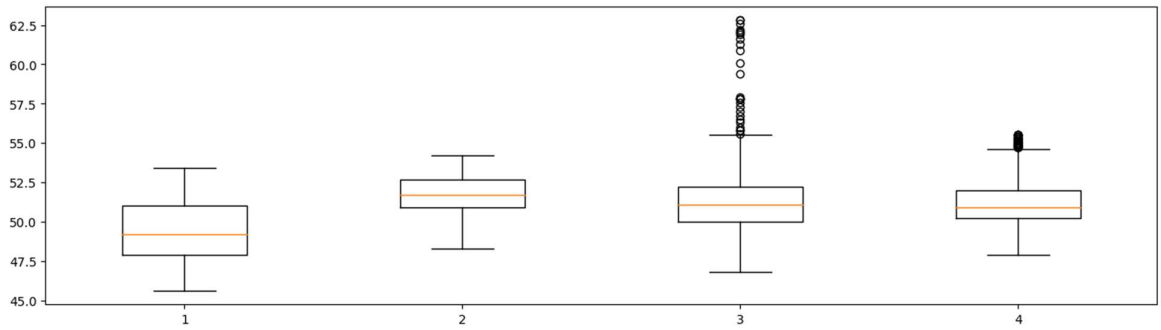
DateTime	humidity
2020-07-16 04:18:00	45.6
2020-07-16 04:17:00	45.6
2020-07-16 04:15:00	45.7
2020-07-16 04:47:00	45.7
2020-07-16 04:16:00	45.7

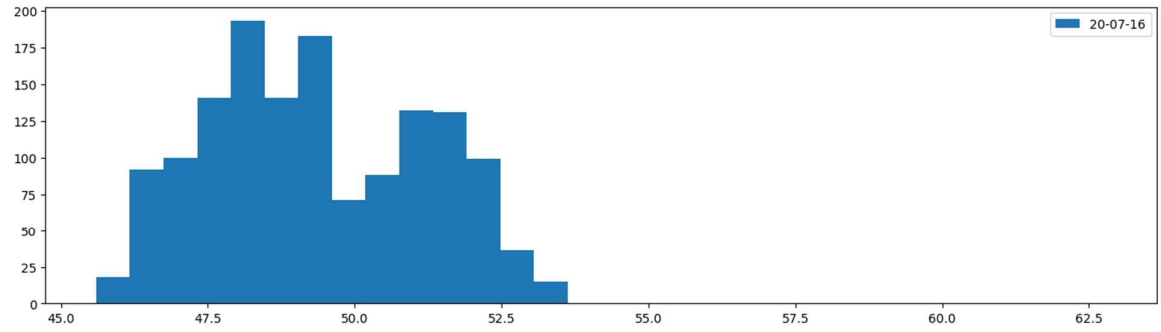
Utilizzando questo tipo di metodo è necessario prima avviare l'iscrizione al topic di proprio interesse sfruttando il servizio MQTT appena descritto e successivamente procedere con il downlink.

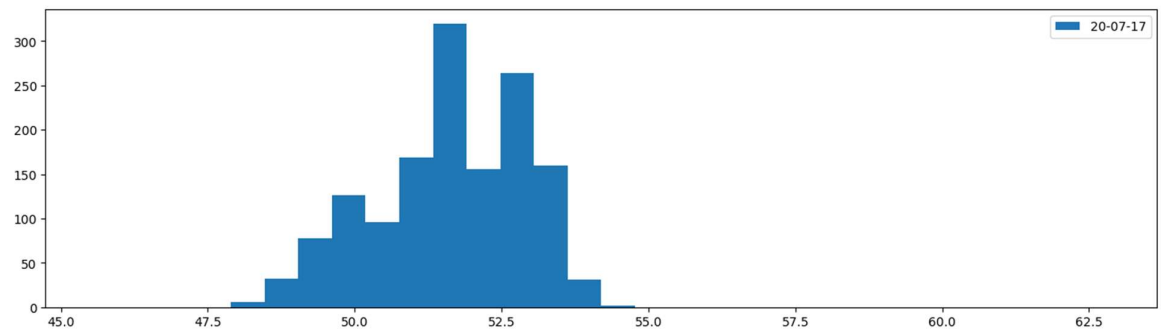
- **Comparazione dati**

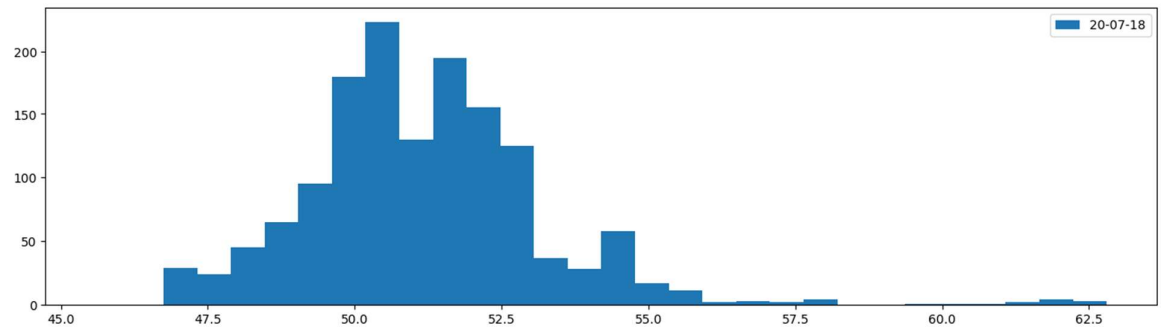
Per ultimo si procede ad una comparazione di tutti i flussi giornalieri ricevuti riproponendo gli stessi elementi analitici, ma con una comparazione diretta tra i flussi.

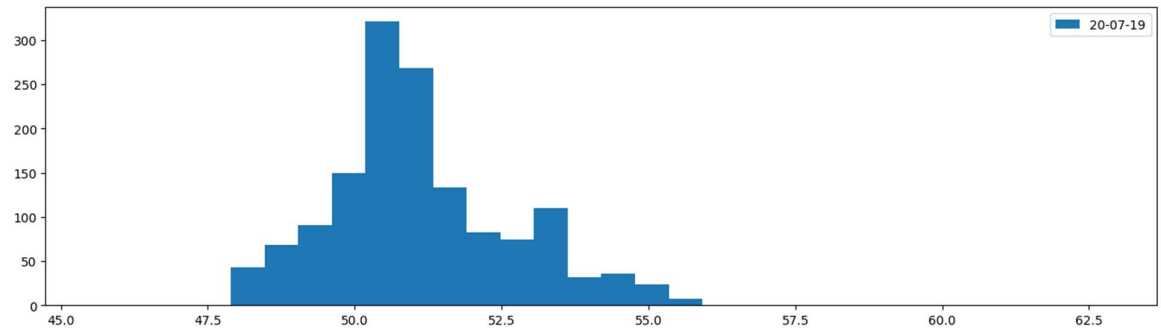



62.5
60.0
57.5
55.0
52.5
50.0
47.5
45.0
1
2
3
4


20-07-16
200
175
150
125
100
75
50
25
0
45.0
47.5
50.0
52.5
55.0
57.5
60.0
62.5


20-07-17
300
250
200
150
100
50
0
45.0
47.5
50.0
52.5
55.0
57.5
60.0
62.5


20-07-18
200
150
100
50
0
45.0
47.5
50.0
52.5
55.0
57.5
60.0
62.5


20-07-19
300
250
200
150
100
50
0
45.0
47.5
50.0
52.5
55.0
57.5
60.0
62.5

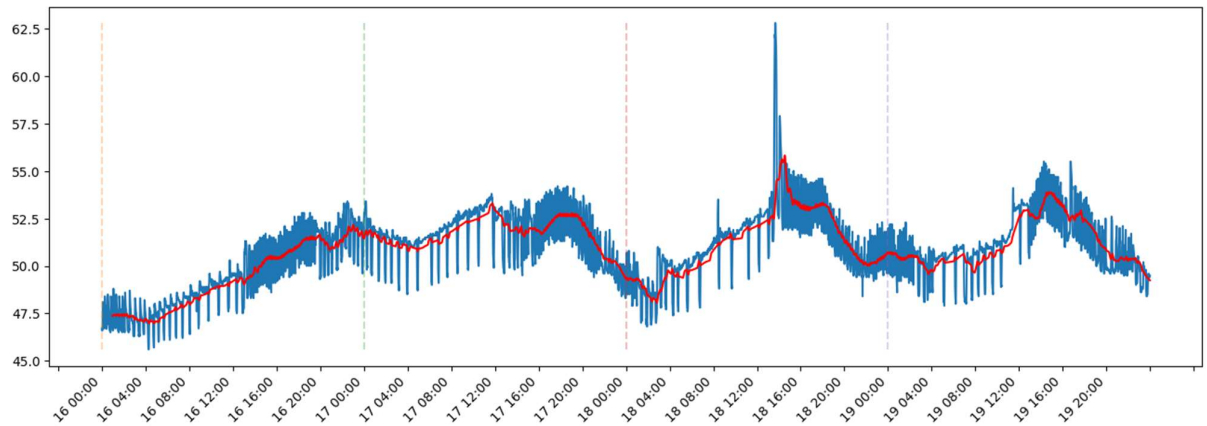
index	20-07-16	20-07-17	20-07-18	20-07-19
count	1440.0	1440.0	1440.0	1440.0
mode	48.4	51.5	50.0	50.7
sum	71088.1	74380.9	73729.8	73609.5
mean	49.36674	51.6534	51.20125	51.11771
std	1.84165	1.26557	2.03981	1.5382
median	49.2	51.7	51.05	50.9
min	45.6	48.3	46.8	47.9
25%	47.9	50.9	50.0	50.2
50%	49.2	51.7	51.05	50.9
75%	51.0	52.7	52.2	52.0
max	53.4	54.2	62.8	55.5

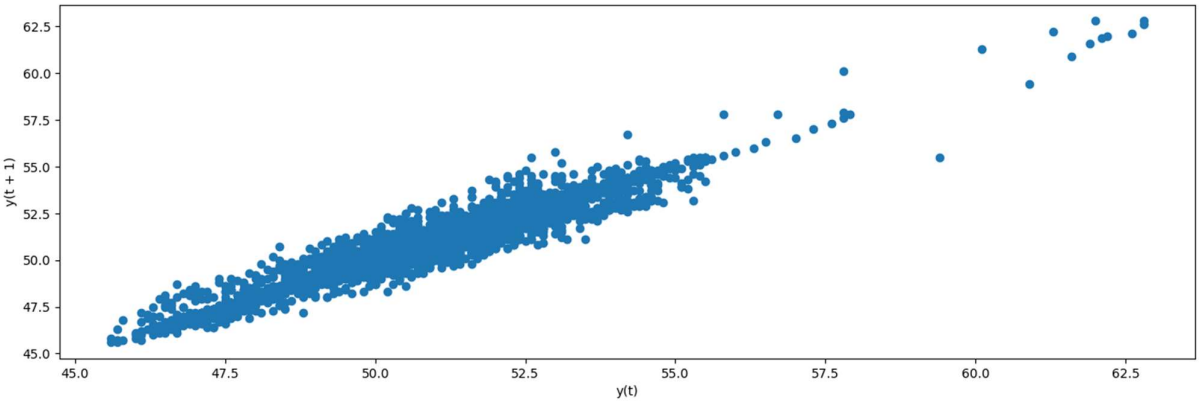
index	DateTime
start	2020-07-16 00:00:00
end	2020-07-19 23:59:00
range	3 days 23:59:00

DateTime	humidity
2020-07-18 13:40:00	62.8
2020-07-18 13:41:00	62.8
2020-07-18 13:42:00	62.6
2020-07-18 13:38:00	62.2
2020-07-18 13:43:00	62.1

DateTime	humidity
2020-07-16 04:18:00	45.6
2020-07-16 04:17:00	45.6
2020-07-16 04:47:00	45.7
2020-07-16 04:46:00	45.7
2020-07-16 04:16:00	45.7

In aggiunta è stata creata una serie storica ottenuta aggregando tutti i dati raccolti, con una media mobile che ha una finestra di 60 minuti. Utilizzando la serie storica appena creata è stato fatto anche un lag plot con un lag pari a 1.





I dati usati coprono l'intera giornata, per soli quattro giorni, e sebbene si tratta di un periodo abbastanza breve si può comunque rilevare un trend lineare positivo, sia indicato da una media mobile in leggera crescita e sia dal lag plot. Gli spike rilevati, in particolare durante il giorno 18 luglio, potrebbero essere determinati da eventi esterni quali l'avvicinamento di persone e/o animali al sensore tale da determinare un brusco innalzamento dell'umidità ed una graduale, ma sempre piuttosto ripida, discesa del valore.

CONCLUSIONI

La nuova tipologia di trasmissione dati LPWAN, attraverso il protocollo più conosciuto LoRaWAN, assieme a tutto ciò che ne è derivato ha permesso un enorme sviluppo del mondo IoT.

Se già tempo addietro si iniziava a strutturare un ambiente IoT non più grande di una casa (smart home), oggigiorno si mira a creare una vera e propria smart city, portando la gestione di tutti i servizi cittadini ad un livello superiore di efficacia e di controllo.
Non bisogna dimenticare il potenziale utilizzo in tutte le realtà di impresa, partendo da un semplice ufficio fino ad arrivare a grossi impianti produttivi (industria 4.0) passando per gli innumerevoli nuovi servizi che potrebbero nascere in virtù di questa nuova tecnologia.

Con questa prospettiva di crescita in numerosi ambiti diventerà fondamentale l'essere in grado di capire, gestire e analizzare i dati che verranno generati da un numero sempre più crescente di dispositivi IoT in un mondo che diventerà sempre più "smart".