Università Politecnica delle Marche

Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Tesi di Laurea

STUDIO DELLA PIATTAFORMA IOT ALTAIR

STUDY OF ALTAIR IOT PLATFORM

Relatore:

Prof. **Ennio Gambi**

Candidato:

Roberto D'Alonzo

Anno Accademico 2018-2019

Ai miei genitori,

a Clara e Mario,

a Silvia e Mattia.

Indice

INTRODUZIONE

"…Se avessimo computer in grado di conoscere tutto ciò che c'è da sapere sulle cose, utilizzando dati raccolti senza alcun aiuto da parte nostra, saremmo in grado di monitorare e conteggiare ogni cosa e di ridurre notevolmente sprechi, perdite e costi. Potremmo sapere quando le cose devono essere sostituite, riparate o richiamate, e se sono fresche o hanno superato il loro momento migliore...".

Sono queste parole di Kevin Ashton[1], datate 1999, che hanno segnato la nascita di una delle tecnologie più in via di sviluppo negli ultimi anni e che possiamo definire come la digitalizzazione online del nostro mondo fisico. La tecnologia in questione è l'Internet of Things (Internet delle cose), in breve IoT.

Fino a qualche anno fa nessuno avrebbe mai immaginato di arrivare a tanto: un qualsiasi oggetto, che prima era considerato "offline", prende vita collegandosi alla rete e comunicando con miliardi di altri dispositivi. Quindi l'oggetto in questione diventa intelligente e in grado di prendere decisioni autonomamente in base a ciò che riesce a misurare dall'ambiente circostante e a ciò che riceve da altre entità connesse. Esempi significativi sono le confezioni di farmaci che possono avvertirci se non li stiamo assumendo come stabilito e ci possono dire come ovviare a una eventuale dimenticanza, oppure un orologio che può ricordarci appuntamenti e verificare se effettivamente li rispettiamo.

Affinché un oggetto possa considerarsi parte dell'IoT necessita di una connessione internet alla quale connettersi, di un indirizzo IP che ne consenta l'identificazione univoca e della capacità di inviare e ricevere dati in modo autonomo.

È in questo ambito che entrano in gioco le piattaforme cloud IoT che si occupano della gestione dei dispositivi, fornendo un'interfaccia semplice agli utenti per acquisire, elaborare e analizzare i dati provenienti dai dispositivi stessi, i quali possono essere PC, Smartphone o semplici oggetti di uso quotidiano dotati di sensori per misurare le grandezze di interesse. Ovviamente, come già detto, tutto deve essere fatto attraverso la connessione a internet.

La diffusione capillare della tecnologia IoT ha permesso lo sviluppo di un gran numero di tali piattaforme cloud come, per citarne alcune, Google Cloud IoT, Thinkspeak (che

[1] **Kevin Ashton** è un ingegnere inglese e ricercatore presso il MIT (Massachusetts Institute of Tecnology); considerato il pioniere dell'IoT.

utilizza Matlab per visualizzare i dati), Altair SmartCore. È proprio sullo studio di quest'ultima piattaforma che verterà il lavoro del presente progetto di tesi, andando ad analizzarne nel dettaglio le caratteristiche, l'organizzazione e il funzionamento. Tutto questo sarà approfondito nel Capitolo 3.
Affinché gli oggetti in questione possano fornire costantemente una grande quantità di dati, si rende necessaria la definizione di un protocollo di comunicazione standard sul quale basare le implementazioni IoT. Tra questi, l'Organization for the Advancement of Structured Information Standard (OASIS) ha dichiarato che il protocollo MQTT[2] di IBM è lo standard di riferimento per la comunicazione per l'Internet delle Cose.

L'obiettivo del suddetto lavoro di tesi è, quindi, quello di offrire una panoramica sul mondo dell'IoT e sul suo protocollo standard MQTT per poi andare a studiare gli aspetti legati alla piattaforma cloud Altair SmartCore e vedere come quest'ultima lavora in questa prospettiva.

[2] **MQTT**: MQ Telemetry Transport è un protocollo open source "leggero", sviluppato e ottimizzato per dispositivi vincolati e reti a bassa ampiezza di banda, ad alta latenza o inaffidabili.

CAPITOLO 1: IoT

1.1 Definizione IoT

Il termine IoT ("Internet of Things" o letteralmente "Internet delle Cose") è un neologismo utilizzato nelle telecomunicazioni e coniato per la prima volta nel 1999 da Kevin Ashton: un ricercatore presso il MIT (Massachusetts Institute of Tecnology). Tale appellativo nasce dall'esigenza di attribuire un nome agli oggetti reali connessi ad internet. Quindi con IoT si indica un insieme di tecnologie che permettono di collegare alla rete qualunque tipo di apparato. Lo scopo di queste soluzioni è sostanzialmente quello di monitorare, controllare e trasferire informazioni/dati per poi svolgere azioni conseguenti. Il significato di IoT si esprime bene con degli esempi: IoT è ad esempio una casa che accende i riscaldamenti appena ci sente arrivare, oppure un climatizzatore che ci avverte o avverte il centro assistenza quando c'è un guasto, o una serratura che si apre e chiude con lo smartphone. Questi sono modelli di IoT, ovvero di oggetti che, collegati alla rete, permettono di unire mondo reale e virtuale. Tale tipo di comunicazione tra "cose" viene generalmente denotato con la sigla **M2M** e sta ad indicare una comunicazione **Machine to Machine** (macchina a macchina) che permette a dispositivi wireless o cablati di dialogare insieme.

Virtualmente qualsiasi oggetto può essere dotato di un dispositivo elettronico con un software in grado di collegarsi a internet o a una rete locale: da una semplice lampadina, un frigorifero, una lavatrice, una macchina del caffè a automobili, segnali stradali, semafori, torni industriali. Insomma tutto ciò che ci circonda ha la possibilità, con l'IoT, di diventare interattivo e "intelligente".

Quindi un oggetto qualunque, per diventare parte dell'Internet of Things, ha bisogno:

- di un sensore in grado di misurare un qualche tipo di dato;
- di una connessione a internet;
- di un **indirizzo IP** che ne consenta l'identificazione univoca;
- della capacità di trasmettere e ricevere dati senza l'intervento umano.

Quest'ultima può essere fornita direttamente dal produttore, sotto forma di firmware[3], ma è probabile che in futuro si diffonderanno sempre di più software commerciali destinati a controllare elettrodomestici e dispositivi di ogni genere.

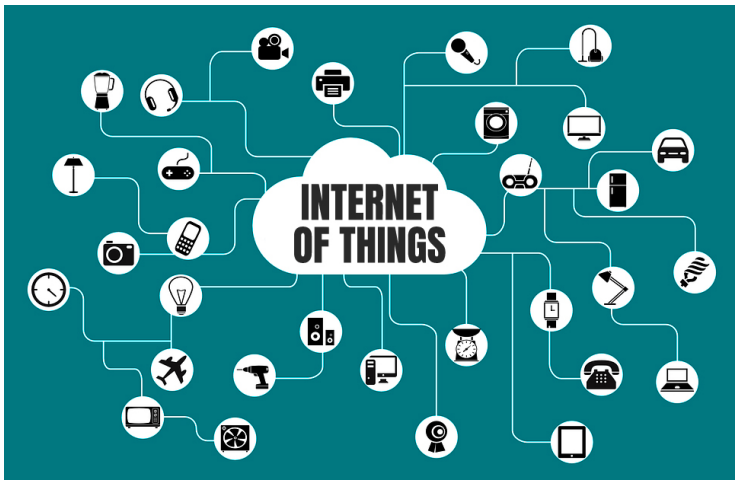


Figura 1: Interconnessione tra dispositivi nell'IoT

Internet si fa quindi mezzo di comunicazione non soltanto tra gli esseri umani, ma anche per le cose che usiamo frequentemente, rendendole in grado di "parlare" con altri oggetti del loro tipo di uso, o delle proprie condizioni attuali.

1.2 Evoluzione IoT

Sebbene il termine Internet of Things sia comparso, come detto, nel 1999 come ideologia della tecnologia che avrebbe drasticamente cambiato il mondo, è solo dopo il 2005 che nasce ufficialmente l'IoT come reale interconnessione tra internet e gli oggetti e, di conseguenza, tra le persone. Questo soprattutto grazie alla nascita degli Smartphone che hanno dato una forte spinta in tal senso.

La fase che possiamo definire precedente all'IoT è rappresentata dalla "sensoristica semplice": sensori in grado di **rilevare** informazioni dall'ambiente circostante (dati su temperatura, qualità dell'aria, movimento dei veicoli,...) trasformandole in dati digitali. Mancava però, in questa fase, la connessione a internet e i sensori erano interrogati "manualmente".

[3] **Firmware**: Insieme delle istruzioni e delle applicazioni presenti permanentemente nella memoria di un sistema e che non possono essere modificate dall'utente.

Il passaggio dalla sensoristica semplice all'IoT, è rappresentato appunto dall'introduzione della connessione alla rete: il sensore in questo caso non si limitava a **rilevare** i dati della "cosa" che "**parla**" (della sua temperatura, della qualità dell'aria,...), ma inoltre **metteva in rete** queste informazioni. È a questo punto che, quindi, si è iniziato a parlare effettivamente di Internet of Things.
A sua volta, anche la vita dell'IoT stessa, è stata oggetto di varie fasi di crescita che fanno riferimento a un lavoro importante sui dati. Possiamo schematizzare tali step come segue:

1. Dispositivi, connessi in rete, in grado di rilevare dati e di comunicarli;
2. Dispositivi, connessi in rete, capaci di misurare più tipologie di dati e di trasferirli;
3. Dispositivi, connessi in rete, che effettuano un primissimo livello di elaborazione (selezione) dei dati per poi trasferire solo quelli che corrispondono a determinati requisiti;
4. Dispositivi, connessi in rete, in grado di raccogliere dati, effettuare un primo livello di selezione, svolgere azioni in funzione di indicazioni ricevute e infine comunicare le informazioni richieste;
5. Dispositivi, connessi in rete, capaci di rilevare e collezionare dati, di selezionarli, di trasmettere solo quelli necessari al progetto nei quali sono coinvolti e di effettuare azioni sulla base delle informazioni ricevute e in funzione di una capacità elaborativa locale.

Le tecnologie usate dagli oggetti per collegarsi alla rete sono diverse: molto comune è la connessione a una WPAN (Wireless Personal Area Network), utilizzata per distribuire informazioni su distanze relativamente brevi e senza cavi; alcuni dispositivi, invece, potrebbero essere dotati di schede SIM in grado di collegarsi a internet tramite un servizio di traffico dati mobile ma è facile supporre che molti dei servizi domestici si collegheranno via WIFI alla rete casalinga, che utilizzeranno quindi per accedere a internet. Tuttavia, sempre nel campo della domotica, molti dispositivi come i grandi elettrodomestici e quelli da incasso, potrebbero anche utilizzare una connessione via Ethernet, dato che si tratta di oggetti fissi e già cablati dalla rete elettrica.

1.3 Funzionamento IoT

Per comprendere a fondo l'essenza e la comodità dell'Internet of Things, è necessario capirne il funzionamento. Per poter svolgere il proprio lavoro, l'IoT ha bisogno di raccogliere e archiviare una grossa mole di dati real time (ad esempio dai sensori, dai semafori o da qualunque altro dispositivo connesso). Le informazioni contenute in questi dati, dopo essere state elaborate e analizzate, serviranno a ottenere il monitoraggio di qualsiasi attività desiderata, sfruttando l'oggetto dedicato allo scopo, ma anche a semplificare il controllo a distanza incentivando l'automazione, anche in ambito domestico. Il tutto viene fatto per migliorare la nostra qualità di vita, per risparmiare in termini di costi e consumi e per aumentare la sicurezza. Non più solo le persone, o le "persone giuridiche", e le imprese sono riconoscibili sulla rete internet, con l'IoT anche le cose possono esserlo. Queste ultime acquisiscono intelligenza e quindi vanno a ricoprire un ruolo attivo nella società comunicando tra loro e apprendendo autonomamente informazioni. Ad esempio una semplice videocamera non è più solo nella condizione di acquisire ed eventualmente inviare immagini e dati, ma lo fa in modo intelligente in funzione di parametri di interesse che possono evolvere nel tempo.

Figura 2: Esempio di funzionamento IoT

Nella Figura 2 è mostrato un esempio molto semplice ma allo stesso tempo molto istruttivo dell'importanza che può assumere l'IoT nella vita di tutti i giorni: le due videocamere sono poste all'ingresso di due parcheggi diversi e sono connesse ad internet; anche l'automobile, che si appresta a cercare parcheggio, è connessa alla rete; possiamo immaginare che la videocamera del parcheggio a sinistra comunichi all'automobile, con un messaggio, l'impossibilità di sostare lì perché tutto occupato, mentre la videocamera posta all'ingresso del parcheggio a destra informi l'automobile che ci sono due posti disponibili. Quindi, dopo aver ricevuto l'ok dalla videocamera di destra, il conducente dell'auto può direttamente recarsi sul posto per parcheggiare risparmiando tempo ed evitando inutili ricerche.
Questo è solo un esempio che descrive a grandi linee la forza di questa nuova tecnologia che sta prepotentemente entrando nella vita di tutti i giorni di ognuno.
Collegandosi **autonomamente** alla rete, gli oggetti creano una sorta di mappa virtuale del mondo reale, in cui convivono scambiandosi tra loro dati e autoistruendosi.

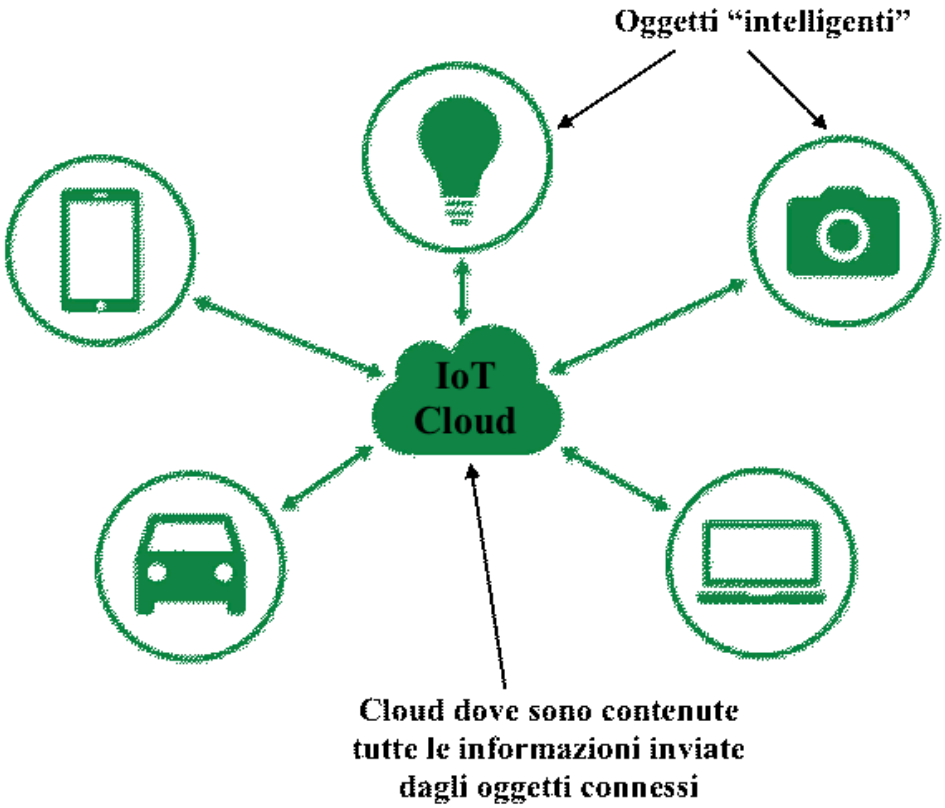


Figura 3: Mappa virtuale oggetti nell'IoT

Ma cosa può essere misurato con i dati? Alcuni esempi di parametri misurabili e corrispettivi oggetti connessi che permettono di monitorare sono:

- **Temperatura dell'aria** attraverso il termostato;

- **Pressione dello spazio** per mezzo del barometro;
- **Movimenti** mediante sensori di movimento;
- **Immagini** utilizzando fotocamere e videocamere;
- **Luminosità dello spazio** attraverso dei sensori specifici;
- **Umidità** tramite sensori di umidità;
- **Tensione e corrente elettrica** mediante rispettivamente voltmetro e amperometro.

Altri oggetti intelligenti capaci di fornire informazioni specifiche sono i cosiddetti dispositivi **wearable** (indossabili), quali braccialetti, fasce, occhiali e orologi connessi. Essi hanno la capacità di misurare la qualità e quantità del sonno, di tenere sotto controllo il battito cardiaco, di controllare il livello di disidratazione del corpo e così via. Il fine, come detto, è quello di perfezionare la qualità di vita.

1.4 Ambiti applicativi

Già oggi, moltissimi oggetti dei tipi più disparati, sono collegati in rete, rendendo l'IoT una realtà in rapidissima espansione, tanto che è difficile calcolare da ora quanti siano i dispositivi connessi. Secondo Gartner, un'agenzia di ricerca che ha sviluppato il tema dell'IoT negli ultimi anni, al momento sarebbero almeno 5 miliardi i dispositivi in grado di dialogare autonomamente con internet per poi passare, nel 2020, a circa 26 miliardi di oggetti collegati in tutto il mondo. Questo significa che ogni essere umano disporrà mediamente di 4 oggetti IoT. Numeri incredibili se pensiamo che fino a qualche decennio fa non eravamo neanche a conoscenza di internet e l'IoT appariva come qualcosa di inarrivabile e immaginario.
Dai numeri stilati da Gartner è evidente che il settore subirà una notevole espansione, con un giro di affari che si calcola attorno agli 80 miliardi di dollari a livello globale.

Gli ambiti applicativi dell'Internet of Things, sia per i consumatori finali, sia per le aziende e la manifattura, sono rappresentati da quei contesti nei quali ci sono "cose" che "parlano" generando nuove informazioni e possono essere classificati secondo 3 gradi di maturità:

- **Applicazioni consolidate**: coincidono con gli utilizzi più semplici e di immediata realizzazione. Si pensi per esempio alla videosorveglianza e alla

sicurezza nelle Smart Home finalizzata al controllo e alla gestione delle flotte aziendali, alla tracciabilità degli oggetti di valore così come al monitoraggio del traffico cittadino in ambito Smart City;

- **Applicazioni in fase sperimentale**: sono quelle che più si avvicinano al paradigma dell'IoT, come i contatori intelligenti (Smart Metering) per misurare i consumi, la sicurezza delle persone e la registrazione dei parametri di guida. Il loro sviluppo procede ancora lentamente ma ci si aspetta anche in tal senso un'impennata nei prossimi anni;
- **Applicazioni in fase embrionale**: sono quelle in cui l'IoT è stato soltanto immaginato, come nell'ambito energetico con la Smart Grid, una rete elettrica dotata di sensori intelligenti che raccolgono informazioni in tempo reale per ottimizzare la distribuzione di energia.

Sicuramente i settori dove l'IoT trova più riscontri positivi sono la domotica e l'industria in cui gli oggetti connessi vengono utilizzati quotidianamente sfruttando le reti casalinghe e/o aziendali. In base all'esperienza di tutti i giorni si può fare una classificazione degli esempi di utilizzo dell'Internet delle Cose:

1. **Smart Home e Smart Building** (case e palazzi intelligenti e connessi): Mentre le Smart Home si rivolgono a consumatori e fruitori finali dei servizi (per esempio regolare la temperatura della casa a distanza o utilizzare sensori di rilevamento per le persone in casa, quindi il settore della domotica), le Smart Building si rivolgono soprattutto al B2B (Business to Business), ovvero alla realizzazione ed ottimizzazione di palazzi e uffici, per dotarli di oggetti intelligenti che interagiscano con l'ambiente interno (ad esempio gestione della luce).
2. **Smart City** (Città intelligenti): Questo campo di applicazione si riferisce all'insieme di strategie di pianificazione urbanistica che migliorano la qualità di vita in città, cercando di soddisfare le esigenze ed i bisogni dei cittadini. Con l'IoT l'obiettivo, in questo senso, è quello di far relazionare le infrastrutture (oggetti) presenti per le strade con gli abitanti. In futuro è possibile che i semafori, ad esempio, siano dotati di rilevatori di traffico che si attivano in caso di code o di rallentamenti. Queste informazioni potrebbero essere accessibili ai navigatori delle auto che consiglierebbero percorsi alternativi deviando il traffico in strade meno battute. In questo modo ci sarebbe una riduzione degli ingorghi e dei tempi di percorrenza con conseguente risparmio di carburante e

città meno inquinate. In Svizzera c'è già un primo approccio di "**semaforo intelligente"** che diventa verde quando "vede" che una macchina è vicina all'incrocio e dall'altro lato non sta transitando nessuno.

3. **Smart Mobility**: Senza di essa non può esserci Smart City; sono tante le imprese che stanno pesantemente investendo in questo settore soprattutto per quanto riguarda le Smart Car (automobili intelligenti) che, solo in Italia, rappresentano un terzo delle auto. Anche nel mondo del trasporto ferroviario l'IoT offre grandissime opportunità di business; ne è un esempio concreto Trenitalia che per la gestione della manutenzione dei treni sfrutta le potenzialità di piattaforme IoT.
4. **Smart Manufacturing**: al giorno d'oggi è uno dei settori più maturi per l'utilizzo dell'IoT e unisce tematiche legate all'automazione con tematiche legate al mondo della robotica. Lo Smart Manufacturing si sovrappone anche con il mondo Industry 4.0, cioè con una nuova politica di sviluppo che estende l'introduzione del digitale nel campo industriale.
5. **Smart Agricolture**: A primo impatto potrebbe sembrare che non ci sia nulla di più distante tra il mondo dell'agricoltura e le tecnologie che prevedono una connessione a internet; ciò nonostante l'IoT sta entrando prepotentemente nei campi dando origine alla cosiddetta agricoltura di precisione allo scopo di aumentare in modo decisivo la qualità e quantità dei raccolti. Per i produttori questo significa poter decidere in tempo reale come distribuire l'acqua, i fitofarmaci o i fertilizzanti con un tempismo altrimenti impossibile.
6. **Smart Health**: È l'IoT applicato alla medicina: oggetti dotati di sensori, che comunicano tra loro e con gli esseri umani, scambiando informazioni, per aiutare a prevenire eventi come attacchi di cuore o malori o, ancora, aiutare lo staff medico a diagnosticare più rapidamente una malattia. Questo è possibile grazie all'uso di algoritmi predittivi e all'intelligenza artificiale, che processano immense quantità di dati. Smart Health sta per sanità "intelligente"; intelligente perché, attraverso i sensori installati nei dispositivi collegati ai pazienti (braccialetti, orologi), è possibile raccogliere dati sullo stato di salute delle persone curandole, anche a distanza, anticipando situazioni critiche che potrebbero presentarsi.

Il grafico che segue sintetizza i principali settori di utilizzo dell'IoT, tenendo conto anche delle ricerche su Google, dei Tweet e dei post su LinkedIn, nell'arco di un mese, contenenti il nome dell'applicazione.

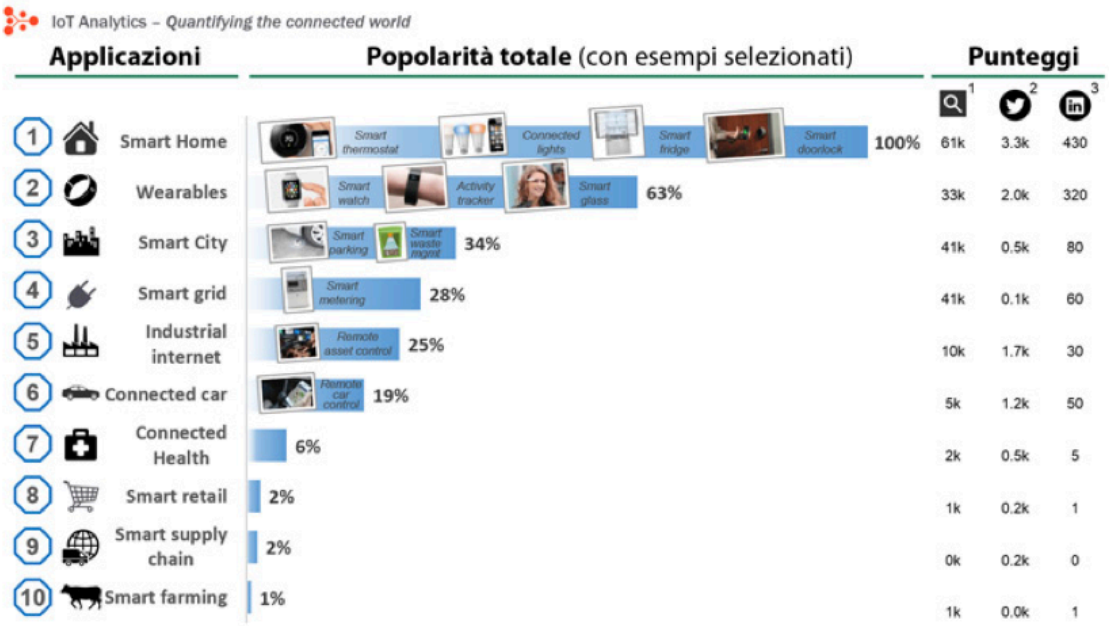


Figura 4: 1. Ricerche mensili su Google per l'applicazione - 2. Tweet mensili contenenti il nome dell'applicazione e #IoT - 3. Post mensili su LinkedIn che includono il nome dell'applicazione.

1.5 Privacy e sicurezza

La diffusione sempre più capillare degli oggetti collegati alla rete al di fuori del controllo dei proprietari, oltre al grande entusiasmo per gli enormi vantaggi introdotti, ha tuttavia innescato diverse polemiche e controversie. Gli esperti sono preoccupati per il **rispetto della privacy** in quanto, se gli oggetti di uso comune sono collegati in rete, è teoricamente possibile raccogliere un'infinità di dati sulle abitudini dei singoli in ogni ambito. Essendo l'IoT parte di ogni contesto (casa, imprese, stabilimenti produttivi, pubbliche amministrazioni, servizi pubblici, ospedali, etc..) il problema della sicurezza diventa di grandissima importanza.

Essere circondati da oggetti connessi alla rete significa anche, potenzialmente, essere costantemente spiati in casa propria e lasciare una via d'accesso a un numero indefinito di ambiti della nostra vita privata. Fino a qualche anno fa, se avessero detto che sarebbe

stato possibile rubare la password dell'account email o altri dati sensibili attraverso ad esempio un tostapane, non ci avrebbe creduto nessuno. Eppure questo scenario, che prima poteva sembrare surreale, oggi, nell'era dell'IoT, è realtà e anche la sicurezza di quei dispositivi che pensavamo "innocui" può diventare un problema.

La società di ricerca Gartner porta l'attenzione sui rischi e sulle vulnerabilità collegate alla diffusione dell'IoT denunciando che negli ultimi 3 anni, solo nel contesto produttivo, qualcosa come un'azienda su cinque ha subito almeno un attacco ai propri ambienti Internet of Things. Per i cyber criminali questo rappresenta un florido ambiente, ricco di dispositivi scarsamente protetti, in cui possono liberamente muoversi grazie a furti d'identità e infiltrazione nei sistemi IT aziendali.

Un gruppo di Hacker ben organizzato, oltre a rubare informazioni, potrebbe teoricamente bloccare il traffico automobilistico, uno stabilimento produttivo, i riscaldamenti nelle case o addirittura lasciare senza energia elettrica e assistenza i pazienti degli ospedali.

Ogni singolo device IoT che si aggiunge a una rete diventa un nuovo punto di attacco potenziale per cui, introducendo una connessione IP su internet, qualsiasi dispositivo deve essere protetto prima di essere fisicamente collegato al network e i sistemi di protezione devono essere aggiornati sistematicamente. Per garantire un ottimo livello di sicurezza ad ambienti, produzione e persone bisogna rispettare una serie di punti irrinunciabili:

1. **L'autenticazione dei device IoT**: Ogni nuovo dispositivo deve essere autorizzato e chiuque tenti di accedere alla rete deve essere sottoposto a un processo di autenticazione, con controllo degli accessi.
2. **Cifratura (encryption)**: La rete deve poter contare su un controllo sui dati condivisi dai device e dalle applicazioni e per proteggerli servono strumenti di cifratura.
3. **Update e aggiornamento di tutte le componenti software**: Ogni apparato industriale, macchinario, oggetto connesso dispone di una componente software sempre più importante. È opportuno prestare massima attenzione a tutti gli aggiornamenti, infatti se alcune parti non vengono aggiornate in modo puntuale, il rischio è quello di aprire varchi rendendo inutile il lavoro su altri ambiti.
4. **Archiviazione**: Anche la fase di memorizzazione dati sui dispositivi storage deve essere svolta con grande attenzione e i sistemi di archiviazione devono garantire un livello di protezione e sicurezza molto elevato.

5. **Analisi dei possibili attacchi**: I pirati informatici spesso sferrano attacchi pensati specificamente per compromettere determinate aziende o attività; è pertanto necessario studiare e progettare forme di protezione personalizzate e indirizzate a proteggere anche da precise minacce.
6. **Suddivisione della protezione in aree**: Per l'IoT la protezione dai rischi non è univoca e assoluta: una falla nella sicurezza di un device collegato a internet può danneggiare l'intera rete. Ecco perché suddividere quest'ultima in aree permette di minimizzare i rischi riducendo la superficie d'attacco.

I maggiori rischi riguardano i casi in cui l'IoT è utilizzato per controllare le attività fisiche, che si tratti dell'impianto di trattamento dell'acqua o addirittura di un peacemaker all'interno di un paziente, le conseguenze dovute a una violazione della sicurezza si estendono, oltre al rilascio non autorizzato delle informazioni, a veri e propri danni fisici che possono anche portare alla morte.
Tutti questi aspetti andranno presi in considerazione dalle aziende e dai privati per la realizzazione futura di dispositivi IoT.

CAPITOLO 2: PROTOCOLLO MQTT

2.1 Modelli e Protocolli di comunicazione

In generale, i dispositivi IoT per poter comunicare utilizzano modelli e protocolli di comunicazione diversi. Per quanto riguarda i **modelli**, esistono tre architetture diverse:

- **Comunicazione da dispositivo a dispositivo (M2M)**: È una delle prime forme di connettività e anche una delle più diffuse; due dispositivi comunicano tra di loro senza un server che fa da intermediario, utilizzando diversi tipi di reti, incluso le reti IP. Questo modello di comunicazione è comunemente usato in applicazioni come sistemi di home automation, che tipicamente usano piccoli pacchetti di dati per parlare. Il protocollo più diffuso in questo senso è il Bluetooth.

Figura 5: Rappresentazione modello di comunicazione da dispositivo a dispositivo

Un esempio concreto è la comunicazione tra personal device come gli Smart Watch connessi direttamente con lo Smartphone.

- **Comunicazione da dispositivo a Piattaforma (Device to Cloud)**: è quella che propriamente rientra nella categoria dell'IoT, dove i dispositivi sono direttamente connessi ad un Cloud Internet. Questo approccio si appoggia su reti di comunicazione esistenti come il WIFI o l'Ethernet per stabilire una connessione tra il dispositivo e la rete IP connessa al Cloud, permettendo anche di interrogare i device da remoto.

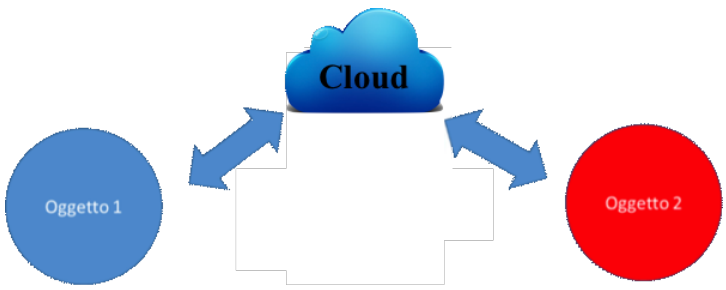


Figura 6: Rappresentazione modello di comunicazione da dispositivo a Cloud

- **Comunicazione da dispositivo a Gateway (Device to Gateway):** è il tipo di comunicazione più utilizzato in ambito IoT integrando le caratteristiche dei primi due. A differenza del modello Device to Cloud, in questo caso, il dispositivi IoT si connette al Cloud attraverso il gateway che funge da intermediario e ha tre compiti importanti:
    - ✓ **Normalizzazione dei dati**: associa ai dati ricevuti dal device, le informazioni o il formato che serve;
    - ✓ **Supporto per la comunicazione con il servizio Cloud**: è in grado di utilizzare diversi protocolli di comunicazione, sia in entrata (connessione con i dispositivi), sia in uscita (trasmissione dati al Cloud);
    - ✓ **Gestione della sicurezza**: il gateway funge da Firewall tra dispositivi e rete, gestendo le informazioni con diversi livelli di autorizzazione.

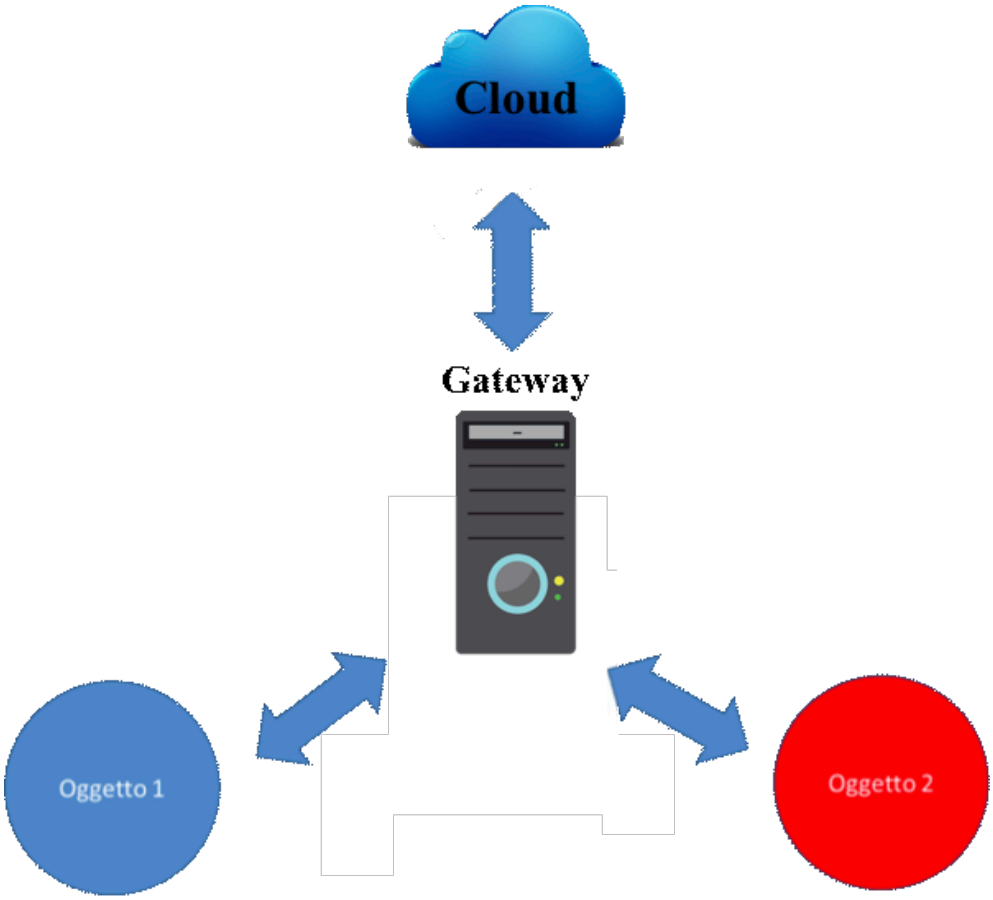


Figura 7: Rappresentazione modello di comunicazione da dispositivo a Gateway

Per quanto riguarda invece il discorso del protocollo di comunicazione che gli oggetti devono utilizzare per comunicare, sono due i paradigmi possibili:

- **Request/Response**: la comunicazione è di tipo sincrono, viene prima fatta una richiesta per ricevere le informazioni, rimanendo in attesa di una risposta. È particolarmente comune nelle architetture Client/Server;
- **Publish/Subscribe**: In questo caso la comunicazione è asincrona; la struttura è formata da una parte che si iscrive (subscriber) a certi topic/argomenti dai quali si vogliono ricevere informazioni, una parte che pubblica (publisher) nuovi dati per certi topic e una parte che gestisce le iscrizioni e le pubblicazioni occupandosi dello scambio di messaggi (dispatcher/broker). In questo modo le comunicazioni sono decisamente più ordinate perché se ci si iscrive ad un determinato topic, si ricevono aggiornamenti solo e soltanto inerenti allo stesso. A differenza del precedente paradigma, il mittente del messaggio (publisher) e il destinatario (subscriber) non sono a conoscenza dell'esistenza l'uno dell'altro, ma si limitano a "pubblicare" e ad "abbonarsi".

Il protocollo considerato standard per le applicazioni IoT è il protocollo MQTT del quale tratterà il prossimo capitolo. Esso utilizza come modello di comunicazione **Device to Gateway** e il paradigma **Publish/Subscribe**.

2.2 Storia e presentazione MQTT

Fin dagli albori, si è sempre cercato il protocollo più adatto, a livello applicativo, per far comunicare gli oggetti IoT. Uno dei protocolli più noti è certamente l'HTTP, HiperText Transfer Protocol, il quale è molto utilizzato per la trasmissione delle informazioni sul web e si basa su un modello di comunicazione Client/Server. Si tratta però di un protocollo adatto all'Internet of Things? Generalmente no: infatti esso è ritenuto dagli esperti "verboso", pesante, dal momento che prevede l'invio di messaggi di grandi dimensioni in quanto inviati in formato leggibile dagli utenti. Per i dispositivi IoT le dimensioni di payload costituiscono spesso un vincolo, poiché generalmente la loro comunicazione è caratterizzata dall'emissione di un numero limitato di bit. Inoltre, un

altro suo limite, è che non rende possibile stabilire una comunicazione asincrona che è un'esigenza classica dell'IoT.

Un protocollo che, invece, nel corso del tempo si è rivelato particolarmente adatto per la trasmissione dei dati prodotti dai dispositivi IoT è sicuramente l'MQTT. Esso è stato inventato da Andy Stanford-Clark e Arlen Nipper, quando lavoravano rispettivamente per l'IBM e per la ARCOM, nel 1999. L'acronimo non avrebbe un vero e proprio significato, normalmente è identificato come MQ Telemetry Transport, dove MQ starebbe per un prodotto sviluppato da IBM per il quale fu inventato questo protocollo. Spesso si associa MQ a Message Queuing, ma impropriamente poiché non c'è una vera coda di messaggi. MQTT nasce per:

- gestire le connessioni M2M, cioè per abilitare la comunicazione tra macchine in maniera estremamente efficiente;
- avere a disposizione un protocollo che riducesse al minimo il consumo delle batterie e il minimo utilizzo di banda nelle comunicazioni tra device;
- creare un protocollo semplice da implementare e con un certo grado di affidabilità;
- l'esigenza di minimizzare i traffico sulle reti e di richiedere poche risorse ai dispositivi per la sua gestione;
- sostituire il modello Client/Server di HTTP, considerato troppo pesante per l'IoT.

Inoltre il protocollo MQTT è open source. Questo insieme di caratteristiche lo rendono ideale per tutti quegli ambienti in cui le risorse disponibili e la larghezza di banda della rete sono limitate, oppure quando si ha a che fare con reti non perfette in termini di stabilità della connessione, soggette a molteplici interruzioni.

Dall'ottobre 2014, il protocollo MQTT è stato standardizzato dall'Organization for the Advancement of Structured Information Standard (OASIS)[4], e attualmente è disponibile la versione 5.0.

[4] **OASIS**: è un'organizzazione no profit globale che lavora allo sviluppo, alla promozione e alla regolarizzazione di standard aperti per la sicurezza, internet delle cose, energia e altre aree.

2.3 Descrizione e funzionamento MQTT

MQTT, come meccanismo di comunicazione, non utilizza il Client/Server, ma bensì segue un paradigma di pubblicazione e sottoscrizione classico, definito publish/subscribe. In questo modello, a differenza del Client/Server dove chi invia dati comunica direttamente con un endpoint (il server), si possono mantenere separati i client tra loro in modo che un certo oggetto (chiamato publisher) possa inviare un determinato messaggio e altri oggetti (chiamati subscriber) possono riceverlo. I client possono ignorare l'esistenza l'uno dell'altro solo grazie all'esistenza di un nuovo attore: il broker. Esso filtra e distribuisce le comunicazioni tra publisher e subscriber permettendo di configurare una messaggistica uno a molti.
Quindi in MQTT ci sono due protagonisti:

- **Il Client**: per Client si intendono i publisher, i subscriber o entrambi, quindi sia chi invia messaggi, sia chi li riceve. Un Client MQTT è qualsiasi dispositivo che sia un microcontroller o un server che ha una libreria MQTT in esecuzione ed è connesso ad un broker su qualsiasi tipo di rete. Le librerie Client MQTT sono disponibili per una vasta gamma di linguaggi di programmazione: Arduino, C, C++, Python, etc.
- **Il Broker**: è il cuore del protocollo Publish/Subscribe in quanto si occupa della gestione dei flussi di dati. Il broker è il primo responsabile per la ricezione di tutti i messaggi, li filtra e decide, in base alle iscrizioni, a chi destinare i messaggi. Un'altra responsabilità del broker è l'autenticazione e l'autorizzazione dei client.

La connessione MQTT è sempre tra un client e il broker, un client non è mai connesso ad un altro client direttamente. Per iniziare un collegamento, il Client invia un messaggio CONNECT al Broker che risponde con un messaggio CONNACK e con un codice di stato. Una volta stabilita la connessione, il Broker la mantiene aperta fino a quando il Client non si disconnette o la connessione di interrompe.

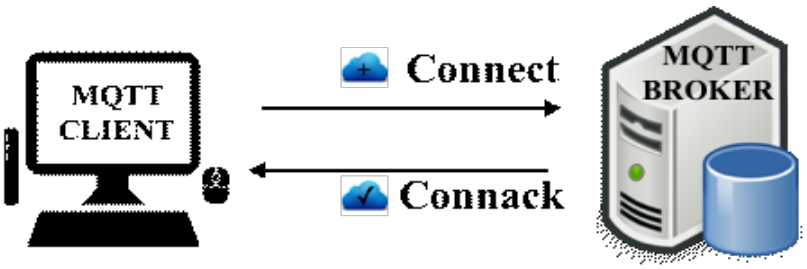
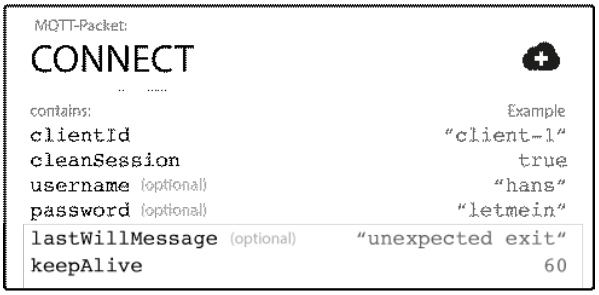


Figura 8: Esempio di connessione MQTT

Il messaggio di connessione (CONNECT) inviato dal Client al Broker è il seguente:



ClientId: è l'identificativo del Client al Broker e deve essere univoco.

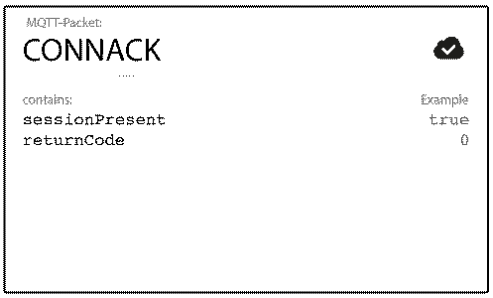
CleanSession: è un flag che indica al Broker se il Client vuole stabilire una sessione persistente (CleanSession=false) o meno (CleanSession=true). Nel primo caso il Broker archivia tutte le sottoscrizioni per il Client, altrimenti non memorizza nulla.

Username/Password: i dati per l'autenticazione e autorizzazione del Client. servono per aumentare la sicurezza e l'affidabilità della connessione.

LastWillMessage: È il messaggio "di testamento", permette di notificare gli altri Client quando avviene una disconnessione non voluta.

Keep Alive: è un intervallo di tempo in secondi che il Client specifica e comunica al Broker quando viene stabilita la connessione. Corrisponde al periodo di tempo più lungo in cui Client e Broker possono stare senza scambiarsi messaggi. Il Client si impegna a inviare al Broker messaggi di richieste PING regolari e quest ultimo risponde con un PING. Questo metodo consente ad entrambe le parti di determinare se l'altra è ancora disponibile.

Allo stesso modo il Broker risponderà con un messaggio di CONNACK:



SessionPresent: è un flag che indica se il Broker ha già una sessione persistente per quel Client dovuta a interazioni precedenti.

ReturnCode: è un codice di ritorno che indica al Client se il tentativo di connessione ha avuto esito positivo o meno. I codici di ritorno possibili, con i loro significati, sono:

Codice di ritorno	Risposta codice di ritorno
0	Connessione accettata
1	Connessione rifiutata, versione del protocollo inaccettabile
2	Connessione rifiutata, identificatore rifiutato
3	Connessione rifiutata, server non disponibile
4	Connessione rifiutata, nome utente o password errati
5	Connessione rifiutata, non autorizzata

Una volta che il Client MQTT si è connesso al Broker, può iniziare a pubblicare messaggi. È qui che entra in gioco un elemento fondamentale che distingue MQTT da altri protocolli: il **TOPIC (ARGOMENTO)** che è un canale tematico a cui ogni destinatario può sottoscriversi (da qui il termine Subscriber) per ricevere informazioni su quell'argomento specifico. Allo stesso tempo, quando il mittente pubblica qualcosa, deve farlo su un determinato topic sul Broker che fa da intermediario distribuendo il messaggio solo ai destinatari sottoscritti all'argomento.
Un altro aspetto tecnologicamente rilevante è che per ciascuna connessione client/Broker può essere specificato un diverso livello di qualità del servizio (QoS):

1. **At most once**: il messaggio viene inviato una sola volta senza conferma di ricezione.
2. **At least once**: il messaggio viene inviato più volte finché non si ottiene una conferma di ricezione.
3. **Exactly once**: il messaggio viene inviato una e una sola volta con conferma di ricezione.

Quindi, quando un Client pubblica un messaggio, il Broker lo riceve e lo spedisce alle parti interessate (che si sono iscritte a quel determinato topic) con un collegamento di questo tipo:

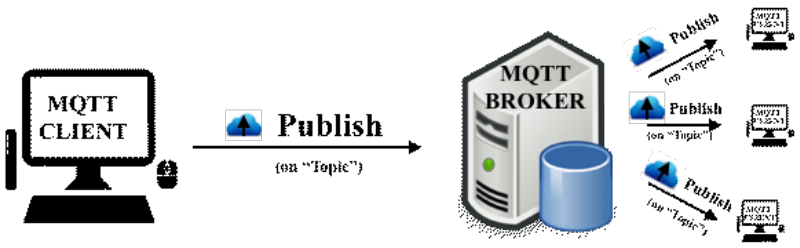
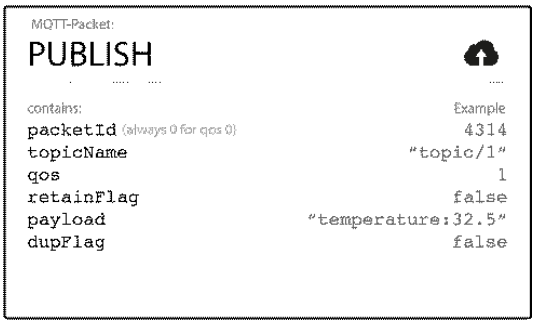


Figura 9: Esempio di pubblicazione di un messaggio (sempre su un "topic" specifico).

Il messaggio di publish che il Client MQTT, connesso al Broker, invierà sarà del tipo:



PacketId: è un identificatore unico tra Client e Broker per individuare il messaggio.
TopicName: è una semplice stringa che indica il nome dell'argomento strutturato gerarchicamente con slash usati come delimitatori. Ad esempio "casa/soggiorno/tempertura".

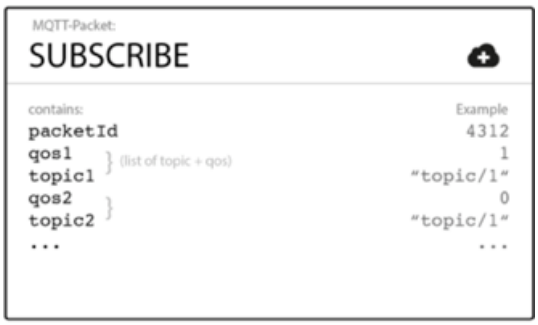
QoS: è un numero che indica la qualità del servizio e può essere: 0 - "At most once", 1- "At least once", 2 - "Exactly once".

RetainFlag: questo flag indica se il messaggio deve essere salvato dal Broker per questo topic come ultimo valore noto; in questo modo quando un Client si iscrive a quel topic riceve subito l'ultimo messaggio disponibile.

Payload: è proprio il contenuto del messaggio che si vuole inviare. È una scelta del mittente decidere se il dato inviato sarà in textual data, XML, o JSON.

DupFlag: indica che questo messaggio è un duplicato perché per quello originale non è stata ricevuta conferma.

Ovviamente, cosi come un Client pubblica un messaggio per un certo argomento con il messaggio di publish, lo stesso può iscriversi a un determinato topic, tramite il comando subscribe che è il seguente:



PacketId: è un identificativo univoco per individuare il messaggio.

Lista di sottoscrizioni: il messaggio può contenere un numero arbitrario di sottoscrizioni per un client, ognuna delle quali è una coppia formata da topic e QoS.

Per cui, come detto, è il Broker che gestisce il flusso dei dati; uno schema riassuntivo è riportato in figura 10:

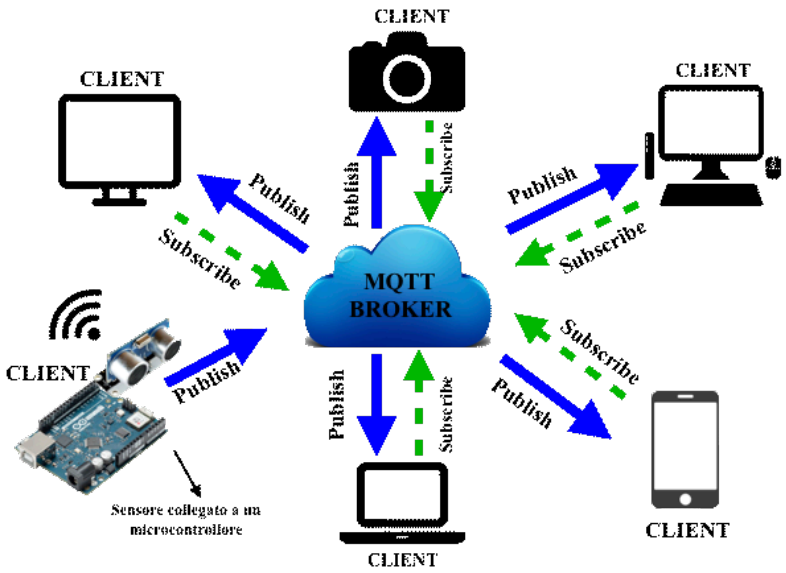


Figura 10: Il sensore, nel momento in cui pubblica un dato, su un determinato topic, al Broker, l'informazione viene pubblicata (publish) dal Broker stesso a tutti i Client che si erano iscritti all'argomento (subscribe).

Tutta l'infrastruttura rappresentata mostra il principale vantaggio del protocollo MQTT, ovvero il disaccoppiamento tra chi produce il dato e chi lo deve utilizzare:

- **Disaccoppiamento spaziale**: chi genera il dato non deve essere direttamente connesso con chi lo usa;
- **Disaccoppiamento temporale**: chi pubblica il dato non deve essere attivo nello stesso momento di chi ne usufruisce;
- **Disaccoppiamento di sincronizzazione**: la produzione del dato non deve essere sincrona con il suo utilizzo.

Per questi motivi si capisce perché, il protocollo MQTT e in generale il paradigma publish/subscribe, è diventato lo standard per l'IoT: i dispositivi più semplici, spesso alimentati a batteria, possono attivarsi ogni tot secondi, inviare i propri dati al Broker, e tornare allo stato di risparmio energetico. Sarà il Broker, sempre attivo, a rendere tali dati disponibili ai vari utilizzatori.

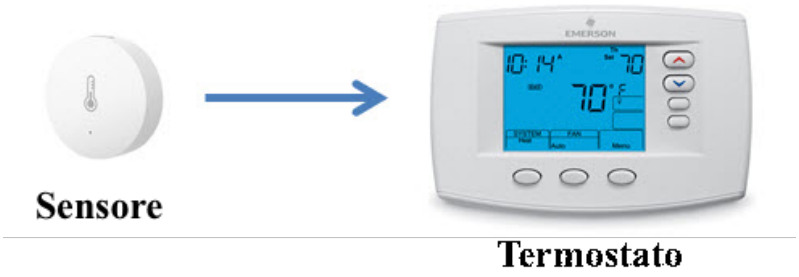
MQTT supporta, infine, anche un certo grado di sicurezza, considerato che con un suo pacchetto è possibile passare direttamente nomi utenti e password. Se si vogliono cifrare i dati trasmessi, tra i metodi suggeriti, ci sono invece l'affiancamento di SSL[5], il che in ogni caso aggiunge pesantezza al messaggio, oppure l'uso di applicazioni di criptatura a monte e a valle della trasmissione.

2.4 Esempio pratico di utilizzo del protocollo

In questo esempio saranno utilizzati un **sensore** che misurerà la temperatura di un ambiente e l'utilizzatore, ovvero il **termostato** che mostra il risultato della misurazione fatta. L'obiettivo è vedere come, attraverso il protocollo MQTT, avviene la trasmissione delle informazioni tra i due oggetti connessi in rete.

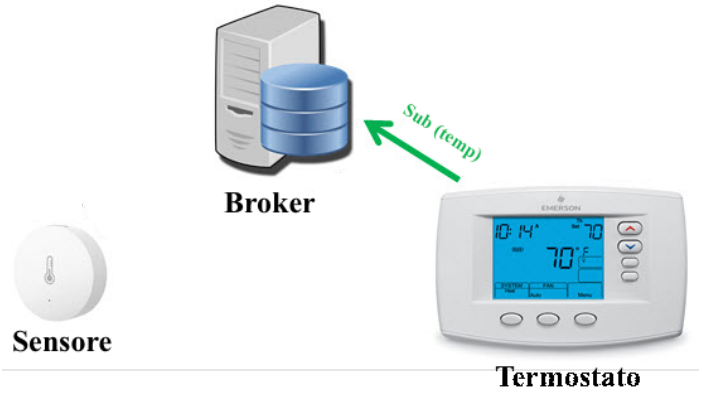
[5] **SSL**: Secure Sockets Layer è un protocollo per consentire la trasmissione di informazioni in modo **sicuro** e **protetto**. Le applicazioni che usano i certificati SSL sono in grado di gestire l'invio e la ricezione di chiavi di protezione e di criptare/decriptare i dati trasmetti usando le chiavi stesse.

In una comunicazione tradizionale, la sorgente del dato (nell'esempio un sensore di temperatura) invia il dato **direttamente** all'utilizzatore finale (il termostato), quindi si tratta di una comunicazione M2M.

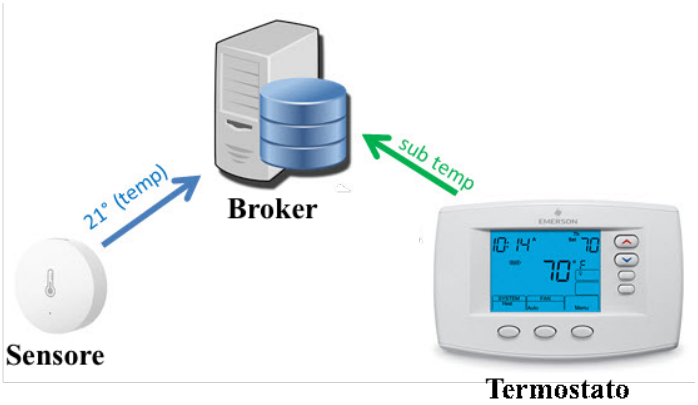


In una comunicazione basata su MQTT, invece, il sensore invia il dato a un intermediario centrale, detto **Broker**, specificandone il topic/argomento; il Broker poi inoltra l'informazione appena ricevuta a tutti gli altri dispositivi che precedentemente si sono iscritti a quel determinato topic.

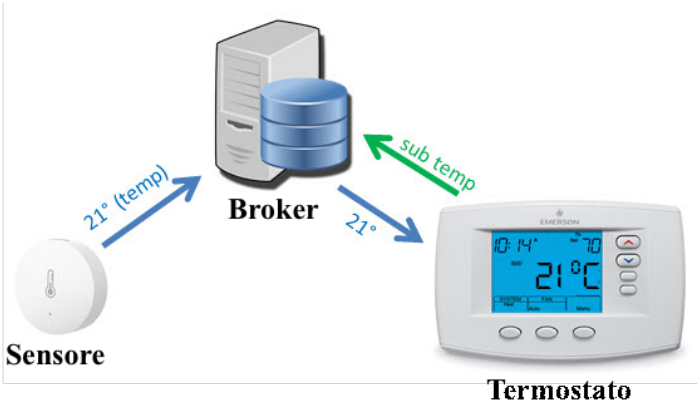
1. In fase di sottoscrizione l'utilizzatore può specificare anche più di un argomento dal quale vuole ricevere i dati. Nell'esempio preso in esame l'utilizzatore (termostato) si iscriverà al topic "**temp**" per ricevere messaggi inerenti alla temperatura:



2. Il sensore, una volta rilevato il dato della temperatura ambientale (21°), lo pubblica sul Broker indicandone il topic "**temp**". Il messaggio pubblicato sarà "21° (temp)":



3. A questo punto, essendo il termostato iscritto al topic "**temp**", il Broker invia il dato sulla temperatura ricevuto dal sensore: 21°, il quale apparirà sulla schermata dell'utilizzatore:



CAPITOLO 3: ALTAIR SMARTCORE

3.1 Descrizione e funzionamento piattaforma

Altair SmartCore è una piattaforma cloud-native, specializzata in IoT, che mette a disposizione un set integrato di servizi e funzionalità per connettere le cose al mondo digitale. È l'ambiente ideale per tutti gli sviluppatori che vogliono eseguire i propri progetti IoT più velocemente, offrendo un ambiente facile da usare, affidabile e altamente scalabile.

La piattaforma supporta il protocollo MQTT che consente una rapida integrazione e comunicazione dei dati tra tanti tipi di dispositivi quali sensori, Arduino, Raspberry Pi e macchine, fondamentalmente qualsiasi hardware che fornisce connettività web. Altair permette di collezionare tutti i dati inviati dai dispositivi IoT che vengono memorizzati nella Big Data Base NoSQL per poter essere usati. Ogni utente iscritto può gestire, controllare, attivare o disattivare i propri dispositivi registrati, anche da remoto, in maniera semplice ed efficace. Infatti, il pannello di controllo di Altair SmartCore è molto intuitivo e facile da comprendere grazie all'organizzazione e all'ordine in cui sono disposte le varie voci che interessano allo sviluppatore.

Un esempio della potenza della piattaforma è rappresentato dal fatto che, se nel flusso di dati inviato da un dispositivo risulta un valore critico e non desiderato, Altair invia immediatamente un sms e/o una mail allo sviluppatore per avvertirlo. Lo stesso, collegandosi al pannello di controllo del sito, può immediatamente intervenire disabilitando il dispositivo per evitare inconvenienti. Possiamo immaginare questa situazione nel caso in cui un sensore di movimento, registrato sulla piattaforma, rilevi la presenza di un ladro nella propria abitazione; in tal caso il proprietario ne viene subito a conoscenza e può agire di conseguenza. Questo "sistema di allarme" può essere implementato grazie alla sezione dedicata ai listeners (che sarà successivamente esaminata in questo progetto di tesi) in cui Altair consente di creare regole personalizzate per la corretta gestione quando si verifica un evento legato al dispositivo selezionato.

Lo stesso listener può essere utilizzato anche solo per aggiornare lo sviluppatore, sempre tramite mail/sms/tweet, sui dati che vengono registrati nel flusso di dati.

Nel prossimo paragrafo sarà presentato il pannello di controllo di Altair, con le relative descrizioni dei protagonisti che entrano in gioco su questa piattaforma.

3.2 Pannello di controllo

Una volta effettuato l'accesso a Altair SmartCore, l'utente viene reindirizzato al pannello di controllo: il principale punto di accesso per la gestione di dati e entità. Esso presenta due aree principali:

- Menù a sinistra per la gestione delle entità;
- Menù principale per l'amministrazione dei **dispositivi**, dei **flussi dati**, degli **allarmi** e degli **ascoltatori**, per la visualizzazione rapida dei limiti e la geolocalizzazione dei dispositivi registrati.

Per quanto riguarda il menù di gestione, particolarmente utili sono le sezioni dedicate alle **REGOLE** che permettono all'utente di stabilire la propria logica aziendale, ai **DATI** in cui sono raggruppate tutte le operazioni relative sugli stessi e alle **IMPOSTAZIONI** dove si possono gestire le connessioni app, utenti e apikeys.

Cos'è un Apikey? È un codice adoperato dai programmi di un computer/applicazione per tenere traccia e controllare le modalità d'uso dell'API, per evitarne un utilizzo improprio e scorretto. L'Apikey agisce sia come identificatore univoco, sia come token segreto per l'autenticazione.

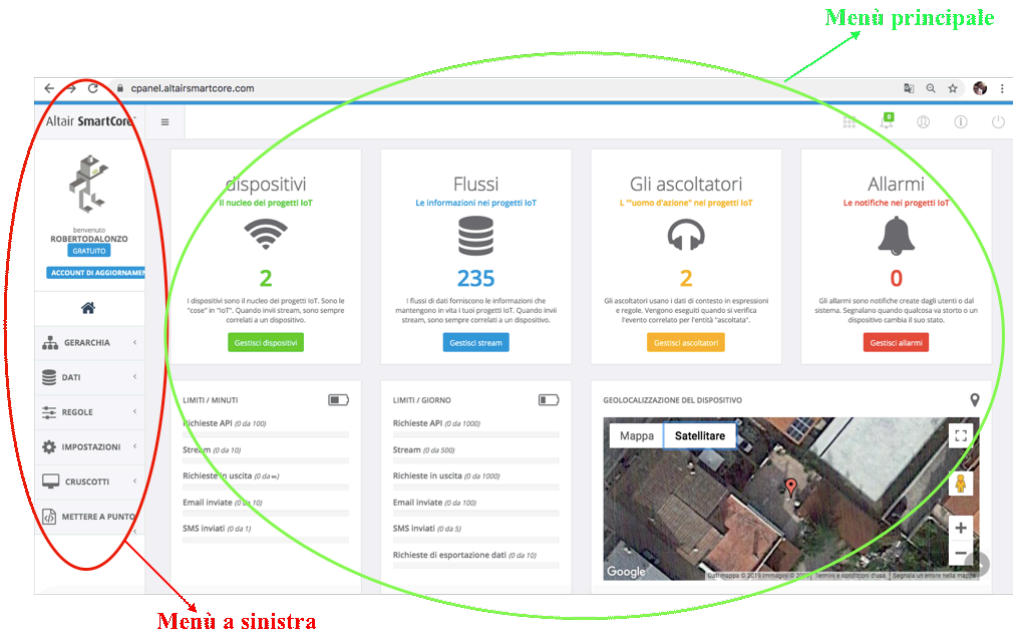


Figura 11: Pannello di controllo di Altair SmartCore

3.2.1 DISPOSITIVI

Nella Home, la prima sezione che si presenta davanti all'utente è quella dei dispositivi: il nucleo dei progetti IoT. Essi non sono altro che le "cose", connesse a internet, appartenenti all'Internet of Things. Quando si inviano stream di dati, essi sono sempre correlati a un dispositivo il cui nome/id deve essere univoco. Altair, nella sua prova gratuita, permette di registrare due device per iniziare a prendere dimestichezza con la piattaforma; ovviamente passando alla versione a pagamento il numero di device che à consentito registrare è decisamente più elevato.
Cliccando su "Nuovo dispositivo" si può iniziare a registrare l'oggetto riempendo tutti i campi richiesti:

- Nome: inserendo il nome verrà di conseguenza generato l'id del device;
- Descrizione (facoltativo);
- Tipo: Arduino/Smartphone/Computer, Router, Robot, altro;
- Sensore (facoltativo);
- Frequenza flusso: Intervallo di tempo in cui un flusso dati deve essere ricevuto dal dispositivo;
- Frequenza stato: Intervallo di tempo in cui un flusso di stato deve essere ricevuto dal dispositivo;
- Geolocalizzazione: Serve per localizzare il dispositivo attraverso "Latitudine" e "Longitudine";
- Abilitazione: Si può decidere se disabilitare il dispositivo o meno;
- MQTT configuration: È possibile scegliere se configurare o meno l'MQTT, inserendo nome, password e aggiungendo topic a cui l'oggetto è iscritto: il nome utente viene creato seguendo il modello [nome-inserito].[id-device], gli argomenti vengono nominati seguendo il modello: [id-device]/[nome-topic-immesso]. L'Host messo a disposizione da altair è mqttbroker.altairsmartcore.com. [della configurazione MQTT si parlerà dettagliatamente del paragrafo **4.1**]

Una volta registrato il device sulla piattaforma, è possibile mostrare tutte le sue informazioni cliccando sulla voce "Mostra dispositivo":

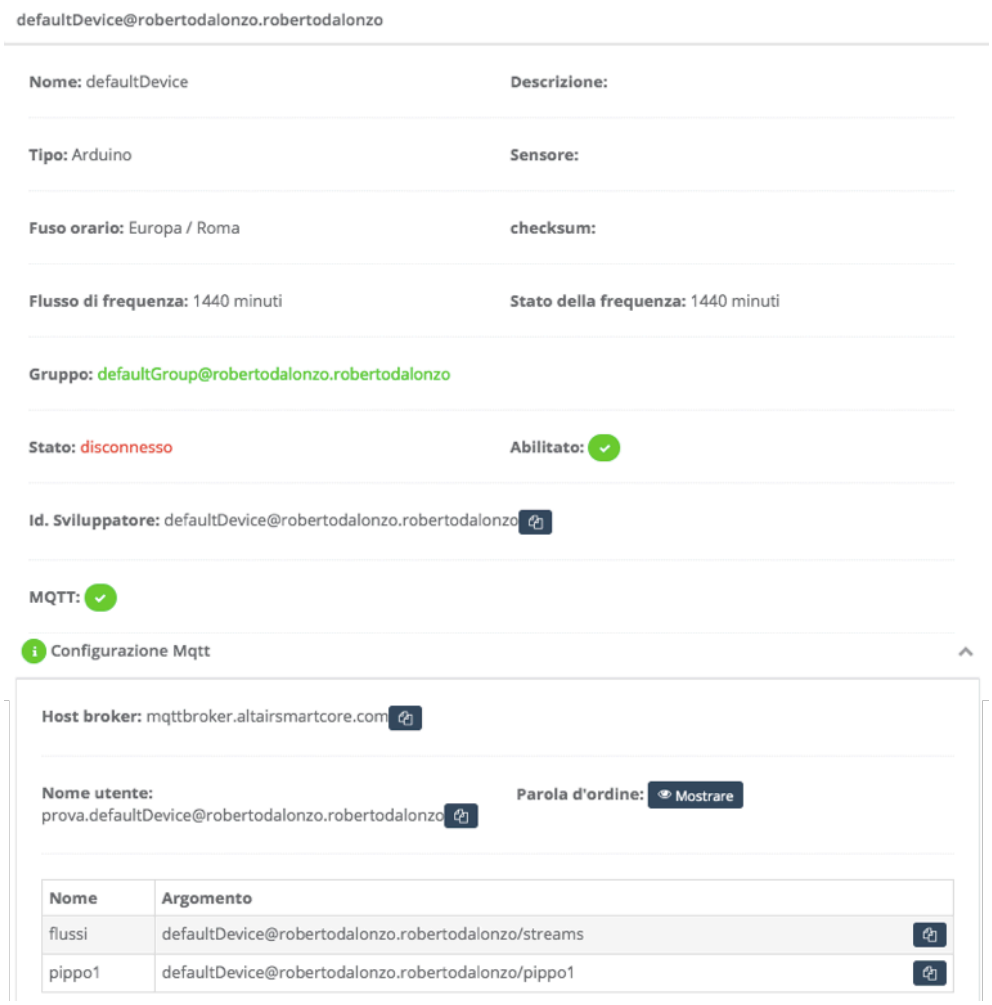


Figura 12: Informazioni del dispositivo di default registrato su Altair.

3.2.2 FLUSSI DI DATI

Dopo aver visto la parte dedicata ai dispositivi, nel pannello di controllo di Altair, troviamo la sezione dedicata ai data streams, ovvero al flusso dei dati. Essi forniscono le informazioni che mantengono in vita i progetti IoT: infatti mostrano i dati che vengono inviati da un certo device registrato nella infrastruttura Internet of Things.

Ciò che viene visualizzato è l'elenco degli streams, attraverso una tabella divisa in 3 parti: data e orario trasmissione, dispositivo che manda il messaggio, dati inviati.

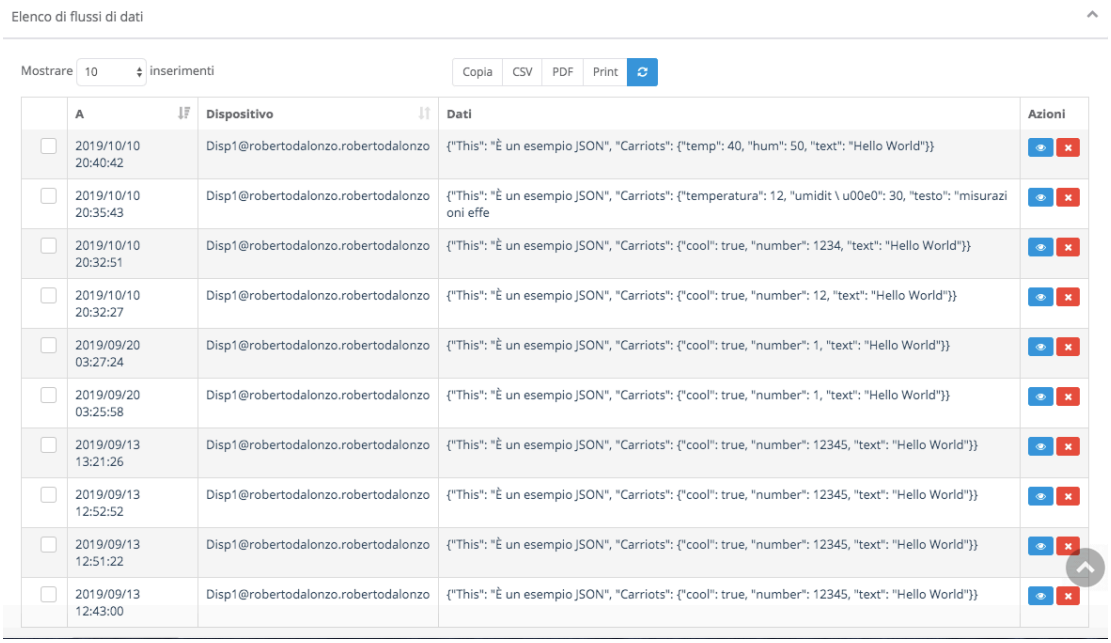


Figura 13: Elenco flussi di dati per dispositivi IoT registrati sulla piattaforma

3.2.3 ASCOLTATORI

In questa sezione troviamo l'elenco dei listeners (ascoltatori) che sono considerati "gli uomini azione" nei progetti IoT. Un ascoltatore "attende" che si verifichi un evento su un determinato dispositivo registrato e ne valuta il contenuto. A questo punto, una volta esaminata l'informazione ricevuta, agisce a seconda di come viene programmato. Ma cosa può fare? Ad esempio può mandare una mail d'allarme allo sviluppatore o a qualcun altro aggiornandolo sul contenuto del data stream oppure inviare un tweet con l'account registrato, o ancora inviare un sms o un messaggio ad un client MQTT, e così via.

Dopo aver creato l'ascoltatore, assegnandogli un nome per l'identificazione univoca, e averlo abilitato, si completa la sua registrazione progettando il codice che definisce la logica del listener, basata su regole create dall'utente. Questo viene fatto grazie allo strumento di flusso di Altair: un visual designer che consente all'utente di aggiungere nodi logici aziendali predefiniti. È proprio questa logica che gli permette di assumere un ruolo attivo comunicando con l'utente attraverso email/sms/messaggi MQTT/richieste a URL.

Vediamo un esempio per capire bene il funzionamento e per comprendere come si progetta il codice logico.

Supponiamo di avere un lettore di schede RFID chiamato rfidreader@smartcore. Ogni lettura invierà all'utente un flusso di dati con il tag RFID.
Il flusso di dati conterrà le seguenti informazioni:

```
{
  "tag" : 12345
}
```

Vogliamo inviare una email all'indirizzo support@smartcore.com quando il tag RFID letto non è 12345 e in caso contrario inviare un segnale di porta aperta (in questo esempio supponiamo che il segnale di porta aperta sia una semplice richiesta all'URL http://contro.entrance:9090/openDoor?token=123456789).
Iniziamo con la creazione dell'ascoltatore che è divisa in 2 passaggi:

1. **DATI GENERALI**

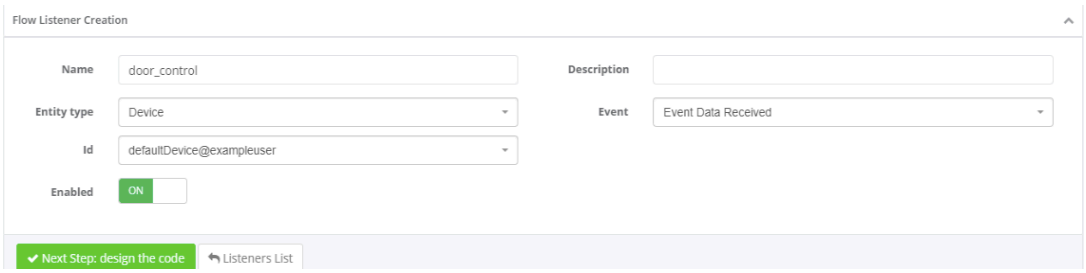
   Nome: door_control;

   Tipo di entità: dispositivo;

   Id entità: defaultDevice@example.example;

   Evento da ascoltare: "Dati evento ricevuti";

   Abilitato: SI;



2. **PROGETTAZIONE LOGICA AZIENDALE CON NODI PREDEFINITI**

   Nell'editor Flow dovremo aggiungere e configurare i seguenti nodi:

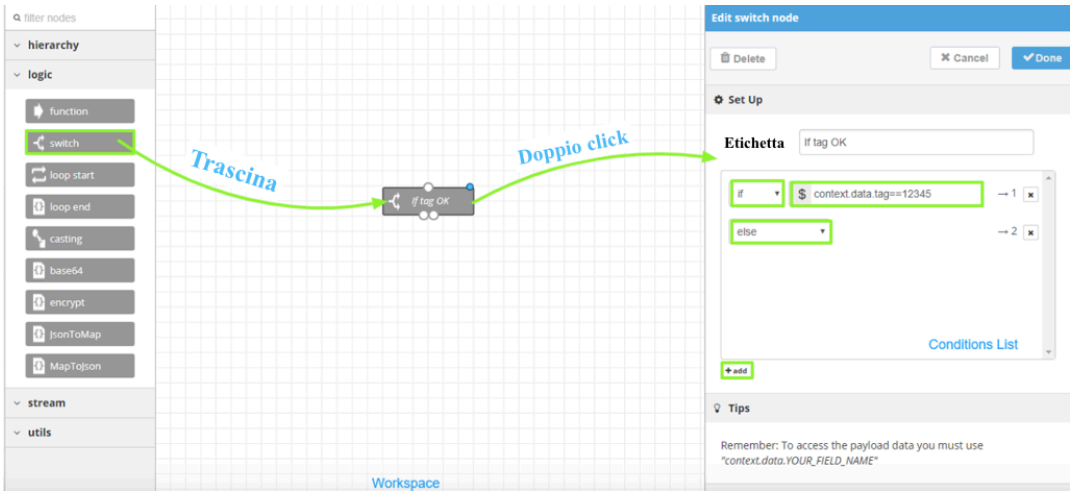
- **Nodo Switch:** trasciniamo nell'area di lavoro dell'editor un nodo "switch" e, facendo doppio clic su di esso, andiamo a configurarlo con i valori:

  Etichetta: Nome descrittivo del nodo nella vista dell'editor. Ad esempio "se tag OK";

  Condizione if: selezionare "if" e compilare il campo con il codice *context.data.tag==1234*;

  Condizione else: fare clic sul pulsante "+ add" e selezionare "altro".

Fare clic sul pulsante "Fine" per applicare le modifiche. Come risultato di queste azioni vedremo nell'area di lavoro dell'editor il nostro nodo appena creato con 2 porte di output. Da sinistra a destra, le porte corrispondono alle condizioni aggiunte nel primo passaggio dall'alto verso il basso, ovvero prima l'**if** e poi l'**else**.



- **Nodo Basic HTTP**: trasciniamo nell'area di lavoro dell'editor un nodo "basic HTTP" e, facendo doppio clic, lo andiamo a configurare con i valori:

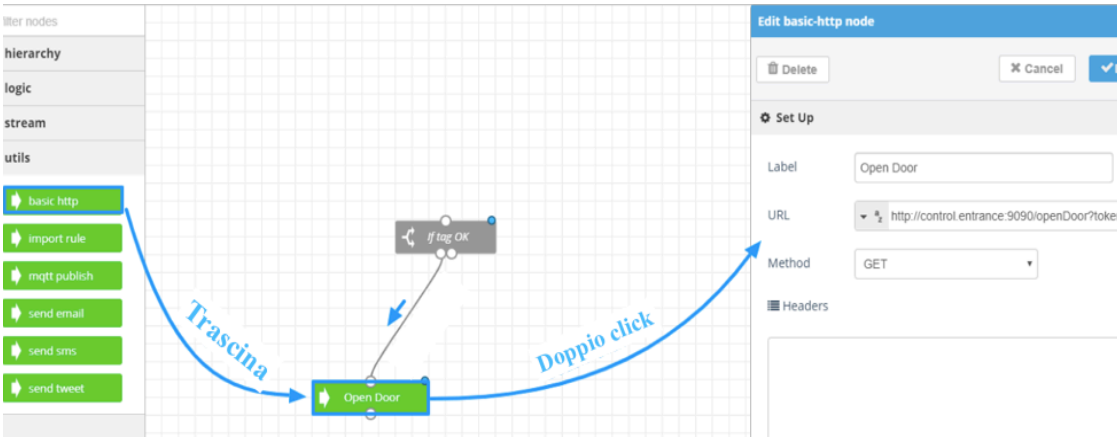
  Etichetta: Nome descrittivo del nodo nella vita dell'editor. Ad esempio "Open Door";

  URL: Digitiamo l'URL a cui vogliamo fare la richiesta: "http://contro.entrance:9090/openDoor?Token=123456789";

  Metodo: selezionare il metodo "GET".

  Fare clic sul pulsante "Fine" per applicare le modifiche.

  A questo punto colleghiamo la prima porta del **nodo switch** con l'ingresso del nodo **Basic HTTP**.



- **Nodo Invia email**: trasciniamo nell'area di lavoro dell'editor un nodo "invia email" e, facendo doppio clic sul nodo, lo configuriamo con i valori:

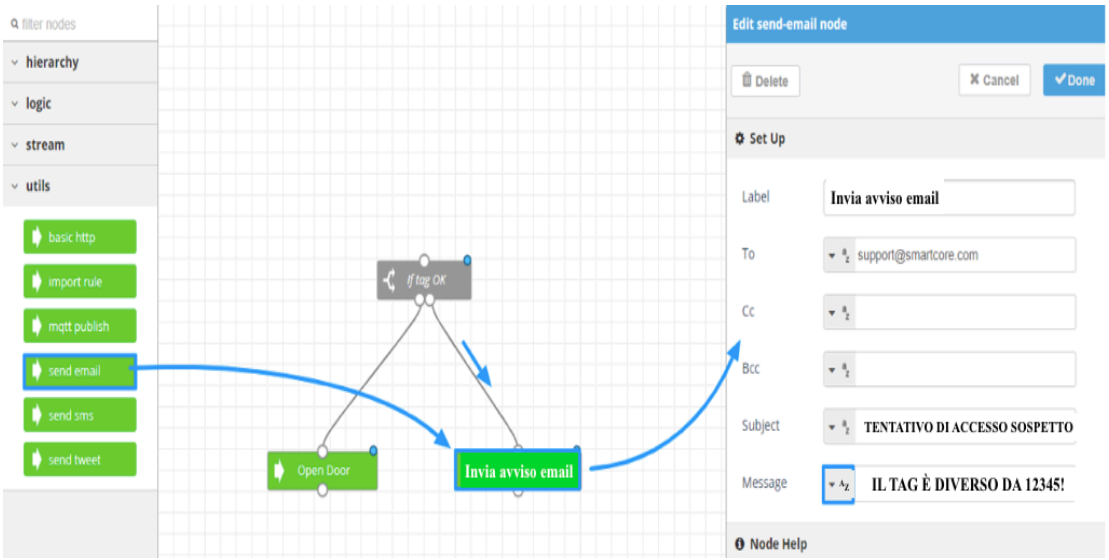
  Etichetta: Nome descrittivo del nodo nella vista dell'editor. Ad esempio "Invia avviso email";

  A: in questo campo mettiamo l'indirizzo email che riceverà l'avviso; ad esempio *support@smartcore.com*;

  Oggetto: digitiamo ad esempio "TENTATIVO DI ACCESSO SOSPETTO";

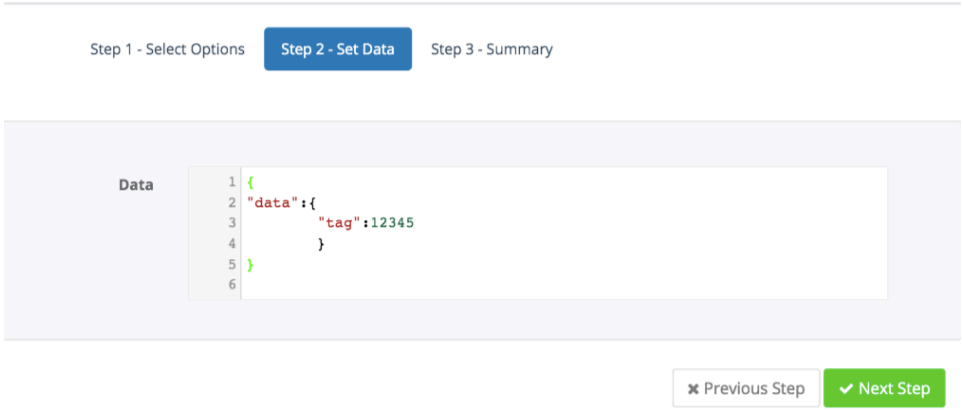
  Messaggio: Nel menù a discesa possiamo selezionare il tipo di messaggio tra stringa, variabile, payload. Ad esempio selezioniamo "string" e riempiamo il campo messaggio con "il tag è diverso da 12345!"

  Fare clic sul pulsante "Fine" per apportare le modifiche e connettere la seconda porta del nodo "**Switch**" con il nodo "**Invia email**" come in figura:



A questo punto salviamo il nostro ascoltatore e testiamo il suo funzionamento.

Per questo esempio sfruttiamo la procedura guidata di streaming (fornita dalla piattaforma stessa nella sezione Streams) per inviare il dato. Selezioniamo il dispositivo registrato rfidreader@smartcore e inviamo il seguente payload:



Procediamo con l'invio del messaggio e, effettivamente avviene la richiesta all'URL inserito. Analogamente, se ripetiamo lo stesso processo modificando il payload, quindi mettendo un numero diverso da 12345 al campo "tag", riceviamo l'email "TENTATIVO DI ACCESSO SOSPETTO " all'indirizzo *support@smartcore.com*.

3.2.4 ALLARMI

In questa sezione sono elencati gli eventuali allarmi, ovvero le notifiche che segnalano, direttamente sul pannello di controllo Altair, quando qualcosa va storto o un dispositivo cambia il suo stato. L'utente può crearne di nuovi, scegliendone nome, descrizione, entità alla quale si riferiscono e gravità, per ricevere un avviso personalizzato riguardo i propri progetti IoT.

Solitamente, però, gli allarmi "si autocreano", cioè se c'è qualcosa che non va quando si invia un flusso dati, viene subito notificato allo sviluppatore, specialmente quando il listener non riesce ad attivarsi dopo aver esaminato l'informazione ricevuta (in tal caso l'utente va subito a controllare il codice dell'ascoltatore che potrebbe presentare degli errori risalenti alla fase di implementazione).

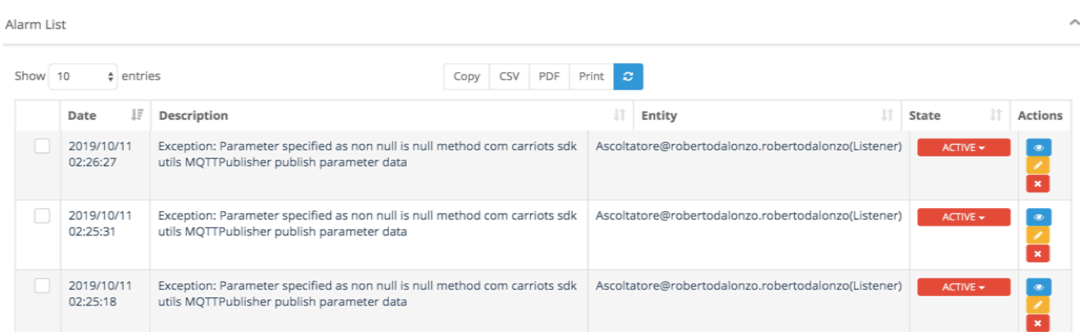


Figura 14: Lista allarmi su Altair SmartCore

In Figura 14 si può notare che la tabella, in cui sono elencati gli allarmi, è divisa in quattro colonne: data e ora attivazione notifica, descrizione dell'allarme, entità alla quale si riferisce e lo stato (che può essere attivo o disabilitato).
I tre allarmi della figura si riferiscono al fatto che l'ascoltatore è programmato per attivarsi in base ai valori numerici riportati nel campo "number" del flusso dati; ma quest'ultimo non presenta, in quei casi, alcun campo "number" per cui l'allarme notifica allo sviluppatore di non aver trovato, nel data stream, l'informazione per la quale è stato progettato.

3.2.5 LIMITI

L'ultima sezione visibile direttamente dal pannello di controllo della piattaforma è quella che permette la visualizzazione rapida dei limiti, ovvero dei parametri che un utente deve rispettare se vuole usare Altair per i propri progetti IoT. Questi limiti sono molto stretti nello studio su cui è centrato questo lavoro di tesi, dato che è stata utilizzata la versione gratuita della piattaforma. Nella Home sono riportate le istantanee delle soglie **limiti/minuto** e **limiti/giorno**.

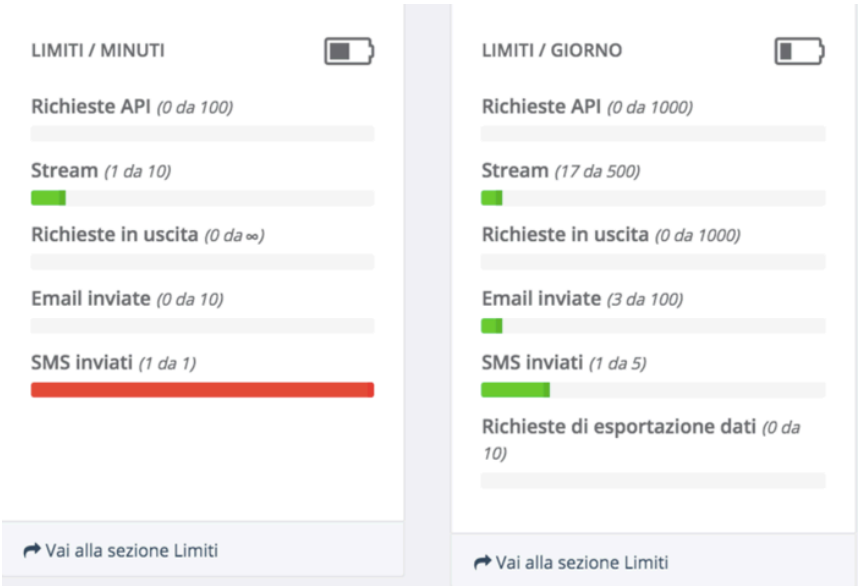


Figura 15: Limiti/minuto e limiti/giorno visibili nel pannello di controllo

Cliccando su "Vai alla sezione limiti" il sito porta l'utente ad una visione più ampia di tutte le limitazioni che deve rispettare, in termini di numero dispositivi registrati e topic MQTT, dimensioni flussi di dati, quantità di email e sms mandati.

CAPITOLO 4: INVIO FLUSSI DATI CON ALTAIR

Dopo aver presentato ampiamente la piattaforma Altair, si può analizzare l'effettivo funzionamento della stessa nell'ambito dell'IoT, sperimentandola nella trasmissione di flussi di dati. Di seguito sono riportate alcune prove fatte per inviare informazioni interfacciandosi con la piattaforma.

La piattaforma Altair SmartCore, come detto, lavora molto bene con il protocollo MQTT, servendosene per due diversi usi:

- **rendere disponibile all'utente un broker dedicato** per comunicare e scambiare le informazioni desiderate. Questi dati non passano attraverso la piattaforma Altair SmartCore, si tratta solo della messa a disposizione del Broker che l'utente può utilizzare come ritiene opportuno;
- **per la trasmissione di flussi di dati** in alternativa al protocollo HTTP. Questi entreranno nel normale flusso della piattaforma e quindi attiveranno ascoltatori, regole, cambi di stato dei dispositivi e allarmi.

4.1 Comunicazione tra Client MQTT attraverso Broker Altair

Può essere utile disporre di un Broker di messaggistica MQTT per scambiare le informazioni con uno o più dispositivi. A tale scopo, Altair Smartcore mette, come detto, a disposizione dell'utente un Broker dedicato e consente di abilitare gli argomenti (topic) su richiesta, per ciascun dispositivo registrato, tramite il pannello di controllo. Lo scambio di informazioni, utilizzando questo processo, non passa attraverso Altair e il suo flusso di regole.
Le operazioni che un utente può eseguire per ogni dispositivo sono: creare e aggiornare le credenziali MQTT, creare ed eliminare argomenti MQTT e rimuovere la configurazione MQTT. Tutte queste azioni sono facilmente realizzabili dal pannello di controllo.
Le credenziali e gli argomenti MQTT possono essere assegnati direttamente nei moduli di creazione/modifica del dispositivo registrato mediante l'attivazione dell'opzione "MQTT ON"; questa permette l'apertura di una sottomaschera nella quale si possono inserire nome utente, password e argomenti per quel device.

Figura 16: Inserimento credenziali e argomenti MQTT nel pannello di controllo di Altair, relativi al dispositivo di default registrato.

L'username MQTT viene creato seguendo il modello [MQTT-nome-utente-inserito].[id-Device], nel caso in figura ***prova.defaultDevice@robertodalonzo.robertodalonzo***.

Gli argomenti vengono generati seguendo il modello [id-device]/[nome-topic-inserito], nel nostro caso ***defaultDevice@robertodalonzo.robertodalonzo/pippo*** e ***defaultDevice@robertodalonzo.robertodalonzo/pluto***.

I dati MQTT inseriti sono poi visibili anche nella pagina "mostra dispositivo" in cui viene riportato anche l'host del Broker: ***mqttbroker.altairsmartcore.com***.

Inoltre MQTT si basa su TCP come protocollo di trasporto, il che significa che, per impostazione predefinita, la connessione non utilizza una comunicazione crittografata. La porta utilizzata per l'uso del protocollo è la porta 1883 e la qualità del servizio (QoS) supportata è QoS 0 (At most once).

A questo punto l'utente ha a disposizione tutte le informazioni MQTT necessarie per stabilire una connessione tra dispositivi.

Nell'esempio sperimentato, si utilizzano due Client MQTT scaricati sul PC e sullo Smartphone e si programmano in base alle informazioni MQTT relative al device, registrato su Altair, al quale si fa riferimento (nel nostro caso il device di default).

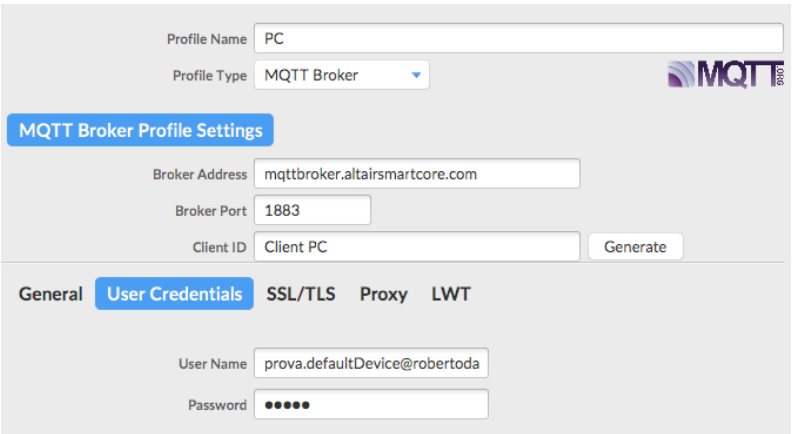


Figura 17: Connessione al broker di riferimento da parte del Client MQTT sul PC

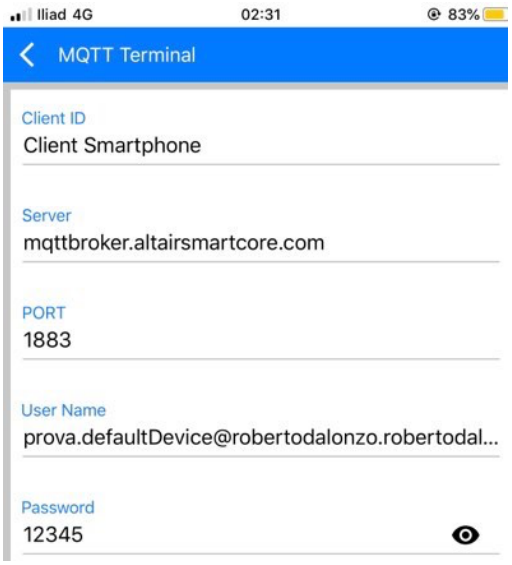


Figura 18: Connessione al broker di riferimento da parte del Client MQTT sullo Smartphone

Una volta connessi entrambi i client all'host messo a disposizione dalla piattaforma Altair, si può procedere con lo scambio di messaggi tra i due protagonisti, ma non prima di aver effettuato la sottoscrizione a uno dei due argomenti relativi al dispositivo di default: ***defaultDevice@robertodalonzo.robertodalonzo/pippo*** e ***defaultDevice@robertodalonzo.robertodalonzo/pluto***.

Ipotizziamo che il Client sul PC si iscriva al topic ***defaultDevice@robertodalonzo.robertodalonzo/pluto***, quindi voglia ricevere tutti i dati inerenti quell'argomento, e che il Client sullo Smartphone pubblichi i suoi messaggi proprio in ***defaultDevice@robertodalonzo.robertodalonzo/pluto***:

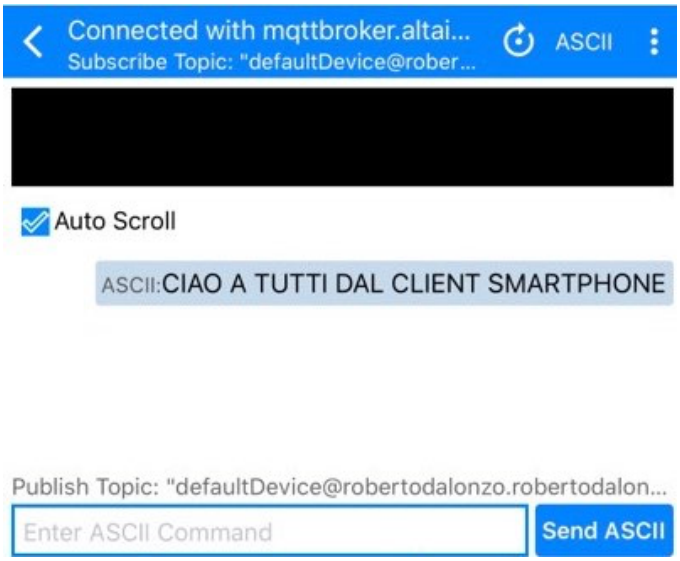


Figura 19: Il Client sullo Smartphone invia un messaggio, in ASCII, al topic *defaultDevice@robertodalonzo.robertodalonzo/pluto*

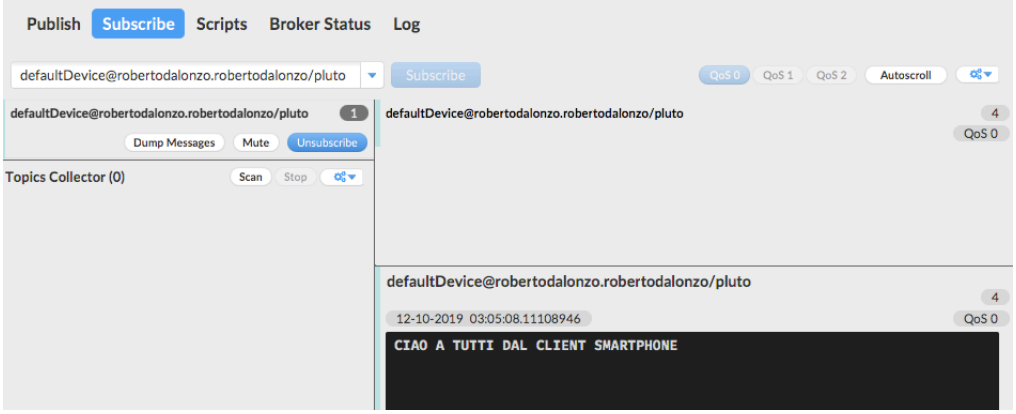


Figura 20: Il Client sul PC, iscritto al topic *defaultDevice@robertodalonzo.robertodalonzo/pluto*, riceve il messaggio mandato dal client sullo Smartphone.

Ovviamente analoga è la situazione se il Client che invia il messaggio è quello sul PC, e quello che lo riceve è lo Smartphone, a patto che gli argomenti, rispettivamente di pubblicazione e di sottoscrizione, siano gli stessi e soprattutto siano tra quelli riportati nelle informazioni del dispositivo (su Altair) a cui si riferiscono. Infatti se, per esempio, uno dei due Client, connesso al Broker Altair, tenta di pubblicare su un topic inventato (per esempio ***Temperatura***) e quindi non presente nelle informazioni del device sulla piattaforma, il tentativo non va a buon fine e non viene trasmesso alcun messaggio.
In questa prova molto semplice sono stati utilizzati soli due client, ma si può pensare anche di utilizzarne a decine per connettere tra loro più dispositivi.

4.2 Invio dati ad Altair utilizzando protocollo MQTT

È possibile inviare flussi di dati a Altair SmartCore tramite il protocollo MQTT e in questo paragrafo verrà approfondito questo scenario con un esempio pratico.
Prima di iniziare bisogna organizzarsi, decidendo cosa si vuole mandare e come lo si vuole mandare. Per questa prova sperimentale sono necessari:

- **Un dispositivo registrato su Altair SmartCore** del quale fondamentalmente ci interessa l'ID: utilizzeremo il dispositivo "Disp1" con ID: *Disp1@robertodalonzo.robertodalonzo*;
- **Un codice Apikey valido**: è un grande token alfanumerico, usato per l'autenticazione univoca, che troviamo nel pannello di controllo di Altair cliccando su impostazioni. Esso risulta essere: **143bf2cb29fca9c3c945447031c971517e0cd0526f0567f6dbe8effbfcaae43c**;

- **I dati che si vogliono inviare**: per questo esempio si considera un semplice set di dati con temperatura e umidità scritto in codice JSON[6]:

```
{
  "temp":21,
  "hum":58
}
```

- **Un Client MQTT**: in questo caso si utilizza il Client Paho Python, scaricabile dal sito **http://www.eclipse.org/paho/clients/python**/.

Dopo aver installato la libreria sul sistema, si può implementare il codice sull'IDLE di Python:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Client paho-mqtt CarriotsMqttServer
# main.py
import paho.mqtt.publish as publish
from json import dumps
from ssl import PROTOCOL_TLSv1

class CarriotsMqttClient():
    host = 'mqtt.altairsmartcore.com'
    port = 1883
    auth = {}
    topic = '/streams'
    tls = None

    def __init__(self, auth, tls=None):
        self.auth = auth
        self.topic = '%s/streams' % auth['username']
        if tls:
            self.tls = tls
            self.port = 8883

    def publish(self, msg):
        try:
            publish.single(topic=self.topic, payload=msg, hostname=self.host, auth=self.auth, tls=self.tls, port=self.port)
        except Exception as ex:
            print(ex)

if __name__ == '__main__':
    auth = {'username': '143bf2cb29fca9c3c945447031c971517e0cd0526f0567f6dbe8effbfcaae43c', 'password': ''}
    # tls_dict = {'tls_version': PROTOCOL_TLSv1}  # ssl version
    msg_dict = {'protocol': 'v2', 'device': 'Disp1@robertodalonzo.robertodalonzo', 'at': 'now', 'data': {'temp': 21, 'hum':58}}
    client_mqtt = CarriotsMqttClient(auth=auth)
    # client_mqtt = CarriotsMqttClient(auth=auth, tls=tls_dict)
    client_mqtt.publish(dumps(msg_dict))
```

username: Apikey

non ssl version

ssl version

Set di dati inviati

ID dispositivo registrato

A questo punto si può mandare in esecuzione il programma, che non dà errori, per cui l'operazione dovrebbe essere andata a buon fine. Come lo vediamo se effettivamente è così? Occorre semplicemente aprire il pannello di controllo di Altair Smartcore e verificare se ci sono nuovi flussi. In effetti, il contatore che ne indica il numero è aumentato e cliccando su "Streams" si può vedere che, nell'elenco dei flussi dati, c'è anche il set di informazioni appena inviato con il Client Paho Python.

[6] **JSON**: acronimo di Javascript Object Notation, è un formato adatto all'interscambio di dati tra applicazioni client/server; fornisce una collezione di dati organizzati logicamente che possono essere facilmente letti da un essere umano e da un computer.

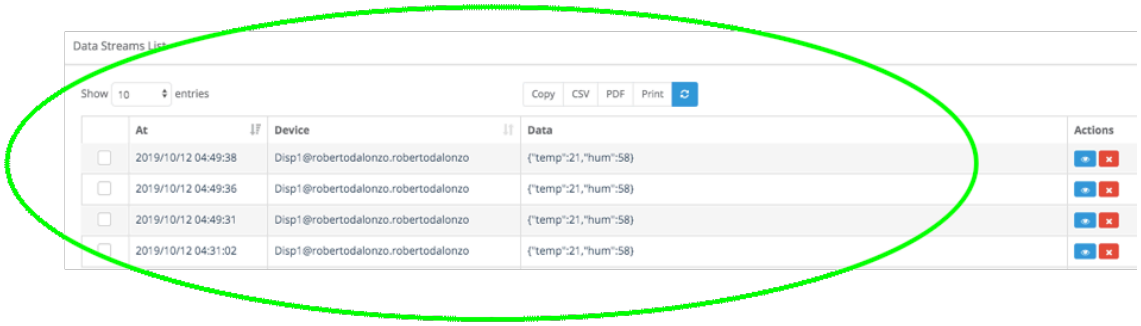


Figura 21: Dopo aver eseguito 4 volte consecutivamente il codice implementato con Python, nell'elenco "data streams" di Altair vengono mostrati i dati inviati.

Per cui i dati inviati entrano nel normale flusso della piattaforma attivando eventuali ascoltatori abilitati.

Ipotizziamo che, nel caso corrente, sia abilitato un ascoltatore che, in base ai dati ricevuti, notifica, tramite email, sms o una pubblicazione MQTT, allo sviluppatore il valore contenuto nel flusso nel campo interessato (ad esempio il campo "temp").

Si progetta quindi una logica aziendale inserendo un nodo "switch" in cui vengono poste, in questo esempio, due condizioni "if" e un "else" in base al dato contenuto nel campo "temp".

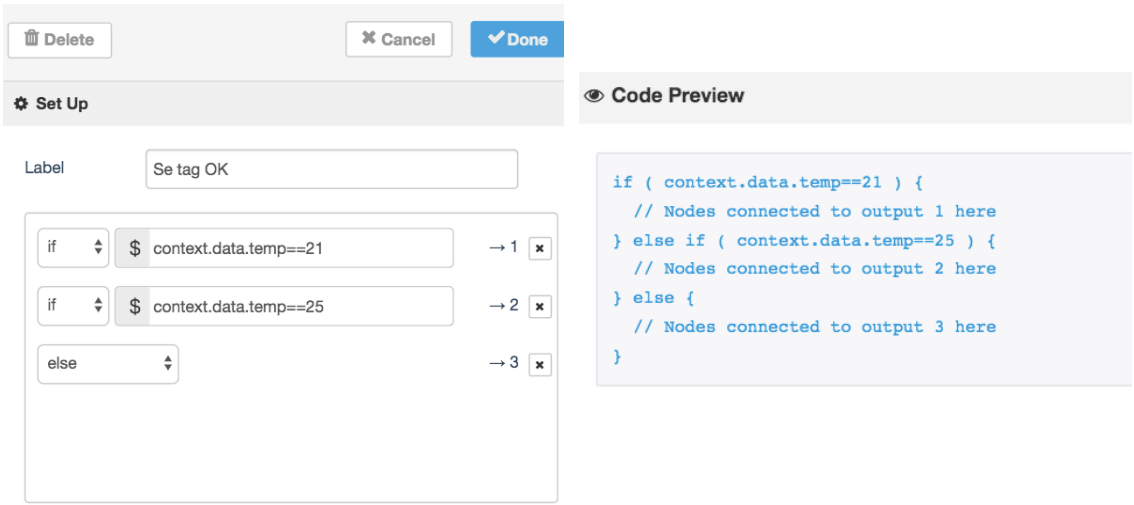


Figura 22: a sinistra sono mostrate le 2 condizioni "if" del nodo "switch": se il contenuto del campo "temp" è 21, se il contenuto del campo "temp" è 25 e la condizione "else". Il codice corrispondente alla tale logica è implementato nella parte destra della figura.

Allora, nell'area di lavoro dell'editor, dal nodo "switch" partirà una cascata di 3 nodi, corrispondenti alle 2 condizioni "if" implementate più l' "else". I nodi aggiunti in questo esempio sono:

- Il **nodo "INVIA EMAIL"** per la prima condizione "if": se il contenuto del campo "temp" è 21;

- Il **nodo "MQTT Publish"** per la seconda condizione "if": se il conetnuto del campo "temp" è 25;
- Il **nodo "INVIA SMS"** per la condizione "else": se il contenuto del campo "temp" non è né 21, né 25.

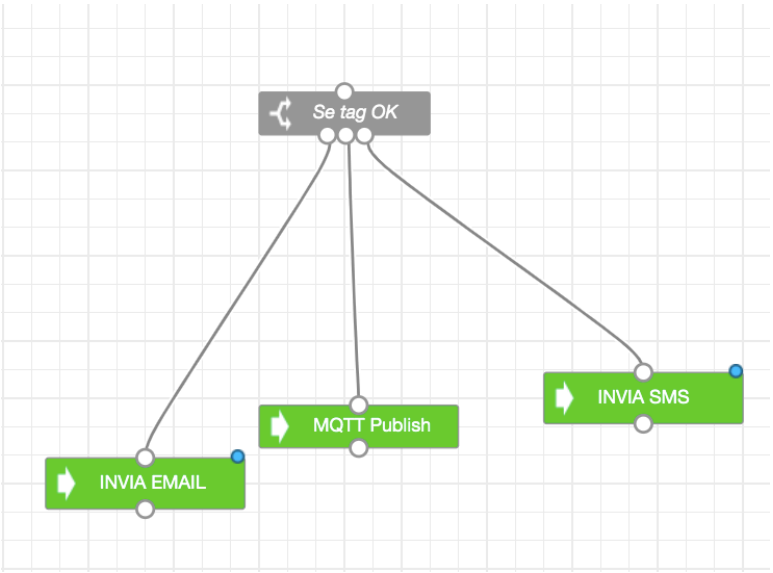


Figura 23: Logica dell'ascoltatore

Allora, se nello stream inviato, il dato su "temp" è 21, l'ascoltatore invierà una email allo sviluppatore, in questo caso all'indirizzo rda@hotmail.it, contenente il messaggio: "**La temperatura è 21 gradi!**".
Se, invece, il dato su "temp" è 25, l'ascoltatore notificherà a un Client MQTT l'informazione con un pubblicazione contenente il messaggio: "**La temperatura è 25 gradi!**".
Infine, se il dato è qualsiasi altro valore diverso da 21 e 25, l'ascoltatore invierà un sms con scritto: "**La temperatura è [valore-campo-temp] gradi!**".

Ovviamente, per la pubblicazione MQTT, l'ascoltatore ha bisogno di conoscere Host del Broker, porta di comunicazione, username e password per l'autenticazione MQTT e topic nel quale pubblicare il messaggio. Pertanto è opportuno configurare il nodo "MQTT Publish", settando tutti i campi con i valori del nostro Client MQTT:

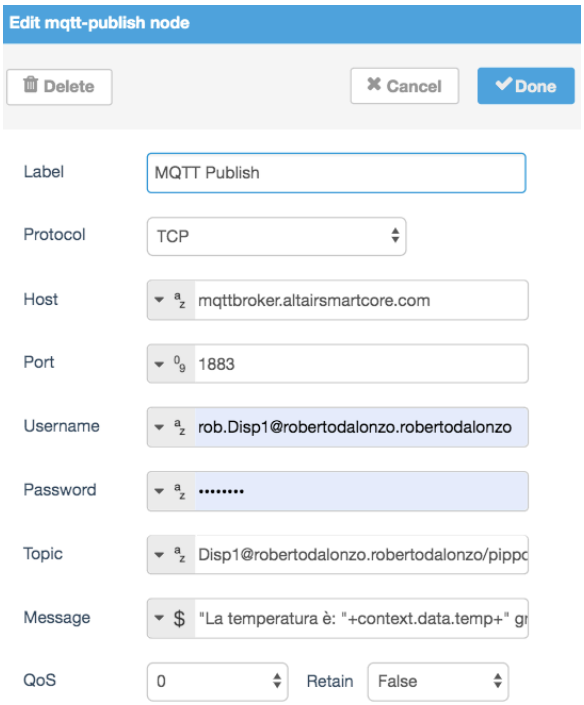
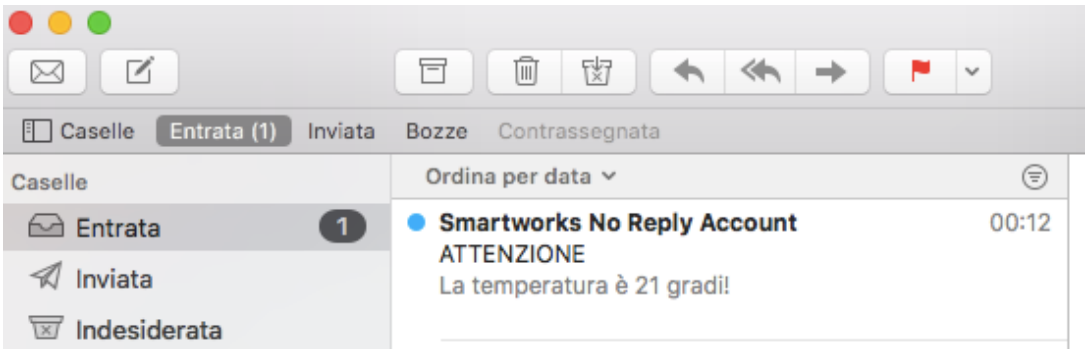


Figura 24: Configurazione nodo "MQTT Publish"

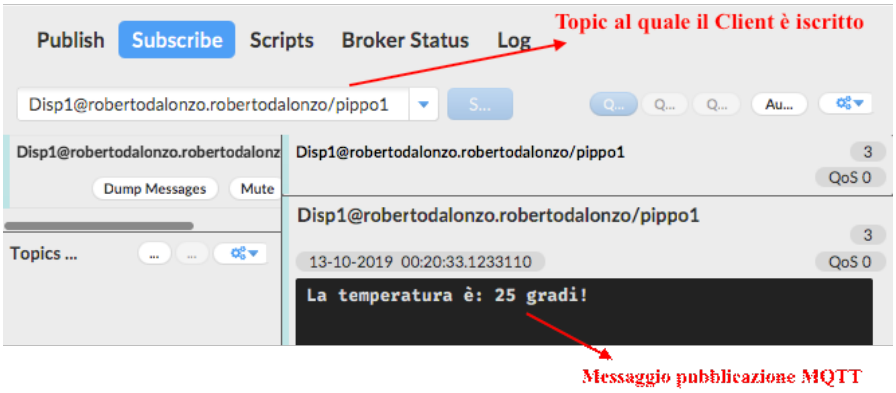
Si può, a questo punto, verificare l'effettivo funzionamento dell'ascoltatore nei 3 casi possibili, per vedere se notifica allo sviluppatore il valore appena ricevuto; per fare questo inviamo 3 volte le informazioni ad Altair tramite il Client Paho Python, modificando, per ogni trasmissione, il valore assegnato a "temp".

1. Invio set di dati {"temp": 21, "hum": 58}: ci aspettiamo una mail all'indirizzo rda@hotmail.it, e infatti subito riceviamo l'avviso:



2. Invio set di dati {"temp": 25, "hum": 58}: ci aspettiamo una pubblicazione MQTT sull'host mqttbroker.altairsmartcore.com (Host del Broker) e nel topic

Disp1@robertodalonzo.robertodalonzo/pippo1 a cui il Client è iscritto; anche in questo caso avviene quanto previsto:



3. In questo caso per il campo "temp" possiamo inserire qualsiasi valore diverso da 21 e da 25; per esempio invio set di dati {"temp": 30, "hum": 58}: ci aspettiamo un SMS contenente l'informazione sul valore della temperatura e infatti, poco dopo, sullo smartphone riceviamo il messaggio:

Con questa semplice prova si è visto come è possibile inviare uno stream, utilizzando MQTT, direttamente ad Altair attivando gli ascoltatori che, a evento ricevuto, provvedono nel notificare all'utente i dati tramite le diverse modalità di avviso.

4.3 Invio dati usando libreria Altair SmartCore per Arduino

L'obiettivo di questo paragrafo è quello di riuscire ad inviare flussi di dati ad Altair utilizzando Arduino[7] e in particolare la sua libreria di invio streams. Lo faremo ricorrendo a un esempio pratico.

Prima di tutto occorre fare un elenco di tutto ciò che servirà in questo test sperimentale:

- **Un "Arduino Uno" dotato di un "Ethernet Shield"** (al suo posto si può usare un WIFI o GRPS Shield) per collegarsi alla rete.

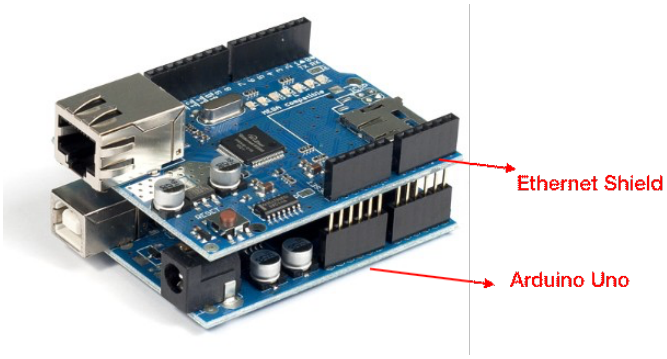


Figura 25: Dispositivo Arduino Uno dotato di Ethernet Shield

- **Un dispositivo registrato sulla piattaforma Altair SmartCore**: sarà utilizzato lo stesso device adoperato nella prova descritta nel paragrafo 4.2, ovvero "Disp1" del quale sostanzialmente ci interessa l'ID: *Disp1@robertodalonzo.robertodalonzo*;
- **Un codice Apikey valido**: anche in questo caso si applicherà l'Apikey usato nel paragrafo 4.2 che si può trovare nel pannello di controllo di Altair cliccando su impostazioni. Questo codice univoco per l'account è quindi: **143bf2cb29fca9c3c945447031c971517e0cd0526f0567f6dbe8effbfcaae43c**;
- **Libreria per Arduino creata per inviare flussi ad Altair SmartCore**: le librerie Arduino consentono agli sketches di avere funzionalità extra. Arduino ne possiede molte che sono già state create o fornite con l'IDE, tuttavia è anche possibile scrivere le proprie librerie o utilizzare quelle scritte da altri. Ogni libreria possiede un file header che descrive ogni metodo e un file sorgente che

[7] **Arduino**: è una piccola scheda elettronica programmabile, equipaggiata di un microcontrollore centrale (il microcontrollore è il cervello del nostro sistema) e di un po' di componenti elettronici così da rendere più semplice il collegamento con dispositivi esterni di vario tipo. Inoltre comprende una parte software, o IDE, che eseguita su un computer, viene usata per scrivere e caricare codice informatico (in linguaggio "C") nella scheda stessa. I codici informatici di Arduino sono chiamati Sketches.

elabora il codice. In questo esempio sarà impiegata la libreria "SendCarriots", che serve per facilitare la comunicazione con la piattaforma Altair, scaricandola dal sito **https://github.com/carriots/arduino_library**.

Una volta completato il download, è necessario decomprimerla e copiarla nel file *Arduino/libreries* per poterla utilizzare. Utilizzando questa libreria è possibile inviare elenchi di coppie chiave-valore alla piattaforma semplicemente chiamando un metodo predeterminato. Il metodo usato per inviare flussi di dati a Altair SmartCore è il metodo "invia" che dispone di 4 parametri da definire per trasmettere correttamente un flusso:

1. Il primo parametro è un **array bidimensionale**: il numero di righe di questo vettore è a scelta dell'utente, ma il numero di colonne deve essere due (una colonna per la chiave e una per il valore);
2. Il secondo parametro è **il numero di righe o elementi presenti nell'array**;
3. Il terzo parametro è **l'Apikey** univoco che identifica il proprio account Altair Smartcore;
4. Il quarto parametro è l'**id_developer del dispositivo** che specifica quale dei device registrati si vuole utilizzare;

Dopo aver effettuato tutti i collegamenti tra Arduino e il router (tramite cavo Ethernet) e tra Arduino e PC, si può iniziare con il test.

Ipotizzando, per questa prova, di voler inviare un set di 3 coppie chiave-valore (numero di righe = 3), in cui i campi sono "place","temp" e "hum" ("luogo", "temperatura" e "umidità") e inserendo i valori relativi ai propri dispositivi nello sketch, si può implementare il codice come segue sull'IDE Arduino:

```
sketch_sep21a §
#include <SPI.h>
#include <Ethernet.h>
#include <SendCarriots.h>                                                    // È necessario includere qui la libreria per poterla utilizzare nello sketch

const String APIKEY="143bf2cb29fca9c3c945447031c971517e0cd0526f0567f6dbe8effbfcaae43c";    // Apikey univoco relativo all'account Altair
const String DEVICE="Displ@robertodalonzo.robertodalonzo";                    // id_developer del dispositivo registrato da utilizzare

const int numElements=3;                                  // Specificare numero di righe dell'array contenente i deti che si vogliono inviare. In questo caso 3

byte mac[] = {  0x90, 0xA2, 0xDA, 0x00, 0xFE, 0xA6 }; // Indirizzo MAC: I più recenti Ethernet Shield ne hanno uno stampato su un adesivo sullo Shield

IPAddress ip(192,168,1,5);                                                    // Indirizzo IP

SendCarriots sender;                                                          // Crea un'istanza della libreria SendCarriots

void setup() {
  Serial.begin(9600);                                                        // avvia la porta seriale
  Serial.println(F("Starting"));

  Ethernet.begin(mac,ip);                                                    // Avvia la connessione Ethernet

}

void loop() {
  String array[numElements][2] =  {{"place", "kitchen"}, {"temp", 28}, {"hum", 58}};          //Definizione dati dell'array da inviare

  Serial.println(sender.send(array, numElements, APIKEY, DEVICE));           // Utilizzando l'istanza della libreria, chiama il metodo invia

  delay(10000); }                                          // invia questo stream ogni 10 secondi (la frequenza è a discrezione dello sviluppatore)
```

Figura 26: Codice Arduino per testare il metodo invia. Ogni riga del codice è spiegata a destra.

Una volta implementato il codice, si può procedere con la verifica e compilazione dello stesso (nel nostro caso non risultano errori) e il successivo caricamento sulla scheda. A questo punto Arduino inizia a funzionare e manda, ogni 10 secondi (come imposto nell'ultima riga di codice), lo stream. Effettivamente, andando a vedere nella sezione flussi del pannello di controllo di Altair, emergono questi set di dati con una frequenza di uno ogni 10 secondi.

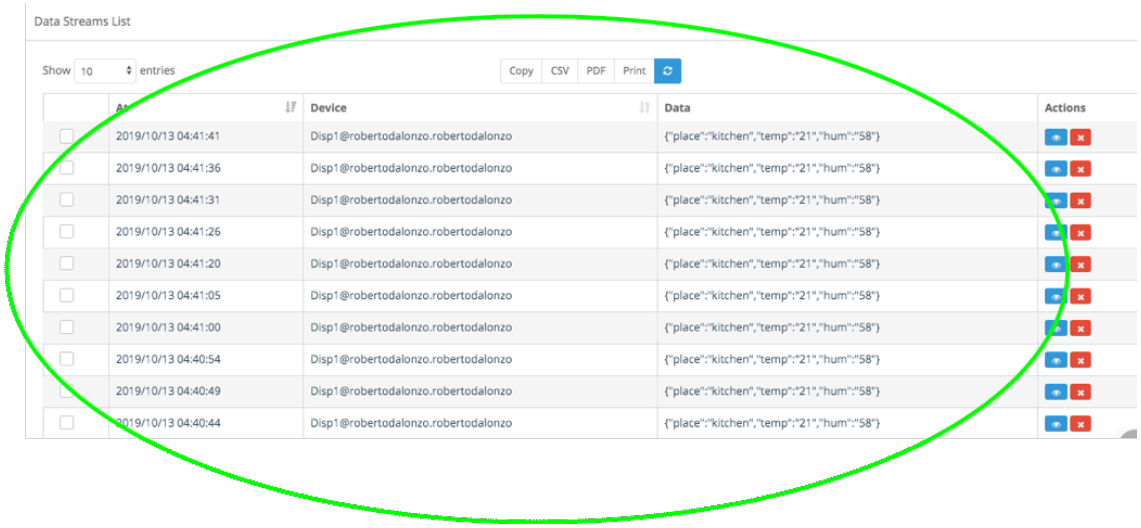


Figura 27: Nell'elenco "data streams" di Altair vengono mostrati i dati inviati da Arduino ogni 10 secondi

Quindi, anche in questo caso, i dati entrano nel normale flusso di Altair SmartCore, attivando gli ascoltatori abilitati. Per cui si può nuovamente progettare una logica aziendale per gli ascoltatori in modo che, una volta ricevuto il dato, esso venga notificato allo sviluppatore sempre mediante email, sms, pubblicazioni MQTT (che in questo esempio saranno inviati "costantemente" ogni 10 secondi), come visto nel paragrafo 4.2.

4.4 Invio dati ad Altair tramite libreria MQTT per Arduino

In questo paragrafo si tenterà di inviare flussi di dati ad Altair SmartCore sempre utilizzando Arduino, ma in questo caso tramite la sua libreria MQTT. Anche per l'esecuzione di questo esempio pratico serve fare l'elenco di tutto ciò che occorrerà per iniziare:

- **Un "Arduino Uno" dotato di un "Ethernet Shield"** (al suo posto si può usare un WIFI o GRPS Shield) per collegarsi alla rete; si utilizzerà lo stesso device visto nel test nel paragrafo 4.3;
- **Un dispositivo registrato sulla piattaforma Altair SmartCore**: sarà utilizzato lo stesso device adoperato nelle prove precedenti, cioè "Disp1" del quale sostanzialmente ci interessa l'ID: *Disp1@robertodalonzo.robertodalonzo*;
- **Un codice Apikey valido**: l'Apikey è lo stesso usato negli altri test dato che è relativo all'account Altair SmartCore registrato: **143bf2cb29fca9c3c945447031c971517e0cd0526f0567f6dbe8effbfcaae43c**;
- **Libreria MQTT per inviare flussi di dati ad Altair**: in questa prova sarà utilizzata la libreria "PubSubClient" che permette una comunicazione tra device e piattaforma semplice e intuitiva. È possibile scaricarla dal sito **https://github.com/knolleary/pubsubclient**.

Come visto anche nella prova precedente, una volta finito il download delle libreria, la si decomprime e la si copia nel file *Arduino/libreries* per utilizzarla. Questa libreria è adeguata per inviare un JSON a Altair SmartCore semplicemente chiamando un metodo predefinito. In realtà per questo esempio, saranno due i metodi adoperati: il metodo "**connetti**" e il metodo "**pubblica**":
Il metodo "connetti" serve per connettersi a Altair SmartCore. Esso dispone di 3 parametri che è necessario definire per inviare correttamente un flusso:

1. Il primo parametro è **USERNAME**, quindi il nome utente dello sviluppatore;
2. Il secondo parametro è l'**APIKEY**: il codice che identifica il proprio account Altair;
3. Il terzo parametro è la **PASSWORD**: è un campo obbligatorio e quindi non può essere lasciato vuoto. Per Altair SmartCore password= NONE;

Il metodo "pubblica" è utilizzato per pubblicare i flussi su Altair SmartCore con MQTT. Esso ha 2 parametri da definire:

1. Il primo parametro è il **TOPIC**, che consiste nella Apikey/MODE (mode può essere "streams" o "status");
2. Il secondo parametro è lo **STREAM** stesso, cioè proprio il flusso di dati che si vuole trasmettere.

Una volta stabilita la connessione tra Arduino, PC e Router (con cavo Ethernet), si può iniziare con la prova nella quale si ipotizza di voler inviare un flusso di dati JSON contenente tre campi, "temp", "hum" e "number", con i relativi valori. Nello sketch sull'IDE di Arduino, si inseriscono i dati relativi ai propri dispositivi (username, apikey, nome device, etc) implementando il seguente codice:

```
sketch_sep03a §
*/
#include <SPI.h>
#include <Ethernet.h>
#include <PubSubClient.h>   // È necessario includere qui la libreria per poterla utilizzare nello sketch

byte mac[] = {0x90, 0xA2, 0xDA, 0x00, 0xFE, 0xA6};   // Indirizzo MAC: I più recenti Ethernet Shield ne hanno uno stampato su un adesivo sullo Shield

// Ipmposta l'indirizzo IP statico da utilizzare se il DHCP non riesce ad assegnarlo
IPAddress ip(192,168,1,5);    // Indirizzo IP statico

////////////////////////////////////////
// DEFINIZIONE CONNETTIVITÀ SMARTCORE //
////////////////////////////////////////
#define SMARTCORE_DEVICE       "Disp1"
#define SMARTCORE_USERNAME     "robertodalonzo"
#define SMARTCORE_APIKEY       "143bf2cb29fca9c3c945447031c971517e0cd0526f0567f6dbe8effbfcaae43c"  // Apikey univoco relativo all'account Altair
#define SMARTCORE_MQTT_PORT    1883
#define SMARTCORE_MQTT_HOST    "mqtt.altairsmartcore.com"
#define SMARTCORE_MQTT_TOPPIC  "/streams"

const String DEVICE_ID = SMARTCORE_DEVICE "@" SMARTCORE_USERNAME "." SMARTCORE_USERNAME;
////////////////////////////////////////

// Inizializza il Client Ethernet
EthernetClient eth_client;
// Crea un'stanza del Client MQTT
PubSubClient mqtt_client(SMARTCORE_MQTT_HOST, SMARTCORE_MQTT_PORT, eth_client);

void mqtt_connect() {
  // Ripeti finchè non siamo connessi a MQTT
  while (!mqtt_client.connected()) {
    Serial.print("Attempting MQTT connection... ");
    // Tentativo di connessione
    if (mqtt_client.connect(SMARTCORE_USERNAME, SMARTCORE_APIKEY, NULL)) {
      Serial.println("Success");
    // Connessione fallita
    } else {
      Serial.print("Failed connection: rc=");
      Serial.print(mqtt_client.state());
      // Attendi un secondo prima di riprovare
      delay(1000);
    }
  }
}

void mqtt_publish(char* frame) {
  Serial.print("Publishing at " SMARTCORE_MQTT_TOPPIC ": ");
  Serial.println(frame);
  // Pubblica frame di dati
  if (mqtt_client.publish(SMARTCORE_APIKEY SMARTCORE_MQTT_TOPPIC, frame)) {
    Serial.println("Published.");
  } else {
    Serial.println("Failed publishing.");
  }
  Serial.println();
}

char* get_frame() {
  // Crea dati JSON.
  String data_json = "{'temp': 32, 'hum':70, 'number':123}";
  // Crea Frame JSON.
  String frame_json =  "{\"protocol\":\"v2\",\"device\":"+DEVICE_ID+",\"at\":1488447613,\"data\":"+data_json+"}";

  int frame_length = frame_json.length() + data_json.length() + 1;
  char frame_buffer[frame_length];
  frame_json.toCharArray(frame_buffer, frame_length);
  return frame_buffer;
}
void setup() {
  // Apri le comunicazioni seriali e attendi l'apertura della porta
  Serial.begin(9600);
  while (!Serial) { //Questo passaggio è necessario solo per Leonardo, non per Arduino Uno
    ;
  // Avvia la connessione Ethernet
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    // Try to congifure using IP address instead of DHCP:
    Ethernet.begin(mac, ip);
  } else {
    Serial.println("Ethernet configured using DHCP");
  }
  Serial.println();
  // concedi allo Shield Ethernet un secondo per inizializzare
  delay(1000);
}

void loop(){
  // Connetti a MQTT.
  if (!mqtt_client.connected()) {
    mqtt_connect();
  }
  // Invia frame
  mqtt_publish(get_frame());
  delay(10000);
}
```

Figura 28: Codice Arduino per testare l'invio di flussi di dati tramite libreria MQTT.

Dopo aver verificato e compilato lo sketch, si può caricare il programma su Arduino che inizierà ad eseguire il proprio compito: inviare flussi di dati contenenti informazioni su temperatura, umidità e numero, con una frequenza di uno stream ogni 10 secondi (come imposto nell'ultima riga di codice). L'oggetto connesso alla rete invierà, allora, i dati direttamente ad Altair, nel cui pannello di controllo appaiono di volta in volta i flussi trasmessi.

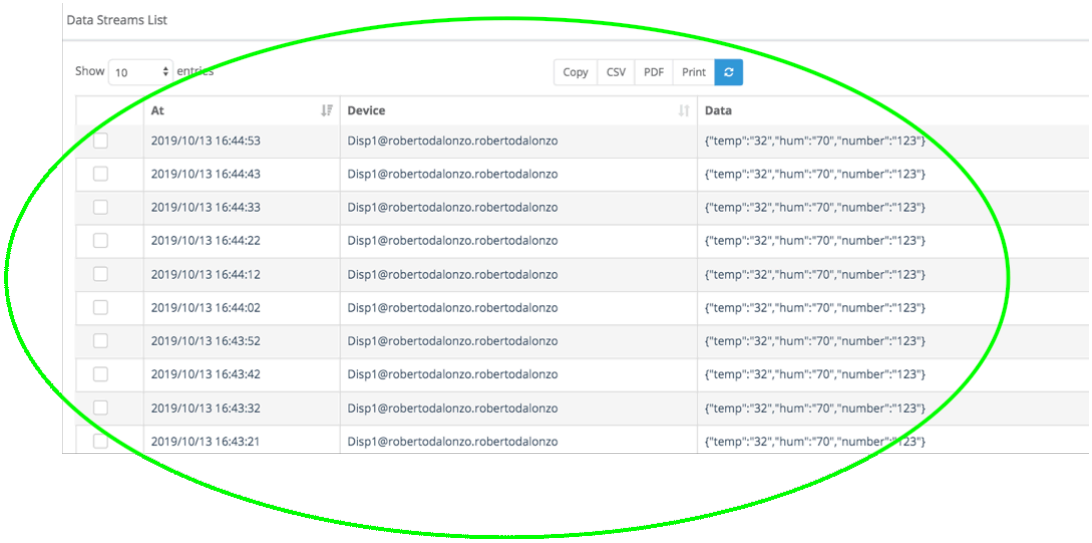


Figura 29: Nell'elenco "data streams" di Altair vengono mostrati i dati inviati da Arduino

Anche in questo esempio, i flussi di dati, ricevuti e collezionati nella sezione "Streams" di Altair, potranno attivare gli ascoltatori e di conseguenza permettere la notifica, allo sviluppatore, dei dati trasmessi.

CAPITOLO 5: CONCLUSIONI ED OSSERVAZIONI

Lo scopo di questo progetto di tesi è stato quello di studiare la piattaforma Altair SmartCore per le tecnologie IoT e di vedere, quindi, come avviene la comunicazione tra gli oggetti "intelligenti" e la piattaforma stessa. Prima di iniziare a testare la reale utilità e l'effettivo funzionamento di Altair sono state fatte delle panoramiche su cosa è la tecnologia IoT e cosa rappresenta nel mondo odierno e sul protocollo MQTT, considerato, ad oggi, il protocollo standard per questo tipo di trasmissioni dati tra device grazie alla sua semplicità e leggerezza.

È stato poi presentato il pannello di controllo di Altair SmartCore, descrivendo tutte le sezioni presenti: cosa sono e come si registrano i device IoT, cosa sono gli ascoltatori, cosa sono gli allarmi, quali sono i flussi di dati e quali sono i limiti da rispettare.

Per quanto riguarda l'invio di flussi dati, Altair rende disponibile all'utente un Broker dedicato per permettere ai client di comunicare tramite il protocollo MQTT, ma consente anche una semplice trasmissione di informazioni tra device e piattaforma stessa (sempre sfruttando MQTT). Di notevole rilevanza è, poi, il compito degli ascoltatori che, in base ai dati collezionati nel data stream della piattaforma, può informare in tempo reale l'utente o l'azienda sui valori ricevuti.

Negli esempi implementati, focalizzati all'apprendimento della logica lavorativa di Altair, si è fatto riferimento a semplici situazioni in cui spettava allo sviluppatore/utente scegliere il set di informazioni da inviare, specificando campi di interesse e valori relativi. Tuttavia, molto spesso nella realtà dell'IoT, non è solo l'uomo a dover selezionare cosa inviare; sono proprio i dispositivi stessi, considerati intelligenti, che misurano dati e informazioni dall'ambiente circostante e li trasmettono. In tal caso i dati non saranno più statici, come quelli utilizzati negli esempi, ma acquisiscono dinamicità in quanto possono evolvere e cambiare nel tempo ad ogni misurazione.

Allora la potenza di Altair è ancora più rimarcata se si considera una moltitudine di sensori, collegati a microcontrollori, che raccolgono dati real time e li inviano costantemente alla piattaforma, dove vengono collezionati nell'elenco flussi dati e dove possono attivare ascoltatori in grado di aggiornare in diretta un qualsiasi utente interessato.

L'invio "continuo" di informazioni è determinato dai microcontrollori, all'interno dei quali viene caricato un codice/programma specifico per fare in loop una o più azioni

caratteristiche (come appunto la misurazione dei dati e la loro trasmissione). Nel progetto corrente, questo meccanismo è stato osservato negli ultimi paragrafi dove, tramite un dispositivo Arduino connesso, i flussi di dati venivano inviati ogni 10 secondi sulla piattaforma.
Per cui, dopo questa ampia panoramica sullo studio di Altair SmartCore per l'IoT, un futuro possibile progetto di lavoro potrebbe riguardare proprio l'apprendimento di come, attraverso sensori di ogni tipo collegati a microcontrollori, i dati possano diventare dinamici e quindi essere trasmessi autonomamente con valori mutanti nel tempo.

Dopo questo studio si può concludere che il rilascio di alcune tecnologie chiave, come appunto le piattaforme Cloud, e l'andamento del mercato, han fatto si che l'Internet of Things sia ora una nuova grande realtà. La prospettiva dell'IoT, definita come un insieme onnipresente di dispositivi connessi, potrebbe radicalmente cambiare ciò che la gente pensa sul significato dell'essere "online". È il Cloud che ha introdotto il significato dell'essere sempre "online" grazie alla continua crescita dell'ubiquità di calcolo; questo fa si che tutto sia connesso a internet e i dati elaborati vengano utilizzati per scopi vari, creando non solo informazione, ma anche conoscenza. Così facendo l'IoT e il Cloud sembrano proprio fatti apposta per lavorare in simbiosi; i piccoli device IoT con poca capacità di calcolo necessitano della potenza di elaborazione che il Cloud può offrire in modo da gestire, memorizzare e analizzare il continuo flusso di dati generati dai dispositivi.

SITOGRAFIA

https://www.altairsmartworks.com/developers/documentation

https://www.altairsmartworks.com/developers/tutorials

https://www.internet4things.it/iot-library/mqtt-cose-e-come-funziona-il-protocollo-alla-base-delliot/

http://www.intelligenzaartificiale.it/internet-of-things/

https://www.html.it/pag/49123/iot-la-scelta-del-protocollo-http-vs-mqtt/

https://www.techeconomy.it/2015/05/07/iot-e-il-futuro-della-societa-interconnessa-intervista-a-kevin-ashton/

http://www.lucadentella.it/2016/10/24/mqtt-introduzione/

https://www.ontrack.com/it/blog/internet-of-things-oggetti-intelligenti-dati/

https://it.emcelettronica.com/mqtt-il-protocollo-delliot

https://www.cadlog.it/iot/