CNR-IIA: PROGETTO ARMONIA

IoTTy

Autore	Francesco D'Amore
Data di creazione	1-Ottobre-2022
Ultima revisione	30-Ottobre-2022
Titolo	D5.2 - IoTTy
Soggetto	WP5 - Infrastruttura ICT
Stato	Completato
Editore	CNR-IIA
Tipo	Deliverable
Identificazione	D5.2
Descrizione	
Contributi	Mariantonia Bencardino, Delia Evelina Bruno, Valentino Mannarino

INDICE

INTRODUZIONE E FINALITÀ

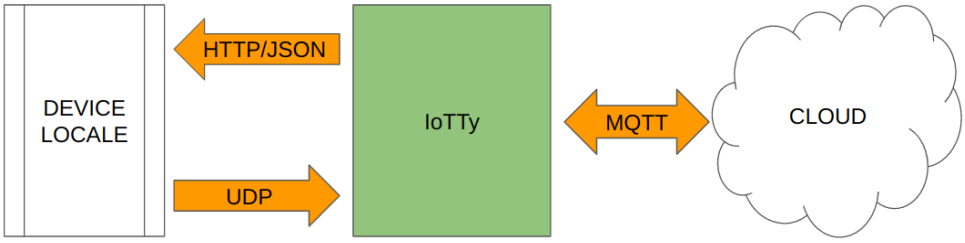
I dati prodotti dalla piattaforma multi sensore sviluppata nel WP2 devono essere trasmessi in risorse computazionali in cloud per poter essere processati e analizzati. Trasmettere i dati in (quasi) tempo reale permette di sviluppare applicazioni orientate al monitoraggio, di interesse stringente date le finalità del progetto ARMONIA. La trasmissione in (quasi) tempo reale non è sempre possibile per mancanza o scarsità di rete dati e deve essere sempre presente l'opzione batch, cioè l'accumulo di dati in sistemi locali e la successiva trasmissione massiva.

I sistemi IoT (Internet delle Cose - Internet of Things) stanno assumendo sempre maggior importanza nel collegare risorse computazionali remote con sorgenti dati ed è interesse di chi scrive sperimentare l'approccio IoT in sistemi di misura orientati all'osservazione della terra. I metodi propri dell'IoT possono essere usati nel contesto di ARMONIA per poter trasmettere in (quasi) tempo reale i dati misurati dalle piattaforme di sensori sviluppate nel WP2.

Nei paragrafi successivi verrà presentata **IoTTy**, una libreria sviluppata nell'ambito del progetto ARMONIA al fine di creare un ponte IoT fra i dati misurati dalla piattaforma multi sensore (deliverables D2.1 e D2.2) e le risorse computazionali progettate per l'analisi e la condivisione del dato (deliverable D5.1).

L'immagine sopra mostra un terminale di IoTTy che comunica all'utente di essere pronto a ricevere messaggi e dati.

STRUTTURA DEL PONTE IoT



In figura viene riportato lo schema sintetico della funzionalità della libreria: IoTTy espone localmente un client UDP che si interfaccia con gli script di produzione dei dati, mentre internamente funziona come un automa a stati finiti. Il passaggio da uno stato all'altro di IoTTy viene comandato da messaggi inviati tramite l'interfaccia UDP, mentre l'interfaccia HTTP viene usata per ottenere stato corrente e le informazioni ad esso associate.

Il client di IoTTy è generalmente un loop che legge i dati da sensore, li organizza secondo un formato prestabilito e ordinale e li invia a IoTTy tramite l'interfaccia UDP. Il client deve interrogare l

'interfaccia HTTP per conoscere lo stato di IoTTy, ad esempio per sapere se IoTTy è in grado di trasmettere attraverso un canale attivo MQTT.

SCHEMA DI FLUSSO DI IoTTy

I comandi che IoTTy può elaborare attraverso la sua interfaccia UDP sono:

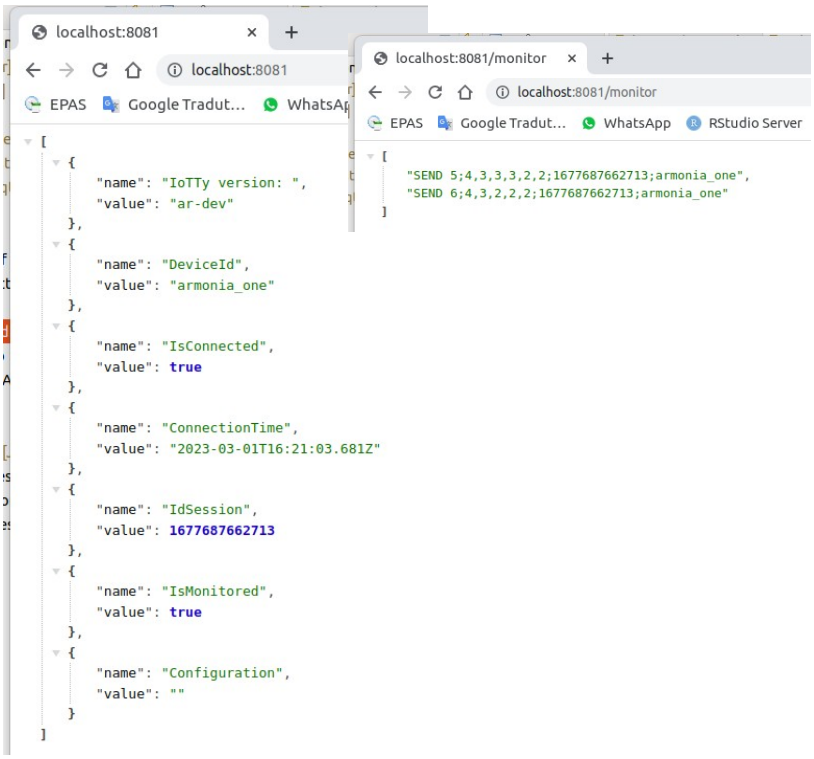
- START
- END
- SEND
- STARTM
- ENDM
- SEND
- UPLOAD

I comandi di **START** e di **END** provocano un cambio di stato dell'automa interno: START pone il sistema nello stato di ricevere dati tramite il comando **SEND**. **STARTM** fa partire il monitoraggio dei dati che vengono così salvati anche localmente e non solo inviati sul canale MQTT mentre **ENDM** termina il salvataggio in locale dei dati. I dati in locale possono essere inviati massivamente su un cloud remoto tramite il comando **UPLOAD**. Il comando **END** termina il canale di invio dati e rilascia le risorse MQTT.

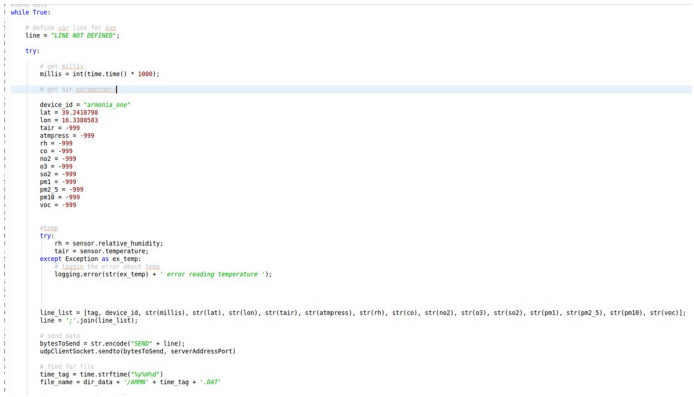
Tutto il testo che segue un comando di SEND viene inviato sul canale MQTT, se attivo: è compito del MQTT Broker del ricevente applicare le logiche che ritiene opportuno per interpretare il messaggio inviato. Questo approccio agnostico sul dato in ingresso conferisce al ponte IoT realizzato da IoTTY la flessibilità necessaria ad essere utilizzato in differenti contesti applicativi, anche al di fuori del progetto ARMONIA.

I comandi di STARTM e ENDM mettono lo stato di IoTTY in modalità monitor: in questo stato i dati inviati tramite SEND vengono anche salvati nel device locale e possono essere inviati tramite il comando UPLOAD, come citato sopra. Questa funzionalità necessità di una configurazione particolare dello storage finale dove verranno inviati e immagazzinati massivamente i dati salvati localmente.

Lo stato dell'automa interno può essere interrogato tramite richiesta GET HTTP, come mostrato nelle due immagini che seguono: sono due messaggi JSON rielaborati dal browser che mostrano nel primo caso lo stato di IoTTY con un canale MQTT attivo e nel secondo caso i dati inviati e gestiti dal monitor interno di IoTTY precedentemente attivato tramite il comando STARTM.



CLIENT UDP: GENERAZIONE DEI DATI DA SENSORE



Il client UDP di IoTTY è essenzialmente un loop che attiva il canale tramite il comando START e poi prova a inviare dati. Nell'ambito del progetto ARMONIA, IoTTy verrà installato in una Raspberry Pi montata sulla piastra multisensore come progettato e descritto nel deliverable D2.2. Nella Raspberry Pi verrà inserita una procedura Python di cui una porzione di codice viene riportato nell'immagine sopra. Per il lettore che ha dimestichezza con il linguaggio in oggetto, si tratta di un loop che raccoglie il dato, qui semplicemente simulato, e lo invia su interfaccia UDP a IoTTY usando il comando SEND. IoTTy invierà tutto quello che viene dopo il comando SEND sul canale MQTT, se attivo. Il canale MQTT è le ragioni della scelta in tal senso verranno presentati nel prossimo paragrafo.

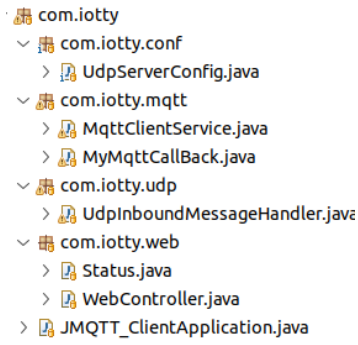
IL PROTOCOLLO MQTT

MQTT è uno dei protocolli abilitanti dell'universo IoT. È un protocollo circuitato, che cioè stabilisce un circuito con il ricevente. I messaggi sono estremamente ottimizzati per il contesto applicativo e la comunicazione è bidirezionale: sul protocollo viaggiano non solo i dati dalla sorgente al destinatario ma anche configurazioni e comandi per la sorgente, informazioni che possono essere usati per configurare i dispositivi che producono dati.

Il ricevente è un MQTT Broker, un dispositivo cioè che gestisce il messaggio MQTT proveniente dal client (in questo caso IoTTY) o che invia al client comandi e configurazioni sempre tramite MQTT. Come indicato nel deliverable D5.1, i dati di ARMONIA vengono gestiti in Google Cloud Platform (GCP): GCP, al tempo in cui si sta scrivendo, espone il servizio **IoT Core** che consente di avere un MQTT Broker pienamente integrato con l'ecosistema GCP. IoT Core verrà dismesso nell'agosto del 2023: questo non darà problemi particolari ai client come IoTTY perchè MQTT è un protocollo standard che potrà funzionare con altri MQTT Broker integrati in GCP come sistemi di terze parti.

L'uso di tali protocolli in ARMONIA consente ai dispositivi sviluppati nel progetto di essere conformi alle metodologie IoT e di ottimizzare quindi l'invio dei messaggi verso le risorse computazionali dove i dati verranno immagazzinati e analizzati.

ASPETTI IMPLEMENTATIVI DI IOTTY



La libreria IoTTy è stata sviluppata in Java. Di fianco la struttura dei package dove viene organizzato il codice sorgente. Attualmente il codice sorgente è gestito in un repository GIT ospitato su Google Cloud. È intenzione di chi scrive di spostare IoTTy su GitHub.