ORDINE DI AVVIO SOFTWARE

Per prima cosa avviare il broker MQTT mosquitto sul raspberry

- systemctl enable mosquitto

Poi avviare data-publisher seguendo le istruzioni nel README

Fare la stessa cosa con data-subscriber

Connettersi al servizio tramite l'indirizzo indicato

DESCRIZIONE PROGETTO

Data Collector

- Arduino Uno + HC05 + HC-SR04
- Riutilizzato il codice del sensore di prossimità del progetto 02, modificato per fornire l'interfaccia giusta
- ProducerTask genera un file json usando la libreria ArduinoJson, lo formatta secondo le specifiche, lo serializza e lo invia a Raspberry
- MsgService e BluetoothMsgService non sono stati modificati
- Rimossi diversi file per la ricezione dati, non utilizzati nel nostro caso

Data Publisher

- Raspberry
- Siamo partiti dal file fornito event-loop.py, per la gestione degli eventi bluetooth, nel quale abbiamo integrato le funzioni di data-publisher (sotto # Funzioni MQTT)
- In questo modo il sistema si connette al broker MQTT, ricerca il dispositivo, si connette, riceve i dati e li decodifica come dati json, poi li pubblica sul topic "data" nel broker.

Data Subscriber

- Raspberry
- mqtt-to-rest.py invariato
- crea un client, si connette al broker MQTT e sottoscrive il topic "data"
- configura il server Flask, che fornisce un endpoint HTTP GET per visualizzare i dati. Nel nostro caso si utilizza l'IP del localhost, porta 1883 con topic data (localhost:1883/data)

RASPBERRY

REST API
-localhost:1883/data

CLIENT VARI
-visualizzano i dati
-per questo esempio, devono essere nella stessa rete del server Flask

BROKER
-mosquitto
-riceve dati da publisher e li condivide ai subscriber in base a topic

SUBSCRIBER
-si sottoscrive al topic "data"
-tramite Flask espone interfaccia rest

PUBLISHER
-riceve json da arduino
-pubblica su un topic nel broker "data"

ARDUINO

COLLECTOR
-genera dati tramite sensore HC-SR04
-li condivite via bluetooth al pub