12_Cloud_MQTT_Relazione

Citterio Giorgio e Colombo Umberto

Lo scopo di questa è utilizzare il protocollo MQTT per lo scambio di messaggi fra dispositivi IoT.

Studio delle funzionalità protocollo MQTT (parte 1)

In questa fase preliminare abbiamo studiato come funziona il protocollo MQTT attraverso dei video.

Utilizzo broker MQTT gratuito (parte 2)

In questa parte abbiamo installato MQTTX e utilizzando il broker gratuito test.mosquitto.org abbiamo fatto delle prove

Publish e Subscribe broker gratuito (parte 3)

Nella terza parte abbiamo fatto la publish e la subscribe da python sotto windows utilizzando il broker gratuito.

codice python provaPublish.py:

```
import paho.mqtt.publish as publish
import time
TOPIC = "umbeGio2"
BROKER = "172.17.4.29"

i = 0
while True:
    i += 1
    msg=str(i)
    print(msg)

    publish.single(TOPIC, msg, hostname=BROKER)

    time.sleep(5)
```

codice python provaSubscribe.py:

```
import paho.mqtt.client as client
TOPIC="umbeGio2"
BROKER = "172.17.4.29"

def on_connect(subscriber, userdata, flags, rc):
    print("Connesso con return code" + str(rc))

    subscriber.subscribe(TOPIC)

def on_message(subscriber, userdata, msg):
    print(msg.topic+" "+str(msg.payload))
```

```
subscriber = client.Client()
subscriber.on_connect = on_connect
subscriber.on_message = on_message

subscriber.connect(BROKER, 1883, 60)

subscriber.loop_forever()
```

Installazione broker mosquitto su Windows (parte 4)

In questa parte dell'attività abbiamo installato il broker gratuito mosquitto e l'abbiamo configurato secondo le nostre esigenze.

Installazione di un broker MQTT su macchina virtuale Debian sotto Virtualbox (parte5, opzionale, non fatto)

Installazione del broker MQTT su Raspberry (parte6, opzionale, non fatto)

Abbiamo preferito tenere il broker su windows, almeno finora.

Trasferimento dei client su Raspberry ed integrazione col sensore/attuatore (parte7)

In questa parte dell'attività abbiamo spostato il publisher(sensore) e il subscriber(attuatore) su raspberry con i seguenti programmi.

codice python publishSensore:

```
import paho.mqtt.publish as publish
import time
import sys
import struct
import json
import pigpio
from nrf24 import *

#costanti mqtt
TOPIC = "tps/sensorePotenza"
BROKER = "172.17.4.29"

#costanti sensore
PIGPIONAME='localhost'
PIGPIOPORT=8888
READINGPIPE='00001'
IDCORRETTO='BE'
DESTINATARIOCORRETTO='P001'
MIO_TIPO='S1'

# connessione a pigpiod
pi = pigpio.pi(PIGPIONAME, PIGPIOPORT)
if not pi.connected:
    print("Pigpiod non connesso. Lanciare: SUDO PIGPIOD")
    sys.exit()
```

```
# Crea l'oggetto NRF24
nrf = NRF24(pi, ce=17, payload_size=32, channel=76,data_rate=RF24_DATA_RATE.RATE_2MBPS, pa_level=RF24_PA.LOW)

# apre la pipe
nrf.set_address_bytes(5)
nrf.open_reading_pipe(RF24_RX_ADDR.P1, READINGPIPE)

#lettura pacchetto 32 byte
while True:
    if nrf.data_ready():
        pack=(struct.unpack("2s 4s 4s 2s 4s 16s",nrf.get_payload()))
        id=pack[0].decode()
        mittente=pack[1].decode()
        destinatario=pack[2].decode()
        tipo=pack[3].decode()
        valoreSensore=pack[4].decode()
        vuoto=pack[5].decode()
        if ((id==IDCORRETTO)and(destinatario==DESTINATARIOCORRETTO)):
            print(pack)
            print("id e destinatario corretti e valoreSensore = " + valoreSensore)
            s=int(valoreSensore)
            dataOra = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
            dizionario = {'DataOra': dataOra, 'Valore' : s}
            data = json.dumps(dizionario)
            publish.single(TOPIC, data, hostname=BROKER)
```

Per l’attuatore abbiamo fatto un solo topic che riceve un json.

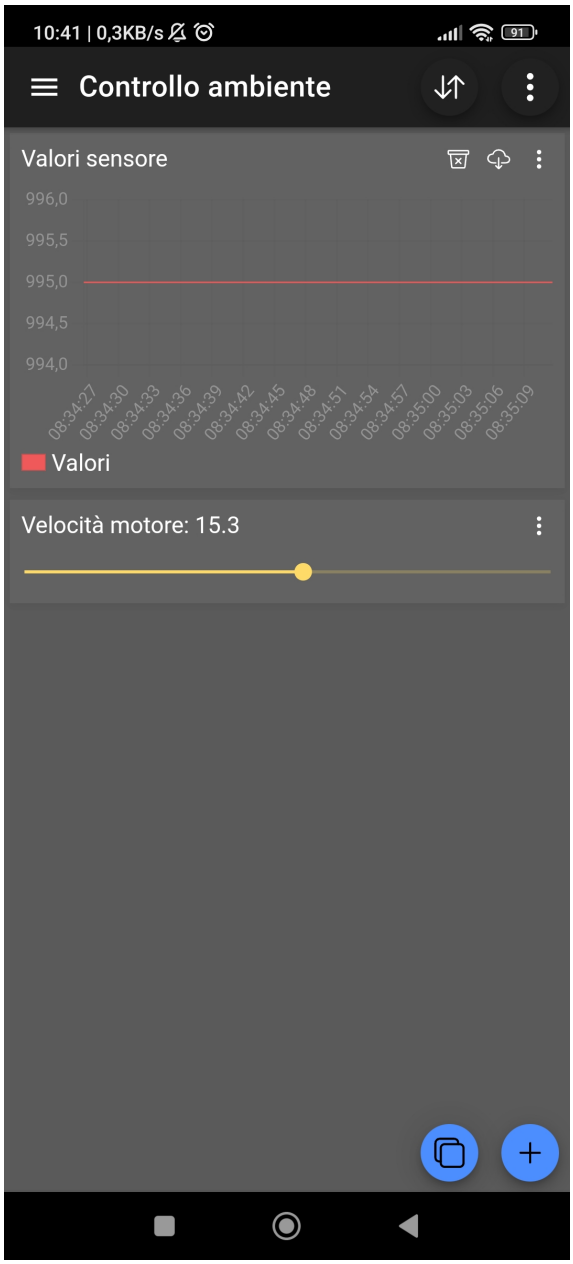
codice python subscribeAttuatore:

```
import paho.mqtt.client as client
import struct
import time
import json
import sys
import pigpio
from nrf24 import *

#costanti rf24
PIGPIONAME='localhost'
PIGPIOPORT=8888
WRITINGPIPE='00001'

#costanti motore
ID=b"BE"
MITTENTE=b"P001"
DESTINATARIO=b"A001"
TIPO=b"A1"
VUOTO=("-"*16).encode()
direzione = b"A"

#costanti mqtt
TOPIC="tps/motoreAtt"
BROKER = "172.17.4.29"

# connessione a pigpiod
pi = pigpio.pi(PIGPIONAME, PIGPIOPORT)
if not pi.connected:
    print("Pigpiod non connesso. Lanciare: SUDO PIGPIOD")
    sys.exit()
```

```
# Crea l'oggetto NRF24
nrf = NRF24(pi, ce=17, payload_size=32, channel=76,data_rate=RF24_DATA_RATE.RATE_2MBPS, pa_level=RF24_PA.LOW)

# apre le pipe
nrf.set_address_bytes(5)
nrf.open_writing_pipe(WRITINGPIPE)

def on_connect(subscriber, userdata, flags, rc):
    print("Connesso con return code" + str(rc))

    subscriber.subscribe(TOPIC)

def on_message(subscriber, userdata, msg):
    #print(msg.topic+" "+str(msg.payload))
    data = json.loads(msg.payload)
    print(data)
    dir = data["direzione"]
    direzione = str(dir).encode()
    val = data["velocita"]
    v = str(val).zfill(3).encode()
    msg=struct.pack("2s 4s 4s 2s 1s 3s 16s",ID,MITTENTE,DESTINATARIO,TIPO,direzione,v,VUOTO)
    nrf.send(msg)
    print(msg)
    nrf.wait_until_sent()
    nrf.power_up_rx()

subscriber = client.Client()
subscriber.on_connect = on_connect
subscriber.on_message = on_message

subscriber.connect(BROKER, 1883, 60)

subscriber.loop_forever()
```

Uso di una dashboard MQTT per il controllo del sensore/attuatore (parte 8)

In quest’ultima parte dell’attività abbiamo scaricato ed utilizzato l’app IoTMQTTPanel per mostrare tramite un grafico i valori ricevuti dal sensore e tramite uno slider controllare la velocità del motore.



codice python sensore:

```
import paho.mqtt.publish as publish
import time
import sys
import struct
import json
import pigpio
from nrf24 import *

#costanti mqtt
TOPIC = "tps/sensorePotenza"
BROKER = "172.17.4.29"
```

```
#costanti sensore
PIGPIONAME='localhost'
PIGPIOPORT=8888
READINGPIPE='00001'
IDCORRETTO='BE'
DESTINATARIOCORRETTO='P001'
MIO_TIPO='S1'

# connessione a pigpiod
pi = pigpio.pi(PIGPIONAME, PIGPIOPORT)
if not pi.connected:
    print("Pigpiod non connesso. Lanciare: SUDO PIGPIOD")
    sys.exit()

# Crea l'oggetto NRF24
nrf = NRF24(pi, ce=17, payload_size=32, channel=76,data_rate=RF24_DATA_RATE.RATE_2MBPS, pa_level=RF24_PA.LOW)

# apre la pipe
nrf.set_address_bytes(5)
nrf.open_reading_pipe(RF24_RX_ADDR.P1, READINGPIPE)

#lettura pacchetto 32 byte
while True:
    if nrf.data_ready():
        pack=(struct.unpack("2s 4s 4s 2s 4s 16s",nrf.get_payload()))
        id=pack[0].decode()
        mittente=pack[1].decode()
        destinatario=pack[2].decode()
        tipo=pack[3].decode()
        valoreSensore=pack[4].decode()
        vuoto=pack[5].decode()
        if ((id==IDCORRETTO)and(destinatario==DESTINATARIOCORRETTO)):
            print(pack)
            print("id e destinatario corretti e valoreSensore = " + valoreSensore)
            s=int(valoreSensore)
            dataOra = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
            dizionario = {'DataOra': dataOra, 'Valore' : s}
            data = json.dumps(dizionario)
            publish.single(TOPIC, data, hostname=BROKER)
```

codice python controlloAttuatore:

```
import paho.mqtt.client as client
import struct
import time
import sys
import pigpio
from nrf24 import *

#costanti rf24
PIGPIONAME='localhost'
PIGPIOPORT=8888
WRITINGPIPE='00001'

#costanti motore
ID=b"BE"
MITTENTE=b"P001"
DESTINATARIO=b"A001"
TIPO=b"A1"
VUOTO=("-"*16).encode()
direzione = b"A"
```

```
#costanti mqtt
TOPIC="tps/motoreAtt"
BROKER = "172.17.4.29"

# connessione a pigpiod
pi = pigpio.pi(PIGPIONAME, PIGPIOPORT)
if not pi.connected:
    print("Pigpiod non connesso. Lanciare: SUDO PIGPIOD")
    sys.exit()

# Crea l'oggetto NRF24
nrf = NRF24(pi, ce=17, payload_size=32, channel=76,data_rate=RF24_DATA_RATE.RATE_2MBPS, pa_level=RF24_PA.LOW)

# apre le pipe
nrf.set_address_bytes(5)
nrf.open_writing_pipe(WRITINGPIPE)

def on_connect(subscriber, userdata, flags, rc):
    print("Connesso con return code" + str(rc))
    subscriber.subscribe(TOPIC)

def on_message(subscriber, userdata, msg):
    print(msg.topic+" "+str(msg.payload))
    val = float(msg.payload.decode())
    val = int(val)
    if val < 0:
        direzione = b"I"
        vel = ~int(val)+1
        v = str(vel).zfill(3).encode()
        msg=struct.pack("2s 4s 4s 2s 1s 3s 16s",ID,MITTENTE,DESTINATARIO, TIPO, direzione, v, VUOTO)
    else:
        direzione = b"A"
        v = str(val).zfill(3).encode()
        msg=struct.pack("2s 4s 4s 2s 1s 3s 16s",ID,MITTENTE,DESTINATARIO, TIPO, direzione, v, VUOTO)
    nrf.send(msg)
    print(msg)
    nrf.wait_until_sent()
    nrf.power_up_rx()

subscriber = client.Client()
subscriber.on_connect = on_connect
subscriber.on_message = on_message

subscriber.connect(BROKER, 1883, 60)

subscriber.loop_forever()
```

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/33a9b821-d922-41d8-ae37-ac07542d8a56/VID-20230420-WA0003.mp4