UNIVERSITÀ
DEGLI STUDI
DI UDINE

MQTT in Python

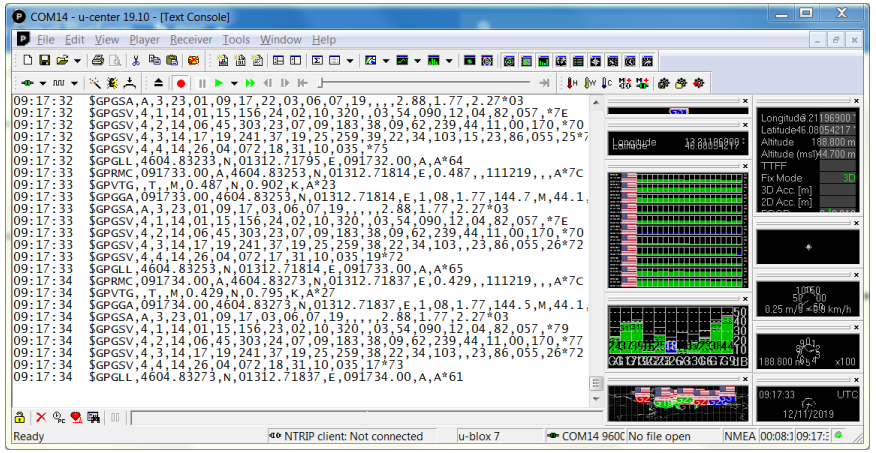
Esempio

Un esempio con MQTT

- L'esempio consente ai publisher di estrarre dati da porte seriali (che possono essere collegate a sensori) e di pubblicarli grazie ad un broker.
- La libreria Python utilizzata è **`paho-mqtt`** (o semplicemente **`mqtt`** nelle versioni più recenti).
- Il broker utilizzato è Eclipse Mosquitto.

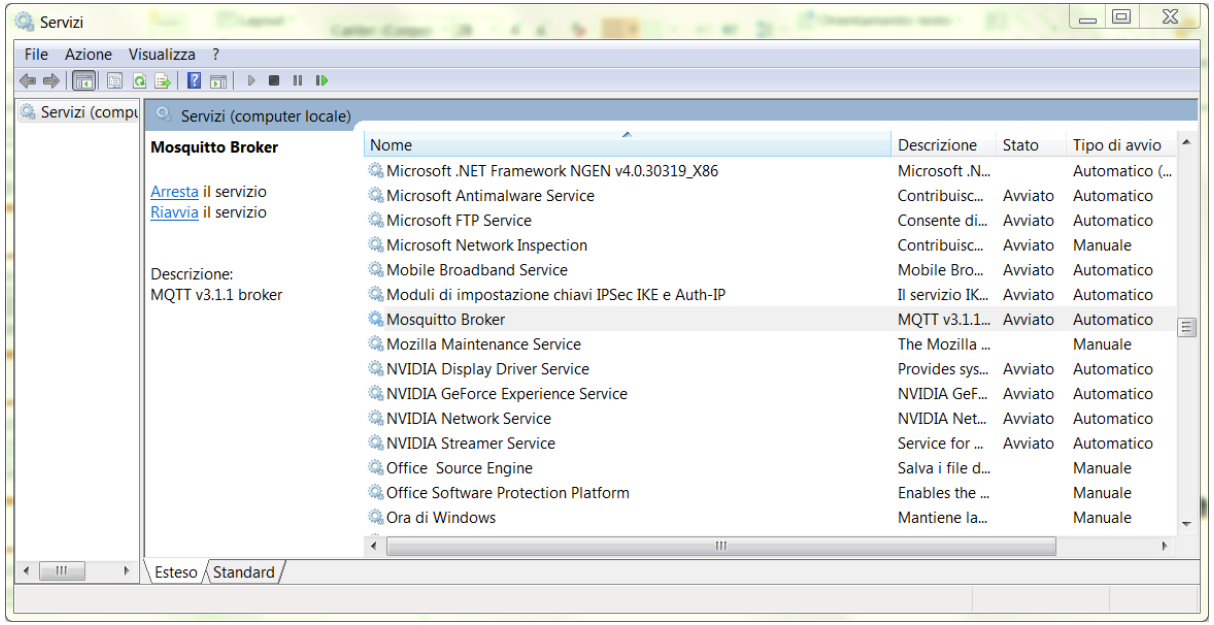
Il sensore

- Utilizziamo un GPS collegabile alla porta USB di un PC: VK-162 G-Mouse USB GPS Dongle.
- Per testarne il funzionamento si può scaricare ed installare il software u-center della u-blox: https://www.u-blox.com/en/product/u-center

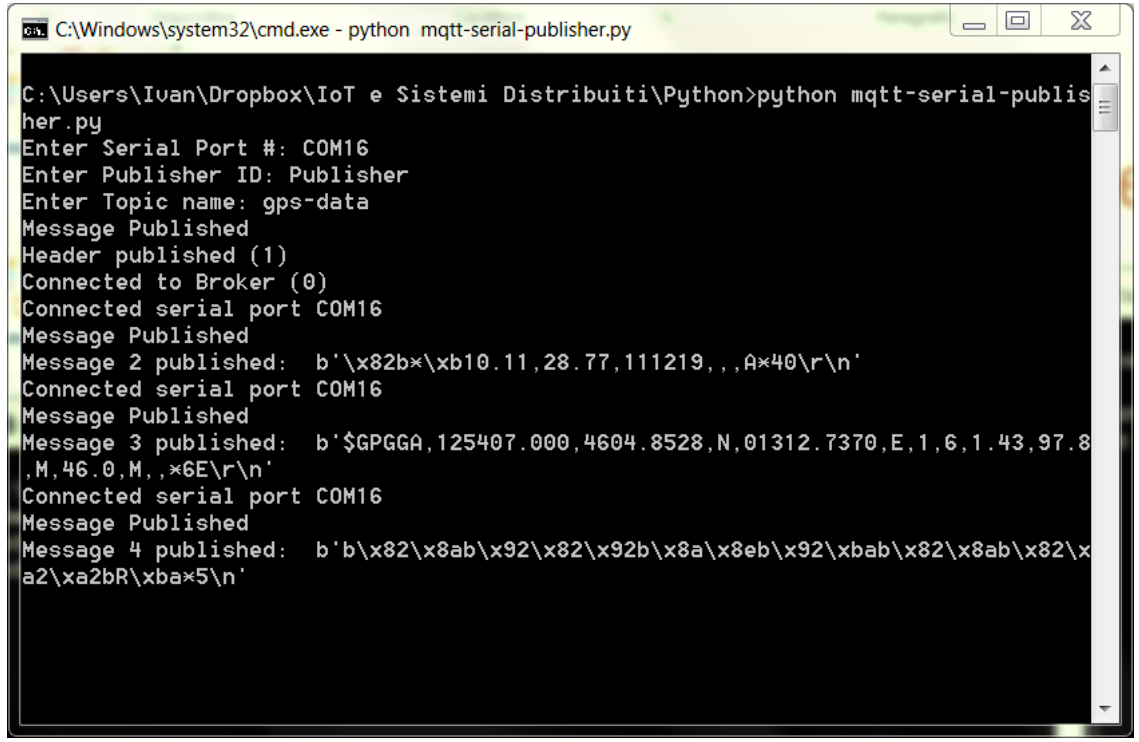


Avviare il broker MQTT

- Scaricare ed installare Eclipse Mosquitto:
  https://mosquitto.org/download/
- Avviare il servizio (in Windows utilizzare **`services.msc`**).

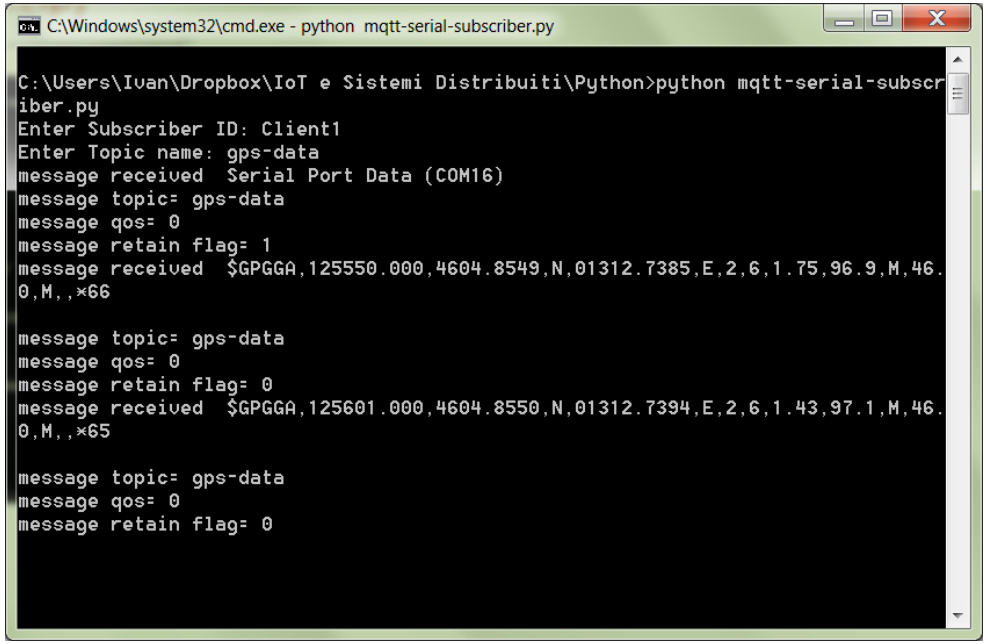


Esempio di sessione
(scaricare mqtt-python.zip)

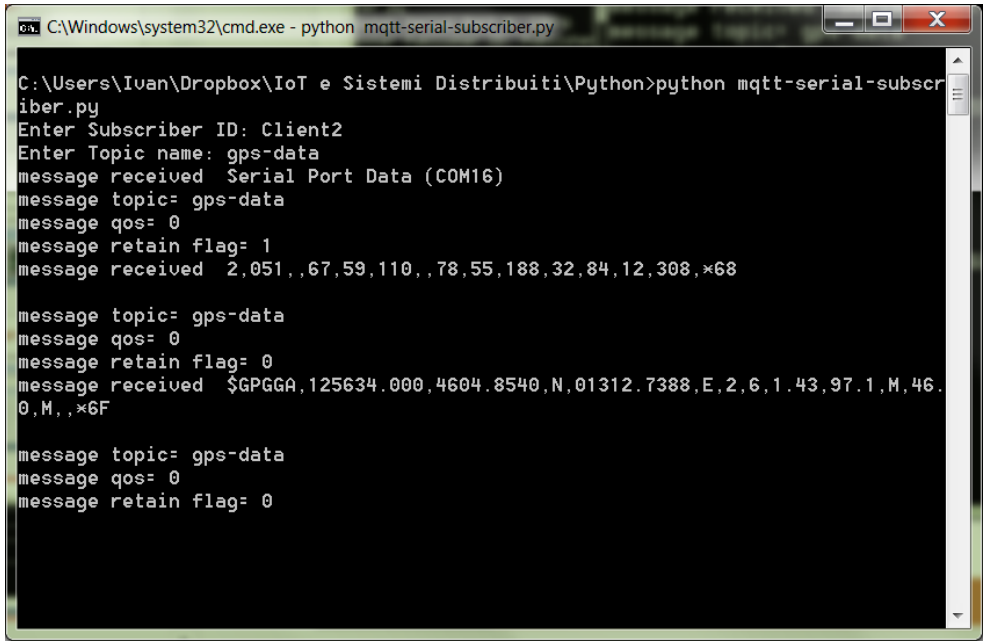


Publisher

Subscriber 1



Subscriber 2



Codice del publisher (I)

```
import paho.mqtt.client as mqtt #import the client
import serial
from time import sleep
import signal
import sys

# callback function
def connect_msg(client, userdata, flags, rc):
    global flag_is_connected
    print('Connected to Broker (%s)' %str(rc))
    flag_is_connected=True

# callback function
def publish_msg(client, userdata, mid):
    print('Message Published')
```

Queste funzioni verranno richiamate automaticamente, rispettivamente, al momento della **connessione al broker** e ogni volta che verrà **pubblicato un messaggio**.

Codice del publisher (II)

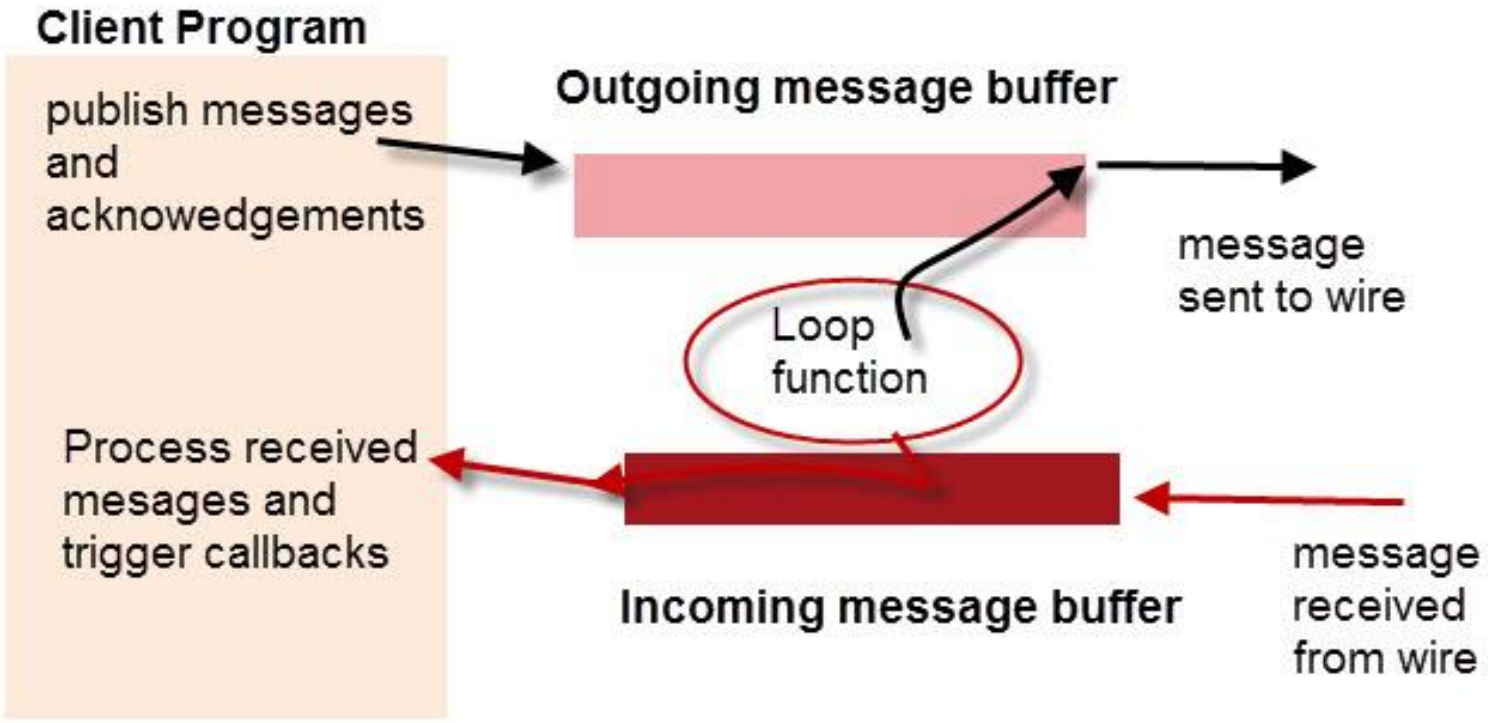
```
def serial_connect(ser, serial_port):
        ser_available=False
        while not ser_available:
                try:
                        ser.port=serial_port
                        ser.timeout=10
                        ser.open() # open serial port
                        print('Connected serial port '+ser.name)
                        ser_available=True
                except:
                        print(serial_port+" busy... Retrying...")
                        sleep(1)
                        ser_available=False

def signal_handler(sig, frame):
                global ser, client, flag_is_connected
                print('Ctrl+C detected!')
                if ser.is_open:
                        ser.close()
                if flag_is_connected:
                        client.disconnect()
                        flag_is_connected=False
                sys.exit(0)
```

Gestione della connessione/riconnessione alla porta seriale

Funzione di callback che viene richiamata quando l'utente preme Ctrl+C, oppure arriva un SIGINT.

Gestione del "loop" nei client MQTT



Paho Python MQTT Client - Loop Function Illustration

Codice del publisher (III)

```
flag_is_connected=False
qos=0                          # Quality Of Service: 0 - at least once ("fire & forget")
retain=True
signal.signal(signal.SIGINT, signal_handler) # Registration of SIGINT (Ctrl+C) handler
serial_port=input("Enter Serial Port #: ")
pub_id=input("Enter Publisher ID: ")
topic=input("Enter Topic name: ")
client = mqtt.Client(client_id=pub_id) # Creating client
client.on_connect = connect_msg        # Connecting callback functions
client.on_publish = publish_msg        # Connecting callback functions
client.connect("127.0.0.1",1883)       # Connect to broker
client.loop()
(result,mid) = client.publish(topic,'Serial Port Data (%s)' %serial_port, qos, retain)

if result==0:
        print('Header published (%d)' %mid)

retain=False
client.loop()
```

Questo primo messaggio viene salvato per essere inviato a tutti i subscriber per primo

Forzo il "triggering" degli eventi

Codice del publisher (IV)

```
line=''
ser=serial.Serial()
serial_connect(ser,serial_port)                                  # Trying to open serial port...

if ser!=None:
          line=ser.readline()                                    # Reading a line from the serial port

while len(line)>0:                                               # Looping until there are some lines to read...
          (result,mid) = client.publish(topic,line,qos,retain)   # Publishing a message with topic

          if result==0:
                    print('Message %d published: ' %mid,line)

          client.loop()                                          # Running a loop
          ser.close()                                            # Closing serial port
          sleep(5)                                               # 5 sec. pause
          serial_connect(ser,serial_port)                        # Trying to open serial port...
          line=ser.readline()                                    # Reading a line from the serial port

if ser!=None:
          ser.close()                                            # Closing serial port

client.disconnect()                                              # Disconnecting
```

Codice del subscriber (I)

```
import paho.mqtt.client as mqtt #import the client
import signal
import sys

# callback function
def process_message(client, userdata, message):
    print("message received ", str(message.payload.decode("utf-8")))
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)

def signal_handler(sig, frame):
                    global client, flag_is_connected
                    print('Ctrl+C detected!')

                    if flag_is_connected:
                              client.disconnect()
                              client.loop_stop()
                    sys.exit(0)

flag_is_connected=False
qos=2 # Quality Of Service (Exactly once)

signal.signal(signal.SIGINT, signal_handler)
```

Funzione di callback che viene richiamata ogni volta che viene ricevuto un messaggio dal broker

Funzione di callback che viene richiamata quando l'utente preme Ctrl+C, oppure arriva un SIGINT.

Codice del subscriber (II)

```
broker_address="127.0.0.1"

sub_id=input("Enter Subscriber ID: ")
topic=input("Enter Topic name: ")

# Create client
client = mqtt.Client(client_id=sub_id)

# Assign callback function
client.on_message = process_message

# Connect to broker
client.connect(broker_address,1883,60)

flag_is_connected=True

# Subscriber to topic
client.subscribe(topic,qos)

# Run loop
client.loop_forever() # Automatic loop handling (the client is "blocked" within).
```

MQTT vs. HTTP: conviene?

- Il costo maggiore per il protocollo MQTT è rappresentato dallo stabilimento della connessione e dalla sua chiusura.
- Quindi conviene, rispetto a protocolli come l'HTTP, quando bisogna inviare molti dati usando la stessa connessione, tenuta aperta per tutta la durata della comunicazione.
- Per i dettagli si veda l'articolo seguente:

  https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols