CAPITOLO 9 VOLUME 2

ESEMPIO PARAGRAFO 3.3 Pubblicazione via MQTT del dati di temperatura acquisiti da Arduino MKR WiFi 1010 con sensore DHT11 impiegando un broker MQTT privato

Si procede nel seguente modo:
a) si installa il broker/server MQTT privato (*mosquitto*);
b) si configura l'IDE di Arduino per operare con la scheda Arduino MKR WiFi 1010;
c) si scrive lo sketch che consente di impiegare Arduino MKR WiFi 1010 come client MQTT *publisher* connesso in rete in modo wireless, via WiFi; si testa il corretto funzionamento del sistema utilizzando un client MQTT *subscriber* per ricevere i messaggi pubblicati attraverso il broker MQTT privato;
d) opzionalmente si configura un accesso al broker MQTT protetto da username e password

a) Installazione del broker/server MQTT privato

Installiamo un broker/server MQTT su un microcomputer *Raspberry Pi* (o su un PC con sistema operativo LINUX *Ubuntu*). Il pacchetto software che fornisce tale funzionalità è *mosquitto* (https://mosquitto.org).
L'installazione è molto semplice in quanto il pacchetto è già presente nel repository principale, per cui è sufficiente dare i seguenti comandi (con i diritti di amministratore):

```
onelio@raspberrypi:~ $ sudo apt-get update
onelio@raspberrypi:~ $ sudo apt-get install mosquitto
```

Per verificare la corretta installazione e attivazione del broker MQTT *mosquitto* è sufficiente dare il comando:

```
onelio@raspberrypi:~ $ sudo service mosquitto status
```

Se si desidera semplicemente testare il corretto funzionamento del broker MQTT *mosquitto* non è necessario effettuare modifiche alla sua configurazione di default (il file di configurazione è `mosquitto.conf` il quale è contenuto nella directory `/etc/mosquitto`).
Per poter contattare in rete il broker MQTT privato è necessario conoscerne l'indirizzo IPv4.
Di default l'indirizzo IP di Rasberry Pi è dinamico e viene assegnato da un server DHCP. In questo caso l'indirizzo IP (indicato come *`inet`*, *internet address*) può essere visualizzato aprendo un terminale locale e digitando il comando `ifconfig` (oppure `ip address`). Se si desidera operare da remoto (via SSH con un programma come PuTTY) è possibile utilizzare da PC un tool di scansione delle reti, per esempio *zenmap* (https://nmap.org/zenmap), per scoprire qual è l'indirizzo IPv4 assegnato a Raspberry Pi.

In alternativa è possibile configurare un indirizzo IPv4 statico editando il file `/etc/dhcpcd.conf`: si entra nella directory **/etc** (`cd /etc`); si apre con l'editor nano il file **dhcpcd.conf** (con i diritti di amministratore, comando `sudo nano dhcpcd.conf`); si de-commentano le righe che esemplificano l'assegnazione di un indirizzo IP statico e si modificano con i propri dati l'indirizzo IPv4 (*Static ip_address*), con la relativa subnet mask in notazione binaria (per esempio 10.0.0.222/24, in cui la subnet mask /24 è composta da 24 "1" e 8 "0" e corrisponde alla classica subnet mask 255.255.255.0), l'indirizzo del router tramite cui si esce dalla rete IP di appartenenza (cioè il default gateway, indicato come *Static routers*) e i server DNS (*Static domain_name_servers*).

b) Configurazione di Arduino MKR WiFi 1010

Come client MQTT *publisher* si impiega la scheda **Arduino MKR WiFi 1010,** con sensore di umidità e temperatura DHT11, FIGURA 1, che ha già a bordo un modulo radio Wi-Fi e quindi può essere collegata in rete in modo wireless.

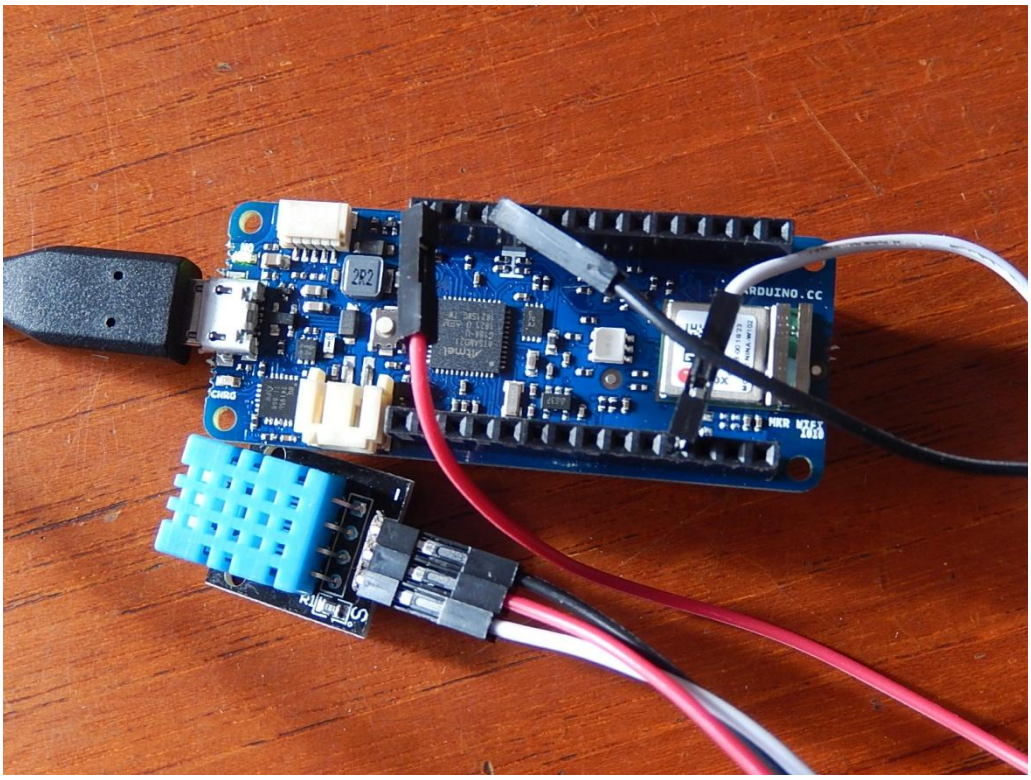

FIGURA 1 Hardware del client MQTT publisher: Arduino MKR WiFi 1010 con sensore DHT11

Il collegamento del sensore DHT11 alla scheda Arduino MKR WiFi 1010 è analogo a quello illustrato nel PARAGRAFO 3.2. Per quanto concerne il software si può procedere nel seguente modo.

1. Si installa il pacchetto per la scheda Arduino MKR WiFi 1010, cliccando su **Strumenti → Scheda → Gestore schede**, cercando e installando il pacchetto per le schede **Arduino SAMD Board**s.
2. Si seleziona tra le schede disponibili la scheda Arduino MKR WiFi 1010 e la relativa porta di comunicazione.
3. Selezionando **Sketch → #include libreria → Gestione librerie** si cerca e si installa la libreria **WiFiNINA** che inserita negli sketch mette a disposizione i metodi per la comunicazione via WiFi di Arduino MKR WiFi 1010. Cliccando su *File → Esempi → WiFiNINA* si hanno a disposizione degli sketch che esemplificano l'impiego della libreria **WiFiNINA** (tra cui l'esempio *ConnectWithWPA*). Per ulteriori informazioni si rimanda al sito ufficiale di Arduino.
4. Si scrive lo sketch che realizza il client MQTT *publisher* connesso in rete in modo wireless.

c) Sketch del client MQTT publisher

Lo sketch è simile a quello del PARAGRAFO 3. 2 per quanto concerne l'acquisizione dei dati dal sensore DHT11, ma ovviamente si differenzia per il tipo di connettività utilizzata (wireless) e per il fatto che il broker MQTT è privato. Un estratto dello sketch che realizza un client MQTT *publisher* è il seguente (deve ovviamente essere noto l'indirizzo IPv4 del broker MQTT). Lo sketch completo è scaricabile dalle risorse online del testo.

```
//------- librerie per la connessione WiFi, al broker MQTT e per
il sensore DHT11
#include <SPI.h>
#include <WiFiNINA.h>
#include <PubSubClient.h>
#include <DHT.h>
#include <DHT_U.h>
//-----------------------------------------------------------------
---------------------
                              ……

// Il nome della rete WiFi (SSID) e la password WPA sono nel file
arduino_secrets.h
#include "arduino_secrets.h"
char ssid[] = MIO_SSID;
char pass[] = MIA_PASS;
// --- inserire qui l'indirizzo IP del server/broker privato -----
------------
byte brokerMQTT [] = {10, 0, 0, 222 };  //indirizzo IP del broker
MQTT privato
#define nomebroker "10.0.0.222"    //indirizzo IP in formato
carattere
#define NOMETOPIC "LAB-IOT/ARDUINO-DHT11"    //nome del topic
#define CLIENTID "Arduino-MKR1010"            //identificativo
scelto
//-----  Temperatura e umidità in formato carattere per la
pubblicazione ------
char * carTemperatura;
char * carUmid;
//---------------  Inizializzazione dei client wifi e MQTT
WiFiClient Client_wifi;
PubSubClient arduinoClient(brokerMQTT,1883,Client_wifi);
                         ……..
}
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
void setup() {
  dht.begin();  // Inizializza il sensore
  Serial.begin(9600); //inizializza il monitor seriale
  beginConnection() ; //Effettua la connessione al server/broker
MQTT
}
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
void loop(void) {
  temp = dht.readTemperature();    // legge la temperatura dal
sensore DHT11
  carTemperatura = converti(temp); //valore di temperatura in
formato carattere
// Trasmetti al broker MQTT un messaggio e la temperatura rilevata
del sensore
```

```
  arduinoClient.publish(NOMETOPIC, " Temperatura (°C) misurata dal
sensore DHT11 su Arduino MKR WIFI 1010:", true) ;
  arduinoClient.publish(NOMETOPIC, carTemperatura, true);
                                ……..
  umid = dht.readHumidity(); //legge l'umidità
  carUmid = converti(umid); //valore di umidità in formato
carattere
// Trasmetti al broker MQTT un messaggio e l'umidità rilevata del
sensore
  arduinoClient.publish(NOMETOPIC, " Umidita' (%) misurata dal
sensore DHT11 su Arduino MKR WIFI 1010:", true);
  arduinoClient.publish(NOMETOPIC, carUmid, true) ;
  Serial.print("Umidità (%) misurata dal sensore DHT11 su Arduino
MKR WIFI 1010:  ");
  Serial.println(carUmid);
  Serial.println("-------------------------------------------------
------------------");
  delay(10000); //aspetta 10 secondi prima di inviare delle nuove
misure
}
//----  Funzione che gestisce la connessione con il broker/server
MQTT --------------
void beginConnection() {
  WiFi.begin(ssid, pass);
  delay(10000);
  int RichiestaConn = arduinoClient.connect(CLIENTID) ;
                              ……..
  }
```

Per evitare di inserire direttamente nello sketch il nome della rete WiFi (SSID) e la password Wi-Fi è possibile creare e includere nello sketch un file qui denominato `arduino_secrets.h`, il quale è contenuto nella stessa cartella dello sketch, che contiene le direttive con cui si associano a MIO_SSID e MIA_PASS gli effettivi SSID e password, cioè le due righe seguenti:

```
#define MIO_SSID "inserire qui l'SSID"
#define MIA_PASS "inserire qui la password"
```

d) Accesso al broker MQTT protetto da username e password

Per proteggere con username e password l'accesso al broker MQTT *mosquitto* si possono eseguire le seguenti operazioni su Rasberry Pi:

1. si entra nella directory `/etc/mosquitto/conf.d` con il comando
   `cd etc/mosquitto/conf.d`
2. con i diritti di amministratore si crea un file contente la/e coppia/e **username:password** valide per l'accesso a broker MQTT, per esempio denominandolo userpw.txt[1]:
   `sudo nano userpw.txt`
   `miouser:miapassword`
3. si crittografa il file con il comando: `sudo mosquitto_passwd -U nomedelfile` (es. userpw.txt)

[1] Con l'editor Linux *nano* si salva premendo i tasti control (ctrl) O e si esce premendo ctrl X

4. Si crea nella directory `/etc/mosquitto/conf.d` un file con estensione `.conf` (esempio con `sudo nano chiedipwd.conf`), inserendo in quel file le due righe seguenti:

```
allow_anonymous false
password_file /etc/mosquitto/conf.d/nomedelfile
```
(percorso e nome del file)

5. Si riavvia il servizio *mosquitto* con il comando: `sudo service mosquitto restart`

Nello sketch per Arduino MKR va modificata una riga della funzione che esegue la connessione, inserendo nella richiesta di connessione lo username e la password (lo username e la password possono essere nascosti operando in modo analogo a quanto fatto per SSID e password WiFi, creando e includendo nello sketch un file `credenziali.h` contenente le direttive `#define` che associano a dei nomi generici le credenziali effettive) :

```
//---------------- la parte precedente è uguale allo sketch senza
password
// -- Funzione che gestisce la connessione con il broker/server
MQTT -------------------
void beginConnection() {
  WiFi.begin(ssid, pass);
  delay(10000); ////aspetta (10 s) che che il client si connetta e
il server DHCP assegni l'indirizzo IP
  int RichiestaConn =
arduinoClient.connect(CLIENTID,"pippo","1234") ;
//connessione con username e password ;
  if (!RichiestaConn) {    //se la connessione non è avvenuta
RichiestaConn = 0 segnala ciò
    Serial.println(RichiestaConn) ;
    Serial.println("Non riesco a connettermi al server/broker
MQTT");
    Serial.println("Controllare la configurazione di Arduino");
    delay(100);
    exit(-1);
  }
  else {

Serial.println("************************************************
******************");
    Serial.print("Connesso al server/broker MQTT:  ");
    Serial.println(nomebroker);
    Serial.print("Invio dei dati al topic MQTT:  ");
    Serial.println(NOMETOPIC);

Serial.println("************************************************
******************");
  }
}
```

Infine per verificare il corretto funzionamento del sistema si deve utilizzare un client MQTT *subscriber* che consenta di inserire username e password, come quello disponibile tra le risorse online del testo.
Per esemplificare meglio l'impiego del protocollo MQTT nella comunicazione tra schede Arduino e broker MQTT, tra le risorse online del Capitolo 9 sono disponibili diversi sketch per *client MQTT publisher* che corrispondono ai seguenti casi:

- Arduino Uno con shield Ethernet e sensore LM35 che si connette al broker pubblico test.mosquitto.org via Internet;
- Arduino Uno con shield Ethernet e sensore DHT11 che si connette al broker pubblico test.mosquitto.org via Internet;
- Arduino MKR WiFi 1010 e sensore DHT11 connesso in rete via WiFi, che si connette al broker pubblico test.mosquitto.org via Internet;

- Arduino Uno con shield Ethernet e sensore DHT11 che si connette a un proprio broker privato (non protetto);
- Arduino Uno con shield Ethernet e sensore DHT11 che si connette a un proprio broker privato protetto con username e password;
- Arduino MKR WiFi 1010 e sensore DHT11, connesso in rete via WiFi, che si collega a un proprio broker privato (non protetto);
- Arduino MKR WiFi 1010 e sensore DHT11, connesso in rete via WiFi, che si collega a un proprio broker privato (non protetto).

Sono poi disponibili diversi esempi di *client MQTT subscriber* scritti in Python:

- senza la possibilità di scegliere broker e topic all'avvio del client (che così una volta editato con l'indirizzo del broker e del topic impiegati e lanciato non necessita di alcun intervento);
- con la possibilità di scegliere broker e topic all'avvio del client per il collegamento a un broker non protetto;
- con la possibilità di scegliere broker e topic all'avvio del client per il collegamento a un broker protetto da username e password.

Tutti i client producono un file di testo che contiene data dell'avvio, broker e topic a cui si è collegato, valori di temperatura o di temperatura e umidità inviati al broker dal client publisher (Arduino), si vedano le FIGURE 2 e 3 come esempio.

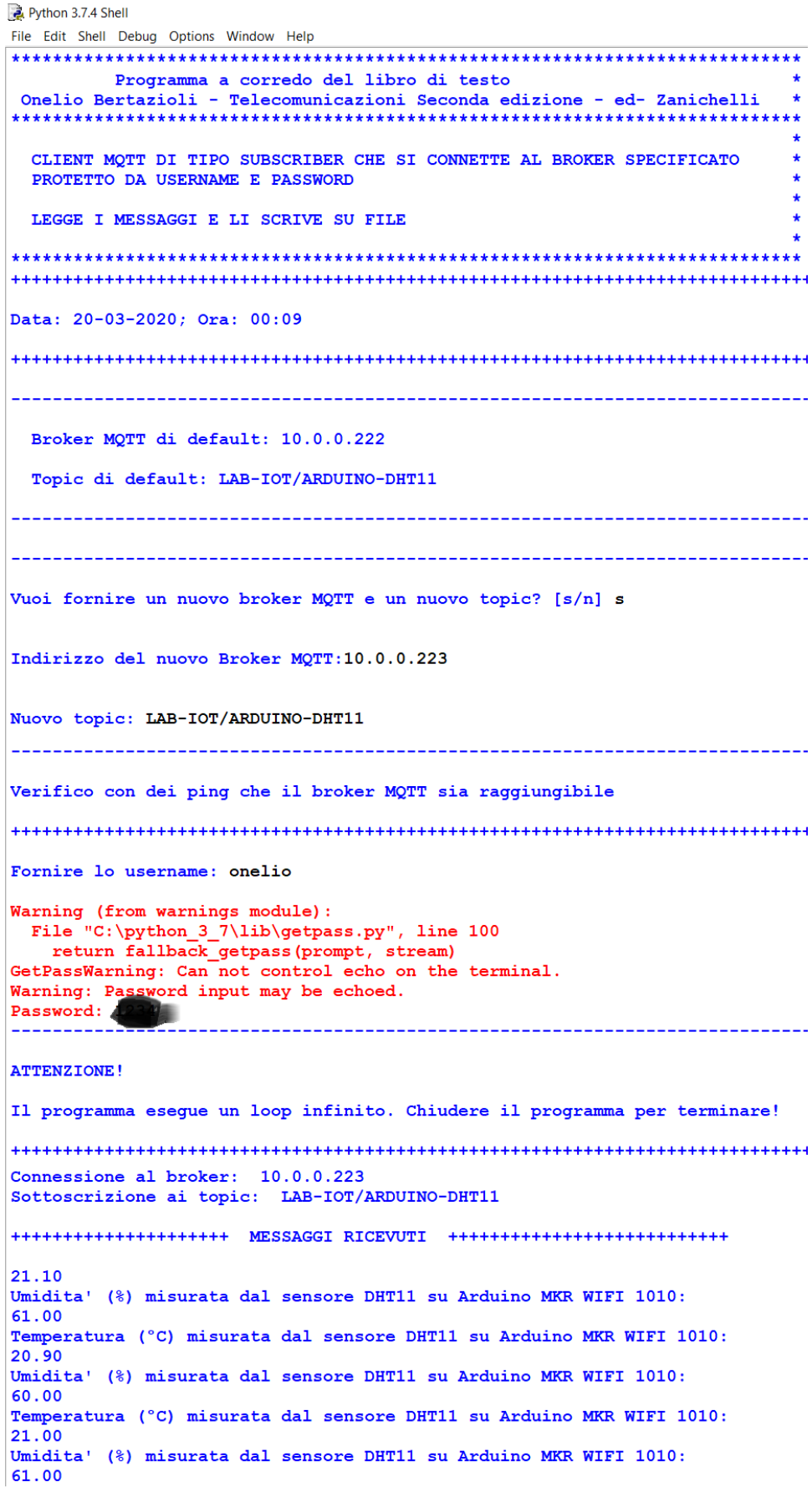

FIGURA 2 Esecuzione del programma Client con IDLE Python

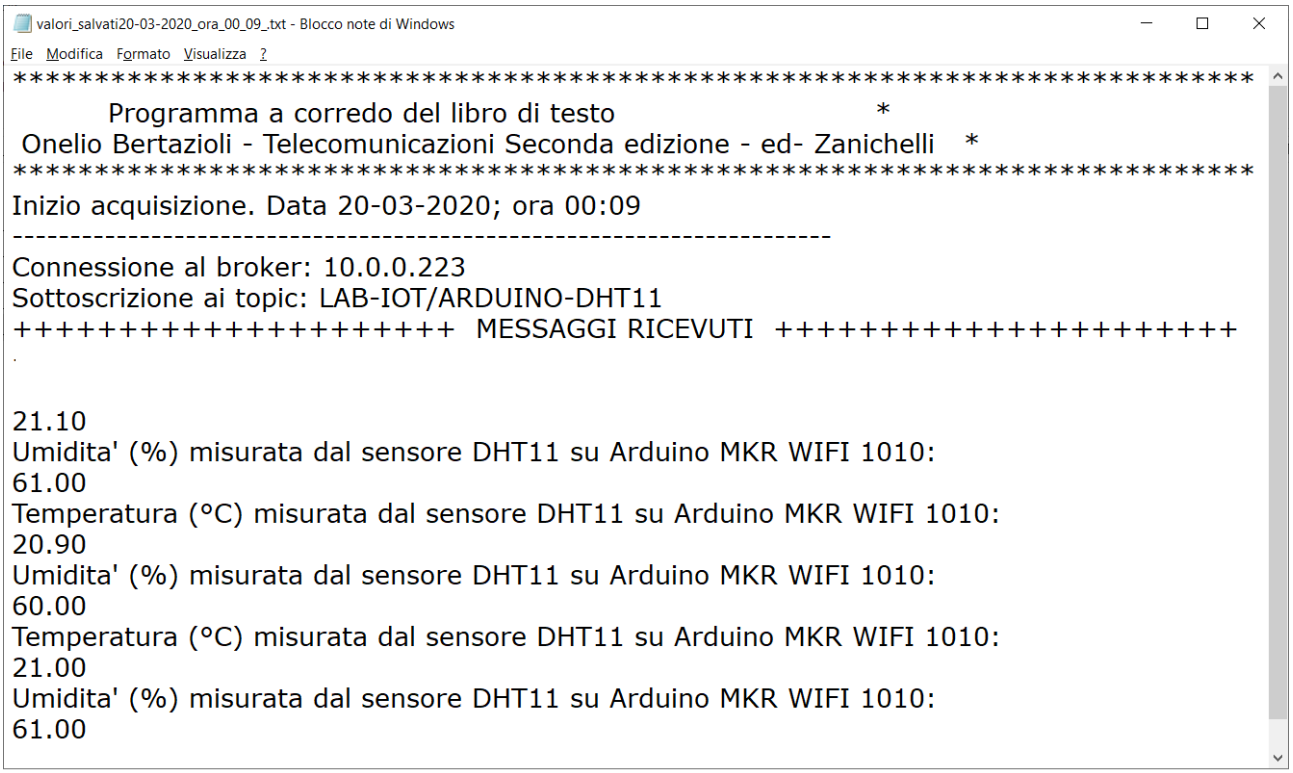

FIGURA 3 Esempio di file generato dal client MQTT