Monitoraggio remoto di umidità e temperatura con un'app Android Java utilizzando MQTT e una scheda ESP8266 NodeMcu

Esempio applicativo di Internet of Things (IoT).

Il progetto illustra come ottenere su un telefono Android i valori di umidità e temperatura di un ambiente remoto utilizzando una scheda ESP8266 NodeMCU e il protocollo di messaggistica MQTT. L"umidità e la temperatura sono rilevate da un sensore DHT11.

Che cos'è l'ESP8266

L'ESP8266 è un SoC (System on a Chip) costituito da un microcontrollore a 32 bit e un ricetrasmettitore Wi-Fi . Così come Arduino, consente di controllare ingressi e uscite e, grazie al basso costo e al modulo Wi-Fi integrato al suo interno, rappresenta una soluzione completa per la maggior parte dei progetti di "Internet of Things" (per connettersi alla rete Wifi e a Internet, per ospitare un Web Server, per controllare un oggetto, ecc.) Inoltre si può programmare facilmente anche con l'IDE di Arduino.

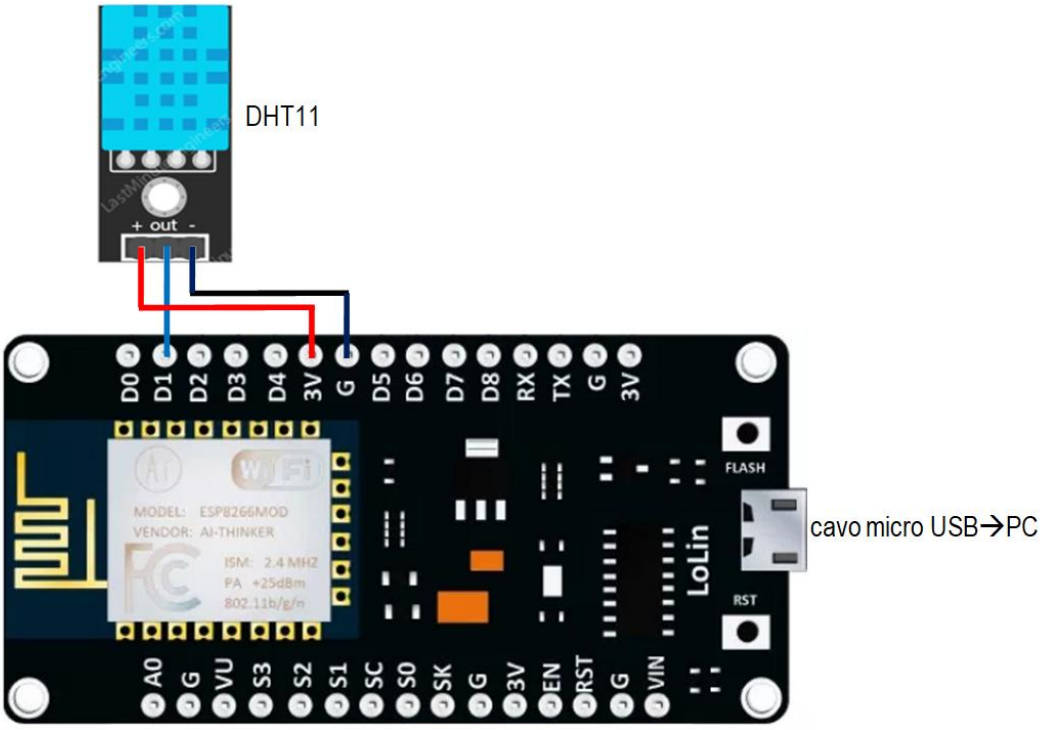
Sul mercato sono disponibili molte schede di sviluppo che utilizzano il chip ESP8266: tra queste ho scelto la scheda **NodeMCU**, forse la più diffusa e popolare sul mercato, con interfaccia USB (convertitore da USB a seriale) integrata.

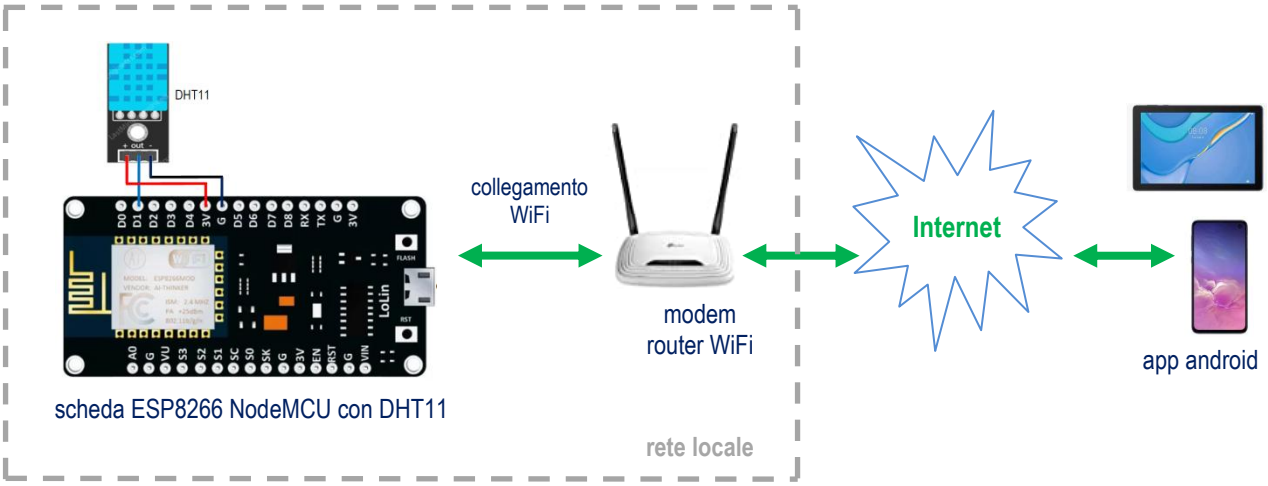
Il sensore DHT11 e i collegamenti con la scheda ESP8266 NodeMCU

Il **DHT11** è un sensore digitale di umidità e temperatura dell'aria costituito da una parte resistiva che si occupa della rilevazione dell'umidità e da un NTC che rileva la temperatura. Il sensore viene fabbricato in due configurazioni: a 3 o a 4 pin. In entrambi i casi i pin VCC, GND e OUT (DATA), che devono essere collegati ad ESP8266 NodeMCU, sono indicati chiaramente. Il pin NC, nel caso della configurazione a 4 pin, non viene collegato. Tra la linea del segnale OUT e VCC (3V) è necessaria una resistenza di pull-up da 4.7K Ohm. Nel caso di configurazione a 3 pin, utilizzata in questo progetto, la resistenza di pull-up è già montata.

Collegare il sensore di umidità e temperatura **DHT11** con l'ESP8266 NodeMCU è molto semplice

Sensore DHT11	ESP8266 NodeMCU
VCC (+)	3V
DATA (out)	D1
GND(-)	G





- La scheda ESP8266 NodeMCU si connette ad Internet tramite il modulo WiFi integrato al suo interno
- l'app android comunica con la scheda tramite la propria connessione Internet

Facendo riferimento al progetto *'Arduino: Comandare da remoto l'accensione/spegnimento di due led'*, pubblicato nella sezione Arduino dell'Area Download del sito www.maurodeberardis.it, potremmo adottare un modello Client/Server tradizionale (utilizzando classiche richieste HTTP).

Ma con un tale approccio bisogna considerare che:

1. il modem/router si collega ad Internet con un IP pubblico (ad esempio 80.182.203.153) e, grazie agli indirizzi IP privati della rete locale, instrada i vari pacchetti di dati verso ESP8266 NodeMCU e gli altri dispositivi in rete (PC desktop, notebook, stampanti di rete, ecc.)
2. la scheda ESP8266 NodeMCU con i due diodi led si collega via WiFi alla rete locale, e quindi al router, con un IP privato (ad esempio 192.168.1.123)

e quindi:

3. per accendere/spegnere da remoto i due led, occorre prima raggiungere l'indirizzo pubblico del router e poi essere instradati verso l'indirizzo privato di ESP8266.

Relativamente al punto 3 si presentano due problemi:

a) bisogna conoscere l'indirizzo IP pubblico del router che però non sempre è fisso; anzi, per le utenze domestiche è generalmente dinamico e può variare con una certa frequenza. L'utility del sito www.maurodeberardis.it *'Mio Numero Ip'* oggi mi dice che l'IP pubblico della mia rete è 80.182.203.153, domani tale indirizzo potrebbe essere un altro. La soluzione migliore è ovviamente quella di attivare un IP fisso sul nostro router, ma questa strada è costosa e spesso impraticabile. In alternativa possiamo utilizzare il servizio di DNS dinamico che diversi siti forniscono gratuitamente

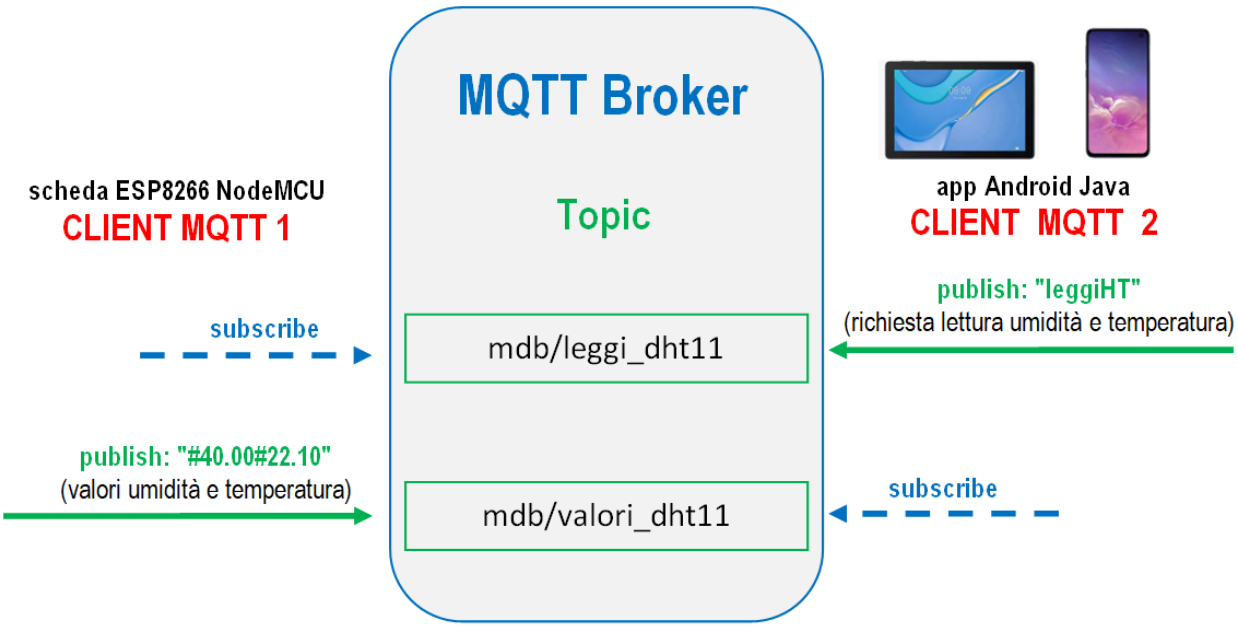
b) occorre configurare il router in modo che, una volta raggiunto da remoto attraverso l'IP pubblico, stabilisca una comunicazione con la scheda ESP8266 NodeMCU attraverso la porta 80. Per fare questo occorre impostare sul router un "port forwarding", ovvero una nuova regola di routing per far si che ogni volta che da remoto giunge una richiesta http sulla porta 80, questa venga reindirizzata all'IP privato a cui è associata la scheda che vogliamo controllare (nel nostro caso di esempio: 192.168.1.123)

Oltre ai problemi sopraelencati, bisogna considerare che il protocollo HTTP è per sua natura poco adatto ad operare nel'IoT principalmente per il fatto che è pieno di regole e troppo pesante per adattarsi ai dispositivi IoT che consumano pochissima energia e spesso hanno una debole connettività di rete ('*HTTP è ok per Internet del Vecchio Mondo ma non per il New world Internet (IoT)!*')

In alternativa ad HTTP e seguendo l'orientamento generale dell'industria, scegliamo di utilizzare il **protocollo MQTT**, un protocollo di messaggistica molto leggero ed efficiente comunemente usato per le applicazioni Internet of Things.

MQTT (acronimo di Message Queuing Telemetry Transport) si basa su un meccanismo di **publish/subscribe** (pubblicazione/sottoscrizione) di messaggi ed opera attraverso un **message broker** in grado di gestire contemporaneamente decine di migliaia di client (dispositivi o applicazioni). Attraverso il protocollo publish/subscribe, i dispositivi e le applicazioni client pubblicano messaggi e sottoscrivono argomenti (**topics**) gestiti da un broker.
I mittenti inviano messaggi relativi a topics specifici, il broker provvede alla trasmissione dei messaggi ai destinatari che hanno sottoscritto gli stessi topic

- sia i mittenti ('editori') che i destinatari ('abbonati') sono **client MQTT**, non si contattano mai direttamente e possono comunicare esclusivamente attraverso il message broker
- ogni client può iscriversi a molteplici topic
- quando viene pubblicato un nuovo messaggio che riguarda un topic, il message broker lo distribuisce a tutti i client iscritti a quel determinato topic



In questo progetto ci sono due client MQTT: la scheda ESP8266 NodeMCU con il sensore DHT11 e l'app android che visualizza temperatura e umidità rilevate dalla scheda tramite il sensore.

- l'app android pubblica sul topic "mdb/leggi_dht11" il messaggio "leggi" con il quale richiede la lettura dei valori remoti di umidità e temperatura: il broker trasmette il messaggio alla scheda (che ha sottoscritto il topic "mdb/leggi_dht11")
- la scheda, ricevuto il messaggio "leggi", rileva i valori di umidità e temperatura attraverso il sensore DHT11 e li pubblica sul topic "mdb/valori_dht11": il broker trasmette questi valori all'app android (che ha sottoscritto il topic "mdb/valori_dht11":)

L'analisi del codice dei due clients MQTT charirà meglio il meccanismo di pubblicazione/sottoscrizione che consente all'app di monitorare via Internet i valori di temperatura e umidità.

La lettura è asincrona, ovvero viene fatta a seguito di una esplicita richiesta dell'app android: in alternativa potrebbe essere programmata ad intervalli di tempo regolari, ad esempio ogni 30 secondi. La modalità asincrona suggerisce come sia possibile estendere il progetto ad altre funzionalità aggiuntive, quali la lettura di ulteriori sensori o il setting on/off di motori, caldaie, allarmi, ecc.

Client MQTT: ESP8266 NodeMCU

Utilizziamo un Client C++ con Arduino IDE

Componenti utilizzati

Scheda ESP8266 NodeMCU collegata ad un PC Windows X con cavo micro USB
Breadboard con cavetteria
Sensore DHT11
Android Studio e smartphone Android

Utilizzo dell'IDE di Arduino per programmare ESP8266 NodeMCU

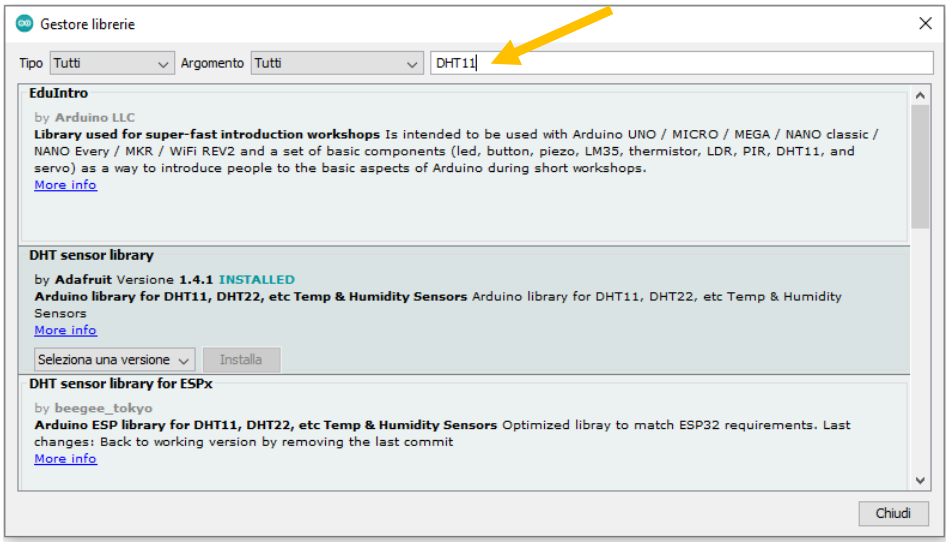
Chi legge questo tutorial, probabilmente è in grado di programmare la scheda ESP8266 NodeMCU con l'IDE di Arduino. Se non è cosi, può far riferimento al progetto *'Web Server ESP8266 NodeMCU'* o al progetto *' ESP8266, RC522 e MySql: Sistema di validazione remota di un tag RFID'* pubblicati nella sezione "Arduino" dell'Area Download del sito www.maurodeberardis.it

La Libreria PubSubClient

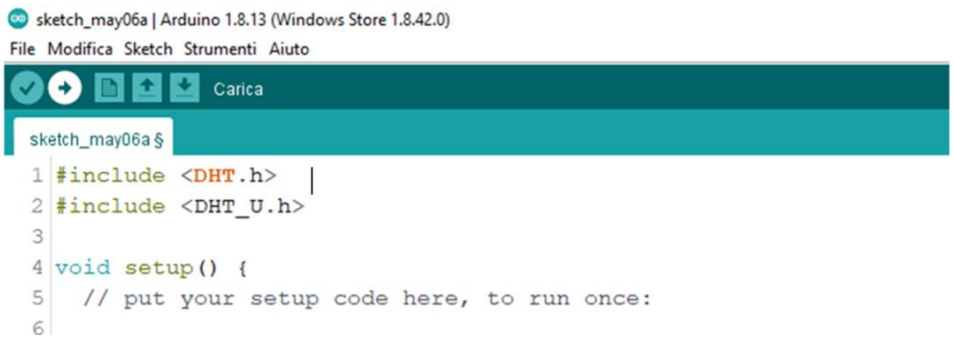
Per questo progetto è necessario utilizzare la **libreria PubSubClient** che consente al client di connettersi con il broker e di pubblicare/sottoscrivere i messaggi di un topic.
Dopo aver scaricato (*da gui*) il file .zip della libreria PubSubClient e averla installata, creare un nuovo sketch e selezionare la Scheda ModeMCU (ESP-12 Module).

La libreria necessaria per utilizzare il sensore di umidità e temperatura DHT11

- Con l'IDE Arduino in esecuzione, scegliere Strumenti-Gestione librerie…
- Quando si apre la finestra del Gestore librerie, nel campo di ricerca inserire "DHT11" e tra le diverse librerie proposte trovare, selezionare e installare "DHT sensor library" (by Adafruit)



- Aprire lo sketch che si vuole programmare, scegliere l'opzione *Sketch-#include libreria* e selezionare "DHT Sensor Library" e nello sketch vengono incluse due librerie (DHT_U.h non è necessaria e se si vuole si può cancellare)



Lo sketch *'esp8266_dht11_mqtt.ino'*

```
/*
 * ---------------------------------------------------------
 * Prof. Mauro De Berardis 2021
 * ---------------------------------------------------------
 * Il progetto illustra  come ottenere su un telefono Android i valori di temperatura e umidità
 * di un ambiente remoto utilizzando una scheda ESP8266 NodeMCU e il protocollo di messaggistica
 * MQTT. L''umidità e la temperatura  sono rilevate da un sensore DHT11.
 * Lo sketch realizza il client MQTT della scheda ESP8266 NodeMCU
 */
#include <ESP8266WiFi.h>
//La libreria ESP8266WiFi permette la connessione della scheda  alla rete WiFi

#include <PubSubClient.h>
/*La libreria PubSubClient consente al client di connettersi al broker MQTT e di
 *pubblicare messaggi e/o sottoscrivere topic
*/
#include <DHT.h>      //libreria necessaria per utilizzare il sensore DHT11
#define sensorePin D1
// pin di ESP8266 NodeMCU collegato all'uscita (pin OUT) del sensore DHT11
#define tipoDHT DHT11
/* definisce il tipo di sensore della famiglia DHT:  in questo caso viene selezionato
   DHT11, ma esistono sono altri sensori quali DHT21 e DHT22
*/
DHT dht(sensorePin,tipoDHT);  //Istanzia l'oggetto dht della classe DHT
float H;          //umidità
float T;         //temperatura
//***Configurazione WiFI: impostazione delle credenziali di rete
const char* ssid = "MySSID";         //Inserire qui l'SSID
const char* password = "MyPassword"; //Inserire qui la password WiFi
// MQTT Broker
/*
 * Per questo progetto, utilizziamo un server broker pubblico e gratuito  per l'apprendimento,
 * il test  e  la prototipazione MQTT. Qualsiasi dispositivo può pubblicare e sottoscrivere
 * argomenti su di esso e non esiste una protezione della privacy. Non usarlo mai in produzione
 */
const char  *mqttServer = "broker.emqx.io";
const char  *mqttUser = "emqx";
const char  *mqttPassword = "public";
const int    mqttPort = 1883;
const char  *topic =  "mdb/leggi_dht11";


//Creazione un oggetto clientMQTT che chiamiamo semplicemente 'client'
WiFiClient  wificlient;
PubSubClient client(wificlient);
char messaggio[30];

void setup() {
  delay(1000);
  pinMode(sensorePin, INPUT);
  dht.begin(); //inizializza l'oggetto dht
  // apre una connessione seriale. A scopo di debug inviamo messaggi al monitor seriale
  Serial.begin(115200);
  WiFi.begin(ssid, password);   //Si connette al router WiFi
  Serial.println("Connessione al WiFi in corso..");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("ESP8266 NodeMCU connessa alla rete WiFI: ");
  Serial.print(ssid);
  Serial.print("  con il numero IP : ");
  Serial.println(WiFi.localIP());    //l'IP di ESP8266 viene assegnato automaticamente dal DHCP
  Serial.println();
```

```
 /* Per creare una connessione ad un broker dobbiamo:
  *  1. usare il metodo setServer  e fornire indirizzo e porta del broker
  *  2. creare una funzione di callback (che chiamiamo MQTTcallback) e
  *  impostare  il metodo setCallback(MQTcallback) per ricevere messaggi
  */
  client.setServer(mqttServer, mqttPort);
  client.setCallback(MQTTcallback);
  connettiMQTT(); // si connette a MQTT
  client.subscribe(topic);
 // il client sottoscrive il topic "mdb/leggi_dht11" che riguarda i messaggi  pubblicati dall'app

} //----chiude setup()--------------------


void loop()
{
  client.loop();
  if( !client.connected()){
    connettiMQTT();         // si riconnette a MQTT e sottoscrive di nuovo il topic
    if(client.connected())  client.subscribe(topic);
  }
}

void MQTTcallback(char* topic, byte* payload, unsigned int length)
{
  /* la funzione di callback serve ad  elaborare i messaggi man mano ricevuti. Accetta  tre
   *  argomenti: topic (char), payload (il messaggio effettivo in byte),lunghezza del payload
  */
  Serial.print("Messagi in arrivo al topic: ");
  Serial.println(topic);

  Serial.print("Messaggio:");

  String message;
  for (int i = 0; i < length; i++) {
    message = message + (char)payload[i];   //Converte il messaggio da byte a String
  }
  Serial.println(message); //in  questo caso è previsto solo l'arrivo del messaggio "leggi"
  if(message == "leggi")
  {
    leggeHT();    // legge l'umidità H  e la temperatura T
    // pubblica i valori di H e T sul topic "mdb/valori_dht11"  sottoscritto dall'app android
    pubblicaHT();
  }
  Serial.println("-------------------------------------------------");

} //----chiude MQTTcallback()--------------------

void connettiMQTT(){

 while (!client.connected())
  {
    Serial.println("Connessione a  MQTT in corso..");
    //
    if (client.connect("ESP8266", mqttUser, mqttPassword ))
    {
        Serial.println("MQTT connesso");
        digitalWrite(Led, 1);
    }
    else
    {
      Serial.print("Errore connessione MQTT");
      Serial.println(client.state());
      delay(2000);
    }
  }
```

```
} // chiude connetti MQTT()--------------

void leggeHT(){
   // faccio 3 letture in modo che i valori letti dal sensore, che non è velocissimo, si
   // stabilizzino
    for(int i=0;i<3;i++)
    {
        H = dht.readHumidity();     // Lettura dell'umidità
        T = dht.readTemperature();  // Lettura della temperatura in gradi Celsius
        delay(200);
    }
}

void pubblicaHT(){
  if (isnan(H) || isnan(T))  //Verifica se  si presenta un errore di lettura
    {
          Serial.println("Errore di lettura...");
          H=0.0;
          T=0.0;
    }
    else
    {
          String s="#"+String(H)+"#"+String(T);
          char msg[30];
          s.toCharArray(msg, 30);
          Serial.print("valori inviati: ");
          Serial.println(msg);
          client.publish("mdb/valori_dht11",msg);
    }
}
```

Client MQTT: l'app Android che visualizza umidità e temperatura rilevate dalla scheda remota ESP8266 NodeMCU

Come funziona l'app

All'avvio l'app non è connessa al broker MQTT e visualizza i dati relativi all'ultima lettura effettuata (data e ora, Umidità H e temperatura T)

Per effettuare una lettura ci dobbiamo connettere. Clicchiamo sul bottone "Connetti MQTT" e dopo pochissimi attimi la label "MQTT Connesso" si colora di verde indicando in tal modo che il broker è connesso.

Il bottone "Leggi H e T" consente di visualizzare l'umidità e la temperatura rilevate in remoto:

1. l'app pubblica il messaggio "leggi" sul topic "mdb/leggi_dht11" e il broker lo trasmette alla scheda ESP8266 NodeMCU che è "abbonata" a quel topic
2. la scheda, tramite il sensore DHT11, esegue le misure e pubblica un messaggio sul topic "mdb/valori_dht11, con i valori letti di umidità e temperatura. Il broker lo trasmette all'app android che che è "abbonata" a quel topic e che visualizza i dati richiesti

Una volta visualizzati i due valori, l'app si disconnette da MQTT: considerando che umidità e temperatura variano molto lentamente, è preferibile connettersi, fare la lettura e poi disconnettersi piuttosto che restare sempre connessi.

Se dopo esserci connessi e aver cliccato su "Leggi H e T" la data non viene aggiornata, vuol dire che il dispositivo remoto non è in funzione.

Come utilizzare il client Android Paho MQTT

Tra le librerie che consentono di gestire il protocollo MQTT su Android , scegliamo di utilizzare la libreria ***Eclipse Paho*** , perché è la più diffusa ed è implementata in diversi linguaggi di programmazione (l'ho già utilizzata per realizzare il client MQTT javaScript nei due precedenti progetti IoT/MQTT)

Per configurare in Android Studio un client e un servizio MQTT, con le librerie necessarie, dobbiamo:

1. aggiungere il servizio Paho Android come dipendenza della nostra app (per il mio progetto ho scelto il nome: **mdb_mqtt_dht11)**. Occorre:

   a) inserire nel file **build.gradle(Module:app)** l'indirizzo del repository Paho:

```
repositories {
 maven {
   url "https://repo.eclipse.org/content/repositories/paho-releases/"
 }
}
```

   b) aggiungere nel file **build.gradle(Module:app)** le dipendenze del servizio Paho Android:

```
dependencies {
   altre righe già presenti…
   implementation 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.1.0'
   implementation 'org.eclipse.paho:org.eclipse.paho.android.service:1.1.1'
 }
```

2. configurare **AndroidManifest.xml** file per registrare il servizio MQTT Paho Android, ottenere le autorizzazioni per accedere a Internet, allo stato della rete e del telefono, e per lasciare che la nostra app rimanga attiva come servizio. Occorre:

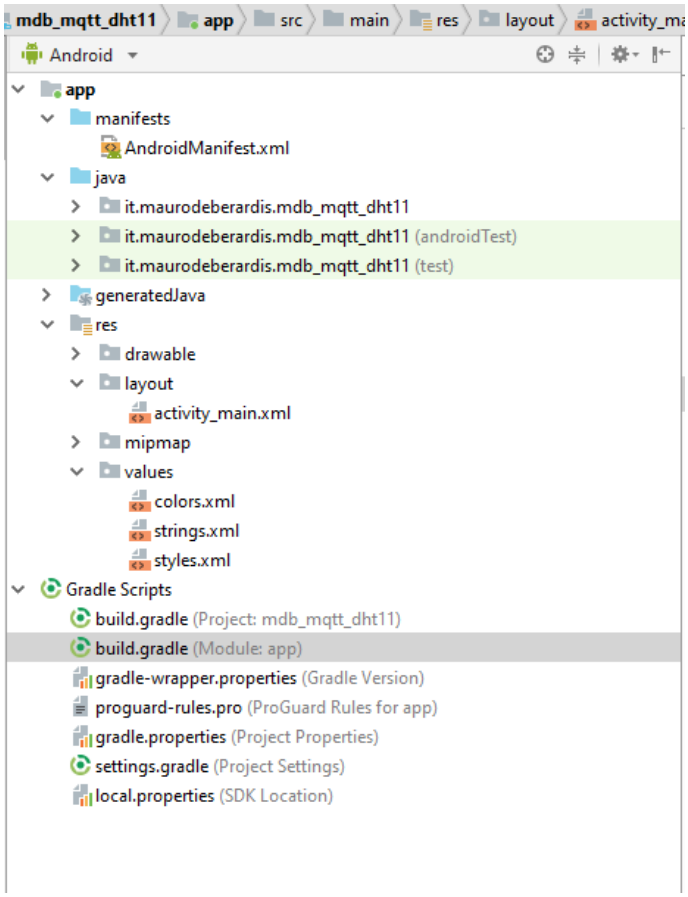
   a) aggiungere all'interno del tag <application …. /> la riga:

```
<service android:name="org.eclipse.paho.android.service.MqttService" />
```

   b) aggiungere prima del tag <application …. /> le righe:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK"/>
```

L' App Android in dettaglio



AndroidManifest.xml (in evidenza le righe da aggiungere)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="it.maurodeberardis.mdb_mqtt_dht11">
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name="org.eclipse.paho.android.service.MqttService" />
    </application>
</manifest>
```

build.gradle(Module:app) (in evidenza le righe da aggiungere)

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "it.maurodeberardis.mdb_mqtt_dht11"
        minSdkVersion 15
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

repositories {
    maven {
        url "https://repo.eclipse.org/content/repositories/paho-releases/"
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:2.0.4'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
    implementation 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.1.0'
    implementation 'org.eclipse.paho:org.eclipse.paho.android.service:1.1.1'
    }
```

res-values-strings.xml

```
<resources>
    <string name="app_name">Controllo remoto MQTT</string>
    <string name="titolo">Umidità e temperatura rilevate\n
                da una scheda ESP8266 NodeMCu\n
                 attraverso un sensore DHT11   "
    </string>
</resources>
```

activity_main.xml


```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="@string/titolo"
        android:textAlignment="center"
        android:textColor="#000080"
        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/tConnesso"
        android:layout_width="0dp"
        android:layout_height="30dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="12dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:background="#c0c0c0"
        android:gravity="center"
        android:text="MQTT non connesso"
        android:textColor="#ffffff"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintWidth_percent="0.4" />
    <TextView
        android:id="@+id/lDataOra"
        android:layout_width="wrap_content"
        android:layout_height="17dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:gravity="center"
        android:text="Data e ora ultima lettura"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.506"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tConnesso"
        app:layout_constraintWidth_percent="0.8" />
    <TextView
        android:id="@+id/tDataOra"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="2dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:gravity="center"
        android:text="data e ora"
```

```
    android:textStyle="italic"
    android:textColor="#000080"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/lDataOra"
    app:layout_constraintWidth_percent="0.9" />
<TextView
    android:id="@+id/labelH"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="32dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    android:text="Umidità H"
    android:textColor="#800000"
    android:textSize="18sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tDataOra" />
<TextView
    android:id="@+id/textH"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="4dp"
    android:text="0.0%"
    android:textAlignment="center"
    android:textColor="#800000"
    android:textSize="24sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/labelH"
    app:layout_constraintWidth_percent="0.4" />
<TextView
    android:id="@+id/labelT"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="24dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    android:text="Temperatura T"
    android:textColor="#800000"
    android:textSize="18sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.501"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textH" />
<TextView
    android:id="@+id/textT"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="4dp"
    android:text="0°"
    android:textAlignment="center"
    android:textColor="#800000"
    android:textSize="24sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/labelT"
    app:layout_constraintWidth_percent="0.4" />
<Button
    android:id="@+id/bConnetti"
```

```
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="2dp"
        android:layout_marginLeft="2dp"
        android:layout_marginBottom="2dp"
        android:background="#018577"
        android:textColor="#ffffff"
        android:text="Connetti MQTT"
        android:textAllCaps="false"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintWidth_percent="0.32" />
    <Button
        android:id="@+id/bDisconnetti"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="4dp"
        android:layout_marginLeft="4dp"
        android:layout_marginBottom="2dp"
        android:background="#018577"
        android:textColor="#ffffff"
        android:text="Disconnetti MQTT"
        android:textAllCaps="false"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@+id/bConnetti"
        app:layout_constraintWidth_percent="0.33" />
    <Button
        android:id="@+id/bLeggiHT"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="4dp"
        android:layout_marginLeft="4dp"
        android:layout_marginEnd="2dp"
        android:layout_marginRight="2dp"
        android:layout_marginBottom="2dp"
        android:background="#018577"
        android:textColor="#ffffff"
        android:text="Leggi H e T"
        android:textAllCaps="false"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/bDisconnetti"
        app:layout_constraintWidth_percent="0.32" />
</android.support.constraint.ConstraintLayout>
```

activity_main.java

```
package it.maurodeberardis.mdb_mqtt_dht11;

import android.content.Context;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.media.MediaPlayer;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import org.eclipse.paho.android.service.MqttAndroidClient;
import org.eclipse.paho.client.mqttv3.DisconnectedBufferOptions;
import org.eclipse.paho.client.mqttv3.IMqttActionListener;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.IMqttToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
```

```
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import java.text.SimpleDateFormat;
import java.util.Calendar;

public class MainActivity extends AppCompatActivity {
    MqttAndroidClient client;
    final String username = "emqx";
    final String password = "public";
    Context context;
    MediaPlayer mp;
    private Button leggiHT,connetti,disconnetti;
    private TextView umidità, temperatura,dataora,labelconnesso;
    String clientId;
    private SharedPreferences pref; //gestisco la dataora, h e t dell'ultima lettura
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        umidità=findViewById(R.id.textH);
        temperatura=findViewById(R.id.textT);
        dataora=findViewById(R.id.tDataOra);
        labelconnesso=findViewById(R.id.tConnesso);
        leggiHT=findViewById(R.id.bLeggiHT);
        connetti=findViewById(R.id.bConnetti);
        disconnetti=findViewById(R.id.bDisconnetti);
        clientId = MqttClient.generateClientId();
        final MediaPlayer mp = MediaPlayer.create(this, R.raw.click);        //suono di un click
        context=this.getApplicationContext();
       //se SharedPreferences non esiste creo il file di configurazione iniziale
        pref = getSharedPreferences("ultimalettura", MODE_PRIVATE);
        SharedPreferences.Editor editor;
        if (!pref.contains("dataora")) {
            editor = pref.edit();
            editor.putString("dataora", "data e ora");
            editor.putString("umidità", "0%");
            editor.putString("temperatura","0.0°");
            editor.commit();
        }
        // leggo e visualizzo  data e ora, umidità e temperatura dell'ultima lettura
        // salvata su SharedPreferences
        if (pref.contains("dataora"))
            dataora.setText(pref.getString("dataora", ""));
        if (pref.contains("umidità"))
            umidità.setText(pref.getString("umidità", ""));
        if (pref.contains("temperatura"))
            temperatura.setText(pref.getString("temperatura", ""));
         // Crea un'istanza di un client MQTT Android, che si collega al servizio Android Paho
        client = new MqttAndroidClient(context,"tcp://broker.emqx.io:1883", clientId);

        leggiHT.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(!client.isConnected())return;
               // se l'app non è connessa a MQTT, la lettura non si può fare…
                mp.start(); // suono di un click  che si può omettere…
                 // pubblica il messaggio "leggi" sul topic "mdb/leggi_dht11" che
                // la scheda ESP8266 NodeMCU ha sottoscritto
                String msg = "leggi";
                try {
                    client.publish("mdb/leggi_dht11", msg.getBytes(),0,false);
                } catch ( MqttException e) {
                    e.printStackTrace();
                }
            }
        }); //chiude leggiHT()----------------------
```

```
    connetti.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(!client.isConnected()){
                mp.start();// suono di un click  che si può omettere…
                ConnettiMqtt();
            }
        }
    });

    disconnetti.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(client.isConnected()){
                mp.start();// suono di un click  che si può omettere…
                DisconnettiMqtt();
            }
        }
    });


    client.setCallback(new MqttCallback() {
        @Override
        public void connectionLost(Throwable cause) {

        }

        // la funzione di callback  messageArrived  viene chiamata ogni volta che il
        // client MQTT riceve un messaggio.
        @Override
        public void messageArrived(String topic, MqttMessage message) throws Exception {
            String temp=new String(message.getPayload());
            // arriva ad esempio #47.00#20.70
            String[] v=temp.split("#");
            umidità.setText(v[1]+"%");
            temperatura.setText(v[2]+"°");
            // ottengo data e ora della lettura
            Calendar c = Calendar.getInstance();
            SimpleDateFormat df = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");
            dataora.setText(df.format(c.getTime()));
            // salvo in shared preferences data e ora, umidità e temperatura
            // dell'ultima lettura in modo tale che quando l'app viene aperta vengono
            // visualizzati gli ultimi dati rilevati
            SharedPreferences.Editor editor;
            if (pref.contains("dataora")) {
                editor = pref.edit();
                editor.putString("dataora", dataora.getText().toString());
                editor.putString("umidità", umidità.getText().toString());
                editor.putString("temperatura",temperatura.getText().toString());
                editor.commit();
            }
            DisconnettiMqtt(); // dopo aver visualizzato H e T si disconnette
        }

        @Override
        public void deliveryComplete(IMqttDeliveryToken token) {
        }
    });

}// chiude onCreate()-------------------
```

```
// Chiamando il  connect  del client cercherà in modo asincrono di connettersi al broker MQTT
// e restituire un token. Quel token può essere utilizzato per registrare callback, per ricevere
// una notifica quando la connessione MQTT viene connessa o si verifica un errore
    private void ConnettiMqtt(){
        try {
            IMqttToken token = client.connect(getMqttConnectionOption());
            token.setActionCallback(new IMqttActionListener() {
                @Override
                public void onSuccess(IMqttToken asyncActionToken) {
                    labelconnesso.setBackgroundColor(Color.parseColor("#008000"));
                    labelconnesso.setText("MQTT Connesso");
                    // sottoscrive il topic "mdb/valori_dht11" per poter  ricevere i valori H e T
                    // che saranno trasmessi dalla scheda ESP8266 NodeMCU
                    try{
                        client.subscribe("mdb/valori_dht11",0);
                    }catch (MqttException e){
                        e.printStackTrace();
                    }
                }

                @Override
                public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
                   String m="MQTT non si è connesso!!"
                   Toast.makeText(MainActivity.this,m,Toast.LENGTH_LONG).show();
                }
            });
        } catch (MqttException e) {
            e.printStackTrace();
        }
    }// chiude funzione ConnettiMqtt()----------------

    private MqttConnectOptions getMqttConnectionOption() {
        MqttConnectOptions mqttConnectOptions = new MqttConnectOptions();
        mqttConnectOptions.setUserName(username);
        mqttConnectOptions.setPassword(password.toCharArray());
        return mqttConnectOptions;
    }

    private void DisconnettiMqtt(){
        try {
            IMqttToken token = client.disconnect();
            token.setActionCallback(new IMqttActionListener() {
                @Override
                public void onSuccess(IMqttToken asyncActionToken) {
                    labelconnesso.setBackgroundColor(Color.parseColor("#c0c0c0"));
                    labelconnesso.setText("MQTT non connesso");
                }
                @Override
                public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
                     String m="MQTT non si disconnette!!"
                    Toast.makeText(MainActivity.this,m,Toast.LENGTH_LONG).show();
                }
            });
        } catch (MqttException e) {
            e.printStackTrace();
        }
    }//chiude funzione DisconnettiMqtt()----------------


}// chiude class MainActivity-----------------------------------------------------
```

Prova del progetto

1. Caricare lo sketch ***'esp8266_dht11_mqtt.ino'*** su ESP8266 NodeMCU
2. Aprire il Monitor Seriale a 115200 baud
3. Premere il pulsante di reset della scheda ESP8266
4. Eseguire l'app android installata sullo smartphone