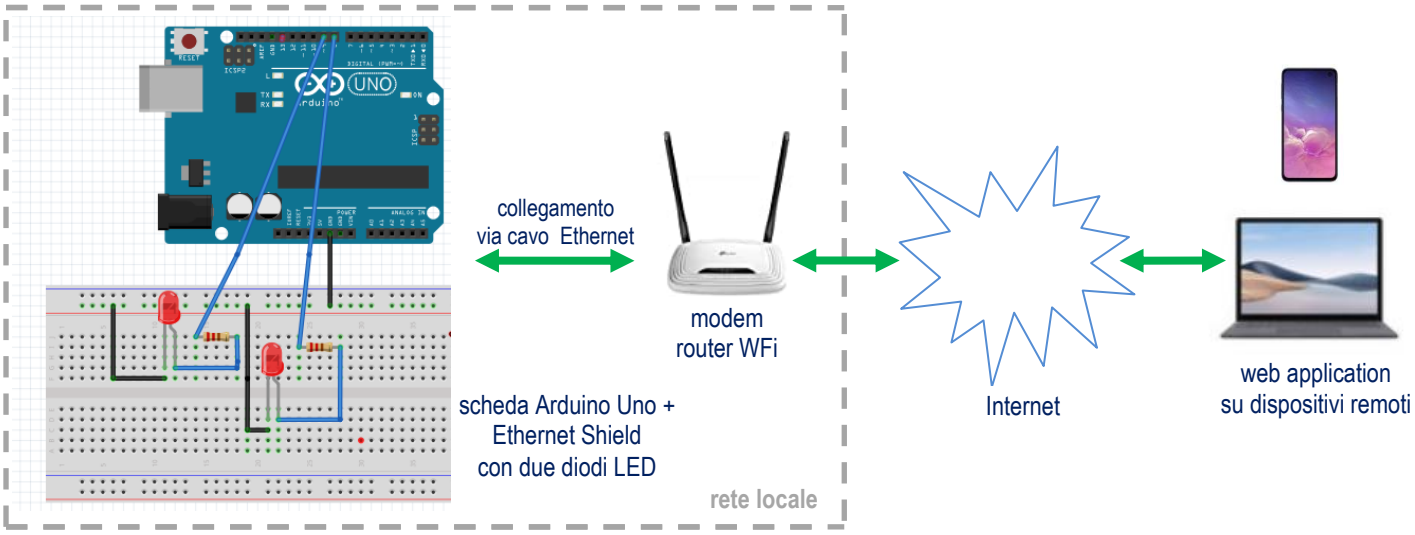
Controllare via Internet una scheda Arduino Uno + Ethernet Shield con una web app e MQTT

In questo progetto realizziamo il controllo remoto di una scheda Arduino Uno + Ethernet Shield che consente di accendere/spegnere due diodi LED: il controllo viene attuato, via Internet, attraverso una web application e utilizzando il protocollo **MQTT.**

Il progetto illustra come utilizzare un protocollo molto diffuso nel campo Internet of Things e può essere facilmente esteso ad applicazioni IoT più complesse, utili e diffuse quali il controllo di un motore o di una caldaia, di un impianto di irrigazione o delle luci di un giardino, di un sistema di allarme o di una telecamera di sorveglianza.



- La scheda Arduino Uno si connette ad Internet tramite Ethernet Shield
- la web application (su pc desktop, tablet, cellulare) 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 approcco 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 e gli altri dispositivi in rete (PC desktop, notebook, stampanti di rete, ecc.)
2. la scheda Arduino Uno + Ethernet Shield 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 Arduino.

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 Arduino+Ethernet Shield 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 esmpio: 192.168.1.123)

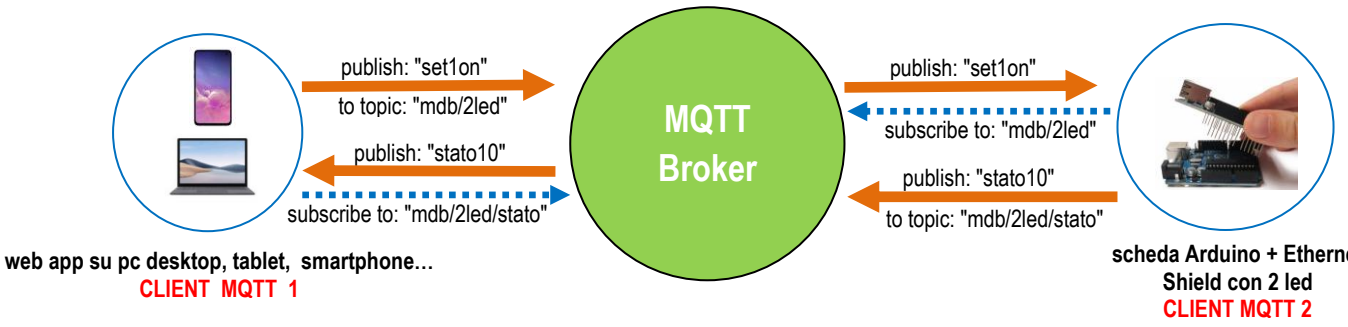
Oltre ai problemi sopraelencati, bisogna considerare che il protocollo HTTP è per sua natura poco adatto ad operare nel'IoT principalmente per il fatto che è pieno di regole e troppo pesante per adattarsi ai dispositivi IoT che consumano pochissima energia e spesso hanno una debole connettività di rete ('*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 che i destinatari sono **client MQTT** 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 scheda Arduino + Ethernet Shield con due diodi LED e la web app che controlla i due led via Internet

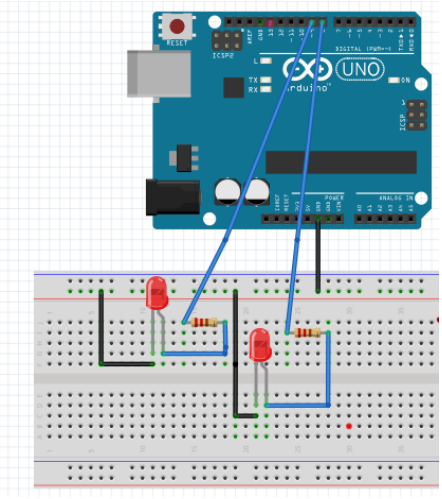
- la web app sottoscrive il topic "mdb/2led/stato" e riceve dal broker i messaggi pubblicati dallo sketch e relativi allo stato dei led della scheda. Ad esempio, riceve il messaggio "10" che significa: led1 accesso e led2 spento
- lo sketch della scheda sottoscrive il topic "mdb/2led" e riceve dal broker i messaggi pubblicati dalla web app e relativi all'azione da attuare (accensione/spegnimento dei led). Ad esempio, riceve il messaggio "set1on" che significa: accendi il led1

L'analisi del codice dei due clients MQTT charirà meglio il meccanismo di pubblicazione/sottoscrizione che consente alla web app di conoscere lo stato dei due led e di controllarne l'accensione e lo spegnimento.

Client MQTT: ESP8266 NodeMCU

Utilizziamo un Client C++ con Arduino IDE

Schema di collegamento di ESP8266 NodeMCU con due LED, LED1 e LED2 connessi rispettivamente a D8 e D9. Le resistenze di limitazione sono di 220Ω



Componenti utilizzati

- scheda Arduino + Ethernet Shield collegata ad un PC Windows X con cavo micro USB . Ethernet Shield connessa al router tramite cavo Ethernet
- Breadboard con cavetteria
- 2 diodi led
- 2 resistenze 220 Ohm

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 qui*) il file .zip della libreria PubSubClient e averla installata, creare un nuovo sketch

Lo sketch *'arduino2led_mqtt.ino'*

```
/* ---------------------------------------------------------
 * Prof. Mauro De Berardis 2021
 * ---------------------------------------------------------
 * In questo progetto realizziamo il controllo remoto di una scheda Arduino + Ethernet Shield
 * che consente di accendere/spegnere due  diodi  LED: il controllo viene attuato,
 * via Internet, attraverso una web application e utilizzando  il protocollo MQTT
 * Lo sketch realizza il client MQTT della scheda Arduino
 */

// La scheda Arduino comunica con lo shield tramite il bus SPI
// Le librerie SPI.h e Ethernet.h sono necessarie per il funzionamento dello Shield
#include <SPI.h>
#include <Ethernet.h>

/*La libreria PubSubClient consente al client di connettersi al broker MQTT e di
 *pubblicare messaggi e/o sottoscrivere topic */
 #include <PubSubClient.h>

/* Ethernet Shield è una scheda  basata sul chip intelligente W5100 che non ha un MAC Address
 * preassegnato e può essere configurato con un indirizzo di 6 cifre esadecimali qualsiasi purchè
 * diverso dai MAC Address di altri dispositivi della LAN. Scegliamo l'indirizzo
 * 80:80:80:80:80:80 */
byte macaddress[] = {0x80, 0x80, 0x80, 0x80, 0x80, 0x80 };

#define Led1 8
#define Led2 9
int sled1=0; // stato led1
int sled2=0; // stato led2
```

```
/* 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  *topic =  "mdb/2led";
const char  *mqttUser = "emqx";
const char  *mqttPassword = "public";
const int    mqttPort = 1883;

// Oggetti relativi a Ethernet e MQTT
EthernetClient  ethclient;          //Creazione di un oggetto client Ethernet
PubSubClient mqttclient(ethclient); //Creazione di un oggetto client MQTT


void setup() {
  delay(1000);
  pinMode(Led1,OUTPUT);
  pinMode(Led2,OUTPUT);

  // apre una connessione seriale. A scopo di debug inviamo messaggi al monitor seriale
  Serial.begin(115200);

  Ethernet.begin(macaddress); //Avvia la connessione Ethernet
  Serial.println("Connessione Ethernet stabilita ");
  Serial.print("Indirizzo IP: ");
  Serial.println(Ethernet.localIP());
  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
 */
  mqttclient.setServer(mqttServer, mqttPort);
  mqttclient.setCallback(MQTTcallback);
  delay(1000);
  while (!mqttclient.connected())
  {

    Serial.println("Connessione a  MQTT in corso..");
    if (mqttclient.connect("Arduino",mqttUser,mqttPassword))
    {
       Serial.println("MQTT connesso");
    }
    else
    {
      Serial.print("Errore connessione MQTT");
      Serial.println(mqttclient.state());
      delay(2000);
    }
  }
  mqttclient.subscribe(topic);
  //il client sottoscrive il topic "mdb/2led" che riguarda i messaggi che saranno pubblicati
  / dalla web app
  mqttclient.publish("mdb/2led/stato","00");
   //il client pubblica sul topic "mdb/2led/stato" lo stato iniziale

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

```
void loop()
{
  mqttclient.loop();
  /* La funzione loop() è una funzione  che legge i buffer di ricezione e invio
   * ed elabora  i messaggi che trova. Dal lato della ricezione, attiva la funzione di
   * callback
   */
}


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("Messaggi 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);
  // a seconda del messaggio ricevuto imposta l'accensione/spegnimento dei 2 led
  if(message == "set1off")   sled1=0;
  if(message == "set1on")    sled1=1;
  if(message == "set2off")   sled2=0;
  if(message == "set2on")    sled2=1;
  digitalWrite(Led1, sled1);
  digitalWrite(Led2, sled2);

  // pubblica un messaggio sul topic "mdb/2led/stato" con il quale informa la web app sullo stato
  // aggiornato dei due led: 00 acceso-acceso 01 acceso-spento 11 acceso-acceso 00 spento-spento
  if(sled1==0 && sled2==0)  mqttclient.publish("mdb/2led/stato","00");
  if(sled1==0 && sled2==1)  mqttclient.publish("mdb/2led/stato","01");
  if(sled1==1 && sled2==0)  mqttclient.publish("mdb/2led/stato","10");
  if(sled1==1 && sled2==1)  mqttclient.publish("mdb/2led/stato","11");

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

Client MQTT: la web app che controlla via Internet la scheda ESP8266

Utilizziamo un Client JavaScript MQTT su websockets: 'controllo2ledarduino_mqtt.html'

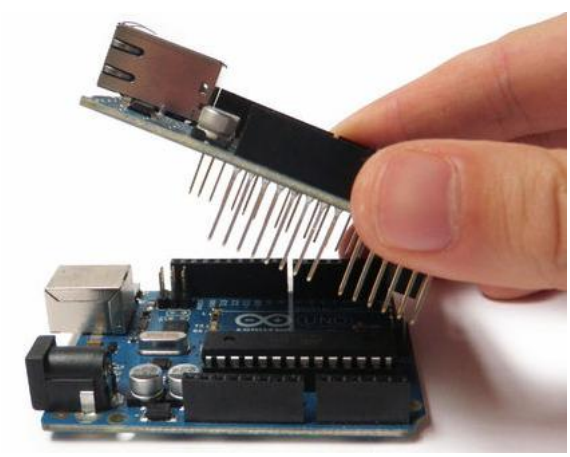
```
<!DOCTYPE html>
<!-- ****** controllo2led-mqtt.html - Mauro De Berardis 2021 ******
In questo progetto  viene utilizzata la libreria MQTT "paho-mqtt" basata su
javascript a  cui si accede tramite il link a cdnjs.cloudflare.com.
Per testare il progetto e in particolare per visualizzare i messaggi che arrivano
alla web app ed eventuali messaggi di errore, viene utilizzata la console JavaScript.
Per attivarla, eseguire questa pagina html in un browser, premere il tasto F12 e
selezionare l'opzione 'Console'-->
<html>
<head>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.min.js"
type="text/javascript">
 </script>
</head>
<style>
<!--Inseriamo un pò di codice css -->
body{font-family: Arial,Helvetica,sans-serif;font-size:14px; }
label, button {
          width:190px;
          display: inline-block;
          text-align: center;
          font-size:140%;
          border-radius: 5px;
          margin-bottom:10px;
}
label {color:#800000;font-weight:bold}
button{height:40px;background-color:#f0f0f0;border:1px solid #c0c0c0}
.stato_on{
          width:40px;
          height:40px;
          display: inline-block;
          border-radius: 50%;
          background-color: #ff0000;
}
.stato_off{
          width:40px;
          height:40px;
          display: inline-block;
          border-radius: 50%;
          background-color: #d0d0d0;
}
p {color:#000080;font-size:150%;text-align:center}
.box {
          width: 400px;
          margin: 0px auto;
          text-align: left;
          padding-left: 10px;
          background-color: #ffffff;
          color: #333;
          border: 1px solid #444444
}
</style>
<body>
<div class="box">
   <p>Arduino Uno + Ethernet Shield <br/> Client JavaScript MQTT <br/>
   <b>Controllo remoto di 2 led </b></p>
   <label>Stato Led1</label> 
   <label>Stato Led2</label><br/>
   <div class="stato_off" id="l1" style="margin-left:75px"> </div>
   <div class="stato_off" id="l2" style="margin-left:150px"> </div>
   <br/><br/><br/>
   <label>Imposta Led1</label> 
   <label>Imposta Led2</label><br/>
```

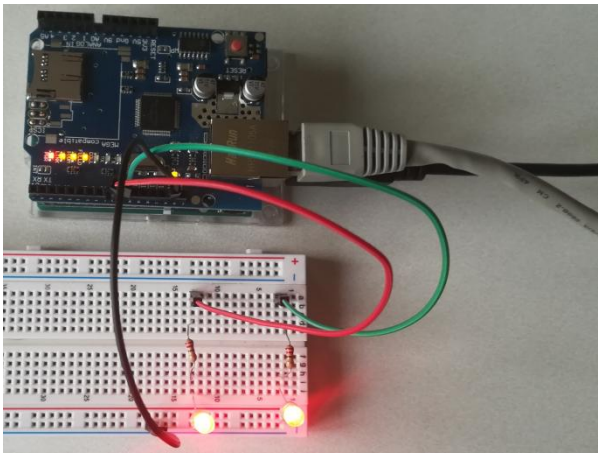
```
<button onclick="azione(1)">Accendi</button> 
<button onclick="azione(3)">Accendi</button><br/>
<button onclick="azione(2)">Spegni</button> 
<button onclick="azione(4)">Spegni</button>
<br/><br/><br/>Mauro De Berardis 2021

<script type="text/javascript">
      // Crea un oggetto clientMQTT: new Client(host, port, clientId)
      idc=parseInt(Math.random() * 100);//clientID random tra 0  e 99
      clientmqtt = new Paho.MQTT.Client("broker.emqx.io", 8084,"web_" + idc);
      // set callback handlers
      clientmqtt.onConnectionLost = onConnectionLost;
      clientmqtt.onMessageArrived = onMessageArrived;
      // 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

 var options =
      {
               useSSL: true,
               userName: "emqx",
               password: "public",
               onSuccess:onConnect,
               onFailure:onFail
      }

      // connesione del client MQTT
      clientmqtt.connect(options);
      // funzione chiamata quando il client si connette
      function onConnect() {
               console.log("onConnect");
               clientmqtt.subscribe("mdb/2led/stato");
               // sottoscrive il topic "mdb/2led/stato"  attraverso il quale
               // la web app riceve lo stato dei  due led della scheda
               message = new Paho.MQTT.Message("Ciao scheda Arduino + Ethernet Shield!");
               message.destinationName = "mdb/2led";
               // manda il messaggio di benvenuto al topic "mdb/2led"
               clientmqtt.send(message);
      }

      function azione(quale) {
               if(quale == 1) { message = new Paho.MQTT.Message("set1on"); }
               if(quale == 2) { message = new Paho.MQTT.Message("set1off"); }
               if(quale == 3) { message = new Paho.MQTT.Message("set2on"); }
               if(quale == 4) { message = new Paho.MQTT.Message("set2off"); }
               message.destinationName = "mdb/2led";
               clientmqtt.send(message);
               // pubblica il messaggio (azione) che sarà letto ed elaborato dal
               // clientESP8266 che ha sottoscritto il  topic "mdb/2led"
      }

function onFail(e){
  console.log(e);
}

// funzione chiamata quando il client perde la connessione
function onConnectionLost(responseObject) {
  if (responseObject.errorCode !== 0) {
    console.log("onConnectionLost:"+responseObject.errorMessage);
  }
}
```

```
  // funzione chiamata quando arriva un messaggio relativo al
  // topic "mdb/2led/stato" che questo client ha sottoscritto
  function onMessageArrived(message) {
    console.log("onMessageArrived:"+message.payloadString);
          stato=message.payloadString;
          // visualizza lo stato dei due led remoti. Il rosso indica lo stato on - led acceso
          if(stato=="00") {

                    document.getElementById('l1').innerHTML = "<span class='stato_off'></span>";
                    document.getElementById('l2').innerHTML = "<span class='stato_off'></span>";
          }
          if(stato=="01") {
                    document.getElementById('l1').innerHTML = "<span class='stato_off'></span>";
                    document.getElementById('l2').innerHTML = "<span class='stato_on'></span>";
          }
          if(stato=="11") {
                    document.getElementById('l1').innerHTML = "<span class='stato_on'></span>";
                    document.getElementById('l2').innerHTML = "<span class='stato_on'></span>";
          }
          if(stato=="10") {
                    document.getElementById('l1').innerHTML = "<span class='stato_on'></span>";
                    document.getElementById('l2').innerHTML = "<span class='stato_off'></span>";
    }
  }
  </script>
</div>
</body>
</html>
```

Prova del progetto

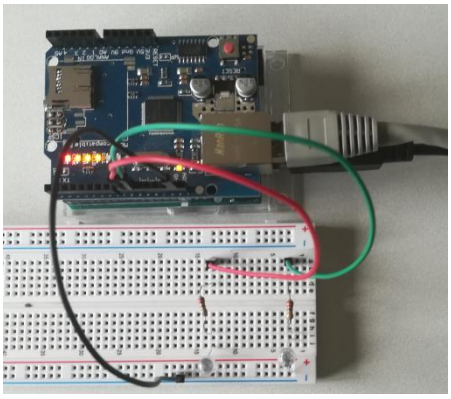
1. Dopo aver inserito Ethernet Shield in Arduino, realizzare, utilizzando i piedini D8, D9 e Gnd di Arduino "replicati" su Ethernet Shield, i collegamenti previsti per realizzare il circuito. Collegare quindi Arduino al PC (cavo USB nero) ed Ethernet Shield allo switch di rete (cavo di rete bianco)

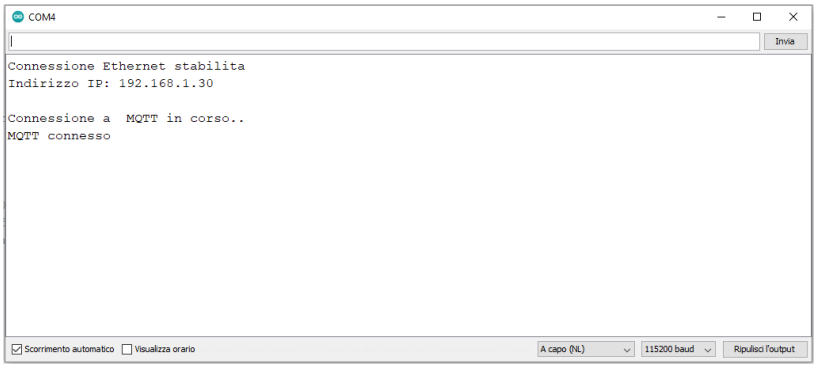




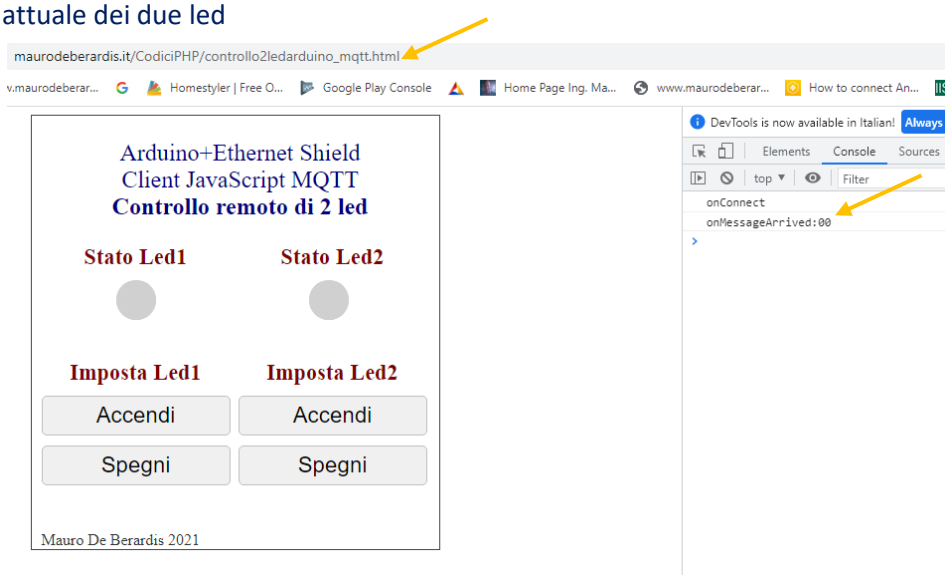
2. Caricare sulla scheda lo sketch **'arduino2led_mqtt.ino'**
3. Aprire il Monitor Seriale a 115200 baud
4. Premere il pulsante di reset della scheda
5. Eseguire in un browser la pagina web **'controllo2ledarduino_mqtt.html'**, memorizzata su un Pc locale collegato ad Internet, ad esempio sul desktop, o su un server web. Per la prova ho caricato la pagina sul mio sito (indirizzo www.maurodeberardis.it/CodiciPHP/controllo2ledarduino_mqtt.html) e ho utilizzato sia un pc desktop che un cellulare
6. Con il pc desktop, aprire la console JavaScript del browser premendo il tasto F12 e selezionando l'opzione 'Console'

Eseguo la pagina Web www.maurodeberardis.it/CodiciPHP/controllo2ledarduino_mqtt.html e apro la Console. Resetto la scheda: il monitor seriale indica la connessione al router con l'IP 192.168.1.30 (IP assegnato via DHCP) e successivamente la connessione al Broker MQTT

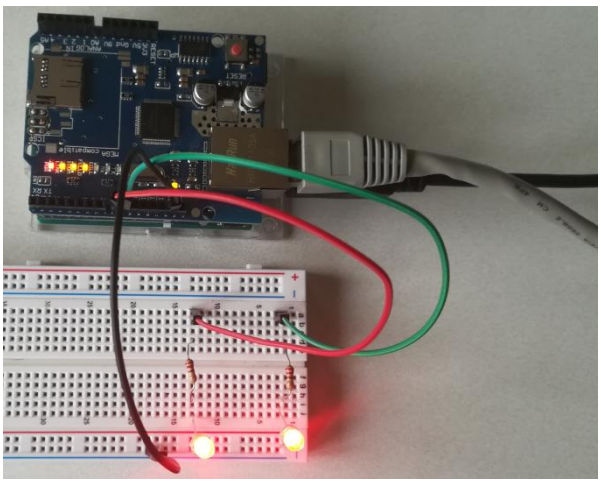


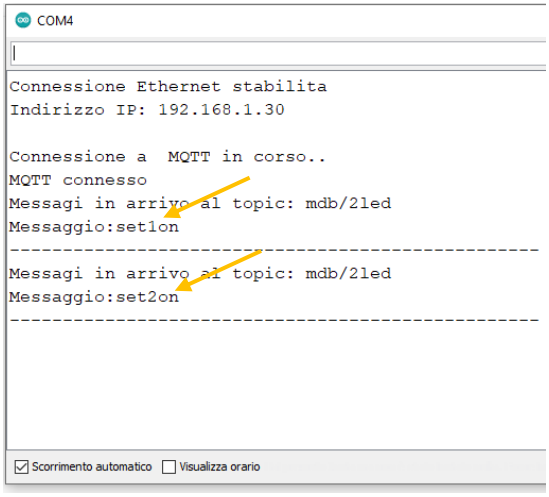


Avendo resettato la scheda, i due led sono spenti e risultano spenti anche sulla web app che ha ricevuto il messaggio MQTT '00'. Se eseguiamo la web app quando la scheda è già in funzione, essa rivelerà lo stato attuale dei due led

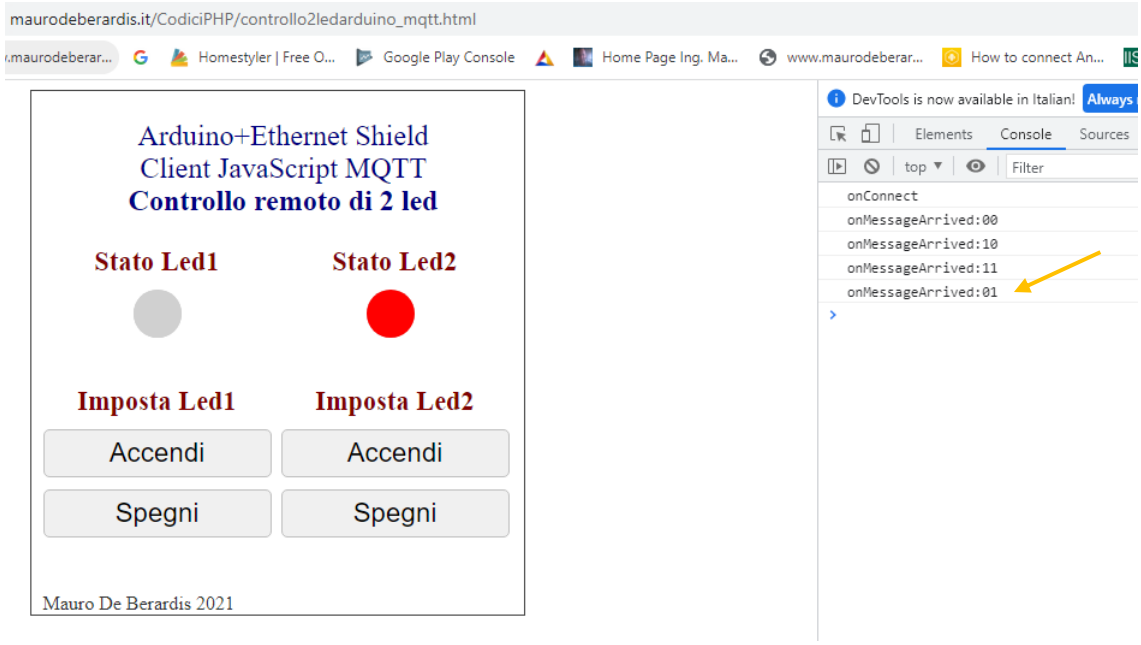


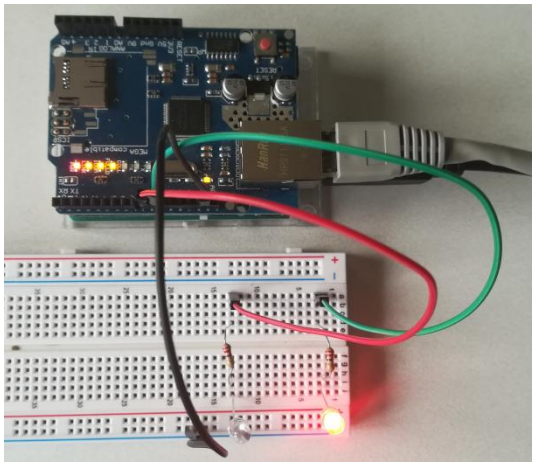
Accendo il led1, la web app pubblica il messaggio 'set1on' il quale viene ricevuto dallo sketch Arduino che accende il led1 e pubblica il messaggio '10' con il quale informa la web app che il led1 è accesso e il led2 è spento. Di seguito accendo il led2, la web app pubblica il messaggio 'set2on' il quale viene ricevuto dallo sketch Arduino che accende il led2 e pubblica il messaggio '11' con il quale informa la web app che entrambi i led sono accesi

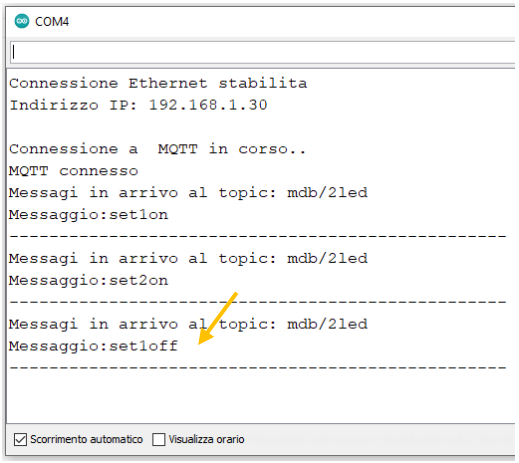




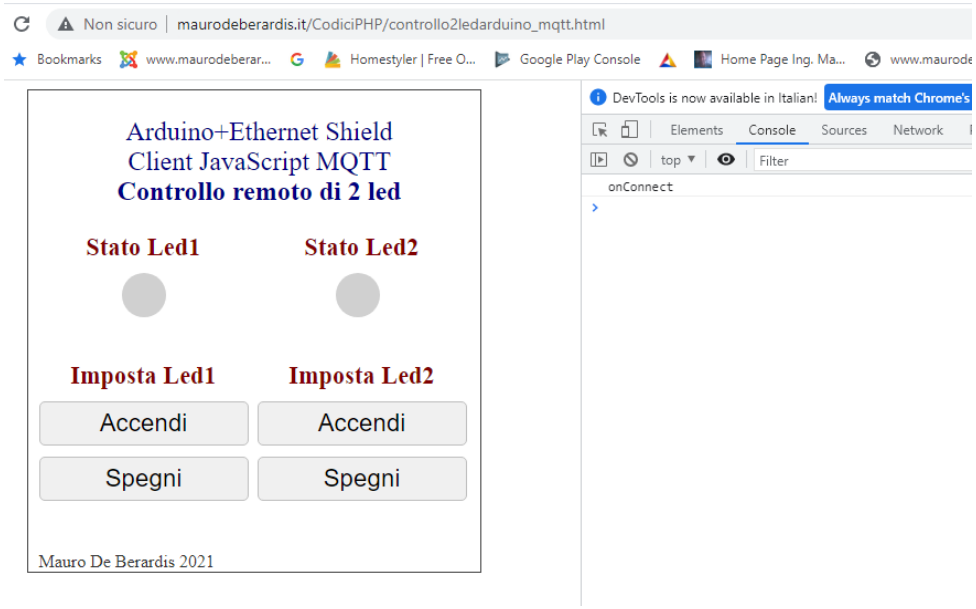
Spengo il led1. La web app pubblica il messaggio 'set1off' il quale viene ricevuto dallo sketch Arduino che spegne il led1 e pubblica il messaggio '01' con il quale informa la web app che il led1 è spento e il led2 è acceso







Se la scheda non è in funzione, la web app rivela i due led spenti e i click sui bottoni di impostazione non hanno alcun effetto. La Console resta vuota perché la scheda ESP8266 non è connessa a MQTT e non invia messaggi né può riceverli



Utilizzando un cellulare