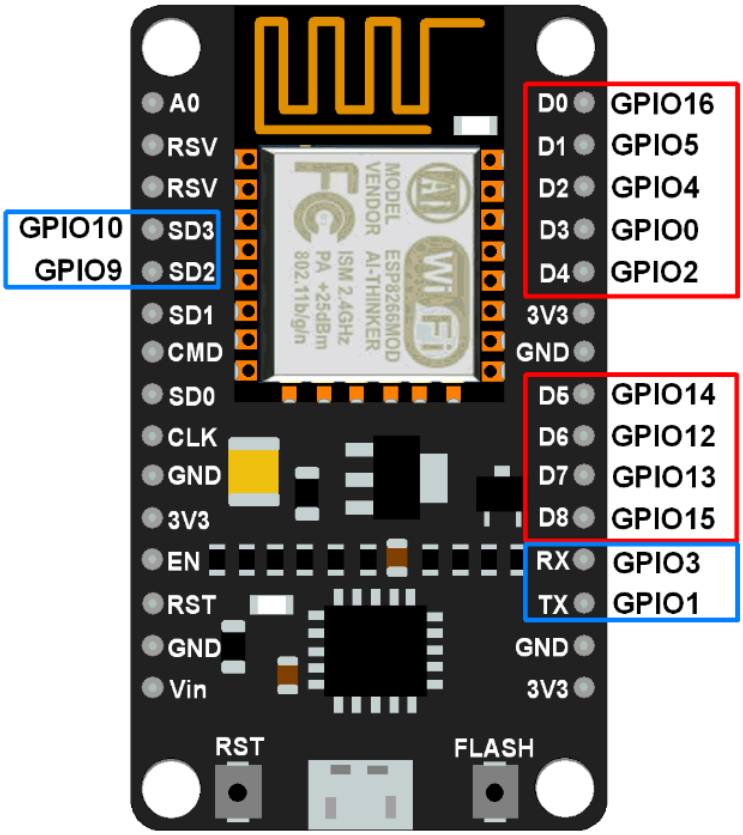
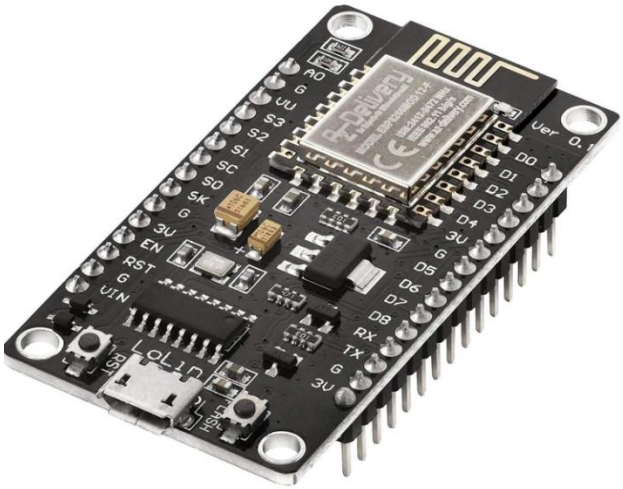
Web Server ESP8266 NodeMCU

Esempio applicativo di Internet of Things (IoT).
Utilizzando la scheda ESP8266 NodeMCU e l'IDE di Arduino, realizziamo un Server Web che monitora l'umidità e la temperatura rilevate da un sensore DHT11. Al Server Web possono accedere tutti i dispositivi della rete wifi, pc desktop, tablet e smartphone, dotati di un browser

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.



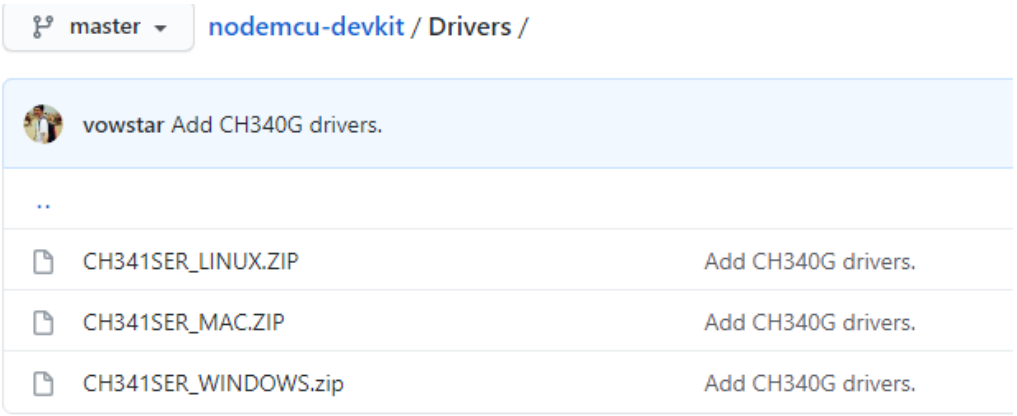


L'**ESP8266 NodeMCU** ha un totale di 30 pin che lo interfacciano con il mondo esterno.

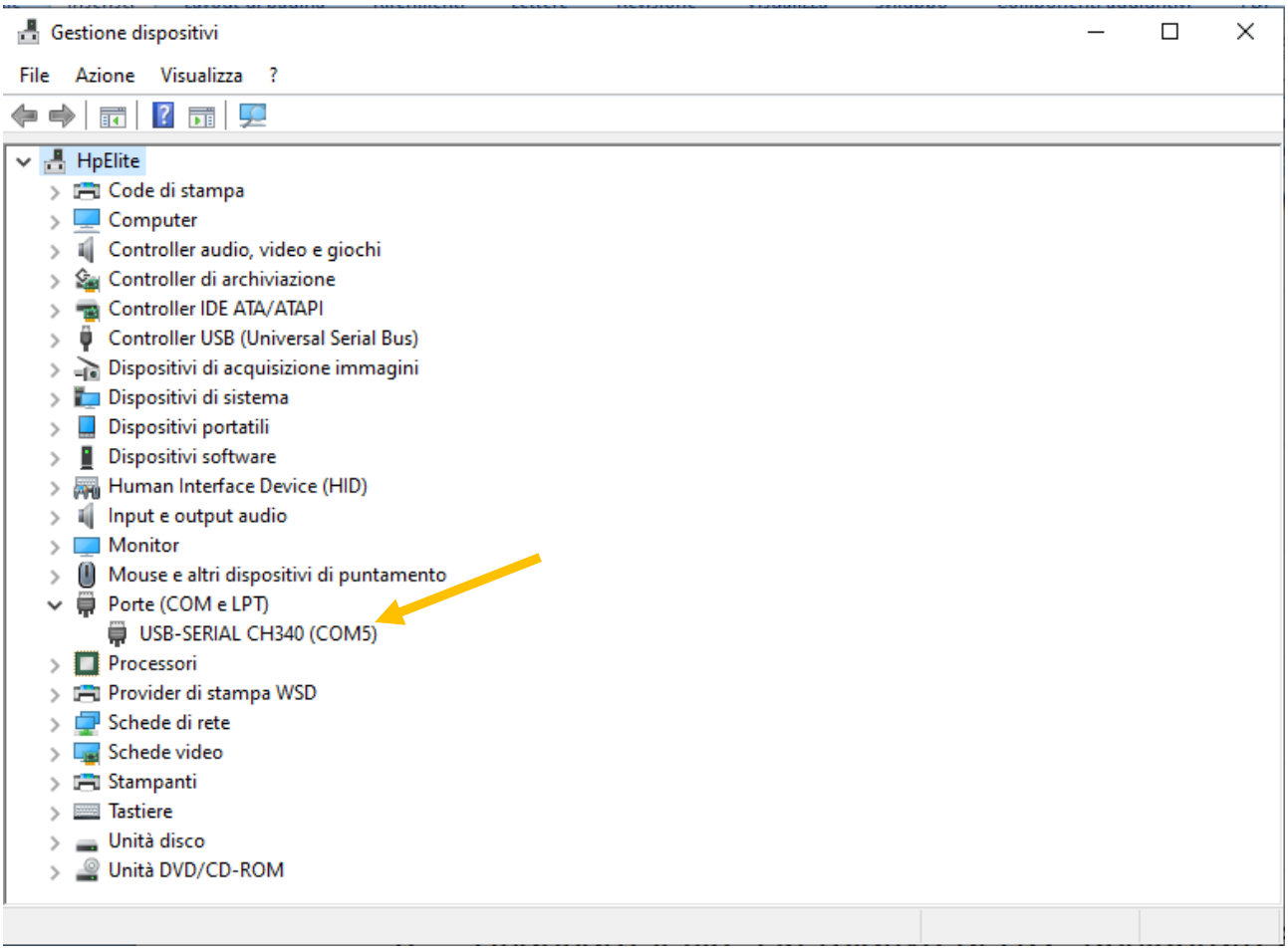
Utilizzo dell'IDE di Arduino per programmare ESP8266 NodeMCU

ESP8266 NodeMCU è un dispositivo IoT open source e per impostazione predefinita è equipaggiato con un firmware scritto con il linguaggio Lua. Spesso però, come nel nostro caso, **viene utilizzato con l'IDE di Arduino**: per questo motivo vediamo di seguito, passo passo, che cosa fare per poter programmare ESP8266 NodeMCU usando il linguaggio C ++ di Arduino.

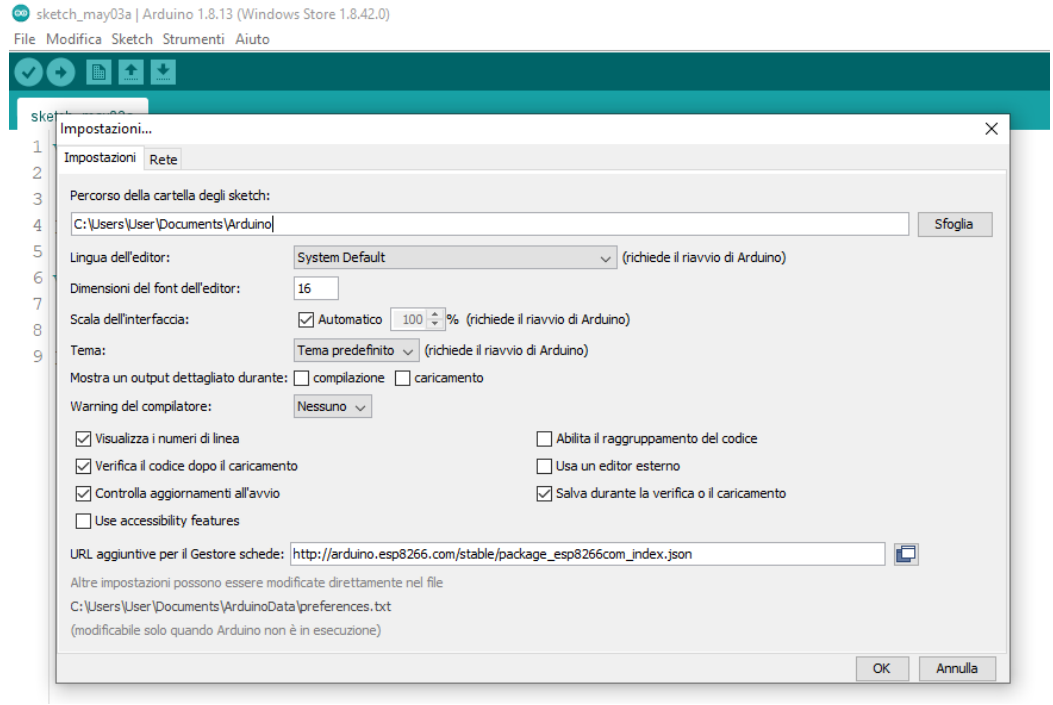
1. Collegare ESP8266 NodeMCU al PC con un cavo micro USB.
2. Aprire il link ***https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers***



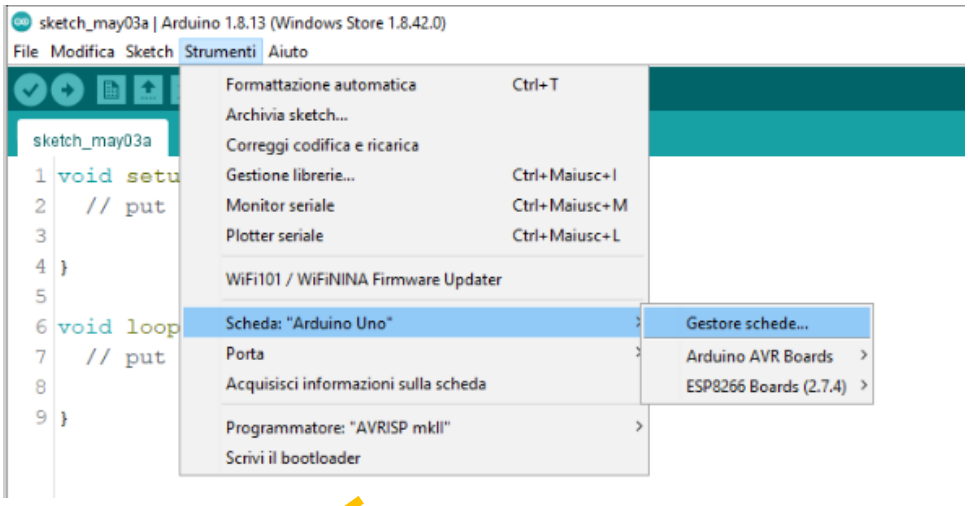
3. Scaricare il file .zip relativo al SO desiderato e scompattarlo. Eseguire quindi il file.exe che installa i driver USB per il chip USB-seriale. Se il SO è Windows, scaricare il file CH341SER_WINDOWS.zip e scompattarlo. Eseguire quindi il file CH341SER.exe e verificare l'installazione del driver, aprendo l'utilitity "Gestione dispositivi"

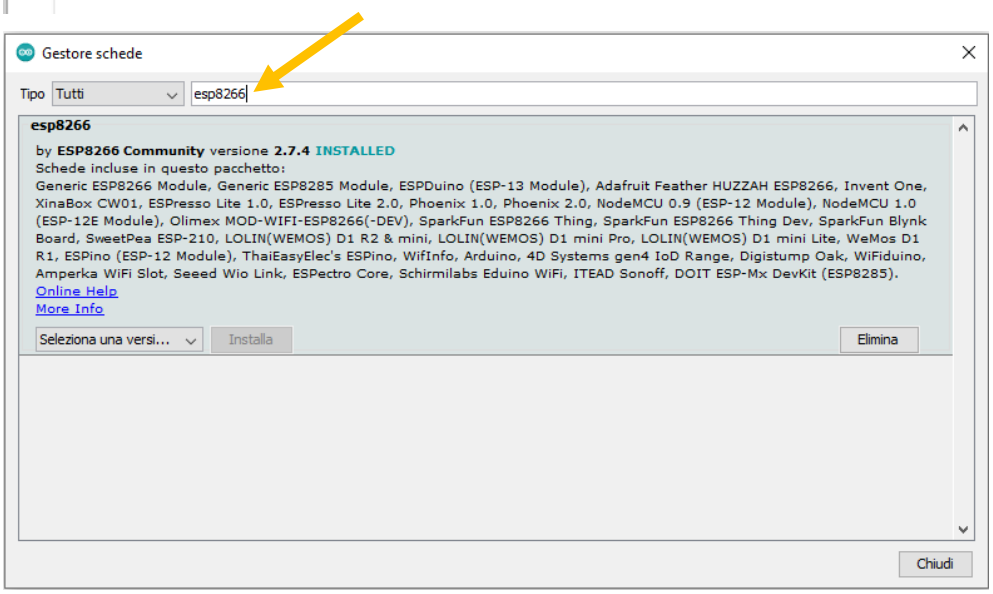


4. Aprire l'IDE Arduino, scegliere l'opzione "Impostazioni" sul menu "File" , copiare il link ***http://arduino.esp8266.com/stable/package_esp8266com_index.json*** nel campo "URL aggiuntive per il Gestore schede" e confermare cliccando su "OK"

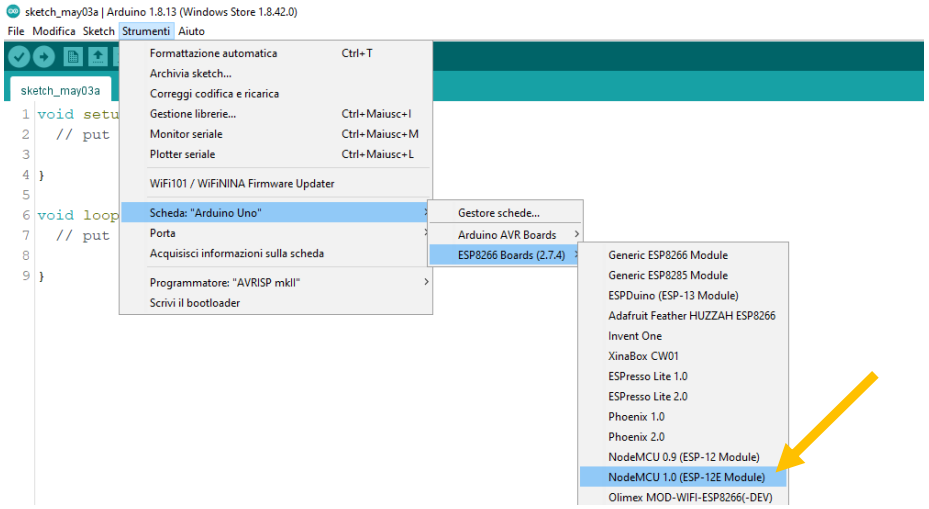


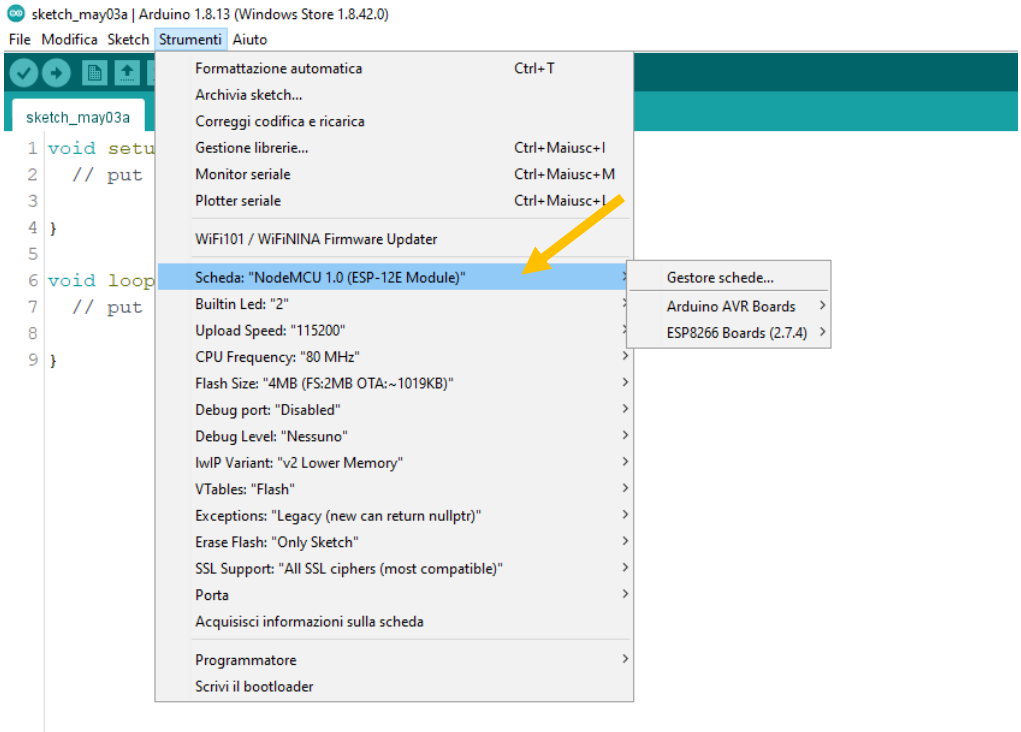
5. Sull'IDE Arduino, scegliere l'opzione "Gestore schede" del menu "Strumenti" e inserire nel campo di ricerca la parola chiave ESP8266. Installare quindi la libreria proposta.

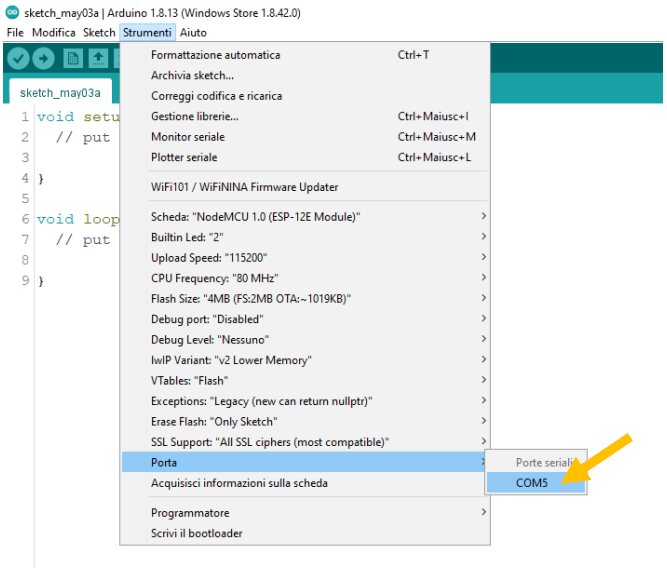




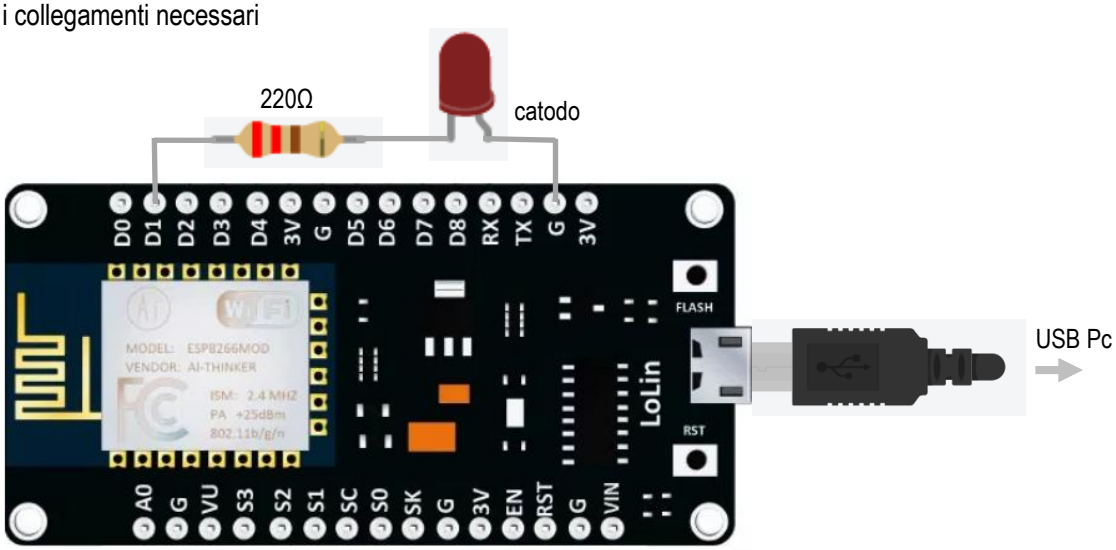
6. A questo punto, tornando all'opzione che permette di scegliere la scheda sul menu "Strumenti", vediamo che oltre alla libreria di Arduino è presente anche quella di ESP8266: la selezioniamo e scegliamo la **scheda NodeMCU (ESP -12E Module)**. Infine, sempre sul sul menu "Strumenti", scegliamo la porta assegnata quando sono stati installati i driver USB per il chip USB-seriale.

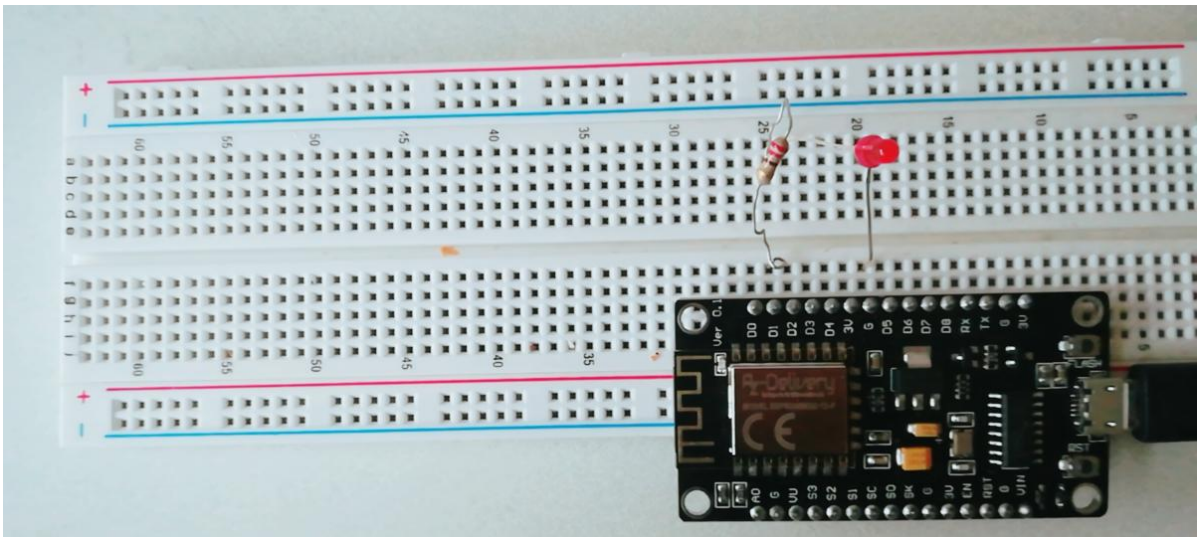






7. Per verificare la correttezza della configurazione dell'IDE di Arduino, proviamo ad implementare un semplice sketch che fa lampeggiare un led collegato sul pin D1
   a. Effettuare i collegamenti necessari





   b. Scrivere questo semplice sketch:

```
#define ledPin D1
void setup() {
  pinMode (ledPin, OUTPUT);
}
void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

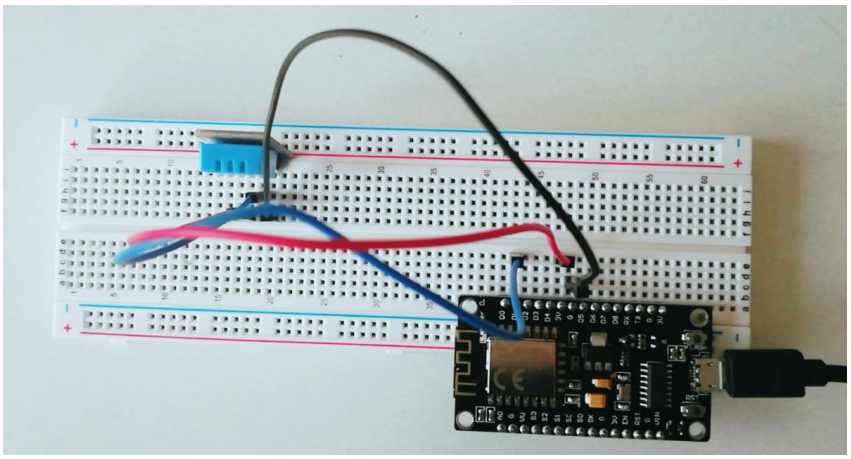
   c. Collegare l'ESP8266 NodeMCU al computer utilizzando il cavo micro USB. Selezionare la scheda NodeMCU (ESP -12E Module) e la porta assegnata
   d. Caricare il programma: il diodo led collegato al pin D1 inizia a lampeggiare

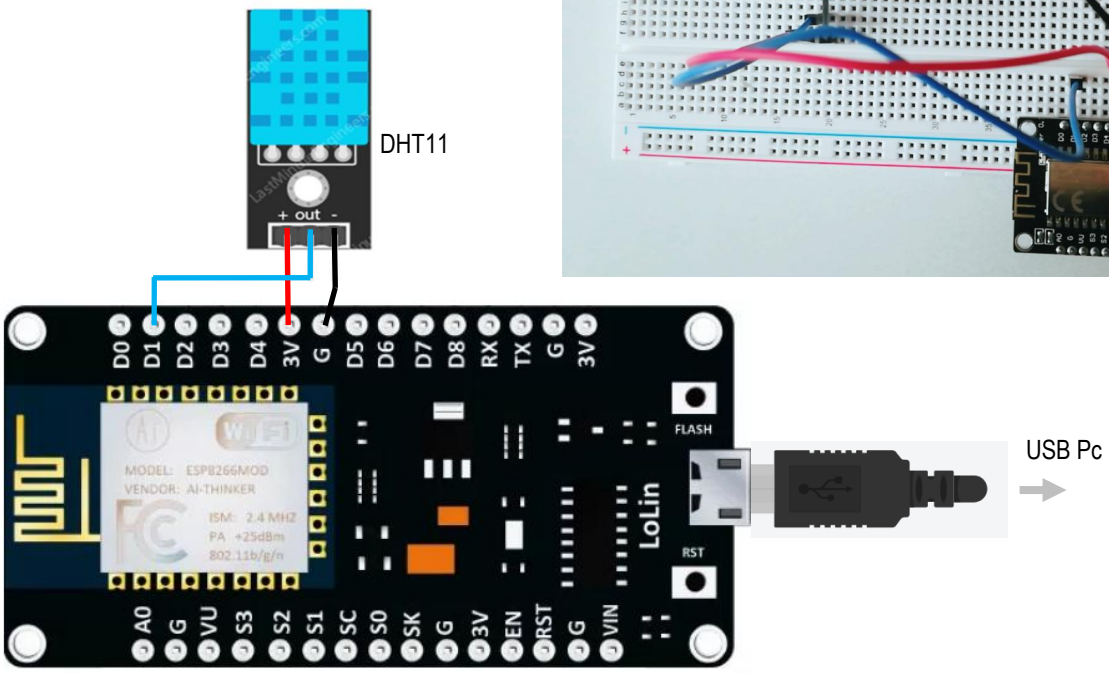
Collegamenti ESP8266 NodeMCU - Sensore DHT11

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

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

Tabella dei collegamenti DHT11-ESP8266





Sketch sull'ESP8266 NodeMCU programmato con l'IDE di Arduino

Vogliamo realizzare un semplice Server Web HTTP che visualizza i valori di umidità e temperatura rilevati dal sensore DHT11 e stampa la data e l'ora della misura. Il sensore effettua la misura ogni volta che un Client (ovvero un dispositivo della rete WiFi) accede alla pagina web o esegue un refresh.

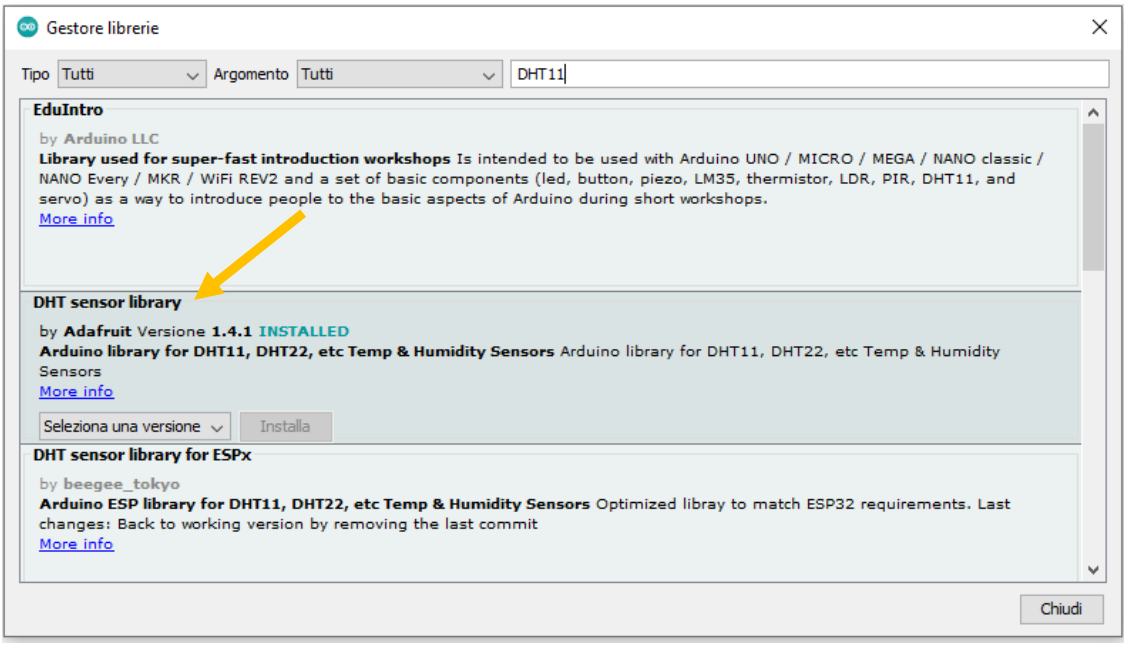
A tal fine programmiamo la scheda ESP8266 utilizzando l'IDE di Arduino (ricordo che è necessario che il componente aggiuntivo ESP8266 sia installato nell'IDE di Arduino e che sia selezionata la porta giusta).

Prima di scrivere il codice, bisogna installare, e quindi includere nello sketch, due librerie:

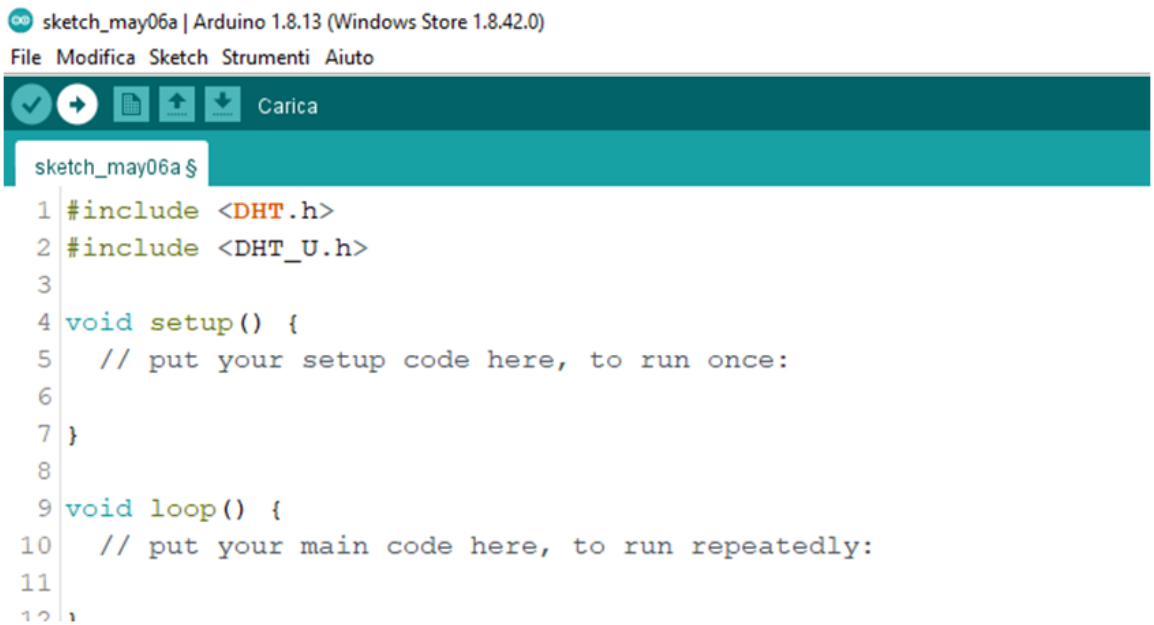
- la libreria necessaria per poter utilizzare il sensore di umidità e temperatura DHT11
- la libreria necessaria per ottenere data e ora da un Server NTP

Libreria necessaria per poter 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)

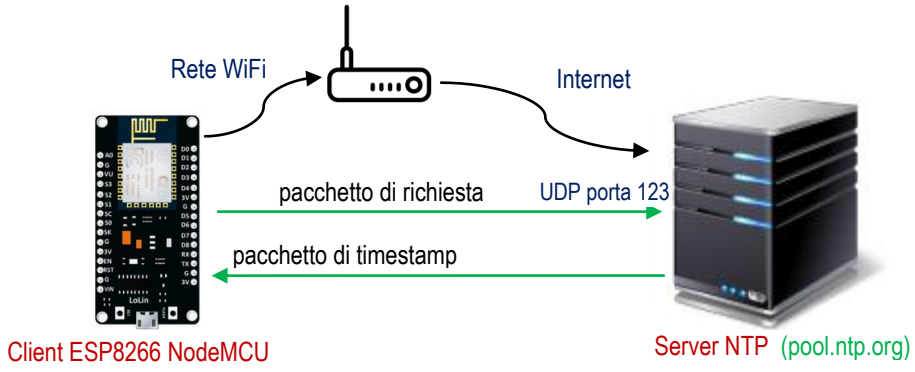


Libreria necessaria per ottenere data e ora da un Server NTP

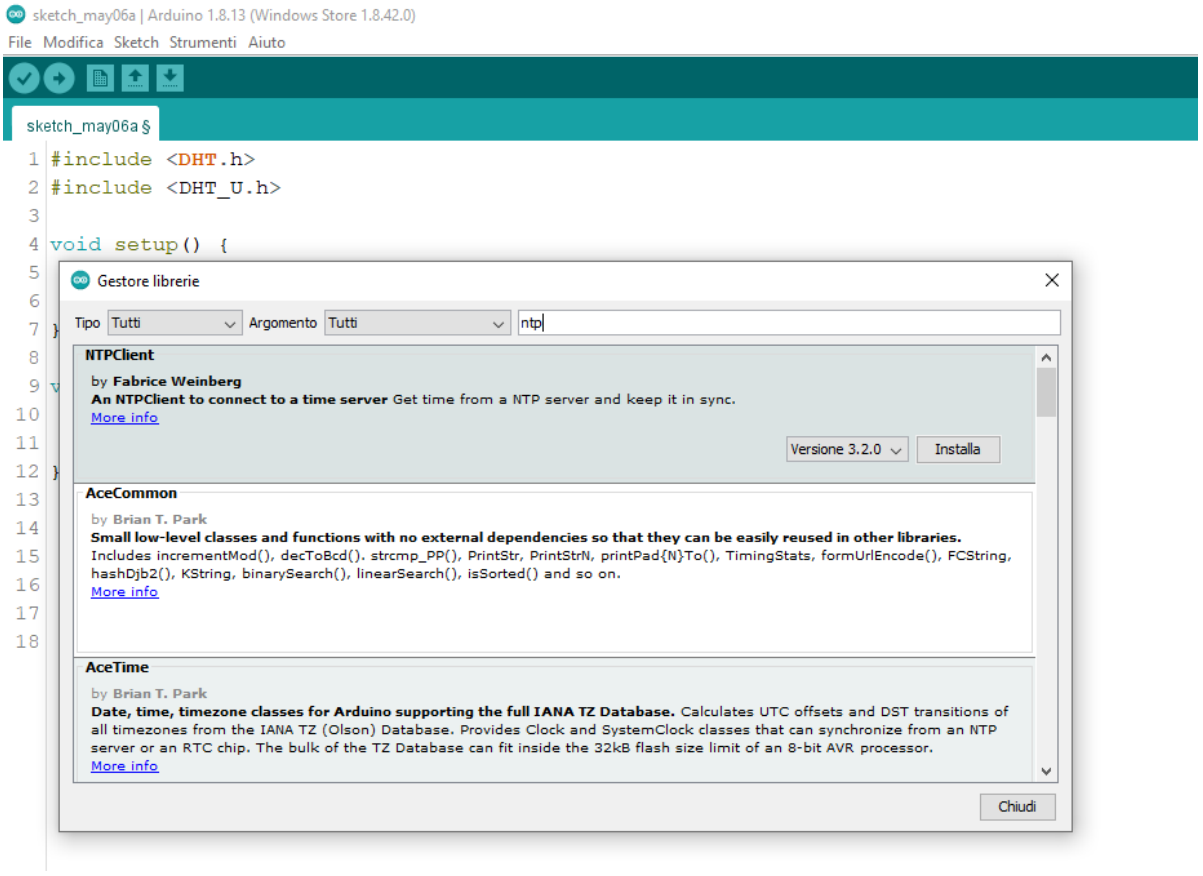
ESP8266 NodeMCU non è un dispositivo RTC (Real Time Clock), ovvero non è un dispostivo con funzione di orologio. Tuttavia se opera all'interno di una rete WiFi con accesso a Internet, può recuperare data e ora da un Server NTP.
NTP (Network Time Protocol) è un protocollo Internet standard per sincronizzare i dispositivi in rete con l' ora UTC (Coordinated Universal Time) entro pochi millisecondi. L'UTC è lo standard temporale utilizzato in tutto il mondo e generato da precisissimi orologi atomici.

NTP opera generalmente in modalità Client-Server. Nel nostro caso:

- l'ESP8266 NodeMCU (il Client) si connette al Server NTP utilizzando il protocollo UDP (User Datagram Protocol) sulla porta 123 e invia un pacchetto di richiesta
- Il Server NTP, in risposta, invia un pacchetto di timestamp



Con l'IDE Arduino in esecuzione, scegliere Strumenti-Gestione librerie... Dopo alcuni secondi si apre la finestra del Gestore librerie: nel campo di ricerca inserire "ntp"e selezionare e installare la libreria "NTPClient"



Lo scketch sull'ESP8266 NodeMCU

Codice ws_esp8266.ino

```
#include <NTPClient.h>  //libreria necessaria per ottenere data e ora da un Server NTP
#include <WiFiUdp.h>
//libreria per utilizzare il protocollo UDP  sulla porta 123 del Server NTP
#include <ESP8266WebServer.h>
#include <DHT.h>      //libreria necessaria per utilizzare il sensore DHT11
#define sensorePin D1
// pin di ESP8266 NodeMCU collegato all'uscita (pin OUT) del sensore DHT1
#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
// Configurazione WiFI: impostazione delle credenziali di rete
const char* ssid = "My_SSID";         //Inserire qui l'SSID
const char* password = "My_password"; //Inserire qui la password WiFi
/* Configurazione IP statico/fisso della scheda ESP8266. Occorre utilizzare un indirizzo
   IP disponibile  nella rete locale, il gateway corrispondente e la subnet mask
   Qui, ad esempio ho utilizzato l'IP 192.168.1.200
*/
IPAddress IP_ESP8266(192, 168, 1, 200);
IPAddress IP_gateway(192, 168, 1, 1);
IPAddress subnet_mask(255, 255, 0, 0);
//
ESP8266WebServer server(80);
/* Dichiara l'oggetto server della libreria ESP8266WebServer. Il costruttore di questo
   oggetto prende come parametro la porta  80 (predefinita per HTTP) dove il server si
   pone in ascolto
 */
WiFiUDP ntpUDP;   //Istanzia l'oggetto ntpUDP della classe WiFiUDP
NTPClient timeClient(ntpUDP, "pool.ntp.org");
// Definisce l'oggetto timeClient per recuperare data e ora dal Server NTP
//
float t;          //temperatura
float h;          //umidità
String dataora;   //data e ora della misura del sensore
//Array dei mesi per scrivere la data nel formato gg mese aa es 7 maggio 2021
String mesi[12]={"gennaio", "febbraio", "marzo", "aprile", "maggio", "giugno",
"luglio", "agosto", "settembre", "ottobre", "novembre", "dicembre"};
//
//
void setup()
{
    pinMode(sensorePin, INPUT);
    dht.begin(); //inizializza l'oggetto dht
    Serial.begin(115200);
    // apre una connessione seriale. A scopo di debug inviamo messaggi al monitor seriale
    delay(1000);
    Serial.println(ssid);
    //con WiFi.config() configura la scheda ESP8266 conl'IP statico sopra impostato
    if (!WiFi.config(IP_ESP8266, IP_gateway, subnet_mask))   {
        Serial.println("Errore nella configurazione");
    }
    WiFi.begin(ssid, password);//Inizializza le impostazioni di rete WiFi
    delay(1000);
    Serial.println(WiFi.status());
    while (WiFi.status()!= WL_CONNECTED)  {
        delay(500);
        Serial.print(".");
    }
```

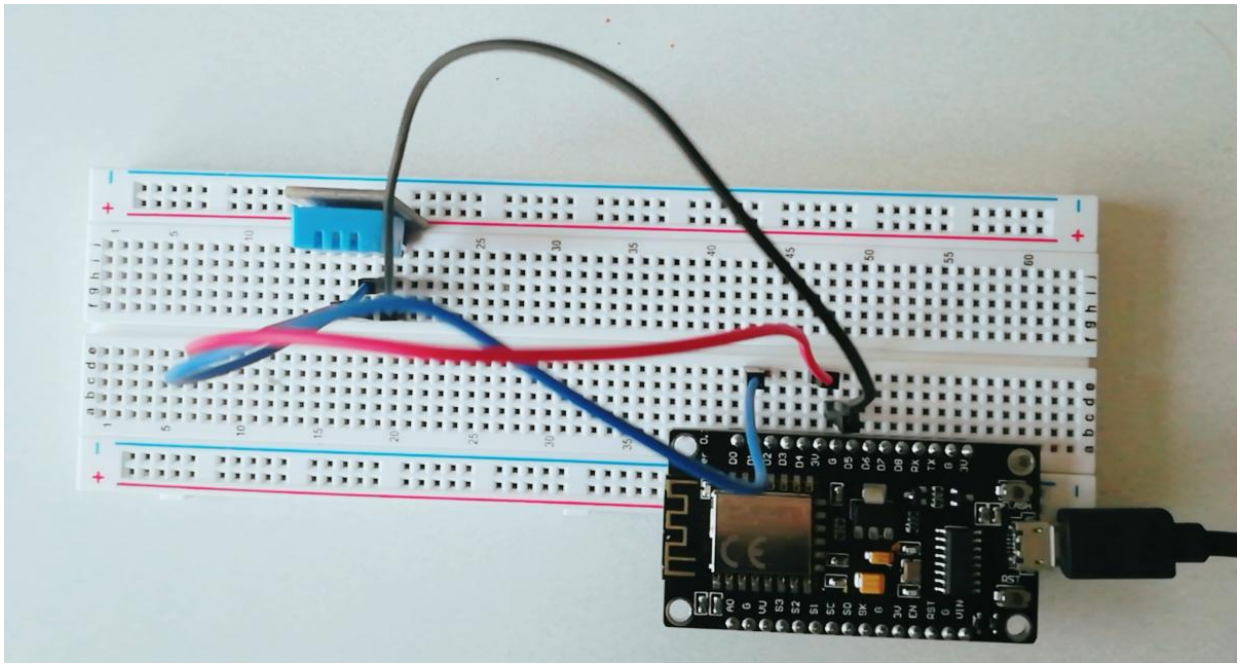
```
    Serial.println("");
    Serial.println("Connesso alla rete WIFI");
    Serial.println(IP_ESP8266);
    /*
      Per gestire le richieste HTTP in arrivo, dobbiamo specificare quale codice
      eseguire quando viene raggiunto un determinato URL. Per fare ciò, usiamo il metodo
      on. Questo metodo accetta due parametri. Il primo è un percorso URL e il secondo è
      il nome della funzione che vogliamo eseguire quando viene raggiunto quell'URL.
     */
    server.begin();
    // inizializzazione del Server Web. Comincia ad "ascoltare" le richieste HTTP
    delay(200);
    Serial.println("Web Server HTTP in funzione");
    server.on("/", handle_OnConnect);
    /*
      quando l'oggetto server riceve una richiesta HTTP sul percorso root (/) viene
      eseguita la funzione handle_OnConnect (potremmo chiamarla anche in altro modo)
     */
    server.onNotFound(handle_NotFound);
    timeClient.begin();// inizializzazione dell'NTPClient timeClient
    timeClient.setTimeOffset(7200); //7200 secondi
    //l'offset tiene conto della differenza oraria dell'Italia rispetto ad UTC (2 ore)
    //quando in Italia vige l'ora legale
 }
//
//
void loop()
{
    /*
      recupero data e ora dal server NTP. A volte l'NTPClient recupera la data
      "1 gennaio1970": forzando l'aggiornamento questo non accade
     */
    while(!timeClient.update())  {
      timeClient.forceUpdate();

    }
  server.handleClient();
  // rimane in ascolto sulla porta 80 per le richieste dei clients
  delay(1000);
}
//
//
void handle_OnConnect()
{
    // 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);
    }
    String tt=timeClient.getFormattedTime();
    unsigned long epoch = timeClient.getEpochTime();

    //dal valore di epoch si ricavano giorno, mese ed anno
    struct tm *ptm = gmtime ((time_t *)&epoch);
    int gg = ptm->tm_mday;
    int mm = ptm->tm_mon+1;
    String mese=mesi[mm-1];
    int aa = ptm->tm_year+1900;
    dataora = "Misura del "+String(gg) + " " + mese + " " + String(aa)+"  "+tt;
```

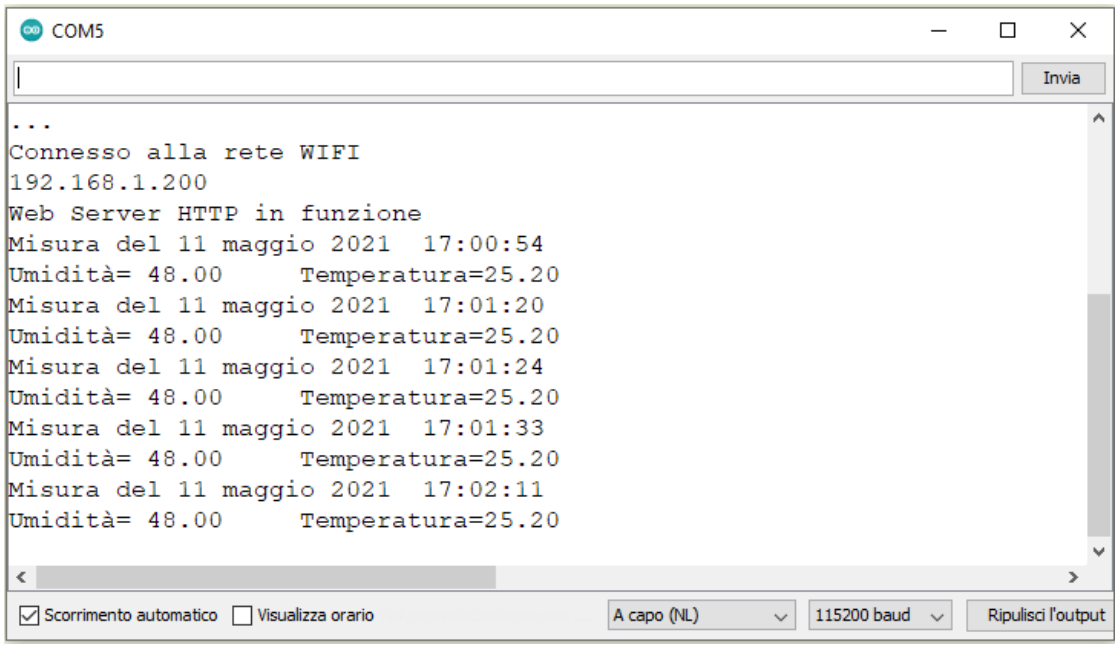
```
if (isnan(h) || isnan(t))  //Verifica se  si presenta un errore di lettura
    {
          Serial.println("Errore di lettura...");
          h=0.0;
          t=0.0;
    }
    String s="Umidità= "+String(h)+"     Temperatura="+String(t);
    Serial.println(dataora);
    Serial.println(s);
    server.send(200, "text/html", SendHTML(t,h,dataora));
   //invia  lo stato 200, cioè l'ok, e una pagina Web, costruita nella funzione
   //SendHTML(), al client/browser
}

void handle_NotFound(){
  server.send(404, "text/plain", "Non trovato!!");
  //invia lo stato 404, cioè l'errore "not found", e un semplice messaggio di errore al
  //client/browser
}
//
/*
  risposta al client/browser con il codice HTML che costruisce la pagina web per
  visualizzare data e ora della misura, temperatura ed umidità rilevate dal sensore
 */
String SendHTML(float temperatura,float umidita,String dataoramisura)
{
  String s = "<!DOCTYPE html> <html>\n";
  s +="<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0,
user-scalable=no\">\n";
  s +="<title>Server Web ESP8266 NodeMCU</title>\n";
  s +="<style>html {text-align: center;background-color:#fffff0}\n";
  s +="h2 {color: #000080;margin-top:30px;}\n";
  s +="p {font-size: 24px;margin-bottom: 10px;}\n";
  s +="</style>\n";
  s +="</head>\n";
  s +="<body>\n";
  s +="<h2>ESP8266 NodeMCU<br/>Server Web</h2><br/>\n";
  s+=dataoramisura;
  s +="<p>Temperatura: ";
  s+=temperatura;
  s+=" gradi</p>";
  s +="<p>Umidit&agrave;: ";
  s+=umidita;
  s +="%</p>";
  s +="</body>\n";
  s +="</html>\n";
  return s;
}
```

Visualizza lo sketch come pagina HTML

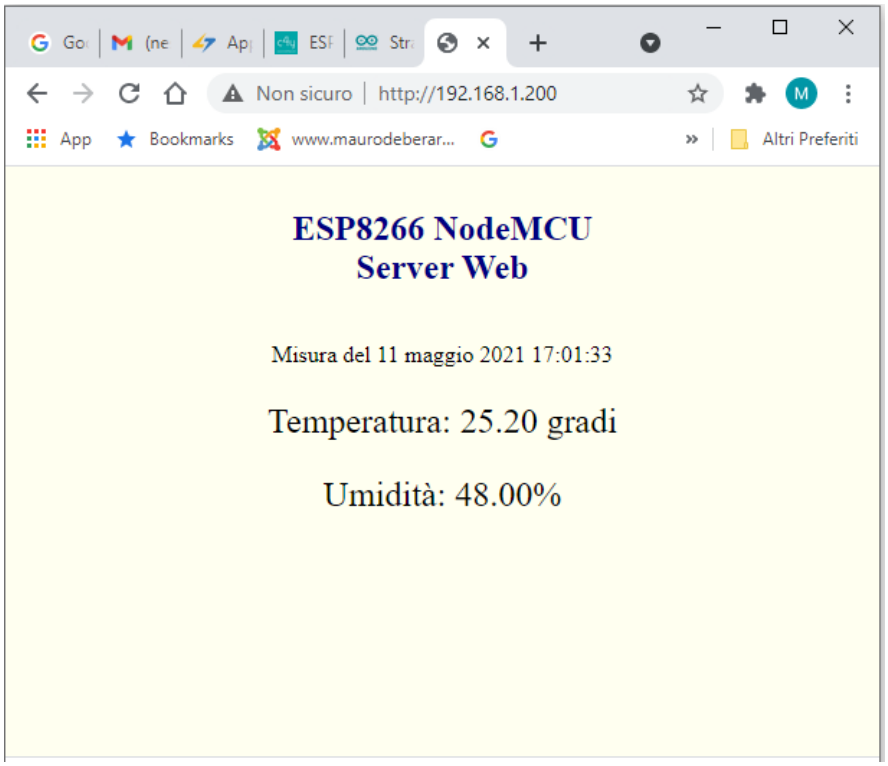
Test del Server Web

1. Caricare lo sketch su ESP8266 NodeMCU
2. Aprire il Monitor Seriale a 115200 baud
3. Premere il pulsante di reset della scheda ESP826 per avviare il Server http: se impiega più di qualche secondo, resettare la scheda una seconda volta
4. Accedere al Server da qualsiasi browser della rete WiFi, digitando l'indirizzo IP impostato nello sketch (192.168.1.200)





Accesso da un Pc desktop



Accesso da uno smartphone