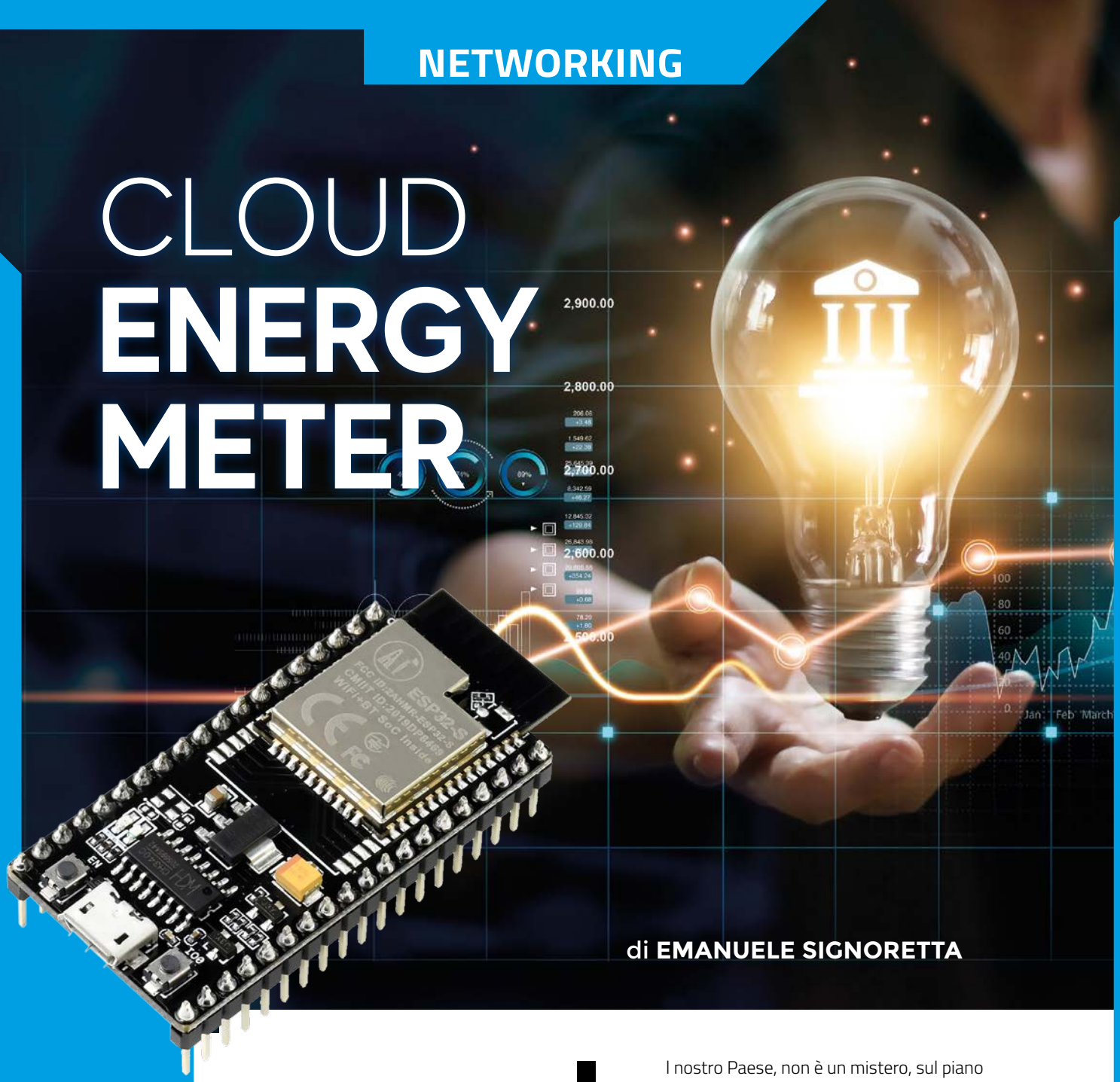
CLOUD ENERGY METER



di **EMANUELE SIGNORETTA**

Contatore di energia e assorbimento basato sulla scheda di prototipazione ESP32, che ci consente di interfacciarci con il cloud.

Il nostro Paese, non è un mistero, sul piano energetico è fortemente dipendente dall'estero perché l'autoproduzione, prettamente da fonti rinnovabili, è inadeguata a soddisfare il fabbisogno nazionale. Questo problema si acuisce ogni volta che fattori esterni causano la limitata disponibilità di fonti fossili e il conseguente incremento del loro prezzo. Sapendo un po' tutti quanto "salate" sono le bollette di luce, gas ci ritroviamo a ridurre i consumi, rinunciando ad alcuni "confort" e a tenerne comunque traccia, per evitare spiacevoli sorprese. Del resto l'attenzione al consumo energetico è mossa prima di tutto (e ce ne rammarichiamo...) dalla paura di spendere troppo, piuttosto che dalle preoccupazioni per i cambiamenti climatici. Comunque, per venire incontro a chi presta attenzione a quanta elettricità consuma, in questo articolo presenteremo un rilevatore di energia consumata, basato su un modulo

Fig. 1
La scheda ESP32.

ESP32 e su un sensore di tensione/corrente siglato PZEM004T, i cui dati verranno inviati in Cloud, nello specifico ai server del servizio on-line InfluxDb.

L'HARDWARE DEL PROGETTO

Per realizzare questo nostro contatore di energia utilizzeremo, per la parte hardware, un modulo WiFi ESP32, acquistabile sul sito *www.futurashop.it* con il codice HR0204 (**Fig. 1**), un sensore PZEM004T (lo vedete in **Fig. 2** incapsulato nel suo contenitore in plastica trasparente) un alimentatore switching con output 5V (se ne scegliete uno ad uscita microUSB potete alimentare con esso la scheda ESP e prendere da VIN l'alimentazione per il PZME004T...) e dei jumper femmina-femmina per Arduino, morsetti, alcuni spezzoni di cavo elettrico e un box di plastica (cod. KZ57J di *www.futurashop.it*). Il sensore di assorbimento è sostanzialmente un trasformatore di corrente dotato di case in plastica e toroide apribile, con un minimo di elettronica disaccoppiata mediante fotoaccoppiatori e interfaccia seriale.
Le scelte degli elementi circuitali sono state fatte in virtù dei costi dell'hardware rimasti comunque contenuti nonostante la crisi dei semiconduttori e per le ottime specifiche dell'ESP32.

Fig. 2
Il sensore PZEM004T con il trasformatore.

CONFIGURAZIONE DI INFLUXDB

Esistono diversi software che permettono di ricevere ed elaborare i dati trasmessi dai dispositivi IoT: non possiamo non citare Home Assistant, Grafana, Blynk, Thingspeak ecc. Per evitare di essere prolissi sull'utilizzo di alcuni software già trattati abbiamo deciso di introdurre l'utilizzo di InfluxDb, il cui logo è proposto nella **Fig. 3**.
InfluxDb è un software open source che permette di creare dashboard, eseguire query e inviare alert. È possibile utilizzarlo in diverse modalità: eseguibile (con numerose architetture supportate), container Docker, e Cloud. Per questioni di "proprietà dei dati" avremmo potuto installare un'istanza su Raspberry Pi, ma a causa dello shortage di componenti, ci siamo visti costretti a ricorrere al servizio **InfluxDb Cloud v2**.
Rechiamoci per prima cosa al link *https://cloud2.influxdata.com/signup* e registriamoci utilizzando o il social login di Microsoft e Google oppure compilando tutti i campi sottostanti (**Fig. 4**).
Una volta registrati al servizio ci apparirà una schermata come in **Fig. 5**, nella quale vengono richieste diverse informazioni, tra cui il provider per il servizio: per il nostro progetto abbiamo scelto Amazon Web Services (AWS). Nella schermata successiva (proposta nella **Fig. 6**) dovremo scegliere il piano di utilizzo: nel nostro caso abbiamo optato il piano free, che comprende archiviazione dati per una durata di 30 giorni e la possibilità di inviare le notifiche tramite Slack.
Confermata la nostra opzione ci apparirà la dashboard personale; da essa, per creare un bucket, andiamo su → **"Load Data"** → **Buckets"** → **"Create new bucket"**. Si aprirà una schermata come quella esemplificata nella **Fig. 7**, nella quale bisognerà compilare il campo relativo al nome e poi cliccare sul pulsante **"Create"**.
Una volta creato il bucket, quest'ultimo apparirà tra le data sources disponibili, come esemplificato nella **Fig. 8**.

Listato 1

```
#include <WiFiMulti.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>
#include <PZEM004Tv30.h>
#include <Every.h>
```

Clicchiamo dunque su **"Add data" → "Client library"**. Nella nuova schermata che si aprirà selezioniamo la voce **"Arduino"** (**Fig. 9**) tra le possibili alternative per l'upload dei dati.
Nella nuova finestra che si apre (**Fig. 10**) comparirà una serie di snippet che andranno a comporre uno sketch dimostrativo. Prendiamo in considerazione il primo di tali snippet e copiamo i dati relativi a *INFLUXDB_URL*, *INFLUXDB_ORG* e *INFLUXDB_BUCKET*.
Come ultimo passaggio è necessario generare un token di accesso al bucket; per fare ciò è necessario recarci sulla sidebar di sinistra e poi su **"Load Data" → "Api Tokens"**.
Nella nuova finestra che si aprirà clicchiamo su **"Generate API Token"** e una volta selezionato il bucket, scegliamo di abilitare i permessi in lettura e scrittura come in **Fig. 11**. Una volta generato il token, copiamolo e mettiamolo da parte poiché ci servirà tra poco.

LO SKETCH

Lo sketch del nostro progetto è frutto dell'unione di diversi sketch delle librerie utilizzate.
È possibile scaricare i file dello sketch mediante il link del repository GitHub raggiungibile dall'indirizzo *github.com/signorettae/ESP32-EnergyMeter*.
Procediamo adesso ad analizzare i nostri sorgenti, suddivisi in tre listati che esporremo, preceduti da una parte relativa all'inclusione delle librerie e delle dipendenze necessarie, proposta qui di seguito.

```
#include <WiFiMulti.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>
#include <PZEM004Tv30.h>
#include <Every.h>
```

A riguardo precisiamo che la libreria *WiFiMulti* serve a gestire la comunicazione WiFi tra l'ESP32 e l'access point, mentre le librerie *ESPmDNS*, *WiFiUDP* e *ArduinoOTA* (*https://github.com/jandrassy/ArduinoOTA*) servono a gestire il caricamento OTA (Over The Air) degli sketch e il nome dell'host.
Facciamo presente che per fare uso di questa funzionalità è necessario avere Python v. 2.7.x installato sul PC che utilizzerete.
Le librerie *InfluxDbClient* e *InfluxDbCloud* servono

Fig. 3 Logo di InfluxDb.

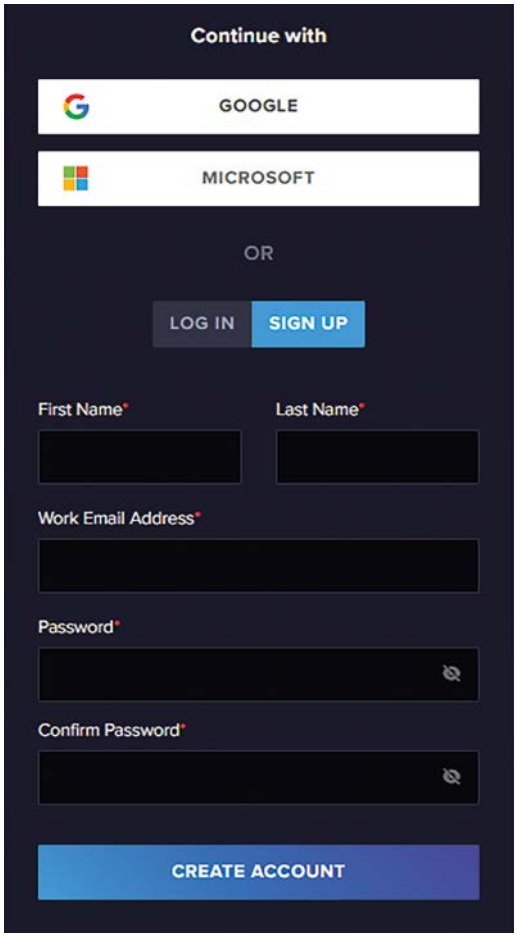


Fig. 4 Pagina di registrazione a InfluxDB.

Listato 2

```
#define REFRESH_TIME 5000 // Delay interval for data upload
#define DEVICE “ESP32”
#define PZEM_RX_PIN 27
#define PZEM_TX_PIN 26
#define PZEM_SERIAL Serial2

/*************************
    ESP32 initialization
   ---------------------

   The ESP32 HW Serial interface can be routed to any GPIO pin
   Here we initialize the PZEM on Serial2 with RX/TX pins 26 and 27
*/
PZEM004Tv30 pzem(PZEM_SERIAL, PZEM_RX_PIN, PZEM_TX_PIN);
WiFiMulti wifiMulti;
IPAddress local_IP(192, 168, 178, 154);
IPAddress gateway(192, 168, 178, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress primaryDNS(192, 168, 178, 1); //optional
IPAddress secondaryDNS(1, 1, 1, 1); //optional
// WiFi AP SSID
#define WIFI_SSID “”
// WiFi password
#define WIFI_PASSWORD “”
// InfluxDB v2 server url, e.g. https://eu-central-1-1.aws.cloud2.influxdata.com (Use: InfluxDB UI -> Load
Data -> Client Libraries)
#define INFLUXDB_URL “https://eu-central-1-1.aws.cloud2.influxdata.com”
// InfluxDB v2 server or cloud API token (Use: InfluxDB UI -> Data -> API Tokens -> Generate API Token)
#define INFLUXDB_TOKEN “”
// InfluxDB v2 organization id (Use: InfluxDB UI -> User -> About -> Common Ids )
#define INFLUXDB_ORG “”
// InfluxDB v2 bucket name (Use: InfluxDB UI ->  Data -> Buckets)
#define INFLUXDB_BUCKET “Energy Meter @ ElettronicaIN”
// Set timezone string according to https://www.gnu.org/software/libc/manual/html_node/TZ-Variable.html
#define TZ_INFO “CET-1CEST,M3.5.0,M10.5.0/3”
// InfluxDB client instance with preconfigured InfluxCloud certificate
InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG, INFLUXDB_BUCKET, INFLUXDB_TOKEN, InfluxDbCloud2CACert);
// Data point
Point sensor(“EnergyMeter”);
```

invece a gestire l’upload dei dati sul Cloud di Influx. La libreria PZEM004Tv30 (*github.com/mandulaj/PZEM-004T-v30*) viene utilizzata per la comunicazione seriale con il sensore e infine, la libreria Every (scaricabile dal sito della rivista) viene invece utilizzata per eseguire dei blocchi di codice ad intervalli regolari senza fare uso dei *delay()*.

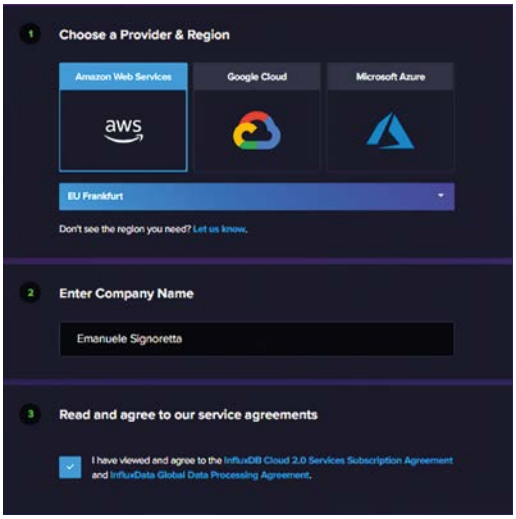


Fig. 5 Scelta del provider e accettazione dei termini di servizio.

Passiamo adesso ad analizzare le successive porzioni di codice, a iniziare da quella proposta nel **Listato 2**, la quale contiene la parte del preprocessore. Con *#define REFRESH_ TIME 5000* impostiamo l’intervallo di caricamento in millisecondi dei dati sul cloud. Vengono successivamente definiti il nome del dispositivo (lo ritroveremo successivamente), i pin per la comunicazione seriale e quale porta seriale vorremo utilizzare. Vengono in seguito creati gli oggetti relativi alla rete WiFi e al sensore. Ancora a seguire, vengono definiti i vari indirizzi IP per configurare una connessione all’AP con IP statico, cambiamo i parametri a seconda della subnet della nostra rete. Procediamo infine a definire i parametri per la connessione WiFi e per l’accesso al cloud di Influx.

Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

Elettronica In

oltre l'elettronica

www.elettronicain.it