## **Energy Consumption Monitoring System (E.C.M.S.)**

*Revisione n.1*

# **1. Introduzione al Progetto**

L'**Energy Consumption Monitoring System (E.C.M.S.)** è un sistema progettato per monitorare il consumo energetico degli elettrodomestici presenti nelle nostre abitazioni. Questo progetto fa uso di **Grafana** una piattaforma open source per il monitoraggio e l’analisi di dati, che consente la realizzazione di dashboard che permettono di avere una visione chiara e dettagliata dei consumi, facilitando la gestione e l'ottimizzazione delle risorse.

Un aspetto chiave del progetto è la semplicità di configurazione: chiunque può installarlo senza particolari difficoltà. Il sistema include un case stampato in 3D che ospita l'hardware necessario. La presa dell'elettrodomestico viene collegata direttamente al case, il quale, a sua volta, si connette alla rete elettrica domestica.

## 1.1 Componenti hardware

* **ESP8266**: Legge i dati provenienti dal sensore corrente ACS712 e li invia tramite socket UDP al raspberry. Gli ESP8266 verranno installati per ogni elettrodomestico/carico che si intende monitorare
* **ACS712:** Sensore di corrente, disponibile in diverse varianti (ad esempio, per 5A o 10A). L’uscita del sensore, che sarà collegata all'ESP8266 è di tipo analogica(proporzionale alla corrente che passa). Il progetto prevede case con la versione da 5A e case con la versione da 10A.
* **Batteria 9V:** Alimenta l'ESP8266.
* **Presa 220V AC:** Permette di collegare l'elettrodomestico al case
* **Spina 220V AC:** Si collega alla presa della rete elettrica domestica.
* **Raspberry pi4 8gb RAM:** è il cervello del sistema. Legge i dati provenienti dagli ESP, gestisce il database mysql, il server broker e la dashboard di grafana, fungendo da hub centrale per l'elaborazione e la visualizzazione delle informazioni.

## 1.2 Funzionamento

# **2. ESP8266**

## 2.1 Introduzione

Gli esp8266 si occupano di monitorare gli elettrodomestici tramite il sensore di corrente. Grazie alle loro dimensioni ridotte rispetto a un classico Arduino, offrono prestazioni maggiori e risultano ideali per l'integrazione nel case stampato in 3D. Nella seguente tabella si mettono a confronto le prestazioni delle due schede.

| **Caratteristica** | **ESP8266** | **Arduino (es. Uno)** |
| --- | --- | --- |
| Microcontrollore | Tensilica Xtensa LX106 (32-bit) | ATmega328P (8-bit) |
| Frequenza di Clock | 80 MHz (espandibile a 160 MHz) | 16 MHz |
| Connettività Wi-Fi | Integrata | Non disponibile (richiede un modulo esterno) |
| Memoria Flash | 4 MB (a seconda del modello) | 32 KB |
| RAM | 64 KB | 2 KB |
| GPIO disponibili | 11-17 (a seconda del modello) | 14 digitali, 6 analogici |
| Tensione operativa | 3.3 V | 5 V |
| Supporto ADC | 1 canale (10-bit) | 6 canali (10-bit) |
| Comunicazioni | UART, SPI, I2C | UART, SPI, I2C |
| Programmazione | IDE Arduino, PlatformIO | IDE Arduino |
| Supporto OTA | Sì (programmazione wireless) | No (richiede cavo USB o ICSP) |
| Costo | Generalmente economico (~3-5 €) | Più costoso (~10-25 €) |

Il funzionamento degli ESP8266 all’interno del progetto può essere suddiviso in diverse fasi operative:

1. **Access Point**
2. **Multicast DNS (mDNS)**
3. **Client NTP**
4. **Socket UDP**

## 2.2 Access Point

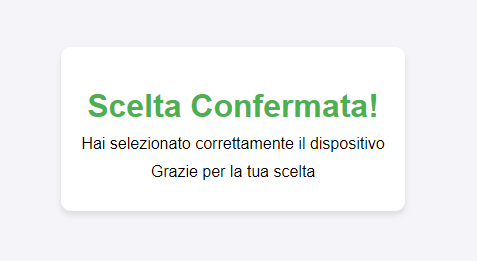
In questa fase, l’ESP8266 funge da access point, quindi crea una rete Wi-Fi locale aperta con **SSID**: “*ECMS\_device*”. Con un computer o uno smartphone, è necessario connettersi alla rete Wi-Fi generata dall’ESP8266., e digitando nella barra di ricerca del browser l’indirizzo ip dell’esp(192.168.1.2), verrà restituita una pagina web di configurazione del dispositivo:



Quindi l’ESP8266 rimane in attesa di richieste http, quando riceve una richiesta di tipo **GET,** restituisce la pagina html sopra riportata.

Dopo aver inserito l’SSID della rete Wi-Fi, la password e il dispositivo dal menù a discesa, cliccando sul tasto salva, il browser invia un’altra richiesta **HTTP** di tipo **GET,** inserendo nella **query string** dell’URL i parametri inseriti.

Quando l’ESP8266 riceve questa la richiesta e rileva la presenza di questi parametri al suo interno, restituisce la pagina di conferma e tenta di connettersi alla rete Wi-Fi identificata dall’SSID fornito nel form



Se la connessione va a buon fine, l’access point(esp) viene chiuso, altrimenti rimane aperto per ritentare l’inserimento dei dati.

Una parte essenziale del programma è la gestione dell’***URL encoding***. Quando l’SSID e la password vengono inviati tramite la **query string** dell’URL, è possibile che contengano caratteri non sicuri (ad esempio, spazi, simboli speciali come *&, =* o *?*) o caratteri che hanno un significato speciale all’interno dell’URL.

Per evitare problemi, i browser utilizzano l’**URL encoding** per codificare questi caratteri in un formato leggibile e sicuro, sostituendoli con sequenze codificate come *%20* (per uno spazio) o *%40* (per il simbolo *@*). Tuttavia, al momento della ricezione, l’ESP8266 deve decodificare queste sequenze per ripristinare i caratteri originali.Queste operazioni sono necessarie per garantire l’Integrità dei dati.

Nel programma, dopo aver ricevuto la query string, i caratteri codificati vengono convertiti nei loro equivalenti originali utilizzando una funzione di **decodifica URL**. Ad esempio:

* *%20* viene sostituito con uno spazio ( ).
* *%3A* viene sostituito con : (due punti).

Questa operazione è fondamentale per garantire che l’SSID e la password risultino "puliti" e pronti per l’uso nella connessione Wi-Fi.

E’ stata utilizzata una richiesta HTTP di tipo GET e non di tipo POST perchè più facile da implementare in C e non comporta rischi per la sicurezza essendo un ambiente locale utilizzato solo per connettersi al Wi-Fi domestico e per scegliere il dispositivo da monitorare.

## 2.3 Multicast DNS (mDNS)

Quando il sistema viene installato in un’abitazione, il Raspberry Pi si connette alla rete domestica tramite cavo Ethernet. Per evitare conflitti con gli indirizzi IP, il Raspberry utilizza un indirizzo IP dinamico assegnato da un server DHCP. Tuttavia, affinché l’ESP8266 possa inviare i pacchetti UDP al Raspberry Pi senza conoscere preventivamente il suo indirizzo IP, è necessario un metodo per identificarlo automaticamente nella rete locale.

Per risolvere questo problema, l’ESP8266 utilizza il **multicast DNS (mDNS)**, un protocollo progettato per risolvere i nomi dei dispositivi in reti locali senza la necessità di un server DNS centralizzato. Questo protocollo consente all’ESP8266 di cercare un dispositivo specifico tramite il nome host.

Il Raspberry Pi ha il supporto mDNS abilitato di default e viene identificato nella rete locale con il nome *pi.local*. Quando l’ESP8266 cerca un dispositivo con questo nome, il protocollo mDNS restituisce l’indirizzo IP del Raspberry Pi. L’ESP8266 può quindi utilizzare questo indirizzo per inviare i pacchetti UDP.

## 2.4 Client NTP

In questo progetto viene usato Grafana per la visualizzazione e il monitoraggio dei dati. I dati, quindi, vengono visualizzati all’interno di grafici e serie storiche. Per questo motivo, Grafana richiede che i dati vengano accompagnati da un timestamp(la data e l’ora relative all’acquisizione del dato).

Per garantire una corretta sincronizzazione temporale, l’ESP8266, nella fase di setup, si connette a un server NTP (Network Time Protocol), diventando un client NTP.

La classe **NTPClient** viene utilizzata per richiedere il timestamp al server ***pool.ntp.org***, un grande cluster di timeserver affidabili e di facile accesso, utilizzati da milioni di utenti in tutto il mondo.

Il timestamp restituito è in formato **UNIX**, ossia il numero di secondi trascorsi dalle ***00:00:00 GMT del 1° gennaio 1970***. Questo formato è standard in molte applicazioni e può essere facilmente convertito in un formato leggibile per data e ora locali.

Quindi ogni lettura del sensore avrà un timestamp associato che viene usato per tenere traccia del valore letto e per creare delle serie storiche.

## 2.5 Socket UDP

Ogni case(che rappresenta un’unità indipendente), dopo aver completato le fasi descritte nei paragrafi precedenti, inizia ad inviare tramite socket UDP le letture del sensore al server centrale(raspberry pi). Ogni pacchetto UDP inviato è composto da 12 byte:

* **6 byte -> indirizzo MAC**: usato per identificare univocamente il case che sta inviando dati. Questo è particolarmente utile per scopi di bug fixing e diagnostica.
* **4 byte -> timestamp:** Il timestamp viene inviato in formato **unix time**. Verrà convertito successivamente dal server centrale quando caricherà la data all’interno del database.
* **1 byte -> selected device:** indica l’elettrodomestico/carico che si sta monitorando. Utilizzato per scopi di diagnostica e per la visualizzazione delle dashboard in grafana.
* **1 byte -> sensor data:** contiene la lettura del sensore in Ampere.

I pacchetti UDP vengono inviati ogni 10 secondi. E’ stato implementato UDP e non altri protocolli orientati alla connessione come TCP, per mantenere il codice e la trasmissione il più veloci possibili. Poiché le letture vengono effettuate ogni 10 secondi e i consumi elettrici non variano significativamente in questo intervallo, la perdita occasionale di pacchetti non compromette la funzionalità del sistema. Utilizzando UDP, si evitano eventuali rischi di **overhead** perché TCP, oltre ad essere più lento, anche se affidabile, è anche più pesante da implementare.

# **3. Codice python**

Il programma in Python è composto principalmente da due file principali, ciascuno con responsabilità distinte. Entrambi i file richiamano eventuali classi o moduli necessari e vengono eseguiti in parallelo come processi daemon tramite ***systemctl***.

## 3.1 file main.py

Il file ***main.py*** si occupa di:

* Ricezione dei pacchetti UDP trasmessi dagli ESP8266 che monitorano i dispositivi. A tale scopo viene utiilizzato il modulo ***socket***, che utilizza il metodo ***recvfrom(nbyte)*** per ritornare una tupla contenente l’indirizzo ip del mittente e i dati in arrivo.
* Estrazione dei dati dal pacchetto UDP, comprendenti il MAC address, il timestamp, il dispositivo monitorato e il dato del sensore.
* Pubblicazione dei dati estratti su un topic di un broker MQTT remoto (utilizzando la piattaforma **EMQX**). A tale scopo è stata creata la classe ***mqtt\_publisher*** che gestisce interamente la pubblicazione dei dati sul broker mqtt, per mantenere il codice più pulito e facilmente leggibile possibile.

Per le operazioni di pubblicazione e di sottoscrizione viene usata la libreria ***paho.mqtt.*** I dati estratti dai pacchetti UDP vengono pubblicati sul broker MQTT in formato **json** utilizzando la libreria python **json**.

## 3.2 file mqtt\_to\_mysql.py

## Il file *mqtt\_to\_mysql.py* si occupa di:

## Si sottoscrive al topic configurato nel broker MQTT remoto (EMQX).

## Ogni volta che riceve un messaggio, estrae i dati e li inserisce in un database remoto ospitato su Aiven, garantendo la persistenza e la disponibilità per la visualizzazione e l’analisi. I dati(inseriti in formato json) vengono estratti singolarmente. L’inserimento dei dati nel database viene gestito dalla classe *mysqlConnect,* per mantenere il codice più pulito e facilmente leggibile possibile. La libreria usata per gestire il database è *mysql.* Il timestamp, quando viene inserito nel database, deve essere convertito nel tipo *date* utilizzando la funzione sql *FROM\_UNIXTIME(timestamp)*.

## 3.3 Considerazioni

Può sembrare una ridondanza utilizzare il protocollo MQTT e quindi caricare e prelevare i dati dal broker MQTT. Tuttavia, questa scelta offre una maggiore scalabilità:

* In futuro, i dati pubblicati sul broker MQTT potrebbero essere utilizzati per altre analisi in tempo reale, integrazioni con dashboard esterne, o notifiche push.
* Separando la logica di ricezione e pubblicazione (***main.py***) da quella di elaborazione e salvataggio (***mqtt\_to\_mysql.py***), il sistema risulta modulare e più semplice da mantenere.

# **4. Dashboard in Grafana**

Grafana è un potente strumento open source che consente di creare dashboard dinamiche, con serie storiche per visualizzare i dati e monitorarli in tempo reale. È ampiamente utilizzato in ambiti come il monitoraggio delle infrastrutture, l'analisi delle performance applicative e il controllo di sistemi IoT. Grazie alla sua versatilità, Grafana supporta diversi tipi di database (come **MySQL**, InfluxDB, e Prometheus) e offre una vasta gamma di pannelli configurabili per rappresentare i dati in modo chiaro e intuitivo.

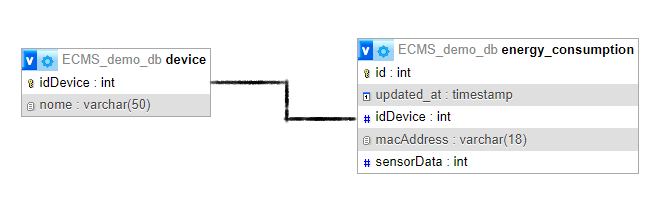
Una delle sue caratteristiche principali è la possibilità di combinare dati provenienti da diverse fonti in un'unica dashboard, permettendo una visione completa e integrata. Inoltre, Grafana è progettato per essere altamente personalizzabile, sia nell'aspetto che nelle funzionalità, grazie ai numerosi plugin disponibili.

In questo caso, i dati sono memorizzati e prelevati da un database remoto MySQL ospitato sui server **AIVEN.**

## 4.1 Il database

La struttura del database utilizzata nel progetto è semplice e composta da due tabelle principali, collegate da una relazione 1-M (uno a molti):

* **Device**: Questa tabella rappresenta gli elettrodomestici monitorati. Ogni riga identifica un dispositivo unico e include i seguenti campi:
  + **idDevice**(***PK***)
  + **nome**(**varchar(50)**)
* **Data**: Questa tabella contiene i dati raccolti dai sensori per ciascun dispositivo monitorato. È collegata alla tabella Device tramite una chiave esterna. E’ composta dai seguenti campi:
  + **id**(***PK*)**
  + **updated\_at**(**timestamp**): contiene l’ora e la data relativa al dato raccolto
  + **idDevice(*PK):*** contiene l’ID del device che ha prodotto il valore memorizzato
  + **macAddress**(**varchar(18)**): contiene l’indirizzo MAC della scheda ESP8266 che ha letto il valore
  + **sensorData(INT):** contiene il dato letto dal sensore



## 4.2 La Dashboard



Questa dashboard è progettata per monitorare e analizzare il consumo energetico di diversi dispositivi elettronici in tempo reale. Contiene le seguenti metriche:

* **Serie storica del consumo di Ampere:** Mostra l'andamento del consumo di corrente (in Ampere) nel tempo, permettendo di identificare picchi o variazioni significative. Ad esempio, risulta utile per indicare in quale momento della giornata c’è un consumo maggiore.
* **Serie storica consumo dei Watt**: Visualizza il consumo di potenza (in Watt) nel tempo, utile per capire l'efficienza energetica dei dispositivi. Nella versione attuale del programma (rev. 1), vengono monitorati solo dispositivi che operano con una differenza di potenziale di 220V in corrente alternata, tipica delle reti elettriche domestiche in molti paesi. Questa metrica risulterà particolarmente utile in implementazioni future, quando sarà possibile monitorare dispositivi con differenze di potenziale differenti (ad esempio, 110V o 380V). Ciò permetterà di confrontare l'efficienza energetica di dispositivi con diverse tensioni di funzionamento, fornendo insights più completi e versatili.
* **Gauge Watt cumulativi**: Rappresenta il consumo totale di energia (in Watt) accumulato in tempo reale.
* **Media Wh giornalieri**: Fornisce una media dei Wattora (Wh) consumati da ciascun dispositivo nell'arco della giornata, permettendo confronti settimanali e analisi statistiche.
* **Gauge per il throughput**: Indica il numero di pacchetti inviati al minuto da ciascun ESP8266, utile per il debugging e per identificare problemi di rete.

La dashboard permette di selezionare un periodo di tempo specifico, riducendo il carico sul database e migliorando le prestazioni. Questa funzionalità è particolarmente utile per focalizzare l'analisi su intervalli di interesse specifici.

Nella dashboard sono stati usati i seguenti tipi di grafici:

* **Grafici a linee**: Utilizzati per mostrare l'andamento temporale del consumo.
* **Grafici a barre**: Utili per confrontare i consumi tra diversi dispositivi

## 4.3 Interrogazioni SQL

Grafana contiene un interprete SQL integrato che permette di eseguire query direttamente sul database, restituendo i dati necessari per la visualizzazione nei grafici. Ogni grafico nella dashboard è associato a una specifica interrogazione SQL, progettata per estrarre e formattare i dati in modo appropriato.

### ***4.3.1 Serie storica del consumo di Ampere e del consumo di Watt***

Le query per la serie storica del consumo di Ampere e del consumo di Watt sono molto simili, ma differiscono nel calcolo della conversione del carico da Ampere a Watt. Entrambe le query selezionano il timestamp, il nome del dispositivo e il valore del sensore, filtrando i dati in base all'intervallo di tempo selezionato.

SELECT

UNIX\_TIMESTAMP(energy\_consumption.updated\_at) \* 1000 AS "time",

device.nome AS "metric",

energy\_consumption.sensorData AS "value"

FROM

energy\_consumption, device

WHERE

device.idDevice = energy\_consumption.idDevice AND

energy\_consumption.updated\_at BETWEEN $\_\_timeFrom() AND $\_\_timeTo()

ORDER BY

energy\_consumption.updated\_at;

*query per la serie storica del consumo di Ampere*

SELECT

UNIX\_TIMESTAMP(energy\_consumption.updated\_at) \* 1000 AS "time",

device.nome AS "metric",

energy\_consumption.sensorData AS "value"

FROM

energy\_consumption, device

WHERE

device.idDevice = energy\_consumption.idDevice AND

energy\_consumption.updated\_at BETWEEN $\_\_timeFrom() AND $\_\_timeTo()

ORDER BY

energy\_consumption.updated\_at;

*query per la serie storica del consumo di Watt*

Le funzioni ***$\_\_timeFrom()*** e ***$\_\_timeTo(),*** che sono funzioni interne a Grafana, sono utilizzate per restituire l’intervallo di tempo e quindi filtrare i dati in base all'intervallo di tempo selezionato dall'utente nella dashboard. Questo permette di visualizzare solo i dati rilevanti per il periodo di interesse, migliorando l'efficienza delle query e la leggibilità dei grafici.

### ***4.3.2 Gauge Watt cumulativi***

SELECT sum(energy\_consumption.sensorData) \* 230 AS "VALUE"

FROM energy\_consumption

WHERE TIMESTAMPDIFF(SECOND, energy\_consumption.updated\_at, CURRENT\_TIMESTAMP) <=11;

Questa query è progettata per calcolare il consumo cumulativo di energia in Watt in tempo reale, aggiornandosi ogni 11 secondi(Gli ESP8266 inviano i dati ogni 10 secondi). Il gauge visualizza la somma dei valori dei sensori (sensorData) moltiplicati per 230 (differenza di potenziale in Volt, tipica delle reti domestiche a 220V-230V), per convertire il carico da Ampere a Watt.

### 

### 

### ***4.3.3 Media Wh giornalieri per dispositivo***

SELECT

DATE(energy\_consumption.updated\_at) as "date",

COALESCE(AVG(CASE WHEN device.nome = 'lavatrice' THEN sensorData ELSE NULL END), 0) AS lavatrice,

COALESCE(AVG(CASE WHEN device.nome = 'frigorifero' THEN sensorData ELSE NULL END), 0) AS frigorifero,

COALESCE(AVG(CASE WHEN device.nome = 'lavastoviglie' THEN sensorData ELSE NULL END), 0) AS lavastoviglie,

COALESCE(AVG(CASE WHEN device.nome = 'asciugatrice' THEN sensorData ELSE NULL END), 0) AS asciugatrice,

COALESCE(AVG(CASE WHEN device.nome = 'condizionatore' THEN sensorData ELSE NULL END), 0) AS condizionatore,

COALESCE(AVG(CASE WHEN device.nome = 'forno' THEN sensorData ELSE NULL END), 0) AS forno,

COALESCE(AVG(CASE WHEN device.nome = 'micronde' THEN sensorData ELSE NULL END), 0) AS micronde

FROM energy\_consumption, device

WHERE

energy\_consumption.idDevice = device.idDevice AND

DATE(energy\_consumption.updated\_at) BETWEEN DATE($\_\_timeFrom()) AND DATE($\_\_timeTo())

GROUP BY DATE(energy\_consumption.updated\_at)

ORDER BY DATE(energy\_consumption.updated\_at);

La query effettua le seguenti operazioni:

1. **Selezione della data** (***DATE(energy\_consumption.updated\_at)***):
   * La query estrae la data dalla colonna updated\_at della tabella *energy\_consumption*. Questo permette di raggruppare i dati per giorno.
2. **Calcolo della media giornaliera (*AVG*):**
   * Per ciascun dispositivo (es. lavatrice, frigorifero, condizionatore), la query calcola la media giornaliera del valore del sensore (*sensorData*).
   * La funzione ***COALESCE*** viene utilizzata per restituire 0 nel caso in cui non ci siano dati per un determinato dispositivo in un giorno specifico. Questo evita che il risultato contenga valori NULL.
3. **Filtro per dispositivo (*CASE WHEN*):**
   * La clausola *CASE WHEN* filtra i dati in base al nome del dispositivo (*device.nome*). Per ogni dispositivo, viene calcolata la media solo dei valori corrispondenti.
4. **Filtro temporale (*BETWEEN DATE($\_\_timeFrom()) AND DATE($\_\_timeTo())*)**:
   * La query filtra i dati in base all'intervallo di tempo selezionato dall'utente nella dashboard. Le variabili *$\_\_timeFrom()* e *$\_\_timeTo()* sono fornite da Grafana e rappresentano l'inizio e la fine del periodo selezionato.
5. **Raggruppamento per data (GROUP BY DATE(*energy\_consumption.updated\_at*)):**
   * I dati vengono raggruppati per giorno, in modo da visualizzare la media giornaliera per ciascun dispositivo.
6. **Ordinamento per data (*ORDER BY DATE(energy\_consumption.updated\_at)*)**:
   * I risultati vengono ordinati in base alla data, garantendo che i dati siano visualizzati in ordine cronologico.

La tabella restituita è la seguente:



### 

### ***4.3.4* Gauge per il throughput**

SELECT device.nome as "metric", count(\*) as "value"

FROM energy\_consumption, device

WHERE device.idDevice = energy\_consumption.idDevice AND

TIMESTAMPDIFF(SECOND, energy\_consumption.updated\_at, CURRENT\_TIMESTAMP) < 60

GROUP BY device.nome;

Questa query calcola il throughput, ovvero il numero di pacchetti di dati inviati da ciascun dispositivo negli ultimi 60 secondi. Utilizzando la funzione ***COUNT(\*)***, la query conta i record presenti nella tabella *energy\_consumption* per ogni dispositivo, filtrando solo quelli ricevuti nell'ultimo minuto grazie alla condizione ***TIMESTAMPDIFF(SECOND, energy\_consumption.updated\_at, CURRENT\_TIMESTAMP) < 60***.

I risultati vengono raggruppati per dispositivo (***GROUP BY device.nome***), permettendo di visualizzare il throughput separatamente per ciascun dispositivo.

Il gauge risultante è uno strumento utile per monitorare l'attività dei dispositivi in tempo reale, verificando che i dati vengano inviati regolarmente e senza interruzioni. Ad esempio, un throughput basso o nullo potrebbe indicare problemi di comunicazione o malfunzionamenti, mentre un valore stabile conferma il corretto funzionamento del sistema.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 4.4 Gli allert

Una delle funzionalità chiave del sistema Energy Consumption Monitoring System (E.C.M.S.) è la possibilità di configurare allert (notifiche) all'interno delle dashboard di Grafana. Gli allert sono strumenti essenziali per monitorare il consumo energetico in tempo reale e prevenire situazioni critiche, come il superamento della potenza contrattuale del contatore elettrico. In questa sezione, verrà descritto come è stato configurato un allert semplice nella dashboard dei Watt cumulativi, che avvisa l'utente quando il consumo si avvicina al limite di 3 kW, e come questa funzionalità si integra con il sistema preconfigurato e le opzioni avanzate per utenti esperti.

### **4.4.1 Allert Pre Configurato per i Watt Cumulativi**

Nella dashboard di Grafana, è stato configurato un allert pre configurato per monitorare i Watt cumulativi. Questo allert è stato pensato per avvisare l'utente quando il consumo energetico totale si avvicina al limite di 3 kW, tipico della potenza contrattuale di molte abitazioni. Questo valore è stato scelto come soglia di sicurezza per dare all’utente il tempo di intervenire prima che il consumo raggiunga i 3kW, evitando il distacco della corrente. L’allert è associato al gauge dei ***Watt cumulativi,*** che visualizza il consumo totale di energia in tempo reale.

Quindi quando il consumo supera i 2.5kW, per più di *30s* l’allert si attiva e viene visualizzato un avviso sulla dashboard. L’allert è configurato per rimanere attivo fino a quando il consumo non scende al di sotto della soglia di 2.5kW.

L'allert è stato configurato utilizzando la funzione di alerting di Grafana. Ecco come è stato configurata la regola di allert:

*SELECT sum(energy\_consumption.sensorData) \* 230 AS "VALUE"*

*FROM energy\_consumption*

*WHERE TIMESTAMPDIFF(SECOND, energy\_consumption.updated\_at, CURRENT\_TIMESTAMP) <= 11;*

Questa query calcola il consumo cumulativo in Watt e attiva l'allert se il valore supera 2.5 kW.

### **4.4.3 Configurazione Avanzata degli Allert per Utenti Esperti**

Sebbene il sistema sia preconfigurato con allert semplici, è possibile configurare allert personalizzati per utenti con competenze tecniche avanzate. Questa funzionalità è pensata per utenti che desiderano monitorare situazioni specifiche o configurare soglie di allert personalizzate.

Gli utenti esperti possono accedere alle impostazioni avanzate di Grafana e creare nuove regole di allert. Ad esempio, un utente potrebbe voler configurare un allert che scatta quando il consumo di un dispositivo specifico (ad esempio, il condizionatore) supera una certa soglia.

Inoltre, è possibile aggiungere anche un server SMTP, per creare degli allert inviati da Grafana tramite **email.**

Quindi gli allert si integrano perfettamente con questa funzionalità, aggiungendo un livello di monitoraggio proattivo al sistema.

#### **Esempio di Configurazione Avanzata:**

**Serie Storiche e Allert:**

* Le serie storiche del consumo di Ampere e Watt (descritte nel paragrafo 4.3.1) possono essere associate a regole di allert per monitorare picchi di consumo o anomalie.
* Ad esempio, un allert potrebbe essere configurato per scattare quando il consumo di un dispositivo supera una certa soglia per un periodo prolungato.

**Gauge dei Watt Cumulativi:**

* Il gauge dei Watt cumulativi (descritto nel paragrafo 4.3.2) è associato all'allert preconfigurato che avvisa l'utente quando il consumo si avvicina ai 3 kW.

**Throughput e Allert:**

* Il gauge per il throughput (descritto nel paragrafo 4.3.4) potrebbe essere associato a un allert che avvisa l'utente se il numero di pacchetti inviati da un dispositivo scende sotto una certa soglia, indicando un possibile malfunzionamento.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## **5. Stampa 3D**

Il sistema viene installato tramite delle BOX stampate in 3d. Sono inserite in serie agli elettrodomestici. La spina dell’elettrodomestico, viene inserita nella presa della box, mentre la spina della box viene inserita nella presa di casa. Questa caratteristica permette al sistema di essere installato facilmente su qualsiasi elettrodomestico/carico da monitorare.

Il case è stato progettato per essere stampato in PLA, un materiale biodegradabile e facile da stampare. Si consiglia di utilizzare un riempimento del 20% e uno spessore degli strati di 0.2 mm per garantire una buona resistenza e una finitura superficiale uniforme.

Per fissare i componenti che conterrà il case, le viti sono avvitate all’interno di appositi inserti per plastica filettati, inseriti all’interno di fori di fissaggio.

Il disegno in 3D è stato progettato tramite il software Fusion 360, ed è diviso in due parti:

## 

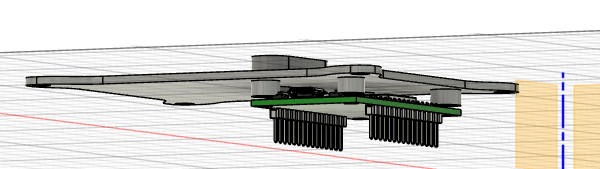
## 

## 

## 5.1 Up Case

Il case superiore serve a chiudere la box tramite 4 viti M3(3mm). Esso mantiene sia l’ESP8266, fissato tramite 4 viti M3, rivolto verso l’interno del case, sia la spina che è rivolta verso l’esterno del case. Ha le seguenti dimensioni:

* **Larghezza**: 70mm
* **Altezza**: 85mm
* **Estrusione**: 2mm
* **Spessori**: Tutti gli spessori sono di 2 mm.(Esclusi gli spessori per il fissaggio degli inserti delle viti)



## 

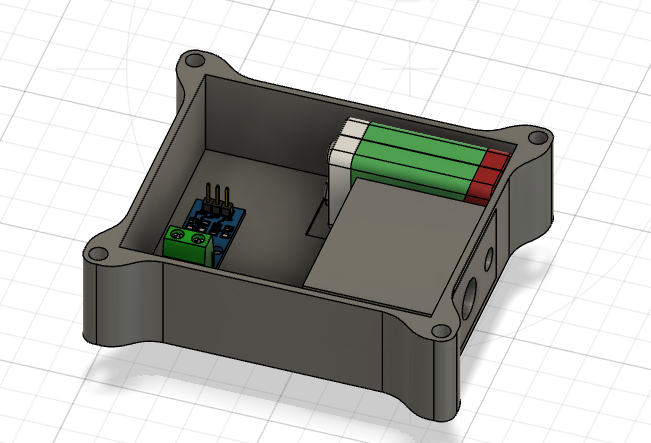
## 

## 

## 5.1 Bottom Case

Il case inferiore serve a chiudere la box tramite 4 viti M3(3mm). Contiene la presa, il sensore **ACS712** fissato tramite una vite di fissaggio M2e la batteria da 9V. Ha le seguenti dimensioni:

* **Larghezza**: 70mm
* **Altezza**: 85mm
* **Estrusione**: 28mm
* **Spessori**: Tutti gli spessori sono di 2 mm.(Esclusi gli spessori per il fissaggio degli inserti delle viti)



*Bottom Case*

## 

## 

## 5.2 Collegamenti elettrici

Il case contiene diversi componenti che devono essere collegati tra loro:

* **ACS712:** 
  + **Vcc:** Da collegare ai 5V dell’ESP8266
  + **GND:** Da collegare al GND dell’ESP8266
  + **Output:** Da collegare ad A0 dell’ESP8266
* **Spina 220V:** 
  + **Fase:** Da collegare alla fase(input) del sensore
  + **Neutro:** Da collegare al neutro della presa 220
* **Presa 220V:**
  + **Fase:** Da collegare alla fase(output) del sensore
  + **Neutro:** Da collegare al neutro della spina
* **Batteria 9V**
  + **+:** Da collegare al Vin dell’ESP8266
  + **-**: Da collegare a GND dell’ESP8266

## 5.3 Avvertenze per la sicurezza

Il sistema opera a 230V AC, che può essere pericoloso o letale se maneggiato in modo improprio. Quindi non si deve aprire o modificare il case quando è collegato alla rete elettrica.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **6. Configurazione multi-utente**

Il progetto Energy Consumption Monitoring System (E.C.M.S.), nella sua configurazione attuale, è stato pensato come un sistema prototipale per fini didattici, progettato per monitorare il consumo energetico in un'unica realtà (ad esempio, una singola abitazione). Tuttavia, è interessante esplorare come il sistema potrebbe essere esteso per supportare una configurazione multi-utente, ovvero la possibilità di monitorare più realtà (ad esempio, più abitazioni o aziende) utilizzando lo stesso sistema centrale. Questo capitolo propone una soluzione prototipale per una configurazione multi-utente, senza entrare nei dettagli implementativi, ma fornendo un'idea chiara di come potrebbe funzionare.

### **6.1 Configurazione del Database MySQL**

#### **Soluzione Proposta: Una Tabella per Ogni Utente**

Per gestire più utenti in un unico database, si potrebbe adottare una soluzione basata su tabelle separate per ogni utente. Questa scelta è motivata dalla necessità di mantenere i dati isolati tra gli utenti, garantendo al contempo una gestione semplice e scalabile del database.

Quando il sistema viene installato per un nuovo utente, il server centrale (Raspberry Pi) richiede all'utente di inserire un nome univoco (ad esempio, il nome dell'utente o un identificatore univoco). Questo nome servirà come chiave per identificare l'utente all'interno del sistema.

Sulla base di questo nome, il sistema crea automaticamente una nuova tabella nel database MySQL, con una struttura identica a quella già esistente. Ad esempio, se l'utente si chiama "Mario Rossi", il sistema creerà la tabella *data\_mario\_rossi*.

Ogni utente avrà la propria tabelle data, che memorizza i dati di consumo energetico.

Questo approccio garantisce l'isolamento dei dati tra gli utenti, poiché ogni utente ha accesso solo alla propria tabella. Questo migliora la sicurezza e la privacy. Inoltre aggiungere nuovi utenti è semplice, poiché richiede solo la creazione di una nuova tabella. Ma questo approccio rende più complesse le query, perchè se si vuole eseguire una query su tutti gli utenti(ad esempio, per scopi statistici), potrebbe essere necessario unire più tabelle, il che potrebbe rendere le query più complesse e meno efficienti.

### **6.2 Configurazione del Database MySQL**

#### **Soluzione Proposta: Unico Server MQTT con Topic Separati per Ogni Utente**

Per il broker MQTT, si potrebbe utilizzare un unico server MQTT, ma con topic separati per ogni utente. Questa scelta è motivata dalla necessità di utilizzare un unico server broker MQTT per tutti gli utenti, evitando uno spreco di risorse hardware. Quindi ogni utente pubblica(***publish***) i dati su topic differenti(***utente1/sensori, utente2/sensori***) e si sottoscrive(***subscribe***) al medesimo topic.

Questo approccio è più semplice da gestire, rispetto ad avere server separati e consuma meno risorse hardware.   
Però, se il server si guasta, tutti gli utenti risentono del guasto. Quindi bisognerebbe implementare delle politiche di ***business continuity*** e ***disaster recovery.***

Per implementare questa soluzione, si deve garantire che un client possa pubblicare o sottoscriversi solo al proprio topic. Quindi il broker MQTT dovrebbe supportare la gestione delle ***autorizzazioni(ACL- Access Control List)***. Ad esempio, utilizzando **EMQX(**un broker MQTT open source), è possibile configurare le autorizzazioni in modo che ogni utente possa accedere solo ai propri topic. Inoltre il sistema potrebbe utilizzare una funzione di creazione dinamica dei topic quando un nuovo utente si registra.

### **6.3 Configurazione di Grafana**

Per rendere multi-utente Grafana, si può sfruttare il sistema di autenticazione nativo di **Grafana**, in cui ogni utente si registra con il proprio account. Questo approccio presenta diversi vantaggi:

* **Facilità di implementazione**: Grafana ha già un sistema di autenticazione integrato.
* **Gestione separata delle dashboard**: Ogni utente può avere la propria istanza delle dashboard senza interferenze.
* **Possibilità di assegnare ruoli**: Se necessario, Grafana permette di definire permessi diversi (Admin, Editor, Viewer).

Quindi il processo funzionerebbe in questo modo:

1. L'utente crea un nuovo account in Grafana.
2. L'amministratore del sistema fornisce un file JSON contenente la configurazione delle dashboard.
3. L'utente importa il file JSON nella propria istanza di Grafana, ottenendo una dashboard identica a quella originale.
4. L'utente può personalizzare la visualizzazione e impostare eventuali filtri.

Questa soluzione ha il vantaggio di **essere semplice da implementare** e permettere agli utenti di avere un controllo diretto sulla propria dashboard.

### 

# 

# 

# 

# 

# 

# 

# 

# **6. Implementazioni future**

***Ostoni Christian Giuseppe***

# 