UNIVERSITÀ degli STUDI di CATANIA

Dipartimento di Matematica e Informatica

Corso di Laurea in Informatica

TESI DI LAUREA TRIENNALE

Davide Scalisi

PowerMonitor

PROGETTAZIONE DI UN SISTEMA DI MONITORAGGIO
E PREDIZIONE DI CONSUMI ENERGETICI

Relatori:
Prof. Corrado Santoro
Prof. Federico Fausto Santoro

ANNO ACCADEMICO 2020/2021

Settembre 2021

Sommario

Abstract

The possibility of being able to monitor and adjust the energy consumption accordingly, from the perspective of private users and the entire organization, is nowadays necessary, and the right choice in this regard may have different effects on profits.

Energy consumption monitoring allows to identify the main key points and weaknesses of the company's system, which is essential for the implementation of effective energy consumption management methods.

Monitoring is based on the actual measurement, recording and processing of the electrical absorption parameters of one or more specific devices, allowing the decision-making on the subsequent processing of these data; proper energy management requires the use of a monitoring system based on scalable hardware and software systems to ensure its effectiveness and maintainability.

In the development of this paper, all methods and standards behind the development of the project PowerMonitor will be involved: a domestic open-source hardware and software ecosystem for measurement, display, and energy consumption prediction over time.

Prefazione

La possibilità di poter monitorare e regolare di conseguenza i propri consumi energetici, al giorno d'oggi è una necessità, sia dal punto di vista di un utente privato, sia da quello di un'intera organizzazione, dove le giuste scelte in questo ambito potrebbero fare la differenza in termini di profitti.

L'attività di monitoraggio dei consumi energetici, che consente di individuare le principali criticità ed i punti deboli del sistema aziendale, è indispensabile per poter attuare un'efficace metodologia di gestione dei consumi energetici.

Il monitoraggio, che si basa sull'effettiva misurazione, registrazione ed elaborazione dei parametri di assorbimento elettrico di una o più determinate apparecchiature, permette di prendere delle decisioni basate sulla conseguente elaborazione di questi dati; una corretta gestione dell'energia richiede l'utilizzo di un sistema di monitoraggio che si basi su sistemi hardware e software scalabili al fine di garantirne l'efficacia e la manutenibilità nel tempo.

Durante lo svolgimento di questa Tesi, verranno trattate tutte le metodologie ed i criteri che stanno dietro allo sviluppo del progetto PowerMonitor: un ecosistema hardware e software open-source domestico, destinato alla misurazione, visualizzazione, nonché alla predizione di consumi energetici nel tempo.

1. Monitoraggio dei consumi

1.1. La nascita dell'esigenza

Il monitoraggio dei consumi energetici in ambito domestico consente di poter valutare e favorire decisioni che spingano verso un risparmio economico, soprattutto in funzione del fatto che, in futuro, la casa della famiglia media vedrà aumentare l'utilizzo di nuovi elettrodomestici, come ad esempio la sempre più popolare e comoda cucina ad induzione, basata sul principio dell'induzione elettromagnetica.

1.2. Caratteristiche di un sistema di monitoraggio energetico domestico

Un sistema di monitoraggio energetico domestico generico permette di:

1. Controllare in tempo reale ed in maniera continua lo stato dei consumi di uno o più dispositivi collegati all'impianto elettrico.
2. Controllare e perfezionare (manualmente o automaticamente) i consumi passivi, quali ad esempio i consumi notturni.
3. Monitorare il rifasamento dell'impianto, per verificare il corretto funzionamento delle apparecchiature e mantenere il fattore di potenza ($cos\varphi$) secondo i valori dettati dalla legge, al fine di evitare il pagamento di penali molto alte.

1.3. Sistemi classici di power monitoring

Da tempo, sono disponibili sul mercato diverse tipologie di dispositivi, molto limitati ed economici, per poter monitorare i consumi energetici di uno o più determinati elettrodomestici.

Questi dispositivi si possono acquistare facilmente e servono a misurare in una determinata frazione di tempo (tipicamente un secondo), i diversi parametri relativi al consumo istantaneo dell'elettrodomestico in questione.

Con questi semplici dispositivi, non è di certo possibile un monitoraggio stabile dei consumi in quanto non è previsto alcun tipo di data logging.

Figura 1: Due misuratori di consumi elettrici tradizionali da Amazon.

1.4. Struttura di base per un sistema IoT di power monitoring.

La progettazione di un dispositivo hardware e di un'architettura software, che consentano un accurato monitoraggio e predizione dei consumi energetici, si basa sullo studio approfondito delle tecniche e delle metodologie di misurazione dei parametri di base dell'impianto elettrico: tensioni e correnti alternate.

Successivamente, una volta ricavati questi due parametri, si potrà procedere al calcolo sia dei consumi effettivi che di altri parametri, come ad esempio il fattore di potenza.

I dati ottenuti, dovranno essere comunicati ad un server che a sua volta conterrà tutta la suite di visualizzazione e gestione del/dei dispositivi ad esso collegati.

A sua volta, questo sistema server dovrà fare uso di un sistema di database esterno per il salvataggio ed il successivo recupero dei dati.

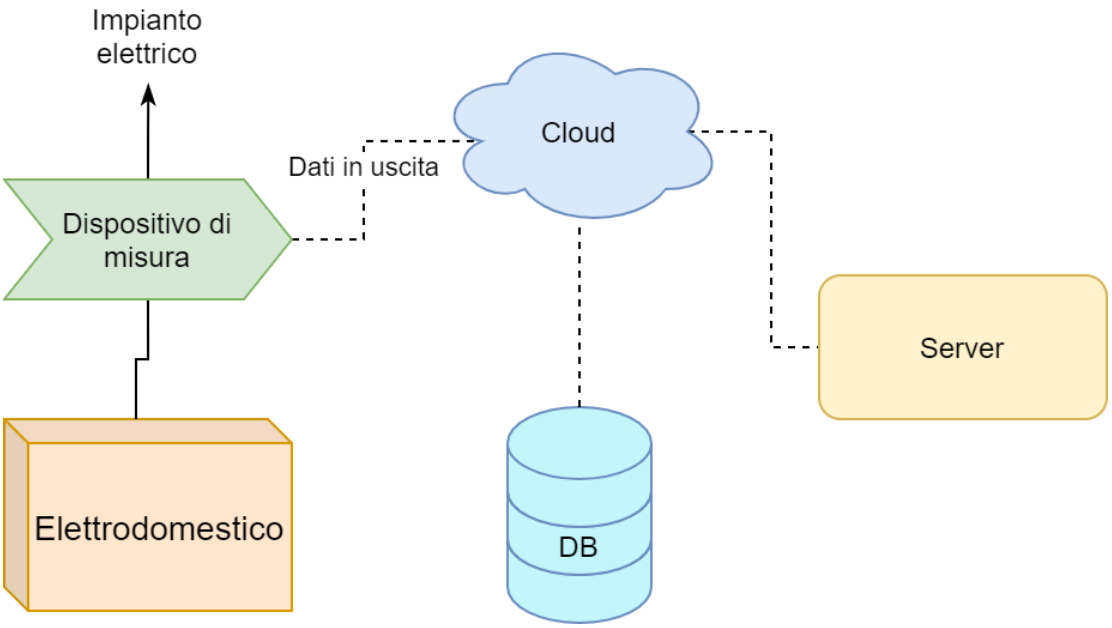

Figura 2: Schema a blocchi del progetto PowerMonitor.

La struttura generica precedentemente descritta, sarà la base per la progettazione e la realizzazione del progetto PowerMonitor trattato in questa Tesi.

2. Stato dell'Arte

Le nozioni presenti in questo capitolo, si occuperanno di argomentare in merito alle varie suite software già in commercio, considerabili anche come una fonte d'ispirazione per lo sviluppo delle varie funzionalità del progetto PowerMonitor.

I principali pacchetti software trattati saranno i seguenti:

1. Tasmota
2. Home Assistant ed ESPHome

2.1. Tasmota

Figura 3: Logo del custom firmware Tasmota.

Tasmota è uno dei più famosi custom firmware per dispositivi IoT basati sui chip ESP8285, ESP8266 e, più recentemente, basati anche su ESP32.

Il firmware veniva chiamato inizialmente Tasmota-Sonoff, perché spesso installato su dispositivi sonoff in commercio, basati su ESP8266 o ESP8285, per poterne ampliare le funzionalità.

Questo firmware, che è disponibile al download presso il repository del sito ufficiale, è adibito al contenimento di tutte le preimpostazioni pensate per i dispositivi in cui verrà installato; esistono, inoltre, varie versioni del firmware principale che si differenziano in base alle funzionalità rese disponibili all'utente finale.

Sul sito ufficiale sarà presente una lista completa delle varie versioni del firmware rilasciate.

Il firmware può essere installato scaricando il file binario e scrivendolo tramite il classico tool a riga di comando esptool.py, ma, per rendere il tutto più semplice, è reso disponibile un tool con interfaccia grafica chiamato Tasmotizer.

Figura 4: Il Tasmotizer.

Questo software permette lo scaricamento diretto dai repository OTA della versione scelta del firmware ufficiale, che l'installazione del firmware sul dispositivo collegato alla porta seriale specificata.

Dopo aver effettuato un backup del firmware originale (nel caso in cui il target per l'installazione sia un dispositivo commerciale, come precedentemente descritto), si potrà procedere al flash del firmware Tasmota tramite il pulsante blu `Tasmotize!` ed anche all'invio delle configurazioni del WiFi e di altri parametri tramite il pulsante viola `Send config`.

In seguito al riavvio, il dispositivo dovrebbe, in maniera automatica, connettersi alla rete WiFi specificata ma, in caso di esito negativo, il sistema, spostandosi in modalità di configurazione, consentirà la configurazione della rete WiFi attraverso l'apertura di un hotspot WiFi con captive portal.

Il firmware, una volta raggiunto tramite HTTP all'indirizzo del dispositivo ottenibile tramite il tasto verde `Get IP`, si presenta come segue:

Tramite `Impostazioni > Modulo`, è possibile scegliere la tipologia di modulo in commercio nel quale si è installato Tasmota; in questo modo, i pin GPIO del dispositivo saranno configurati in modo tale da utilizzare tutte le periferiche connesse fisicamente dal produttore del modulo stesso.

Sono disponibili molti moduli preconfigurati per poter sfruttare una moltitudine di moduli fisici diversi: ad esempio, il modulo selezionato in figura 5, rappresenta il modulo Sonoff Basic, molto spesso acquistato e successivamente aggiornato dall'utente finale, attraverso l'installazione del firmware Tasmota; questo modulo è riconoscibile tramite il primo tasto ON/OFF che serve per poter eseguire da web l'abilitazione e la disabilitazione dell'unico relè presente nel sonoff.

Figura 5: Home del portale principale di Tasmota.

In alternativa, è presente un modulo Generic, che consente la configurazione come input o output di ogni pin del microcontrollore; questo si rende utilizzabile quando, ad esempio, si installa Tasmota su un dispositivo DIY o fatto in casa.

Tramite l'interfaccia web di Tasmota, l'utente finale ha accesso alle varie altre funzionalità del dispositivo, come la connessione ad un broker MQTT remoto per il log di dati e per il controllo delle varie uscite, la configurazione di timer, alla cui scadenza può essere associato un determinato evento, la possibilità di poter usare una console web che rispecchia la console su porta seriale del dispositivo per l'utilizzo di vari comandi offerti dal firmware in modalità remota e la possibilità di aggiornare il firmware tramite OTA (Over The Air).

2.2. Home Assistant ed ESPHome

Figura 6: Logo di Home Assistant.

Home Assistant è un sistema operativo minimale atto alla gestione di dispositivi commerciali o DIY di IoT e di domotica.

Tramite questa suite di gestione dispositivi domestici, compatibile con un'infinità di dispositivi già in commercio, è possibile usufruire di funzionalità molto importanti, come MQTT, Timer, log di dati e molte altre.

Per poter provvedere alla gestione dei vari dispositivi e sensori interfacciati con la LAN, Home Assistant avrà bisogno di una macchina server stand alone su cui poter installare il proprio container o sistema operativo.

Nella sezione installazione del sito ufficiale, sono presenti varie modalità di installazione, ognuna con i suoi vantaggi ed i suoi svantaggi; la versione più popolare è quella rilasciata per i sistemi Raspberry Pi, i quali hanno un consumo energetico irrisorio ed una potenza computazionale più che adatta alla gestione di Home Assistant.

Una volta eseguita la procedura di installazione, tramite il portale di Home Assistant, raggiungibile in HTTP all'indirizzo della macchina fisica / virtuale alla porta 8123 si avrà accesso al wizard della prima configurazione e, successivamente, ad una moltitudine di variegate configurazioni:

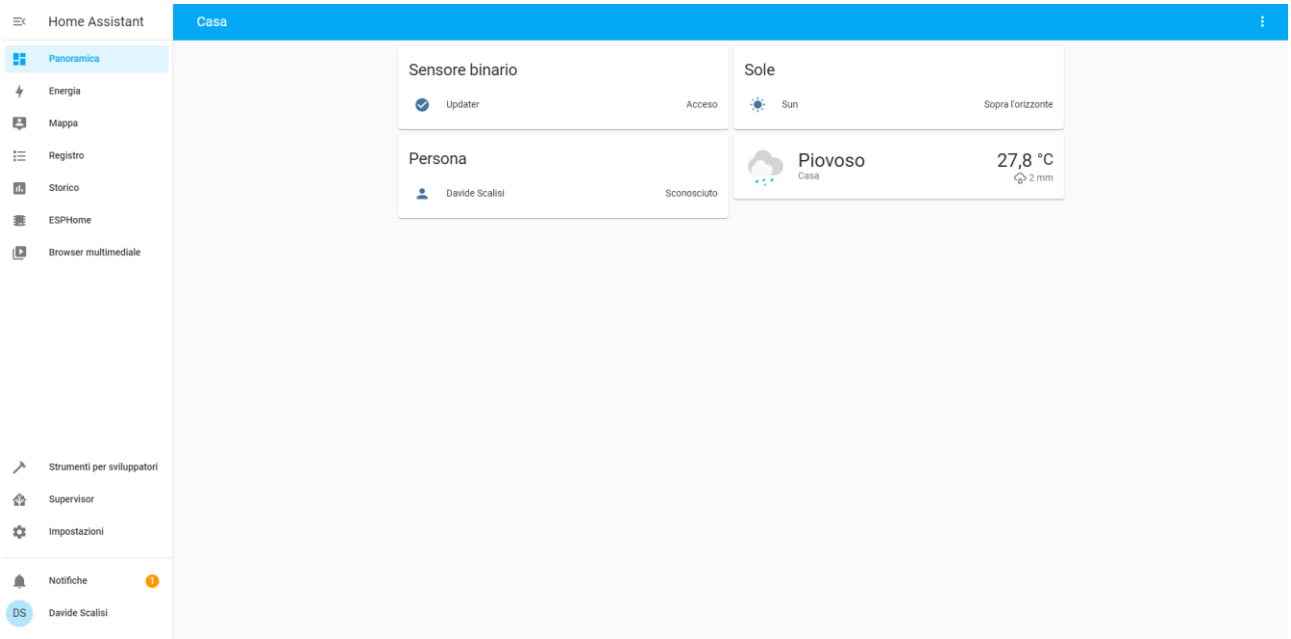

Figura 7: Dashboard web di Home Assistant.

Attraverso uno dei menù più interessanti, che si trova sotto

`Impostazioni > Integrazioni`

è prevista la ricerca in rete locale di tutti i dispositivi smart compatibili con l'ecosistema di Home Assistant e, tramite l'apposito tasto `Aggiungi Integrazione`, sarà possibile aggiungere un plugin molto importante ai fini dell'integrazione di dispositivi basati su ESP8266, ESP8285 ed ESP32, chiamato appunto ESPHome.

Figura 8: Logo del firmware ESPHome.

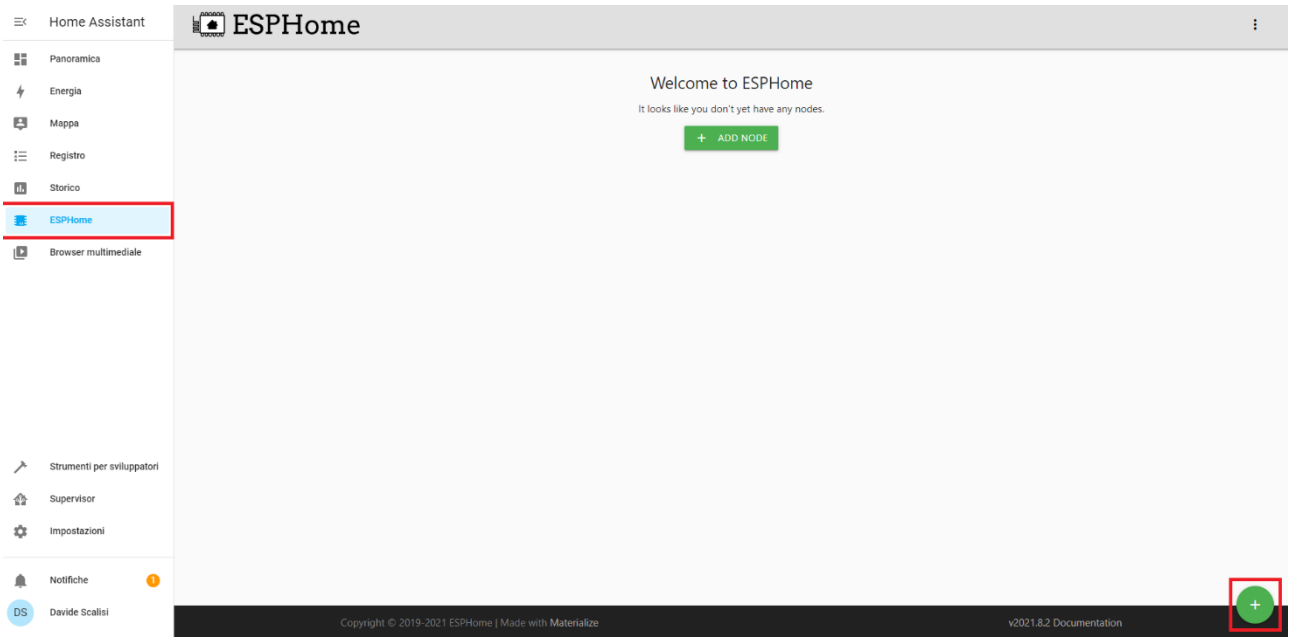

Figura 9: Portale di configurazione del plugin ESPHome all'interno del portale di Home Assistant.

Tramite l'utilizzo di questo plugin, sarà possibile creare un piccolo custom firmware, su misura e molto leggero, da installare sul dispositivo fisico che, a questo punto, diventerà pienamente compatibile con Home Assistant e tutte le sue funzionalità.

La scrittura del firmware ESPHome, non avviene attraverso codice scritto in un determinato linguaggio di programmazione, ma per mezzo della stesura di un semplice file di testo yaml, nel quale si andranno a specificare tutte le configurazioni relative ai pin di ingresso / uscita del dispositivo fisico.

Per poter aggiungere un nuovo dispositivo alla dashboard di ESPHome, basterà cliccare sul pulsante + o sul pulsante `Add Node`: questo porterà ad inserire il nome del

dispositivo da monitorare e le impostazioni della rete WiFi a cui il dispositivo stesso dovrà connettersi.

Inizialmente, dopo l'inserimento delle configurazioni prima trattate, nella dashboard di ESPHome, il dispositivo comparirà nello stato di `Offline`; successivamente alla stesura, alla compilazione ed all'installazione del firmware sul dispositivo ESP fisico, questo stato verrà commutato in `Online`.

Per la scrittura del firmware e per la sintassi del file yaml, editabile dalla dashboard tramite il pulsante `Edit` presente sul singolo dispositivo, si rimanda il lettore alla documentazione ufficiale.

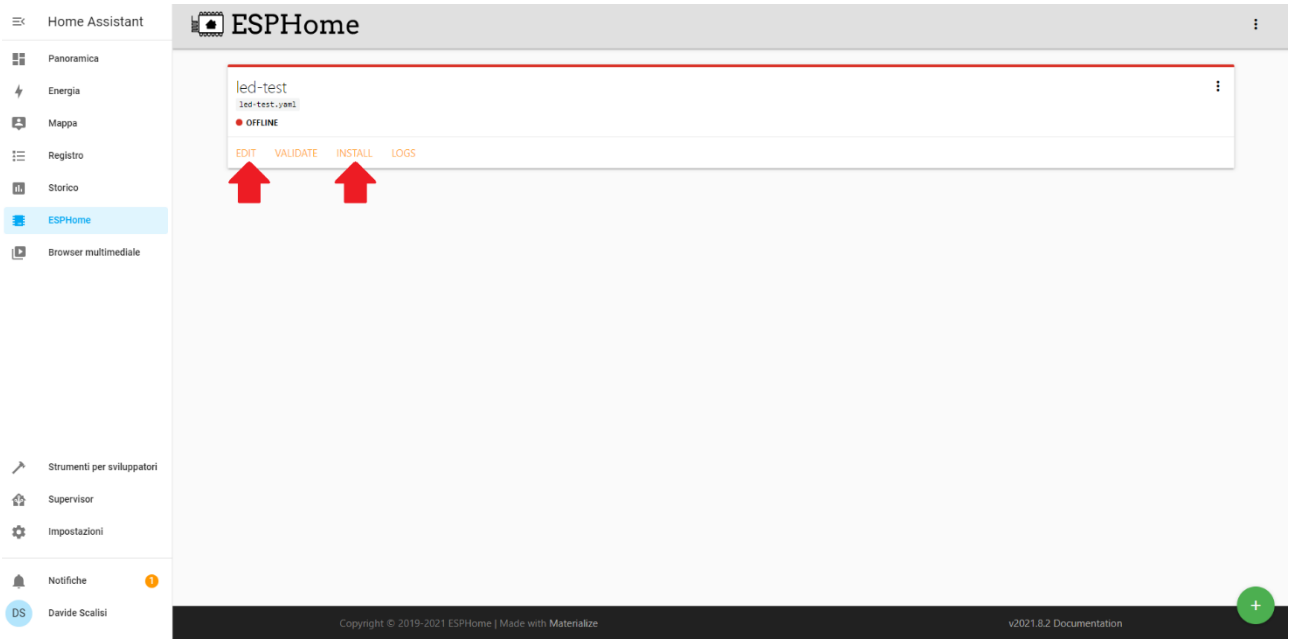

Figura 10: Un nuovo dispositivo di nome led-test aggiunto alla dashboard di ESPHome.

Una volta scritto il firmware sotto forma di file yaml, basterà cliccare il pulsante `Install` per poter scaricare, solo per la prima installazione, il file binario del firmware compilato risultante dal file yaml.

Questo file binario sarà, come per Tasmota, installabile tramite il classico tool a riga di comando esptool.py, ma, sempre per semplicità d'utilizzo, è stato realizzato il software ESPHome-Flasher, che si preoccupa di offrire all'utente finale una comoda interfaccia grafica per il flash del firmware prima creato.

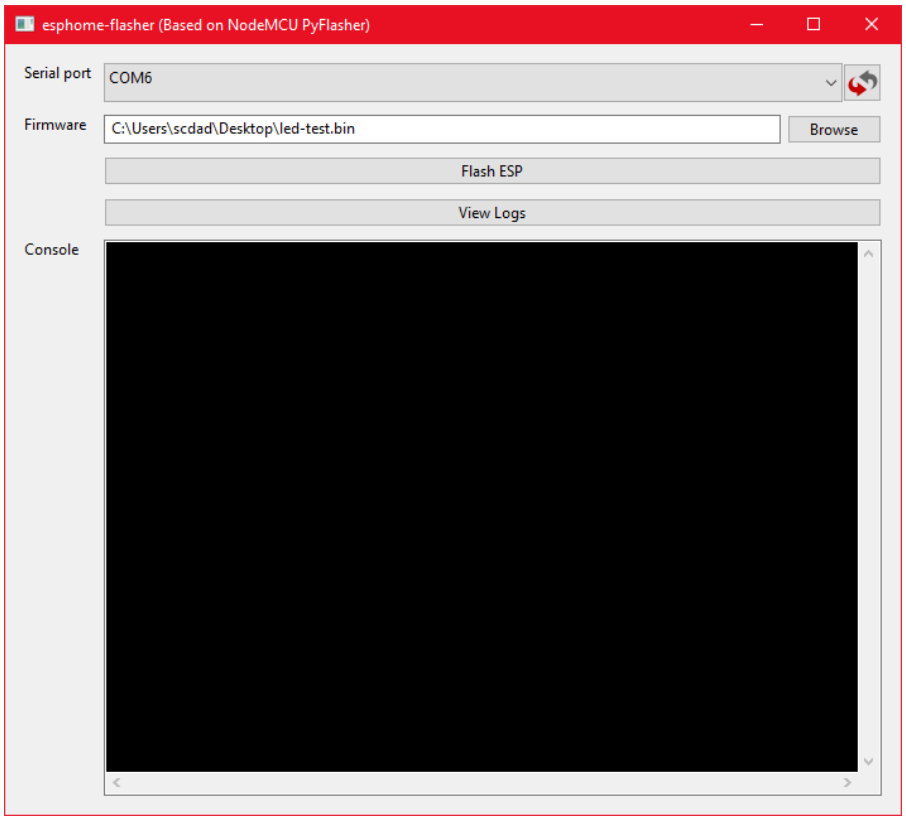

Figura 11: Software ESPHome-Flasher.

Analogamente al software Tasmotizer descritto prima, tramite la selezione della porta seriale corrispondente al dispositivo fisico ed alla selezione del file binario del firmware prima scaricato, è possibile eseguire la prima programmazione del microcontrollore tramite il pulsante `Flash ESP`.

Una volta installato il firmware, basterà riavviare il dispositivo per poi controllare se, nella dashboard di ESPHome, risulta `Online`; successivamente, basterà tornare nella sezione `Impostazioni > Integrazioni` per poter constatare che il dispositivo appena programmato è stato rilevato da Home Assistant.

A questo punto, si potrà procedere con l'aggiunta del dispositivo alla dashboard di Home Assistant e, da questo momento in poi, tutti gli ingressi e le uscite configurati all'interno del file yaml, saranno visibili e controllabili direttamente dalla dashboard di Home Assistant.

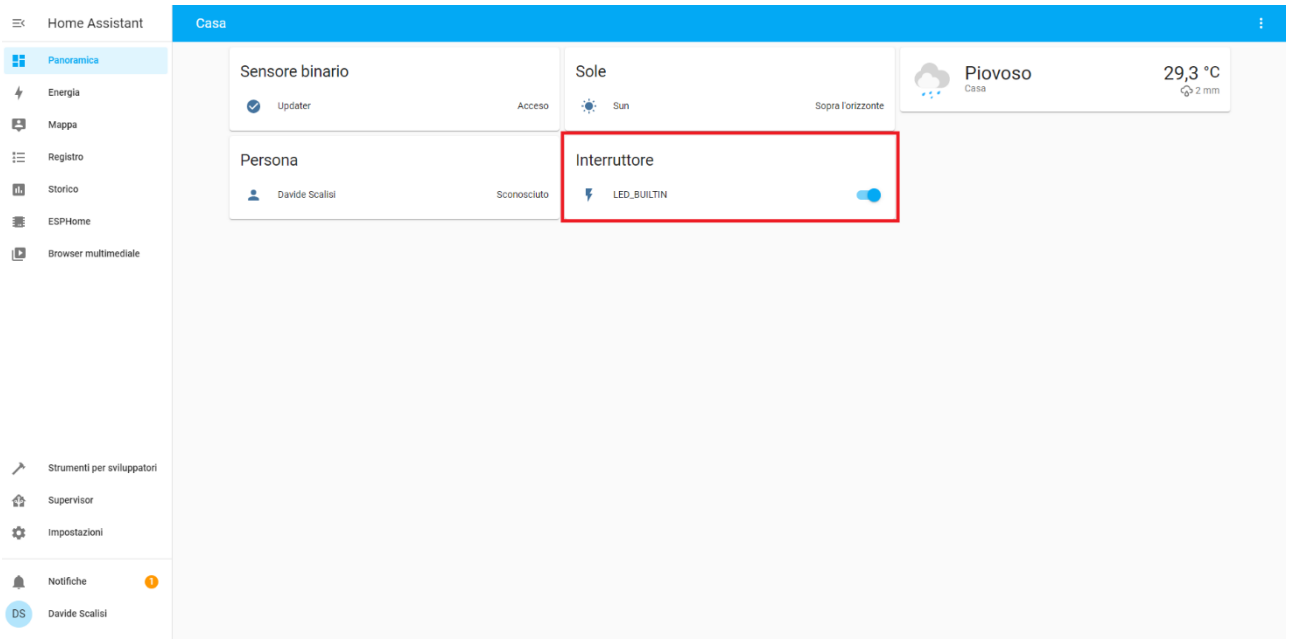

Figura 12: Dispositivo test-esp appena aggiunto: tramite la confugurazione del file yaml, è possibile controllare, a titolo d'esempio, il LED integrato di una ESP8266-NodeMCU direttamente dalla dashboard di Home Assistant.

2.3. Paragone tra Tasmota e ESPHome

La scelta del corretto firmware, da utilizzare per il proprio sistema DIY di IoT domestico, dipende dalle esigenze soggettive dell'utente finale; tuttavia, di seguito, è fornita una tabella di comparazione generica tra i due firmware per agevolare la scelta:

Tasmota	ESPHome
Pesante.	Leggero.
Monolitico.	Estremamente modulare e personalizzabile.
Firmware standalone.	Firmware appartenente della suite di Home Assistant tramite l'omonimo plugin della stessa.
Non necessita di un nodo server centrale.	Necessita di un nodo server centrale in cui installare una della versioni di Home Assistant.
Modificabile senza il bisogno di ricompilare e ricaricare il firmware.	Per ogni tipo di modifica software, è richiesta una riprogrammazione (che dopo la prima, può anche essere eseguita via OTA).

3. Tecniche di misurazione dei parametri elettrici

3.1. Parametri da calcolare

In un sistema di power logging generico, i parametri ricavati dalle misurazioni sono principalmente due:

1. Valore RMS della tensione
2. Valore RMS della corrente

Da questi due valori ottenuti, seguiranno una serie di calcoli atti a ricavare nuovi parametri come:

1. Potenza
    a. Attiva
    b. Reattiva
    c. Apparente
2. Fattore di potenza (o $\cos\varphi$)

Questi ultimi parametri sono molto importanti in quanto si lavora su un sistema a regime alternato.

3.2. Digitalizzazione dei segnali

Quando si devono digitalizzare ed acquisire delle grandezze fisiche sotto forma di differenze di potenziale, o, più genericamente, dei segnali, occorre ricorrere alla tecnica della digitalizzazione.

Un segnale si dice analogico quando è a tempo continuo e valori continui; viceversa, si dice digitale se è a tempo discreto e valori discreti.

Un segnale digitalizzato è una rappresentazione numerica che, di volta in volta, esprime l'ampiezza del segnale analogico in istanti successivi di tempo; è possibile ottenere questi numeri, o campioni, tramite dei dispositivi hardware fisici chiamati ADC (Analog Digital Converter).

Un ADC riceve in input un segnale continuo a tempo continuo e restituisce in uscita un segnale discreto a tempo discreto, in grado di essere elaborato da un qualsiasi calcolatore digitale.

Per poter eseguire questa conversione, è necessario:

1. Trasformare un segnale a tempo continuo in uno a tempo discreto.
2. Trasformare un segnale a valori continui in uno a valori discreti.

Il primo processo prende il nome di campionamento, mentre il secondo si definisce quantizzazione.

3.2.1. Cenni al campionamento

La trasformazione di un segnale a tempo continuo in un segnale a tempo discreto prende il nome di campionamento: operazione che si occupa di discretizzare l'asse dei tempi.

Tramite la tecnica del campionamento, si considerano solo alcuni valori del segnale tra loro equidistanti e con questi si descrive l'intero segnale.

La distanza T tra un campione e l'altro è definita come periodo di campionamento o come l'inverso della tasso di campionamento (sampling rate), misurato in Hertz.

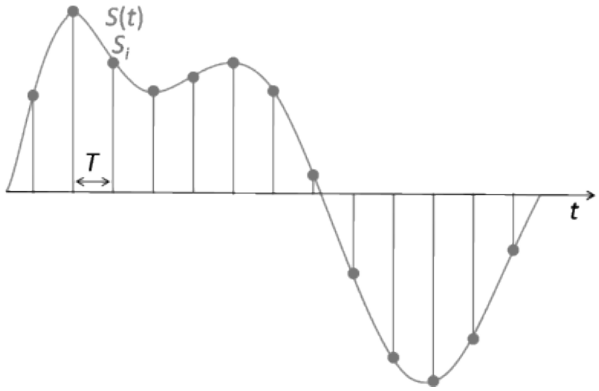

Figura 13: Campionamento di un segnale S(t); T rappresenta il periodo di campionamento.

3.2.2. Cenni alla quantizzazione

Se al campionamento corrisponde la discretizzazione dell'asse dei tempi, alla quantizzazione è associata la discretizzazione dell'asse dei valori.

Ad ogni valore continuo del segnale analogico, sarà associato un nuovo valore in un insieme discreto di livelli. Questo introdurrà un certo errore poiché valori originariamente differenti possono collassare nello stesso livello, divenendo così indistinguibili; più livelli disponibili indicano una maggior precisione nella quantizzazione.

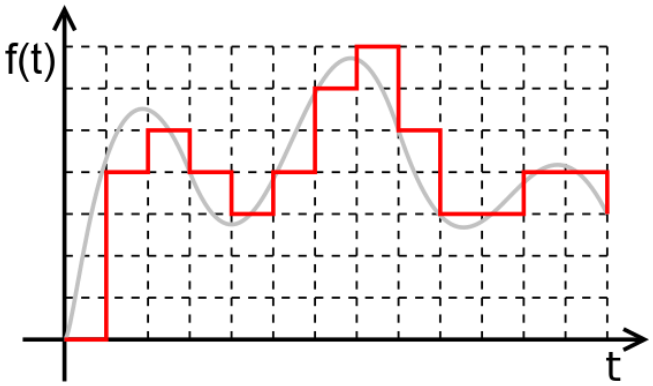

Figura 14: Quantizzazione di un segnale f(t): ad ogni valore che ricade in un determinato quanto, è associato lo stesso valore.

3.2.3. Eseguire un buon campionamento

La scelta di un valore di T adeguato (figura 13) è strettamente correlata alla frequenza di Nyquist f_n, cioè alla massima frequenza che il segnale da campionare assume.

La frequenza di Nyquist può anche essere definita come la più alta frequenza presente nello spettro del segnale.

3.2.4. Teorema del campionamento di Nyquist-Shannon

Il teorema del campionamento di Nyquist-Shannon enuncia quanto segue: per poter ricostruire fedelmente un segnale campionato, è necessario che il tasso di campionamento sia almeno il doppio della frequenza di Nyquist.

$$f_c > 2f_n$$

Notare bene che la disequazione non comprende l'uguaglianza in quanto, se il tasso di campionamento scelto f_c dovesse essere esattamente pari a $2f_n$, si andrà incontro alla problematica del campionamento critico, cosa che porterà successivamente ad ottenere un segnale ricostruito completamente differente dal segnale originale.

Tipicamente, per poter eseguire un campionamento adeguato, si sceglie un tasso di campionamento sufficientemente maggiore della frequenza di Nyquist.

3.3. Acquisizione e mappatura dei campioni di tensione e corrente

Per poter acquisire i campioni, che rappresentano le due grandezze fisiche interessate, bisogna partire dal presupposto che la massima frequenza della tensione di rete è pari a 50Hz; per il teorema di Nyquist sopra enunciato, sarà sufficiente prendere

$$f_c \gg 100Hz.$$

Per sicurezza, il valore scelto per il progetto PowerMonitor è di 10kHz.

Per quanto riguarda la quantizzazione dei valori acquisiti, l'ADC assegnerà dei valori corrispondenti ad ogni campione, in un intervallo discreto di valori compreso tra $[0,\ 2^r]$, con r pari alla risoluzione del dispositivo.

Il parametro r (risoluzione), come il tasso di campionamento massimo, sono parametri intrinsechi dell'ADC fisico che si prende in considerazione; un aumento di tale parametro, comporta un aumento dei valori numerici discreti assegnabili alle diverse ampiezze del segnale analogico.

Ancora una volta, per il progetto PowerMonitor è stato scelto un ADC con una risoluzione pari a 12 bit.

Una volta acquisiti i campioni, questi dovranno essere riportati in un range di valori da $[\sim -315,\ \sim 315]$ per i campioni del segnale della tensione e da $[Imin,\ Imax]$ per i campioni del segnale della corrente, con $Imin$ ed $Imax$ pari rispettivamente ai

valori massimi e minimi di corrente attesi tra l'impianto elettrico ed il carico da monitorare.

Con il termine mappatura, si indica la conversione dei valori dai massimi e minimi generati dall'ADC ad i massimi e minimi da noi stabiliti.

Una descrizione approfondita di questi valori verrà trattata nei prossimi capitoli.

3.4. Potenza attiva e reattiva

N.B.: Tutte le grandezze da questo punto in poi sono da considerarsi come grandezze istantanee.

In fisica, la potenza è definita come l'energia emessa da un sistema nell'unità di tempo.

Nel sistema internazionale, l'unità di misura la potenza si misura in watt W, come rapporto tra unità di energia in joule J e unità di tempo in secondi s: $\frac{J}{s}$.

Se si lavora con tensioni e correnti continue, la potenza emessa da un generico bipolo può essere definita come il prodotto della tensione ai suoi capi per la corrente che lo attraversa $P = V * I$.

Diventa invece più complicato calcolare la potenza emessa utilizzata (quella potenza che compie effettivamente un lavoro nel corso del tempo), quando si lavora con tensioni e correnti in un sistema a regime alternato.

Se in un circuito chiuso a regime alternato si verificano entrambe le seguenti condizioni:

1. Applicazione di una certa tensione alternata (supponendo che quest'ultima sia, per semplicità, sempre sinusoidale), che implica lo scorrimento di una certa corrente alternata.
2. Presenza di bipoli atti a conservare e rilasciare energia, come induttanze o capacità (anche detti componenti a memoria).

Questo causerà un'oscillazione di corrente tra il generatore ed il carico, definita come corrente reattiva.

Un metodo per tracciare la generazione di corrente reattiva consiste nell'osservare se la tensione ai capi del carico è in fase con la corrente che lo attraversa; se così non fosse, nel carico saranno sicuramente presenti anche componenti induttive e/o capacitive.

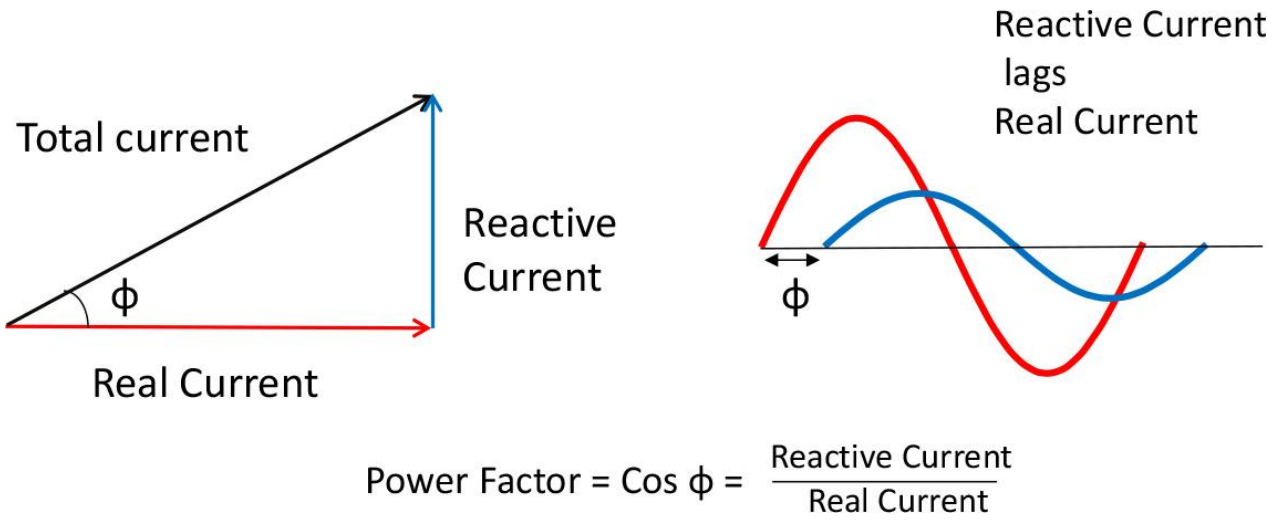

Figura 15: A sinistra, il triangolo delle correnti, a destra, lo sfasamento tra tensione e corrente generato da un carico reattivo.

Questa corrente reattiva, a differenza della normale corrente attiva, non svolge alcun lavoro, ma sarà un semplice movimento oscillatorio di elettroni dal generatore al carico.

Il prodotto tensione e corrente attiva, origina un parametro che prende il nome di potenza attiva, misurata in W, che rappresenta l'analogo della potenza dissipata dal carico; allo stesso modo, il prodotto tensione e corrente reattiva, origina un parametro chiamato potenza reattiva, misurata in VAr (Volt Ampere reattivi).

Diverse normative regolamentano e limitano l'immissione da parte di un utente privato, di un determinato quantitativo di corrente reattiva; questa, infatti, essendo comunque un flusso di elettroni che attraversa un conduttore, oscillando tra generatore (dalle linee elettriche urbane fino a dove l'energia è prodotta) e carico (dispositivo che genera potenza reattiva), arriva a produrre, tramite effetto Joule, delle perdite sotto forma di calore nelle linee elettriche.

Questo succede perché le linee elettriche non sono costruite con materiale superconduttore e la presenza di una resistenza serie interna, anche se molto piccola, provocherà sulle lunghe distanze ed al passaggio di una grossa corrente, una dissipazione di potenza sotto forma di un surriscaldamento inutile dell'intero percorso che la corrente attraversa, comportando uno spreco che non dovrebbe aver luogo in virtù delle normative sopra citate.

La seconda legge di Ohm $R = \rho \frac{l}{A}$, con ρ costante pari alla resistività del materiale e con A ed l pari rispettivamente all'area della sezione ed alla lunghezza del conduttore, afferma che per poter diminuire la resistenza le uniche opzioni valide nel caso pratico sono quella di aumentare l'area del conduttore o quella di cambiare il materiale di cui è costruito; entrambe le opzioni farebbero alzare i costi di produzione di molto.

Il calcolo è equivalente se lo si vuole applicare, ad esempio, all'impianto elettrico di un'abitazione: i cavi conduttori devono per forza essere sovradimensionati per poter

essere sicuri di reggere un massimo di corrente attiva e reattiva senza andare in fiamme sotto la potenza dissipata.

Il monitoraggio della potenza reattiva emessa da un dispositivo è importante in quanto, anche se non citata sulle bollette dei privati, l'emissione di livelli eccessivi potrebbe, oltre che esporre a salate sanzioni, rivelarsi nel peggiore dei casi potenzialmente pericoloso per un potenziale rischio incendio.

3.5 Potenza apparente e fattore di potenza

La potenza apparente in un circuito a corrente alternata è definita con la stessa legge della potenza in un circuito a corrente continua: infatti essa è pari alla tensione istantanea di rete per la corrente istantanea che transita tra l'impianto ed il carico e si misura in VA (Volt Ampere).

Questa potenza apparente, a differenza della potenza classica (che si ricorda, è misurata in W), è pari alla combinazione della potenza attiva consumata e reattiva generata da un carico.

Indicate le tre potenze con le seguenti nomenclature:

1. Potenza apparente: S
2. Potenza attiva: P
3. Potenza reattiva: Q

La legge che lega queste tre grandezze sarà la seguente

$$S = \sqrt{P^2 + Q^2}$$

Tramite questa legge, che deriva dal teorema di Pitagora, si può costruire un triangolo che prende il nome di triangolo delle potenze siffatto:

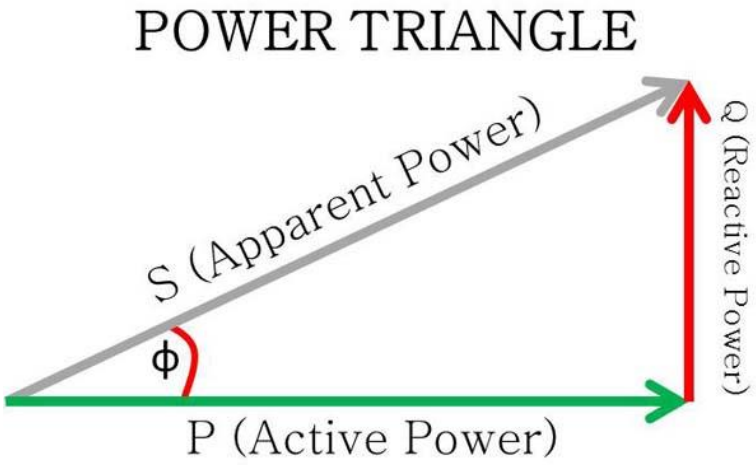

Figura 15: Triangolo delle potenze.

In questo triangolo, più il cateto Q sarà minimo, più la potenza reattiva generata dal carico sarà bassa e più l'angolo φ sarà acuto: questo angolo prende il nome di angolo

di sfasamento, in quanto, come precedentemente enunciato, una delle condizioni per la generazione di potenza reattiva, è quella di avere uno sfasamento di un massimo di 90° tra i segnali della tensione alternata ai capi del carico e della corrente alternata che lo attraversa.

In particolare, il coseno di questo angolo φ, indicato come $\cos\varphi$, è un parametro molto importante compreso nell'intervallo $[0, 1]$, in quanto, se questo parametro è pari ad 1, la potenza emessa dal carico sarà puramente attiva e lo sfasamento tra tensione e corrente sarà nullo (caso ottimale); al contrario, se $cos\varphi$ vale 0, la potenza emessa dal carico sarà puramente reattiva e lo sfasamento tra tensione e corrente sarà pari 90° (caso pessimo).

In aggiunta, il teorema sui triangoli rettangoli garantisce la validità della relazione $cos\varphi = \frac{P}{S}$.

3.6. Valor medio e valore efficace o RMS

Quando si lavora con dei segnali, è importante conoscere quanta energia il segnale stesso riesce a trasportare.

Questa informazione è ricavabile da due parametri molto importanti:

1. Valor medio
2. Valore efficace o RMS (Root Mean Square)

Entrambi gli estimatori indicano un valore corrispondente di tensione/corrente continua, trasportata effettivamente dal segnale in tensione/corrente alternata.

Es.: La tensione di rete ha un valore di picco di 315V, ma il suo valore efficace è di 220Vrms; questo sta a significare che nel tempo questo segnale alternato, se applicato ad un carico resistivo costante, può fornire la stessa energia di un segnale continuo con tensione pari a 220V.

Il valor medio, come suggerito dal nome stesso, è ricavato dalla media di tutti i campioni ottenuti dalla digitalizzazione del segnale interessato.

Questo estimatore ha però un problema: se il segnale ha un'area pari ad $A = \int_{t_0}^{t_1} f(t)dt$ e, prendendo in considerazione il segnale della tensione $f(t) = V_p * \sin(t)$, con V_p pari al picco massimo della tensione di rete, quest'area risulterà nulla, in quanto $f(t)$ presenterà un'area sottesa positiva pari a quella negativa; in queste circostanze, l'estimatore del valor medio diventerà poco utile e sarà necessario utilizzare l'RMS.

L'RMS (acronimo definito dalle operazioni che lo compongono lette al contrario), è un estimatore che non soffre del problema sopra descritto ed è calcolato come segue:

1. Se il segnale ha legge sinusoidale:

a. Come nel caso della tensione di rete, il calcolo è banale: $V_{RMS} = V_p * \sqrt{2}$

2. Se il segnale non ha legge sinusoidale:
   a. Si dovranno elevare tutti i campioni al quadrato.
   b. Si dovrà eseguire la media di questi quadrati.
   c. Si dovrà calcolare la radice quadrata di questa media.

3.7. Calcolo delle potenze a partire dai campioni di tensione e corrente

Nel dispositivo di misurazione, come scritto capitolo 3.1., gli unici due parametri acquisiti dall'ADC dovranno essere solamente i segnali di tensione e corrente; da questi due segnali sarà poi possibile ricavare i quattro parametri precedentemente citati (le tre potenze ed il $cos\varphi$).

Questo è possibile grazie alle seguenti formule:

$$S(VA) = V_{RMS} * I_{RMS}$$

Tramite questa formula, si otterrà la potenza apparente; questo accade in quanto nel calcolo dell'RMS, come prima descritto, vengono utilizzati dei quadrati e tramite questi ultimi, si riesce ad ignorare lo sfasamento presente tra tensione e corrente, ottenendo così la potenza apparente.

$$P(W) = \frac{1}{n}\sum_{i=1}^{n}(V_i * I_i)$$

Questa formula permette invece, tramite i campioni precedentemente acquisiti, di calcolare la potenza attiva dissipata effettivamente dal carico tramite la somma dei prodotti di tensione e corrente istantanea misurati.

Una volta ottenuto questo valore, basterà dividerlo per il numero di campioni acquisiti, facendo così una media.

Tramite questa formula basata su una semplice media, non si tiene in considerazione lo sfasamento tra tensione e corrente, ricavando così la potenza attiva.

$$Q(VAr) = \sqrt{S^2 - P^2}$$

$$cos\varphi = \frac{P}{S}$$

Basandosi sulla correlazione descritta in figura 16 tra questi quattro parametri, allora questi ultimi due parametri verranno banalmente ricavati tramite l'utilizzo del teorema di Pitagora e del teorema sui triangoli rettangoli.

4. PowerMonitor

4.1. Descrizione del progetto

Il progetto PowerMonitor nasce come un sistema open-source domestico che può essere installato e monitorato in maniera abbastanza semplice; l'installazione verrà trattata nel capitolo successivo.

Facendo sempre riferimento allo schema a blocchi generico illustrato in figura 2, questo capitolo tratterà una descrizione dettagliata delle tecnologie associate per la realizzazione di questo progetto.

4.2. Circuito elettrico

In questa sezione, verrà trattato l'intero schema elettrico progettato per il dispositivo di misurazione in questione.

4.2.1 Circuito analogico di acquisizione

Una volta chiara la teoria, la progettazione del circuito elettrico di acquisizione segue di conseguenza.

Gli elementi utilizzati per l'acquisizione dei due segnali di tensione e corrente sono un piccolo trasformatore abbassatore per la tensione ed una pinza amperometrica per la corrente.

Figura 16: A sinistra, un piccolo trasformatore abbassatore; a destra, una pinza amperometrica: l'SCT013-000.

Questi due sensori sono in grado di fornire due segnali in tensione corrispondenti rispettivamente alla tensione di rete ed alla corrente tra la rete elettrica ed il carico.

Per questioni di sicurezza, è importante che i due sensori restituiscano un segnale elettricamente isolato dalla rete elettrica; infatti, se l'operatore intento a testare il circuito dovesse entrare in contatto con lo stesso, potrebbe rimanere folgorato.

I due segnali, generati tramite un accoppiamento elettromagnetico, sono sicuri, in quanto non correlati elettricamente in alcun modo con la fase o con il neutro dell'impianto elettrico.

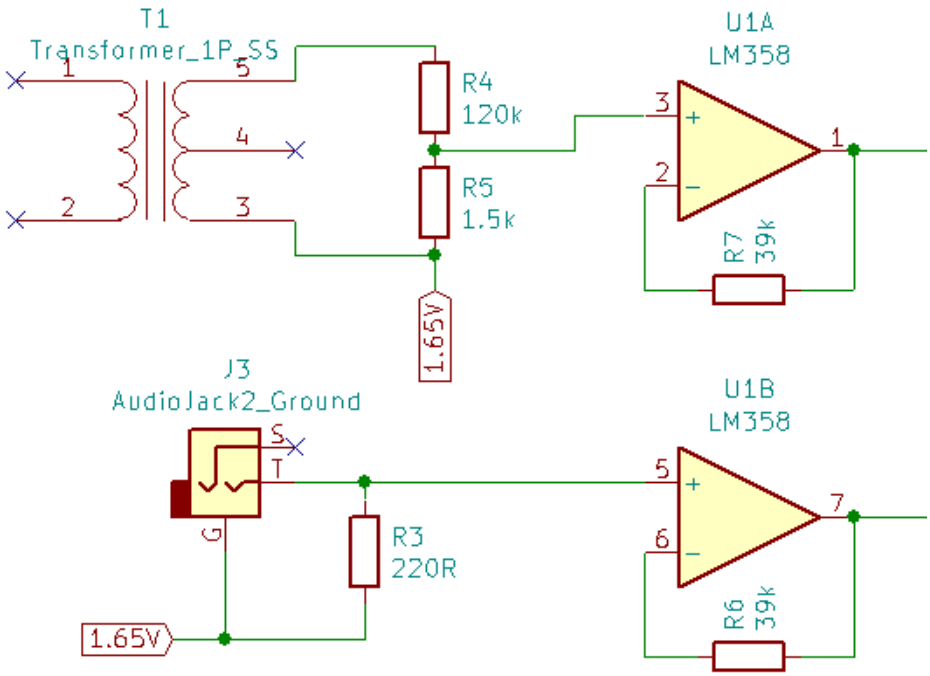

Figura 17: Circuito analogico di acquisizione dei due segnali di tensione e corrente.

In figura 18, è rappresentato il circuito analogico per l'acquisizione dei due segnali; il sistema a microcontrollore scelto in questo progetto sarà una NodeMCU, una piccola scheda basata sul microcontrollore ESP8266 operante ad una tensione di $V_{cc} = 3.3V$: questa informazione è molto importante in quanto fissa il limite superiore sopra il quale i due segnali in ingresso non dovranno mai salire, pena, il danneggiamento del microcontrollore stesso.

Allo stesso modo, i segnali non dovranno in alcun modo scendere sotto la tensione di soglia di $gnd = 0V$ e diventare più negativi rispetto alla massa: anche in questo caso si andrebbe a danneggiare irreparabilmente il microcontrollore.

Lo stadio di acquisizione di entrambi i segnali ha infatti uno dei due poli riferito ad una massa virtuale di valore pari a $V_{ref} = \frac{V_{cc}}{2} = 1.65V$ per poter mantenere il segnale nell'intervallo $[gnd; V_{cc}]$, a patto che il valore picco-picco del segnale stesso V_{pp} resti minore o uguale di V_{cc}.

$$V_{pp} \leq V_{cc}$$

4.2.2. Stadio di acquisizione della tensione

Nello stadio di acquisizione della tensione, è presente un trasformatore collegato ad un partitore resistivo, che serve a riportare il segnale tra $0V$ e $3{,}3V$.

Il valore resistenze R_4 ed R_5 del partitore, dipende dalle caratteristiche intrinseche del trasformatore scelto.

In questo progetto, il trasformatore scelto ha un ratio o fattore di trasformazione pari a $r = 0.144$.

Questo dato è pari a $r = \frac{N_1}{N_2} = \frac{V_1}{V_2}$, dove N_1 ed N_2 indicano rispettivamente il numero di spire al primario ed al secondario del dispositivo e V_2 e V_1 indicano rispettivamente la tensione alternata a vuoto localizzatasi al secondario a seguito di una tensione alternata applicata al primario.

La costante r, una volta ottenuta e moltiplicata per il valore di picco della tensione in ingresso $V_{p_{in}}$ (cioè della tensione di rete), restituirà come risultato la tensione di picco della tensione di uscita $V_{p_{out}}$, cioè il massimo valore che potrà assumere.

$$V_{p_{out}} = r * V_{p_{in}}$$

È buona norma considerare il valore di picco della tensione di rete quando questa ha il suo valore efficace massimo: nel caso di tensione efficace a 230V, l'intervallo consentito in Italia è compreso tra 207V e 253V.

Per calcolare il valore di picco partendo dal valore efficace di una sinusoide, basta moltiplicare questo valore per $\sqrt{2}$.

Questo valore di picco $V_{p_{out}}$, come detto prima, non deve assolutamente superare la tensione di soglia di V_{cc}, quindi, tramite un partitore resistivo, si riporta il valore di $V_{p_{out}}$ ad un valore massimo di V_{cc}.

$$gnd \leq V_{p_{mapped}} = V_{p_{out}} * \frac{R_5}{(R_4 + R_5)} \leq V_{cc}$$

Applicando questa legge è possibile, tramite la corretta scelta di R_4 ed R_5, ottenere una tensione $V_{p_{mapped}}$ di picco che rientri nel range massimo di $[gnd; V_{cc}]$.

4.2.3. Stadio di acquisizione della corrente

Nello stadio di acquisizione della corrente, si è fatto uso del sensore SCT013-000: una pinza amperometrica che espone in uscita i due terminali di una spira avvolta su un nucleo ferromagnetico interno.

Si può considerare questo sensore come un trasformatore formato da un primario con una sola spira (uno dei due conduttori di corrente tra la rete elettrica ed il carico) ed un secondario interno.

Per la legge di Faraday-Lenz-Neumann, quindi, verrà indotta nel secondario una corrente proporzionale alla corrente che scorre al primario; in questo modo, tramite l'utilizzo della legge di Ohm, basterà chiudere l'anello su una resistenza R_3 di misura per poter convertire questa corrente indotta in una tensione indotta.

Per gli stessi motivi citati nella sezione precedente, la resistenza di misura R_3 deve essere scelta in modo tale che la tensione massima di picco $V_{p_3} = i_p * R_3$, con i_p pari alla corrente di picco indotta nel secondario, non superi V_{cc} e non scenda sotto gnd.

Il sensore ha un rapporto di conversione $100A = 50mA$, quindi, per poter calcolare R_3, bisogna prima fissare una potenza apparente massima al primario S_{max}: per il progetto in questione si è utilizzato un valore di $S_{max} = 2200VA$.

Una volta fissato S_{max}, si procede a ricavare la corrente massima al primario I_{max}

$$I_{max} = \frac{S_{max}}{V_{max}}$$

con V_{max} pari alla tensione efficace di rete, cioè $220V$.

Assumendo che la forma d'onda di corrente al primario sia una sinusoide, ipotesi non sempre verificabile, si calcola il valore di picco approssimativo che avrà il segnale di corrente:

$$I_{p_{max}} = I_{max} * \sqrt{2}$$

A questo punto, basterà calcolare la proporzione tra le due correnti ed il rapporto di trasformazione del sensore citato prima:

$$\frac{100A}{50mA} = \frac{I_{p_{max}}}{i_{p_{max}}} \quad \rightarrow \quad i_{p_{max}} = \frac{I_{p_{max}}}{2000}$$

Infine, il valore di R_3 sarà dato dalla legge di Ohm come segue:

$$R_3 = \frac{V_{cc}}{i_{p_{max}}}$$

È buona norma scegliere una resistenza leggermente inferiore in quanto così, in caso di picchi di assorbimento di corrente al primario, il valore rimanga contenuto nell'intervallo $[gnd; V_{cc}]$.

Come per il circuito di acquisizione della tensione di rete, uno dei due poli del sensore di corrente sarà collegato alla stessa massa virtuale V_{ref} per poter permettere al segnale di variare attorno al suo livello.

4.2.4. Stadio buffer con amplificatori operazionali

Per evitare che durante il funzionamento picchi dannosi di tensione finiscano direttamente a contatto con l'ADC, rischiandone il danneggiamento, sono stati previsti nel circuito due OPAmp (amplificatori operazionali) U_1 ed U_2 in configurazione di buffer non invertente, atti a ripetere il segnale dato loro in ingresso.

La configurazione di buffer non invertente è facilmente riconoscibile in quanto il segnale è direttamente iniettato nel pin non invertente del componente ed è presente solamente una resistenza di retroazione dall'uscita al pin invertente.

Questo garantisce due principali vantaggi:

1. Gli amplificatori operazionali sono alimentati con una tensione pari a V_{cc}, quindi serviranno anche da soppressori di picchi sia positivi che negativi.
2. Il segnale in uscita dagli operazionali è ad impedenza molto minore rispetto al segnale diretto in uscita dai sensori, cosa che è sempre meglio avere in quanto nella fase di acquisizione del campione è richiesta la carica di una piccola capacità di acquisizione interna all'ADC.

4.2.5. Circuito digitale di campionamento

Una volta che i segnali sono passati oltre lo stadio di buffer, essi verranno acquisiti da un ADC per poter essere digitalizzati.

Per questo progetto, si è scelto di utilizzare un ADC ad approssimazioni successive: l'MCP3202 della MICROCHIP.

Questo ADC, che possiede un doppio canale con 12bit di risoluzione, si interfaccia tramite il protocollo SPI ad un qualsiasi sistema esterno compatibile.

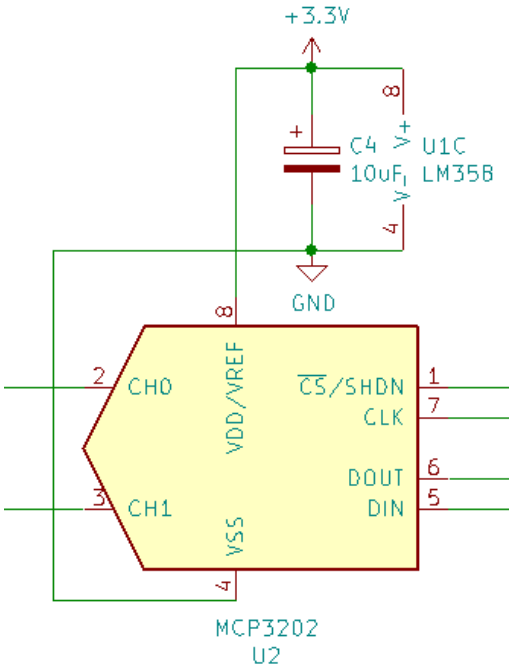

Figura 18: Circuito digitale di acquisizione dei due segnali di tensione e corrente.

In questo progetto, i due canali saranno utilizzati in modalità single-ended o separata, tuttavia, questo tipo di ADC permette anche l'utilizzo di una modalità pseudo-differenziale sotto differente configurazione dei due canali.

4.2.6. Acquisizione dei dati

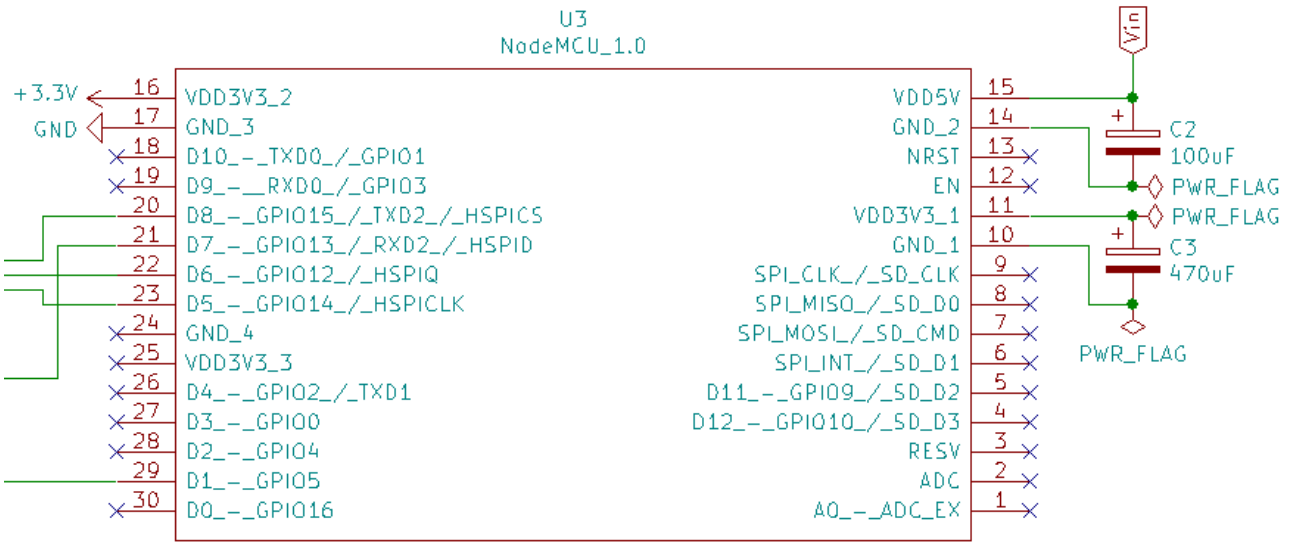

Figura 19: Schema di interfaccia tra l'ADC ed il NodeMCU.

Per poter acquisire i dati che l'ADC invia tramite il bus SPI, si è deciso di utilizzare, come menzionato precedentemente, un microcontrollore della famiglia ESP8266: l'ESP-12F in formato NodeMCU 1.0.

Un ESP8266 possiede le seguenti caratteristiche tecniche:

Specifiche	ESP8266
MCU	Xtensa Single-Core 32-bit L106
802.11 b/g/n Wi-Fi	Si, HT20
Frequenza core	80MHz
SRAM	160KB
FLASH	SPI Flash 16 MB
GPIO	17
Hardware / Software PWM	NO / 8 CANALI
SPI / I2C / I2S / UART	2 / 1 / 2 / 2
ADC	10bit

Questo potente microcontrollore è ottimo per poter eseguire tutti i calcoli necessari al fine di ottenere il data log dei consumi di cui si necessita.

L'ESP8266 dispone anche di un adattatore WiFi interno che, successivamente, permetterà l'interfaccia in rete locale con il server principale di data logging.

4.2.7. Controllo abilitazione uscita ed alimentazione del dispositivo

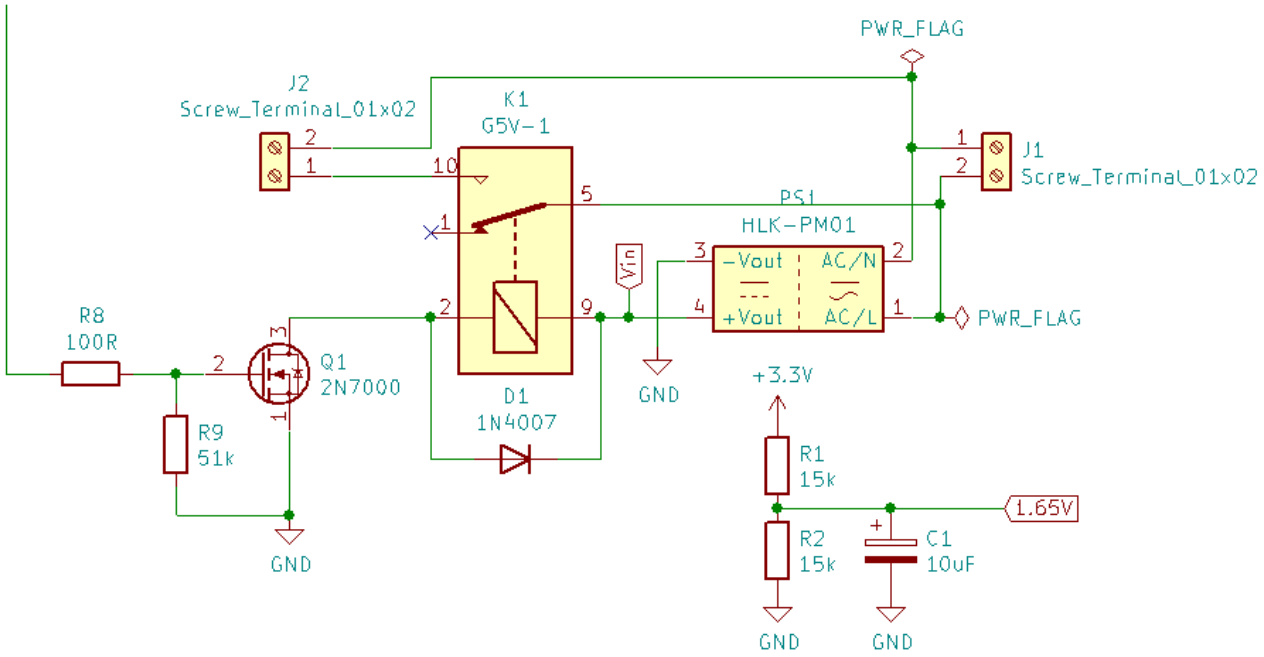

Figura 20: Scheda di alimentazione e di controllo abilitazione dell'uscita.

L'alimentazione dell'intero dispositivo è affidata ad un piccolo convertitore AC-DC che genera una tensione di 5V (necessaria all'alimentazione del relè K_1) con una massima corrente disponibile di 700mA, più che sufficiente al dispositivo per poter svolgere il suo compito.

La tensione V_{cc} di 3.3V è fornita dal regolatore lineare low-dropout integrato nel circuito della NodeMCU.

L'abilitazione della bobina del relè è data dal pin GPIO5 dell'ESP8266, a sua volta collegato al gate del transistor MOSFET-N Q_1: se questo pin viene abilitato via software, Q_1 andrà in conduzione, richiamando corrente attraverso la bobina del relè, attivandolo.

Q_1 è stato scelto in modo tale da essere un "Logic Level MOSFET", in quanto la tensione di comando è inferiore alla tensione di drain: in questo modo, Q_1 potrà sempre essere messo in piena conduzione superata una fissata soglia di tensione minore di 3.3V.

Il contatto del relè è stato messo in serie ad uno dei due conduttori che trasportano la tensione di rete tra la rete stessa ed il carico, in modo da poterlo disconnettere quando se ne ha la necessità.

Particolare importanza riveste il diodo D_1 di free wheeling o di ricircolo: infatti, quando si vorrà disconnettere il carico, si andrà a disattivare Q_1; essendo la bobina del relè un induttanza, essa cercherà per sua natura di opporsi alle variazioni di corrente.

Questo causerà un eccesso di cariche ai suoi due estremi dovuto al collasso del campo magnetico generato precedentemente che farà elevare la tensione ai suoi capi a livelli tali da innescare archi elettrici che potrebbero danneggiare altri componenti vicini.

Onde evitare questo fenomeno, deve essere piazzato un diodo in antiparallelo alla bobina, in modo tale da permettere alla corrente di defluire senza creare problemi.

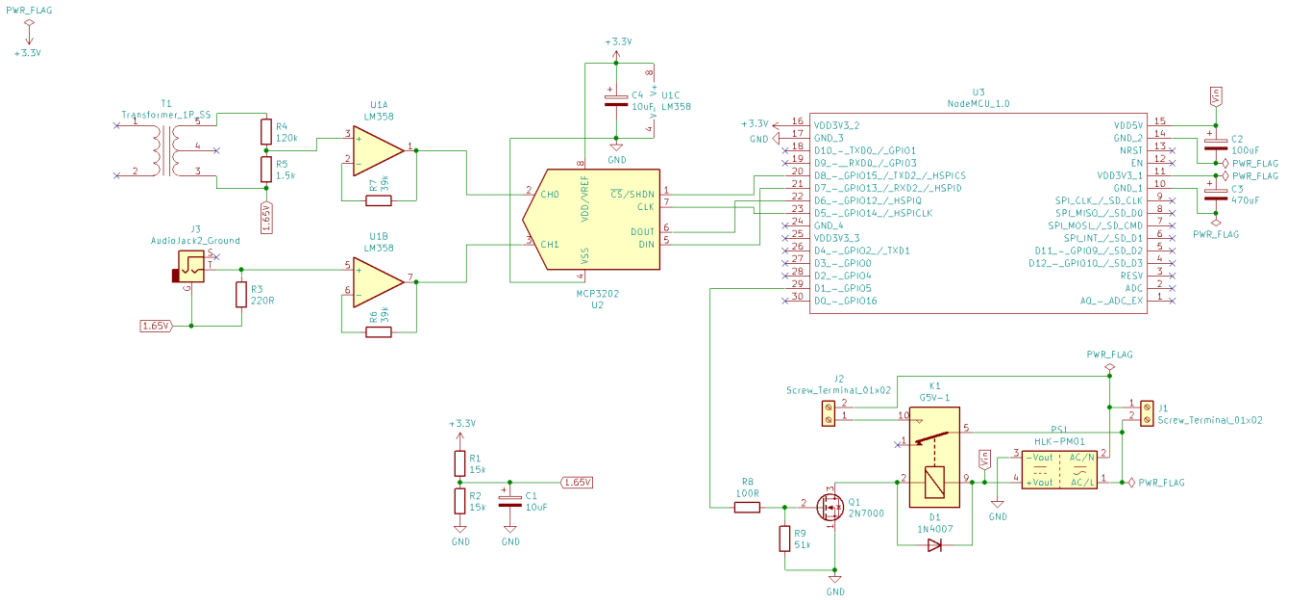

Figura 21: Schema elettrico completo del progetto PowerMonitor.

4.3. Firmware

Per la scrittura del firmware del dispositivo si è scelto di utilizzare l'IDE PlatformIO: un'estensione del sempre più famoso IDE Visual Studio Code.

PlatformIO è attualmente compatibile con un'infinità di piattaforme e framework e permette di scrivere codice in maniera ordinata e precisa grazie soprattutto all'integrazione con VS Code.

La scelta del framework da utilizzare all'interno di PlatformIO per la scrittura del firmware del dispositivo è ricaduta su Arduino, in quanto per questo framework esistono moltissime librerie compatibili per la famiglia di microcontrollori ESP8266.

4.3.1. La libreria MCP3202

Allo scopo di gestire l'interfaccia tra la periferica SPI interna al microcontrollore e l'ADC, è stata implementata una piccola libreria che sfrutta a sua volta la libreria SPI del framework Arduino per la gestione dell'MCP3202.

4.3.2. Struttura della libreria MCP3202

La libreria permette di comunicare un comando di campionamento all'ADC e di leggere il risultato di tale operazione; altre operazioni, come ad esempio l'utilizzo della modalità pseudo-differenziale, non sono state implementate nella suddetta libreria in quanto esulano dall'utilizzo voluto dell'ADC in questo progetto.

```
class MCP3202{
  private:
    uint8_t CS; //Chip select.

  public:
    //CS: PIN chip select.
    MCP3202(uint8_t CS);

    //ch: Channel (0, 1).
    uint16_t read(uint8_t ch);
};
```

In questo frammento di codice, è rappresentata la struttura della classe per la gestione dell'ADC.

I metodi definiti sono solamente il costruttore `MCP3202(uint8_t CS)`, che permetterà l'inizializzazione la proprietà CS della classe ed una `read(uint8_t ch)`, che servirà a scatenare una lettura su uno dei due canali dell'ADC per poi restituirne i risultanti campioni.

La proprietà CS della classe indica il pin fisico alla quale è collegato il pin di Chip Select dell'ADC.

Il corpo della funzione `read(uint8_t ch)` è stato implementato grazie alle specifiche fornite nel datasheet dell'MCP3202: tramite la sua chiamata, vengono inviate sul bus SPI delle specifiche sequenze di bit atte ad attivare una lettura su uno dei due canali dell'ADC.

A questo punto, una volta digitalizzato il valore, l'ADC ritornerà come messaggio successivo al messaggio di richiesta di un campione, il campione stesso a 12 bit codificato in due byte in entrata dal bus SPI.

Questo infine verrà letto, codificato in una variabile intera senza segno a 16 bit e restituito dalla funzione `read()`.

4.3.3. La libreria PowerMonitor

In questa libreria, implementata allo scopo di raggruppare tutte le operazioni di trasformazione e manipolazione dei campioni letti per rendere il codice più compatto e leggibile, sono presenti tutte le principali operazioni descritte nel capitolo precedente per la misurazione dei parametri elettrici e per l'esecuzione dei calcoli effettivi.

4.3.4. Struttura della libreria PowerMonitor

L'operazione principale di questa libreria è quella di stimare i parametri descritti nel capitolo precedente; il calcolo effettivo è eseguito chiamando il metodo `calculate()` della classe.

L'utilizzo della classe è il seguente:

1. Grazie ad una `struct PM_Parameters` interna alla libreria, si specificano diversi parametri di configurazione, come alcune costanti citate nel capitolo 3 necessarie per alcuni calcoli.

Tipo parametro	Nome parametro	Valore predef.	Significato
`int`	`bitResolution`	`N/A`	Risoluzione in bit del singolo campione.
`float`	`ADCMaxVoltage`	`N/A`	Tensione di lavoro dell'ADC.
`float`	`transformerRatio`	`N/A`	Ratio o fattore di trasformazione del trasformatore per la misura della tensione di rete.
`float`	`voltageDividerRatio`	`N/A`	Fattore di trasformazione del partitore resistivo ottenibile da $^{R_5}/_{R_4}$.
`float`	`currentClampRatio`	`N/A`	Fattore di trasformazione della pinza amperometrica; nel caso del sensore SCT013-000, il fattore vale $^{50mA}/_{100A}$.
`int`	`currentClampResistor`	`N/A`	Il valore in Ohm della resistenza R_3.
`float`	`V_correctionFactor`	`0`	Un fattore di correzione che si va a sommare a transformerRatio.
`float`	`I_correctionFactor`	`0`	Un fattore di correzione che si va a sommare a currentClampRatio.

2. Una volta specificati i vari parametri, basterà istanziare un oggetto di tipo `PowerMonitor` a cui passare come parametri del costruttore i due riferimenti ai buffer dove sono salvati i campioni e la loro lunghezza.

3. Successivamente si dovrà chiamare il metodo `setDataParameters` a cui si dovrà passare come parametro la `struct PM_Parameters` prima creata.

4. A questo punto, la libreria possiede tutti i dati di cui necessita, quindi basterà invocare la funzione `calculate()` per avviare il calcolo effettivo.

5. Alla fine del calcolo, basterà ricavare tutti i parametri calcolati tramite i vari metodi getter della classe.

4.3.5. Struttura del firmware

Di seguito, sono elencate tutte le librerie utilizzate nel firmware del dispositivo:

Libreria	Descrizione
`<ESP8266WiFi.h>`	Per includere tutte le funzioni di controllo delle varie periferiche a bordo dell'ESP8266.
`<MCP3202.h>`	Per la gestione dell'ADC esterno.
`<PowerMonitor.h>`	Per la gestione delle operazioni di calcolo dell'intero progetto.
`<PubSubClient.h>`	Per la comunicazione con un broker MQTT.
`<ESP8266WebServer.h>`	Per la gestione di un Web Server sincrono.
`<ESP8266HTTPClient.h>`	Per formulare e gestire richieste HTTP.
`<ESP8266httpUpdate.h>`	Per poter eseguire aggiornamenti OTA del filesystem e del firmware tramite il download di file da un server HTTP esterno.
`<ArduinoJSON.h>`	Per serializzare e deserializzare stringhe in JSON.
`<LittleFS.h>`	Per poter utilizzare un piccolo filesystem locale dove memorizzare file ed impostazioni.

Il firmware è stato scritto per poter implementare le seguenti funzionalità:

1. Campionamento dei due segnali e calcolo dei vari parametri sopra discussi.
2. Client MQTT per:
    a. Eseguire il data logging dei dati calcolati in formato JSON ad un server esterno di cui si tratterà nel prossimo capitolo.
    b. Offrire un endpoint per il controllo di abilitazione del dispositivo stesso.
3. Server HTTP per:
    a. Eseguire la configurazione delle impostazioni del WiFi e di altri parametri.
    b. Funzionare da endpoint per le varie richieste di informazioni sul dispositivo e di keep-alive.

4.3.6. Servizi esposti dal server HTTP del dispositivo

Le varie route servite dal server HTTP del dispositivo sono le seguenti:

- Staticamente, tutti i file presenti nel filesystem LittleFS.
- `/networks` per eseguire una scansione di tutte le reti WiFi viste dal dispositivo.
- `/alive` per ottenere tutte le impostazioni di configurazione del dispositivo.
- `/update`
    - `/wifi` per poter aggiornare le impostazioni di connessione alla rete WiFi del dispositivo.
    - `/name` per poter aggiornare il nome del dispositivo.
    - `/fs` per poter aggiornare il firmware del dispositivo*.
    - `/fw` per poter aggiornare l'immagine binaria del filesystem LittleFS nel dispositivo*.

*Aggiornamento di filesystem o firmware: all'esecuzione di questa procedura, il file `firmware.bin` o `littlefs.bin`, deve essere disponibile presso la root di un web server in ascolto dalla stessa macchina da cui si effettua la richiesta di aggiornamento: in questo modo, il dispositivo riuscirà a prelevare il file via HTTP, aggiornandosi; un web server configurabile da utilizzare a tale scopo è Live Server, disponibile sotto forma di estensione VS Code ed utilizzato in questo progetto.

4.3.7. Servizi esposti tramite richiesta via MQTT

Tramite dei dati in JSON opportunamente formattati ed inviati al topic MQTT `/PowerMonitor`, è possibile richiedere l'esecuzione di una determinata operazione riguardante il dispositivo stesso.

I dati in JSON sono strutturati come segue:

```
{
  "name":   "nome_registrato_del_dispositivo",
  "ip":     "ip_registrato_del_dispositivo",
  "action": tipo_di_richiesta
}
```

Tramite la chiave intera `action`, è possibile specificare una delle seguenti operazioni da eseguire:

Valore di `action`	Azione corrispondente
0	Disabilitazione dell'uscita
1	Abilitazione dell'uscita
2	Stato ed informazioni sul dispositivo

4.3.8. Interfaccia tra firmware e web app

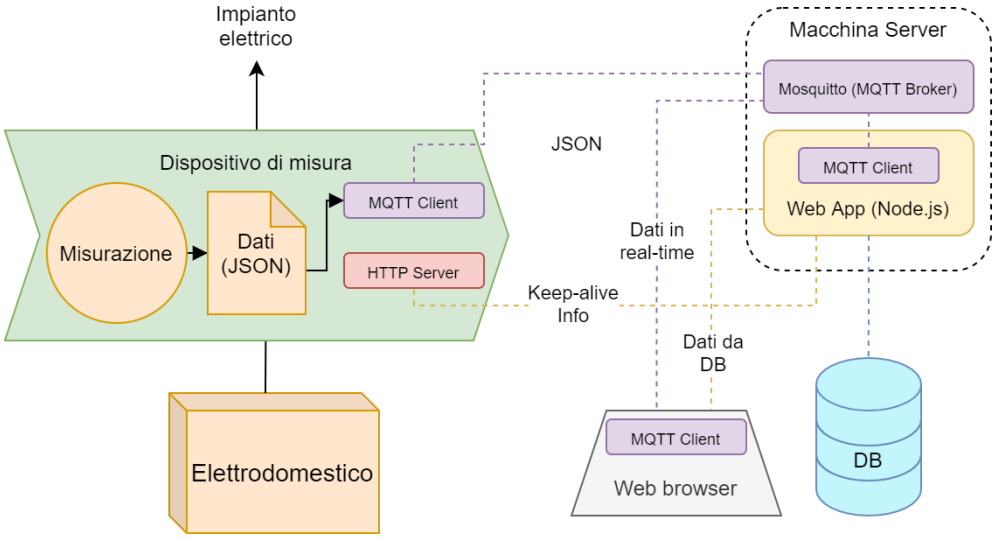

Figura 22: Schema approfondito di interfaccia tra il firmware, server, DB e web browser.

Una volta calcolati i vari parametri d'interesse, il dispositivo di misura si connetterà ad un broker MQTT per la pubblicazione dei messaggi contenenti i dati.

Questi messaggi, rispettando lo standard JSON, sono facilmente decodificabili ed utilizzabili dal ricevente; il protocollo MQTT consente inoltre la notifica automatica di ogni client MQTT connesso nel momento della pubblicazione di un nuovo messaggio, garantendo così il real-time data logging.

I messaggi, come da specifica MQTT, verranno inoltrati a tutti i client sottoscritti ad un determinato topic per il log dei dati; dopodiché, questi verranno:

1. Salvati sotto forma di dati JSON all'interno del database.
2. Mostrati all'utente finale tramite interfaccia web.

4.3.9. Configurazione del dispositivo

Il firmware, come detto prima, offre anche un servizio di server HTTP, che espone all'utente finale un piccolo portale di configurazione delle impostazioni del singolo dispositivo; tra queste impostazioni si possono trovare il nome assegnato al dispositivo e le corrispettive impostazioni di collegamento alla rete WiFi.

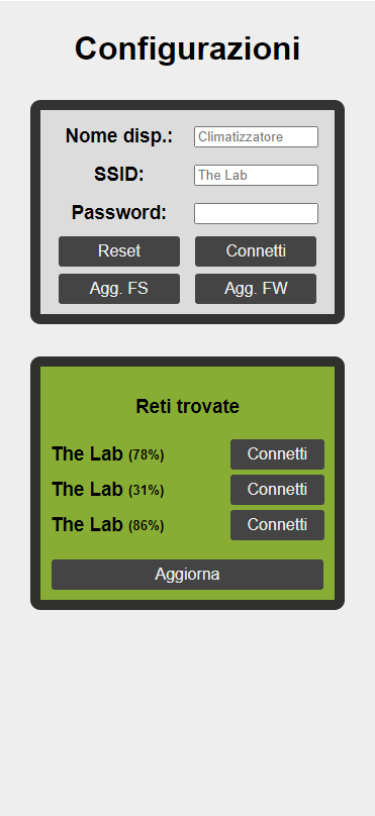

Figura 23: Il portale di configurazione offerto dal server HTTP del dispositivo.

Durante il collegamento alla rete WiFi, il LED integrato della NodeMCU continuerà a lampeggiare ad intervalli di mezzo secondo.

Nell'eventualità di connessione non riuscita alla rete WiFi, come nel caso della prima configurazione, il dispositivo andrà in modalità di configurazione avviando un hotspot WiFi con SSID pari a `PowerMonitor`, al quale si potrà effettuare la connessione per poter successivamente accedere al portale di configurazione all'indirizzo `http://192.168.4.1/`.

La modalità di configurazione si può riconoscere facilmente dal LED integrato della NodeMCU acceso in maniera persistente.

È importante assegnare anche un nome al dispositivo in questione in quanto il software sul server web principale e sul client web, lo utilizza come identificativo per poter filtrare i messaggi in arrivo dai possibili molteplici dispositivi già configurati in precedenza.

Una volta connesso al WiFi, il dispositivo cercherà automaticamente di connettersi all'indirizzo del broker MQTT specificato nel codice (non modificabile in questa release dal portale web).

4.4. Tecnologie utilizzate

Le tecnologie, i protocolli di codifica e di trasmissione dati e le piattaforme di sviluppo utilizzate saranno trattate in questa sezione.

4.4.1. JSON

JSON (JavaScript Object Notation) è al giorno d'oggi lo standard per lo scambio di dati tra applicativi più utilizzato in ambito web, che permette, tramite una sintassi di facile comprensione, di trasportare in una sola stringa di testo una rappresentazione facilmente decodificabile di dati singoli o multipli sotto forma di coppie chiave-valore.

Es.: Nel progetto PowerMonitor, un esempio di dati JSON trasmessi tramite MQTT tra il dispositivo PowerMonitor ed il server di data-logging / client, può essere il seguente:

```
{
    "v": { "p_p": 315, "n_p": -315, "pp": 630, "rms": 220 },
    "i": { "p_p": 2, "n_p": -2, "pp": 4, "rms": 1.41 },
    "p": { "VA": 315, "W": 315, "VAr": 0, "pf": 1 }
}
```

L'oggetto JSON è composto, come detto prima, da coppie chiave-valore ed una chiave è definita principalmente come una stringa; una volta trasformata l'intera stringa, che rappresenta l'oggetto JSON, in un oggetto Javascript vero e proprio, si potrà fare riferimento al valore della chiave sopra citata tramite la dot notation.

Un valore di un oggetto JSON può invece essere definito come un sotto-oggetto JSON, come un Array di valori, tramite la notazione `[val_1, val_2, …, val_n]`, o semplicemente come un singolo valore.

Si noti bene che un array può contenere come elementi non solo altri array, dando così la possibilità di definire strutture più complesse, come le matrici, ma anche altri oggetti, rendendo possibile persino la rappresentazione di array di oggetti.

Come è facilmente intuibile, lo standard è molto elastico e leggero in quanto, una volta riconvertito l'oggetto Javascript in stringa JSON, verranno eliminati tutti gli spazi (non importanti nella fase di decodifica quanto invece per la leggibilità), alleggerendo di molto il peso del payload del messaggio.

L'alternativa allo standard JSON più popolare in passato era l'XML (eXtensible Markup Language), che, come l'HTML, presentava una struttura gerarchica a tag, dove però questi ultimi potevano essere definiti dall'utente.

Questo standard risulta essere più dispendioso per la codifica, trasmissione e decodifica delle informazioni rispetto a JSON.

4.4.2. Node.js

Come accennato nel capitolo precedente, per l'implementazione della logica di back-end del progetto, si è scelto l'utilizzo del sempre più popolare Node.js.

Node.js è un runtime Javascript, ovvero un ambiente che permette l'esecuzione di codice Javascript come se fosse un qualsiasi linguaggio di programmazione.

Javascript nasce inizialmente come un linguaggio web di scripting eseguibile solamente all'interno di un web browser e, successivamente, grazie a Node.js, è diventato anche un linguaggio di programmazione come qualsiasi altro.

Ciò riveste una notevole importanza, considerando che Javascript è uno dei linguaggi più veloci e facili da apprendere ed è anche uno dei più conosciuti al mondo.

La potenza di Node.js deriva anche dall'ampia disponibilità di moduli scaricabili; un modulo è da considerarsi come una libreria che verrà poi inclusa nel progetto.

La gestione della configurazione del progetto e dei moduli è affidata ad npm (Node Package Manager), che permette l'installazione e l'inclusione nel progetto, di un qualsiasi modulo a noi necessario, potendone così sfruttate tutte le funzionalità.

Grazie alla vasta gamma di moduli disponibili per Node.js ed anche grazie alla flessibilità e all'intuitività del linguaggio Javascript, lo sviluppo di tutte le funzionalità previste dal progetto è risultata non troppo difficoltosa.

Come moduli utilizzati nel progetto sono presenti:

Modulo	Descrizione
Express-generator (installato globalmente)	Wizard generatore della struttura di un progetto Node.js basato sul framework Express.js.
Express.js	Framework utilizzato in questo progetto.
Ejs	Template engine scelto da abbinare ad Express.js
Mongoose	Modulo di interfaccia con il database NoSQL MongoDB.
Mqtt	Modulo per la gestione di un client MQTT.
TensorFlowJs per Node.js	Modulo che permette la creazione, la gestione e l'utilizzo di reti neurali atte a predire i dati dei futuri.
Najax	Modulo per poter eseguire richieste HTTP in stile JQuery direttamente da un server Node.js
Ip-to-int	Modulo per la conversione da indirizzo IP ad intero.

4.4.3. Il framework Express.js

Express.js, o semplicemente Express, è un framework per Node.js progettato per creare non solo web app, ma anche API ed è ormai definito come il server framework standard per Node.js.

Express può essere installato ed utilizzato globalmente tramite il comando
`npm -g install express-generator`
Ad installazione completata, il comando wizard `express`, atto a generare una struttura scheletro per un progetto Express, sarà direttamente invocabile dalla shell che si sta utilizzando.

Una volta installato globalmente `express-generator`, basterà creare una nuova directory chiamata con il nome desiderato del progetto e, dopo aver aperto una shell nella directory stessa, basterà richiamare il comando `express --view ejs`, che andrà a generare tutti i files e le directory necessarie al progetto Express.

Infine, per completare l'installazione di tutti i moduli mancanti necessari, basterà eseguire nella stessa directory il comando `npm install`.

Con il parametro `--view ejs`, si specifica al wizard che, come template engine del progetto per le view, si vuole utilizzare EJS.

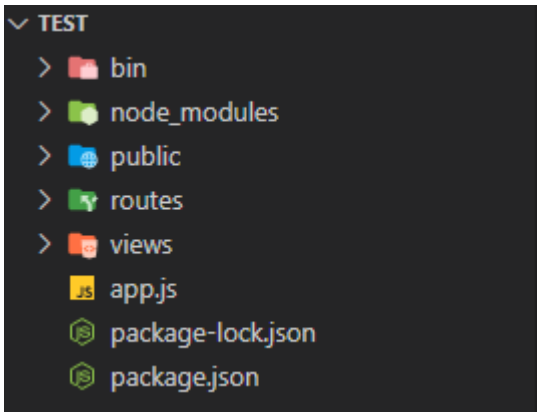

Figura 24: Struttura base di un progetto Node.js basato sul framework Express.js.

All'interno dei files generati dal wizard, sarà anche presente del codice boilerplate da poter espandere a proprio piacimento a seconda delle funzionalità che è necessario implementare nella web application.

Directory o file	Descrizione
node_modules	Directory predefinita di npm per il salvataggio di tutti i moduli richiesti dall'utente.
bin	Directory contenente i file di esecuzione relativi ad Express.
public	Document root del server Node.js.
routes	Directory contenente tutti i file che specificano gli endpoint per i client web.
views	Directory contenente tutte le views ejs che possono essere renderizzate come risposta ad una determinata route.
package.json	File JSON contenente tutte le configurazioni e le dipendenze del progetto.

Per l'avvio del progetto, basterà richiamare il comando *npm start*, che andrà ad eseguire automaticamente il Javascript presente nel file ./bin/www, facendo così

avviare un web server sulla porta 3000. Questa porta è quella predefinita utilizzata da Express ed è modificabile editando il file ./bin/www stesso.

4.4.4. MongoDB e Mongoose

Il modulo Mongoose rende possibile l'interfaccia da un progetto Node.js al database non relazionale MongoDB.

MongoDB è un DBMS (DataBase Management System) open source con la particolarità principale di poter organizzare i dati salvati in documenti in formato JSON; questi documenti sono organizzati in collezioni, dove un database può essere composto da una o più collezioni.

Questa filosofia di database si adatta molto bene al progetto PowerMonitor che, come formato principale per lo scambio di dati tra i vari dispositivi, utilizza proprio JSON.

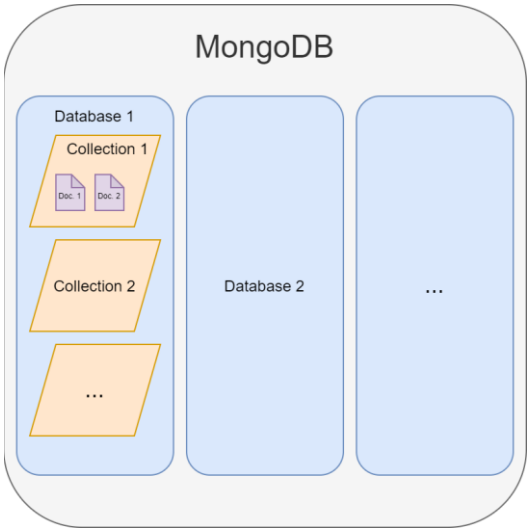

Figura 25: Struttura di un generico database MongoDB.

4.4.5. TensorFlow.js

Per poter eseguire delle inferenze sui dati al fine di poter predire i consumi futuri ipotizzati, si è reso necessario l'utilizzo di un modulo atto a gestire strutture dati matematiche più complesse apposite per questo compito, cioè, le reti neurali.

Tramite l'utilizzo di reti neurali, si ha la possibilità di eseguire, nello specifico, anche dei task di regressione; questo non avviene tramite la ricerca di una funzione che al meglio approssima l'andamento reale dei dati raccolti, come ad esempio farebbe una regressione lineare, bensì tramite:

1. L'addestramento (train) di un modello di predizione.
2. Le inferenze sul modello di predizione stesso per poter ottenere i valori della regressione.

Una rete neurale è composta da livelli (layer) ed ogni livello ha un certo numero di neuroni:

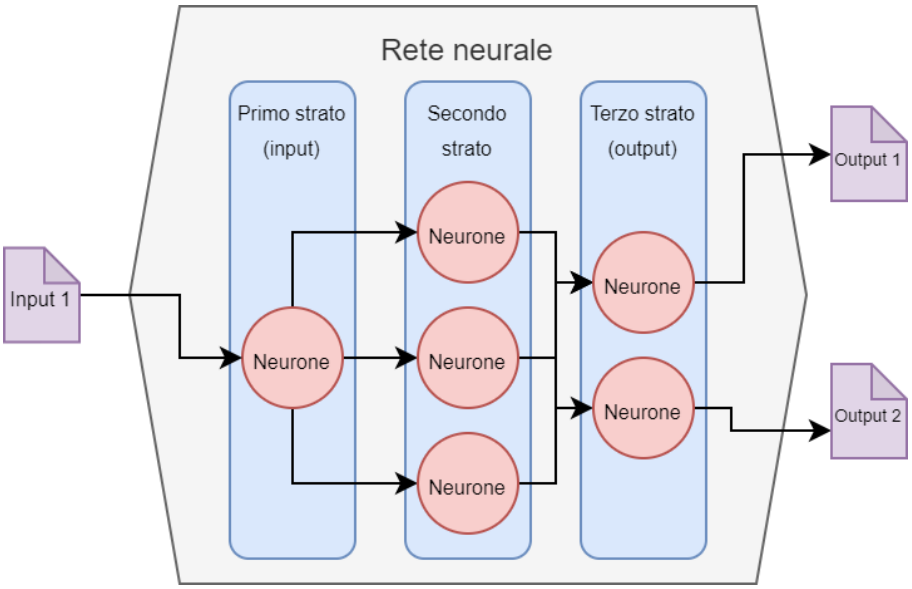

Figura 26: Struttura di una generica rete neurale con un input e due output.

In figura 27, la funzione che la rete approssimerà, sarà della forma $f: IR \rightarrow IR^2$, in quanto la rete prende in input uno o più valori monodimensionali e ritorna come risultato un output appartenente ad uno spazio bidimensionale.

Per poter consentire la predizione di dati veritieri, il modello deve essere prima addestrato con dati già presenti; questo va fatto per poter "regolare" i parametri relativi a ciascun neurone interno alla rete in modo corretto ed attinente ai dati che vengono utilizzati al momento del training.

Il train di un modello di predizione avviene con i seguenti passi:

1. Specificando la struttura di base della rete neurale associata al modello e definendo alcuni parametri della rete stessa come il numero di livelli, il numero di neuroni per ogni livello, la funzione di attivazione per ogni livello ecce cc...
2. Specificando le impostazioni specifiche del training come il learning rate ed il batch size.
3. Aggiungendo dei valori normalizzati appartenenti al dominio della funzione approssimante con i loro corrispettivi valori normalizzati di codominio, cioè, in termini più pratici, aggiungendo i dati raccolti per poter fare il training del modello.
4. Avviando effettivamente l'operazione di training che avviene analogamente al metodo delle flashcard a livello umano; questo accade perché l'algoritmo di training presenta alla rete i dati aggiunti precedentemente con dei valori noti in ordine sparso e casuale, "abituando" così la rete al riconoscimento di determinati pattern presenti in questi stessi dati.

Come è intuibile, più dati si hanno a disposizione e più il modello della rete (a livello di struttura della rete stessa) sarà adatto ai dati, più le inferenze eseguite in futuro sul modello saranno precise.

Il processo di training, specialmente nella fase 4, è molto pesante dal punto di vista computazionale; per questa ragione, il modello ottenuto da questa operazione può essere salvato sotto forma di file e ricaricato in altri dispositivi solo per l'esecuzione di inferenze.

TensorFlow.js include infatti un metodo per il salvataggio del modello ed in più, in questa stessa libreria, si potrebbero anche includere modelli esportati provenienti da ambienti completamente diversi, come, ad esempio, provenienti dal framework Keras per Python.

Se si vuole invece eseguire un training direttamente da Node.js tramite TensorFlow.js, come nel progetto PowerMonitor stesso, gli sviluppatori del modulo mettono a disposizione principalmente tre versioni del modulo stesso

1. `@tensorflow/tfjs-node`
2. `@tensorflow/tfjs-node-gpu`
3. `@tensorflow/tfjs`

La differenza tra questi tre moduli è data dall'accelerazione hardware disponibile in ognuno di essi, per poter accelerare processi computazionalmente pesanti come il training.

La libreria TensorFlow.js offre tre versioni diverse di accelerazione hardware:

1. Supporto di un eseguibile compilato in linguaggi più a basso livello dipendenti dall'architettura hardware, come C e C++, che potranno sfruttare al 100% tutte le potenzialità della CPU dell'host.

2. Supporto di un eseguibile compilato, ancora una volta, in linguaggi più a basso livello, che andranno a sfruttare, se presente, l'accelerazione grafica di una scheda video basata su CUDA; questo modulo, se abbinato ad un hardware adatto, sarà il più performante.

3. Nessuna: questo modulo, che è stato scritto per poter eseguire tutti i calcoli richiesti tramite puro Javascript, è il più lento dei tre, ma anche il più compatibile e può anche essere eseguito browser-side.

Una volta ottenuto il modello addestrato, può essere utilizzato in maniera molto semplice da qualsiasi tipologia di dispositivo avente installato un interprete Javascript; in questa fase, lo sforzo computazionale è minimo.

La fase di predizione, cioè la fase di calcolo delle inferenze tramite l'ausilio di un modello, è suddivisa nei seguenti passi:

1. Normalizzazione dei dati in input per i quali si vuole ottenere un output.
2. Calcolo della predizione con quel dato input.
3. De-normalizzazione del risultato e restituzione dei dati.

In questa fase, se il modello è stato ben allenato, i dati risultanti dalle inferenze saranno molto attendibili.

4.4.6. MQTT e Mosquitto

Tramite l'utilizzo del protocollo MQTT e del modulo MQTT per Node.js, è stato possibile interfacciarsi con un broker MQTT esterno per lo scambio di dati.

MQTT è un protocollo utilizzato per lo scambio dei dati tra vari client, i quali possono assumere il ruolo di publisher o subscriber; i client MQTT, per poter comunicare, si dovranno connettere ad un broker MQTT.

Un broker MQTT è un server atto a raccogliere tutti i messaggi inviati dai publisher ed a inoltrarli a tutti i subscriber sottoscritti al topic specificato nel messaggio del publisher; il ruolo di MQTT Broker nel progetto in questione è stato assegnato all'applicativo Mosquitto.

Questo tipo di broker supporta anche le connessioni su protocollo WebSocket, aspetto molto importante in quanto il client da browser web, per potersi connettere al broker, può solamente usare questa modalità di connessione.

Prima di poter ricevere un qualsiasi tipo di messaggio, un subscriber dovrà per prima cosa sottoscriversi ad un determinato topic, cioè ad una sorta di canale broadcast dove in futuro verranno pubblicati dei messaggi.

Un topic è formato da una generica stringa, ma, per convenzione, spesso si utilizza un path name, per dare un significato intrinseco al topic.

Es.: Topic

1. `/PowerMonitor:` topic per la pubblicazione di richieste.
2. `/PowerMonitor/data:` topic di risposta alle richieste.
3. `/PowerMonitor/state/data:` topic di risposta per informazioni sullo stato di abilitazione del dispositivo.

All'arrivo di un messaggio, tutti i client sottoscritti al topic del messaggio verranno notificati per poter gestire il messaggio stesso.

4.4.7. Tecnologie di design e scripting browser-side

Per il design della UI della web app, sono state usate le classiche tecnologie di programmazione web: HTML, CSS e Javascript.

L'utilizzo di Javascript sia lato server, sia lato client, ha semplificato di molto lo sviluppo e l'integrazione tra server e client; ancora una volta, l'utilizzo dello standard JSON permette un'immediata conversione da una stringa JSON ad un oggetto Javascript pronto per l'uso.

4.4.8. JQuery

Per l'interfaccia con il DOM (Document Object Model) e per l'esecuzione di richieste AJAX al server stesso o al dispositivo PowerMonitor, si è scelto di utilizzare la famosissima ed intuitiva libreria JQuery.

JQuery è una libreria (o più correttamente un piccolo framework) in grado di semplificare la programmazione in Javascript estendo le sue funzionalità native e garantendo il funzionamento cross-browser degli script.

Questa libreria è distribuita sotto forma di file Javascript includibile facilmente nello script lato client ed è accessibile ed utilizzabile tramite l'oggetto globale $ o JQuery.

Tramite JQuery, l'interfaccia con il DOM risulta essere molto semplificato: tramite la funzione `$("css-selector")`, si è in grado di selezionare uno o più elementi dal DOM tramite il `css-selector` specificato come stringa.

Una volta selezionati gli elementi a cui si vogliono apportare modifiche, si richiama sul valore ritornato (un'oggetto JQuery) il metodo desiderato, applicando così a quell'oggetto i cambiamenti desiderati.

La libreria JQuery, inoltre, offre dei metodi semplificati per la formulazione ed esecuzione di richieste AJAX tramite i seguenti metodi:

1. `$.ajax({settings})`
   Il metodo tradizionale per la formulazione di una richiesta AJAX in JQuery: settings è un oggetto Javascript contenente le varie impostazioni della richiesta.
2. `$.get(url, {parameters})`
   Metodo abbreviato dove il verbo per la richiesta HTTP è GET.
3. `$.post(url, {parameters})`
   Metodo abbreviato dove il verbo per la richiesta HTTP è POST.

Tutti e tre i metodi sopra citati, in assenza di un altro parametro che rappresenta una funzione di callback atta ad essere eseguita alla risoluzione della richiesta, ritorneranno un oggetto Promise.

In quanto compatibili con le Promise, i seguenti metodi possono essere concatenati con i classici metodi `.then()` e `.catch()` delle Promise o con i più moderni operatori `async` ed `await`.

Utilizzando questo stile, il codice scritto nel progetto in questione è risultato più facile da implementare e re interpretare in quanto questi due operatori fanno lavorare con uno stile asincrono, un codice che semanticamente possiede uno stile sincrono.

4.4.9. Chart.js

Nel progetto, per la visualizzazione dei dati in dei grafici tempo vs valore, si è fatto uso della libreria lato client Chart.js.

Chart.js è una libreria molto completa in grado di creare e gestire grafici che possono essere anche molto complessi e particolareggiati; ancora una volta, grazie alla ricca documentazione della libreria messa a disposizione dello sviluppatore, l'implementazione dei grafici interessati è risultata non troppo complicata.

La libreria sfrutta un canvas HTML per il rendering del grafico, dove tutte le sue configurazioni vengono specificate tramite un'oggetto Javascript, le quali chiavi ed il loro relativo significato sono specificate nella documentazione.

Una volta scelti lo stile del singolo grafico e le varie configurazioni dello stesso, basterà creare un costruttore Javascript parametrizzato atto a ritornare l'oggetto contenente le configurazioni complete da passare alla libreria.

4.5. Web Application

La web-application che costituisce il back-end dell'applicazione insieme al database, sono installabili in qualsiasi tipo di sistema server con una adeguata potenza di calcolo che supporti appunto Node.js e MongoDB.

4.5.1. Struttura del back-end

Il progetto PowerMonitor, come già spiegato in parte prima, si appoggia ad un back-end scritto in Node.js basato sul framework Express per l'orchestrazione delle operazioni principali di archiviazione, ritrovamento e predizione dei dati.

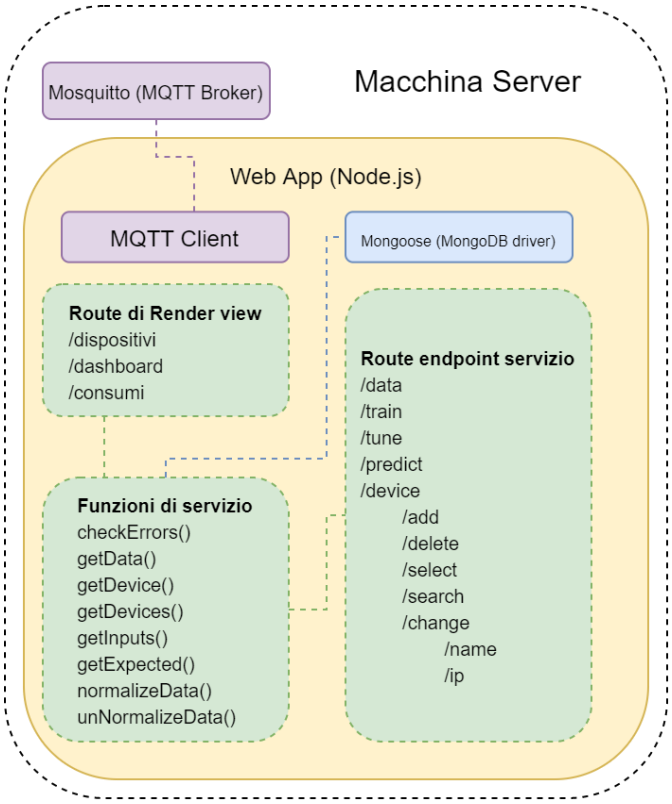

Figura 27: Struttura approfondita della Web App.

4.5.2. MQTT client

Il client MQTT presente sul server è utilizzato per l'interfaccia con l'applicativo MQTT Broker presente nella stessa macchina; in questo progetto, come già menzionato prima, si fa uso dell'MQTT broker Mosquitto.

Questo client, una volta sottoscritto al topic "/PowerMonitor/data", riceverà sullo stesso i dati provenienti dai dispositivi publisher registrati in precedenza.

4.5.3. Mongoose

Il driver di connessione al database è stato affidato al modulo Mongoose.

Grazie alla ricca documentazione presente sia per l'utilizzo di MongoDB tramite shell sia per l'utilizzo di Mongoose, lo sviluppo delle query di salvataggio, ritrovamento e raggruppamento dati, è risultato essere lineare.

Mongoose cerca, infatti, di offrire un driver per l'interrogazione del database MongoDB con una sintassi quanto più vicina possibile a quella usata nella documentazione ufficiale per l'interrogazione del database tramite la MongoShell (o `mongosh`).

Molti dei comandi sono infatti analoghi tra le due versioni, specialmente quelli del framework Aggregate di MongoDB, il quale permette operazioni varie in pipeline, tra cui l'aggregazione dei dati.

Il driver è pienamente compatibile sia con uno stile di programmazione basato sulle callback, sia su uno basato sulle promise.

4.5.4. Route di render view ed EJS

Per poter gestire le richieste di visualizzazione di pagine web, come accennato prima, è stato utilizzato un template engine integrato in Express chiamato EJS.

Tramite la creazione di file .ejs sotto la directory *views*, è possibile scrivere un codice dinamico composto da HTML statico misto a testo generato dinamicamente da codice Javascript lato server che può cambiare in funzione, appunto, dello stato attuale del server stesso e del database.

All'interno di un file .ejs, possono essere inclusi altri file .ejs tramite la direttiva

```
<%- include("nome_file") %>
```

questo permette la costruzione di layout finali con parti statiche in tutto il sito web.

Nel progetto in questione, esistono due view ejs nominate `header.ejs` e `footer.ejs`, che non vengono mai richiamate direttamente dal server per poter essere risolte ed inviate al client sotto forma di testo HTML, ma servono invece per essere incluse rispettivamente prima e dopo di ogni altra view, proprio per mantenere un header ed un footer statici per tutte e tre le pagine di gestione previste nell'applicativo.

Una view EJS potrà chiaramente essere richiamata dentro il codice di una determinata route per poter essere inviata come risposta al client che effettua la richiesta.

Al momento della chiamata per il render della view, possono essere passati dei parametri: questi potranno essere ripescati nel codice Javascript annegato nel file della view per poter essere stampati in determinate parti del codice HTML tramite l'istruzione `<%= nome_variabile %>`.

Il meccanismo delle view permette quindi al programmatore di semplificare di molto la generazione dell'HTML dinamico da inviare al client.

Le route per il render delle pagine web, cioè quelle richiamabili direttamente dal web browser, sono le seguenti:

1. `/` o `/dashboard` per la visualizzazione della dashboard.
   Essa permette di monitorare i consumi in real-time del dispositivo selezionato tramite la pagina `/dispositivi` e di abilitare o disabilitare il data logging e/o l'abilitazione del dispositivo stesso.
   Nella dashboard sono disponibili tre grafici:
   1. Tempo vs VA, W e VAr
   2. Tempo vs power factor
   3. Tempo vs V_{rms} ed I_{rms}

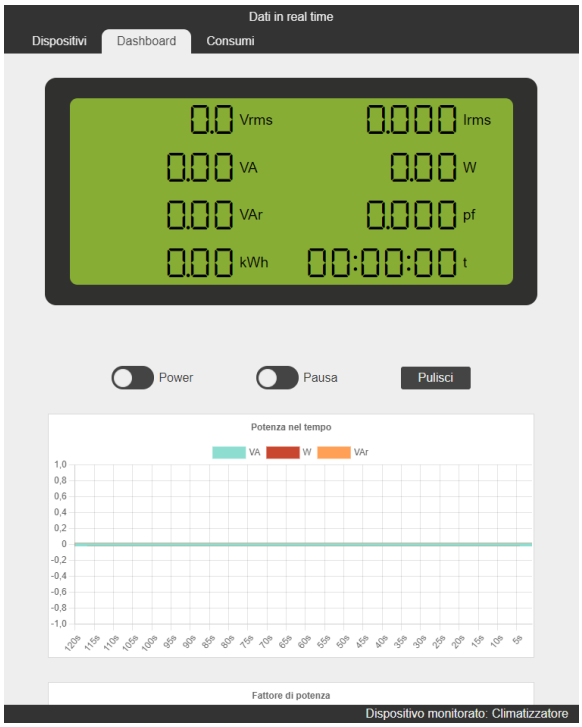

Figura 28: La dashboard del progetto.

2. `/dispositivi` per la visualizzazione di tutti i dispositivi PowerMonitor registrati nel sistema e del loro specifico stato in rete; questa pagina permette anche di registrare nuovi dispositivi nel sistema e di modificare anche alcuni parametri relativi a quelli già registrati. Un'altra funzionalità fondamentale di questa pagina è quella di poter selezionare il dispositivo da visualizzare nella scheda dashboard e del quale ricavare i consumi registrati dalla scheda consumi.

Figura 29: Pagina per la gestione dei dispositivi.

3. `/consumi` per la visualizzazione di tutti i dati relativi ai consumi del dispositivo selezionato; in questa pagina è possibile interrogare il database per poter ricavare i dati registrati e per poterli raggruppare per secondi, minuti, ore o giorni.

   Inoltre, sono presenti due tasti per poter eseguire il train di un nuovo modello di predizione per TensorFlow.js basandosi sui dati raggruppati per ore disponibili nel database o per poter eseguire un fine-tuning basato sui dati nuovi raccolti dopo la creazione di un primo modello.
   Se nei selettori per le date, le due date dovessero essere future, allora il sistema passerà in modalità predizione, rendendo possibile l'utilizzo del modello precedentemente addestrato per poter stimare i dati sui futuri consumi.

Figura 30: Pagina per la visualizzazione dei consumi.

   In questa pagina sarà anche presente una label nella quale, una volta ottenuti i dati desiderati, comparirà il consumo energetico calcolato relativo ad i dati visualizzati espresso in kilowattora (kWh).

4.5.5. Route di endpoint per i servizi

Queste route servono in risposta alle chiamate AJAX effettuare dai client web e sono le seguenti:

- `/data` per l'ottenimento di dati relativi al dispositivo selezionato; in questa route è anche presente l'algoritmo per l'interpolazione degli zeri, che si occupa di aggiungere dei dati vuoti atti a riempire gli istanti temporali dove non sono presenti dati registrati nel database.
- `/train` per l'avvio del training di un nuovo modello di predizione per TensorFlow.js.
- `/tune` per l'avvio del fine tuning di un modello di predizione precedentemente creato.
- `/predict` per il calcolo dei dati predetti relativi al dispositivo selezionato.

- `/device`
  - `/add`, `/delete`, `/select`, `/search`, rispettivamente per l'aggiunta, la cancellazione, la selezione e la ricerca di dispositivi.
    Riguardo alla ricerca di nuovi dispositivi, è stata implementata in modo tale da interrogare via HTTP, ogni dispositivo rientrante nel range di indirizzi IP della rete a cui è collegato il server; questo avviene tramite l'utilizzo della Subnet Mask e dell'indirizzo IP del server.
  - `/change`
    - `/name` per il cambio del nome di un dato dispositivo.
    - `/ip` per il cambio dell'indirizzo IP di un dato dispositivo.

4.5.6. Funzioni di servizio

Queste funzioni servono ad eseguire determinate operazioni di appoggio a quelle definite nelle route di servizio e sono le seguenti:

- `chechErrors(group, from, to)`
  Funzione utilizzata nelle richieste in cui sono presenti due date in input che rappresentano un intervallo di tempo; l'obiettivo della funzione è quello di validare le due stringhe ISO delle date verificando:

  1. Se le date inserite sono valide.
  2. Se la data finale è antecedente a quella iniziale.
  3. Che i dati richiesti rientrino in un range temporale minimo e massimo dipendente del raggruppamento voluto dei dati.
  4. Se il raggruppamento dei dati selezionato è valido.

- `getData(group, from, to, collection)`
  Funzione utilizzata per l'ottenimento ed il raggruppamento di dati da una determinata collezione.
  Questi dati devono essere compresi tra le due date specificate nei parametri.
  Per la validazione delle due date passate in input, si fa ricorso alla funzione di servizio checkErrors().

- `getDevice()`
  Funzione utilizzata per l'ottenimento delle informazioni da database riguardanti il dispositivo attualmente selezionato.

- `getDevices()`
  Funzione utilizzata per l'ottenimento delle informazioni da database riguardanti tutti i dispositivi registrati.
- `getInputs(data)`
  Funzione di conversione: il suo compito è quello di ricavare da dei dati provenienti dal database, un'oggetto contente i vari parametri da dare in input al modello TensorFlow.js; essi rappresentano i diversi input per i quali si vogliono fare inferenze o per i quali si vuole addestrare il modello.
- `getExpected(data)`
  Funzione di conversione: il suo compito è quello di ricavare da dei dati provenienti dal database, un'oggetto contente i vari parametri da dare in input al modello TensorFlow.js; essi rappresentano l'output previsto dei corrispondenti dati in input ottenuti da getInputs() nella sezione dedicata al training o al fine tuning del modello di predizione.
- `normalizeData(input, expected = null)`
  Funzione per la normalizzazione dei dati in input al modello di predizione; i due parametri input ed expected sono i due oggetti risultanti rispettivamente da getInput() e getExpected().
- `unNormalizeData(resultTensor)`
  Funzione per la de-normalizzazione dei dati in output dal modello di predizione.

4.5.7. Struttura del database

La struttura del database PowerMonitor è stata suddivisa in quattro collection:

1. `devices`
   Questa collection serve a contenere tutti i dati relativi ai dispositivi registrati nell'applicativo.
2. `values`
   Collection principale del progetto: serve a contenere per ogni dispositivo registrato, i dati che questo invia alla web app; i dati sono raggruppati per secondi.
3. `values_per_hours`
   Collection buffer: serve per contenere tutti i dati presenti in values, ma raggruppati per ore.

4. `predicted_values_per_hours`
   Collection buffer: ad ogni predizione effettuata, la collection viene svuotata; è utilizzata per contenere i dati predetti (raggruppati per ore) provenienti dal modello di predizione.
   In un secondo momento, verrà riutilizzato il framework Aggregate di MongoDB per poter raggruppare questi dati per unità di tempo maggiori all'ora (in questa release del progetto, solamente per giorni).

5. Installazione

L'installazione del dispositivo richiede parecchio tempo in quanto è richiesta sia la strumentazione adatta nella fase di montaggio del circuito elettrico e, dato che si lavora con la tensione di rete, anche una certa competenza nel campo dell'elettronica.

Il processo relativo alla fase di configurazione del software, risulta essere invece abbastanza lineare, ma si darà per assunto che la macchina server utilizzata sia una macchina Linux con una versione installata basata su Debian.

5.1. Fasi del processo di installazione.

Il processo di installazione dell'applicativo è diviso in sette fasi:

1. Creazione del circuito elettrico del dispositivo.
2. Flash del firmware e del filesystem LittleFS sull'ESP8266.
3. Installazione di MongoDB.
4. Installazione di Mosquitto.
5. Installazione di NVM e Node.js.
6. Configurazione della web app e del dispositivo.
7. Avvio del progetto.

5.2. Creazione del circuito elettrico del dispositivo

Per la creazione del circuito elettrico, sono state utilizzate due basette millefori insieme a dei componenti discreti THT.

Una volta presente lo schema elettrico, la realizzazione deve essere fatta seguendolo scrupolosamente e controllando che nelle saldature non ci siano corto circuiti non voluti e, cosa più importante, che alla fine del lavoro di saldatura non ci sia continuità tra il positivo di alimentazione e la massa.

5.2.1. Versione finale del circuito

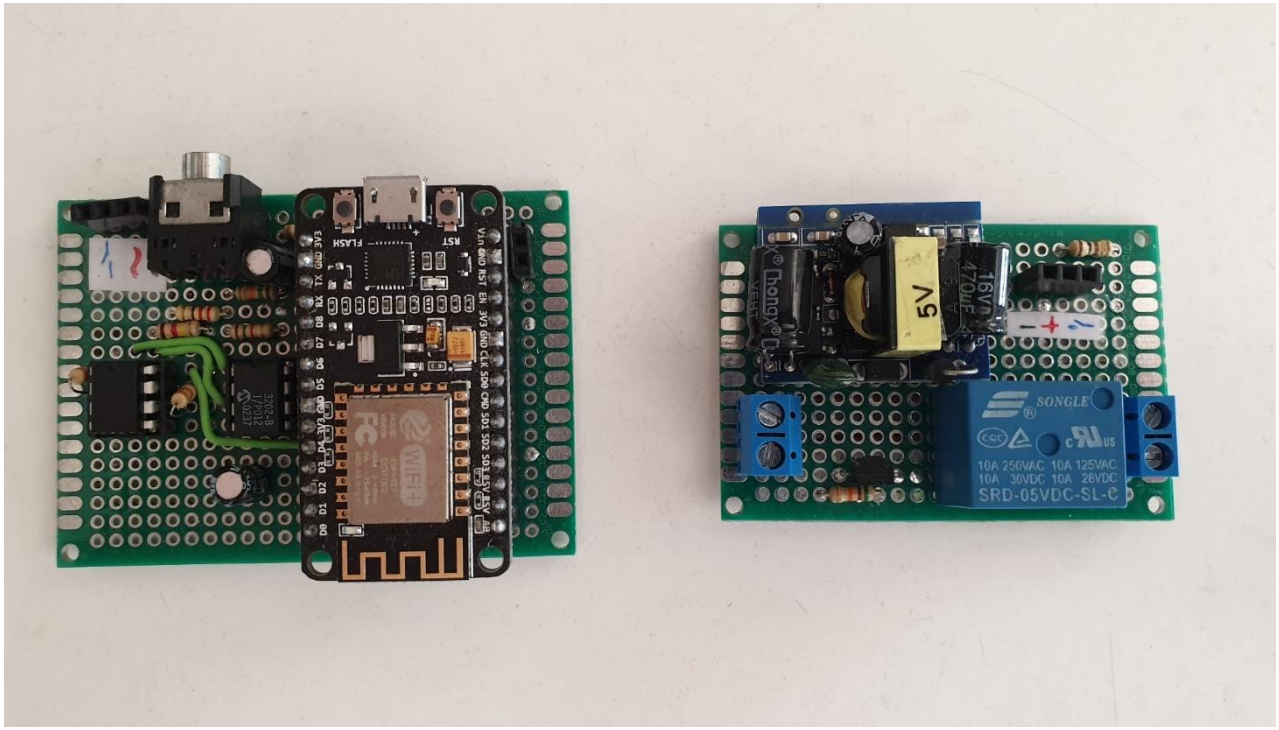

Figura 31: Fronte circuito: a sinistra, la scheda principale, a destra, la scheda di controllo abilitazione e di alimentazione.

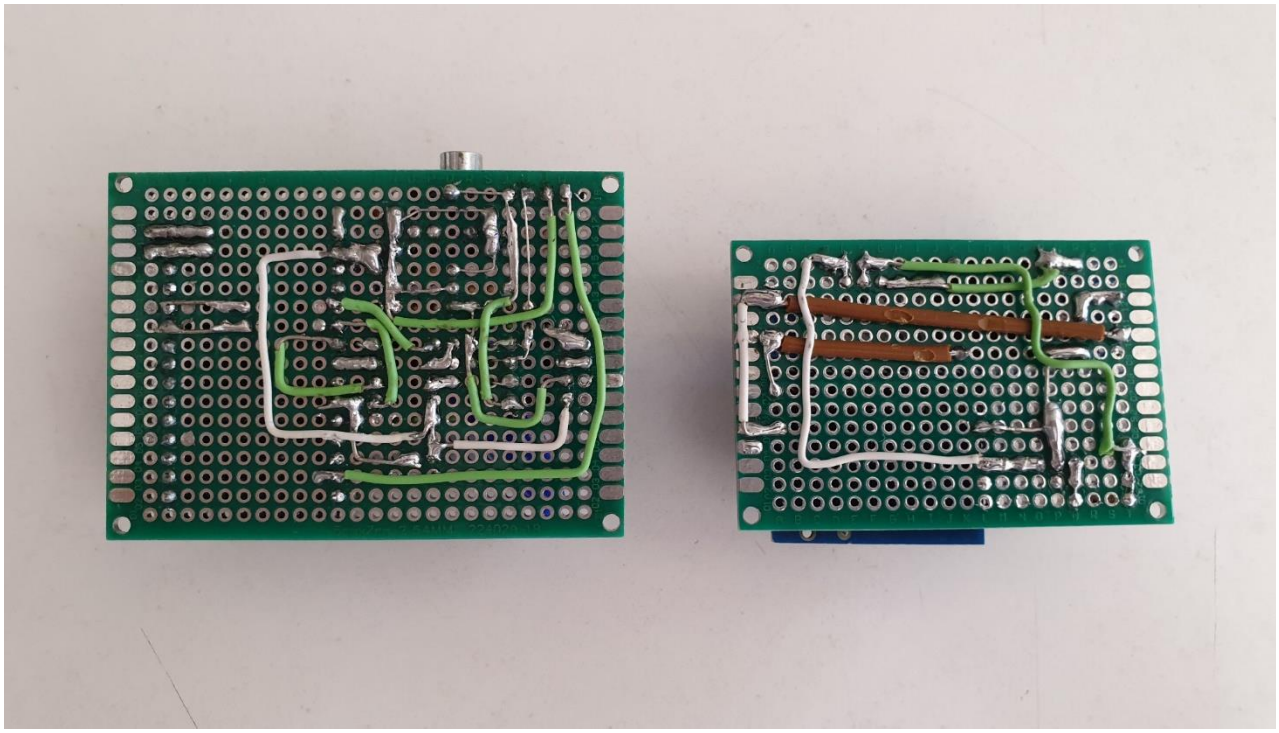

Figura 32: Retro circuito: a sinistra, la scheda principale, a destra, la scheda di controllo abilitazione e di alimentazione.

5.2.2. Listino componenti

- Resistenze da ½W
  - 1 x 100Ω
  - 2 x 15kΩ
  - 2 x 39kΩ
  - 1 x 51kΩ
  - NB: Per il calcolo di R_3, R_4 ed R_5, fare riferimento al capitolo 3

- Condensatori elettrolitici
  - 2 x 10µF
  - 1 x 100µF
  - 1 x 470µF

- 1N4007 o generico diodo raddrizzatore
- 2N7000 N-MOSFET
- Relè generico alimentato a 5V
- LM358 Op Amp
- MCP3202 ADC
- NodeMCU 1.0 con ESP-12E o ESP12-F
- Modulo AC-DC 5V per alimentazione della scheda.
- 2 x Screw terminal per il collegamento dell'ingresso e dell'uscita.
- SCT013-000
- Generico trasformatore abbassatore della tensione di rete
- Audio jack generico a tre poli per il collegamento dell'SCT013-000.
- 2 x 15 pin female stripline per la creazione di un socket sulla basetta millefori sopra il quale montare la NodeMCU.
- Una grande basetta millefori o due basette più piccole.
- Filo rigido (ad esempio, di doppino telefonico) per la creazione delle piste.

5.3. Flash del firmware e del filesystem LittleFS sull'ESP8266

Il firmware ed il filesystem del dispositivo vanno compilati e caricati per la prima volta nell'ESP8266 tramite la connessione diretta della NodeMCU al PC.

Per la compilazione del firmware, bisogna per prima cosa scaricare l'IDE VS Code, per poi installare dentro di essa l'estensione PlatformIO.

Per poter supportare il microcontrollore ESP8266, deve essere prima installata l'intera piattaforma di sviluppo relativa a questa famiglia di microcontrollori; questo può essere fatto da

`Home PlatformIO > Platforms > Embedded > Espressif 8266`

Al termine dell'installazione dei software, sarà necessario scaricare i file sorgenti del firmware da qui.

Una volta importato il progetto in PlatformIO all'interno di VS Code, si dovrà specificare la porta seriale su USB all'interno del file `platformio.ini`: quindi aprire il file e cambiare la riga `upload_port = COM6` nella porta fisica alla quale corrisponde effettivamente la NodeMCU una volta collegata al PC.

Per la compilazione ed il caricamento del firmware, basterà cliccare in basso a destra il pulsante `PlatformIO: Upload` ed attendere il processo di caricamento.

Figura 33: Pulsante di upload del firmware.

Una volta caricato il firmware, bisognerà anche caricare l'immagine binaria di LittleFS tramite

`Icona PlatformIO > Cartella Platform > Upload Filesystem image`

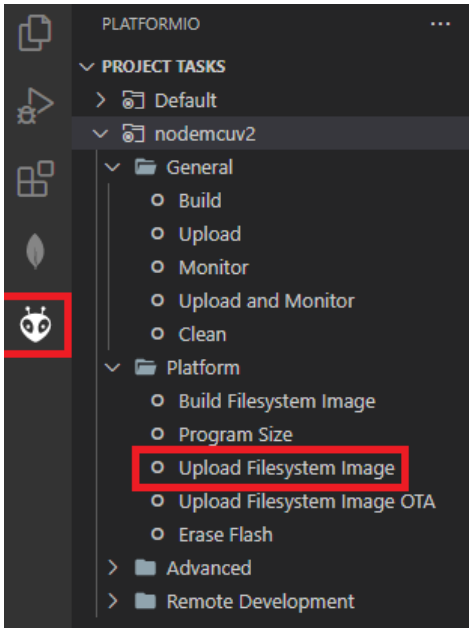

Figura 34: Procedura di caricamento dell'immagine binaria del filesystem.

5.4. Installazione di MongoDB

Il progetto PowerMonitor utilizza la versione 4.4 di MongoDB server installabile servendoci del gestore di pacchetti `apt`.

Per la giusta configurazione di `apt`, si rimanda il lettore alla documentazione ufficiale, in quanto la procedura di installazione cambia in base alla versione di Linux installata sulla macchina server.

Per ragioni legate alla sicurezza, è fortemente consigliata l'installazione del database nella stessa macchina dove in seguito verrà installato anche Node.js, in quanto il database, una volta installato, sarà in ascolto solamente all'indirizzo `localhost`.

È tuttavia possibile eseguire l'installazione su un'altra macchina in rete locale atta a gestire solamente il database tramite la modifica della stringa `bindIp: 127.0.0.1` nel file di configurazione di MongoDB `/etc/mongod.conf`; è necessario che la macchina in questione non sia esposta all'esterno della rete locale, in quanto ogni procedura di autenticazione prevista dal database, sarà di default disattivata.

Infine, è fortemente sconsigliato l'utilizzo di un database fuori dalla rete locale perché in questa versione dell'app PowerMonitor, non sono in alcun modo utilizzate le procedure di autenticazione al database previste dal driver Mongoose.

5.5. Installazione di Mosquitto

L'installazione di Mosquitto avviene tramite il gestore di pacchetti apt con il comando `apt install mosquitto`; una volta installato il pacchetto, deve anche essere abilitata la sua esecuzione all'avvio della macchina tramite il comando `sudo systemctl enable mosquitto.service`.

Come discusso nei capitoli precedenti, l'applicativo client MQTT da browser, per potersi connettere al broker MQTT Mosquitto, avrà bisogno che questo supporti la modalità di connessione MQTT tramite WebSocket.

Per poter abilitare la suddetta modalità, basterà creare un nuovo file di configurazione sotto `/etc/mosquitto/conf.d/mosquitto.conf` con il seguente contenuto:

```
# this will listen for mqtt on tcp
listener 1883
allow_anonymous true

# this will expect websockets connections
listener 9001
protocol websockets
allow_anonymous true
```

Queste configurazioni permetteranno al broker MQTT di ascoltare sia sulla porta MQTT standard 1883 per le normali connessioni, sia sulla porta 9001 per le connessioni MQTT su WebSocket.

Una volta scritto il file, il processo di Mosquitto deve essere riavviato per poter caricare le nuove configurazioni tramite il comando

```
sudo systemctl restart mosquitto.service
```

Il broker Mosquitto così configurato è ora pronto all'uso; al fine di fare debug sull'applicativo in via di sviluppo, sono stati installati ed utilizzati anche due comandi appartenenti al pacchetto `mosquitto-clients`: `mosquitto_pub` e `mosquitto_sub`.

Questi due applicativi permettono rispettivamente di far assumere alla shell che li invoca, il ruolo di publisher e di subscriber MQTT.

5.6. Installazione di NVM e Node.js

Per l'installazione di Node.js, si è scelto di utilizzare un NVM o Node Version Manager. Tramite l'utilizzo di questo applicativo, è resa possibile l'installazione di molteplici versioni di Node.js, ognuna con le proprie configurazioni ed i propri moduli globali.

Per l'installazione di nvm, basterà eseguire il seguente script da GitHub

```
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
```

e riavviare la shell che si sta utilizzando.

Se, dopo aver riavviato la shell, il comando `command -v nvm` stampa la stringa "nvm", allora lo script è andato a buon fine; se invece non viene stampato niente, si dovrà fare riferimento alla sezione di troubleshooting della documentazione ufficiale di nvm per poter risolvere il problema.

A questo punto, l'installazione di nvm è completata e, per poter installare npm e Node.js, basterà eseguire il comando `nvm install node`, che ne installerà la versione più recente.

Nello specifico, il progetto PowerMonitor è stato implementato utilizzando la versione LTS fermium (14.17.6) di Node.js; questa è installabile tramite il comando `nvm install lts/fermium`.

5.7. Configurazione della web app e del dispositivo

Una volta installati npm e Node.js, si dovrà scaricare l'app da qui per poi estrarla in una cartella e, successivamente, servirà installare anche tutti i suoi moduli accedendo alla cartella appena estratta con una shell per poi digitare il comando `npm install`.

Prima di poter avviare la web app, occorre cambiare l'indirizzo di connessione al database editando il file `/data/connection.json`.

Per la configurazione del dispositivo PowerMonitor fisico, dopo aver compilato e caricato il firmware ed il filesystem (come spiegato precedentemente), basterà inserire la NodeMCU nel suo socket per poi dare tensione.

Una volta eseguito il primo avvio, bisognerà configurare le impostazioni WiFi tramite un altro dispositivo dotato di WiFi e web browser, come uno smartphone.

La NodeMCU andrà in modalità di configurazione, aprendo così un hotspot WiFi con SSID pari a `PowerMonitor`, quando il suo LED integrato smetterà di lampeggiare e si illuminerà in maniera fissa.

A questo punto, basterà connettersi alla rete WiFi `PowerMonitor` per poi recarsi alla pagina web del portale all'indirizzo `http://192.168.4.1`.

Successivamente, basterà inserire il nome da assegnare al dispositivo e le configurazioni della propria rete WiFi nei campi predisposti per poi cliccare sul pulsante `Connetti`.

Infine, dopo il riavvio automatico del dispositivo, se la procedura di connessione è andata a buon fine, il LED integrato rimarrà spento.

5.8. Avvio del progetto

Una volta terminata la procedura di configurazione del dispositivo, si potrà avviare la web app tramite il comando `npm start`; ad avvio riuscito, recandosi all'indirizzo http://localhost:3000/dispositivi, sarà possibile avviare una nuova ricerca dispositivi.

Alla fine della procedura di ricerca, sarà trovato in rete il dispositivo con il nome prima assegnato: procedere registrandolo nel sistema e selezionandolo tramite l'apposito interruttore.

Da questo momento in poi, quando l'app sarà in esecuzione, i dati dei dispositivi registrati verranno salvati nel database ed inoltre, si potranno visualizzare i consumi in real-time del dispositivo selezionato tramite la dashboard ed i consumi salvati, sempre del dispositivo selezionato, tramite la scheda consumi.

6. Conclusioni

PowerMonitor rappresenta un'ottima soluzione per un'infrastruttura open-source completa per il monitoring dei consumi energetici a livello domestico.

Sicuramente, l'alternativa più complessa formata dalla suite Home Assistant combinata con il driver ESPHome, riesce ad offrire funzionalità ed integrazione su un livello sicuramente superiore (si pensi ad esempio, alla possibilità di installare la suite Home Assistant su dispositivi a basso consumo energetico basati su architettura ARM, come ad esempio un Raspberry Pi), ma l'intero progetto è nato con la premessa di essere interamente home made e svincolato da altre grandi suite software.

Il progetto è inoltre facilmente riproducibile e adattabile alle più svariate esigenze, grazie alla possibilità di modificare con facilità le varie configurazioni dei vari attori hardware e software presenti nell'intera suite.

6.1. Miglioramenti

I miglioramenti che in futuro potrebbero essere apportati all'infrastruttura del progetto sono molteplici, alcuni dei quali sono:

- L'integrazione con Amazon Alexa.
- La possibilità di aggiungere al dispositivo fisico un piccolo display OLED o LCD, per poter stampare i dati letti in real-time.
- L'inserimento di pulsanti fisici per il controllo dell'uscita.
- La progettazione di un PCB interamente composto da componenti THT facili da assemblare.
- La progettazione di un alloggiamento da poter stampare tramite una stampante 3D.

6.2. Ringraziamenti

Vorrei ringraziare innanzitutto i miei relatori, il ***Prof. Federico Fausto Santoro*** ed il ***Prof. Corrado Santoro***, entrambi professori di due stupende materie, che mi hanno permesso di scrivere una tesi così particolareggiata in bilico tra il mondo dell'informatica e quello dell'elettronica.

Sono molto grato ad entrambi, innanzitutto per la disponibilità che mi è stata concessa per poter appagare ogni mio dubbio relativo a questo e ad altri piccoli progetti passati.

Ringrazio inoltre i miei relatori per avermi sostenuto, incoraggiato ed avermi consigliato soluzioni e punti di vista alternativi per la realizzazione delle varie sezioni di questa tesi.

Tengo inoltre molto a ringraziare il ***Prof. Filippo Milotta*** ed il ***Prof Filippo Stanco***¸ per avermi sempre ispirato in merito a temi riguardanti la multimedialità, come la digitalizzazione, il ***Prof. Salvatore Antonio Riccobene*** ed il ***Prof. Giuseppe Pappalardo*** per avermi sempre spinto ad informarmi ben oltre gli argomenti trattati a lezione e la ***Prof.ssa. Cristina Milazzo***, il ***Prof Giuseppe Di Fazio*** ed il ***Prof. Fabio Raciti***, per avermi insegnato le basi dell'algebra lineare e dell'analisi matematica, nozioni che si sono rivelate d'importanza decisiva ai fini dello sviluppo della teoria alla base dell'intero progetto.

Vorrei ringraziare anche il ***Prof Daniele Francesco Santamaria***, per avermi fornito le basi per una corretta stesura di articoli accademici, che mi sono tornate utili per la compilazione di questa stessa tesi e di altra documentazione legata anche ad altri progetti.

Ringrazio inoltre i miei ex professori di scuola superiore: il ***Prof. Sebastiano Tropea***, il ***Prof. Michele Gricone*** ed il ***Prof. Giuseppe Arcoria***, i quali mi hanno fornito tutte le basi necessarie per poter andare avanti nello studio dell'informatica.

Ringrazio i miei colleghi di corso ***Aldo Fiorito***, ***Sergio Maccarrone***, ***Simone Scionti***, ***Francesco Petrosino***, ***Alessio Dinatale***, ***Giorgio Privitera*** ed ***Ottavio Castiglione***.

Più sul personale, vorrei ringraziare i miei ***genitori*** e la mia intera ***famiglia***, che mi sono sempre stati vicini e mi hanno sempre supportato nei momenti di dubbio e di confusione della mia vita.

Ringrazio anche tutti i miei ***amici*** e ***familiari*** che sono sempre stati presenti nel corso della mia carriera accademica, scolastica, nonché della mia vita: ***Marco Scalisi***, ***Alessio Maravigna***, ***Riccardo Di Mauro***, ***Luca Innocenti***, ***Claudio Cinardi***, ***Monica Pernice***, ***Francesco Leoni***, ***JeanMarc Anarratone***, ***Sebastiano Battaglia***, ***Enzo Battaglia***, ***Alberto Avola*** ed ***Andrea Ferrarotto***.

Vorrei inoltre ringraziare il lettore che, con molta pazienza, è riuscito ad arrivare fino alla fine di questa mia tesi. Grazie.

Davide Scalisi