Progetto PCTO TERMO-CARE

ITIS Planck corso: Telecomunicazioni – Classe: 4DIT Periodo Stage: dal 22–05-23 al 17-06-23

Presso: IMET S.r.l. Via Ronche, 93 33077 Sacile (PN)

|  |  |  |
| --- | --- | --- |
| AUTORI | | |
| Bertazzon Marcoelia Foto  o Firma | Cadorin Alan Foto  o Firma | Gallon Davide Foto  o Firma |

Sommario

[Premessa 4](#_bookmark0)

[Scopo 4](#_bookmark1)

[ISO9001 5](#_bookmark2)

[Il processo dell’ISO 5](#_bookmark3)

[La fase preliminare 5](#_bookmark4)

[La fase di sviluppo 6](#_bookmark5)

[La fase di produzione 7](#_bookmark6)

[Il nostro processo 7](#_bookmark7)

[La fase preliminare 7](#_bookmark8)

[La fase di sviluppo 7](#_bookmark9)

[La fase di produzione 8](#_bookmark10)

[Specifiche 8](#_bookmark11)

[Ricerca e Studio Fattibilità 8](#_bookmark12)

[Sensori di Temperatura 8](#_bookmark13)

[Uscita Relè 9](#_bookmark14)

[Verifica stabilità del sensore 9](#_bookmark15)

[Interferenze 9](#_bookmark16)

[Filtro software 10](#_bookmark17)

[Tick per secondo e oscilloscopio 10](#_bookmark18)

[Interrupt per la cattura delle informazioni 11](#_bookmark19)

[Altri metodi di conversione 11](#_bookmark20)

[HARDWARE 12](#_bookmark21)

[PCB (Printed Circuit Board) 12](#_bookmark22)

[Processi per la realizzazione di un PCB 12](#_bookmark23)

[La fattibilità 12](#_bookmark24)

[Definizione di architettura 12](#_bookmark25)

[Progettazione dello schema elettrico 13](#_bookmark26)

[Progettazione del PCB 13](#_bookmark27)

[L’ordine del PCB 14](#_bookmark28)

[I file gerber 14](#_bookmark29)

[La BOM 14](#_bookmark30)

[Il file Pick-and-Place 15](#_bookmark31)

[L’ordine del PCB 15](#_bookmark32)

[Pick and place 15](#_bookmark33)

[FIRMWARE 16](#_bookmark34)

[Cos’è il Firmware 16](#_bookmark35)

[Cross-compilazione 16](#_bookmark36)

[Passaggi per creare il codice macchina 16](#_bookmark37)

[Familiarizzazione e primo Firmware 17](#_bookmark38)

[Circuito di conversione 17](#_bookmark39)

[Programmazione scheda (ESP32) 17](#_bookmark40)

[RETI 19](#_bookmark41)

[Protocolli ISO/OSI 19](#_bookmark42)

[Protocollo TCP/IP 20](#_bookmark43)

[MQTT 21](#_bookmark44)

[QoS 21](#_bookmark45)

[Confronto dei vari protocolli 22](#_bookmark46)

[DHCP 24](#_bookmark47)

[IOT 24](#_bookmark48)

[SOFTWARE 25](#_bookmark49)

[Cosa è un software 25](#_bookmark50)

[Linguaggio di programmazione 25](#_bookmark51)

[Flutter 26](#_bookmark52)

[Implementazione delle classi 27](#_bookmark53)

[Ciclo di vita di un stateful widget 28](#_bookmark54)

[Grafica dell’applicazione 29](#_bookmark55)

[Comunicazione col broker 29](#_bookmark56)

[Aggiornamento grafica dell’app 29](#_bookmark57)

[Creazione degli APK 30](#_bookmark58)

[Risultato finale 30](#_bookmark59)

[IMPLEMENTI FUTURI 30](#_bookmark60)

[Inverter 30](#_bookmark61)

[AI 30](#_bookmark62)

Relazione PCTO v1.0.0.docx

# Premessa

Durante lo stage saremo in collaborazione con il reparto di progettazione di IMET, che ci supporterà sia per lo sviluppo del progetto che per la formazione.

Il progetto PCTO lo abbiamo chiamato TERMO-CARE perché è un nome che richiama subito l’attenzione sullo scopo che volevamo raggiungere, ovvero quello di un termostato “intelligente” con l’obbiettivo del benessere della casa e degli utenti di essa.

# Scopo

La formazione per il progetto PCTO è suddivisa nelle seguenti fasi:

* **Formazione Sicurezza**: Conoscenza dei rischi a cui si viene sottoposti nel luogo di lavoro e il come comportarsi di fronte a essi.
* **ISO9001**: Imparare a strutturare il progetto per fasi e processi seguendo una procedura (linea guida) e verificando man mano lo stato dello sviluppo.
* **HW**: stesura delle specifiche, studio fattibilità, scelta componenti, schema elettrico, circuito stampato
* **FW**: Ambiente di sviluppo IDE, plugin, librerie, linguaggio di programmazione, tecniche specifiche, verifiche strumentali
* **Rete**: protocolli di rete, WiFi, TCP/IP, MQTT
* **SW**: Sviluppo applicazione per smartphone, linguaggio di programmazione

In generale per tutte le fasi vengono stimolate competenze per il lavoro in team con particolare attenzione al problem solving.

Lo scopo è di realizzare un termostato smart diviso in:

* Una scheda a microcontrollore che legge la temperatura la trasmette in cloud
* Una scheda a microcontrollore che si collega in cloud e attiva una o più uscite in base alle regole programmate
* Realizzare un applicativo multipiattaforma per PC e Smartphone che consente di visualizzare e modificare i paramenti delle regole del termostato

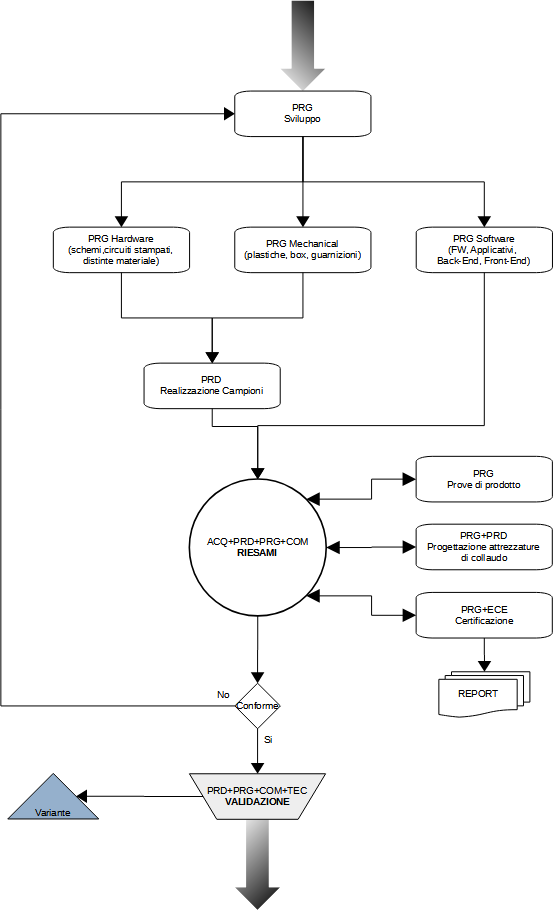
# ISO9001

L’ISO 9001 è uno standard creato che stabilisce i criteri per un sistema di gestione della qualità e si concentra sull'efficacia dei processi, con lo scopo di aiutare a lavorare in modo più efficiente e a ridurre gli errori nei prodotti.

## Il processo dell’ISO

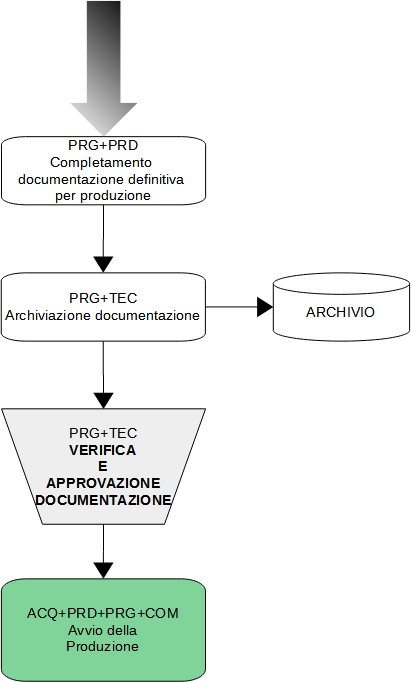
## La fase preliminare

In questa prima fase vi è un’attenta analisi che serve a comprendere le potenzialità e il possibile successo del nostro prodotto in base alla richiesta del mercato o alla necessità della riduzione dei costi. Parte tutto da un’idea, che viene sottoposta a quesiti di fattibilità e a delle ricerche a cui consegue un report, che può portare o all’inizio effettivo del nostro progetto, oppure alla chiusura di questo.

La fase di sviluppo

Lo sviluppo prevede il lavoro in parallelo di tre organi della nostra azienda, l’elaborazione del firmware e applicativi, lo sviluppo dei supporti meccanici del nostro prodotto e la parte di circuiti elettrici (le ultime due saranno sottoposte anche a test con dei campioni).

Successivamente a una serie di riesami e accertamenti nasce la necessità delle certificazioni, come può essere ad esempio la certificazione sul rispetto delle norme sul campo elettromagnetico, e una volta ottenute queste si esegue un ulteriore report.

La fase di produzione

Dopo aver inserito negli archivi le caratteristiche del nostro prodotto si fa un’ultima verifica e si può passare alla produzione in massa del prodotto.

## Il nostro processo

La fase preliminare

Nel nostro caso l’idea non è nata allo scopo di abbattere i costi o per la produzione di un nuovo prodotto, ma allo scopo prevalentemente didattico. La prima parte a cui abbiamo fatto fronte è stata la scelta dei componenti, noi abbiamo selezionato: un ESP32 perché imposto a livello progettuale, un relè, a cui è abbinato un diodo per l’eliminazione di correnti parassite, allo scopo di pilotare poiché risulta un contatto più pulito e alimentato a 5V perché è quanto ci fornisce la scheda, la sonda è un NTC perché ha una buona sensibilità e è più economico rispetto alle altre opzioni che avevamo (LM) e un transistor NPN allo scopo di interruttore.

La fase di sviluppo

Per quanto ci riguarda noi abbiamo sviluppato prima un circuito con tutti i componenti scelti nella fase precedente e verificato il suo corretto funzionamento, arrivando poi a scrivere un firmware su misura per le nostre necessità. Poi ci siamo posti il problema delle interferenze magnetiche, che

rientra nelle certificazioni, risolvendolo con l’aggiunta di un condensatore a livello fisico e di un filtraggio via software.

La fase di produzione

Argomento a cui noi non dovremmo far fronte.

# Specifiche

* Ci è stato imposto l’utilizzo di un una scheda **ESP32-DevKitC-32UE**, questo è un microcontrollore di nuova generazione sviluppato dalla Espressif Systems, è una scheda valida per progetti di elettronica e di automazione, è adatta per progetti che richiedono connessioni WiFi o Bluetooth. Noi la utilizziamo perché è adatta ad un uso amatoriale.
* Valutazione temperatura sensori di temperatura: LMT84LPM o NRMR104F3435B2F0061
* Ottenere una precisione della temperatura di 0,1 °C
* Valutazione comandi di uscite con relè
* Tempo di reattività del sistema: 1s
* Applicativo PC/Smartphone per la configurazione e report dei dati: monitoraggio temperatura e modifica set point con sensibilità di 0,5 °C

# Ricerca e Studio Fattibilità

## Sensori di Temperatura

Si riporta in tabella i pro e contro delle tipologie di sensori che si trovano nel mercato:

|  |  |  |
| --- | --- | --- |
| **Tipologia** | ✔ **Pro** | X **Contro** |
| Termocoppia | Elevato range  Ideale per ambienti industriali | Scarsa sensibilità  Circuito costoso perché richiede misurazione temperatura giunto  freddo |
| PTC | Alimentabile con poca energia Pochi errori di lettura | Elevati costi  Auto riscaldamento |
| NTC | Accurato Economico | Non lineare Scarso range |
| DHT22 (Esempio di integrato con uscita  digitale) | Basso consumo energetico  L’uscita è già digitale | Prezzo |

I sensori forniti da IMET che andremmo a provare sono:

* LMT84LPM
* NRMR104F3434B2F

*APPROFONDIMENTO*

*La ricerca dei componenti e delle loro proprietà di mercato come disponibilità e prezzo*

*l’abbiamo effettuata con* [*www.mouser.it*](http://www.mouser.it/) *. Questo catalogo online ci permette di avere dei prezzi anche per piccole produzioni di 100/1000 pz, ovviamente nel web ne esistono altri come:* [*https://it.rs-online.com*](https://it.rs-online.com/) *,* [*https://it.farnell.com*](https://it.farnell.com/) *,* [*https://www.digikey.it*](https://www.digikey.it/)

## Uscita Relè

Abbiamo scelto di utilizzare un relè perché fornisce un contatto pulito con un isolamento adeguato per alimentare rete 230Vac.

Per alimentare il relè abbiamo deciso di inserire un transistor collegato a massa pilotato tramite un pin dell’ESP, tra il pin e il transistor abbiamo deciso di inserire una resistenza di 1KΩ per limitare la corrente in ingresso di esso.

Inoltre abbiamo stagnato un diodo di ricircolo alla bobina del relè che ha la funzione di raddrizzatore della tensione per evitare che vengano portate le tensioni a livelli critici sul transistor danneggiandolo. Il relè si attiva solo se viene alimentato da una tensione corrispondente ad almeno il 70% della sua alimentazione standard. Abbiamo deciso di utilizzare un relè da 5V perché possiamo alimentarlo direttamente da un pin della scheda a differenza di quello a 12V che per alimentarlo avremmo dovuto progettare un amplificatore o aggiungere una batteria (soluzione più costosa).

## Verifica stabilità del sensore

Per verificare che la conversione in temperature del nostro software sia corretta, abbiamo letto nel datasheet quale fosse la temperatura corrispondente a un’uscita resistiva di 10K, ovvero 25 °C.

Poi abbiamo controllato se la conversione fosse corretta sostituendo con una resistenza il sensore, ottenendo una temperatura attorno a 24,5 °C, che è apprezzabile in quanto sono presenti delle incertezze di: resistenze, uscita uC e conversione.

## Interferenze

Dopo aver progettato il circuito abbiamo fatto dei test di stabilità, verificando la variazione di temperatura segnata dal sensore applicandogli un disturbo di un walky-takie con una potenza di circa 5W. Applicando il disturbo si otteneva una differenza di ben 10°C perciò per rimediare abbiamo cercato quali potrebbero essere i modi più efficaci e meno costosi per eliminare le EMI (es: la schermatura costa tanto e non si può schermare il sensore perché deve leggere la temperatura esterna). Quindi dopo un po’ di ricerche abbiamo deciso di usare un filtro per rimuovere il rumore di fondo. (PROGETTAZIONE DEL FILTRO) Siccome lavoriamo in corrente continua abbiamo bisogno di rimuovere tutte le frequenze che possono creare rumore, ovvero quelle superiori alla nostra. Quindi usiamo un filtro passa basso aggiungendo un condensatore da 100nF in parallelo alla nostra resistenza da 10KΩ usata nel circuito del sensore NTC. Questi filtri sono necessari per soddisfare delle caratteristiche necessarie per ottenere una certificazione sulle interferenze.

## Filtro software

La temperatura uscita dal sensore non è sempre perfettamente regolare, in quanto può variare leggermente e creare cosi un errore che in alcuni casi può essere un problema. Abbiamo cosi deciso di implementare un sistema di filtraggio via software che ci fornisca una temperatura più regolare e stabile.

Per fare ciò abbiamo analizzato diverse tattiche differenti tra cui:

### Filtro a media mobile semplice

Questo tipo di filtro calcola la media aritmetica dei campioni all'interno di una finestra di campionamento fissa. Ogni campione contribuisce in modo uguale al calcolo della media.

La finestra di campionamento può essere di dimensione fissa o variabile. Il filtro a media mobile semplice offre una risposta temporale liscia ma può causare un certo ritardo nella risposta.

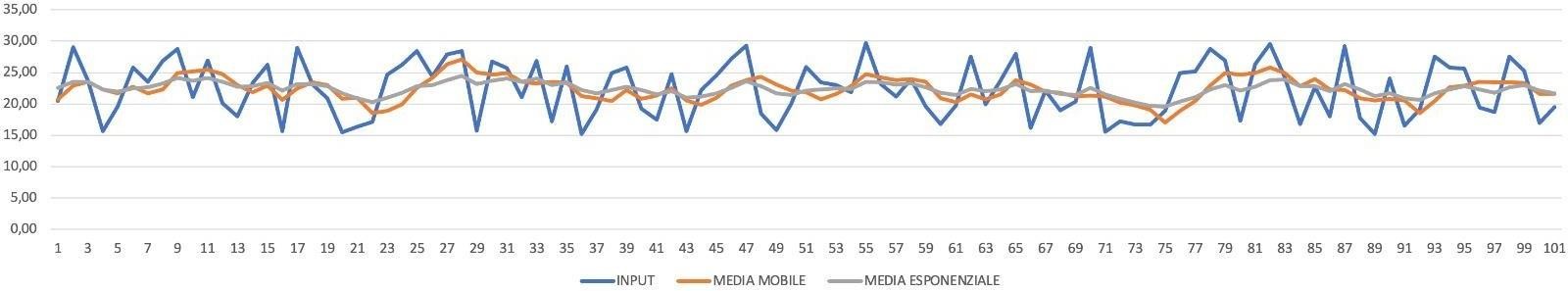
### Filtro a media mobile ponderata

Questo tipo di filtro calcola la media ponderata dei campioni all'interno di una finestra di campionamento. Ogni campione contribuisce al calcolo della media in base al suo peso associato. I pesi possono essere assegnati in base a una funzione predefinita o a criteri specifici. Il filtro a media mobile ponderata permette di assegnare un peso maggiore ai campioni più recenti, offrendo una risposta più rapida ai cambiamenti nel segnale.

### Filtro a media esponenziale

Questo tipo di filtro calcola una media ponderata dei campioni, ma utilizza una costante di smoothing per regolare l'influenza dei campioni precedenti. La costante di smoothing determina il peso relativo dei campioni passati e correnti nel calcolo dell'output filtrato.

Maggiore è la costante di smoothing, maggiore è l'influenza dei campioni passati e minore è la sensibilità alle variazioni rapide del segnale. Il filtro a media esponenziale offre una risposta rapida ai cambiamenti nel segnale e può essere implementato in modo efficiente.

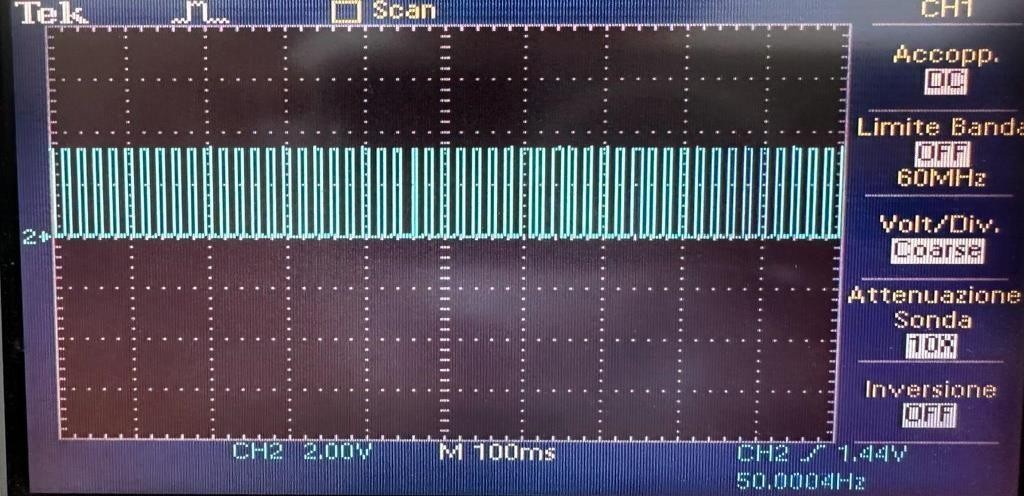
Dopo aver analizzato le varie tecniche tramite un grafico excel prendendo 100 valori casuali, abbiamo deciso di usare la media esponenziale perché leggermente più accurata rispetto alla media mobile (la ponderata darebbe dei pesi ai valori e quindi non si addice alle nostre circostanze.

Questo filtraggio, insieme a quello hardware prevede anche casi in cui ci sono interferenze con

l’alimentazione a 220V a 50/60hz.

## Tick per secondo e oscilloscopio

Per verificare che il nostro sistema funzionasse correttamente e non ci fossero problemi legati alla stabilità abbiamo usato un oscilloscopio collegato ad un pin che alterna il suo stato (ON/OFF) ad ogni ciclo della macchina a stati.



Siccome la lettura ACD (Analog Digital Converter) del microcontrollore non è sempre stabile e può presentare delle incertezze, perciò abbiamo deciso di usare una tecnica di oversampling, cioè misuriamo più volte, prendendo più campioni e facendo la media della somma delle misurazioni, riducendo l’incertezza e aumentando la stabilità.

Ci siamo accorti che calcolando la temperatura e la media ad ogni lettura del sensore utilizzavamo moltissime risorse del uP inutilmente, quindi abbiamo implementato una tecnica, nella quale utilizziamo una funzione che legge la tensione sul sensore ogni millisecondo e faccia la media dei valori ogni secondo convertendo poi il tutto in temperatura, questo per ridurre i calcoli con numeri in float che rendono il programma più pesante. Facendo così abbiamo ridotto di molto le risorse utilizzate dal uC e abbiamo reso le misure molto più precise.

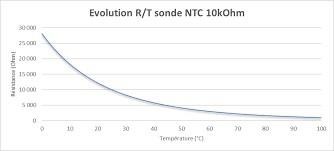
## Interrupt per la cattura delle informazioni

Siccome abbiamo bisogno di eseguire la lettura della temperatura in momenti determinati abbiamo deciso di creare degli interrupt per la lettura del sensore (PER STABILITA). Abbiamo riscosso la necessità dell’uso degli interrupts poiché nella prova con l’oscilloscopio abbiamo notato che ad intervalli irregolari risultavano dei “buchi” nelle nostre misurazioni, ovvero delle misurazioni mancate. L’interrupt, implementato con la funzione esp\_timer\_get\_time() (cronometro inizializzato ad ogni avvio della macchina), ci permette di eseguire la misurazione in un momento preciso e con alta priorità.

## Altri metodi di conversione

In alcune macchine meno potenti del ESP32 che usiamo potrebbe non essere presente il modulo ALU nel processore, rendendo più complicato fare calcoli, ad esempio per convertire la tensione dell’NTC in temperatura dovremmo usare metodi come:

* l’intersezione dei punti tramite un grafico



* linee di tendenza

Per quest’ultima procedura si devono seguire i seguenti passaggi:

1. **Raccogli i dati:** Misurare la temperatura utilizzando il sensore e annotare le letture del sensore insieme ai corrispondenti valori di temperatura noti.
2. **Analizza i dati:** Tracciare i punti dati in un grafico bidimensionale, con le letture del sensore sull'asse delle x e i valori di temperatura noti sull'asse delle y.
3. **Scegli il tipo di linea di tendenza:** In base alla distribuzione dei dati, scegliere il tipo di linea di tendenza che meglio si adatta al set di dati.
4. **Calcola la linea di tendenza:** Utilizzando strumenti come un software di foglio di calcolo o un'applicazione di analisi dati, calcolare la linea di tendenza utilizzando i punti dati. Questo darà un'equazione matematica che rappresenta la relazione tra le letture del sensore e i valori di temperatura.
5. **Applica la linea di tendenza:** Una volta ottenuta l'equazione della linea di tendenza, si utilizza per convertire le future letture del sensore in valori di temperatura. Basta inserire la lettura del sensore nell'equazione e risolvere per il valore di temperatura corrispondente.
6. **Valida i risultati:** Per assicurarti che la linea di tendenza sia accurata, confrontare i valori di temperatura calcolati utilizzando la linea di tendenza con valori di temperatura noti.

# HARDWARE

## PCB (Printed Circuit Board)

I PCB sono supporti composti da diversi materiali: il rame, che connette elettricamente i vari componenti, una fibra di vetro [fr4], avente funzione di isolante tra gli strati (che possono variare numericamente), il solder mask, è il materiale che ricopre quasi interamente scheda, ad eccezione dei footprint, isolandola evitando il contatto con polveri e l’ossidazione. Il footprint, (tradotto letteralmente dall’inglese come “impronta del piede”) è il simbolo che viene dato a ciascun componente e rappresenta il punto in cui verrà saldato sul pcb. Per far comunicare i livelli si usano dei vias che sono dei fori metallizzati utilizzati per far comunicare due o più strati.

## Processi per la realizzazione di un PCB:

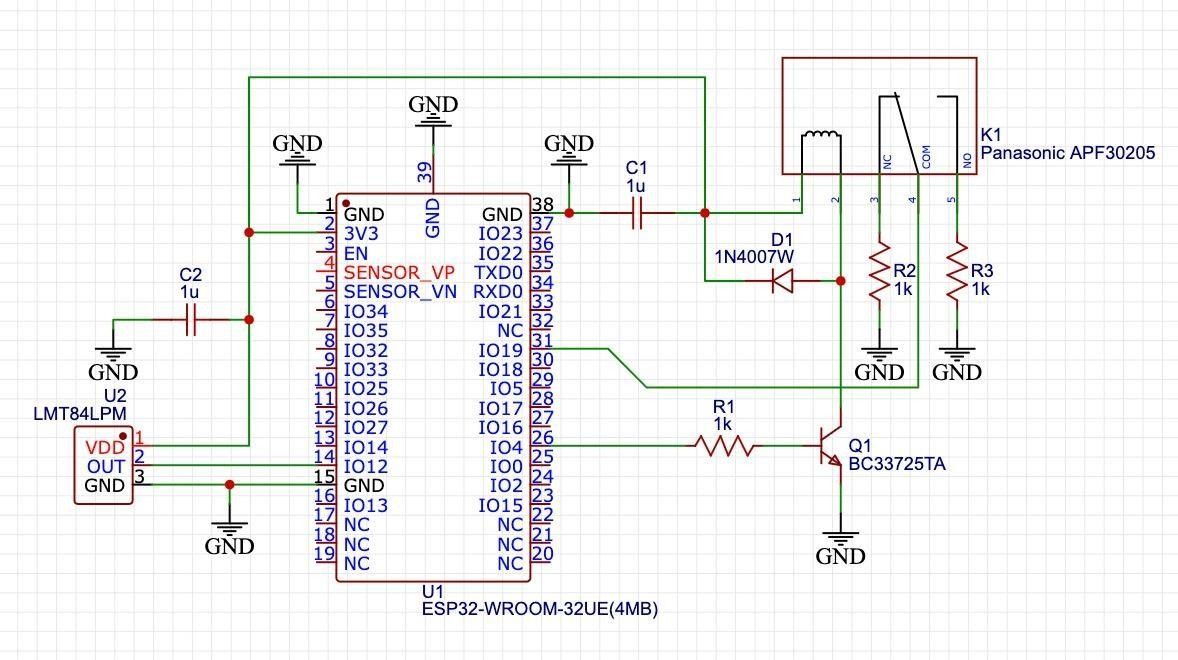
## La fattibilità

Prima di realizzare un progetto dobbiamo innanzitutto capire se l’idea che vogliamo realizzare è fattibile quindi la si sottopone a dei test o ad uno studio di fattibilità. Nel nostro caso il progetto risulta fattibile, questa conclusione la abbiamo tratta in seguito a un attento esame dei datasheet e delle caratteristiche fisiche e comportamentali dei nostri componenti.

## Definizione di architettura

Dopo aver verificato la fattibilità del progetto viene definita l’architettura del nostro PCB, ovvero il ruolo di ciascun componente per raggiungere il nostro scopo, dato che non è detto che esista un unico componente che faccia tutto quello di cui abbiamo bisogno.

## Progettazione dello schema elettrico



[Programma fatto con EasyEDA]

Nello schema elettrico vengono usati dei simboli per rappresentare i componenti reali ed essi sono collegati tra di loro tramite “fili” che rappresentano un collegamento elettrico.

## Progettazione del PCB

Sfruttando il progetto del circuito fatto in precedenza e collaudato, attraverso simulazioni e prototipi fatti su bread board. Abbiamo fatto il progetto del PCB, sviluppato in due livelli, il livello Top è identificato dal colore rosso, mentre il Bottom dal colore blu. A collegare il livello superiore con quello inferiore ci sono dei fori con un rivestimento metallico, i componenti sono collocati tutti nel layer superiore (nel nostro caso).

I componenti possono essere di due tipologie:

* SMD (Surface Mounted Devices):

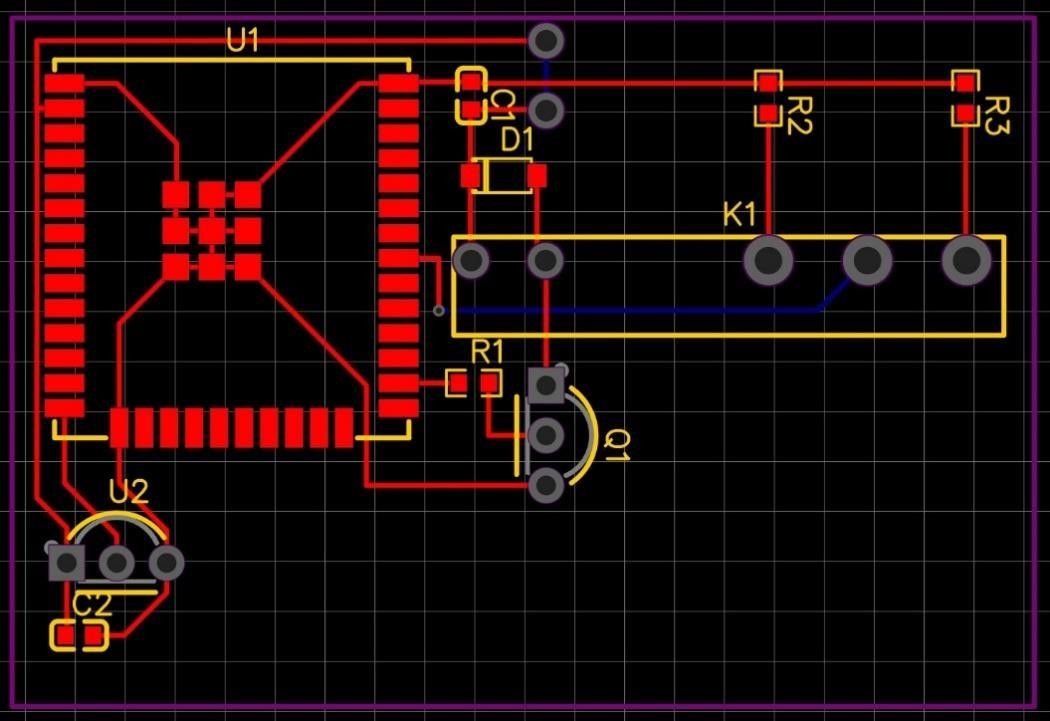
sono componenti elettronici montati direttamente sulla superficie di un circuito stampato.

* PTH (Pin Trought Hole):

sono componenti elettronici che vengono montati sulla scheda elettronica con fori passanti.

Il circuito stampato deve essere realizzabile (ad esempio che non necessiti di fori nell’ordine

dei nanometri), per verificarlo basta impostare le regole della realizzazione del design.



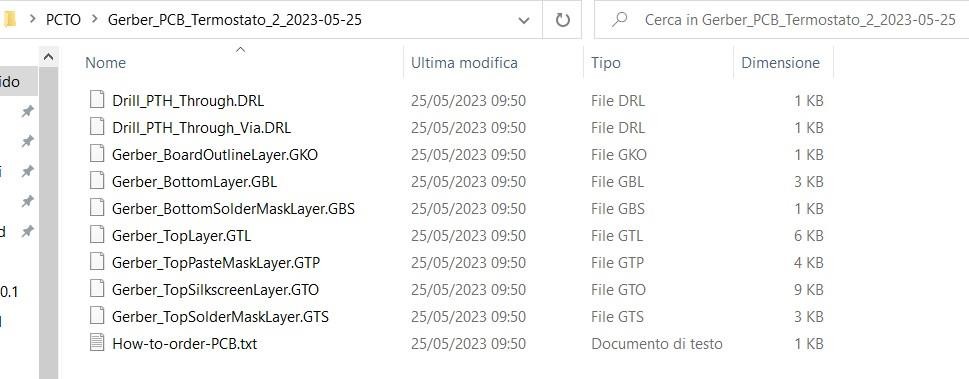
## L’ordine del PCB

Dopo aver progettato il PCB si generano i documenti per il circuito stampato. Il file da mandare a

un’ipotetica azienda addetta alla realizzazione della nostra scheda sono:

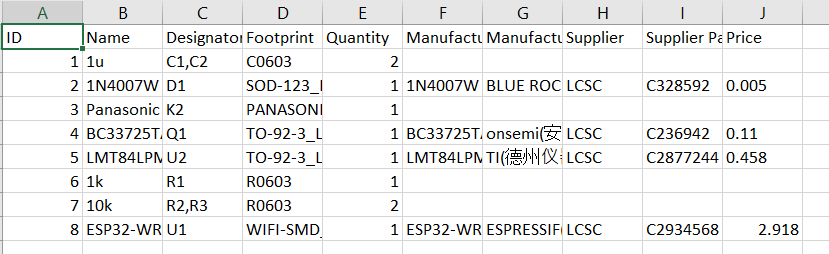
## I file gerber

Sono composti da documenti per ogni strato del pcb, questi strati tramite un programma specifico (come gerbV), possono essere visualizzati tutti insieme o selezionare solo i desiderati. Questo file è uno standard de-facto utilizzato per la produzione di circuiti stampati (PCB) per tracciare le connessioni elettriche quali piste, vias, e piazzole.



## La BOM

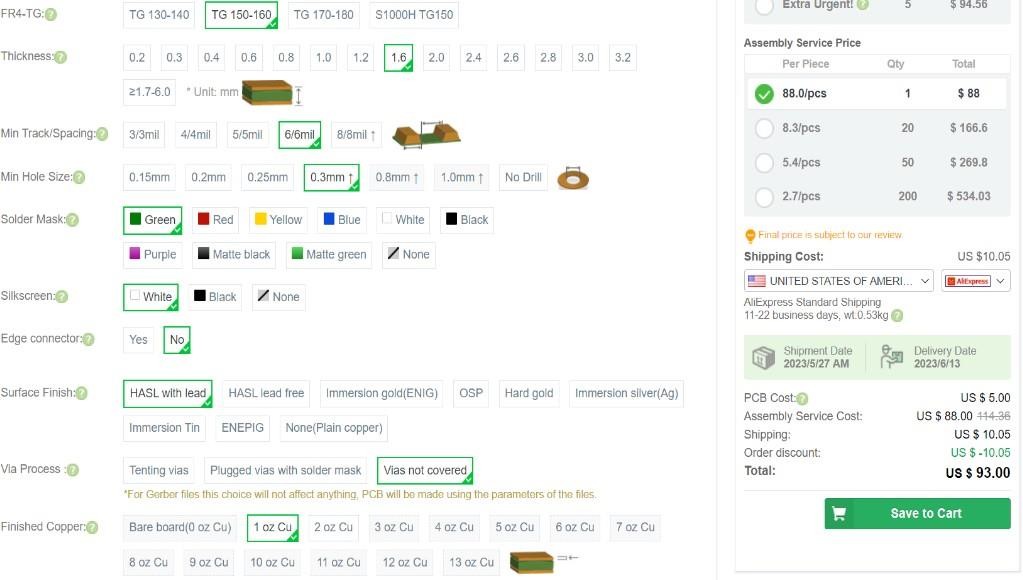
BOM = bill of material è lista dei componenti



## Il file Pick-and-Place

definisce le coordinate dei componenti nel PCB, questo serve al macchinario che posiziona i componenti nella scheda.

## L’ordine del PCB

Abbiamo provato a fare un acquisto di un PCB quindi abbiamo dovuto decidere diversi parametri.

L’ordine necessita della compilazione delle caratteristiche desiderate del nostro PCB, come può essere ad esempio il surface finish, ovvero il materiale di cui saranno fatte le finiture, si può scegliere tra oro, HASL ( hot air solder leveling). L’HASL è un processo dove il PCB viene tipicamente immerso in un bagno di stagno fusa in modo che tutte le superfici di rame esposte siano coperte dalla saldatura. La saldatura in eccesso viene rimossa facendo passare il PCB tra le lame ad aria calda.

## Pick and place

Ovvero il processo di montaggio dei componenti nella scheda:

Nella fase di assemblaggio del PCB i componenti vengono montanti tramite una saldatura con una pasta composta da stagno e flussante che funge da termo conduttore e permette una fusione più rapida. La brasatura è una tipologia di saldatura utilizzata nel campo dei pcb. I componenti già brasati e che hanno già fatto il passaggio in forno in precedenza, rimarranno saldati perché lo stagno si scioglie più lentamente della pasta, rendendo così possibile il passaggio in forno più volte per permettere al nostro pcb la saldatura dei componenti da entrambi i lati quando richiesto.

# FIRMWARE

## Cos’è il Firmware

Innanzitutto il termine *firmware* viene utilizzato per descrivere il software permanentemente programmato o incorporato in un dispositivo hardware. Esso è una forma di software che è stata scritta direttamente sulla memoria di sola lettura/scrittura del dispositivo durante il processo di produzione (non è stato il nostro caso visto che lo abbiamo programmato in post produzione). Essenzialmente il firmware è parte del dispositivo e viene eseguito ogni volta che il dispositivo viene acceso.

## Cross-compilazione

La cross-compilazione è il processo di compilazione di un programma su un sistema operativo o un'architettura diversa da quella su cui verrà eseguito il programma. Questo può essere utile quando si desidera compilare un programma per una piattaforma diversa da quella in cui si sta sviluppando, ad esempio compilare un'applicazione per dispositivi mobili su un computer desktop.

Passaggi per creare il codice macchina

Un qualsiasi programma scritto in un determinato linguaggio di programmazione passa attraverso delle fasi ben precise prima di potersi trasformare in file.exe in grado di essere eseguito per dare all’utente l’output, che non è altro che la risposta del programma in base agli input e agli algoritmi inseriti a priori.

Le fasi di per creare il codice eseguibile dalla macchina sono:

### Precompilazione:

La primissima fase è quella della precompilazione, attraverso un programma a sé stante chiamato appunto precompilatore. Essa consiste nella sostituzione automatica di parti di codice sorgente ad esempio nella dichiarazione #define PIGRECO 3.1415, Il precompilatore sostituisce automaticamente a tutte le occorrenze nel testo della parola PIGRECO il valore 3.1415.

### Compilazione -> Codice oggetto

La fase di compilazione è molto complessa ed è determinata da diversi sotto-passaggi che esegue un programma a sé stante chiamato appunto Compilatore. Essa consiste nella verifica di quello che è il codice sorgente e in particolare esistono 3 tipologie di analisi:

* 1. Analisi Lessicale, ovvero il Riconoscimento dei simboli
  2. Analisi Sintattica, ovvero il Riconoscimento delle istruzioni
  3. Analisi Semantica, ovvero il Significato delle istruzioni.

Il compilatore analizza se i simboli scritti all’interno del programma sono scritti in maniera corretta. Successivamente si passa all’analisi delle istruzioni e come ultima cosa si analizza, nonostante la grammatica di quanto scritto sia corretta, se ha senso l’intero corpo del programma, ovvero se tutto ha un senso logico.

### Linking -> Codice eseguibile

Una volta che tutte le verifiche hanno riscontrato un esito positivo, allora il codice sorgente di trasforma in codice oggetto (.obj).

### Caricamento -> Programma pronto per essere eseguito

Fase di caricamento in memoria RAM del codice oggetto

### Esecuzione -> Output del programma

esecuzione del file eseguibile

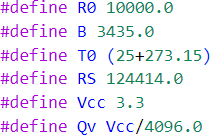
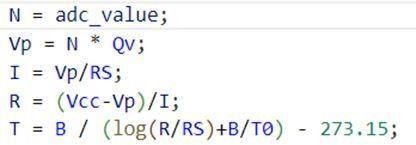
## Familiarizzazione e primo Firmware

Noi, appena abbiamo visto la scheda (ESP32-DevKitC-32UE), abbiamo deciso di prendere familiarità con essa facendo un semplice programma che stampasse “Hello World”. Ovviamente abbiamo dovuto installare un’estensione di Esspressif in VisualStudioCode per poter implementare il Firmware nella scheda. All’istante abbiamo riscontrato dei problemi, uno fra questi è stato il modo in cui stampava a terminale la nostra stringa. Abbiamo pensato che il problema fosse il baud-rate. Dopo parecchio tempo a cercare un possibile valore di baud-rate che ci soddisfacesse ci hanno informato che il problema risiedeva nella versione dell’estensione di Esspressif che avevamo installato in VisualStudioCode, infatti l’ultima versione rilasciata dell’estensione conteneva un bug che dava problemi a livello del terminale. Di conseguenza abbiamo provveduto a installare una versione precedente dell’estensione e tutto si è risolto.

## Circuito di conversione

Dopo aver realizzato il nostro primo Firmware abbiamo deciso di collegare una sonda di temperatura alla nostra scheda. Noi abbiamo avuto la possibilità di scegliere fra due sonde: l’NRMR104F3435B2F (NTC) e l’LMT84LPM (LM35). Dopo qualche ricerca abbiamo scelto di utilizzare l’NRMR104F3435B2F sostanzialmente per un fattore di costo del componente in ottica di una futura produzione del termostato. Dalle nostre ricerche abbiamo constatato che per utilizzare quel sensore avremmo dovuto fare una conversione visto che la sonda ha un’uscita resistiva. Innanzitutto abbiamo dovuto ricavare dal datasheet il valore della resistenza che gli occorreva (R25 = 10KΩ), il valore della costante per la conversione (B = 3435) e non meno importante il campo di misura della sonda (-40 °C, 125 °C). Infine ci siamo calcolati il valore di RS (media fra R(-40 °C) e R(125 °C), esempio: R( -40°C) = R25 \* e B(1/T - 1/To) = 10K \* e3435(1/233.15 - 1/298.15) = 248276.5 Ω).

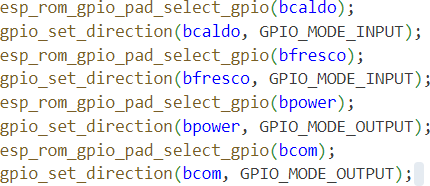
Esempio del codice di inizializzazione/conversione:

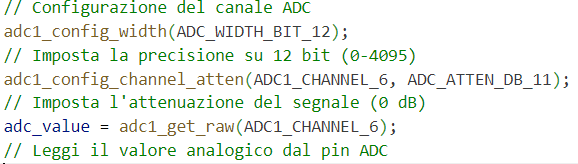
## Programmazione scheda (ESP32)

La scheda che ci è stata fornita può essere programmata mediante diversi linguaggi, tra cui possiamo trovare: Arduino, C e Python. Noi inizialmente avevamo optato per la programmazione in codice Arduino, giungendo poi alla conclusione che non si sarebbe potuto proseguire per quella via, in quanto per implementare la comunicazione mediante protocollo MQTT era necessario l’utilizzo del linguaggio C.

Riportiamo in seguito alcune righe di codice che operano con la parte fisica della scheda:

* 1. Questa sezione di codice ha lo scopo di assegnare a ciascuna variabile il suo ruolo (INPUT o OUTPUT).
  2. Questa sezione di codice ha lo scopo di leggere l’ingresso digitale (un INPUT di tipo booleano) di uno specifico pin.
  3. Questa sezione di codice ha lo scopo di assegnare all’uscita

digitale un valore “True” oppure “False”.

* 1. Questa sezione di codice ha lo scopo di leggere l’ingresso analogico di un pin, che ritornerà al µC un valore compreso fra 0 e 4095.
     1. Ad alcuni di noi la lettura del pin tornava un valore errato, abbiamo quindi deciso di implementare una correzione software che utilizza un’attenuazione ed un coefficiente per convertire il valore analogico letto in tensione.

*APPROFONDIMENTO*

***Markdown:*** *è un linguaggio di formattazione leggero e intuitivo utilizzato per creare documenti con formattazione semplice e leggibile. Con Markdown, puoi facilmente formattare il testo utilizzando caratteri speciali per creare titoli, testo in grassetto o corsivo, collegamenti e immagini. È ampiamente utilizzato per scrivere file README.md per documentare progetti software o fornire istruzioni chiare. La sintassi di Markdown è facile da imparare e ti permette di creare documenti strutturati senza dover ricorrere all'HTML o ad altri linguaggi complessi.*

***Doxygen****: Doxygen è uno strumento di documentazione del codice sorgente ampiamente utilizzato che automatizza il processo di generazione di documentazione leggibile e ben formattata per progetti software. Supporta numerosi linguaggi di programmazione, inclusi C, C++, Java, Python e molti altri. Doxygen analizza il codice sorgente annotato con commenti speciali (solitamente seguendo la sintassi del linguaggio di programmazione) e genera documentazione in diversi formati, come HTML, PDF e file di testo. La documentazione generata da Doxygen può includere informazioni sulle classi, le funzioni, le variabili, i diagrammi delle dipendenze, i diagrammi di ereditarietà e molto altro. Questa documentazione è estremamente utile per i programmatori che lavorano su un progetto, facilitando la comprensione del codice, la collaborazione e la manutenzione del software. Doxygen è un software open-source e altamente configurabile, che lo rende uno strumento flessibile per la generazione di documentazione di alta qualità per progetti di sviluppo software. All’interno del nostro progetto lo abbiamo ritrovato in alcuni file di librerie e esempi su cui abbiamo sviluppato il nostro lavoro.*

# RETI

## Protocolli ISO/OSI

L'International Standards Organization ha definito un modello di riferimento per la descrizione delle funzioni dei sistemi di rete detto: Open-Systems Interconnection (Modello di interconnessione dei sistemi aperti). Il modello descrive le funzioni necessarie all'interconnessione di due sistemi mediante una gerarchia di sette livelli:

1. **Physical:** Si occupa di trasmettere sequenze binarie sul canale di comunicazione, a questo livello si specificano i valori di tensione che rappresentano 0 e 1 e le caratteristiche dei cavi e dei connettori.
2. **Data link:** Ha come scopo la trasmissione sufficientemente affidabile di frame (frame: sequenze di pacchetti contenenti dati e informazioni di controllo). Verifica la presenza di errori di trasmissione aggiungendo dei FCS (Frame Control Sequence) e può gestire meccanismi di correzione di tali errori.
3. **Network Routing:** A questo livello vengono definite le modalità utilizzate per trasferire l'informazione fra due nodi, non direttamente interconnessi, ma attraverso una "rete". Esso gestisce l'instradamento dei messaggi: determina se e quali sistemi intermedi devono essere attraversati dal messaggio per giungere a destinazione, quindi deve gestire delle tabelle di instradamento e provvedere ad instradamenti alternativi in caso di guasti.
4. **Transport:** Si occupa di fornire un trasferimento dati affidabile. Compiti tipici del livello 4 saranno: la frammentazione delle informazioni provenienti dal livello superiore in pacchetti, garantire la corretta sequenza di arrivo dei pacchetti, la ritrasmissione in caso di errore quando si richiede un collegamento di tipo "affidabile", la prevenzione della congestione della rete. Il livello 4 è il livello più basso a trascurare la topologia della rete e la presenza di sistemi intermedi e quindi è il primo livello definito end-to-end.
5. **Session:** È responsabile dell'organizzazione del dialogo tra due programmi e del conseguente scambio di dati. A questo livello vengono definite le modalità per attivare e disattivare una connessione fra utente e

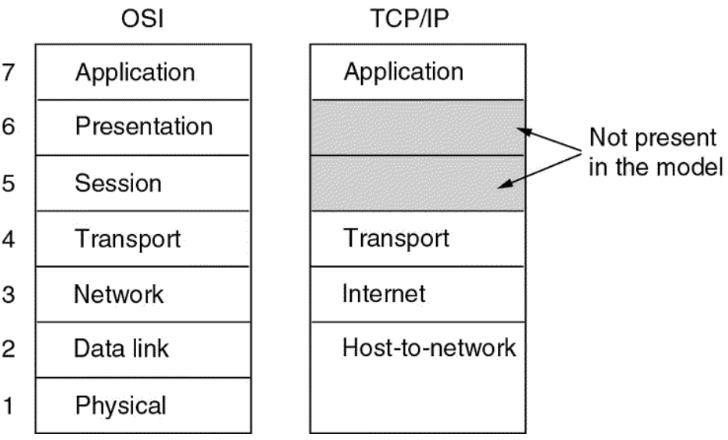
applicazione (o fra due applicazioni), nonché altre funzioni di alto livello, come ad esempio la gestione di una periferica condivisa.

1. **Presentation:** Gestisce la sintassi dell'informazione da trasferire (ad es. la codifica ASCII-EBCDIC). A questo livello vengono definite le modalità per garantire la corretta interpretazione dei dati (ad esempio conversione fra codici alfanumerici o l’applicazione di algoritmi crittografici).
2. **Application:** Sono le librerie su cui si appoggiano direttamente gli applicativi per effettuare: trasferimenti di file, richieste di pagine web, l'adattamento di terminali diversi ad una stessa applicazione e conversioni di regole per la denominazione di più file.

## Protocollo TCP/IP

Il protocollo TCP/IP è uno standard “de facto”, cioè si è un insieme di regole e protocolli che viene utilizzato

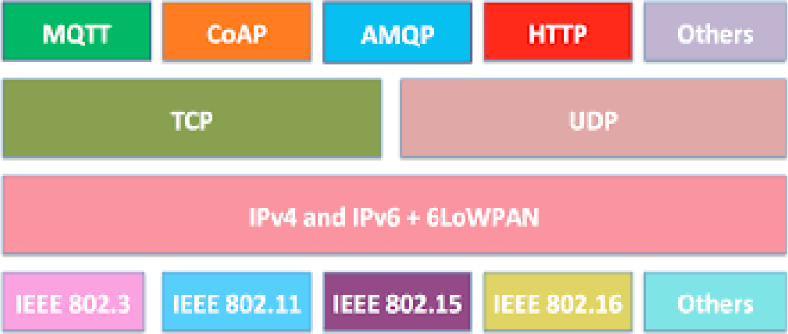
in tutto il mondo per il collegamento internet. Le differenze principali fra ISO/OSI e TCP/IP sono:



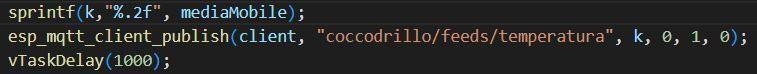
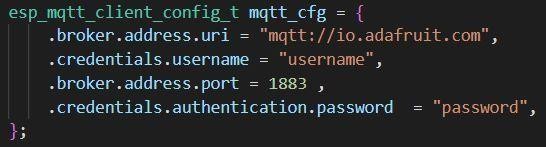
Delle nozioni basi del TCP/IP sono:

* MAC Address: esso appartiene al secondo livello del modello OSI, è composto da 6 byte: i primi 3 identificano il produttore della scheda di rete mentre gli ultimi 3 sono il codice univoco assegnato dal produttore stesso. Esso viene utilizzato per identificare in modo unico un dispositivo all’interno dellarete LAN
* IP Address: esso appartiene al terzo livello del modello OSI, può essere composto da 4 o 16 byte. L’indirizzo IP permette di identificare univocamente un dispositivo nella rete e consente la comunicazione tra dispositivi attraverso l’instradamento dei pacchetti di dati. I pacchetti contengono sia l’indirizzo IP di origine che quello di destinazione. Gli indirizzi IP possono essere assegnati staticamente (un amministratore di rete configura manualmente l’indirizzo IP ) o dinamicamente (vengono utilizzati protocolli come il DHCP che assegnano in automatico gli indirizzi IP).

## MQTT

MQTT (Message Queuing Telemetry Transport) è un [protocollo](https://it.wikipedia.org/wiki/Protocollo_di_rete) [ISO standard](https://it.wikipedia.org/wiki/International_Organization_for_Standardization) di messaggistica di tipo [publish-](https://it.wikipedia.org/wiki/Publish/subscribe) [subscribe](https://it.wikipedia.org/wiki/Publish/subscribe) posizionato in cima a [TCP/IP](https://it.wikipedia.org/wiki/Suite_di_protocolli_Internet). Il [pattern](https://it.wikipedia.org/wiki/Pattern) publish-subscribe richiede un [broker di messaggistica](https://it.wikipedia.org/wiki/Message-oriented_middleware), che nel nostro caso è Adafruit, il quale è responsabile della distribuzione dei messaggi ai client destinatari. Nella tipologia pubblish-subscribe le informazioni che vogliamo mandare vengono pubblicate nel broker, e poi quando qualcuno le vorrà leggere si iscriverà a quel feed. Nel

caso in cui noi pubblichiamo dati sul nostro broker, ovvero Adafruit, essi vengono inoltrati nel feed scelto. Possiamo anche iscriverci ad un feed specifico, quindi ogni volta che un nuovo dato viene aggiunto al canale, una variabile viene aggiornata con il dato corrente.



## QoS

La Quality of Service (QoS) si riferisce alla capacità di fornire livelli di prestazioni, affidabilità e priorità specifici in base alle esigenze delle applicazioni o dei servizi. Nel campo delle reti di telecomunicazioni, il termine qualità del servizio o più semplicemente QoS è utilizzato per indicare i parametri usati per caratterizzare la qualità del servizio offerto dalla rete (ad esempio perdita di pacchetti, ritardo), o gli strumenti o tecniche per ottenere una qualità di servizio desiderata.

## Confronto dei vari protocolli

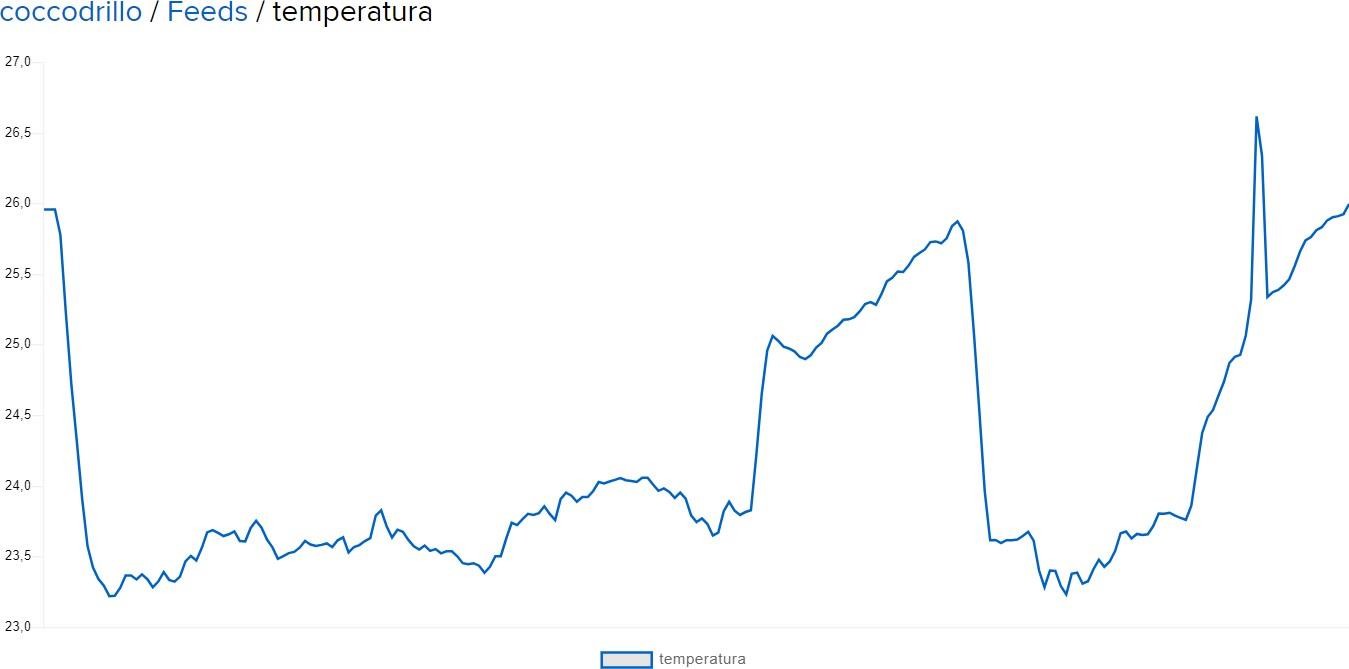
Riportiamo sottostante una tabella comparativa fra diversi protocolli di messaggistica per avere una panoramica più dettagliata.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Protocollo** | **Utilizzo Principale** | **Modello di messaggistica** | **Scalabilità** | **Overhead** | **Affidabilità** |
| **COAP** | IoT | Publish/subscribe | elevata su reti a bassa potenza e con restrizioni di larghezza di banda | Leggero, basso overhead | Non garantisce l'affidabilità delle consegne dei  messaggi |
| **MQTT** | IoT | Publish/subscribe | elevata su reti a bassa potenza e con restrizioni di larghezza di banda | Leggero, basso overhead | Non garantisce l'affidabilità delle  consegne dei messaggi |
| **AMQP** | Messagistica e integrazione di sistemi | Messaggistica punto-punto e publish/subscribe | elevata e supporto per scenari complessi di messaggistica e  integrazione di sistemi | Maggiore overhead rispetto a COAP e MQTT | Garantisce l'affidabilità delle consegne dei messaggi |
| **HTTP** | Comunicazione web | Request/response | elevata e supporto completo per le comunicazioni web | Maggiore overhead rispetto a COAP e MQTT | Garantisce l'affidabilità delle consegne dei messaggi |

Motivi perché abbiamo utilizzato MQTT a discapito di altri linguaggi:

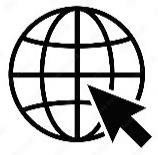
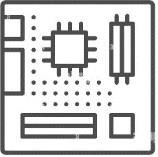
* è il protocollo più diffuso tra quelli sopraelencati
* è più semplice da comprendere e da utilizzare soprattutto per persone poco esperte
* esistono molti strumenti, librerie e risorse disponibili
* esso può essere anche usato anche quando la connettività IP non è disponibile

Esempio di feed:



Il nostro ESP32 per attuare la comunicazione col broker passa per una serie di collegamenti:

1. L’ESP svolge la funzione di client che manda i dati della temperatura rilevata
2. Per avviare la comunicazione si utilizza un access point che sarebbe il nostro WI-FI
3. Tutti gli access point sono collegati a uno switch
4. Riusciamo a interfacciarci con il mondo esterno grazie al dispositivo di gateway, nel nostro caso è un router



## DHCP

Il DHCP (Dynamic Host Configuration Protocol) è un protocollo di rete utilizzato per assegnare automaticamente indirizzi IP e altre informazioni di configurazione di rete ai dispositivi all’interno di essa. In sostanza il DHCP semplifica il processo di configurazione di una rete, consentendo ai dispositivi di ottenere automaticamente gli indirizzi IP e altre informazioni necessarie senza che sia richiesto un intervento manuale. Normalmente, quando un dispositivo si connette a una rete, ha bisogno di un indirizzo IP univoco per poter comunicare con gli altri dispositivi sulla rete. Tradizionalmente, l'amministratore di rete avrebbe dovuto assegnare manualmente un indirizzo IP a ciascun dispositivo. Tuttavia, con il DHCP, questo processo avviene in modo automatico e dinamico. Inoltre il server DHCP può fornire altre informazioni di configurazione di rete, come l’indirizzo del server DNS e il gateway predefinito. Il DHCP facilita anche il reindirizzamento dei dispositivi sulla rete in caso di modifiche alla configurazione della rete stessa.

*APPROFONDIMENTO*

*Il DNS (Domain Name System) , in* [*informatica*](https://it.wikipedia.org/wiki/Informatica) *e* [*telecomunicazioni,*](https://it.wikipedia.org/wiki/Telecomunicazioni) *indica un sistema utilizzato per assegnare nomi agli host (*[*nodi della rete*](https://it.wikipedia.org/wiki/Host)*). Indica inoltre: il protocollo che regola il funzionamento del servizio, i programmi che lo implementano, i* [*server*](https://it.wikipedia.org/wiki/Server) *su cui questi vengono elaborati, l'insieme di questi server che cooperano per fornire il servizio più intelligente.*

*L'operazione di conversione da nome a indirizzo IP è detta "risoluzione DNS”. Quest’ultimo ha una struttura gerarchica ad albero rovesciato ed è diviso in domini (com, org, it). Ad ogni dominio o nodo corrisponde un nameserver, che conserva un* [*database*](https://it.wikipedia.org/wiki/Base_di_dati) *con le informazioni di alcuni domini di cui è responsabile e si rivolge ai nodi successivi quando deve trovare informazioni che appartengono ad altri domini. Degli esempi di IP DNS sono: 169.254.169.253 (IPv4, indirizzo IP di Amazon), 8.8.8.8 (IPv4, indirizzo IP di Google) .*

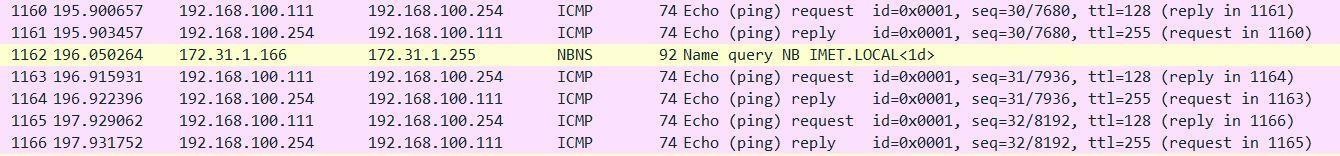
## IOT

L’internet of things (IoT) è un concetto che si riferisce alla connessione di oggetti fisici al mondo digitale attraverso la rete Internet. Questi oggetti, chiamati **"dispositivi intelligenti"**, includono una vasta gamma di elementi, come sensori, telecamere, elettrodomestici, veicoli e molto altro ancora. L'IoT consente a questi oggetti di comunicare tra loro e con gli utenti, consentendo una vasta gamma di applicazioni e servizi. L'idea chiave dell'IoT è di rendere gli oggetti più intelligenti, consentendo loro di **raccogliere dati**, scambiarli e prendere **decisioni** in modo autonomo. Questi dati possono essere raccolti dai sensori incorporati nei dispositivi e possono riguardare vari aspetti come temperatura, posizione, movimento, umidità, pressione e molto altro ancora. I dispositivi possono inviare questi dati a un sistema centrale o ad altri dispositivi connessi per analisi e elaborazione. Il nostro progetto segue le linee guida dell’IOT in quanto abbiamo una scheda che lavora come “dispositivo intelligente” scambiando i dati di temperatura autonomamente e il funzionamento del termostato è tutto su un piano di lavoro astratto.

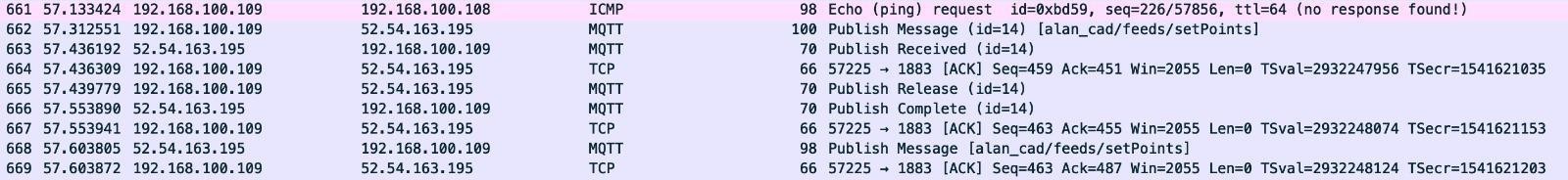
*APPROFONDIMENTO*

#### Wire-Shark:

*Ping eseguito all’indirizzo del dispositivo di Gateway:*



*Ping a un PC:*



# SOFTWARE

## Cosa è un software

Un software è un insieme di programmi e dati che consentono al computer di svolgere specifiche funzioni o attività. È un'entità intangibile composta da istruzioni logiche e algoritmi che consentono al computer di eseguire determinati compiti. In altre parole, un software è una serie di codici scritti in un linguaggio di programmazione che viene eseguito su un computer per svolgere una varietà di attività. I software possono essere classificati in diverse categorie, tra cui:

1. **Sistemi operativi:** Un sistema operativo è un software di base che controlla le risorse e le attività del computer. Fornisce un'interfaccia tra l'hardware del computer e gli utenti, consentendo loro di eseguire applicazioni e gestire file e dispositivi.
2. **Applicazioni software:** Le applicazioni software sono progettate per svolgere attività specifiche per gli utenti finali. Possono includere software per l'ufficio come elaboratori di testo, fogli di calcolo e programmi di presentazione, nonché applicazioni per la produttività, l'elaborazione delle immagini, la progettazione grafica, i giochi e molto altro.
3. **Software di sviluppo:** Questi software sono utilizzati per creare, debuggare, testare e mantenere altri software. Comprendono ambienti di sviluppo integrati (IDE), compilatori, debugger e altre utility utili agli sviluppatori nel processo di creazione di nuovi software.

## Linguaggio di programmazione

Il linguaggio utilizzato all’interno del nostro software è il linguaggio Dart. Quest’ultimo è un linguaggio di programmazione sviluppato da Google, progettato per la creazione di applicazioni per il web, per dispositivi mobili e per il server-side. È un linguaggio di tipizzazione statica che combina caratteristiche di programmazione orientata agli oggetti con elementi di programmazione funzionale.

Ecco alcune caratteristiche principali di Dart:

1. **Tipizzazione statica:** Dart utilizza la tipizzazione statica, il che significa che i tipi delle variabili devono essere dichiarati esplicitamente e vengono controllati a tempo di compilazione. Ciò consente di rilevare errori di tipo durante la fase di sviluppo e rende il codice più sicuro e robusto.
2. **Object-oriented:** Dart è un linguaggio orientato agli oggetti, il che significa che tutto in Dart è un oggetto. Supporta classi e oggetti, ereditarietà, incapsulamento e polimorfismo. Questa caratteristica rende Dart adatto alla creazione di applicazioni modulari e riutilizzabili.
3. **Async/await:** Dart supporta il modello di programmazione asincrona basato su futures e async/await. Questo permette di scrivere codice conciso e leggibile per le operazioni asincrone, come le chiamate di rete o l'accesso ai database, senza bloccare il thread principale.

## Flutter

Flutter è un framework open-source sviluppato da Google per la creazione di app mobili native per diverse piattaforme, tra cui Android, iOS, Web e Desktop, utilizzando un unico codice base. I punti di forza di flutter sono:

1. **Architettura:** Flutter utilizza un'architettura chiamata "Flutter SDK" che consente di creare interfacce utente reattive e moderne. La sua architettura si basa sul concetto di widget, che rappresentano elementi dell'interfaccia utente come pulsanti, campi di input, layout, ecc. Flutter offre una vasta gamma di widget predefiniti e personalizzabili che possono essere combinati per creare interfacce complesse.
2. **Hot Reload:** Una delle caratteristiche distintive di Flutter è il suo "Hot Reload". Questo strumento permette agli sviluppatori di apportare modifiche al codice sorgente e di visualizzare immediatamente i risultati nell'app in esecuzione, senza dover ricompilare l'intero progetto. Questo acceleratore di sviluppo consente di iterare rapidamente sul design e sul funzionamento dell'app, riducendo i tempi di sviluppo.
3. **Performance:** Flutter è noto per le sue prestazioni elevate. Utilizzando il proprio motore di rendering, Flutter è in grado di creare app dal design fluido e reattivo, con tempi di avvio rapidi e una buona fluidità delle animazioni. Inoltre, Flutter elimina la necessità di un "bridge" tra il codice Flutter e la piattaforma nativa, consentendo di ottenere prestazioni comparabili o addirittura migliori rispetto alle app native.

*APPROFONDIMENTO*

#### Framework:

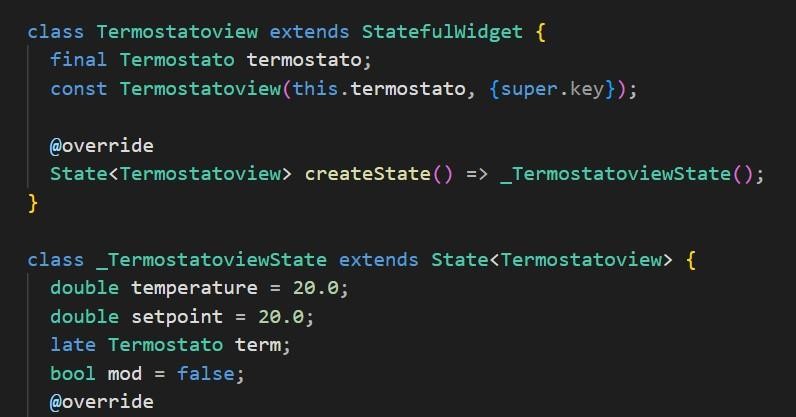
*Un framework è come una struttura di base che aiuta gli sviluppatori a creare applicazioni o siti web. È un insieme di strumenti, librerie e linee guida che semplificano il processo di sviluppo.*

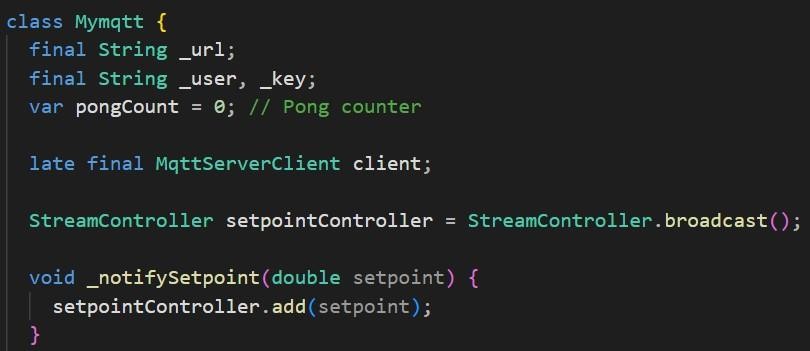
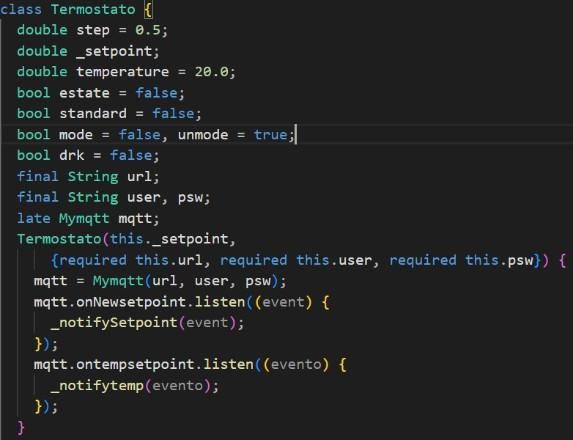
*Un framework fornisce un'infrastruttura predefinita per lo sviluppo di un'applicazione o di un sito web. Include funzionalità comuni, come la gestione dei dati, l'interazione con il database, la gestione degli errori e la sicurezza. I programmatori possono concentrarsi sulla scrittura del codice specifico dell'applicazione, utilizzando il framework come base solida su cui costruire. I framework sono progettati per essere riutilizzabili, flessibili e scalabili. Sono sviluppati da esperti nel campo e offrono una struttura solida e ben testata per lo sviluppo di software. In sintesi, un framework è uno strumento che semplifica lo sviluppo di software fornendo una struttura di base e librerie predefinite per creare applicazioni o siti web in modo più efficiente.*

## Implementazione delle classi

Per quanto riguarda il nostro progetto abbiamo avuto la necessità di creare delle classi che ci permettessero il corretto funzionamento del programma. Ma innanzitutto definiamo cosa è una classe:

Una classe rappresenta un modello o una struttura che definisce le caratteristiche e il comportamento di un oggetto. Essa è una **struttura** di dati che definisce le proprietà (**attributi**) e i comportamenti (**metodi**) di un oggetto. I membri di una classe includono variabili per memorizzare dati e funzioni per eseguire operazioni su quei dati. Gli attributi di una classe sono spesso dichiarati come privati o protetti per nascondere l'implementazione interna e fornire un'interfaccia controllata per l'interazione con l'oggetto. Una classe può essere ereditata da altre classi, questo concetto si chiama **ereditarietà** ed è utile quando si desidera definire una nuova classe basata su una classe esistente. La classe figlia (o sottoclasse) eredita gli attributi e i metodi della classe genitore (o superclasse) e può anche aggiungere nuovi membri o sovrascrivere quelli esistenti. Questo permette di scrivere codice più generico ed estendibile, in cui un metodo può comportarsi in modo diverso a seconda del tipo di oggetto che lo esegue. Per utilizzare una classe e creare oggetti, è necessario istanziare la classe. **L'istanza** di una classe è un singolo oggetto che viene creato utilizzando il **costruttore** della classe. L'oggetto creato avrà le caratteristiche definite dalla classe e avrà accesso ai suoi metodi. Le classi permettono di organizzare il codice in moduli riutilizzabili. Una volta definita una classe, può essere istanziata più volte per **creare diversi oggetti** con lo stesso schema di attributi e metodi. Questo favorisce il riuso del codice, in quanto non è necessario scrivere lo stesso codice ripetutamente per oggetti simili. Le classi sono uno strumento fondamentale per organizzare e strutturare il codice in modo modulare ed estendibile. Consentono di creare oggetti con caratteristiche e comportamenti specifici e promuovono una migliore organizzazione, manutenzione e riusabilità del codice.

Nel nostro caso abbiamo creato delle classi per:

1. La parte grafica dell’app, ovvero il codice per la parte visualizzabile dell’applicazione
2. La parte del collegamento con il broker, ovvero la parte del protocollo di rete MQTT
3. La parte di codice che invece regola comportamenti del nostro termostato in base a temperatura e dati forniti dall’utente

## Ciclo di vita di un stateful widget

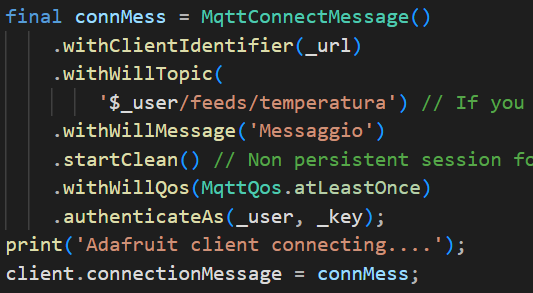
Il ciclo di vita di uno Stateful Widget in Flutter comprende diversi momenti chiave durante il suo utilizzo. Ecco un breve riassunto:

1. **Creazione (createState)**: Il widget viene creato utilizzando il costruttore.
2. **Mounting (Montaggio):** Il widget viene "montato" nell'albero dei widget, rendendolo visibile nell'interfaccia utente. Durante questa fase, il metodo build() viene chiamato per costruire l'interfaccia utente iniziale.
3. **Aggiornamento:** Quando il widget viene aggiornato, il metodo build() viene richiamato nuovamente per riflettere le modifiche apportate allo stato o alle proprietà del widget. Questo può essere innescato da chiamate a setState(), che notifica Flutter che il widget deve essere ridisegnato.
4. **Unmounting (Smontaggio):** Se il widget viene rimosso dall'albero dei widget, viene "smontato". Durante questa fase, il metodo dispose() viene chiamato per eseguire eventuali operazioni di pulizia, come la rimozione di ascoltatori o l'annullamento di risorse allocate.

## Grafica dell’applicazione

Il codice che si occupa dell’applicazione è nella classe termostato.view, scritto in linguaggio Dart, che ci permette di cambiare lo sfondo e aggiungere scritte personalizzabili in tutti i loro parametri (dimensione carattere, colore carattere, font) e aggiungere pulsanti con delle risposte alla loro pressione e anche essi personalizzabili (colore, icona dimensione). Nelle nostre applicazioni (che variano l’uno dall’altro) si possono scegliere a discrezione dell’utente diversi parametri grafici e in base alla modalità selezionata l’app assume un aspetto differente.

## Comunicazione col broker

Nella nostra applicazione il broker è Adafruit e la comunicazione con esso è ottenuta con il protocollo di rete MQTT fatto nella classe “mqtt”. In questa classe si stabilisce una connessione all’account di Adafruit inserendo le credenziali di questo come in figura:

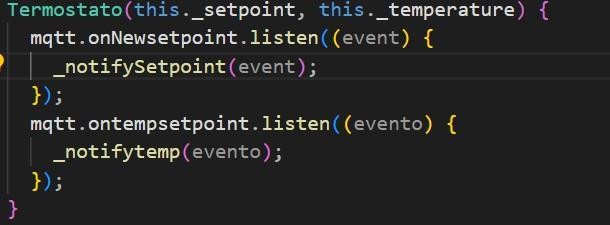
In seguito nel codice vengono create delle funzioni:

1. “listen”, che rileva le variazioni dei valori inseriti nel feed a cui si è iscritti in precedenza
2. “send”, che manda i dati al feed selezionato

Queste funzioni vengono poi chiamate nel resto del codice con il comando *mqtt.’nome della funzione’*.

## Aggiornamento grafica dell’app

Le scritte nell’applicazione che indicano la temperatura e il setpoint vengono aggiornate in base alle modifiche scelte dell’utente o che vengono rilevate dal sensore, questo avviene grazie al broker dove sono caricati i dati di temperatura e di setpoint in tempo reale.

Essi vengono letti con la funzione mqtt.onNewsetpoint.listen() che viene utilizzata con il seguente metodo:

## Creazione degli APK

Una volta che la applicazione è stata ultimata si può procedere alla generazione dell’APK, che servono per l’eventuale distribuzione dell’applicazione. Gli APK sono file utilizzati per installare e eseguire applicazioni su dispositivi Android, contengono: il codice dell'app, le risorse e le informazioni necessarie per la sua distribuzione. Gli APK possono essere scaricati da fonti affidabili come il Google Play Store o altri siti web, ma è importante fare attenzione alle fonti non attendibili per evitare rischi di sicurezza. In sintesi, gli APK sono il formato standard per la distribuzione delle app Android. Nel nostro caso personale l’APK è stato generato grazie al comando da terminale: “flutter run—release” che genera un percorso, il quale una volta copiato da un dispositivo android ne permetterà il download.

## Risultato finale

Il risultato finale, che viene ottenuto aggiungendo il sensore operante in parallelo al nostro software, ha un’applicazione che ci aggiorna ogni 10s sulla temperatura e un setpoint, modificabile dall’utente, che una volta che viene superato scatta il relè azionando un ipotetico riscaldamento/clima.

# IMPLEMENTI FUTURI

## Inverter

Un'altra aggiunta che si potrebbe considerare per un ipotetico futuro è quella di sostituire il sistema del semplice relè, che per quanto funzionale e economico ci permette solo di impostare il clima in ON/OFF, con un più versatile inverter. Quest’ultimo permetterebbe al nostro sistema di climatizzazione domestico di implementare una regolazione dinamica della temperatura prevedendo anche mezze vie di regolazione di essa diverse dal solito ON/OFF.

## AI

Nella possibilità in cui questo progetto avrà uno sviluppo futuro, questo sfocerebbe indubbiamente nell’AI (Artificial Intelligence). Grazie a questo sviluppo si otterrebbe un termostato che studia le nostre abitudini e grazie a un sensore può essere conscio della nostra presenza all’interno della casa. Così facendo potrebbe autonomamente regolare la temperatura in base alle abitudini dell’utente e farlo solo se esso è presente nell’abitazione. Quest’implementazione avrebbe delle conseguenze positive in: comfort, consumi e impatto sull’ambiente, poiché verrebbero azionati riscaldamento e clima solo nei momenti di reale bisogno e in cui verrebbero utilizzati effettivamente. Quindi l’intelligenza artificiale rappresenta realmente il futuro anche in ambiti di un così poco rilievo, perciò non si può fare a meno di pensare ai milioni di scenari in cui la potremmo ritrovare.