# Analisi Dettagliata di Home Assistant e Open Therm

## 1. Obiettivo del Progetto

L'obiettivo principale di questo progetto è sviluppare un'applicazione per Home Assistant che consenta la gestione "smart" (monitoraggio e controllo) di un boiler collegato a una caldaia, sfruttando il protocollo OpenTherm.

## 2. Introduzione: Il Protocollo OpenTherm

OpenTherm, ideato nel 1996, è un protocollo di comunicazione non proprietario che permette a caldaie (soprattutto quelle **modulanti**), condizionatori e termostati di dialogare in modo efficiente e intelligente.

**Vantaggi e Funzioni Chiave di OpenTherm:**

- **Efficienza Energetica:** Aumenta l'efficienza del sistema di riscaldamento, consentendo alla caldaia di modulare la potenza della fiamma in base alla richiesta effettiva di calore (andando oltre il semplice stato "acceso/spento").
- **Affidabilità:** Riduce i cicli di accensione/spegnimento ("start/stop") della caldaia, prolungandone la vita utile e riducendo i consumi di gas.
- **Diagnostica e Controllo Remoto:** Permette di trasmettere e ricevere codici di errore e lo stato operativo, facilitando la diagnostica e il controllo da remoto (essenziale per l'integrazione smart).
- **Installazione Semplificata:** Richiede solo due fili per l'installazione e non necessita di batterie, in quanto l'alimentazione può essere ricavata dai collegamenti OpenTherm.

## 3. Introduzione: Home Assistant

Home Assistant è la piattaforma open-source scelta per fungere da hub di controllo domotico. La sua architettura modulare e la filosofia di integrazione locale sono fondamentali per questo progetto.Concetti Chiave di HA per il Progetto:

Piattaforma di Integrazione: HA gestisce migliaia di integrazioni, permettendo di connettere dispositivi diversi. Nel nostro caso, verrà utilizzato per integrare il protocollo OpenTherm tramite MQTT.

- Automazioni (Automation): Consente di definire logiche complesse basate su trigger, condizioni e azioni, permettendo la gestione autonoma del riscaldamento (es. spegni la caldaia se la finestra è aperta).
- Dashboard: Offre interfacce utente personalizzabili per monitorare e controllare lo stato della caldaia in tempo reale.
- MQTT (Message Queuing Telemetry Transport): È l'integrazione cruciale per il progetto, in quanto funge da ponte di comunicazione tra il gateway hardware (ESP32) e il Core di Home Assistant.

# 4. Architettura Proposta (Hardware e Software)

## 4.1. Componenti Fisici (Hardware)

L'interfaccia fisica tra la caldaia (OpenTherm) e la rete domestica (Home Assistant) sarà gestita dai seguenti componenti:

| Componente | Ruolo | Dettagli Tecnici |
|---|---|---|
| **Microcontrollore** | Unità di elaborazione e comunicazione | ESP32 (per connettività Wi-Fi) |
| **Interfaccia Protocollo** | Trasmissione dati Caldaia | Dispositivo OpenTherm |
| **Comunicazione Rete** | Scambio dati con HA | Modulo MQTT |

## 4.2. Flusso Logico dei Dati

Il flusso dati previsto sarà il seguente:

1. **Ricezione Dati:** Il Dispositivo OpenTherm riceve i dati dalla caldaia (es. temperatura attuale) e li inoltra all'ESP32.
2. **Elaborazione e Trasmissione:** Il codice Python sull'ESP32 elabora i dati e li pubblica sul broker MQTT.
3. **HA Ascolta:** Home Assistant (tramite l'integrazione MQTT) sottoscrive i topic e riceve i dati.
4. **Invio Dati (Controllo):** HA pubblica un comando (es. setpoint di temperatura) su un topic MQTT specifico.
5. **ESP32 Esegue:** Il codice Python sull'ESP32 sottoscrive il topic di comando, riceve l'istruzione e la inoltra al Dispositivo OpenTherm, che la trasmette alla caldaia.

# 5. Requisiti e Informazioni dalla Documentazione Ufficiale di Home Assistant

Per l'integrazione con un sistema basato su MQTT, Home Assistant richiede specifiche configurazioni e componenti.

## 5.1. Integrazione MQTT di Home Assistant

Home Assistant supporta nativamente la piattaforma MQTT. Per ricevere e inviare dati, Home Assistant necessita di:

- **Broker MQTT:** È richiesto un broker MQTT funzionante (es. Mosquitto), accessibile dalla rete domestica.
- **Integrazione MQTT:** L'integrazione MQTT deve essere configurata in Home Assistant.
- **Definizione dei Componenti:** Per rappresentare i dati della caldaia, è necessario definire entità specifiche in HA che utilizzino MQTT come piattaforma di comunicazione.

La configurazione in HA per il progetto prevede:

- **Ricezione Dati (Stato del Boiler):** Per la ricezione dei dati dalla caldaia (es. temperatura attuale, stato del bruciatore), è necessario definire dei **Sensori MQTT**.
- **Invio Dati (Controllo):** Per inviare comandi al termostato virtuale/boiler (es. impostazione temperatura target), si utilizzerà un **Termostato MQTT (MQTT HVAC / Climate)**.

## 5.2. Integrazione Software e Logica con Home Assistant (MQTT)

L'integrazione con Home Assistant sfrutta la logica di comunicazione software MQTT per simulare e controllare il dispositivo.

### 1. Ricezione Dati (Simulazione Sensori)

Per ricevere i dati (es. temperatura attuale, stato operativo del bruciatore) dal "boiler virtuale" su Home Assistant, si definiscono:

- **Sensori MQTT (MQTT Sensor):** Entità che "ascoltano" specifici topic MQTT. Quando il dispositivo hardware (ESP32) pubblica un messaggio sul topic, il sensore su Home Assistant si aggiorna.
    - *Esempio topic:* `homeassistant/boiler/temperatura_attuale`
    - *Esempio topic:* `homeassistant/boiler/stato_bruciatore`

### 2. Invio Comandi (Simulazione Termostato)

Per inviare comandi (es. impostare la temperatura desiderata - *setpoint*, accendere/spegnere il sistema) si usa l'entità standard per la climatizzazione:

- **Termostato MQTT (MQTT HVAC / Climate):** Entità complessa che gestisce il controllo di un sistema di riscaldamento. Funziona "pubblicando" comandi su specifici topic e "sottoscrivendo" i topic di stato.
    - **Topic di Comando:** Home Assistant pubblica la temperatura desiderata (*setpoint*) su un topic (es. `homeassistant/boiler/setpoint/command`).
    - **Topic di Stato:** Il sistema sottostante (ESP32/Dispositivo OpenTherm) legge questo comando e, dopo averlo eseguito, pubblica il nuovo stato su un topic di stato (es. `homeassistant/boiler/setpoint/state`) per confermare l'avvenuta modifica.

### Schema di Funzionamento Logico (Integrazione Software):

1. L'Utente imposta la temperatura target (*setpoint*) sul Termostato MQTT di Home Assistant (es. 20°C).

2. Home Assistant pubblica il valore "20" sul *command topic* (es. `homeassistant/boiler/setpoint/command`).
3. Il Sistema di Controllo Fisico (ESP32) sottoscrive questo topic e legge il valore "20".
4. **Logica di Controllo:** L'ESP32 confronta la *temperatura_attuale* (ottenuta da un sensore fisico/OpenTherm) con il *setpoint* "20" e invia il comando OpenTherm appropriato alla caldaia.
5. **Aggiornamento Stato:** Il sistema pubblica il nuovo stato (es. stato del bruciatore: "on") sui relativi *state topic*, che vengono visualizzati dai Sensori e dal Termostato di Home Assistant.

### Topic Essenziali per Termostato MQTT

La piattaforma `mqtt` per `climate` (climatizzazione/termostato) è l'ideale per gestire dispositivi come i termostati che richiedono un'impostazione e restituiscono uno stato. Il termostato MQTT richiede specifici topic per funzionare:

| **Topic Richiesto** | **Scopo** |
|---|---|
| `temperature_command_topic` | Dove HA pubblica la temperatura target desiderata. |
| `temperature_state_topic` | Dove l'ESP32 pubblica la temperatura target attualmente in uso/impostata. |
| `mode_command_topic` | Dove HA pubblica la modalità operativa (es. `heat`, `off`). |
| `mode_state_topic` | Dove l'ESP32 pubblica la modalità operativa attuale. |

## 6. Requisiti di Comunicazione Dati

### 6.1. Cosa ha bisogno Home Assistant per ricevere e inviare dati?

Home Assistant ha bisogno di un'integrazione MQTT correttamente configurata e della definizione delle entità (Sensor, Climate/MQTT HVAC) che puntino ai topic MQTT specifici usati dal dispositivo fisico (ESP32 + OpenTherm).

### 6.2. Cosa ha bisogno il Termostato (Virtuale/OpenTherm) per inviare dati al Dispositivo OpenTherm?

Il codice Python in esecuzione sull'ESP32 funge da "Termostato Virtuale" o gateway tra MQTT e OpenTherm.

1. **Librerie Python per MQTT:** È necessario utilizzare una libreria Python per la connettività MQTT (es. paho-mqtt o librerie specifiche per MicroPython/ESP32).
    - **Compito:** Sottoscrivere i topic di comando (es. `boiler/comando/#`) e pubblicare i topic di stato (es. `boiler/stato/#`).
2. **Libreria/Codice Python per OpenTherm:** È necessaria una libreria o un'implementazione custom che gestisca il protocollo OpenTherm. Questo codice deve:
    - Tradurre i comandi MQTT in messaggi OpenTherm validi da inviare al Dispositivo OpenTherm.
    - Tradurre i messaggi OpenTherm ricevuti dal Dispositivo OpenTherm in payload JSON/stringa da pubblicare su MQTT.

## 7. Analisi di un'Alternativa Commerciale: Termostato WiFi OpenTherm (DIYLESS)

Per contestualizzare l'obiettivo del progetto, è utile analizzare una soluzione commerciale o semi-commerciale esistente che persegue un risultato simile, come il termostato WiFi OpenTherm di DIYLESS Electronics.Caratteristiche Principali del Termostato DIYLESS

Il dispositivo è un termostato intelligente basato su chip ESP32 che funge da gateway OpenTherm/WiFi/MQTT per la caldaia.

- **Piattaforma Hardware:** Basato su un microcontrollore ESP32.
- **Protocollo:** Pienamente compatibile con tutte le versioni del protocollo OpenTherm.
- **Integrazione:** Dispone di integrazione nativa con **Home Assistant** tramite protocollo MQTT.
- **Controllo:** Utilizza un algoritmo di controllo **PI (Proportional-Integral)** per modulare la potenza della caldaia.
- **Funzionalità Aggiuntive:** Offre una propria interfaccia web per la configurazione e il supporto per sensori di temperatura esterni (Bluetooth o cablati).

Nota sulla Pertinenza al Progetto (Soluzione **Non Idonea** allo Scopo)

Questa configurazione, pur essendo un prodotto finale funzionale, **non è una soluzione idonea né un'alternativa valida** per l'obiettivo di questo progetto.

Il nostro progetto si concentra sullo **sviluppo e sull'implementazione da zero** della logica di comunicazione (Python su ESP32 e topic MQTT) che interconnetta l'hardware OpenTherm a Home Assistant. Acquistare una soluzione pre-fatta bypasserebbe l'intera fase di studio e apprendimento tecnico alla base della documentazione. L'analisi del prodotto DIYLESS serve unicamente come riferimento per le funzionalità e l'architettura di un prodotto completo.Il cuore del nostro progetto risiede nello **sviluppo e**

**nell'implementazione autonoma** della logica di comunicazione tra l'hardware OpenTherm e Home Assistant. Questo include la creazione del codice Python per ESP32 e la gestione dei *topic* MQTT. L'acquisto di una soluzione commerciale preesistente vanificherebbe l'obiettivo primario di apprendimento tecnico approfondito e la fase di studio documentale. Pertanto, l'analisi del prodotto DIYLESS ha il solo scopo di fornire un riferimento in termini di funzionalità e architettura per il nostro prodotto completo.

Pro e Contro del Termostato DIYLESS (Valutazione Critica)

| **Pro (Punti di Forza Teorici)** | **Contro (Punti Critici e Limiti per il Progetto)** |
|---|---|
| **Pronto all'Uso:** Soluzione già assemblata e testata. | **Bypass della Logica di Sviluppo:** Utilizzare questo dispositivo annullerebbe lo scopo didattico e di sviluppo del progetto (creare il proprio Termostato MQTT). |
| **Integrazione Immediata:** Connettività Home Assistant già implementata (se l'obiettivo è solo l'utilizzo, non la comprensione). | **Firmware a Scatola Nera:** La logica di controllo PI è pre-programmata e non può essere studiata o modificata a livello di codice (anche se si può cambiare il firmware, si perde il controllo PI ottimizzato del produttore). |
| **Algoritmo PI Avanzato:** Offre un controllo della caldaia più efficiente di un semplice on/off. | **Dipendenza Hardware:** Si è vincolati al design del PCB e non si ha la flessibilità di modificare o espandere l'hardware (es. scelta di un altro microcontrollore o layout della scheda). |
| **Costo Contenuto:** Relativamente economico per un prodotto commerciale con questa integrazione. | **Costo/Complessità Mascherata:** Nonostante sia un "kit", la sua precisione dipende dall'acquisto e integrazione di un sensore di temperatura esterno (cablato o Bluetooth) e di un alimentatore USB, aumentando la complessità finale. |
| **Hardware Aperto (Teorico):** Basato su ESP32, permettendo in teoria l'installazione di firmware custom (come ESPHome), sebbene ciò richieda lavoro aggiuntivo. | **Sensore Interno Inaffidabile:** Il sensore di temperatura integrato è di fatto inutilizzabile a causa dell'autoriscaldamento del chip ESP32, rendendo obbligatorio l'uso di sensori esterni. |