ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

HOME AUTOMATION: INTEGRAZIONE ED INTEROPERABILITÀ CON HOME ASSISTANT

Elaborato in
SISTEMI EMBEDDED E IOT

Relatore
Prof. ALESSANDRO RICCI

Presentata da
RICCARDO BACCA

Anno Accademico 2020 – 2021

Indice

Introduzione

La Home Automation è un ambito nella quale, l'IoT ha le potenzialità di fornire strumenti capaci di garantire nuove funzionalità, non ottenibili mediante l'approccio ad una visione più limitata da fattori economici ed ecosistemi proprietari. La chiusura di certi sistemi, legati perciò a specifici protocolli non open-source, ne limitano l'impiego sul mercato da parte di differenti produttori, per ampliare il proprio ambito di utilizzo. In questo contesto, negli ultimi anni con l'avvento e la rapida espansione dell'Internet Of Things anche all'interno di dispositivi semplici, si sono ottenute nuove possibilità di utilizzo e funzionalità, tali da avvicinare il mondo dell'Home Automation ad un range molto più elevato di persone. L'avvento di dispositivi intelligenti, connessi ad internet, ha portato però un problema: come integrare tutti questi dispositivi, all'interno di un'ambiente domestico, permettendo il loro utilizzo simultaneo ed interconnesso?

Tali apparati, sono studiati e messi in commercio da differenti produttori, i quali utilizzano spesso differenti protocolli di comunicazione per lo scambio di informazioni, non favorendo l'espansione, l'economicità e l'utilizzo di massa dell'Home Automation. A questo proposito, diverse aziende, hanno realizzato propri hub, in grado di integrare in un unico sistema (proprietario o open-source) differenti servizi o dispositivi, i quali utilizzano differenti protocolli di comunicazione e di sicurezza, ampliando non poco la possibilità di creare nuovi scenari ed aree applicative, non previsti dal singolo produttore originario. Sotto questo aspetto, anche Apple e Google hanno sviluppato sistemi, nati con questo obbiettivo, ma infine ristretti per motivi economici e di marchio, obbligando eventuali produttori esterni con il desiderio di sviluppare prodotti da integrare nel loro ecosistema, a rispettare determinati vincoli progettuali restrittivi, quali ad esempio specifici protocolli di comunicazione o di sicurezza, specifici tecnologie integrate da utilizzare (Wi-Fi, Bluetooth) ecc.

Diversamente dalle aziende sopra citate, Samsung ed Home Assistant, hanno adottato un approccio del tutto contrario: adottando una tecnologia Open-source viene messo a disposizione del produttore esterno di dispositivi IoT, l'intero apparato dell'ecosistema nel quale integrarsi, consentendo una maggior fruibilità del sistema con minori costi d'acquisto, maggiore semplicità e

supporto a numerosi protocolli e tecnologie differenti.

In questo contesto, si ambienta il caso di studio della tesi: Home Assistant, pensato e realizzato con lo scopo di integrare, con il minor costo possibile, il maggior numero di dispositivi e servizi esterni, forniti da produttori differenti, utilizzando altrettanti protocolli e tecnologie di comunicazione, fornendo l'apertura software e hardware di cui l'Home Automation è purtroppo carente. Lo scopo di questo documento, è stato quello di fornire uno strumento in grado di avvicinare al mondo dell'Home Automation un pubblico ancora contrariato all'utilizzo di tali tecnologie nell'ambiente domestico, dati gli elevati costi e la chiusura a determinati dispositivi di certi ecosistemi, fornendo allo stesso tempo una guida ed una analisi approfondita su Home Assistant, uno strumento open-source, con l'obbiettivo di integrare differenti apparati all'interno di un unico ecosistema, permettendo una visione d'insieme molto più ampia di quella del singolo costruttore di oggetti IoT, con conseguenti espansioni e riduzioni dei costi per l'utente finale.

Analizziamo ora in breve, la suddivisione dell'elaborato, per aiutare la ricerca di un eventuale argomento specifico da parte del lettore. Il primo capitolo tratta una breve introduzione contestuale all'Home Automation, inserendo cenni storici, caratteristiche principali ed eventuali scenari ed aree applicative, approfonditi con l'avvento dell'Internet of Things, nel secondo capitolo. La seconda sezione, tratta approfonditamente il mondo dell'IoT, con particolare riguardo ai vantaggi (e svantaggi) che questo ha portato con la sua introduzione nell'Home Automation. In seguito vengono elencati ed introdotti i principali sistemi più utilizzati presenti ad oggi sul mercato, introducendo, in breve, il caso di studio della tesi. Il terzo capitolo, tratta infatti un'analisi approfondita sulla struttura di Home Assistant, partendo dalle sue basi, passando per la sua composizione, arrivando allo studio delle sue componenti ed a come interagiscono tra di loro. Infine, il quarto capitolo, tratta lo sviluppo di un semplice sistema domotico, per il controllo dell'intensità luminosa all'interno di un ambiente lavorativo. In particolare si procede dall'analisi dei casi d'uso, per definire il range di utilizzo di un tale sistema, passando per lo studio dei componenti di Home Assistant utilizzati, arrivando allo sviluppo ed allo studio delle automazioni utilizzate, componenti cardine del sistema. A conclusione vengono esposte alcune valutazioni su quanto realizzato.

Capitolo 1

Home Automation

Il vero problema nella riprogettazione della casa per aprirla alle soluzioni future consiste in uno sforzo creativo nel capire come utilizzare al meglio le nuove funzionalità e l'integrazione tra sistemi, a prescindere dalle particolari soluzioni tecnologiche che verranno adottate nella successiva realizzazione fisica.[12]

Da molti la domotica viene definita come la *tecnologia che studia l'automazione domestica*, ma l'obbiettivo di tale tecnologia è arrivare a considerare l'abitazione come un unico sistema intelligente che integra tra loro i numerosi dispositivi connessi presenti, portando questi ultimi ad un livello di automazione molto superiore rispetto alla somma dei singoli sottoinsiemi.

1.1 Definizione

Dal punto di vista etimologico, il termine domotica, è un francesismo derivato dall'unione del termine *domus* (che in latino significa casa), e dal suffisso greco *ticos*. Quest'ultimo indica la scienza interdisciplinare che si occupa dello studio delle tecnologie adatte a migliorare la vita nella casa. Ci si può spesso imbattere in alcuni dei suoi anglismi più comuni, tra cui: Home Automation, Smart Home e Home System.

Dal punto di vista funzionale, l'Home Automation viene definita da esperti del settore, come *l'integrazione di differenti servizi all'interno di un'abitazione, utilizzando un sistema di comunicazione comune. Assicurando un'economica, sicura e confortevole gestione della casa, includendo un alto grado di intelligenza, funzionalità e flessibilità.*

Possiamo notare che il concetto cardine all'interno di una Smart Home, è l'integrazione tra i diversi dispositivi, i quali possono eventualmente utilizzare protocolli di comunicazione differenti, quindi sono necessari applicativi che

possano interagire mediante protocolli ed oggetti IoT assai differenti tra di loro.

1.2 Brevi cenni storici

Le origini della Smart Home, possono essere ricondotte a numerosi personaggi nel tempo trascorso dalla sua nascita. Primo fra tutti William Penn Powers, un costruttore del Wisconsin, che nel 1891, in seguito alla costituzione della Power Regulator Company avviò la produzione di regolatori di temperatura automatici.

L'idea fu appunto quella di creare un dispositivo molto semplice, provvisto di un liquido che rispondesse ai cambiamenti di temperatura, capace di regolare l'apporto di energia ad un riscaldamento

Figura 1.1: Simbolo dell'epoca indicante il regolatore di temperatura (meccanicamente) automatico[30]

Successivamente, nel 1907, a Chicago fu costruito il primo hotel dotato di un impianto automatico di condizionamento, e negli anni 50, sempre negli Stati Uniti, viene realizzato il System 320. Quest'ultimo fu il primo dispositivo di controllo multiplo di tutte le informazioni degli impianti presenti in diversi edifici, inaugurando l'era della Building Automation.

In seguito, nel 1966, *Jim Sutherland*, ingegnere presso la *Westinghouse Corporation*, creò l'ECHO IV (Electronic Computing House Operator), il primo dispositivo di automazione ad applicazione domestica, in grado di regolare e controllare automaticamente la temperatura ed alcune apparecchiature elettroniche.

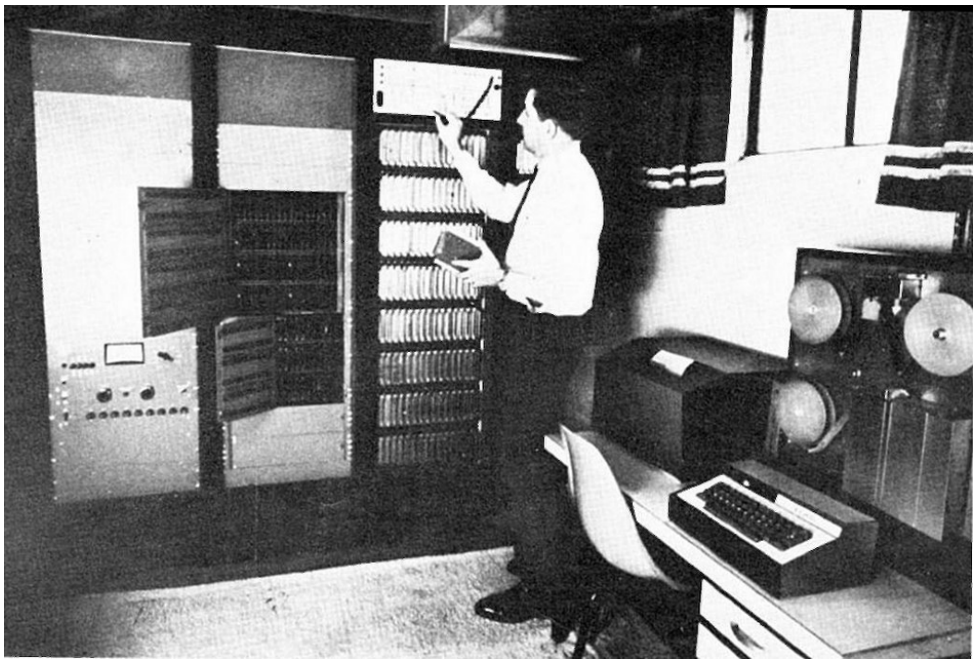

Figura 1.2: Jim Sutherland al lavoro sull'ECHO IV[30]

Grazie alla sua invenzione, Sutherland riuscì a prevedere con molta precisione, come sarebbero state le Smart Home del futuro: la sua idea era quella di progettare un dispositivo che regolasse la temperatura dell'abitazione basandosi sulle condizioni atmosferiche esterne, ed inserire nella dispensa della cucina un computer in grado di conoscere automaticamente quali prodotti servissero, riuscendo a creare un menù con uno specifico contenuto calorico.

Nel 1970, un gruppo di ingegneri della *Pico Electronics* sviluppò una base di X10, uno degli standard industriali ancora oggi più utilizzati. Questo utilizza le onde convogliate, in inglese PLC- Power Line Communication, per la segnalazione ed il controllo. Anche se questo fu il primo standard creato e con il passare del tempo sono stati sviluppati nuovi metodi alternativi per la comunicazione, tale standard rimane ancora oggi popolare in ambiente domestico con milioni di unità in uso in tutto il mondo, economico e vitale grazie alla continua introduzione di nuovi componenti sul mercato[30].

Probabilmente il primo progetto di una vera e propria Smart House, fu la casa di Ahwatukee ideata tra il 1970 ed il 1980. Questo edificio, fu il primo ad integrare il maggior numero di tecnologie presenti sul mercato, allo scopo di raggiungere un nuovo modo di abitare.

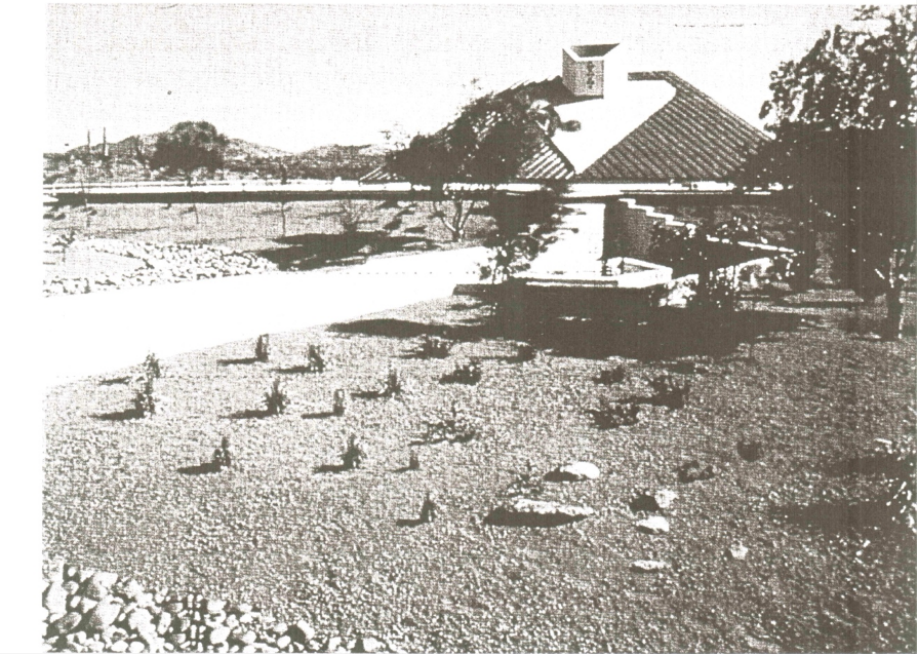

Figura 1.3: La casa Ahwatukee

1.3 Caratteristiche principali

Negli ultimi decenni sono stati condotti progetti di automazione domestica, che convogliano differenti idee, funzioni ed utilità. Per questo motivo le case domotiche, si stanno espandendo e distinguendo in differenti rami progettuali ponendo particolare attenzione all'interesse di ricercatori, richieste dell'utente utilizzatore finale ed aspettative.

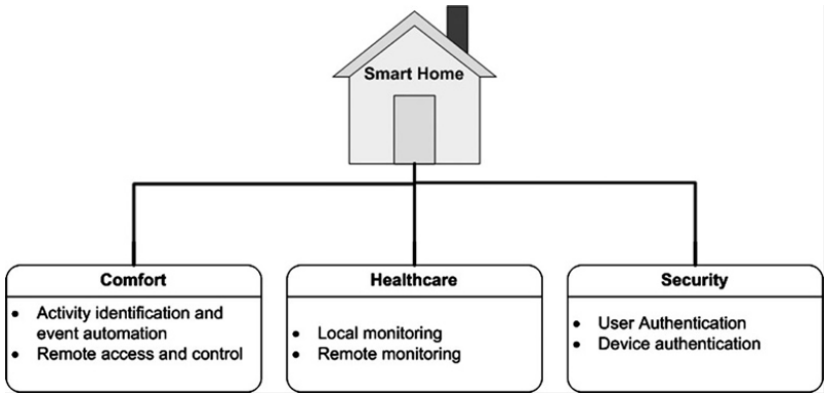

Figura 1.4: Categorizzazione degli ambiti domotici in accordo ai servizi forniti[13]

Le case domotiche, forniscono innanzitutto maggiore comfort, servizi a disposizione degli abitanti dell'edificio e sicurezza. Mentre i primi 2 benefici possono essere forniti localmente come a distanza, il tema della sicurezza è più complicato, si divide infatti in due aspetti: security e safety.

Analizziamo nel dettaglio questi tre aspetti peculiari delle Smart Home[30]:

- Comfort: Può essere raggiunto mediante il controllo remoto dell'abitazione con l'ausilio di applicazioni per la gestione degli apparati o identificando le attività compiute dall'utente, automatizzando determinati eventi ripetuti nel corso della giornata. Alla base di tutto ciò deve essere soddisfatto un requisito cardine per permettere alla Smart Home di prendere decisioni funzionali: acquisire consapevolezza del contesto di funzionamento. Senza questa caratteristica, la domotica non sarebbe capace di distinguere località, identità, attività ed orario. Per questo motivo, per provvedere ad un migliore Comfort e vivibilità dell'ambiente domestico si procede sempre più rapidamente verso il tracciamento, l'identificazione e l'automatizzazione di attività compiute regolarmente dall'utente.

- Servizi a disposizione dell'utente: prendiamo in considerazione servizi sanitari erogati a distanza mediante l'ausilio degli apparati domotici. Innanzitutto è possibile monitorare localmente il paziente, determinando l'attuale stato di salute, producendo analisi a lungo termine a riguardo della salute dell'utente. Queste possono poi essere analizzate dall'utente o da un medico. Il futuro del monitoraggio locale, sarà il monitoraggio remoto. Tale servizio sarà applicato con l'utilizzo di apparecchiature tali da provvedere automaticamente alla chiamata dei soccorsi in tempi notevolmente ridotti.

- Sicurezza: entrambe le parti nelle quali la sicurezza si specializza, condividono i meccanismi di funzionamento intelligenti. Infatti un sistema domotico ben programmato, nel caso in cui scatti un allarme può agire localmente sia con segnalazioni che con azioni di intervento, provvedendo allo stesso tempo alla segnalazione a distanza dell'accaduto tramite i collegamenti ad internet di cui dispone, eventualmente richiedendo l'intervento tempestivo delle forze dell'ordine. Inoltre se fossero presenti telecamere di sorveglianza sarebbe possibile, da parte dell'utente, visualizzare tramite apposita applicazione i filmati di sorveglianza.
  - Security: con questo termine si indica, in particolare, la protezione contro intrusioni non autorizzate o rapine
  - Safety: viene indicata la protezione globale dell'abitazione contro fughe di gas, incendi, allagamenti o altri eventi dannosi

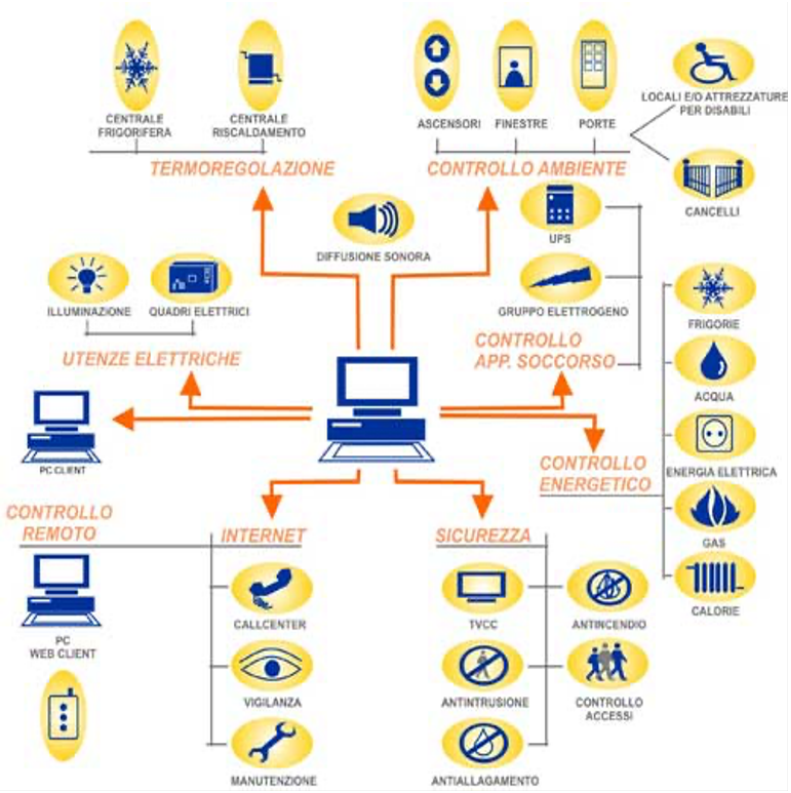

Figura 1.5: Applicazioni di un sistema domotico

Per fornire in modo semplice ed efficace i servizi sopra indicati, i prodotti domotici devono essere dotati di una tecnologia raffinata che si coniuga con

altre apparecchiature intelligenti presenti all'interno dell'abitazione. Tutto ciò convive in uno stesso ambiente grazie all'impiego di un sistema di controllo che garantisce in modo contemporaneo i seguenti requisiti fondamentali[29]:

- Semplicità: il sistema è facile da usare, nonostante gestisca tecnologie integrate in un sistema complesso, fornendo allo stesso tempo differenti interfacce per adattarsi agli utilizzatori.
- Affidabilità: il sistema nel tempo conserva le prestazioni iniziali, ed in caso di guasti dispone delle capacità di autodiagnosticare gli errori, comunicando in modo autonomo con un centro di servizi richiedendo eventuale intervento.
- Apertura: deve essere possibile inserire nuovi dispositivi o tecnologie, permettendo la comunicazione in vari modi senza l'aggiunta di numerose interfacce.
- Integrazione: il sistema deve poter comunicare con diverse tipologie di impianti, eseguendo le interazioni fra componenti differenti ove necessario.
- Flessibilità: il sistema è versatile ed adattabile alle richieste dell'utenza grazie al software, modificabile in qualsiasi momento.
- Espandibilità: l'applicativo si adegua alle esigenze applicative ed abitative, consentendo nel tempo l'implementazione delle componenti e delle sue funzioni.
- Continuità di funzionamento: viene garantita la continuità di funzionamento di tutte le applicazioni previste (ad esempio mediante l'utilizzo di gruppi soccorritori UPS in caso di black out).

1.4 Scenari ed aree applicative

L'obbiettivo cardine della Smart Home è la gestione ed il totale controllo dei servizi e la capacità di realizzazione nuove operazioni complesse.

Le funzioni ottenibili sono sostanzialmente infinite, tuttavia possono essere categorizzate in cinque aree di riferimento [12]:

- Sicurezza: include i concetti di Security e Safety, precedentemente esplicitati, è stato il primo ambito applicativo della domotica in quanto l'adozione di tecnologie intelligenti in questo ambito fornisce numerosi vantaggi.

- Climatizzazione: fu oggetto dei primi apparati domotici all'inizio del secolo scorso (seppur in maniera meccanica). Permette di programmare i tempi e gli orari di funzionamento, monitorando la temperatura dell'ambiente interno e le condizioni atmosferiche esterne, favorendo un minor consumo energetico.

- Gestione dei consumi e risparmio energetico: comprende innanzitutto la gestione dell'illuminazione e della qualità luminosa di un determinato ambiente, monitorando mediante opportuni sensori la luminosità e le condizioni atmosferiche, garantendo un consumo energetico ridotto.

- Comunicazione: comprende la gestione delle comunicazioni entranti e uscenti dall'abitazione, fornendo allo stesso tempo accesso ad internet alle apparecchiature domotiche.

- Intrattenimento e tempo libero: si fa riferimento ad apparecchi Hi-Fi per la riproduzione e diffusione audio, integrati all'interno del sistema domotico che ne permette l'utilizzo mediante un sistema di gestione delle sorgenti audio-video.

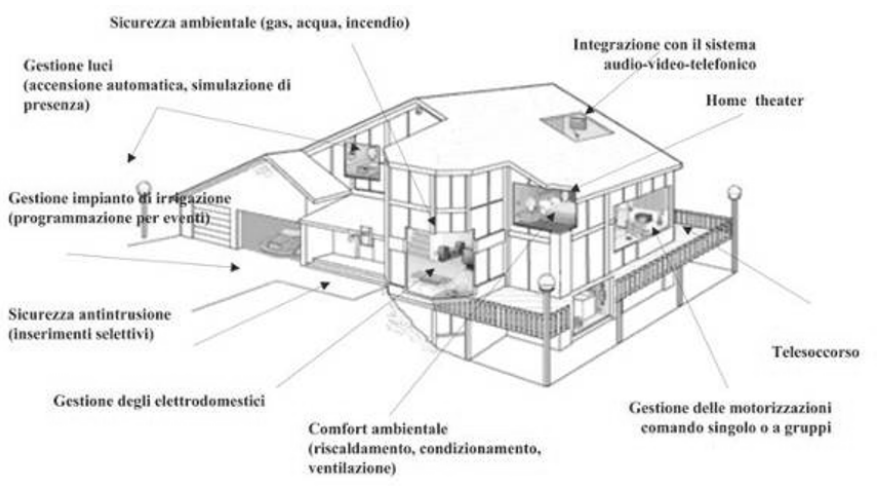

Figura 1.6: Funzioni possibili all'interno di una casa domotica

Capitolo 2

Internet Of Things

L'ambito dell'automazione domestica fu introdotto inizialmente con il solo ausilio di tecnologie limitanti (per range d'utilizzo e velocità di funzionamento) quali Bluetooth, Raggi Infrarossi. In seguito alla diffusione delle tecnologie moderne, nelle quali la connettività tra oggetti intelligenti è effettuate mediante internet, la domotica ha visto un incremento di utenti target notevole, in quanto le tipologie e ambiti di utilizzo sono aumentati esponenzialmente.

Il ruolo ricoperto dall'Internet Of Things in questo ambito è fondamentale: consente ad esempio la conversione di un campanello tradizionale in un oggetto domotico, in questo modo il proprietario può registrare il video della persona di fronte al proprio campanello e può eventualmente aprire la porta mediante il proprio cellulare. Inoltre l'IoT supera il limite di tecnologie precedentemente utilizzate: con l'avvento della tecnologia Wireless e di sensori smart economici i costi vengono notevolmente ridotti, il range di utilizzo viene massimizzato ed il consumo energetico viene minimizzato.

2.1 Definizione

Tecnicamente IoT è un ambiente composto da oggetti (things) inter-connessi ai quali viene assegnato un indirizzo IP ed hanno la possibilità e l'abilità di connettersi ad una rete senza intervento manuale dell'utente. Tale tecnologia può trasferire informazioni senza bisogno che l'utente debba intervenire sugli apparati fisici (human-to-human o human-to-computer). Un sistema Internet Of Things consiste dunque di oggetti hardware intelligenti (ad esempio microprocessori, sensori etc.) i quali sono responsabili dello scambio di informazioni con un determinato server o microcontrollore.

L'IoT sta diventando una delle tecnologie emergenti degli ultimi anni. Rappresenta l'espansione dei servizi forniti dalla rete ed ha modificato radicalmente il modo di vivere fornendo connettività ovunque con chiunque. Con il rapido

sviluppo di questa tecnologia, sono però nate numerose tecnologie di comunicazione differenti, che necessitano di adattamenti in modo da soddisfare i requisiti dell'Internet Of Things: efficienza energetica, velocità, sicurezza ed affidabilità.

2.2 Caratteristiche principali

L'Internet of Things è un'infrastruttura globale per la moderna società dell'informazione, abilita servizi avanzati interconnettendo oggetti (things) fisici e virtuali basandosi sulle attuali tecnologie dell'informazione e della comunicazione in continua evoluzione. L'architettura in discussione non è quindi un'unica tecnologia, ma un insieme di software e dispositivi smart.

Conseguentemente, per fornire soluzioni integrate basate sull'attuale tecnologia dell'informazione, determinate tecnologie necessitano di adattamenti per soddisfare i principali requisiti di IoT [25]:

- Inter-connettività: ogni oggetto riguardante l'internet delle cose può essere interconnesso con l'infrastruttura di comunicazione globale.
- Servizi relazionati agli oggetti: l'internet delle cose può fornire servizi entro i limiti degli oggetti, come protezione dalla privacy e consistenza semantica tra oggetti virtuali e corrispondenti oggetti fisici.
- Eterogeneità: I dispositivi all'interno dell'internet delle cose sono eterogenei, in quanto basati su differenti piattaforme hardware e reti. Possono interagire con altri oggetti o piattaforme mediante differenti reti.
- Cambiamenti dinamici: lo stato degli oggetti cambia dinamicamente (ad esempio connesso/disconnesso). Inoltre anche il numero di dispositivi connessi può cambiare dinamicamente.
- Crescita esponenziale: il numero di dispositivi che dovranno essere gestiti e che comunicheranno attivamente sarà almeno un ordine di grandezza maggiore ai dispositivi attualmente connessi alla rete. Ancora più critica sarà la gestione delle informazioni prodotte e generate dai dispositivi, e l'interpretazione di queste ultime ai fini applicativi.
- Safety: L'internet delle cose fornisce una enorme quantità di servizi, aumentando notevolmente il rischio per i dati personali che viaggiano sulla rete. Mettere in sicurezza i punti di interconnessione, le reti e le informazioni che vengono trasportate significa creare un paradigma di sicurezza in scala molto maggiore.

- Connettività: abilita l'accessibilità e la compatibilità delle differenti reti. Da un lato l'accessibilità su una rete deve essere sempre disponibile, mentre la compatibilità fornisce la comune capacità di utilizzare e produrre informazioni.

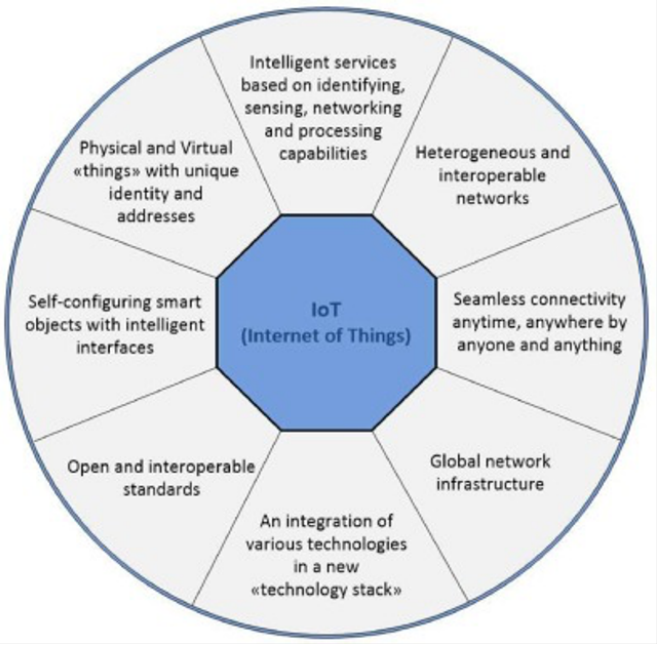

Figura 2.1: Internet Of Things: caratteristiche fondamentali[25]

L'architettura consiste di differenti livelli. Di seguito le funzionalità implementate da ogni livello[25]:

1. **Oggetti smart**

   Il livello di base è formato da dispositivi smart integrati con sensori. I sensori abilitano l'interconnessione della parte digitale e fisica, consentendo la memorizzazione e l'elaborazione di informazioni in tempo reale.

2. **Gateways e reti**

   I sensori, presenti nello strato sottostante, producono un'enorme quantità di informazioni, richiedendo un'infrastruttura di rete (cablata o wireless) robusta e prestante. Attualmente, però, le reti sono legate con protocolli molto differenti. Con l'ingente domanda di servire una gamma più ampia di servizi e applicazioni IoT (ad esempio servizi transazionali ad alta velocità) sono necessarie più reti con varie tecnologie e protocolli di accesso che possano lavorare tra loro eterogeneamente. Queste reti

possono essere pubbliche, private o ibride e sono implementate in modo da supportare i requisiti di: larghezza di banda, latenza e sicurezza.

3. **Servizi di gestione**

Il servizio di gestione rende possibile la gestione delle informazioni attraverso l'analisi, i controlli di sicurezza, la modellazione dei processi e la gestione dei dispositivi. Per quanto riguarda l'analisi dei dati, viene spesso utilizzata l'analitica in-memory consentendo la memorizzazione di grandi volumi di dati sulla RAM piuttosto che su dischi fisici, riducendo drasticamente il tempo di interrogazione dei dati. L'alternativa è lo streaming analytics nella quale l'analisi dei dati è effettuata in tempo reale, in quanto i dati sono considerati in movimento, permettendo di prendere decisioni in pochi secondi. La gestione dei dati è la capacità di gestire il flusso informativo. Le informazioni vengono rese accessibili, integrate. Perciò le applicazioni di livello superiore sono esonerate dall'elaborazione di dati non necessari, riducendo i costi ed il rischio di rilevazione della privacy della fonte dei dati.

4. **Livello applicativo**

L'applicazione IoT copre ambienti e spazi intelligenti in domini quali: trasporti, edilizia, stile di vita, agricoltura, emergenza, sanità, interazione con gli utenti, ambiente ed energia.

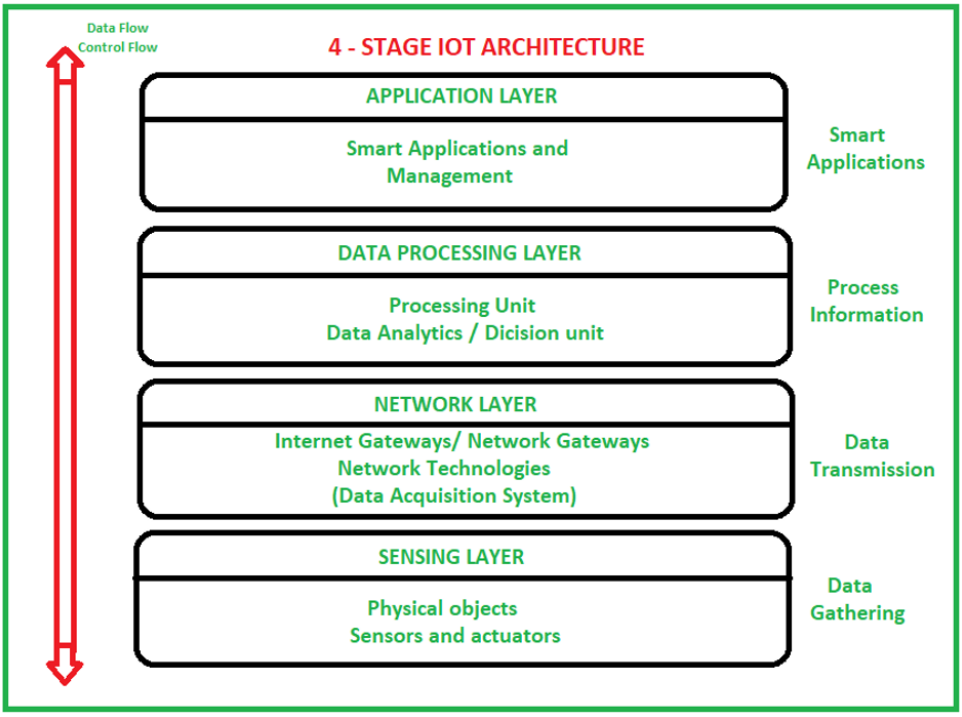

Figura 2.2: IoT Architecture[25]

2.3 Future evoluzioni

Lo sviluppo di tecnologie abilitanti come l'elettronica dei semiconduttori, le comunicazioni, i sensori, i telefoni intelligenti, il cloud networking, saranno essenziali per permettere ai dispositivi fisici smart di operare in ambienti mutevoli, rimanendo allo stesso tempo connessi sempre ed ovunque.

Inoltre l'Internet delle cose punta ad una sempre più radicale integrazione tra il mondo fisico ed il mondo virtuale, tuttavia l'eterogeneità dei dispositivi e delle tecnologie di comunicazione sottostanti rappresentano una sfida per l'espansione in corso su scala globale.

In conclusione, le principali problematiche che devono essere affrontate per permettere un'adozione di massa dell'Internet of Things sono categorizzabili in: sicurezza e privacy, Costo versus usabilità, Interoperabilità e Gestione delle informazioni prodotte dall'enorme mole di sensori attualmente in rete, in continua espansione.

Figura 2.3: Sfide future per l'Internet Of Things

2.4 IoT e Home Automation

Come descritto nei paragrafi precedenti, l'obbiettivo dell'integrazione dell'Home Automation con IoT, è di integrare tra loro differenti sistemi ed apparati mediante l'utilizzo di sistemi capaci di utilizzare differenti protocolli di comunicazione, per controllare i differenti apparati intelligenti. Fino ad oggi però i sistemi di Home Automation hanno riguardato singole aziende, utilizzavano protocolli di comunicazione proprietari ed incompatibili l'un con l'altro. Questi atteggiamenti protettivi verso le proprie tecnologie hanno provocato costi elevati per l'utente finale e non compatibilità con le apparecchiature già presenti all'interno dell'abitazione, non contribuendo all'espandersi della tecnologia in questione. Oggi con l'avvento della nuova generazione di sistemi, si ha la possibilità di integrare e connettere apparati differenti, nella maniera più semplice ed economica possibile. Questi ultimi spesso richiedono un dispositivo che lavori come hub (ovvero un dispositivo di rete che funge da nodo di smistamento per il traffico su una specifica rete), spesso però questi dispositivi sono implementati in maniera esclusivamente software. I differenti approcci appena descritti, possono essere riassunti in due categorie. Da un lato chi realizza un sistema con l'obbiettivo di integrare ed incorporare il maggior numero di dispositivi di produttori differenti, sfruttando numerosi e differenti protocolli di connessione allo stesso tempo oppure utilizzando protocolli web ed eventuali gateway nel caso di protocolli proprietari. Dall'altro sono presenti

i grandi nomi dell'attuale stato dell'arte della tecnologia, quali ad esempio Apple e Google, i quali sfruttando il fatto di avere un grosso pubblico che utilizza i loro prodotti, tentano di realizzare una propria infrastruttura e nuovi protocolli di comunicazione proprietari, che altri produttori saranno obbligati ad utilizzare per integrare i loro dispositivi in questi specifici ambienti. I criteri che Apple e Google hanno fissato per la propria Smart Home Automation, sono perciò vincolanti per coloro che desiderano far parte del loro sistema, infatti HomeKit richiede che i dispositivi lavorino esclusivamente tramite Bluetooth o Wi-Fi, ma richiedendo allo stesso tempo forti requisiti di cifratura e particolari requisiti hardware. Tutti i dispositivi compatibili con questo sistema sono infatti dotati delle stesse dotazioni di sicurezza: criptazione end-to-end delle comunicazioni, chiavi di cifratura non riutilizzabili ed autenticazione a due fattori. Mentre per Apple HomeKit e Brillo di Google è essenziale l'inclusione di nuovi dispositivi nei loro sistemi, dall'altro per sistemi che puntano all'integrazione quali Samsung SmartThings ed Home Assistant, è essenziale la presenza di un Hub, per integrare i diversi standard che i dispositivi utilizzano.

Apple HomeKit HomeKit è un framework che permette la configurazione, il controllo e la comunicazione con dispositivi Smart presenti all'interno dell'abitazione. Lo scopo, perciò, è quello di promuovere un protocollo comune per i dispositivi della home automation, e delle API pubbliche per poterli configurare e per poterci scambiare informazioni. I costruttori possono implementare HomeKit all'interno dei loro prodotti rendendoli facili da controllare, garantendone lo scambio di dati in modo sicuro ed in grado di lavorare con iPhone ed iPad.

Figura 2.4: Logo Apple HomeKit [14]

Questo framework permette ad app di terze parti di compiere tre funzioni principali:

- Rilevare accessori della Smart Home compatibili con HomeKit, aggiungendoli ad un database persistente e trasversale di configurazione della Smart Home.

- Visualizzare, modificare ed eseguire azioni sui dati presenti nel database nominato al punto precedente.

- Comunicare con gli accessori ed i servizi inseriti ed integrati per eseguire comandi.

Diversamente da altre piattaforme Apple non offre un dispositivo che svolge il ruolo di Hub centrale, ma il proprio ecosistema è basato sul fatto che l'utente possegga già un dispositivo IOS. Inizialmente era possibile utilizzare Apple HomeKit solo se si disponeva di una Apple TV (che fungeva quindi da Hub), oggi però viene offerto anche un HomeKit cloud, tramite il quale i dispositivi Wi-Fi possono lavorare anche se lo smartphone non è presente. All'interno di questo database sincronizzato, sono presenti tutti gli oggetti integrati su Apple HomeKit e le loro informazioni in tempo reale, in questo modo chiunque abbia accesso a queste informazioni per controllare la smart home, potrà avere le informazioni sempre aggiornate in tempo reale, potendo controllare le apparecchiature anche se non fosse presente lo smartphone. Per quanto riguarda la sicurezza di questo sistema, Apple obbliga alla cifratura end-to-end tra gli accessori e i dispositivi iOS, utilizzando chiavi di cifratura uniche e non riutilizzabili.

Per realizzare un dispositivo smart compatibile con HomeKit, viene richiesta ai produttori la sottoscrizione all'MFI Program per poter accedere alle risorse dedicate a chi vuole integrare la tecnologia in questione. Inoltre, insieme alle specifiche tecniche si riceve il logo MFi ed il conseguente diritto di apporlo sulle proprie confezioni ed un supporto tecnico dedicato all'hardware.

Per quanto concerne il funzionamento lato utente della tecnologia in esame, la Casa, viene vista come una collezione di accessori e le informazioni vengono organizzate in modo gerarchico definendo: Home, Room, Accessori, Service e Zone. Le Home possono essere partizionate in Zone e Room, mentre gli accessori sono assegnati ad una singola Room, inoltre si sottolinea il fatto che un singolo accessorio può permettere il controllo di più Servizi, ad esempio il dispositivo per l'apertura del garage può avere un servizio per aprire o chiudere la porta o un servizio per gestire la luce del dispositivo di apertura.

Google Brillo e Weave Nel Maggio del 2015, Google annunciò la propria visione dell'Internet Of Things all'interno di un'ambiente domestico, Brillo.

Brillo, è un sistema operativo, basato su Android, sviluppato e progettato essenzialmente per dispositivi embedded dotati di poca memoria e risorse utilizzabili per il calcolo. Può supportare dispositivi dotati di un minimo di 32mb di spazio d'archiviazione interno. Insieme a Brillo, Google annunciò un insieme di servizi per la comunicazione tra dispositivi intelligenti: Google Weave.

Weave è un protocollo di comunicazione tra dispositivi, con l'obbiettivo di fornire un linguaggio comune per lo scambio di informazioni Object-to-Object o Object-To-Cloud. Brillo supporta al suo interno nativamente, il protocollo Weave, ma questo protocollo può essere utilizzando anche da dispositivi Android ed iOS, sotto forma di librerie SDK. Perciò gli sviluppatori potranno sviluppare le proprie applicazioni per controllare uno o più dispositivi, basandosi sulla libreria Weave fornita da Google.

Architetturalmente parlando Brillo è una versione semplificata del sistema operativo Android: l'intero livello di framework basato su Java, è stato rimosso. Consiste unicamente dei livelli basici di Android ad esempio il Kernel ed Android HAL. Successivamente al livello HAL, sono presenti differenti librerie per facilitare alcune operazioni svolte dal sistema operativo Brillo, come: connettività ad internet, monitoraggio dei dispositivi, gestione energetica etc. I dispositivi Brillo possono essere basati su differenti architetture, anche per merito della somiglianza al sistema operativo Android: è possibile sviluppare ed assemblare dispositivi Brillo su architetture ARM, x86 o anche MIPS.

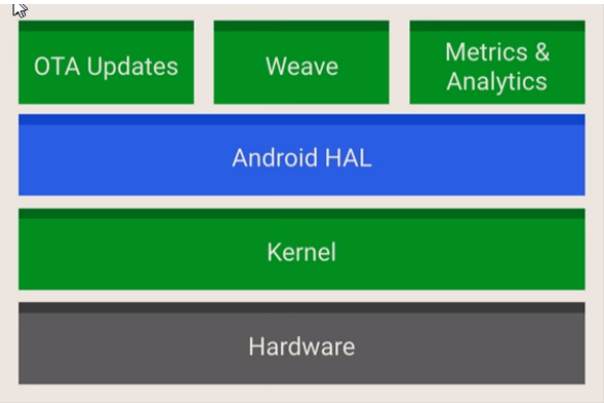

Figura 2.5: Architettura di Google Brillo [23]

Weave è una piattaforma di comunicazione basata su Json che comprende il supporto ad operazioni quali: ricerca attiva di dispositivi in rete, autenticazione e fornitura di altri servizi.

I dispositivi possono comunicare utilizzando Weave tramite Wi-Fi o Bluetooth LE. Viene inoltre supportata, la comunicazione sulla rete locale, o attraverso il cloud, infatti nel caso in cui un dipositivo/smartphone fosse connesso

ad una rete differente, le informazioni saranno indirizzate al ricevente mediante il cloud. Nel caso in cui, invece, i dispositivi si trovino sulla stessa rete, il messaggio verrà recapitato direttamente al ricevente, senza passaggi intermedi. Questo protocollo di comunicazione si compone di tre componenti fondamentali per il corretto funzionamento: cloud, device e client.

I servizi cloud forniscono operazioni di gestione dei dispositivi associati, della gestione degli accessi e delle notifiche push ad applicazioni e dispositivi. Forniscono inoltre una REST API per applicazioni e servizi web.

Per quanto riguarda il lato device, viene fornita una libreria per garantire la compatibilità con Weave, chaimata libweave. Quest'ultima è una libreria in linguaggio C++ che fornisce una visione di alto livelo sul Sistema Operativo ed implementa il protocollo Weave. E' presente un'ulteriore libreria, denominata libuweave, progettata e sviluppata per l'esecuzione del protocollo su microcontrollori, sui quali Linux non può essere installato.

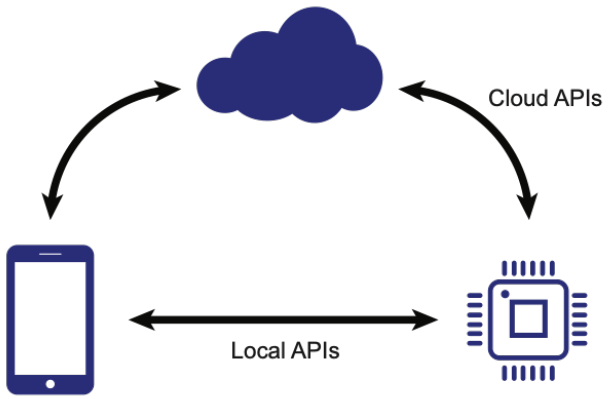

Figura 2.6: Weave API[23]

Quando viene realizzato un dispositivo che utilizza Weave, è necessario definire uno specifico schema di quello che lo stato del dispositivo rappresenta, dichiarandone i comandi accettati e tutti i possibili stati di funzionamento: ad esempio per un pulsante, lo stato può essere *on* o *off*, aggiungendo opzionalmente *brightness* nel caso in cui si comandi una lampadina dimmerabile.

Brillo fornisce, oltre a Weave, altri servizi ereditati da ChromeOS come Metrics e Crash Report, i quali forniscono in tempo reale informazioni sui dispositivi posizionati, quale versione del firmware hanno e quali comandi gli sono stati inviati da parte dell'utente. Inoltre possono fornire informazioni sullo stato di salute dei dispositivi in modo da consentire riparazioni ed analisi dei crash report in tempi ragionevolmente minori. OTA è un altro servizio

fornito da Google Brillo, il quale consente di preparare una nuova immagine, caricarla sulla Google Developer Console ed automaticamente propagarla a tutti i dispositivi attualmente connessi.

Per quanto concerne aspetti di sicurezza, per merito della somiglianza di Brillo al sistema operativo nativo Android, vengono ereditate alcune importanti caratteristiche, tra cui: sandboxing e process isolation. All'interno del Bootloader di Brillo è presente un meccanismo che assicura l'autenticità e la sicurezza delle immagini scaricate, in questo modo, nel caso in cui venga rilevato un problema, il sistema non carica l'immagine in memoria e semplicemente esegue un riavvio sulla vecchia versione del firmware. Viene tenuto conto degli aspetti di sicurezza anche per ciò che concerne il livello di comunicazione, infatti l'accesso ad un dispositivo abilitato all'utilizzo di Weave è possibile solo mediante un account Google, cosi che tutti i meccanismi di sicurezza forniti da Google siano presenti. Inoltre tutto il traffico dati inviato mediante l'utilizzo del protocollo di comunicazione è dotato di una criptazione end-to-end, sia nel caso di comunicazione diretta tra device, sia tra device e Cloud.

Samsung Smart Things ed Home Assistant Fino ad ora abbiamo descritto produttori, quali Apple e Google, che con l'obbiettivo di espandere e migliorare la propria visione dell'IoT e della smart home, hanno realizzato sistemi chiusi, a discapito dell'utilizzatore finale, in quanto più costosi e con minor disponibilità di prodotti. Analizziamo ora altri produttori, quali ad esempio Samsung, che hanno adottato un approccio radicalmente differente: la scelta è stata quella di realizzare un Hub centrale (hardware o software) connesso alla propria rete domestica, in grado di utilizzare tutti i differenti protocolli di comunicazione per l'IoT (se open source), con lo scopo finale di integrare il maggior numero di dispositivi smart possibile.

Primo fra tutti, analizziamo il sistema di Samsung: SmartThings. Questo sistema oltre a supportare tutti i dispositivi prodotti dalla casa madre (da prodotti per la cucina, all'intrattenimento ecc.), supporta una grande varietà di produttori di oggetti IoT, tra cui: Honeywell, Osram, D-Link ed altri. SmartThing richiede la connessione di un proprio hub (necessariamente fisico sotto due forme: router o vero e proprio hub) alla propria rete domestica, al quale verranno successivamente collegati tutti i dispositivi smart all'interno dell'abitazione. L'Hub smisterà tutte le informazioni in transito sulla rete, indirizzandole correttamente, eventualmente appoggiandosi su servizi cloud, per controlli dall'esterno dell'abitazione. Samsung supporta ZigBee e Z-wave ed è prevista l'integrazione del protocollo bluetooth all'interno del proprio hub. Inoltre, nel caso in cui venga meno l'alimentazione, l'hub di seconda generazione dispone di 4 batterie in grado di garantire anche fino a 10 ore di autonomia continuativa.

Home Assistant è un hub o un software per la domotica, sviluppato e progettato come alternativa completamente open source a sistemi quali Samsung SmartThings ed altri. Nativamente questo software è eseguito localmente, ma è possibile rendere l'istanza di HA accessibile da remoto: l'approccio più comune è quello di impostare l'instradamento per qualsiasi porta, dal tuo router alla porta 8123 sul computer che ospita Home Assistant. Questo approccio può risultare non efficiente, in quanto determinati ISP (Internet Service Providers) forniscono unicamente indirizzi IP dinamici, causando la perdita dell'accesso ad Home assistant dall'esterno. Problema però risolvibile utilizzando un servizio di DNS dinamico, come DuckDNS. Hassio, nome originario di questa tecnologia, può essere eseguito su un server locale, o su un Raspberry PI, necessariamente connesso ad internet. I benefici di un sistema completamente Open Source sono diversi [27]:

- Velocità: quando un'attività viene attivata, nessuna informazione viene obbligatoriamente inviata ad un Web Server, per questo motivo le operazioni sono svolte con velocità molto maggiore.
- Compatibilità: non sfruttando alcun servizio cloud di terzi, i rischi dell'esclusione di un proprio prodotto dal mercato, sono notevolmente ridotti.
- Sicurezza: con home assistant, un minor numero di dispositivi è connesso ad internet, quindi sono presenti meno dispositivi che permettono l'accesso ad internet sulla nostra rete.
- Controllo: utilizzando hub locali ed open source, si ha completa libertà di decidere come utilizzarlo.
- Affidabilità: molte circostanze mettono a rischio il funzionamento dei dispositivi connessi al cloud (server down, cloud service down, internet down). Utilizzando un hub locale per mantenere e controllare i dispositivi all'interno dell'ambiente domestico, questi problemi non possono essere presenti.

2.5 Considerazioni

Come descritto nei capitoli precedenti, l'obbiettivo culmine dell'introduzione dell'IoT nell'automazione domestica, è l'espandibilità, la facilità d'utilizzo e la sempre più numerosa disponibilità di prodotti smart per fini domestici. Con l'avvento delle nuove tecnologie, ad esempio HomeKit e Brillo, il pericolo maggiore va direttamente ad opporsi agli obbiettivi della primaria introduzione dell'IoT: si rischia di rimanere legati ed abbracciare un unico particolare

standard, vincolante per i produttori di oggetti e soprattutto per l'utente finale, in quanto questa scelta comporta inevitabilmente un aumento dei costi. Ad esempio è poco probabile che Apple, con HomeKit, investa nello sforzo di rendere compatibili standard di altri produttori con il proprio sistema proprietario, lasciando che altri dispositivi si possano connettere tramite Wi-Fi con eventuali problemi di sicurezza. Questo atteggiamento mantiene alti i prezzi e limita le possibili scelte dell'utente finale. Un altro aspetto peculiare, è la privacy: Apple ha politiche molto restrittive riguardanti la privacy dei propri utenti, mentre Google, ha basato il proprio business sul data mining. I sistemi proprietari Apple e Google, dispongono però di un grande vantaggio: viene fornita documentazione e linee guida ben realizzate e spesso parti intere di codice a disposizione da cui prendere informazioni, consentendo di ridurre, non di poco, il tempo necessario al rilascio di un nuovo prodotto.

Home Assistant è un sistema completamente software, non comprende quindi la necessità di sottoscrivere contratti per la realizzazione di prodotti compatibili, o l'appoggio del sistema su un Cloud di terzi, sottoposto a regole e rischi maggiori e restrittivi. L'obbiettivo di questo sistema Open Source è infatti l'adozione del maggior numero possibile di dispositivi e protocolli di comunicazione compatibili (se open source), riducendo i costi ed aumentando notevolmente le possibilità di scelta per l'utente finale.

Capitolo 3

Home Assistant

3.1 Architettura di Home Assistant

Home Assistant fornisce una piattaforma software completamente gratuita ed open source per il controllo e le automazioni di un ambiente domestico. Può essere descritto come un sistema incorporato che fornisce un'esperienza come quella di altri prodotti di consumo *off-the-shelf*: l'inserimento, la configurazione, l'aggiornamento e l'espansione del sistema mediante integrazioni aggiuntive, è tutto effettuato attraverso un'interfaccia user-friendly, facile ed intuitiva nell'utilizzo. Il sistema è composto da tre componenti fondamentali [7]:

- Il **Sistema Operativo**, che fornisce un ambiente Linux minimale e basilare per provvedere al corretto funzionamento dei componenti superiori.
- Il **Supervisor** che gestisce il sistema operativo sottostante.
- Il **Core** che interagisce e mette in comunicazione: l'utente, il supervisor ed i dispositivi e servizi IoT.

Nel corso di questo capitolo i tre componenti introdotti, verranno descritti e studiati in modo approfondito.

3.2 Architettura nel dettaglio

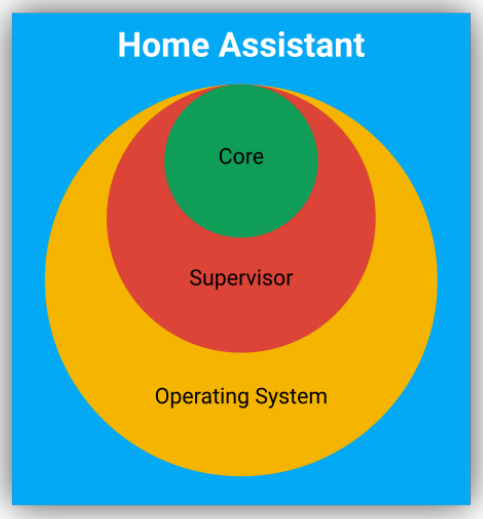

Figura 3.1: Schema architetturale di Home Assistant [7]

Come descritto al paragrafo precedente, il sistema in esame è sviluppato sfruttando un'architettura a livelli: Sistema Operativo, Supervisor e Core. In questa sezione analizzeremo nel dettaglio questi tre componenti.

3.2.1 Sistema Operativo

Il sistema operativo, posto alla base di Home Assistant (noto formalmente come HassOS) viene basato su una particolare distribuzione di Linux ed ottimizzato per ospitare Home Assistant e le relative integrazioni. Viene utilizzato un Docker come container, quindi il sistema viene eseguito in un ambiente isolato e sicuro, leggero e contenente tutte le informazioni necessarie al corretto funzionamento del sistema. Il Docker viene inoltre utilizzato anche dal Supervisor di Home Assistant, il quale utilizza questo container, per controllare il Core ed i componenti aggiuntivi, ognuno in un container separato.

Le principali caratteristiche di HassOS sono quindi elencate di seguito:

- Leggerezza ed efficienza nell'utilizzo della memoria.
- Operazioni di I/O minimizzate per maggiore leggerezza e reattività.
- Aggiornamenti OTA (Over The Air).

- Disponibilità di aggiornamenti offline.
- Sistema modulare utilizzando un Docker.

Bootloader All'interno di HassOS vengono utilizzati due bootloader differenti, per sfruttare al meglio le caratteristiche del dispositivo sul quale il software viene eseguito: BareBox e U-Boot. BareBox è un bootloader sviluppato ed ottimizzato per sistemi embedded, capace di essere eseguito su differenti architetture (ad esempio x86, ARM, MIPS, PowerPC ecc.). Si pone l'obbiettivo di versatilità e flessibilità, non solo per l'avvio di sistemi linux embedded, ma anche per sviluppi e studi iniziali sull'hardware utilizzabile. All'interno di HassOS viene utilizzato nei dispositivi che nativamente supportano il boot EFI.

U-Boot pone le fondamenta per il proprio su sviluppo su 10 golden rules, per perseguire i seguenti obbiettivi: leggerezza, velocità, semplicità, portabilità, configurabilità, usabilità, manutenibilità, bellezza e standard open source. All'interno di HassOS è utilizzato nei dispositivi che nativamente non supportano l'EFI boot.

Sistema Operativo Come detto nei paragrafi precedenti, viene utilizzato, come sistema operativo all'interno di HassOS BuildRoot LTS Linux.

Buildroot è uno strumento semplice, efficiente e facile da usare per sviluppare sistemi Linux embedded attraverso una procedura denominata: cross-compilation. La cross-compilazione è la tecnica mediante la quale si compila un codice sorgente con un cross-compilatore, ottenendo un file binario eseguibile su di un elaboratore, basato su un'architettura differente da quella della macchina sulla quale si è lanciato il cross-compilatore. Un caso d'uso di questa tecnica, vi è nel momento in cui è necessario compilare un programma per un sistema operativo differente dalla macchina su cui si trova il compilatore. Buildroot, include quindi una serie di file precompilati e patch, in grado di semplificare ed automatizzare il processo di sviluppo di un ambiente Linux completo per sistemi embedded, utilizzando allo stesso tempo la tecnica della cross-compilazione permettendo lo sviluppo di più piattaforme target su un singolo sistema di sviluppo basato su Linux. Buildroot può fornire automaticamente i tool necessari per la cross-compilazione, creare un file system di root, compilare un'immagine del kernel Linux e generare un boot loader per il sistema embedded target, oppure può eseguire qualsiasi combinazione indipendente di questi passi.

File Systems Su HassOS, vengono implementati due differenti file systems per sfruttare le capacità della macchina sulla quale il sistema operativo

viene lanciato: ShashFS per sistemi read-only e ZRAM per directory /tmp, /var e swap.

Squashfs è un filesystem compresso di sola lettura sviluppato per ambiente Linux. Utilizza la compressione zlib, lz4, lzo o xz per comprimere file e directory. Squashfs è inteso per un uso generale come filesystem di sola lettura, per l'archiviazione (cioè nei casi in cui può essere usato un file *.tar.gz), e in sistemi di dispositivi/memoria a blocchi limitati (ad es. sistemi embedded) dove è necessario un basso overhead.

ZRAM, invece, crea dispositivi a blocchi basati sulla RAM. Le pagine scritte su questi dischi sono compresse e memorizzate nella memoria stessa. Questi dischi permettono un I/O molto veloce e la compressione fornisce buone quantità di risparmio di memoria. Alcuni casi d'uso, quali ad esempio HassOS includono la memorizzazione /tmp, l'uso come dischi di swap, varie cache sotto /var ecc.

Container Platform I container sono una forma di virtualizzazione di sistemi operativi, possono essere usati per sviluppare e testare qualsiasi applicazione, in quanto dispongono al loro interno di tutti gli eseguibili, codice binario, librerie e file di configurazione necessari al corretto avvio del software. I principali vantaggi, nell'utilizzo di un container sono:

- Minor overhead: per l'esecuzione sono richieste minori risorse rispetto a macchine virtuali tradizionali, in quanto non vengono incluse immagini di un sistema operativo.
- Maggiore portabilità: applicazioni sviluppate all'interno di containers, possono essere facimente impiegate in diversi sistemi operativi e piattaforme hardware.
- Funzionamento più stabile: i team di sviluppo per applicazioni all'interno di un container, sanno che l'applicazione rimarrà stabile nel tempo, indipendentemente dall'ambiente d'esecuzione.
- Efficienza maggiore: i container permettono alle applicazioni di essere rilasciate, aggiornate e corrette più rapidamente.
- Miglior sviluppo dell'applicazione: viene supportata una progettazione agile, con conseguente aumento della velocità nello sviluppo, test e cicli di produzione.

All'interno di HassOS viene utilizzato un container Docker sul quale eseguire Home Assistant ed i suoi componenti. Oltre ai vantaggi di un container tradizionale, un Docker utilizza un'architettura client-server consentendo

un'ancor più marcata separazione dell'applicazione dall'infrastruttura. Il client scambia informazioni con il Docker Daemon, colui che svolge le mansioni cruciali riguardanti lo sviluppo, l'esecuzione e la distribuzione dei container Docker. Entrambi i componenti client e daemon possono essere eseguiti sullo stesso sistema, ma è anche possibile connettere un client ad un daemon connesso in remoto. La comunicazione avviene mediante una rest API, su socket UNIX o utilizzando interfacce network. Un ulteriore Docker client è il Docker compose, che consente lo sviluppo di applicazioni realizzate su un set di container.

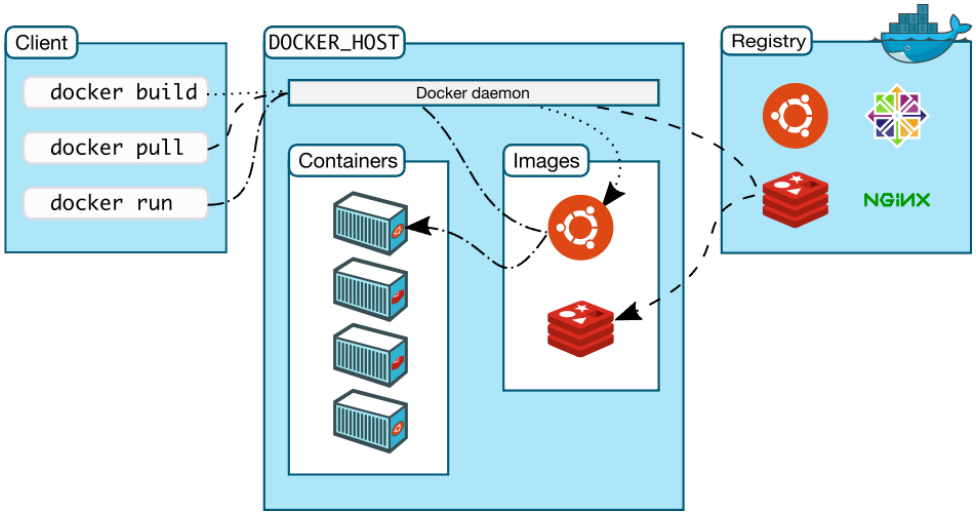

Figura 3.2: Architettura client/server di un container Docker

Di seguito un approfondimento sui ruoli svolti dai componenti architetturali di un Docker:

- Docker daemon: rimane in ascolto per richieste Docker API e gestisce oggi Docker come immagini, container, reti e volumi. Può inoltre communicare con altri daemons per gestire servizi Docker.

- Docker client: è il modo principale con cui molti utilizzatori di Docker interagiscono con un Docker. Quando utilizziamo comandi sul docker (dalla Docker API), il client invia questi al dockerd il quale li porta a destinazione. Un client può inoltre communicare con uno o più daemon.

- Docker registries: un registro Docker memorizza le immagini Docker all'interno di un hub Docker. Un Hub è un registro pubblico al cui interno sono memorizzate tutte le immagini dei Docker utilizzati. È possibile utilizzare anche registri privati.

- Docker objects: quando si utilizza un container docker, vengono create ed utilizzate immagini, container, reti, volumi, add-on ed altri oggetti.

3.2.2 Supervisor

Il supervisor è quella parte di Home Assistant, che permette all'utente di gestire in modo completo ed efficiente la propria istanza del software. Questo componente dell'architettura, ricopre le seguenti responsabilità nella gestione del sistema:

- Avvia ed esegue l'Home Assistant Core.
- Gestisce le procedure d'aggiornamento del Core.
- Esegue e ripristina backup dell'intero sistema.
- Gestisce i componenti aggiuntivi del sistema.
- Unifica le sorgenti e le uscite audio.
- Gestisce le procedure per l'aggiornamento di HassOS.

Le funzioni attualmente svolte dal Supervisor, sono quindi le seguenti:

- **Crash reporting**: il Supervisor può condividere con il team di sviluppo di HA, dati diagnostici anonimi, per contribuire al miglioramento del sistema. L'opzione può ovviamente essere disabilitata e lo è per default. Grazie a questa funzione, da Settembre 2020, 25.000 eventi diagnostici sono stati esaminati, risolvendo fino ad oggi più di 80 potenziali problemi.
- **Funzione Watchdog per componenti aggiuntivi**: gli Add-ons permettono agli utenti l'esecuzione di applicazioni di terze parti sul proprio server. Mediante la funzione WatchDog, se attivata, il Supervisor controllerà il funzionamento dei componenti aggiuntivi, utilizzando le procedure indicate dallo sviluppatore. Nel caso in cui l'Add-on smetta di funzionare correttamente, il Supervisor lo riavvierà, indipendentemente dal motivo della sospensione (crash/sospensione manuale). Per questo motivo, il watch dog non conosce i motivi per cui un Add-on smetta di funzionare, quindi va abilitato solo nel caso in cui il componente sia stabile e correttamente configurato.
- **Network manager**: Tramite l'interfaccia del Supervisor è possibile visualizzare e modificare le impostazioni di rete di Home Assistant.

- **Observer plugin**: il Supervisor fornisce parte dei suoi servizi mediante plugins esterni. L'observer viene inserito per controllare a sua volta il Supervisor, mettendo a disposizione una porta di diagnostica sul server (standard: 4357). Ad esempio nel caso in cui perdessimo accesso al Supervisor, tramite questa porta potremo diagnosticare il problema e risolverlo.

- **Audio migliorato**: mediante l'utilizzo di un nuovo server audio basato su PulseAudio, tutti i componenti possono utilizzare l'audio anche contemporaneamente, sfruttando anche altoparlanti bluetooth.

3.2.3 Core

L'Home Assistant Core interagisce con: l'utente, il supervisor ed i dispositivi e servizi all'interno del sistema. Viene composto dalle seguenti parti:

- **Event bus**: facilita l'avvio e l'attesa di eventi, permettendo ad ogni integrazione di causare o rispondere ad eventi. Ad esempio ogni cambiamento di stato di una specifica entità, verrà annunciato mediante l'event bus come un evento di tipo state changed, contenente lo stato precedente e lo stato attuale.

- **State machine**: tiene traccia degli stati delle entità e lancia eventi state changed all'event bus, nel caso in cui lo stato di un'entità venga modificato.

- **Service registry**: rimane in ascolto, come listener, sull'event bus per eventi di tipo call service. Questa tipologia di eventi può essere utilizzata nelle automazioni del sistema, consentendo di richiamare uno specifico servizio di un'entità.

- **Timer**: invia eventi di tipo time changed ogni secondo all'event bus, per comunicare e tenere traccia dello scorrere del tempo.

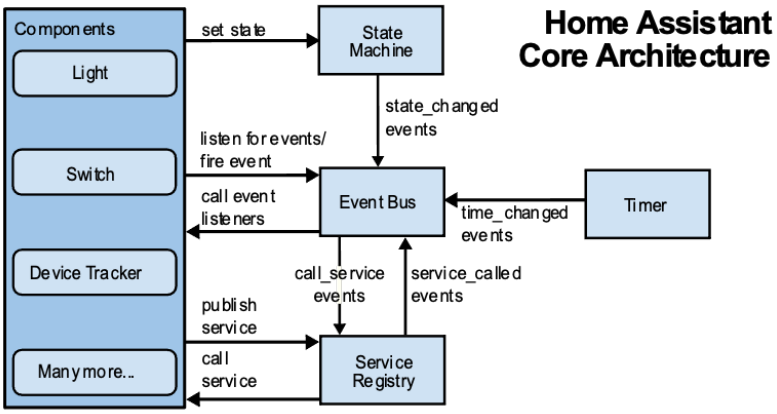

Figura 3.3: Home Assistant Core Architecture

3.3 Struttura di Home Assistant: Componenti ed Integrazioni

Il **Core** di Home Assistant può essere esteso e migliorato mediante le Integrazioni. Ogni integrazione è responsabile per uno specifico dominio all'interno di Home Assistant, fornendo la corretta logica di funzionamento per specifiche funzionalità. Per introdurre in modo corretto il concetto di integrazione, è però necessario fornire alcune definizioni di base, utilizzate in Home Assistant [**?**]:

- **Dispositivo**: tipicamente è un oggetto fisico. È l'oggetto reale, che si vuole automatizzare e controllare mediante la domotica.
- **Componente**: è un modello software idealizzato di un dispositivo. Idealmente, sono i mattoni virtuali utilizzati in Home Assistant per creare una rappresentazione virtuale e fedele, di tutti i dispositivi presenti all'interno dell'abitazione. Ad esempio, un componente, può essere:
  - Switch: questo componente modella un dispositivo che, in modo elementare, può essere acceso o spento. Questo include prese elettrice, relays ecc.
  - Luce: viene modellato un dispositivo simile allo Switch, ma oltre a poter essere acceso e spento, dispone di altre proprietà come la possibilità di regolare l'intensità luminosa o il colore. Vengono inclusi dispositivi di illuminazione a Led, lampadine ecc.
  - Cover: modella un dispositivo che può essere aperto o chiuso, includendo: porte dei garage, tende da sole, tapparelle ecc.

  - Climate: modella dispositivi che controllando il riscaldamento e/o il raffreddamento dell'abitazione, quindi i termostati.
  - Sensori: modella un dispositivo che riporta un valore (numerico o testuale), includendo sensori di temperatura, di umidità, di pressione, monitoraggio del clima ecc.
  - Sensori binari: modella dispositivi che riportano uno stato binario (acceso/spento, aperto/chiuso, su/giù, movimento/no movimento), includendo sensori di movimento per una porta, sensori di contatto ecc.

  Ne sono presenti molti altri, come Ventole, Notifiche. È importante sottolineare il fatto che un dispositivo complesso, potrebbe richiedere più componenti per modellarlo completamente su Home Assistant.

  Un Termostato Ecobee, ad esempio, viene modellato utilizzando un componente Climate, un Sensore, un Sensore Binario, Notifiche e Condizioni meteo.

- **Integrazione**: fornisce i mezzi per connettersi e controllare un dispositivo, nonché per modellare il dispositivo in questione, utilizzando uno o più componenti. L'integrazione Z-Wave comunica con i dispositivi abilitati al protocollo di comunicazione Wireless omonimo, rappresentandoli tramite componenti come Luce, Ventilatore, Serratura, Climate, Sensore Binario ecc.

- **Entità**: rappresenta un'istanza di un componente. Ad esempio se abbiamo dieci lampadine fisiche, ed ognuna viene modellate con un componente Luce, ogni istanza di questo componente è un'entità. Avremmo quindi dieci entità per dieci lampadine. È possibile avere differenti entità per uno stesso dispositivo, nel caso in cui, quel dispositivo venga modellato mediante più di un componente. Ad esempio il termostato Ecobee, necessiterà della creazione di numerose entità per un solo dispositivo fisico: un'entità per il componente Climate, una per il componente Sensore, una per il componente Notifiche ecc.

- **Piattaforma**: viene associata alla definizione di entità. Se una nostra lampadina, viene controllante mediante l'integrazione ZigBee, l'entità creata per quella lampadina, utilizza la piattaforma ZigBee.

Ora analizziamo lo schema sotto riportato, il quale indica un esempio di schema di funzionamento per poter automatizzare l'accensione di una luce in seguito alla rilevazione di movimento da parte di un sensore collocato in una specifica area.

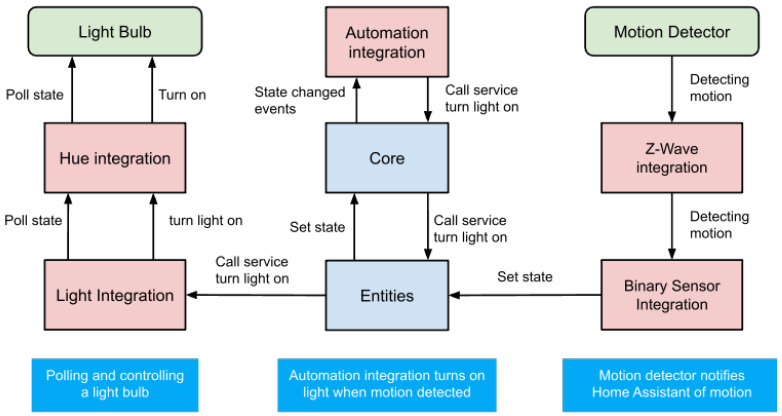

Figura 3.4: Architettura delle integrazioni su Home Assistant [7]

Indichiamo innanzitutto i dispositivi fisici: la lampadina e il sensore di movimento.

Per poter controllare questi dispositivi, sfruttiamo inizialmente due integrazioni, che ci forniscono le funzionalità necessarie alla connessione ed al controllo di particolari classi di dispositivi:

- Per il motion detector utilizziamo l'integrazione Z-Wave che permette di osservare e controllare dispositivi connessi mediante l'omonimo protocollo di comunicazione wireless.
- Per la lampadina, considerando il fatto (a titolo di esempio) che appartenga alla catena Philips Hue, sfruttiamo l'integrazione apposita chiamata Hue, la quale permette di controllare e monitorare tutti i prodotti appartenenti alla medesima linea.

In seguito, l'integrazione Z-Wave, consente di rappresentare il sensore di movimento, mediante la creazione di un Sensore Binario, il quale assumendo solamente due valori (presenza/non presenza o 0/1), ci indica la presenza o meno di movimento nell'area predisposta. L'entità centrale, rappresenta quindi, l'istanza del sensore binario per quanto riguarda il sensore di movimento, e l'istanza della Luce, per quanto riguarda la lampadina. Per la lampada, sfruttiamo come detto precedentemente l'integrazione Hue, per connetterci e controllarla, sfruttando il componente Luce. Generalizzando i concetti appena espressi, si può altrettanto dire che l'entità creata per il sensore di movimento utilizza la piattaforma Z-Wave, mentre l'entità creata per la lampadina utilizza la piattaforma Hue.

Home Assistant, fornisce *out-of-the-box* numerose integrazioni pre-inserite. Il supporto alle integrazioni viene fornito dall'attiva community essendo un

applicativo Open Source, ed attualmente è possibile l'inserimento fino a 1800+ integrazioni, tra cui: Alexa, Google Assistant, ESPHome, Ikea, Plex, Philips Hue, Zigbee ecc. Di seguito vengono approfondite le principali integrazioni parte di Home Assistant: Automazioni, Template, MQTT ed HACS

3.3.1 Automazioni e script

Le automazioni all'interno di Home Assistant permettono di rispondere a determinati eventi con un comportamento pre-configurato dall'utente (tramite apposita interfaccia). Ad esempio è possibile accendere le luci esterne al tramonto o mettere in pausa la musica quando si riceve una chiamata.

Le automazioni *blueprints* sono automazioni pre-costruite che possono facilmente essere aggiunte alla propria istanza di Home Assistant. Queste ultime necessitano solamente di una configurazione guidata per poter essere utilizzate. Per ricercarle è possibile sfruttare la community ed i forums di Home Assistant, continuamente aggiornati dagli utenti più attivi.

Per quanto concerne la struttura di un'automazione, è composta da un *trigger*, cioè un evento che causa l'avvio dell'automazione ed un *action* cioè uno o più regole scritte dall'utente, attivate dal trigger. Opzionalmente è presente una condizione, la quale limita i casi d'attivazione dell'automazione. Prendiamo come esempio la seguente automazione:

Quando paolo arriva a casa dopo il tramonto: accendi le luci nella sala da pranzo.

Secondo le componenti prima esplicitate, possiamo suddividere questa automazione in tre parti:

- Trigger: *Quando paolo arriva a casa.*
- Condizione: *Dopo il tramonto.*
- Action: *Accendi le luci nella sala da pranzo.*

Il *trigger* descrive perciò gli eventi che dovranno scatenare l'automazione, in questo caso una persona che arriva a casa. Un'informazione di questo tipo può facilmente essere recuperata monitorando il cambiamento dello stato della persona da *not home* ad *home*.

Le condizioni, sono opzionali, e limitano lo scatenarsi della automazioni in specifici casi d'uso.

La terza parte, *action*, viene richiamata solo nel caso in cui un *trigger* viene richiamato e tutte le condizioni sono soddisfatte. Ad esempio una azione può accendere una luce, modificare la temperatura ecc.

Gli script, sono componenti del sistema, che derivano in modo diretto dalle automazioni: sono composti in modo del tutto similare, ad esclusione del componente *trigger*. Tale componente non è infatti presente, in quanto gli script possono essere utilizzati (anche in combinazione con una automazione) per l'esecuzione di una o più azioni con eventualmente l'inserimento di alcune condizioni, senza però avere la possibilità di un loro avvio, eseguito da parte dell'ambiente nel quale il sistema domotico opera.

3.3.2 Template

Un template consente la creazione di entità che derivano le proprie informazioni, da quelle di altre entità già presenti all'interno di Home Assistant. Questo è compiuto specificando un template per le proprietà di un'entità, come il nome o lo stato.

Prendiamo ad esempio i sensori basati sullo stato di una specifica entità: ad esempio è visibile nell'immagine sottostante un template che calcola la media dei valori di due sensori, assegnandola ad un terzo sensore.

```yaml
template:
  - sensor:
      - name: "Average temperature"
        unit_of_measurement: "°C"
        state: >
          {% set bedroom = states('sensor.bedroom_temperature') | float %}
          {% set kitchen = states('sensor.kitchen_temperature') | float %}

          {{ ((bedroom + kitchen) / 2) | round(1) }}
```

Figura 3.5: Esempio di un template

3.3.3 MQTT (Message Queue Telemetry Transport)

MQTT è un protocollo di connessione e scambio di informazioni tipico dell'IoT, basato sull'utilizzo del protocollo di rete e di trasporto TCP/IP. Viene utilizzato all'interno del mondo IoT in quanto estremamente pratico, semplice ed efficace nell'utilizzo per lo scambio di dati tra oggetti connessi. In sostanza, i mittenti inviano messaggi relativi ad argomenti specifici, i destinatari si iscrivono ai temi che trovano interessanti e i broker provvedono alla trasmissione dei messaggi tra le due parti. Sia i mittenti che i destinatari sono client MQTT, e possono comunicare solamente attraverso un message broker. I broker gestiscono la ricezione ed il successivo invio di messaggi ai subscribers (sottoscrittori), provvedendo allo stesso tempo alla gestione delle liste di argomenti a cui i destinatari sono interessati. Secondo il modello publish-subscribe, sul quale MQTT si fonda, vengono definite due tipologie di entità nella rete: i client ed un broker.

Un broker è un server, che gestisce tutto il traffico dati in rete, indirizzando le informazioni ai relativi destinatari, mentre un client è un qualsiasi oggetto IoT in grado di interagire con il broker per inviare e ricevere messaggi. Il protocollo di comunicazione è quindi il seguente:

1. Il client si connette al broker e può sottoscriversi ad un qualsiasi argomento.

2. Il client pubblica i messaggi relativi ad un argomento inviandoli al broker.

Ogni messaggio di questo protocollo consta quindi di 2 sole parti: un'intestazione ed un payload, contribuendo alla leggerezza ed alla facilità d'utilizzo del software.

L'aggiunta di tale protocollo ad Home Assistant può essere introdotta mediante una apposita integrazione dotata di interfaccia grafica, scegliendo innanzitutto un broker, una porta sulla quale connettersi, eventuali username e password ed un opzionale certificato di sicurezza nel caso in cui i messaggi scambiati siano criptati.

3.3.4 HACS (Home Assistant Community Store)

HACS è un'integrazione, parte di Home Assistant. Come sappiamo dai paragrafi precedenti, l'aggiunta di componenti esterni è semplificata dall'utilizzo della UI fornita da Home Assistant, ma la gestione di questi componenti esterni rappresenta ancora il tallone d'achille per molti. A tal proposito, interviene HACS: un manager per installare e mantenere in stato di funzionamento componenti aggiuntivi sul sistema. In particolare, verrà verificata la presenza di aggiornamenti sui componenti installati ogni 30 minuti e ad ogni avvio, ed eventuali aggiornamento saranno visibili nello stato del sensore che viene aggiunto alla creazione di HACS sul proprio sistema. Viene inserito un sensore denominato *sensor.hacs*, e lo stato di questo sensore indicherà il numero di aggiornamenti in attesa di essere eseguiti. Inoltre, quando HACS opera sul proprio sistema, mediante l'event bus, lancia eventi ai quali è possibile registrarsi per il loro utilizzo nelle automazioni. Gli eventi sono di tue tipologie [6]:

- Hacs/update

- Hacs/repository

Come introdotto precedentemente è possibile utilizzare gli eventi lanciati da HACS in alcune automazioni. In particolare, HACS, include due automazioni:

- New repository added: crea una notifica a tutti i dispositive collegati all'istanza di Home Assistant, indicante la repository appena aggiunta ad HACS.

- Updates pending: viene creata una notifica persistente su Home Assistant indicante le integrazioni che necessitano di essere aggiornate.

3.4 Autenticazione

Il sistema di autenticazione, presente nativamente, in Home Assistant, rende sicuro l'accesso al proprio server.

Al primo avvio del sistema, viene creato l'account proprietario di quella specifica istanza. Questo account dispone di alcuni privilegi per la gestione dell'istanza, tra cui [7]:

- Creare e gestire altri account.

- Configurare le integrazioni attive ed eventualmente inserirne di nuove ed altre impostazioni riservate.

Dalla pagina di gestione del proprio account user o owner, è possibile svolgere le seguenti operazioni:

- Modifica della password.

- Abilitazione o Disabilitazione dell'autenticazione a più fattori.

- Rimozione di chiavi di sicurezza aggiornate (create nel momento in cui si accede con il proprio account su un nuovo dispositivo), rimuovendo l'accesso al proprio account da un determinato dispositivo.

- Creazione di chiavi d'accesso *Long Lived*, cosi che gli script possano interagire liberamente con Home Assistant.

- Effettuare il log out dalla propria istanza.

Architetturalmente parlando, il sistema si compone delle seguenti componenti, che analizzeremo nel corso di questo paragrafo:

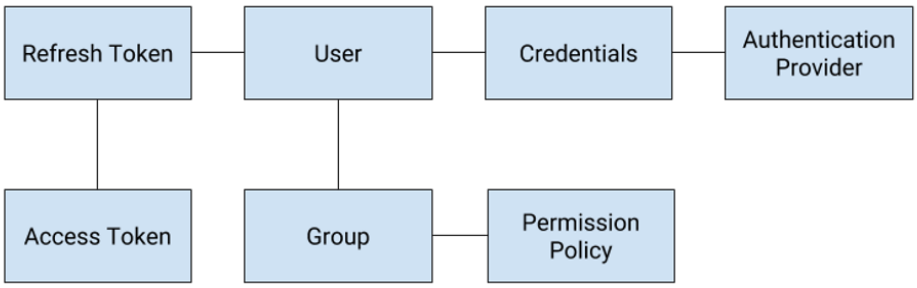

Figura 3.6: Schema autenticativo di Home Assistant [7]

Authentication providers Un authentication provider viene utilizzato dagli utenti per autenticarsi. Compito del provider è scegliere opportunamente quale metodo di autenticazione e quale backend utilizzare per verificare le credenziali inserite. Di default viene abilitato l'Home Assistant authentication provider, contenuto nativamente all'interno dell'installazione. In particolare, l'authentication provider che verrà utilizzato dalla propria installazione viene specificato all'interno del file *configuration.yaml*.

Credenziali Le credenziali memorizzano l'autenticazione di un utente con uno specifico authentication provider, vengono quindi prodotte nel momento in cui l'utente si autentica con successo. Permettono la ricerca di uno specifico utente all'interno di un sistema, e nel caso in cui non esista, ne verrà creato uno nuovo. Quest'ultimo non verrà attivato automaticamente ma necessiterà dell'approvazione da parte del profilo owner.

Utenti Ogni utilizzatore all'interno del sistema è un utente. Nel momento in cui, uno specifico utente effettua il log in, riceve un token d'accesso ed uno d'aggiornamento, in modo tale da poter effettuare richieste ad Home Assistant.

Owner L'utente creato al primo avvio dell'istanza di Home Assistant viene marcato come *owner* (proprietario). Il proprietario gode di alcuni privilegi rispetto ai normali utenti, ad esempio può gestire gli altri utenti inseriti ed avrà sempre accesso a tutti i permessi del sistema.

Gruppi Gli utenti del sistema, appartengono ad uno o più gruppi. L'appartenenza ad un gruppo determina i permessi di cui quello specifico utente gode.

Permission policy La permission policy, all'interno di un sistema d'autenticazione descrive a quali risorse, un determinato gruppo di utenti, può avere accesso. Sarà approfondita nel corso di questo capitolo.

Token d'accesso e d'aggiornamento Un token è un insieme di informazioni, contenente la minor quantità di dati, tali da facilitare il processo di determinazione dell'identità di un utente o consentirne l'accesso a determinate operazioni. All'interno di Home Assistant sono presenti per il funzionamento corretto del sistema d'autenticazione, due tipologie di token: *access* tokens e *refresh* tokens.

Applicazioni esterne che desiderano interagire con Home Assistant, necessitano di un codice d'autorizzazione, fornito da un utente autorizzato. Questo codice permette la richiesta dei token.

Gli *access* token, vengono generati dall'authentication provider, nel momento in cui un utente effettua il log in. Questo artefatto, permette inoltre ad applicazioni esterne di effettuare operazioni scambiando informazioni con l'API del server. Quando un'applicazione esterna ha bisogno di accedere a risorse protette su un server per conto di un utente, il token di accesso permette all'applicazione di segnalare al server che ha ricevuto l'autorizzazione dall'utente per eseguire certi compiti o accedere a determinate risorse. In particolare il contenuto di un token d'accesso è il seguente:

```
{
  "iss": "https://YOUR_DOMAIN/",
  "sub": "auth0|123456",
  "aud": [
    "my-api-identifier",
    "https://YOUR_DOMAIN/userinfo"
  ],
  "azp": "YOUR_CLIENT_ID",
  "exp": 1489179954,
  "iat": 1489143954,
  "scope": "openid profile email address phone read:appointments"
}
```


Figura 3.7: Contenuto di un token d'accesso

Il token d'accesso agisce perciò come delle credenziali per accedere a risorse protette, piuttosto che per l'identificazione di uno specifico utente. Un token di questo tipo, è caratterizzato dall'avere un periodo di validità limitato, dopo il quale può essere rigenerato con un *refresh* token.

Un *refresh* token, è quindi una credenziale digitale che permette ad una applicazione esterna di ottenere un nuovo *access* token senza il bisogno di

effettuare una richiesta ad un utente autorizzato. Di conseguenza, un token di aggiornamento che ha una durata di vita molto lunga potrebbe teoricamente dare un potere infinito al portatore del token per ottenere un nuovo token di accesso per accedere alle risorse protette in qualsiasi momento. Il portatore del token di aggiornamento potrebbe essere un utente legittimo o un utente malintenzionato.

Vi sono tre differenti tipologie di *refresh* token:

- Normali: chiavi generate quando un utente autorizza un'applicazione esterna all'interazione con Home Assistant.

- *Access* token *Long Lived*: sono chiavi d'aggiornamento che estendono la durata di una chiave d'accesso. Sono creati internamente al sistema e non sono mai comunicati all'utente.

- Sistema: sono chiavi che possono essere generate ed utilizzate unicamente da entità di Home Assistant, ad esempio HassOS o Supervisor. Anche questi codici non vengono mai comunicati all'utente.

3.5 Le Entità

Le integrazioni all'interno di Home Assistant rappresentano e forniscono le funzioni per connettere e controllare specifici dispositivi (o servizi web). Le entità sono standardizzate dai componenti, rappresentano perciò istanze di questi ultimi. A loro volta, i componenti rappresentano dispositivi generici come luci, pulsanti ecc. Questi componenti, dispongono di differenti servizi per controllare i dispositivi che astraggono, ma un'integrazione può fornire anche dei propri servizi specifici nel caso in cui l'oggetto associato non appartenga ad alcun componente.

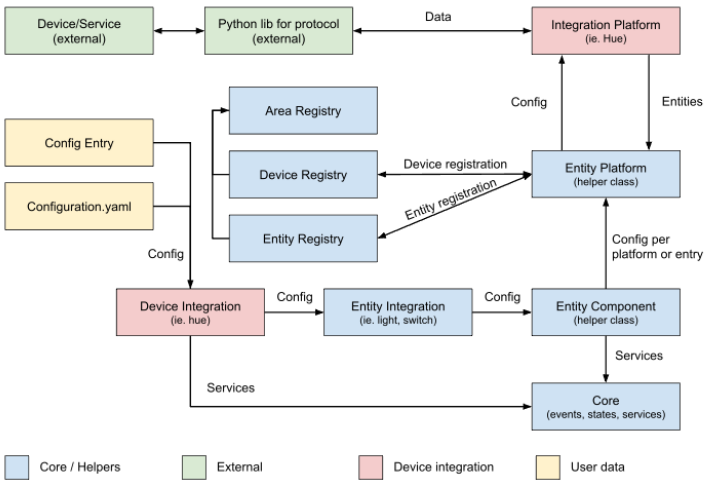

Figura 3.8: Configurazione per l'integrazione di nuovi dispositivi e servizi su Home Assistant

Un'entità, quindi, astrae il funzionamento interno di Home Assistant, permettendo a sviluppatori esterni che intendono integrare nuovi servizi e dispositivi, di estendere classi ed entità per il tipo di dispositivo o servizio voluto, senza doversi preoccupare del funzionamento interno a basso livello, costituito dai componenti.

L'integrazione di un accessorio esterno nel sistema (ad esempio Philips Hue) utilizzerà lo schema di funzionamento sopra riportato, per instaurare una connessione con il rispettivo dispositivo/servizio, avendo cosi la possibilità di controllarlo e monitorarlo.

Analizzando lo schema riportato, notiamo che il tutto prende vita dalla fase di configurazione dell'integrazione. Tale configurazione può essere fornita dall'utente mediante delle voci di configurazione, chiamate Config Entry o mediante una configurazione manuale, che può avvenire all'interno del file Configuration.yaml.

Le Config entries, sono dati di configurazione per una specifica integrazione, creati mediante l'interfaccia utente. I passi che compongono l'interfaccia, vengono definiti dallo sviluppatore mediante un Config Flow Handler. Quest'ultimo ha il compito, oltre che di definire gli step di configurazione, di gestire la creazione delle config entry partendo dalle informazioni fornitegli dall'utente mediante l'interfaccia o mediante altre risorse del sistema. Inoltre, gli handlers, controllando i dati memorizzati nelle Config entries, per questo motivo non c'è necessità di validare le configurazioni fornite, all'avvio di Home Assistant.

In seguito alla configurazione, mediante l'integrazione del dispositivo (*Device Integration*) si instaurerà una connessione con il dispositivo esterno, inol-

trando le config entries, per configurare le relative entità con i relativi componenti (Luce, Switch ecc.), eventualmente registrando servizi specifici non presenti nei componenti astratti.

A sua volta, il componente astratto (*Entity Integration*) delle entità create, definisce le classi astratte delle entità e/o dei servizi per il controllo delle entità create.

L'*entity component helper* è responsabile per la distribuzione delle config entries alle piattaforme utilizzate dalle entità, per la ricerca dinamica di nuovi dispositivi e per selezionare le entità per le service calls.

In seguito viene richiamato l'*Entity Platform Helper*, che gestisce tutte le entità per la piattaforma utilizzata al momento della configurazione, eseguendo eventuali procedure di aggiornamento. Inoltre questo componente, è responsabile per la registrazione delle entità e dei dispositivi nei rispettivi registri, mantenuti da Home Assistant.

Successivamente, la piattaforma d'integrazione di Home Assistant (ad esempio hue.luci) utilizza la configurazione fornita per l'interrogazione del dispositivo esterno per creare le entità volute. Determinate piattaforme possono inoltre registrare servizi specifici per determinate entità: questi servizi saranno disponibili in tutte le entità del dispositivo integrato (ad esempio tutte le entità luci delle philips hue inserite).

3.6 Tenere traccia di Entità, Dispositivi ed Aree

Home Assistant, tiene traccia dei dispositivi e delle aree nei quali sono installati e delle entità inserite nel sistema, mediante appositi registri, compilati al momento dell'aggiunta dei componenti esterni. Il registro delle entità, tiene traccia di tutte le entità presenti nel sistema. Ogni entità, è dotata di un proprio ID univoco, memorizzato all'interno di questo registro. La registrazione, mediante codice alfanumerico univoco, su tale registro, consente diversi vantaggi:

- La stessa entità otterrà sempre lo stesso ID.
- Altre entità per il motivo precedente, non potranno utilizzare un ID non univoco.

È importante sottolineare il fatto che, un utente, non può modificare l'ID univoco di un'entità, in quanto potrebbe comportare la perdita di tutte le config entries, relative a quello specifico ID. Inoltre un'entità viene rintracciata all'interno dell'apposito registro, basandosi sulla combinazione di: nome del componente (ad esempio Luce, Switch), nome dell'integrazione (ad esempio

Hue) e ID univoco dell'entità, perciò le entità non dovrebbero includere nell'ID il nome del componente o del dominio. Infine, se il dispositivo, ha un singolo ID univoco, ma fornisce più entità (ad esempio se un unico sensore misura temperatura ed umidità, fornisce due entità), l'ID dovrà essere obbligatoriamente formato come segue: unique id - sensor type.

Non sono invece accettabili come identificatori, i seguenti valori:

- Indirizzi IP.
- Nome del dispositivo.
- Nome dell'host.
- URL.
- Indirizzi e-mail.
- Nomi utente.

Home Assistant tiene traccia anche di tutti i dispositivi fisici, attualmente collegati al sistema, mediante un apposito registro dei dispositivi. Notiamo innanzitutto che un dispositivo può essere rappresentato in Home Assistant, da una o più entità a seconda delle funzioni svolte (si rimanda all'esempio del sensore di temperatura ed umidità).

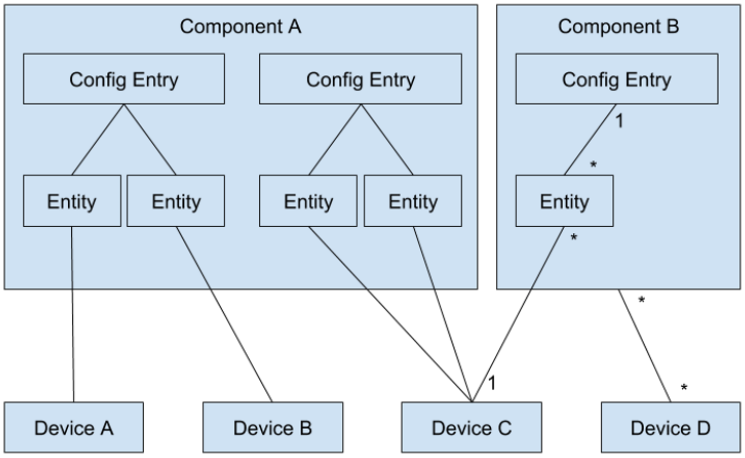

Figura 3.9: Architettura di controllo per componenti, entità e dispositivi [7]

Un dispositivo, su Home assistant, è un oggetto fisico, dotato di propria unità di controllo, che non deve necessariamente essere smart, ma dovrebbe

avere il controllo di ciò che accade ai suoi apparati. Ad esempio un termostato Ecobee, con 4 sensori per le stanze, equivale a 5 dispositivi su Home Assistant: uno per il termostato ed i sensori al suo interno ed uno per ogni sensore esterno. Ogni dispositivo esiste, perciò in un'unica area geografica, avendo eventualmente più di un input o output in quell'area. In particolare, un dispositivo, viene caratterizzato dai seguenti attributi:

- Id univoco (generato dal sistema)
- Nome
- Connessioni: un set di tuple formate da tipo di connessione, identificatore di connessione
- Identificatori per il dominio
- Costruttore, modello e versione firmware
- Area nel quale viene installato e relativo ID
- Eventuali config entries
- Tipologie di dati trasferiti: valore a scelta tra *None* e *Service*

Ogni entità, consente di rintracciare un dispositivo, mediante la proprietà *device info*. Questa proprietà viene letta quando l'entità viene aggiunta ad Home Assistant mediante una Config entry. Un dispositivo, viene quindi abbinato mediante gli identificatori forniti, ad un dispositivo esistente, e conseguentemente inserito nell'apposito registro.

Altrimenti, è possibile procedere con la registrazione manuale del dispositivo: procedura obbligatoria nel caso in cui non siano presenti entità che rappresentino il dispositivo: ad esempio un Hub che comunica con particolari lampadine smart.

Infine, Home Assistant tiene traccia anche delle aree, inserite nel sistema dall'utente, nelle quali sono presenti determinati dispositivi. Ogni area rappresenta quindi un luogo fisico e viene caratterizzata (all'interno dell'apposito registro) dai seguenti attributi:

- ID univoco
- Nome, assegnatogli dall'utente creatore, mediante apposita procedura di configurazione.

Mediante le aree è possibile, svolgere ed implementare automazioni in modo più efficiente, ad esempio, inserendo l'area *Cucina* è possibile implementare particolari azioni specifiche per quell'area: si semplifica il funzionamento per l'accessione delle luci, richiamando tutte le lampadine presenti nella cucina, che si accenderanno. Senza l'utilizzo dell'area predisposta, si avrebbe dovuto rintracciare ogni entità di tipo Luce, all'interno di quella specifica Area, ed accenderla singolarmente.

Capitolo 4

Caso di studio

Home Assistant è un software gratuito e open-source per la domotica, progettato per essere il sistema di controllo centrale per i dispositivi dell'abitazione, con particolare attenzione al controllo locale ed alla privacy.

Sfruttando le conoscenze acquisite dalla dettagliata analisi dei capitoli precedenti, il seguente capitolo tratterà nel dettaglio lo sviluppo di un progetto basato su tale sistema. In particolare l'obbiettivo è quello di realizzare un apparato domotico esemplificato per mostrare le potenzialità del sistema mettendo in risalto l'utilizzo delle automazioni e le integrazioni delle differenti componenti mediante altrettanti differenti protocolli di comunicazione, più o meno specifici.

Si vuole domotizzare la gestione luminosa di un ambiente lavorativo dotato dei seguenti apparati domotici:

- Shelly Button: per consentire all'utente di fornire indicazioni relative al comfort fornito dal sistema.
- Shelly 1: relay domotico, che sostituisce un relay fisico per il comando di alcune tende motorizzate.
- Shelly Motion: sensore di movimento e di luminosità ambientale.
- Una lampada Philips hue per consentire l'aggiunta di luminosità artificiale se quella naturale non fosse sufficiente.

Si presta maggiore priorità alla luminosità ambientale, in quanto secondo alcuni studi, fornisce un maggiore confort anche visivo, oltre che per il risparmio energetico dell'illuminazione ambientale artificiale, anche se di secondaria importanza nel caso in esame.

4.1 Analisi dei requisiti

Diagramma dei casi d'uso In base a quanto esposto nell'introduzione del problema, si possono identificare differenti casi d'uso, con altrettante differenti reazioni da parte del sìstema domotico che dovrà adeguarsi all'ambiente nel quale lavora. Innanzitutto si tratta della luminosità in ambiente lavorativo, che secondo l'istituto nazionale per la sicurezza del lavoro, non deve essere inferiore a 500 lux, con una intensità preferibilmente compresa tra i 550 e 750 lux. Effettuando ipotesi, sull'ambito di lavoro, possiamo innanzitutto prevedere l'utilizzo su Home Assistant delle seguenti integrazioni: Hue, Weather ed MQTT.

Le seguenti integrazioni verranno utilizzate come segue:

- Mediante l'integrazione Philips Hue, potremo controllare la lampada installata, eventualmente modificando la luminosità della lampada stesso, se disponibile.

- Sfruttando l'integrazione Weather, sapremo la posizione del sole rispetto all'ufficio in cui ci troviamo, le condizioni meteo attuali e la luminosità esterna attuale.

- Infine, mediante l'ìntegrazione del noto protocollo di comunicazione, sfrutteremo gli accessori compatibili con tale sistema, in particolare nel caso preso in esame, accessori Shelly, tra cui: un sensore di luminosità e movimento ed un pulsante.

Il caso d'uso del sistema, che analizziamo, individua l'ingresso dell'utente nella stanza. Successivamente il sensore installato rileva il movimento nella stanza, comandando, successivamente ad alcune scelte legate alla logica di funzionamento, gli attuatori per le tende o la lampada installata.

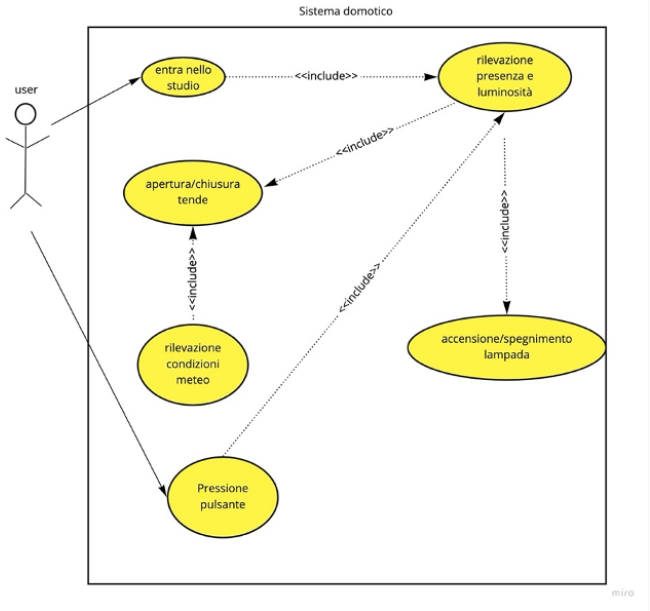

Figura 4.1: Diagramma dei casi d'uso per il sistema in esame

Come si può notare, l'apertura o la chiusura delle tende motorizzate, non è solamente vincolato alla rilevazione della luminosità con l'ingresso dell'utente, bensì vengono prese in considerazione anche le condizioni meteorologiche, in quanto con condizioni avverse, la luminosità non sarebbe sufficiente e provocherebbe eventuali distrazioni dell'utente.

Inoltre l'utente ha a disposizione un pulsante, da poter premere, nel caso in cui la luminosità non sia correttamente impostata. Premendo infatti, il pulsante, verrà riavviata la procedura per aprire/chiudere le tende ed eventualmente accendere/spegnere la lampada.

Logica di funzionamento Ora passiamo alla descrizione della dettagliata di funzionamento del sistema, per fornire delle basi sulle quali costruire le dovute automazioni, mostrata nel dettaglio, nella figura sottostante.

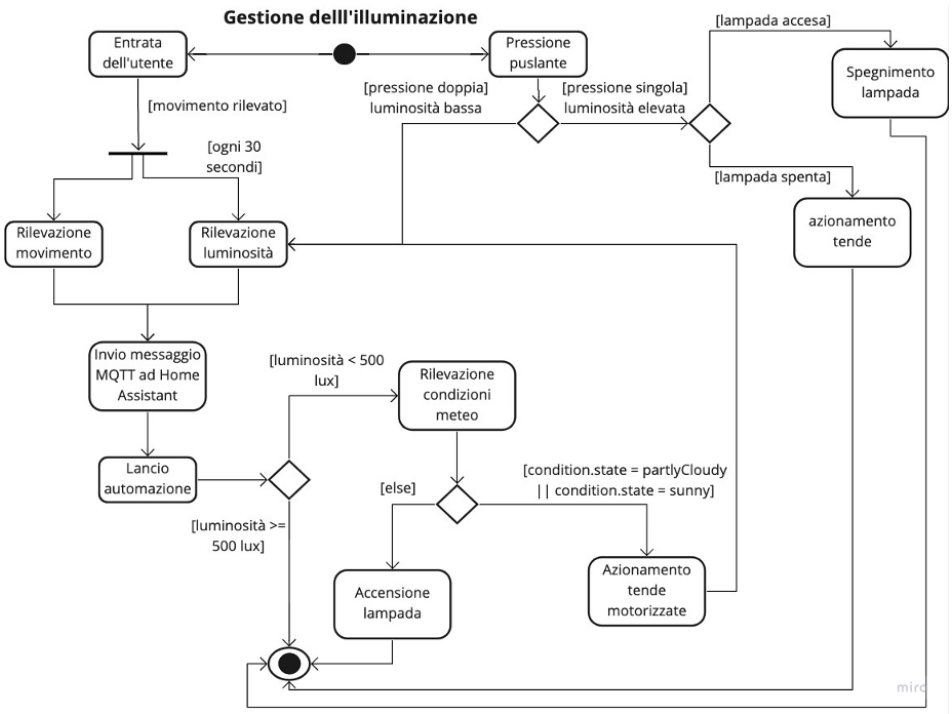

Figura 4.2: Diagramma di attività, mostrante il dettagliato funzionamento del caso di studio

Innanzitutto, individuiamo i dispositivi a disposizione e le relative responsabilità o capacità:

- Shelly Motion: sensore collegato al sistema mediante il protocollo MQTT con il quale è nativamente compatibile, mediante il quale è possibile rilevare movimento in un determinato ambiente, opzionalmente richiedendo il valore in lux rilevato.

- Shelly Button: bottone disponibile per l'utente, sfruttato anch'esso mediante il protocollo MQTT. Inserito in Home Assistant come sensore binario, in quanto restituisce solamente valori 1 o 0. Consente all'utente di fornire un feedback passivo sul funzionamento del sistema.

- Philips Hue: Lampadina appartenente all'omonima linea di prodotti. Integrata mediante la piattaforma Hue.

- Shelly 1: relay domotico. Installato nell'ambiente collegato alle tende robotizzate. Fornendo comandi al relay, sarà quindi possibile aprire o chiudere le tende, senza però possibilità di regolazione dell'apertura o della chiusura.

L'ingresso dell'utente verrà quindi rilevato dal Shelly Motion, il quale mediante il protocollo MQTT invierà un messaggio ad Home Assistant, che sarà in ascolto sul canale di comunicazione predisposto. Alla ricezione del messaggio, contenente un timestamp e il valore in luminosità rilevato (oltre che alla conferma di avvenuto movimento), verrà attivata un'automazione, la quale:

- Verificherà mediante l'integrazione Weather, le condizioni meteorologiche e in caso queste ultime siano favorevoli, provvederà all'apertura delle tende.
- Successivamente si attenderà il messaggio periodico del Shelly Motion indicante la luminosità ambientale, consentendo, in seguito alla lettura del messaggio, l'eventuale accensione della lampada Philips Hue, se la luminosità non fosse sufficiente.

Nel caso in cui, l'utente non fosse soddisfatto della luminosità ambientale, per un rapido cambiamento meteoreologico o altri fattori soggettivi, potrà premere il pulsante apposito Shelly Button. Il sensore, invierà un messaggio MQTT ad Home Assistant, indicante l'avvenuta pressione. Questo consentirà l'avvio di una seconda automazione, la quale ricontrollerà le condizioni meteoreologiche (eventualmente chiudendo le tende) ed in seguito all'attesa del messaggio del Shelly motion, accenderà o spegnerà la lampada.

Per quanto concerne lo spegnimento delle luci e la chiusura delle tende in periodo di inattività, verrà predisposta una automazione, la quale dopo che per 10 minuti non siano stati ricevuti messaggi di avvenuto movimento da parte di Shelly Motion:

- Provvederà allo spegnimento della lampada.
- Avvierà la chiusura motorizzata delle tende inviando opportuni messaggi al sensore Shelly 1.

4.2 Integrazione dei dispositivi: MQTT e Hue

Per poter procedere alla realizzazione delle automazioni, cuore del sistema in esame, è innanzitutto necessario predisporre i sensori che si vogliono utilizzare.

Mediante Home Assistant, procediamo all'installazione delle integrazioni MQTT e Philips Hue, le quali permettono di ottenere i mezzi necessari alla comunicazione con i dispositivi esterni. Noteremo che l'integrazione MQTT, dispone di un proprio config flow, per configurare correttamente l'ambiente di lavoro, come il seguente:

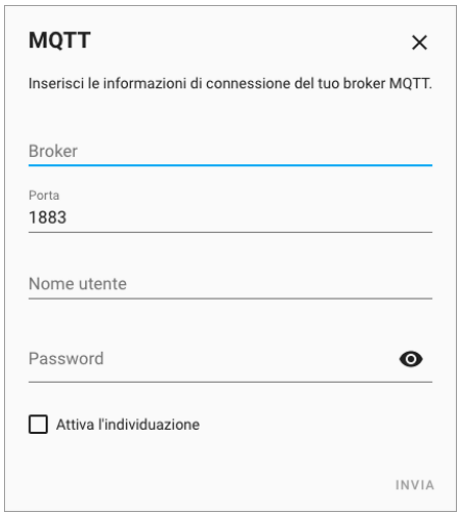

Figura 4.3: MQTT Config flow

Inserendo le opportune informazioni, potremo connetterci al Broker che smisterà le informazioni e i dati in transito dai sensori shelly. In particolare, verrà create, la seguente config entry, memorizzata all'interno del file *configuration.yaml*:

```yaml
# Example configuration.yaml entry
mqtt:
  certificate: "PATH_TO_CA.crt"
  broker: "IP_ADDRESS_BROKER"
  port: 8883
  username: "MQTT_USERNAME"
  password: !secret MQTT_PASSWORD
```

Figura 4.4: MQTT Config entry

Analizzando le seguenti informazioni di configurazione, notiamo le informazioni che potevano essere inserite mediante interfaccia grafica fornita, ed un opzionale certificato di autenticazione per potersi collegare al proprio broker utilizzando un livello addizionale di sicurezza sul protocollo MQTT.

Ora possiamo procedere alla configurazione manuale dei dispositivi Shelly, con i quali decidiamo di comunicare mediante il protocollo descritto precedentemente.

Procediamo alla configurazione del sensore Shelly Motion. Questa tipologia di sensore consente di ottenere dai messaggi in entrata (opzionalmente temporizzati) informazioni oltre che sulla presenza del movimento, anche sul timestamp del messaggio ricevuto. Dal messaggio in entrata potremo quindi individuare differenti sensori, richiedendo particolari porzioni del messaggio (importante notare il formato json del messaggio in entrata). In particolare, optiamo per la configurazione seguente:

```
  sensors:
- platform: mqtt
  name: “IsShellyMotionDetected”
  state_topic: "shellies/shellymotionsensor-60A4139929F8/status"
  value_template: "{{ value_json.motion }}"
- platform: mqtt
  name: “IsShellyMotionVibrationDetected"
  state_topic: "shellies/shellymotionsensor-60A4139929F8/status"
  value_template: "{{ value_json.vibration }}"
- platform: mqtt
  name: “ShellyMotionLux”
  state_topic: "shellies/shellymotionsensor-60A4139929F8/status"
  value_template: "{{ value_json.lux }}"
  unit_of_measurement: ‘lx’
- platform: mqtt
  name: “ShellyMotionBattery”
  state_topic: "shellies/shellymotionsensor-60A4239939F8/status"
  value_template: "{{ value_json.bat }}"
  unit_of_measurement: '%'
```

Figura 4.5: Configurazione Shelly Motion

Riferendoci al seguente messaggio esemplificativo come comunicazione Json del sensore Shelly Motion:

”bat”:”94”,”vibration”:”False”,”lux”:”208”,”motion”:”True”

Come possiamo notare, abbiamo inserito e configurato 4 sensori relativi ad un unico dispositivo esterno, collegato mediante l’integrazione MQTT:

- IsShellyMotionDetected ci fornisce indicazioni per la rilevazione del movimento, selezionando il valore motion del messaggio Json fornitoci dal sensore.
- IsShellyVibrationDetected fornisce anch’esso un valore booleano, indicando l’eventuale attivazione della vibrazione del dispositivo, alla rilevazione di movimento.
- ShellyMotionLux indica la luminosità rilevata dal dispositivo.
- ShellyMotionBattery indica la percentuale di batteria rimanente del dispositivo installato.

Analizziamo in dettaglio uno dei seguenti sensori, ad esempio: ShellyMotionbattery. Notiamo:

- L’indicazione della piattaforma con la quale il sensore è connesso ad Home Assistant.
- Un nome (in quanto l’identificatore univoco viene assegnato da Home Assistant e non può essere assegnato dall’utente)

- State topic indica il topic MQTT al quale si è iscritti per ricevere i valori del sensore.
- Value template è inserito per definire un template con il quale estrarre uno specifico valore dal messaggio ricevuto.
- Unit of measurement indica l'unità di misura con la quale rappresentare I valori ottenuti in seguito all'elaborazione del messaggio con il template fornito.
- Opzionalmente può essere indicata l'icona da utilizzare per la rappresentazione del sensore sull'interfaccia, in questo caso ad esempio: *icon: mdi:battery-level*.

La configurazione indicata dovrà essere inserita all'interno del file configuration.yaml o in un file predisposto, richiamato all'interno del file di configurazione principale, utilizzando la seguente dicitura:

sensor: !include sensor.yaml

Il file *sensor.yaml* dovrà essere formattato correttamente per essere riconosciuto, indicando un elenco come all'immagine 5.2.3, senza l'aggiunta di attributi non utilizzati.

L'aggiunta e la successiva configurazione degli altri sensori Shelly, sarà equivalente a quella appena descritta. L'installazione della lampada Philips Hue, differisce dalle precedenti invece, in quanto può essere completamente automatizzata mediante il proprio config flow fornito dall'omonima integrazione, il quale creerà le opportune config entry nel file di configurazione.

4.3 Studio e sviluppo delle automazioni

Successivamente all'aggiunta dei sensori voluti, si procede all'analisi approfondita delle automazioni necessarie.

La logica di funzionamento del sistema domotico sarà infatti contenuta all'interno delle automazioni, perciò ne realizzeremo due:

- User trigger: consentirà l'avvio della logica di funzionamento, quando il sensore isShellyMotionDetected cambierà il suo stato da *false* a *true*. In seguito, nel componente *action* verrà richiamata la seconda automazione.
- Set user lux: questa automazione verrà richiamata dalla precedente, e consentirà il set della luminosità ambientale secondo questo ordine:

  - Controllo delle condizioni meteo esterne, se favorevoli, si richiamerà una terza automazione che si occuperà dell'apertura delle tende mediante Shelly 1, altrimenti verrà richiamata la quarta ed ultima automazione, che si occuperà dell'accensione della luce nella stanza.

- User gone: automazione del sistema, la quale, dopo che siano trascorsi 10/15 minuti dall'ultima rilevazione di movimento del sensore IsShellyMotionDetected, provvederà alla chiusura delle tende ed allo spegnimento della lampada.

- Button pressed: consente la rilevazione della pressione dello Shelly Button da parte dell'utente, il quale fornendo un doppio click esprimerà luminosità troppo ridotta, mentre fornendo un click singolo, indicherà una luminosità troppo elevata.

Non vengono elencate eventuali automazioni di minore importanza, utilizzate e realizzate a supporto delle principali, indicate nell'elenco antistante.

User trigger Analizziamo nel dettaglio la seguente automazione, suddividendo i requisiti richiesti nei tre componenti principali *trigger*, *condition* e *action*.

- Trigger: Il grilletto dell'automazione, sarà costituito dal cambiamento di stato del sensore IsShellyMotionDetected, in particolare il suo cambiamento da *false* a *true*.

- Condition: Non sono presenti condizioni sulla seguente automazione.

- Action: L'azione sarà composta dall'utilizzo del servizio fornito dall'integrazione Automation di Home Assistant *automation.trigger* specificando mediante l'attributo entity id il nome dell'automazione, preceduto dal prefisso *automation*.

Procediamo alla scrittura del codice:

```
automation:
    alias: user_trigger
     # Call another automation when the user enter the room #
     trigger:
          - platform: state
            entity_id: sensor.isShellyMotionDetected
            from: false
            to: true
```

```
    condition: []
    action:
        - service: automation.trigger
          entity_id: automation.set_user_lux
```

Listato 4.1: Automazione user trigger

Notiamo il codice inserito, e come conclusione, il richiamo della second automazione *setUserLux*.

Set user lux Suddividiamo la seguente automazione nelle tre componenti fondamentali:

- Trigger: non sarà presente, in quanto questa automazione, viene eseguita sotto specifico comando dell'automazione precedente.
- Condition: non sarà presente, le condizioni per il richiamo delle due differenti automazioni successive, saranno incapsulate all'interno del componente Action.
- Action: svolgerà la funziona cardine di questa automazione: se le condizioni meteo fossero favorevoli, passerebbe il controllo all'automazione *open curtains* altrimenti passerà il controllo all'automazione *set light* per l'accensione della lampada.

```
alias: Set_user_light
description: []
trigger: []
condition: []
action:
  - choose:
      - conditions:
          - condition: state
            entity_id: weather.casa
            state: sunny
        sequence:
          - service: automation.trigger
            data: {}
            entity_id: automation.open_curtains
      - conditions:
          - condition: state
            entity_id: weather.casa
            state: partlycloudy
```

```
      sequence:
        - service: automation.trigger
          data: {}
          entity_id: automation.open_curtains
    - conditions: []
      sequence:
        - service: automation.trigger
          data: {}
          entity_id: automation.set_light
  default: []
mode: single
```

Listato 4.2: Automazione set user light

Come si può notare, i componenti *trigger* e *condition* non sono presenti, infatti la logica di questa automazione è presente solamente nel componente *action*. Al suo interno, mediante un'azione di tipo *choose*, si sceglie una delle tre opzioni inserite, a seconda delle condizioni meteo esterne (ottenute mediante l'integrazione Weather di Home Assistant). Nel caso in cui sia soleggiato o parzialmente nuvoloso si sceglie di aprire le tende, in tutti gli altri casi, si opta per l'accensione della luce mediante l'automazione *set light*.

Open curtains Con l'utilizzo di questa automazione, apriamo le tende dell'ufficio (motorizzate mediante l'attuatore Shelly 1 collegato con il protocollo MQTT) e rileviamo la luminosità dopo l'apertura. Nel caso in cui quest'ultima non sia sufficiente, si richiamerà l'automazione *set light*.

I tre componenti sono quindi organizzati come segue:

- Trigger: non presente perché l'automazione parte sotto chiamata di *set user light*.
- Condition: non presente.
- Action: include l'invio del comando di apertura a Shelly 1 e la chiamata di una nuova automazione, denominata *detect light*.

```
alias: open_curtains
description: []
trigger: []
condition: []
action:
  - service: mqtt.publish
```

```
    data_template:
        topic: shellies/shelly1-60AB45KWI/relay/0/command
        payload: on
  - delay: 00:01:00
  - service: mqtt.publish
    data_template:
        topic: shellies/shelly1-60AB45KWI/relay/0/command
        payload: off
  - service: automation.trigger
    data: {}
    entity_id: automation.detect_light
```

Listato 4.3: Automazione open curtains

Come possiamo notare, l'intera logica dell'automazione in considerazione è contenuta nelle azioni svolte. Mediante il servizio *mqtt.publish* è infatti possibile l'invio di precisi comandi all'attuatore Shelly 1 per aprire le tende (considerando nell'esempio un tempo di apertura pari a circa un minuto). In seguito viene richiamata la prossima automazione, la quale controllerà la sufficienza della luminosità ambientale, eventualmente accendendo la lampada integrata precedentemente.

Detect light Mediante questa automazione, rileviamo con l'ausilio del sensore di movimento e luminosità shelly, la luminosità dell'ambiente di lavoro. Se quest'ultima fosse minore di 550 lux, accendiamo la luce.

I componenti dell'automazione sono quindi strutturati come segue:

- Trigger: non presente.
- Condition: non presente.
- Action: comprende il richiamo dell'automazione *set light*, se la luminosità ambientale non è sufficiente.

```
alias: detect_light
description: []
trigger: []
condition:
  - condition: numeric_state
    entity_id: ShellyMotionLux
    below: 500
action:
```

```
  - service: automation.trigger
    data: {}
    entity_id: automation.set_light
```

Listato 4.4: Automazione detect light

In questo caso, utilizziamo l'operatore *condition* in quanto consente di semplificare il funzionmento dell'automazione, eseguendo le dovute e necessarie azioni solamente nel caso in cui siano necessarie. Per accendere la lampada sfruttiamo ancora una volta la modularizzazione del problema, fornendo un'ulteriore automazione apposita.

Set light Mediante la seguente automazione, accendiamo la luce, quando l'automazione viene richiamata. Viene considerata una lampadina non dimmerabile.

I componenti della seguente automazione sono perciò strutturati come segue:

- Trigger: non presente.
- Condition: non presente.
- Action: include il comando sulla luce Hue.light.

```
alias: set_light
description: []
trigger: []
condition: []
action:
  - data:
    entity_id: light.office
   service: light.turn_on
```

Listato 4.5: Automazione set light

User gone Con la seguente automazione, gestiamo il momento in cui l'utente lascia l'ufficio, magari la sera. Verifichiamo la non presenza di movimento, per un periodo superiore a 15 minuti. Nel caso in cui questa condizione fosse verificata, spegneremo la luce e chiuderemo le tende.

I componenti sono perciò strutturati come segue:

- Trigger: l'automazione viene eseguita quando non rileviamo movimenti sul sensore Shelly per un periodo maggiore di 15 minuti.

- Condition: non presente, in quanto la condizione è intrinseca all'avvio dell'automazione.

- Action: chiusura delle tende mediante il servizio *mqtt.publish* e spegnimento della luce mediante il servizio *light.turnOff*.

```
alias: user_gone
description: []
trigger:
  - platform: state
    entity_id: IsShellyMotionDetected
    from: true
   to: false
   for: 00:15:00
condition: []
action:
  - service: light.turn_off
    entity_id: light.office
  - service: mqtt.publish
    data_template:
        topic: shellies/shelly1-60AB45KWI/relay/0/command
        payload: on
  - delay: 00:01:00
  - service: mqtt.publish
    data_template:
        topic: shellies/shelly1-60AB45KWI/relay/0/command
        payload: off
```

Listato 4.6: Automazione user gone

Come notiamo, questa è l'automazione più corposa dell'intero sistema. Innanzitutto parte della logica di funzionamento è contenuta direttamente all'interno del componente *trigger*, specificando l'avvio delle successive operazioni solo dopo che, per almeno 15 minuti, non sono stati rilevati movimenti. Nel caso in cui questa condizione sia verificata, si procede con lo spegnimento della luce e la chiusura delle tende (ipotizzando, come al caso precedente, un tempo di chiusura pari a circa un minuto).

Button pressed Si decide di scomporre la pressione del pulsante in due automazioni separate: una per la pressione doppia ed una per la pressione singola.

Analizziamo la prima, denominata *button double shortpush*. La seguente automazione verrà richiamata con la doppia pressione del pulsante da parte dell'utente, indicando che la luminosità ambientale è troppo ridotta.

L'automazione sarà quindi strutturata come segue:

- Trigger: mediante la piattaforma MQTT verrà rilevata la doppia pressione con i messaggi in entrata su uno specifico topic.
- Condition: non presente.
- Action: si opterà per l'accensione della lampada (se non già accesa) e l'apertura delle tende in qualunque condizione meteorologica esterna.

La seconda automazione, denominata *button shortpush* indicante la pressione singola del pulsante, fornirà indicazioni sulla luminosità troppo elevata dell'ambiente. Perciò si procederà all'eventuale spegnimento della lampada (se accesa) altrimenti alla parziale chiusura delle tende.

Viene di seguito indicato il completo svolgimento della seconda automazione, a titolo esemplificativo.

```
alias: button_shortpush
description: []
trigger:
  - platform: mqtt
    topic: shellies/shellybutton1/input_event/0
    payload: S
    value_template: {{ value_json.event }}
condition: []
action:
  - choose:
      - conditions:
          - condition: state
            entity_id: light.office
            state: on
        sequence:
          - service: light.turn_off
            data: {}
            entity_id: light.office
      - conditions:
        sequence:
```

```
          - service: mqtt.publish
            data_template:
                 topic: shellies/shelly1-60AB45KWI/relay/0/command
                 payload: on
          - delay: 00:00:30
          - service: mqtt.publish
            data_template:
                topic: shellies/shelly1-60AB45KWI/relay/0/command
                payload: off
    default: []
mode: single
```

Listato 4.7: Automazione button shortpush

Per la rilevazione dell'evento di input, si è potato per l'utilizzo di un template, ottenendo il valore specifico dell'evento rilevato, in quanto i messaggi ottenuti dal sensore, sono come il seguente: *"event":"S","eventCnt":2*. Mediante il template inserito, otterremo in questo caso il valore S. Secondo la documentazione fornita da Shelly, i valori ottenibili sono i seguenti:

- *S* indicante una pressione breve e singola.
- *SS* indicante una doppia pressione breve.
- *SSS* indicante una tripla pressione breve
- *L* indicante una singola pressione prolungata

Secondo le informazioni ad ora ottenute, sarebbe perciò possibile estendere il sistema, consentendo all'utente di fornire numerose informazioni utili al miglioramento del sistema in fase di test, quali ad esempio: temperatura interna elevata/bassa o rapido richiamo manuale dell'automazione *user gone* per chiudere le tende e spegnere la luce rapidamente all'uscita dall'ufficio.

4.4 Analisi del sistema e considerazioni finali

Successivamente ad alcuni test, pratici e teorici, del sistema progettato, sono emerse alcune peculiarità ed alcuni svantaggi nell'utilizzo di un tale sistema per il completo controllo della luminosità di un ambiente di lavoro.

Innanzitutto si è optato per lo studio di un ambiente di lavoro e non casalingo, in quanto all'interno di una abitazione sono presenti molte più variabili, tra cui orario del giorno, periodo dell'anno ecc. che avrebbero comportato un dispendio progettuale molto più elevato ed inconsistente.

Le prime informazioni sul corretto funzionamento del sistema, sono state ottenute dall'automazione *user gone* la quale comanda lo spegnimento e la chiusura di tende e illuminazione quando non viene più rilevato movimento. Sarebbe possibile migliorare il sistema sotto questo punto di vista inserendo innanzitutto un eventuale sensore di presenza o equivalente, per monitorare in modo più accurato la presenza dell'utente nell'ufficio, ed eventualmente optare per la configurazione di specifici orari di lavoro, nei quali non chiudere/spegnere le luci, o solamente spegnere le luci. Ad esempio:

- Optiamo per l'inserimento nel progetto del seguente orario di lavoro: 8-12 e 15-18.
- Secondo l'orario inserito è prevista una pausa pranzo di circa 3 ore.
- Sarebbe perciò ipotizzabile un funzionamento simile al seguente: all'interno dell'orario lavorativo (non comprendendo la pausa) non sarà abilitata l'automazione *user gone*, mentre durante l'orario di pausa 12-15, verrà limitata la suddetta automazione, ad esempio spegnendo l'illuminazione e non chiudendo le tende, intervenendo di conseguenza quando l'utente si ripresenterà.
- Mentre fuori dall'orario lavorativo, prima delle 8 e dopo le 18, l'automazione consentirà la chiusura e lo spegnimento di tutti gli apparati.

Inoltre, sono stati presi in considerazione, eventuali cambiamenti luminosi esterni, o improvvise modifiche delle condizioni meteo, le quali non sono facilmente rilevabili dal semplice impianto domotico installato. Sarebbe perciò necessario l'installazione di un sensore di luminosità esterno aggiuntivo, da integrare nell'automazione *detect light*, tale da consentire una maggiore precisione nella determinazione della quantità luminosa nell'ufficio.

Sarebbe eventualmente possibile, mediante l'integrazione per smartphone di home assistant, utilizzare i dati ricavati dal sensore di luminosità del cellulare per migliorare ancora questo aspetto.

Per quanto concerne valutazione sui costi e sull'efficienza del progetto, è ipotizzabile una leggera diminuzione del consumo energetico della lampada inserita nel sistema, nelle stagioni maggiormente calde e luminose, per le quali la luminosità che perviene all'interno dell'ufficio dalle finestrature, sia sufficiente per l'illuminazione ambientale. I costi per la realizzazione di progetto come quello descritto fino ad ora, potrebbero però lievitare considerando tutti i fattori possibili, ad esempio umore dell'utente, temperatura esterna, temperatura interna, concedendo però molti vantaggi in termini di comodità ed automazione. Ad esempio durante la stagione invernale, le temperature si abbassano

drasticamente, perciò la mattina al lavoro sarebbe possibile aprire le tende non solo per la luminosità ma anche per scaldare l'ambiente grazie all'irraggiamento solare, allo stesso tempo risparmiando sul costo del riscaldamento dell'ambiente lavorativo.

Inoltre è possibile, mediante i sensori considerati nel progetto, provvedere ad aspetti secondari riguardanti la sicurezza. Il sensore di movimento considerato (Shelly Motion) è dotato di camera infrarossi, permettendo la rilevazione di movimento durante l'orario notturno, consentendo non solo aspetti minimi di sorveglianza, ma se collegato ad un sistema d'allarme domotico e connesso, renderlo parte di tale sistema, migliorando ancora di più il suo utilizzo.

Infine, viene considerata la soggettività dell'utente sulla luminosità ambientale. Infatti secondo numerosi studi, la luminosità dell'ambiente di lavoro, tanto quanto quella di un ambiente casalingo, è un fattore estremamente soggettivo, variando la percezione dell'utente in base ad esempio al suo umore, a cosa indossa, alla temperatura ed umidità esterna ed interna, alla stagione, al carico di lavoro ecc. ecc. Perciò in casi estremi sarà richiesto all'utente, un breve periodo di adattamento al sistema, nonostante l'utente, abbia la possibilità di esprimere il suo parere sulla luminosità introdotta. Attualmente, per ragioni prototipali, si è utilizzato un semplice bottone, che fornisce comandi alle automazioni di Home Assistant, con eventuali conseguenze. Utilizzando un sistema testato e correttamente funzionante, sarebbe invece possibile l'utilizzo dei più comuni assistenti vocali (quali ad esempio Google o Alexa) per fornire comandi ad Home Assistant. Tali sistemi, mediante le proprie API, traducono i comandi vocali ricevuti in un linguaggio macchina interpretabile, il quale produce il voluto output, che nel caso d'uso attuale, consisterebbe nell'attivazione di una specifica automazione su Home Assistant. In particolare, con l'utilizzo di Amazon Alexa, sarebbe possibile creare innanzitutto una routine sull'assistente vocale, la quale possa ricercare tra tutti i dispositivi e servizi all'interno della Smart Home, la voluta automazione o script (che appare come servizio), richiamandola. Successivamente l'automazione richiamata potrà eseguire le volute azioni.

Conclusioni

Lo scopo di questo elaborato è stato quello di avvicinare al mondo dell'Home Automation Open-Source, utenti insicuri nell'approccio di queste tecnologie, dati elevati costi e poca espandibilità, mediante l'analisi dello sviluppo di un sistema di Home Automation, su un contesto reale. La spinta che l'IoT ha fornito all'Home Automation, soprattutto in ambiti residenziali, riguarda soprattutto ambiti quali il confort, la sicurezza e la gestione degli automatismi. In particolare, ha consentito l'integrazione di differenti servizi e dispositivi all'interno di un unico ecosistema, tramite il quale scambiare informazioni.

In particolare, ha consentito l'avvento di particolari applicativi, quale Home Assistant, con le potenzialità di integrare differenti protocolli e dispositivi e con l'obbiettivo di permettere ad un sempre maggiore numero di persone l'utilizzo di tali tecnologie, con un costo ridotto. In seguito all'analisi del funzionamento del caso d'uso dell'elaborato, si è deciso di sviluppare un semplice applicativo di Home Automation con l'utilizzo di Home Assistant. In particolare l'obbiettivo era quello di studiare un sistema composto da componenti poco dispendiosi e di semplice utilizzo, per la regolazione dell'illuminazione all'interno di un ambiente lavorativo.

Sotto questo punto di vista, l'utilizzo di un software come Home Assistant ha permesso l'integrazione e lo scambio di informazioni con differenti protocolli di comunicazione, come MQTT o Hub proprietari. Le potenzialità sono perciò estremamente varie, consentendo l'eventuale espansione del sistema in futuro. A fini didattici, non si sono infatti tenuti in considerazioni eventuali aspetti di secondaria importanza, quali l'eventuale adozione di sensori aggiuntivi per un utilizzo più accurato del sistema o eventuali cambiamenti di temperatura per regolare l'illuminazione basandosi sulla temperatura interna o esterna.

La peculiarità del progetto sviluppato, infine, è proprio la possibilità di futuri sviluppi ed espansioni, utilizzando apparati precedentemente integrati per comporre e sviluppare differenti automazioni. Diviene possibile infatti, mediante l'integrazione di differenti servizi e dispositivi, l'utilizzo dei dispositivi integrati, per il soddisfacimento di automazioni eventualmente a fini di sicurezza (sensore di movimento) o a fini di risparmio energetico (regolazione della luminosità basandosi sulla temperatura interna ed esterna).

Sarebbe stato interessante mostrare i vantaggi concreti che si possono ottenere adottando il progetto sviluppato all'interno di un ambiente reale, quali soddisfacimento degli utenti, eventualmente adottando un'analisi sui dati storici memorizzati da Home Assistant, per consentire il miglioramento dell'esperienza d'utilizzo e dell'eventuale risparmio energetico sull'ambiente in oggetto.

Bibliografia

[1] Bruh automation: Smart home fundamentals. https://www.bruhautomation.io/fundamentals/. (Accessed: 2021-09-15).

[2] Das u-boot – the universal boot loader. https://www.denx.de/wiki/U-Boot. (Accessed: 2021-09-15).

[3] Docker documentation. https://docs.docker.com/get-started/overview/. (Accessed: 2021-09-15).

[4] Everything i learned about googles brillo and weave at ubiquity dev summit. https://www.getstructure.io/blog/everything-i-learned-about-googles-brillo-and-weave-at\-ubiquity-dev-summit. (Accessed: 2021-09-15).

[5] Home assistant. https://en.wikipedia.org/wiki/Home_Assistant. (Accessed: 2021-09-15).

[6] Home assistant community. https://community.home-assistant.io/. (Accessed: 2021-08-31).

[7] Home assistant developers' documentation. https://developers.home-assistant.io/docs/. (Accessed: 2021-08-31).

[8] Kernel documentation. https://www.kernel.org/doc/Documentation. (Accessed: 2021-08-31).

[9] Netapp: What are containers? https://www.netapp.com/devops-solutions/what-are-containers/. (Accessed: 2021-08-31).

[10] Smart home university: Smart things vs home assistant: What is the best smart home hub. https://smarthome.university/smartthings-vs-home-assistant/. (Accessed: 2021-09-15).

[11] Wikipedia: Samsung smart things. https://en.wikipedia.org/wiki/SmartThings. (Accessed: 2021-09-15).

[12] Gian Piero Abbate. *Domotica: Aprire la casa al futuro.* Abitare editore n. 348, October 1996.

[13] Muhammad Raisul Alam, Mamun Bin Ibne Reaz, and Mohd Alauddin Mohd Ali. A review of smart homes—past, present, and future. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1190–1203, 2012.

[14] Apple. Apple homekit. `www.apple.com/ios/homekit`. (Accessed: 2021-08-31).

[15] Vishwajeet Hari Bhide and Sanjeev Wagh. i-learning iot: An intelligent self learning system for home automation using iot. In *2015 International Conference on Communications and Signal Processing (ICCSP)*, pages 1763–1767, 2015.

[16] Maurantonio Caprolu, Roberto Di Pietro, Flavio Lombardi, and Simone Raponi. Edge computing perspectives: Architectures, technologies, and open security issues. In *2019 IEEE International Conference on Edge Computing (EDGE)*, pages 116–123, 2019.

[17] Tanmay Chakraborty and Soumya Kanti Datta. Home automation using edge computing and internet of things. In *2017 IEEE International Symposium on Consumer Electronics (ISCE)*, pages 47–49, 2017.

[18] Tushar Chaurasia and Prashant Kumar Jain. Enhanced smart home automation system based on internet of things. In *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 709–713, 2019.

[19] Mohamed E. El-hawary. The smart grid – state of thr art and future trends. `https://www.tandfonline.com/doi/full/10.1080/15325008.2013.868558`. (Accessed: 2021-08-31).

[20] E. Franke. Embracing the joint technical architecture (jta) for communications terminals of the 21st century. In *Proceedings of the IEEE 1997 National Aerospace and Electronics Conference. NAECON 1997*, volume 1, pages 106–113 vol.1, 1997.

[21] Aaron Godfrey. Building an home assistant custom component: complete guide. https://aarongodfrey.dev/home(Accessed: 2021-07-30).

[22] Google. Google weave. `https://weave.googlesource.com/`. (Accessed: 2021-08-31).

[23] InfoQ. Google brillo short description. `https://www.youtube.com/watch?v=ig9GKAFzDxQ`. (Accessed: 2021-08-31).

[24] John Jaihar, Neehal Lingayat, Patel Sapan Vijaybhai, Gautam Venkatesh, and K. P. Upla. Smart home automation using machine learning algorithms. In *2020 International Conference for Emerging Technology (INCET)*, pages 1–4, 2020.

[25] K Patel Keyuer and M Patel Sunil. Internet of things-iot: definition, characteristics, architecture, enabling technologies, application and future challenges. *opjstamnar*, 2018.

[26] Ivan Lazarevic, Milan Sekulic, Milan S. Savic, and Velibor Mihic. Modular home automation software with uniform cross component interaction based on services. In *2015 IEEE 5th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, pages 363–365, 2015.

[27] Marcin. What is home assistant and what it can do ? `https://futurehousestore.co.uk/what-is-home-assistant-and-what-it-can-do`. (Accessed: 2021-09-15).

[28] Janakiram MSV. Google brillo vs. apple homekit: The battleground shifts to iot. `http://www.forbes.com/sites/janakirammsv/2015/10/29/google-brillo-vs-apple-homekit-the-battleground-shifts-to-iot/`. (Accessed: 2021-09-15).

[29] Giuseppe Gustavo Quaranta. *La domotica per l'efficienza energetica delle abitazioni*. 2009.

[30] R. Katre Sharda and V. Rojatkar Dimesh. Home automation: Past, present and future. *Internation Research Hournal of Engineering and Technology (IRJET)*, 4(10):343–345, October 2017.

[31] Harsh Kumar Singh, Saurabh Verma, Shashank Pal, and Kavita Pandey. A step towards home automation using iot. In *2019 Twelfth International Conference on Contemporary Computing (IC3)*, pages 1–5, 2019.

[32] Maggie Tillman. Apple homekit and home app: What are they and how do they work ? `http://www.pocket-lint.com/news/129922-apple-homekit-explained-is-it-available-yet-and\how-does-it-work`. (Accessed: 2021-09-15).

[33] Mahmood Waqas, Kashan Syed, and Shah Ali. Smart home automation using iot and its low-cost implementation. *International Journal of Engineering and Manufacturing*, October 2020.