Università degli Studi di Messina

Dipartimento Scienze Matematiche e Informatiche,

Scienze Fisiche e Scienze della Terra

Corso di Laurea Triennale in Informatica

Progettazione e implementazione di interazioni basate su Cloud tra sistemi domotici ed assistenti digitali

Relatore:
Prof. Antonio Puliafito

Candidato:
Alessio Mobilia

Correlatore:
Ing. Giovanni Merlino

Anno Accademico 2018/19

Programmare oggi è una gara tra i tecnici del software che lottano per costruire programmi migliori ed a prova di idiota, e l'Universo che cerca di produrre migliori e più grandi idioti. Al momento, l'Universo sta vincendo.

Rick Cook

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

> "*Internet ben presto esploderà in modo spettacolare, come una supernova, e nel 1996 collasserà*" - Robert Metcalfe, 1995

Robert Metcalfe, fondatore della 3Com, pioniere della tecnologia e coinventore dello standard Ethernet, due anni dopo aver pronunciato questa frase si rimangiò pubblicamente le sue parole, letteralmente. Prese un foglio con su scritta la sua frase, lo frullò e lo mangiò[14].

Appena 30 anni fa era praticamente impossibile pensare che saremmo potuti essere connessi costantemente ad una rete di informazioni sconfinata come internet. Invece oggi è questa la realtà. La rete continuerà ad espandersi sempre di più, finché ogni singolo oggetto che ci circonda non sarà connesso alla rete internet.

Si vuole presentare questa tesi con la speranza che sempre nuove tecnologie possano svilupparsi, per poter migliorare la vita dell'uomo e dell'ambiente in cui vive

L'idea a suo fondamento è quella di rendere sempre più facile l'integrazione di dispositivi eterogenei in un sistema domotico affidabile e a basso costo, così da permettere a chiunque di automatizzare la propria abitazione, sopratutto a chi ne ha veramente la necessità a causa di difficoltà fisiche.

Si architetta e sviluppa, quindi, un sistema domotico altamente modulare, a basso costo, con supporto ai moderni assistenti vocali.

A garanzia dell'utilità del progetto vi è il brevetto richiesto e ottenuto dal Dottore

Mauro Staiti e dagli Ingegneri Giovanni Merlino e Davide Aloisio.
Il brevetto è pubblicamente consultabile alla banca dati dei brevetti europei con il codice identificativo IT201600120472 (A1) e il nome "Sistema Domotico Modulare".

Per sviluppare un progetto affidabile e sicuro è stato utilizzato un approccio di ingegneria del software orientato al riuso, scegliendo attentamente sistemi software open source con licenze d'uso permissive dal punto di vista commerciale.

La tesi è divisa in 5 capitoli. Il primo capitolo tratta del background della tesi, quindi la Domotica, l'IoT, con i loro vantaggi e svantaggi, gli assistenti virtuali e la filosofia dell'open source.
Il secondo capitolo tratta tutte le tecnologie sfruttate nel progetto, software e hardware, ne spiega il funzionamento. La parte è necessaria per comprendere i successivi capitoli.
Il terzo descrive il cuore del progetto stesso, l'architettura del sistema e la sua implementazione.
Nel quarto capitolo sono mostrati dei test del sistema sviluppato.
Infine, nell'ultimo capitolo, si conclude la tesi con i risultati ottenuti e le prospettive future del progetto.

Capitolo 1

Background

1.1 Domotica e IoT

La **domotica** è la tecnologia che studia l'automazione della casa. Il termine domotica deriva dal francese domotique, che a sua volta deriva dall'unione della parola greca "domus"(casa) e "automatique"(automatica). Quindi domotica significa letteralmente "casa automatica".

Con questo termine si vuole, quindi, indicare tutte quelle tecnologie atte ad automatizzare le operazioni all'interno di un'abitazione.

Si potrebbe pensare che la domotica sia una invenzione recente, in realtà già da tempo l'uomo ha portato in casa tecnologie atte ad automatizzare diversi comportamenti. Un esempio è lo spegnimento automatico nel televisore, oppure l'attivazione automatica della caldaia, che mantiene la temperatura costante controllando i dati raccolti da un sensore. Il passo in più che porta alla vera e propria domotica è la comunicazione tra i diversi dispositivi [23].

La domotica a differenza del'IoT non sfrutta la rete internet, ma tutti i dati sono scambiati all'interno di una rete locale, perciò viene rispettata la privacy e nessun dato può essere utilizzato o venduto a scopi di marketing o statistici. L'**internet delle cose**

(IoT, Internet of Things) si basa su tecnologie che trasformano oggetti "inanimati", dotati

di sensori che raccolgono e scambiano dati, in dispositivi interconnessi.
Attorno all'Iot ruotano protocolli e piattaforme di interscambio di dati che consentono di combinare soluzioni hardware e software per ottenere servizi che dispongono di componenti fisiche strettamente connesse a componenti software.

La connettività tra oggetti, mediato attraverso internet, rappresenta anche una leva per la creazione di valore statistico che permette l'utilizzo di grandi volumi di traffico (big data) a supporto di iniziative di marketing sempre più mirate e non solo[21].
Per questo è utile mantenere una maggiore asimmetria tra le informazioni contenute nella rete locale e quelle fornite ai diversi sistemi cloud.

1.1.1 Vantaggi

L'automazione della casa può portare a numerosi e importanti vantaggi, come il **risparmio sui consumi**. Infatti controllare in modo intelligente e automatico le diverse fonti energetiche può ridurre i costi di gestione(dal 20% al 30%). I consumi possono essere tenuti sotto controllo per ridurre gli sprechi e ottimizzando le performance dell'impianto.

Si ha anche un notevole aumento per quanto riguarda la **sicurezza** personale e della casa. Grazie a diversi sensori è possibile riconoscere fughe di cas, perdite d'acqua, guasti elettrici ed eventualmente attivare autonomamente delle contromisure. Come anche l'impianto antifurto può essere integrato al sistema domotico e quindi avvertire il proprietario e le forze dell'ordine in caso di intrusione e permettere di controllare tramite delle telecamere in qualsiasi momento lo stato della casa anche se ci si trova in un altro luogo.

Una casa intelligente permette di avere il **controllo totale** della casa stessa. Tramite uno schermo touch, uno smartphone o comandi vocali è possibile accendere e spegnere tutti i dispositivi, programmare il loro funzionamento ad orari stabiliti e vedere lo stato di ogni dispositivo.

La **semplicità di utilizzo** è sicuramente un aspetto da non sottovalutare. Prima era difficile installare dispositivi del genere nei propri appartamenti, invece ora si tende a

creare dispositivi sempre più inutitivi e quindi user-friendly[13].

Un impianto domotico ormai è anche un **valore aggiunto** che rende più appetibile l'immobile nel mercato. A detta dell'Associazione Nazionale Agenti e Mediatori d' Affari *"Il vantaggio è compreso tra l'8 e il 12%, con punte che sfiorano il 30%"*.

L'IoT non si ferma però solamente all'uso casalingo, è possibile automatizzare un'intera città, per migliorare le condizioni di vita dei cittadini, ridurre i costi e gli sprechi e contribuire all'eliminazione delle barriere architettoniche raccogliendo dati da diversi dispositivi eterogenei sparsi per la città [9]
Anche la città di Messina, con l'aiuto dell'Università, si sta muovendo in questa direzione grazie al progetto **#SmartMe**[4]

L'IoT permette a persone con deficit motori di eseguire con maggiore facilità i compiti di tutti i giorni, grazie anche all'utilizzo degli assistenti vocali. Ad esempio con dei dispositivi collegati in rete è possibile monitorare lo stato di salute del paziente tramite dei sensori per il battito cardiaco e facilitargli i movimenti utilizzando delle sedie a rotelle comandate anche vocalmente [11].

1.1.2 Sicurezza e privacy

Sebbene l'automazione della casa permetta una maggiore sicurezza grazie all'istallazione di telecamere smart e sistemi di allarme, questa potrebbe essere un'arma a doppio taglio. Una cattiva gestione di dispositivi accessibili dalla rete internet potrebbe fornire una via di accesso alla nostra abitazione a persone non autorizzate.

Maggiore è il controllo di questi dispositivi sulla casa e maggiore è l'importanza che assume la sicurezza informatica dei dispositivi. Si immagini che un malintenzionato potrebbe utilizzare i nostri stessi dispositivi di sicurezza per controllarci e spiarci.

È possibile evitare intrusioni attraverso una buona architettura del sistema e un utilizzo di protocolli di sicurezza per l'autenticazione dei dispositivi, ma, come insegna la breve

storia dell'informatica, non esiste un dispositivo veramente sicuro.

Nonostante si possano prendere diverse precauzioni è spesso necessario fare affidamento ad infrastrutture di terzi, come i server Amazon o Google per utilizzare i relativi assistenti vocali. Questo comporta una inevitabile fuga di informazioni private e l'unica opzione possibile per utilizzare i servizi di terze parti è fidarsi di loro. Informazioni che potrebbero essere vendute a scopi di marketing o a scopi statistici.

I recenti accadimenti ci dimostrano che i sistemi IoT di ultima generazione permettano di creare una rete di controllo molto ampia per stabilire una maggiore sicurezza dei cittadini, ma che allo stesso modo ne comprometta la privacy. Un sistema del genere, come quello in sviluppo in Cina, può essere utilizzato come strumento di repressione, per aumentare quindi il potere del governo, e limitare, se non eliminare, il libero arbitrio.
È stato persino ideato un sistema di punteggio per catalogare le persone e, in un certo senso, dividerle tra "buoni" e "cattivi".

La tecnologia fornisce ogni giorno nuovi strumenti sempre più potenti, è l'uomo che deve decidere come utilizzarli, non vi è assolutamente niente di intrinsecamente malvagio nel'IoT.

1.2 Assistenti Virtuali

Un **assistente virtuale intelligente** è un software che esegue delle operazioni in risposta a dei comandi vocali. Alcuni assistenti virtuali sono capaci di interpretare il linguaggio umano e rispondere grazie ad una voce sintetizzata. Gli utenti possono fare delle domande, controllare i propri dispositivi smart, chiedere la riproduzione di musica, aggiungere sveglie, promemoria e molto altro [24].

Gli assistenti permettono quindi di sviluppare performance vocali naturali ed offrono agli utenti un modo intuitivo per interagire con la tecnologia.

Attualmente gli assistenti vocali si trovano in una fase di rapida crescita, nuovi prodotti

vengono immessi nel mercato e gli assistenti vocali già affermati vengono sempre più incorporati nei nuovi dispositivi, come telecomandi, smartphone, smartwatch, speaker e smart TV.

Gli assistenti vocali più diffusi, nonchè quelli attualmente più avanzati, sono:

- **Amazon Alexa**:

  L'assistente vocale di Amazon basato su cloud. È l'inteliggenza virtuale che alimenta i dispositivi Amazon Echo. Amazon fornisce una documentazione [2] molto dettagliata per approcciarsi ad Alexa e creare le Skill, applet che ne estendono le funzionalità, questo ha permesso un sua rapida diffusione. Ogni Skill prima di essere pubblicata deve passare un rigoroso controllo da parte del team di sviluppo Amazon che si accerta dell'assenza di falle di sicurezza.

- **Google Assistant**:

  L'assistente vocale creato dalla famosa azienda statunitense Google è ormai presente in ogni smartphone moderno e permette di controllarlo in modo quasi totale, permettendo quindi di chiamare e mandare messaggi senza toccare lo smartphone. Anche Google ha fornisce la catena di altoparlanti intelligenti, i Google Home, che sfruttano l'assistente.

  Google fornisce una discreta documentazione [10] per creare le Actions, applets per estendere le funzionalità dell'assistente.

- **Microsoft Cortana**:

  L'assistente digitale sviluppato da Microsoft prende il nome di Cortana. Questa è preinstallata sul sistema operativo Windows 10 e su quello della console Xbox One. La sua funzione principale è quella di fornire supporto per l'utilizzo di queste piattaforme e non prevede una facile aggiunta di funzionalità, questo, insieme alla poca compatibilità con sistemi Linux, ne riduce le possibilità di utilizzo e attualmente non tiene il passo agli altri assistenti digitali per quanto riguarda la domotica.

- **Apple Siri**:

  Siri è invece l'assistente digitale sviluppato da Apple Inc., è presente in tutti i mo-

derni sistemi operativi sviluppati dalla stessa azienda. Siri permette di controllare i dispositivi che supportano HomeKit o Airplay 2.

1.2.1 Sicurezza e Privacy

Utilizzare degli assistenti vocali significa circondarsi di microfoni in costante ascolto. Teoricamente non registrano e non elaborano alcuna informazione detta se non è preceduta da una determinata wake word, come ad esempio "Alexa" o "ok, Google".

Essi utilizzano software proprietario che non fornisce certezze sui loro algoritmi, proprio per questo sono stati creati dei progetti opensource come Project Alias[3].

Project Alias è un sistema basato su una board Raspberry che ha lo scopo di evitare che il Google Home rimanga in costante ascolto producendo del rumore bianco sopra i microfoni del dispositivo. Sarà proprio Project Alias a rimanere in ascolto per una wake word e attivare eventualmente il Google Home. Si può quindi essere sicuri di non essere ascoltati in continuazione, in quanto tutto il sistema è Open Source e i codici possono essere visti e liberamente modificati da chiunque.

Rimangono comunque dei possibili problemi di sicurezza. Vi è sempre la possibilità di trovare degli exploit che permettano ad utenti malevoli di recuperare informazioni e addirittura ascoltare intere conversazioni. Più sono i dispositivi connessi, maggiori sono le possibili vulnerabilità[5].

Rimane anche il problema della presenza di informazioni su server cloud, dove vi è alcun controllo su chi possa accedervi.

1.3 Open Source

Il termine **open source** indica un tipo di software e il suo modello di sviluppo. Un software è reso tale per mezzo di una licenza attraverso cui i detentori dei diritti ne favoriscono la modifica e lo studio.

"*Quando parliamo di software libero ci riferiamo alla libertà, non al valore monetario*" sostiene **Stallman** [25], il principale esponente del movimento del software libero. È vero che il codice open source è gratuito, ma l'idea di base non consiste nel rendere i software gratuiti, ma nel dare la libertà di espressione a chiunque anche attraverso il codice. "*Non pensiamo a 'free come in 'free beer', ma piuttosto come in 'free speech*" [25].

Chiunque può modificare il codice open, apportare miglioramenti, e sopratutto studiarlo e imparare.

L'open source impone una maggiore sicurezza e affidabilità al codice. Più persone possono vedere e controllare il software, quindi nessuno può mentire su ciò che effettivamente esegue. Inoltre maggiore controllo sul codice significa meno bug, più feature e sviluppo più veloce.

Attualmente, l'open source tende ad assumere sempre maggiore rilievo filosofico. Crea una concezione della vita aperta, e pronta alla condivisione della conoscenza.

Condividendo questa visione filosofica e includendo tutte le caratteristiche positive del software open source, come la maggiore affidabilità e la sicurezza, anche il progetto discusso in questa tesi utilizza solamente software open source.

Capitolo 2

Sistemi e tecnologie

2.1 Arancino

L'Arancino è una board creata da SmartMe grazie all'esperienza nel mondo Arduino combinata con l'esperienza universitaria.
Questo prodotto include il Compute Module 3+, ovvero una Raspberry pi 3 b+ con un form factor adatto ad applicazioni industriali.

Figura 2.1: Arancino in preparazione

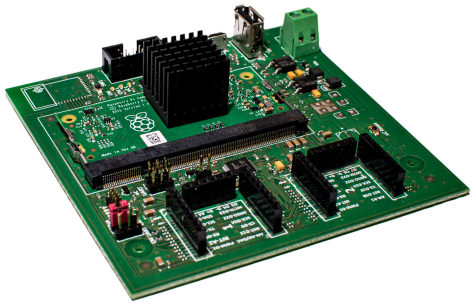

Figura 2.2: Arancino fritto

2.1.1 Raspberry e Raspbian

La CM3+(Compute Module 3+) integra:

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.2GHz;
- 1GB LPDDR2 SDRAM;

Questo è all'interno in una piccola scheda che si inserice in un connettore SODIMM standard. La memoria flash è collegata direttamente al processore, ma le interfacce rimanenti sono accessibili tramite i pin del connettore.

Il sistema operativo raccomandato per la scheda Raspberry pi è Raspbian. È Open Source ed è basato su Debian. Sulla board Arancino è installato un sistema operativo basata su Raspbian.

2.2 OpenHAB

Open Home Automation Bus è una piattaforma open source per l'automazione della casa. Permette di integrare molti device e sistemi, fornendo quindi un tool molto flessibile per realizzare l'automazione completa della casa.

OpenHAB comunica elettronicamente con dispositivi smart e non.
Per riuscire in questo divide in compartimenti funzioni e operazioni.

Permette la gestione de dispositivi attraverso interfacce pratiche e user-friendly. Dando comunque la possibilità di collegare qualsiasi tipo di dispositivo, grazie alla sua struttura modulare.

Di seguito una breve descrizioni delle operazioni principali:

Concetti	Significati
Bindings	sono i componenti che forniscono l'interfaccia per interagire elettronicamente con i dispositivi
Things	Sono la prima rappresentazione (software) dei dispositivi su OpenHAB
Channels	Sono le connessioni tra "Items" e "Things"
Items	Sono la rappresentazione delle informazioni che riguardano i dispositivi
Rules	Sono le regole che permettono l'automatizzazione di certi comportamenti
Sitemap	E' l'interfaccia web che rappresenta le informazioni e permette le interazioni con dispositivi

Tabella 2.1: Tabella dei concetti di OpenHAB [19]

2.2.1 Things

Le Things (cose) sono entità che possono essere aggiunte fisicamente al sistema e che possono avere più funzionalità. È importante capire che le Things non devono essere dispositivi fisici, ma potrebbero essere anche servizi web o qualsiasi altra fonte controllabile.

I Things possono avere delle proprietà da configurare opzionali o obbligatorie.

2.2.2 Items

OpenHAB separa il mondo fisico (Things) da quello software. Gli Item sono la rappresentazione software di una funzionalità di una Thing. Gli item conservano lo stato. Un item ha un tipo che può essere Colore, Data, Immagine, ecc..

2.2.3 Channels

I Channels (canali) rappresentano le differenti funzioni che una Thing fornisce. Se la Thing è l'entità fisica, il Channel è la funzione concreta che permette di controllare il comportamento della Thing.

I Channels sono collegati agli Items. Sono sostanzialmente il collegamento tra il livello fisico e quello virtuale.

La separazione tra diversi concetti permette una maggiore libertà. Un oggetto con più funzionalità potrebbe essere identificato tramite una Thing e le sue funzionalità potrebbero essere controllate utilizzando diversi Item. Si dovrebbero linkare gli Item ad un canale e questo collegarlo alla Thing.

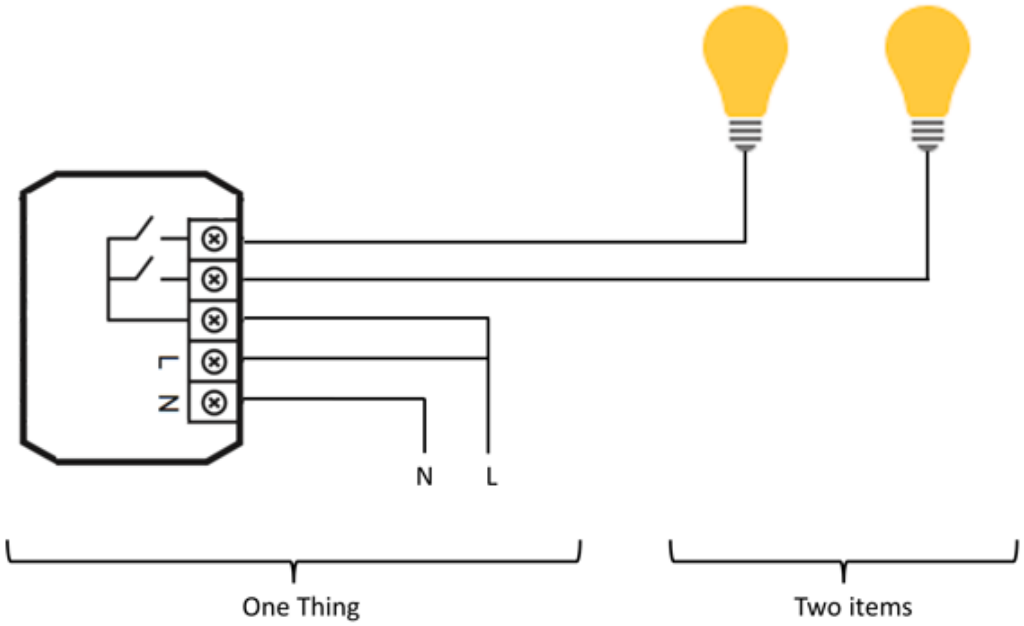

Figura 2.3: Thing con più funzionalità in OpenHAB[19]

2.2.4 Bindings

I Binding sono l'interfaccia di comunicazione effettiva tra le Thing e i dispositivi fisici. È possibile aggiungere Binding già creati dalla comunity di OpenHAB o crearne altri.

2.2.5 Rules

le Rules sono delle vere e proprie regole che l'utente può definire per automatizzare il comportamento dei dispositivi, definendo un'azione in risposta ad un evento (trigger).

2.2.6 Sitemap

Le Sitemap servono all'utente per interaggire con i diversi dispositivi. Possono essere composte e personalizzate attraverso dei file .sitemap.

2.2.7 Interfacce

OpenHAB è accessibile tramite console, rest api, file di configurazione e sopratutto tramite interfaccia web.
Non esiste solamente una interfaccia web, ne esistono differenti e ognuna con uno scopo differente:

- **Paper UI** è l'interfaccia grafica per l'amministrazione del sistema e include l'accesso agli items.
- **Classic UI** è un'interfaccia ibrida che risale alla prima versione di OpenHAB.
- **Basic UI** è un'interfaccia moderna e basilare che permette di sfruttare OpenHAB al meglio dai dispositivi mobile.
- **HABPanel** è un'interfaccia personalizzabile che permette di interagire con i diversi dispositivi. È pensata per essere usata su dei tablet a muro.
- **HABmin** è una console di amministrazione completa.

L'accesso alle interfacce può essere limitato utilizzando delle credenziali di accesso. La rest interface utilizza l'autenticazione di base http oppure un'autenticazione basata su certificati.

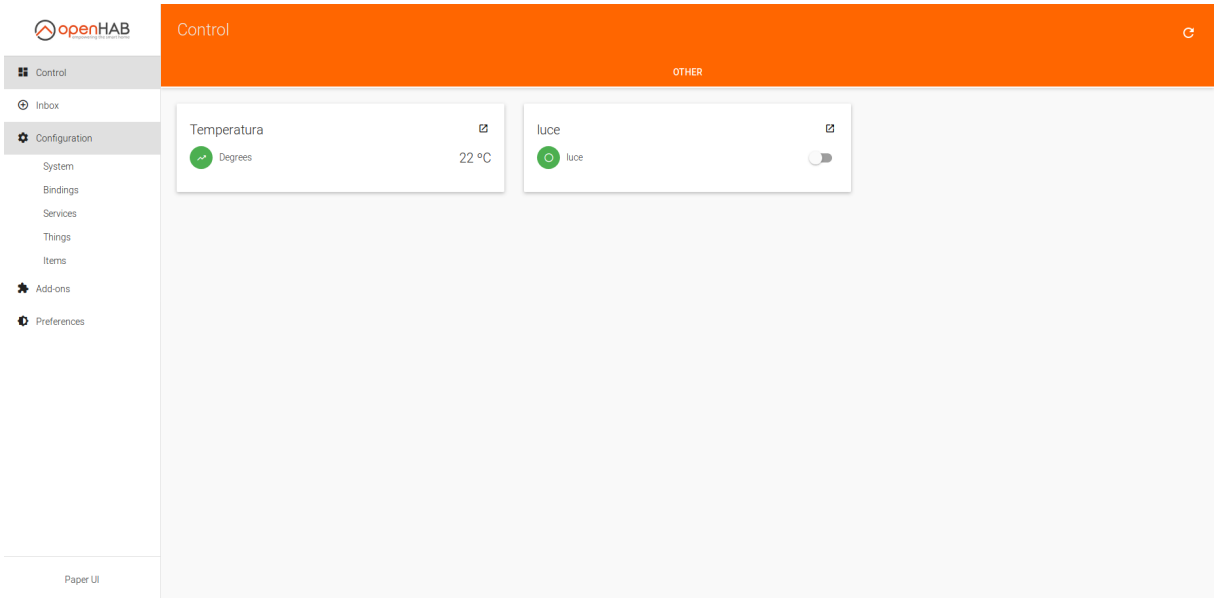

Figura 2.4: Interfaccia Paper UI con il controllo dei dispositivi

2.2.8 Eclipse SmartHome

OpenHAB è costrutito sopra al **framework Java Eclipse SmartHome**, infatti molti dei concetti visti su OpenHAB sono già presenti nel framework.

Eclipse SmartHome si concentra nel fornire le API per i seguenti argomenti:

- **Data Handling:** il framework fornisce delle api per gestire facilmente i dati dei diversi dispositivi, compresi quelli astratti. Fornisce meccaniscmi per scambiare informazioni con altri sistemi attraverso i binding.
- **Rule Engines:** Vengono forniti dei sistemi flessibili per permettere di creare e cambiare regole di funzionamento a runtime. Le regole sono divise in trigger, actions, logic modules e templates.
- **Declarative User Interfaces:** Il framework prevede anche la presenza di interfacce utente customizzabili e include dei set di icone, widget e grafici.
- **Persistence Management:** Fornisce anche infrastrutture che permettono il processamenteo automatico di dati basate su file di configurazione e permettono quindi

lo storage di informazioni. I servizi persistenti sono estenzioni che possono essere aggiunti modularmente e possono essere qualisasi cosa, da un file di log a un servizio cloud.

2.3 Stack4Things

Stack4Things è un framework per l'IoT sviluppato da **MDSLab** (Mobile and Distributed Systems Lab) all'Università di Messina. È un progetto open source il cui scopo è gestire dispositivi IoT senza preoccuparsi della loro locazione fisica, della rete in cui si trovano e della loro tecnologia.

È una soluzione orientata al cloud integrata con OpenStack che fornisce la virtualizzazione, la personalizzazione e l'amministrazione dei dispositivi[15].

Stack4Things permette quindi di interagire con diversi dispositivi, controllarne le proprietà software e hardware. Fornisce gli strumenti per accedere ai dispositivi in modo remoto e aggregarli in gruppi. Aggira problemi legati a NAT e firewall e fornisce sistemi di amministrativi e di multi-tenancy dei dispositivi.

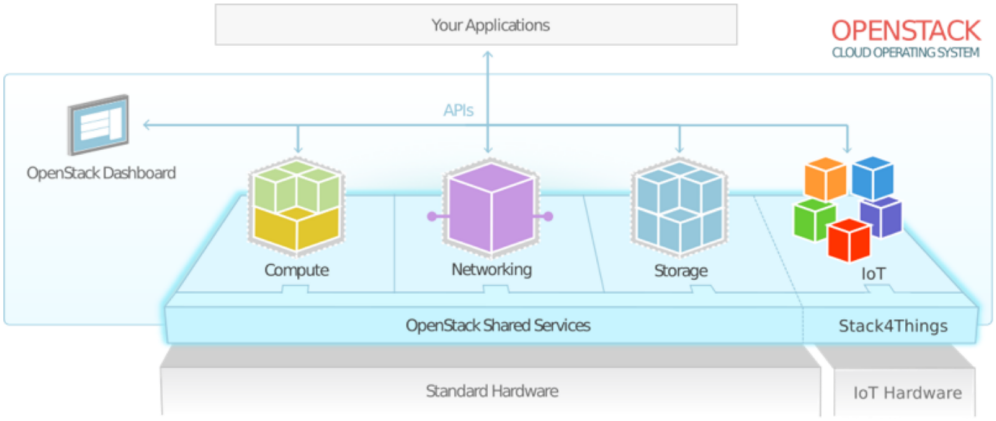

Figura 2.5: Stack4Things

Stack4Things coinvolge due principali componenti: **IoTronic Service** e **Lightning-rod Agent**.

2.3.1 IoTronic Service

IoTronic è il servizio di managment delle risorse IoT per il lato cloud di Stack4Things. Permette di gestire le risorte IoT come una parte di un OpenStack data center.

2.3.2 IoTronic Lightinig-Rod Agent

Lightinig-Rod è il componente nel lato del dispositivo all'interno dell'architettura di Stack4Things.

Il suo funzionamento è stato testato con:

- Raspberry Pi 2 and 3
- Arduino YUN / Linino ONE
- Ubuntu 16.04 / 14.04
- Docker containers
- LXD containers

[15]

2.4 NodeRed

NodeRed è un tool di programmazione per collegare visivamente diversi device hardware, API e servizi online. Fornisce un editor browser-based che rende facile collegare insieme diversi flussi di informazioni utilizzando i numerosi nodi a disposizione.

I nodi sono i punti di collegamento con i diversi servizi e offrono anche un modo per definire delle funzioni javascript per controllarne il funzionamento.

I nodi rendono NodeRed uno strumento modulare e flessibile, infatti possono essere sviluppati e aggiunti in modo indipendente.

Il runtime è costruito su Node.js e ne prende i vantaggi del suo modello non bloccante event-driven. Questo lo rende ideale per funzionare su dispositivi edge anche con hardware a basso costo[18].

È un software open source con licenza "Apache License 2.0", facilmente reperibile sulla piattaforma GitHub[17].

2.5 Microcontrollori

2.5.1 ESP32

Gli ESP32 sono una famiglia di system on a chip (SoC) a basso costo e a basso consumo energetico capaci di sfruttare la tecnologia WI-FI e bluetooth.

Il dispositivo possiede caratteristiche ottime per un edge device a basso consumo energetico:

- Processore principale: Tensilica Xtensa 32-bit LX6 microprocessor Dual Core
- Wi-Fi: 802.11 b/g/n/e/i (802.11n @ 2.4 GHz up to 150 Mbit/s)
- Bluetooth: v4.2 BR/EDR and Bluetooth Low Energy (BLE)
- ROM: 448 KiB
- SRAM: 520 KiB
- input/output: interfaccia con DMA
- Sicurezza:
  - IEEE 802.11 standard security
  - Secure boot

– Flash encryption

– 1024-bit OTP

– Accelerazione hardware crittografica: AES, SHA-2, RSA, ECC, RNG

[8]

Figura 2.6: Board con ESP32

2.5.2 ESP8266

L'ESP8266 è un chip a basso costo e a basso consumo energetico capaci di sfruttare la tecnologia WI-FI con pieno supporto al protocollo TCP/IP

Il chip possiede pochi componenti e lo ha reso molto facile da produrre in larga scala a basso costo. Il dispositivo possiede caratteristiche ottime per un edge device a basso consumo energetico, nonostante non sia particolarmente performante:

- Processore: microprocessore RISC L106 a 32 bit
- Wi-Fi: IEEE 802.11 b/g/n
- RAM: 64 KiB per le istruzioni, 96 KiB per i dati
- comunicazione: 16 pin GPIO, SPI, DMA, I2C, I2S, UART

[8]

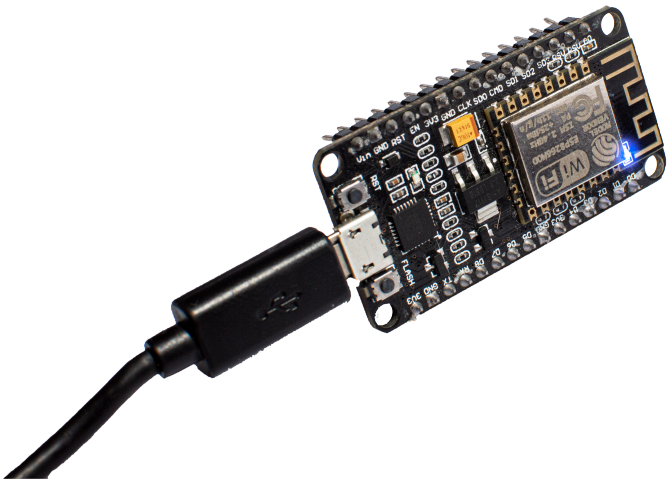

Figura 2.7: Board con ESP8266 con led acceso

2.6 MQTT

MQTT [16] (Message Queue Telemetry Transport) è un protocollo ISO e OASIS standard di messaggistica di tipo publish-subscribe posizionato sopra la stack TCP/IP.

E' progettato per essere utilizzato in situazioni in cui è richiesto un basso impatto e dove la banda è limitata.

Il protocollo MQTT è perfetto per l'utilizzo in IoT per via del basso inpatto nella banda internet e nelle risorse dei dispositivi. Inolte i protocolli di tipo publish/suscribe si sposano bene alla comunicazioni asincrona tipica di questi dispositivi.

Richiede un message broker responsabile della distribuzione dei messaggi ai client destinatari.

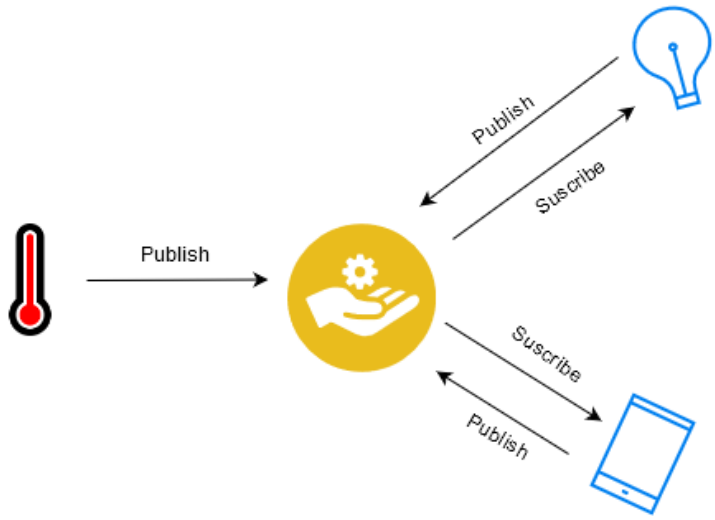

Figura 2.8: Funzionamento MQTT

2.6.1 Eclipse Mosquitto

Eclipse Mosquitto è il broker open source (con licenza "Eclipse Public License 1.0") con supporto a MQTT v3.1.1 e v5.0 che è stato utilizzato nel progetto di cui tratta questa tesi. È leggero e adatto all'utilizzo in device a basso consumo energetico[7].

2.6.2 Homie convention

Homie è una convenzione per la comunicazione effettuata tramite protocollo MQTT tra dispositivi IoT e entità di controllo.

MQTT non definisce la struttura e il contenuto dei messaggi, perciò nasce la necessità di una comunicazione standard affinché i dispositivi possano annunciare la loro presenza su un topic e scambiare messaggi. Homie definisce un modo per effettuare il discovery automatico, la configurazione e l'uso dei dispositivi.

esempio di scambio di messaggi utilizzando la Homie convention:

```
homie/device123/$homie ->  3.0
homie/device123/$name ->  My device
homie/device123/$state -> ready
```

```
homie/device123/$nodes -> mythermostat

homie/device123/mythermostat/$name -> My thermostat
homie/device123/mythermostat/$properties -> temperature

homie/device123/mythermostat/temperature -> 22
homie/device123/mythermostat/temperature/$name -> Temperature
homie/device123/mythermostat/temperature/$unit -> °C
homie/device123/mythermostat/temperature/$datatype -> integer
homie/device123/mythermostat/temperature/$settable -> true
```

[22]

2.6.3 Homie per ESP8266

homie-esp8266 è un'implementazione della convenzione homie per dispositivi esp32 e esp8266.

Homie per ESP8266 opera in 3 modalità:

- la modalità **configuration** genera un AP ed espone delle rest api attraverso le quali è possibile caricare un file di configurazione in JSON.
- la modalità **normal** è la modalità in cui il dispositivo si dovrebbe trovare per la maggior parte del tempo. Se è presente il file di configurazione allora il dipositivo entrerà automaticamente in questa modalità. Il dispositivo si connette alla rete wifi e al broker mqtt configurato.
- la modalità **standalone** permette di avviare il dispositivo senza entrare nella modalità di configurazione, anche se non è presente il file config.json, così da evitare di generare un AP che lascerebbe il dispositivo vulnerabile ad attacchi informatici.

[12]

2.7 Amazon Alexa

Amazon Alexa è un assistente personale intelligente sviluppato dall'azienda statunitense Amazon.

È un'intelligenza artificiale in grado di interpretare il linguaggio naturale e dialogare fonendo diverse informazioni. Può riprodurre musica, gesire liste, impostare promemoria e sveglie, riprodurre podcas e audiolibri, fornire in generale informazioni recuparabili online.

Alexa può controllare diversi dispositivi smart anche utilizzando se stessa come sistema di gestione della domotica. La selezione di dispositivi compatibili è però limitata e le informazioni dei dispositivi sono conservate nel cloud Amazon.

La maggior parte dei dispositivi che integrano Alexa permettono di attivarla tramite l'uso della parola di attivazione "Alexa". Per alcuni dispositivi, come l'app Alexa è necessario premere un pulsante [1].

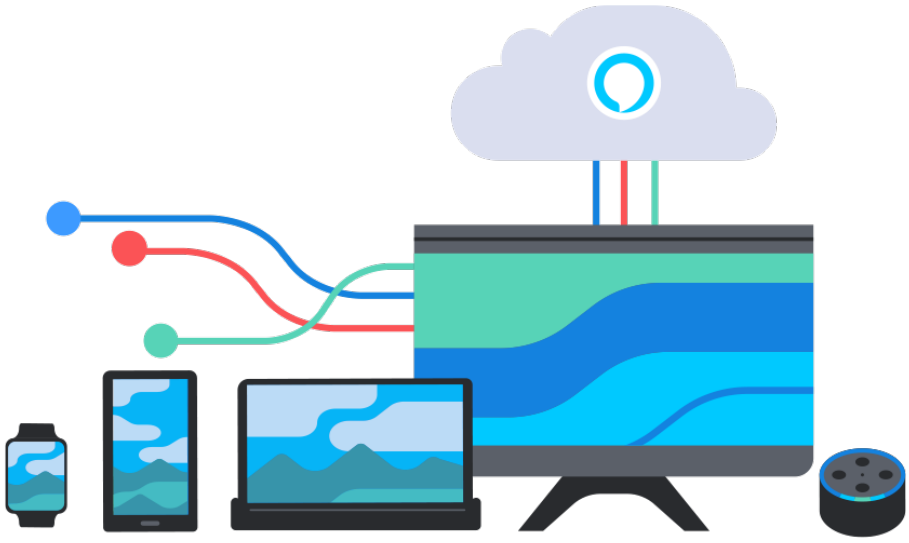

Figura 2.9: Dispositivi Alexa

2.7.1 Skill

Alcune funzioni di Alexa sono native, mentre altre richiedono l'utilizzo di app di terze parti, dette Skill.

L'Alexa Skills Kit (ASK) è una raccolta di API, strumenti e documentazioni[6] che rendono lo sviluppo delle skill più rapido e semplice.

Esistono diverse tipologie di Skill, quelle che lasciano più spazio di manovra sono le Custom Skills, mentre la tipologia di Skill utilizzate nel progetto di tesi sono quelle SmartHome.

Le parti principali necessarie per il funzionamento di una skill sono:

Concetti	Significati
Intents	rappresentano le azioni che un utento può eseguire attraverso la skill
Utterance	Sono delle frasi che l'utente può dire per invocare gli intents.
Invocation name	È il nome che identifica la skill.
Set di file	set di file immagini, audio e video che si vogliono includere nella skill
Cloud-based service	Un servizio cloud che accetta gli intent come richieste strutturate ed agisce di conseguenza.
Configuration	lega tutti le parti sopra elencate in modo che Alexa può inoltrare le richiesta alla skill.

Tabella 2.2: Tabella dei concetti fondamentali delle Skill di Alexa

esempio di Utterance:

Voglio viaggiare da {fromCity} fino a {toCity} il {travelDate}

Le parole tra parentesi sono gli Slots e rappresentano delle variabili.
Si veda l'appendice B.1 per un esempio di definizione degli Intent e delle Utterance.

Dopo che l'utente ha espresso il comando la frase viene mappata da Alexa con il relativo intent e inviato in formato JSON alla Skill.

Le Skill Smart Home non necessitano di essere invocate tramite il loro Invocation name e prevedono degli Intent e delle Utterance predefinite. Perciò Alexa impone un ambiente standardizzato per la gestione di ambienti domotici, questo rende più facile all'utente utilizzare diverse skill potendosi approcciare allo stesso modo.

Le API per Smart Home Skill forniscono interfacce che permettono allo sviluppatore di descrivere i dispositivi, le loro proprietà e le direttive che supportano. Queste informazioni devono essere inviate ad Alexa in risposta ad una richiesta di discovery. Queste informazioni verranno poi utilizzate da alexa per utilizzare le funzionalità dei dispositivi.

Esistono numerosi tipi di dispositivi, i principali sono:

- Cameras
- Cooking appliances
- Entertainment devices
- Lights
- Locks
- Sensors
- Thermostats

Ogni tipologia di dispositivo può fornire determinate funzionalità, Alexa interagisce con le Skills inviando le richieste relative alla funzionalità indicando l'id del dispositivo a cui si riferisce, il comando e altri valori utili.

2.7.2 AWS Lambda

Le skill devono risiedere su un server sicuro e devono poter comunicare direttamente con il cloud di Alexa, che manderà le richieste HTTPS con i parametri dedlla richiesta in formato JSON.

Come Amazon stessa consiglia, la skill del progetto viene ospitata sui server di Amazon, sfruttando le funzionalità event-driven di AWS Lambda.
AWS Lambda permette di eseguire il codice senza dover gestire il server. Le tariffe sono calcolate secondi i tempi di elaborazione.

Lambda ridimensiona le risorse dell'applicazione mentre esegue il codice in risposta ad ogni trigger. Il codice viene eseguito in parallelo, gestendo separatamente ogni richiesta.

2.7.3 DynamoDB

Amazon DynamoDB è un database con supporto di modelli di dati di dipo key-value e di tipo documento. È un database offerto da Amazon come parte del portfolio di Amazon Web Service (AWS). Offre prestazioni ottime anche in larga scala e dispone di meccaniscmi di replicazione sincrona tra più data center per garantire durability e avaliability.

DynamoDB utilizza una sintassi JSON per la sua semplicità ed ubicuità. Offre solamente operazioni di Put, Get, Delete e Update. L'utilizzo del database DynamoDB è una soluzione molto semplice, e quasi obbligata, quando si necessita di storage dei dati in combinazione con le funzioni Lambda.

2.8 Google Assistant

Google Assistant è un assistente virtuale sviluppato da Google. Deve il suo successo alla ottima integrazione con il sistema operativo Android che gli permette di poter eseguire la maggior parte delle operazioni comuni che vengono effettuate con uno smartphone.

L'Assistente Google è in grado di fornire informazioni, effettuare ricerche su google, riprodurre musica, gestire liste, impostare promemorie e sveglie.

Figura 2.10: Assistente Google su smartphone Android

L'assistente è anche in grado di controllare dispositivi domotici compatibili. A questo scopo Google vende la catena di prodotti Google Home, dei dispositivi che integrano l'assistente vocale e possono operare come controllore per i propri dispositivi domotici compatibili.

Tutti i dispositivi che integrano Google Assistant possono essere attivati pronunciando la wake word "Ok Google".

2.8.1 Action

Nel 2016 Google ha lanciato "Actions on Google", una piattaforma per sviluppare le Action, ovvero il corrispettivo delle Skill Alexa per Google. Permettono di aggiungere maggiore profondità di interazione con l'utente.

Come Alexa fornisce le Api per sviluppare differenti tipologie di skill, Google fornisce API differenti per ogni tipologia di Action. Nel caso dello sviluppo domotico sono utili le API Home Graph.

A differenza di Alexa attualmente google prevede un esperienza d'uso personalizzata per ogni utente. Ad esempio un utente potrebbe chiedere al Google Home mini in cucina "accendi la luce della mia camera" e l'assistente comprenderebbe chi sta parlando dal tono di voce e accenderebbe la luce nella sua stanza. Se Invece la stessa frase l'avesse pronunciata un secondo utente nella stessa abitazione, avrebbe acceso un'altra luce.
Questa funzionalità su Alexa è in via di sviluppo e non è ancora presente in tutte le regioni.

Le Action di Google presentano dei concetti molto simili a quelli visti nelle Skill Alexa, ma la documentazione delle Action risulta essere più confusionaria e meno dettagliata[10]. Questo porta spesso gli sviluppatori a decidere di apliare le funzionalità di Alexa e non quelle di Google.

2.8.2 Firebase

Firebase è una piattaforma di sviluppo web e mobile creata da Google. Offre servizi di analitica, sviluppo, storage, hosting e testing.

Permette di sviluppare le Action per l'assistente Google e di ospitarle sui server proprietari.

Capitolo 3

Design e Implementazione

Si è realizzato un sistema domotico intelligente che permette di integrare dispositivi eterogenei e non necessariamente smart.
I dispositivi possono essere controllati sia tramite comandi vocali, sia tramite un'interfaccia web.

Il sistema domotico è costituito da un dispositivo di controllo, l'arancino, che posizionato all'interno della propria rete domestica gestisce i diversi dispositivi. È possibile aggiungere modularmente dispositivi eterogenei in modo veloce e comodo attraverso l'interfaccia di OpenHAB raggiungibile dalla rete interna sulla porta 8080 oppure anche dalla rete esterna tramite un indirizzo prestabilito. Tramite questa stessa interfaccia è possibile visionare e controllare i dispositivi connessi.

Il dispositivo di controllo integra il broker MQTT Mosquitto, perciò è possibile integrare facilmente dispositivi che utilizzano questo protocollo semplicemente configurandoli dall'interfaccia.

Abilitando la Skill sul proprio Account Amazon è possibile controllare i dispositivi tramite Alexa, quindi anche tramite comandi vocali, utilizzando un qualsiasi dispositivo abilitato, come uno smartphone o un Amazon Echo.

3.0.1 Use Case Diagram

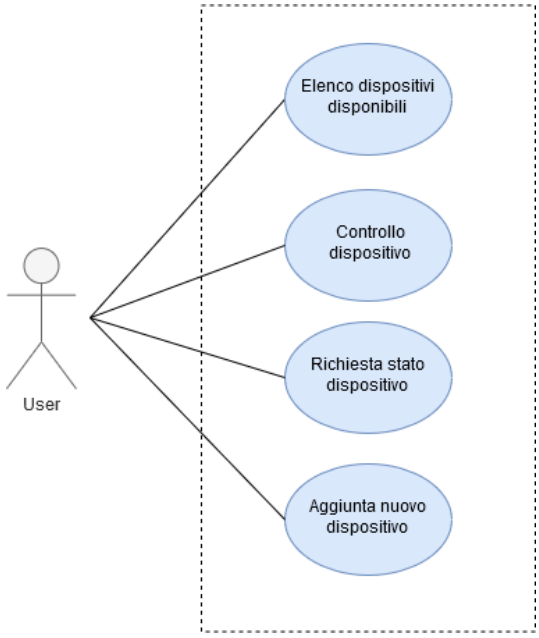

Figura 3.1: Use Case Diagram

3.1 Architettura 1.0

L'architettura iniziale del progetto è differente da quella che poi è stata utilizzata, in quanto in fase di sviluppo sono sorte delle problematiche che saranno chiarite più avanti in questo testo.

L'architettura originale prevedeva:

- L' **Assistente vocale** Alexa che permette di mandare dei comandi ad una skill tramite utilizzo della voce, si può utilizzare tramite l'applicazione per smartphone, come anche tramite gli appositi device Amazon Echo.
- Una **Skill** Alexa che permette di gestire i comandi ricevuti tramite Alexa, è ospitata sui server di Amazon e utilizza il servizio AWS Lambda.

- **IoTronic**, ovvero il sistema progettato dall'università che permette di esporre le porte dell'Arancino alla skill Alexa. Grazie a questo sistema la Skill può comunicare con l'Arancino.

- L'**Arancino** come hardware con sistema operativo linux, fisicamente installato nella rete di casa in modo da permettere la gestione dei dispositivi. Vi sono stati installati e configurati:

  - **OpenHAB**, software per la gestione dei dispositivi, comprendente una rest interface, una web interface, una console di comando, e una bash interface.

  - **Mosquitto**, broker mqtt necessario per lo scambio dei dati tra dispositivi che utilizzano il protocollo mqtt. Il broker è utilizzato anche per scambiare informazioni con la Skill Alexa

  - **NodeRed**, software per la gestione dei dei flussi di informazione. Fornisce un maggiore controllo sulle informazioni in arrivo e in uscita e permette di collegare le informazioni ricevute tramite il broker all'istanza di OpenHAB

- Un dispositivo **ESP32** che funziona da lampadina smart e che utilizza il protocollo MQTT e la Homie convention per ricevere comandi e inviare informazioni sullo stato.

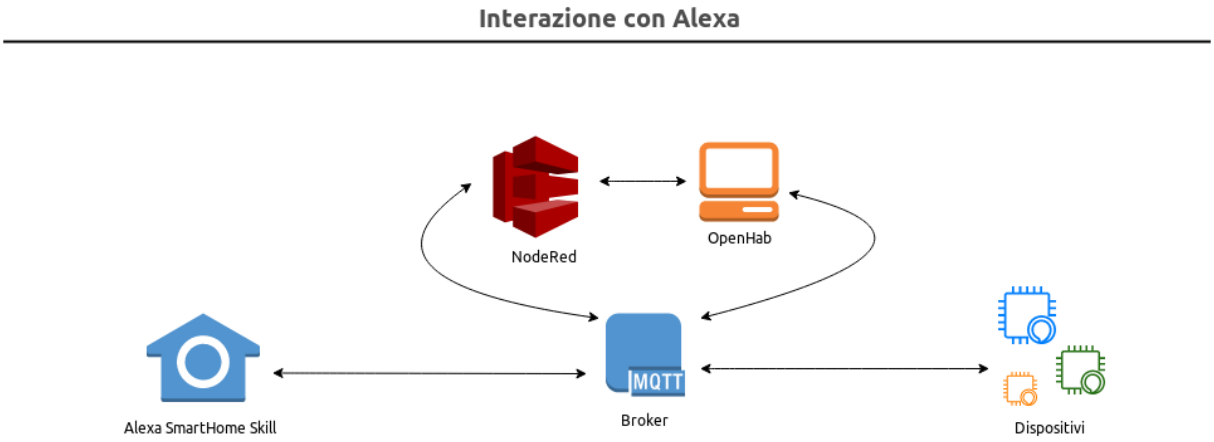

Figura 3.2: Interazione con Alexa

L'architettura prevede un flusso di informazioni che ha inizio dall'utente, che può interagire con Alexa tramite comandi vocali o tramite l'interfaccia dell'applicazione stessa. L'utente può anche comandare direttamente i dispositivi tramite un interfaccia di OpenHAB, così da evitare di interagire con Alexa.

Si supponga che l'utente invii un comando tramite Alexa.
Il comando viene inoltrato da Alexa alla Skill, il software ospitato dai server di Amazon.

La Skill analizza la richiesta e invia il comando richiesto al broker MQTT Mosquitto sul topic 'Alexa/input'.

Per arrivare al broker in funzione sull'Arancino il messaggio passa attraverso il cloud di Stack4Things che permette di esporre alla Skill alcune porte dell'arancino nonostante questo si trovi in una rete locale.

Il broker mqtt ha il compito di inoltrare il messaggio a tutti i subscriber, ovvero NodeRed.

NodeRed riceve il comando e dopo averlo formattato correttamente lo inoltra ad OpenHAB tramite la rest interface.

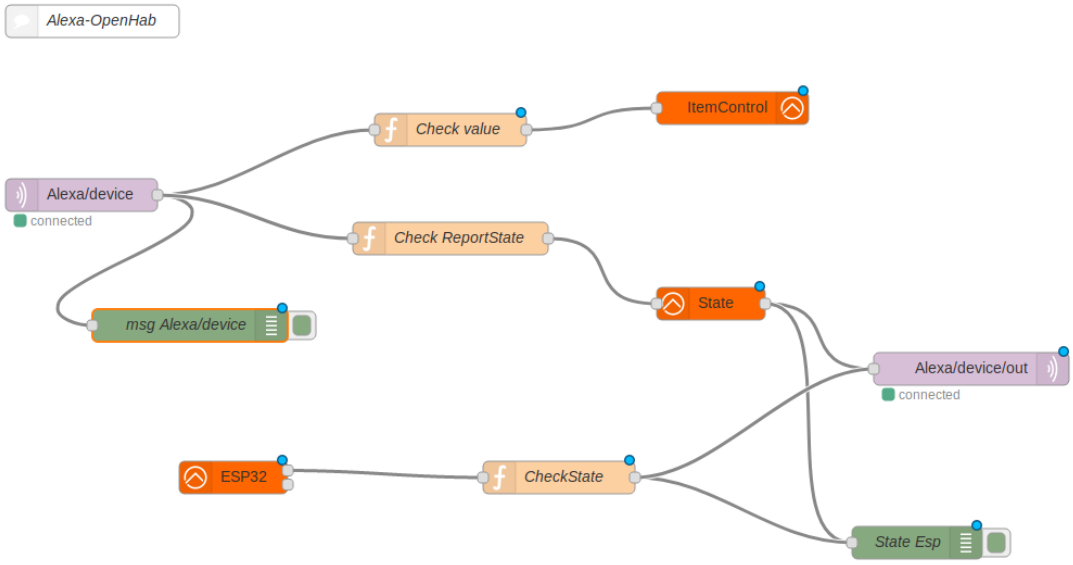

Figura 3.3: Flusso su NodeRed che connette la Skill Alexa

OpenHAB quindi richiede al dispositivo di effettuare l'azione richiesta.

Se, ad esempio, il comando richiesto fosse quello di accendere il dispositivo ESP32, OpenHAB invierebbe il comando di accensione della luce sul broker Mosquitto utilizzando la convenzione homie. Di conseguenza Mosquitto inoltrerebbe il messaggio all'ESP32 in quanto subscriber.

Infine il dispositivo ESP32 accenderebbe la luce e confermerebbe il suo stato.

3.1.1 Sequence Diagram

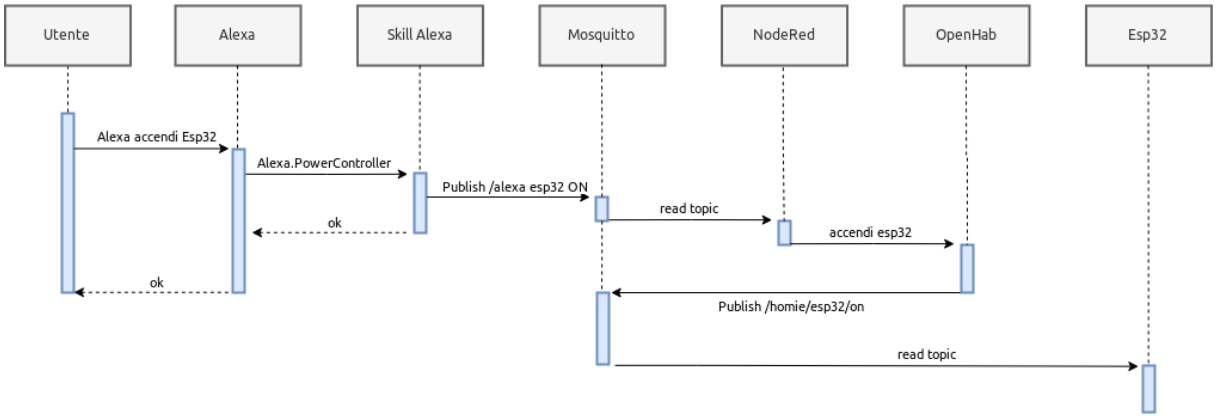

Figura 3.4: Sequence Diagram dell'architettura 1.0

3.1.2 Component-Deployment Diagram

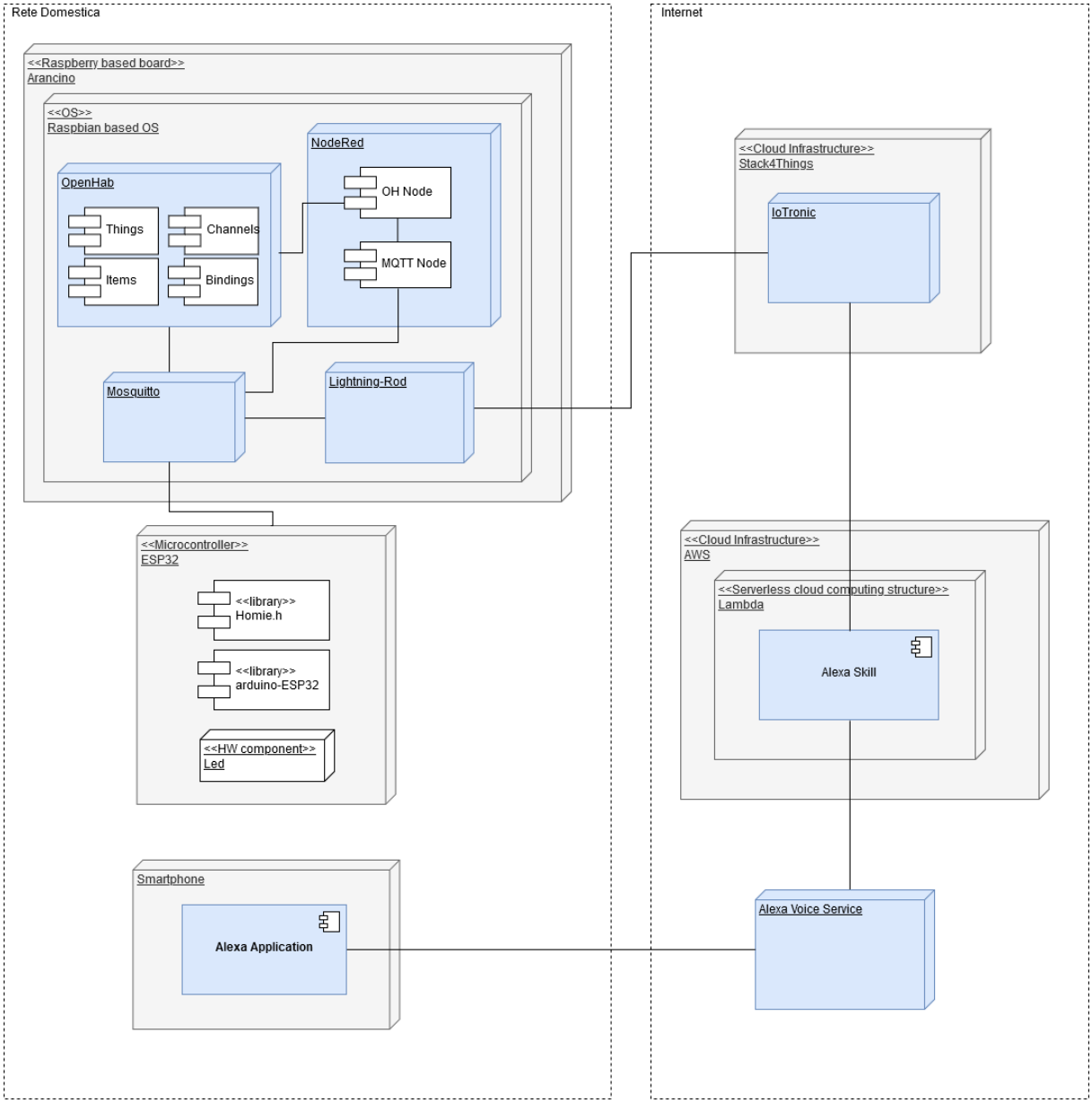

Figura 3.5: Deployment Diagram dell'architettura 1.0

3.1.3 Activity Diagram

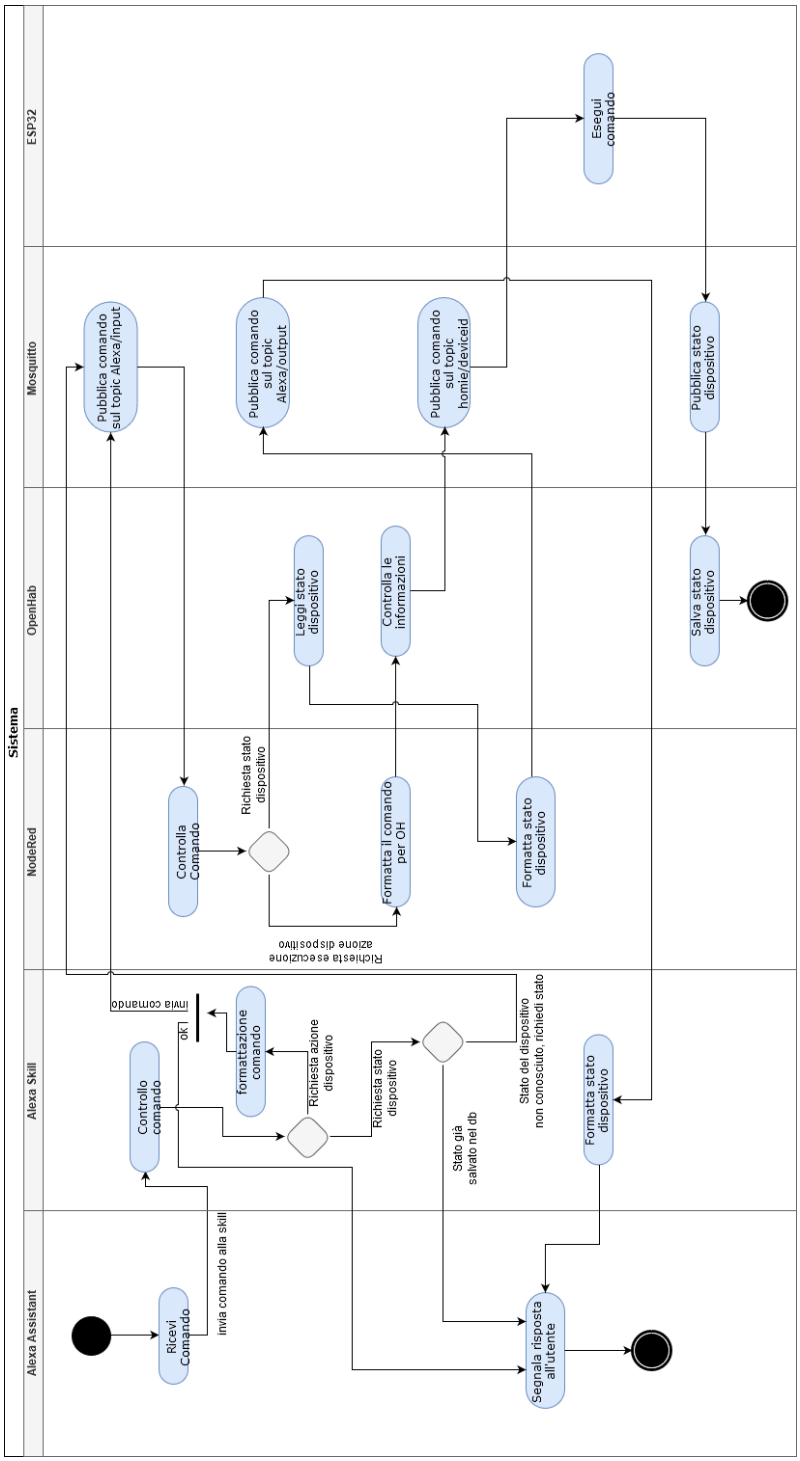

Figura 3.6: Activity Diagram dell'esecuzione di un comando nell' architettura 1.0

3.1.4 Skill

La Skill è stata sviluppata in python per funzionare con il servizio Lambda ed è in grado di gestire richieste sia dalla vecchia generazione di dispositivi Alexa (v.2) sia dalla nuova generazione (v.3).

La Skill sviluppata è solamente un prototipo, scritto per verificare il funzionamento dell'architettura basata su MQTT.

Per gestire i messaggi verso e dal broker MQTT è stata utilizzata la libreria open source paho-mqtt.

Il prototipo non è in grado di rispondere ad una richiesta Discovery con tutti i dispositivi configurati sull'Arancino, ma solamente con il dispositivo di test. Inoltre è in grado di rispondere correttamente solamente a richieste di tipo "Alexa.PowerController" e "Alexa.ReportState" le quali permetterebbero di dimostrare il funzionamento dell'architettura.

Il prototipo ha dimostrato il correntto funzionamento del collegamento tra la Skill e l'Arancino attraverso il cloud Stack4Things, infatti le richieste di accensione e spegnimento dispositivo sono sempre state risolte correttamente.

Il prototipo ha però messo in luce un problema diverso, ovvero l'impossibilità di effettuare il subscribe da parte della skill ad un topic utilizzando la libreria paho-mqtt. O meglio, il subscribe è stato effettuato, ma la connessione al broker veniva interrotta dopo poco tempo. Questo a causa delle politiche di funzionamento di AWS Lambda. Questo comporta che la Skill non riesce a ricevere correttamente le informazioni dall'arancino, ma solo a inviarle.

Quando un cambiamento di stato da parte di un dispositivo viene segnalato sul topic MQTT allora la Skill conserva lo stato sul database DynamoDB.

Non appena la skill riceve una richiesta di stato di un dispositivi allora questa controlla

se vi sono entry nel DataBase, in caso contrario la Skill richiede lo stato del dispositivo utilizzando sempre il protocollo MQTT.

Le informazioni su DataBase, a cusa del problema precedentemente descritto, sono risultate quasi sempre obsolete se non totalmente assenti. Quindi il modello architetturale non funziona correttamente.

Publish del comando:

```
def handle_PoweSet(request,value):
    appliance = request["directive"]["endpoint"]["endpointId"]
    print("appliance="+appliance)

    client1= mqtt.Client("Alexa_skill_publisher")
    client1.connect(broker,port) #establish connection
    ret= client1.publish("Alexa/device",
    ↪  json.dumps({"device":appliance,"operation":"powerset", "value":value }))
    ↪  #publish

    print ("value="+value)
    return value
```

salvataggio dello stato sul DB:

```
appliance = request["directive"]["endpoint"]["endpointId"]
print("appliance="+appliance)
print("get_item")
item = db.get_item(TableName="DeviceState",Key={'DeviceId':{'S':appliance}});
print(item)
return item['Item']['State']['S']
```

3.2 Architettura 2.0

Il modello di architettura precedentemente esposto si è rivelato poco funzionale. Sebbene il protocollo MQTT si presti molto bene allo scambio di messaggi in ambito IoT, la struttura stessa del cloud serverless event-driven Lambda rende impossibile il corretto funzionamento del subscribe ad un topic MQTT, ma questo è stato evidente solo una volta compilato ed eseguito il prototipo della skill

Per questo motivo l'architettura è stata progettata nuovamente per sfruttare, invece, le REST API fornite da OpenHAB.

In pratica il design della nuova architettura ha permesso, in fase di sviluppo, di sfruttare la Skill open source già testata e funzionante prodotta da OpenHAB stesso. Nonostante fosse stata ideata per essere utilizzata con il Cloud OpenHAB, la combinazione OpenHAB - IoTronic ha svolto egregiamente il suo lavoro.

La nuova architettura prevede:

- L' **Assistente vocale Alexa** che permette di mandare dei comandi ad una skill tramite utilizzo della voce, si può utilizzare tramite l'applicazione per smartphone, come anche tramite gli appositi device Amazon Echo.
- Una **Skill** Alexa che permette di gestire i comandi ricevuti tramite Alexa, è ospitata sui server di Amazon e utilizza il servizio AWS Lambda.
- **IoTronic**, ovvero il sistema progettato dall'università che permette di esporre le porte dell'Arancino alla skill Alexa.
- L'**Arancino** come hardware con sistema operativo linux, fisicamente installato nella rete locale in modo da permettere la gestione dei dispositivi utilizzando:
  - **OpenHAB**, software per la gestione dei dispositivi, comprendente una rest interface, una web interface, una console di comando, e una bash interface.
  - **Mosquitto**, broker mqtt necessario per lo scambio dei dati tra dispositivi che utilizzano il protocollo mqtt.

- Un dispositivo **ESP32** programmato per funzionare come una lampadina smart e che utilizza il protocollo mqtt e la homie convention per ricevere comandi e inviare informazioni sullo stato.

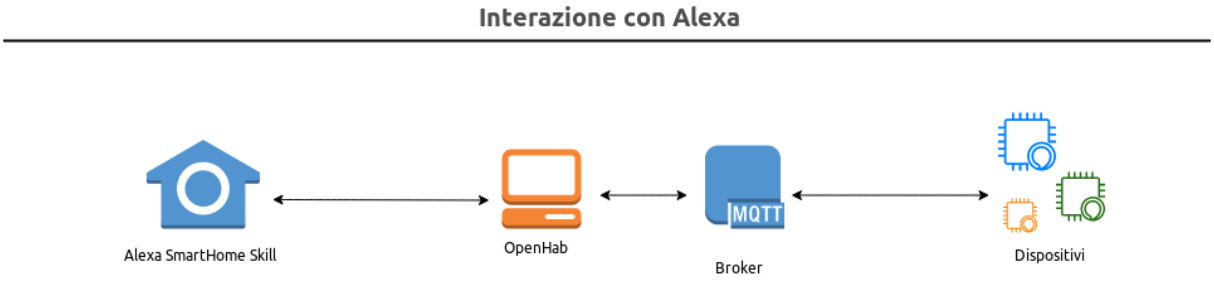

Figura 3.7: Interazione con Alexa

L'utente può richiedere un servizio all'assistente vocale Alexa, come ad esempio l'accenzione di una lampadina. Anche con questa implementazione l'utente può interagire direttamente con OpenHAB, bypassando Alexa.

Supponiamo che l'utente mandi un comando ad Alexa.
Il comando viene inviato da Alexa alla Skill che lo formatta e lo inoltra direttamente alla REST interface di OpenHAB (con l'ausilio di IoTronic) presente sull'Arancino. OpenHAB porta a compimento il comando richiedendo al dispositivo indicato di effettuare l'azione richiesta.

Ipotizzando che il dispositivo indicato dall'utente sia la lampadina smart con chip ESP32, il comando è inviato al broker sul topic homie/deviceid/funzione. Quindi Mosquitto inoltra il messaggio all'ESP32 in quanto subscriber di quel topic. Il dispositivo esegue l'azione richiesta (accende/spegne la luce) e conferma il suo stato.

3.2.1 Sequence Diagram

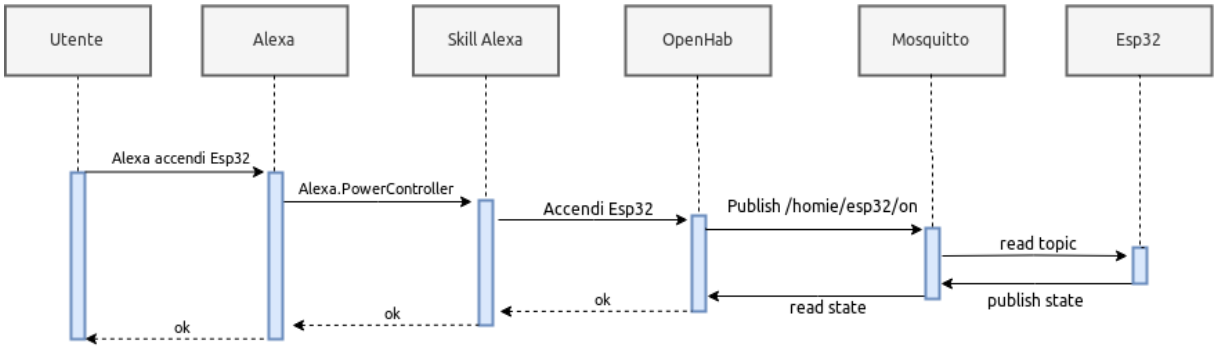

Figura 3.8: Sequence Diagram dell'architettura 2.0

3.2.2 Component-Deployment Diagram

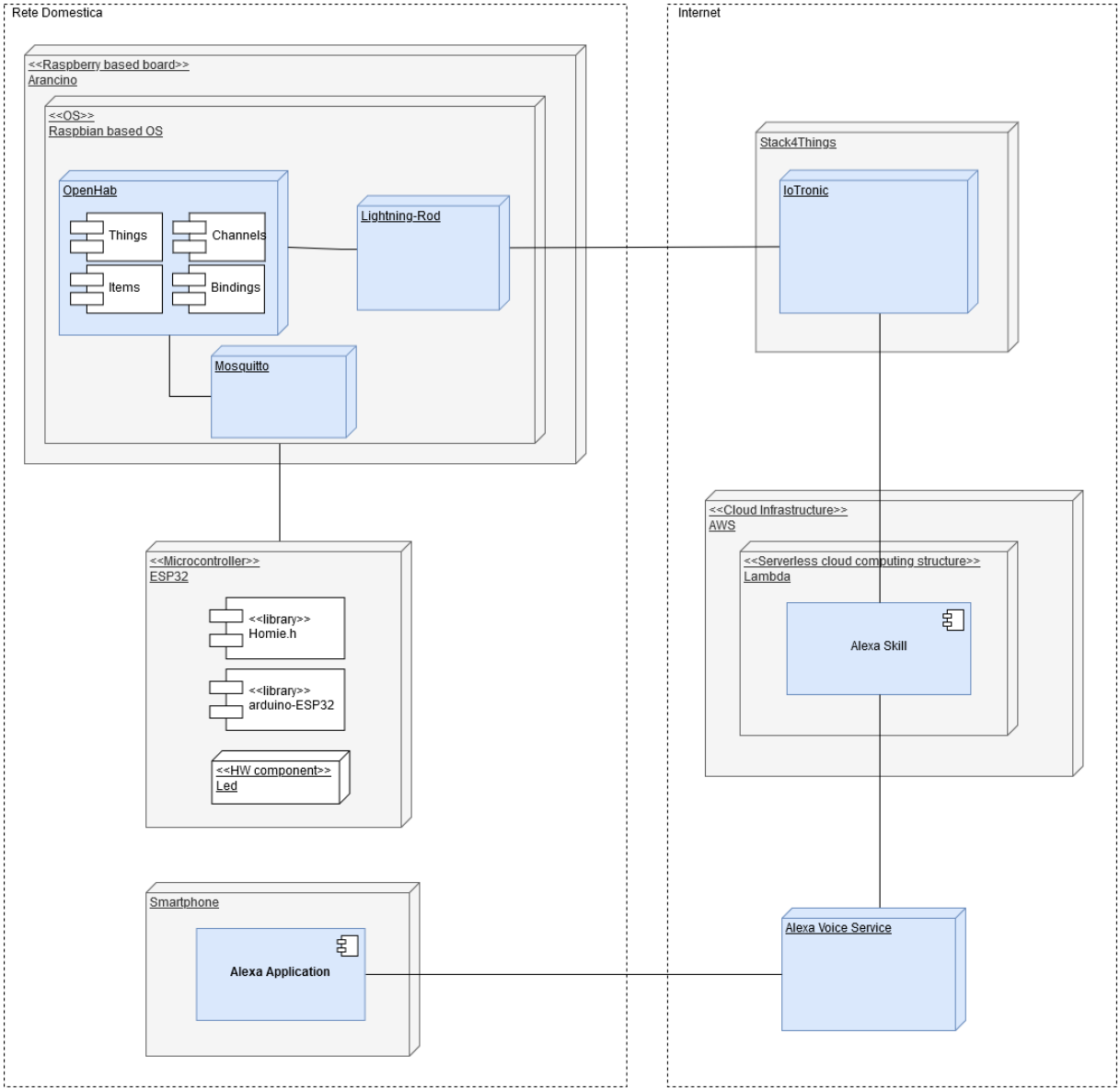

Figura 3.9: Deployment Diagram dell'architettura 2.0

3.2.3 Activity Diagram

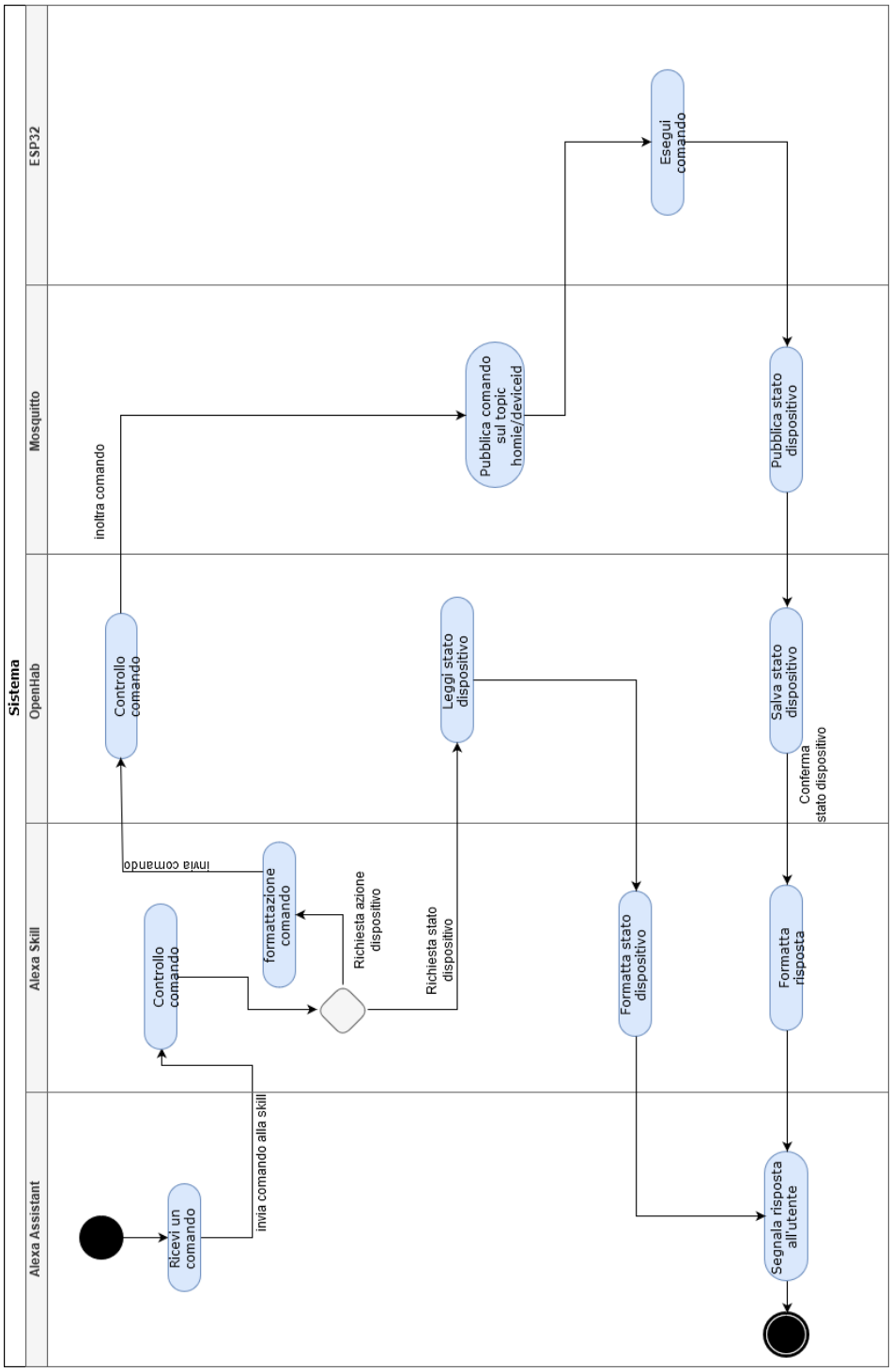

Figura 3.10: Activity Diagram dell'esecuzione di un comando nell' architettura 2.0

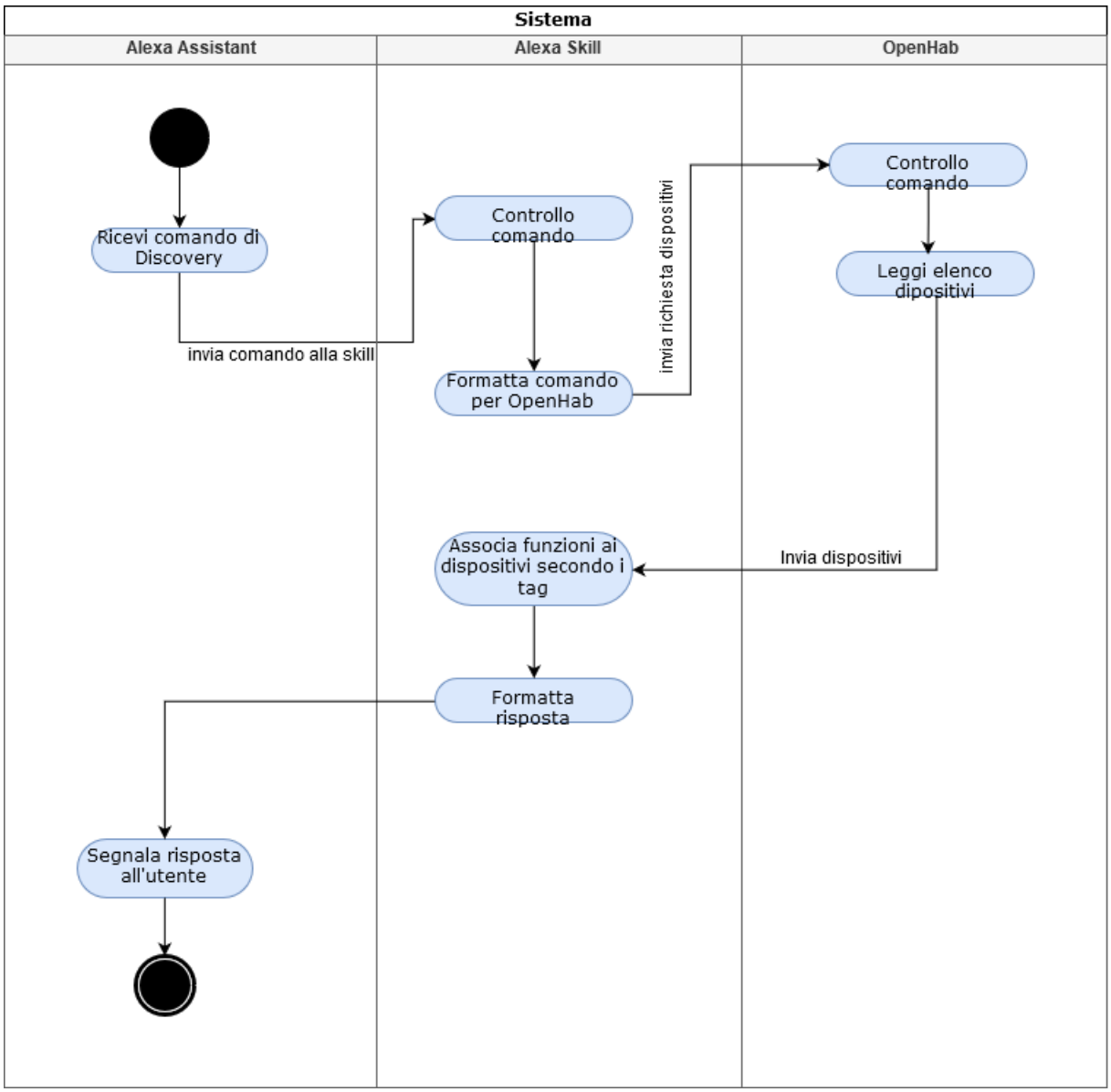

Figura 3.11: Activity Diagram del Discovery degli Item

3.2.4 Skill

La Skill[20] utilizzata è una applicazione Nodejs che connette Alexa Smart Home API con un'istanza di OpenHAB. È in grado di rispondere a richieste sia della versione v2 sia v3. Per comprenderne meglio il funzionamento è stato effettuato il reverse engineering del codice che ha portato allo sviluppo del seguente diagramma.

Component Diagram

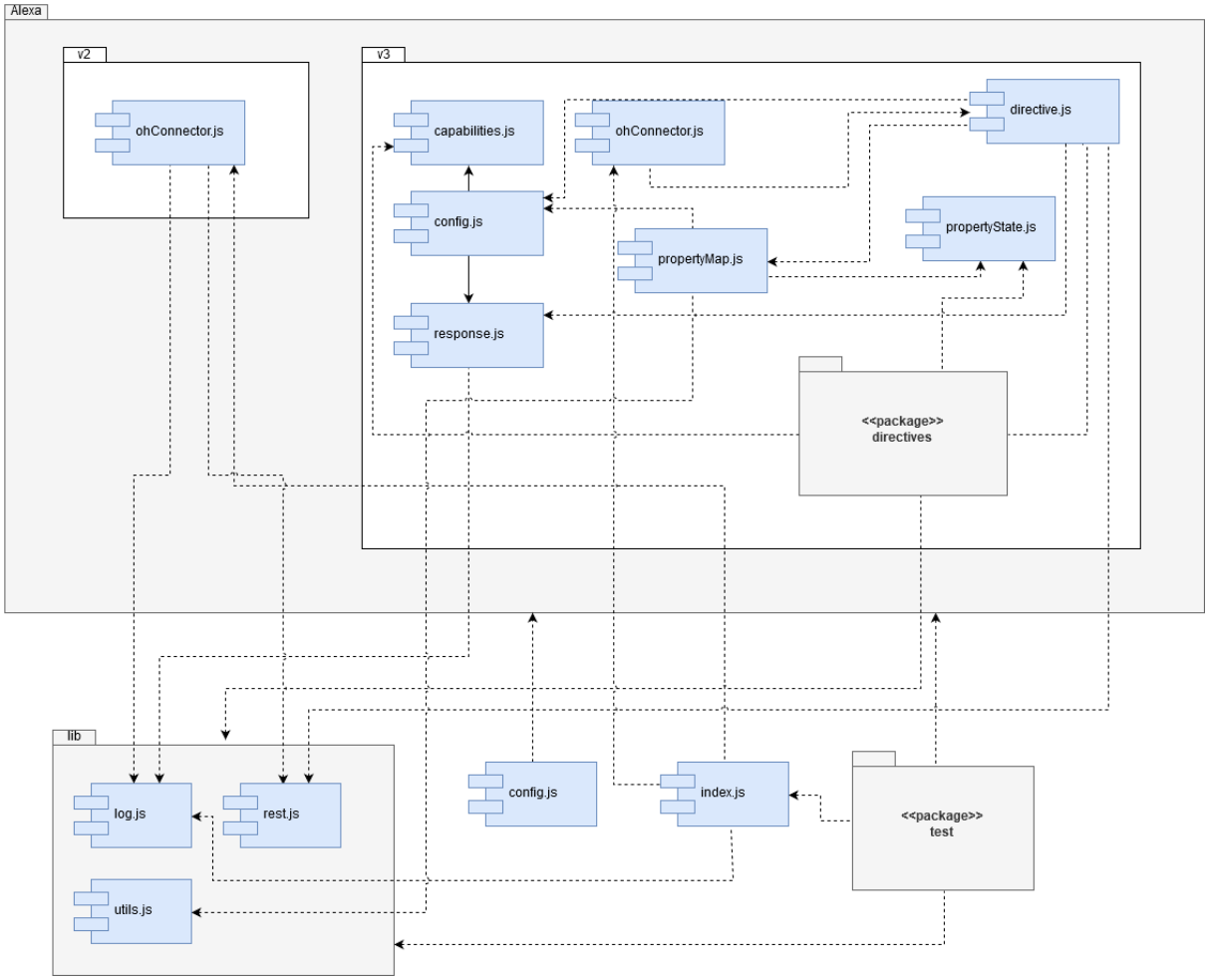

Figura 3.12: Component Diagram della Skill per Alexa sviluppata da OpenHAB

3.2.5 Tag OpenHAB

Per integrare la Skill con OpenHAB e fornire maggiori informazioni ad Alexa sulle specifiche dei componenti OpenHAB fornisce dei tag. Questi sono necessari affinchè la skill di Alexa mostri correttamente il dispositivi e ne utilizzi le funzionalità.

I tag possono essere aggiunti ad un Item utilizzando la rest interface oppure la console di comando di OpenHAB.

Esistono numerosi tag, ne sono elencati alcuni di seguito, il loro scopo è facilmente intuibile dal nome stesso.

- Switchable
- Lighting
- Lock
- CurrentHumidity
- CurrentTemperature
- EntertainmentChannel
- MediaPlayer
- MotionSensor
- WaterAlarm

Purtroppo la disponibilità delle funzionalità su Alexa varia da regione a regione, perciò non tutti i tag sono disponibili in tutte le regioni.

Capitolo 4

Test

4.1 Discovery dei Dispositivi

Per testare il funzionamento del sistema sono stati programmati, e configurati su OpenHAB, due dispositivi.

Per poterli utilizzare con Alexa è necessario richiedere la ricerca dei dipositivi tramite comandi vocali o l'app Alexa. Quest'ultima invierà una richiesta di discovery alla skill che a sua volta chiederà le informazioni all'istanza di OpenHAB. La skill riporterà quindi ad Alexa i dispositivi correttamente configurati con le funzionalità previste dai tag.

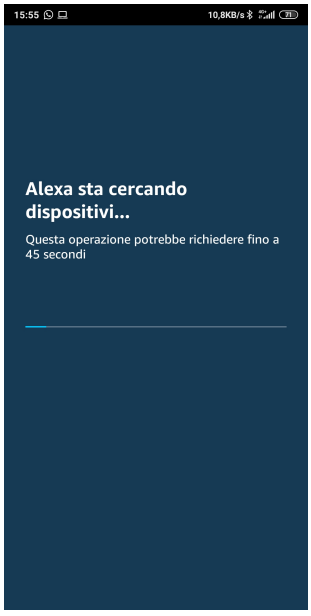

Figura 4.1: Ricerca dispositivi tramite app

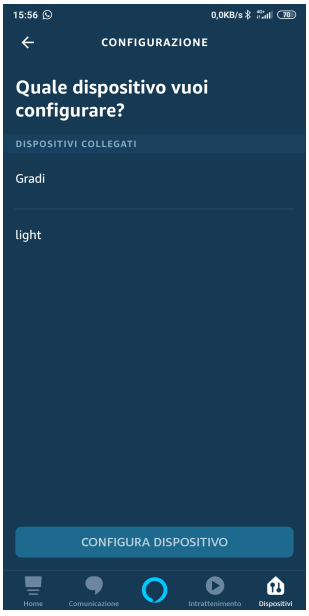

Figura 4.2: Dispositivi trovati

Alexa permette di suddividere in gruppi i dispositivi in modo da identificarli più facilmente. Questo permette anche di effettuare delle azioni sull'intera lista di dispositivi appartenenti ad un determinato gruppo.
Un esempio potrebbe essere il chiedere di spegnere tutte le luci del gruppo dedicato alla cucina.

Figura 4.3: Dispositivi divisi in gruppi tramite Alexa

4.2 Dispositivo Luce

Il Dispositivo "Luce", utilizzato per testare l'architettura, è stato creato utilizzando una scheda con chip ESP8266. Questa, è stata programmata in linguaggio C attraverso l'IDE Arduino.

Per comunicare con il broker MQTT è stato utilizzato il branch develop-v3 della libreria homie-esp8266 [12] che sfrutta la versione 3 del protocollo MQTT e la convenzione homie.

Una volta connesso alla rete WiFi e al broker MQTT, il dispositivo si "presenta" secondo la convenzione homie. Accende la luce led se riceve il valore 'true' per il parametro 'on', mentre la spegne se riceve il valore 'false'.

Affinchè la skill di Alexa comprenda le funzionalità del dispositivo gli sono stati assegnati i tag "Lighting" e "Switchable".

Figura 4.4: Dialogo con Alexa per spegnere la luce

Figura 4.5: Luce spenta sull'app Alexa

Una volta inviato il comando al dispositivo tramite Alexa, anche su OpenHAB il cambiamento si riflette sull'interfaccia in tempo reale e il dispositivo esegue correttamente il comando richiesto.

Figura 4.6: Schermata OpenHAB con la luce spenta

Il dispositivo è stato programmato per inviare anche tramite porta seriale le informazioni inviate e ricevute.

```
Light is on
 Sending statistics...
  • Interval: 65s (60s including 5s grace time)
  • Wi-Fi signal quality: 98%
  • Uptime: 67s
Light is off
Light is on
Light is off
Light is on
```

Figura 4.7: Messaggi dell' ESP8266 su porta seriale

Come discusso precedentemente, i dispositivi comunicano utilizzando la convenzione homie su protocollo MQTT, il risultato del subscribe al topic homie/# è il seguente:

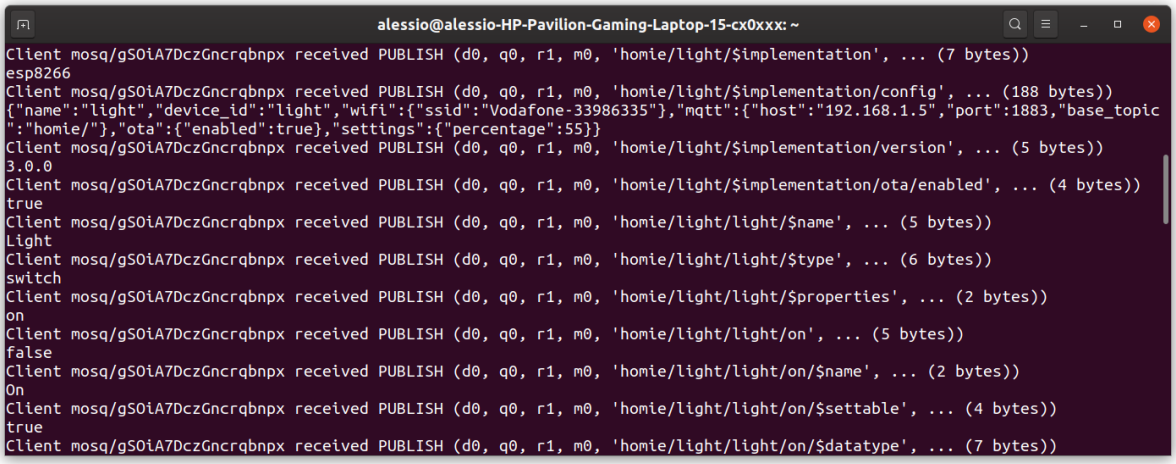

Figura 4.8: Informazioni del dispositivo Luce inviate tramite MQTT

4.3 Dispositivo Temperatura

Sempre al fine di verificare che tutto funzioni correttamente è stato programmato un dispositivo ESP8266 per simulare un sensore di temperatura. Anche questo si connette al broker MQTT, utilizzando la convenzione homie, grazie alla libreria homie-esp8266 [12].

Il dispositivo invia informazioni riguardanti la temperatura in gradi Celsius.

Per far presente le funzionalità del dispositivo alla Skill gli è stato assegnato su OpenHAB il tag “CurrentTemperature”.

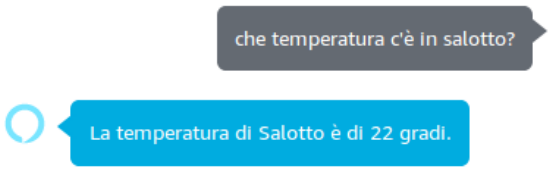

Figura 4.9: Richiesta della temperatura del Salotto con comandi

Figura 4.10: Temperatura mostrata sull’app Alexa

Come previsto i dati ricevuti da Alexa sono coerenti con quelli mostrati su OpenHAB.

Figura 4.11: Schermata di OpenHAB con il monitoraggio della temperatura

Il subscribe al topic homie/# mostra lo scambio di informazioni tra i dispositivi.

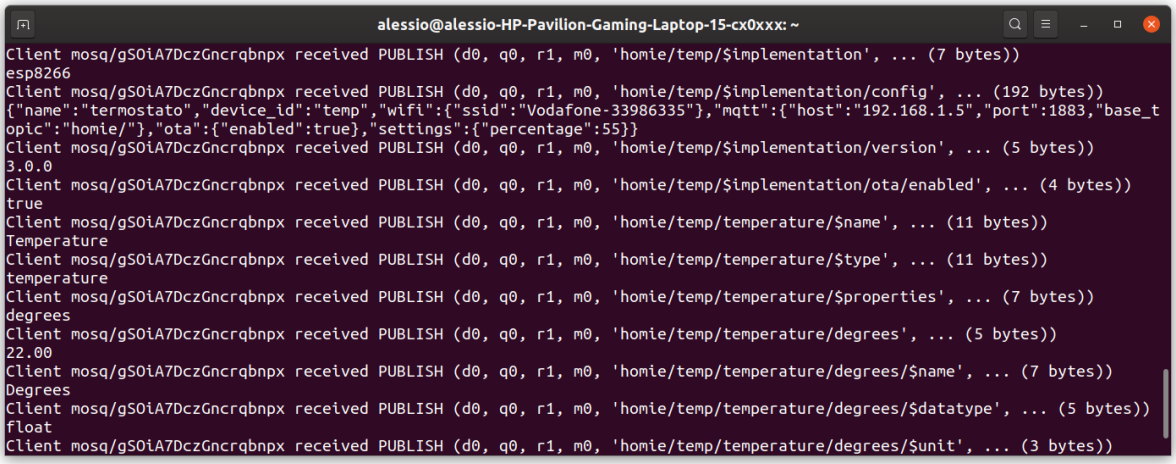

Figura 4.12: Informazioni del dispositivo Temperatura inviate tramite MQTT

4.4 Tempi di risposta

Sono stati effettuati anche dei test quantitativi per stimare i ritardi di risposta della Skill Alexa.

E' stato prodotto un grafico con i ritardi di 100 richieste di accensione del dispositivo luce precedentemente mostrato.

Per effettuare questi test è stato utilizzato il seguente script bash:

```
1  n=$1
2  echo 'start test singolo '$n' volte' >> result.txt
3  i=0
4  # esegui i test a distanza di 1 secondo
5  while [ $i -lt $n ]
6  do
7          ./test.sh
8          sleep 1
9          i=$(($i+1))
10 done
```

```
11
12 echo 'end test singolo '$n' volte' >> result.txt
```

Questo script esegue n volte (100) lo script il seguente script bash che effettua l'effettivo test:

```
1 start=$(($(date +%s%N)/1000000))
2 aws lambda invoke --function-name
  ↪ arn:aws:lambda:eu-west-1:290228652647:function:alexa-openhab --invocation-type
  ↪ RequestResponse --payload file://input.json response.json
3 end=$(($(date +%s%N)/1000000))
4 time=$((end - start))
5 echo $time >> result.txt
```

Lo script "test.sh" chiama la funzione di AWS CLI che invia il comando in formato JSON alla skill. Al termine salva su un file il tempo trascorso in millisecondi.
Il JSON per richiamare la skill si può esaminare in appendice B.3

I tempi di risposta utilizzati corrispondono al periodo di tempo indicato in rosso nel seguente diagramma di attività:

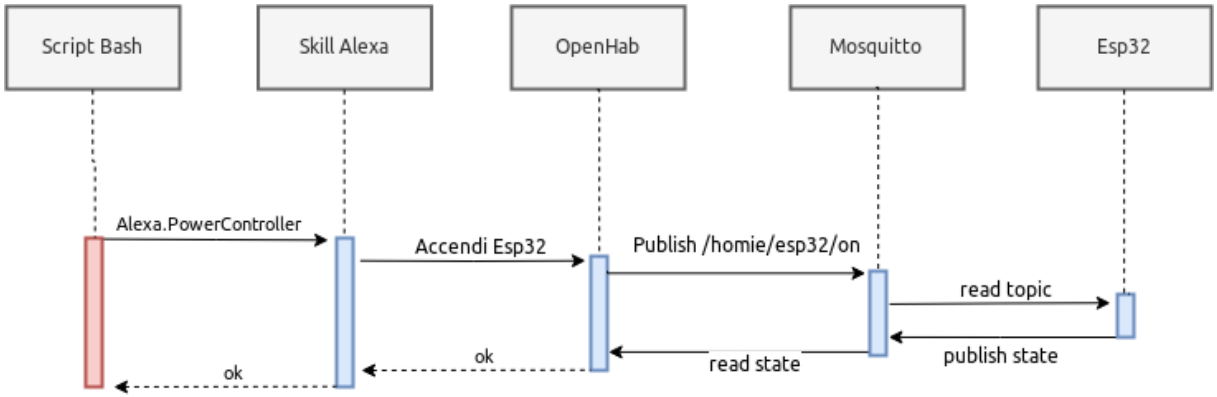

Figura 4.13: Activity Diagram della richiesta di accensione luce

Il ritardo è indicato sull'asse delle y ed è rappresentato in millisecondi.

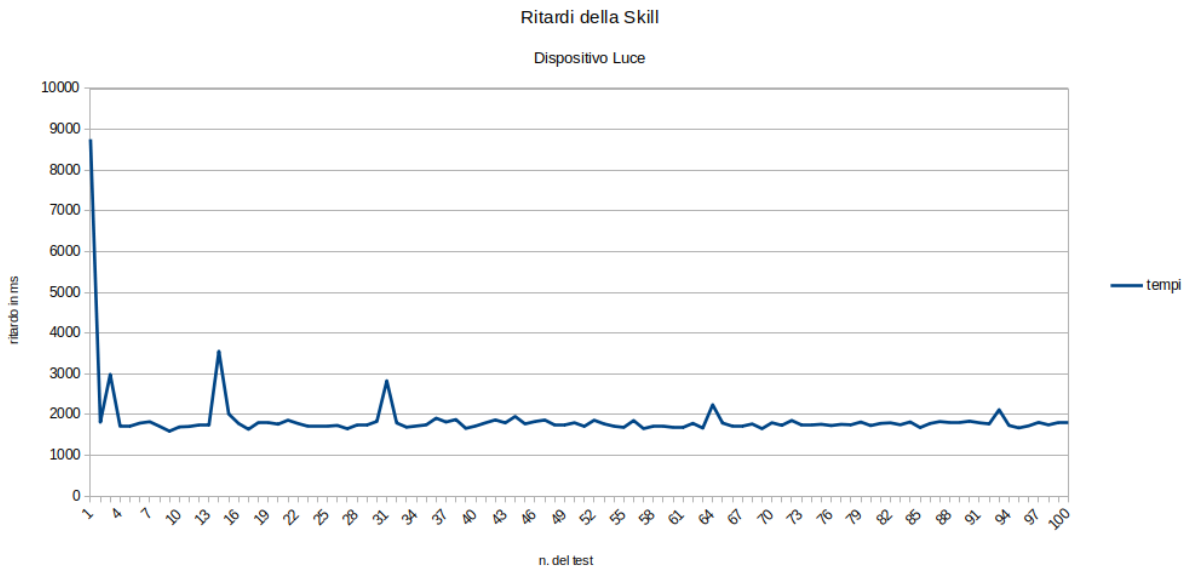

Figura 4.14: Tempi di risposta della skill per il comando di accensione luce

Il grafico mostra un andamento pressocchè costante a parte alcuni picchi di maggiore ritardo dovuto probabilmente ad un momento di congestione della rete o dei server Amazon. I dati mostrano un **ritardo medio di 1883,41 ms**.

E' stato effettuato lo stesso test con il dispositivo temperatura. In questo caso OpenHAB non è però costretto a colloquiare con il dispositivo ogni volta che intende conoscere la temperatura. Viene piuttosto indicata soltanto l'ultima stima che il dispositivo ha comunicato ad OpenHAB.
Il JSON utilizzato per effettuare questo testi si può trovare in appendice B.4

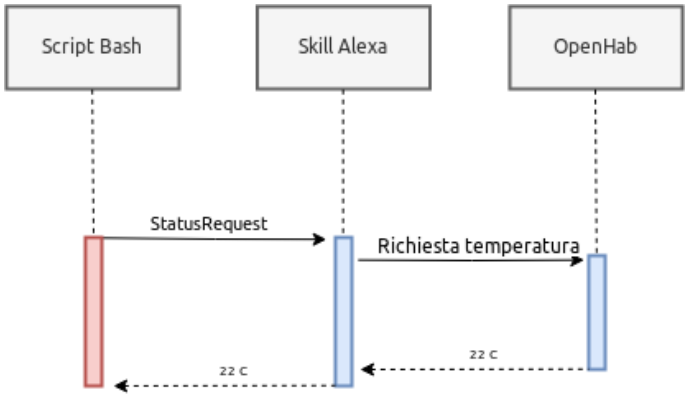

Figura 4.15: Activity Diagram della richiesta della temperatura

Come mostrato nel diagramma di attività vi è un minore dialogo tra i componenti, che porta a dei ritardi nettamente inferiori. Si parla di un **ritardo medio di 1288,29 ms** contro il ritardo di 1883,41.

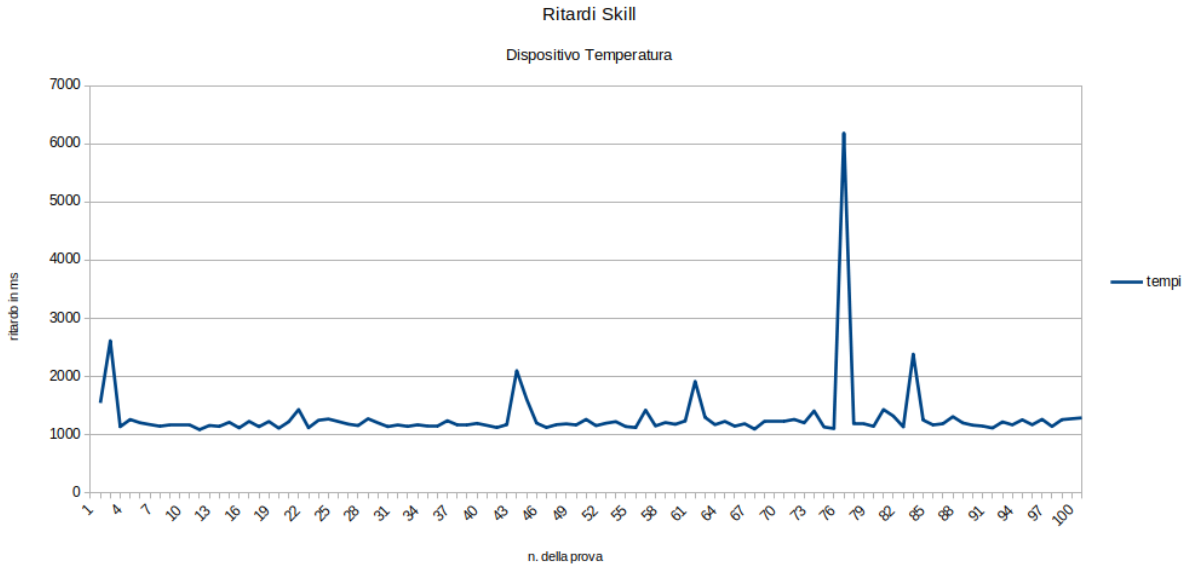

Figura 4.16: Tempi di risposta della skill per la richiesta di temperatura

Questo mostra come il ritardo aggiunto dal cloud Stack4Things sia minimo rispetto alle altre componenti e che, in ogni caso, i ritardi sono accettabili per un impianto domotico.

Guardando i due grafici i picchi di tempo iniziale possono anche far pensare ad una sorta di caching da parte della servizio Lambda. Chiaramente il tempo percepito dall'utente sarà leggermente maggiore in quanto a questi tempi si aggiungeranno quelli di elaborazione da parte dell'intelligenza virtuale di Alexa. Questa dovrà interpretare il linguaggio e comprendere il significato della frase. Inoltre si potrebbero aggiungere dei ritardi dovuti ad eventi aleatori, come congestione della rete o guasti.

4.4.1 Test con richieste multiple

Per testare la degradazione delle prestazioni della skill in caso di richieste contemporanee si sono effettuati dei test attui a ricreare il fenomeno in questione.
I test sono stati effettuati sulla richiesta della temperatura.

Lo script utilizzato per effettuare questo test:

```
1  n=$1
2  echo 'start test '$n' volte' >> result.txt
3  i=0
4  # esegui i test salva i pids in array
5  while [ $i -lt $n ]
6  do
7          ./test.sh &
8          i=$(($i+1))
9          pids[${i}]=$!
10 done
11
12 # wait per ogni processo
13 for pid in ${pids[*]}; do
14     wait $pid
15 done
16
17 echo 'end test '$n' volte' >> result.txt
```

Questo script richiama n volte lo script riguardante il test singolo mostrato in precedenza. Questa volta non aspetta il termine di ogni test, ma richiama tutti i test insieme

creando un thread separato per ognuno.
Infine attende la terminazione dell'ultimo processo. Di seguito il grafico generato dai dati ricavati dal test:

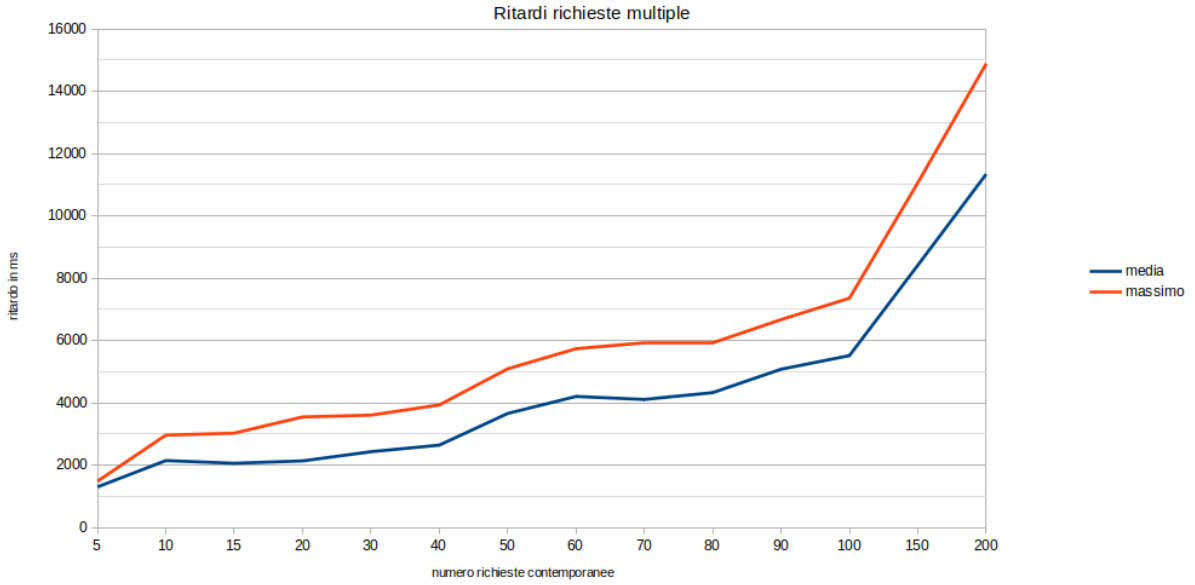

Figura 4.17: Caption

Come si può vedere dal grafico vi è un effettivo rallentamento dei tempi di risposta per molte richieste contemporanee, la degradazione è però accettabile per il livello di traffico atteso da un software come quello in questione. Aumentando però il numero di richieste contemporanee inviate alla skill aumentano anche le variabili in gioco. Inviando contemporaneamente diverse richieste http da un singolo endpoint si rischia di sovraccaricare la rete e creare quindi un rallentamento dei tempi di risposta, non dovuto solamente ai server di Amazon.

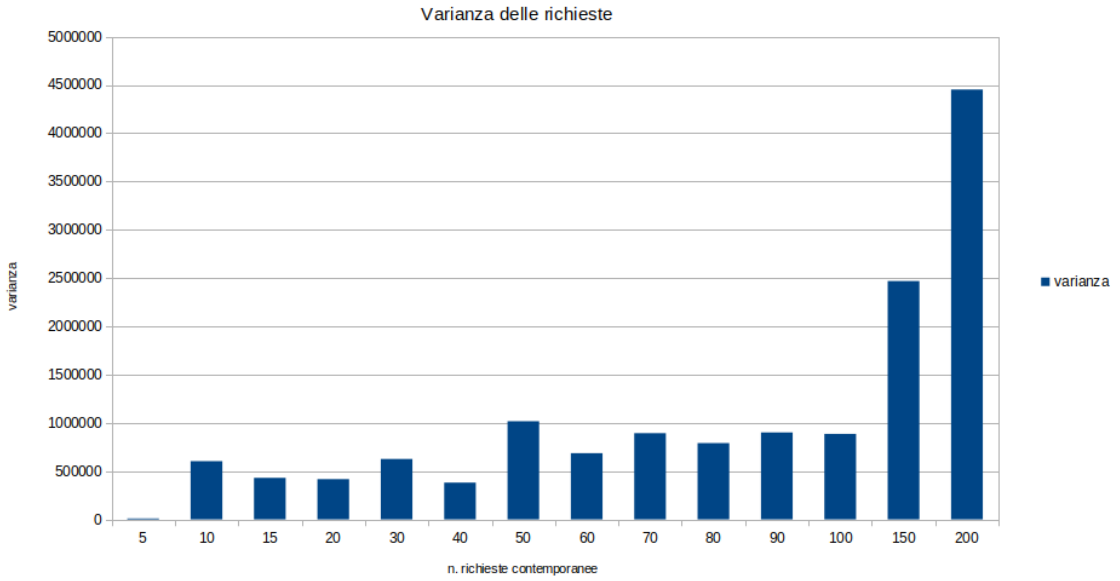

Figura 4.18: Caption

Questo grafico mostra la varianza riscontrata tra i dati ricavati dal test precedente. Si può notare che tra le 10 e le 100 richieste contemporanee la varianza rimane pressocchè stabile, mentre dopo i 150 la varianza aumenta notevolmente. Questo potrebbe essere segno che la degradazione delle prestazioni dipenda anche da fattori esterni alla skill.

Capitolo 5

Conclusione e sviluppi futuri

E' stato effettivamente portato a compimento lo sviluppo di un sistema domotico modulare capace di connettere sistemi eterogenei con facilità. Al sistema è stata aggiunta la possibilità di interazione tramite l'assistente digitale Alexa e sono stati effettuati con successo diversi test.

Oltre a portare un risultato concreto, lo sviluppo di questo progetto ha fornito al candidato una visione più ampia delle tecnologie attuali, toccando quindi framework, protocolli e sistemi mai incontrati nel corso del proprio percorso universitario ma che, grazie alle fondamenta create dal percorso stesso, non sono risultate di difficile approccio .

La Domotica è un campo in continuo mutamento e sviluppo, verranno introdotte sicuramente delle novità che porteranno il progetto a continuare ad espandersi.
Durante lo stesso svolgimento di questa tesi, le API di Alexa sono state modificate più volte, quindi tutto lascia credere che continueranno a mutare.

Già da ora è possibile apportare miglioramenti e modifiche, si potrebbe ad esempio aggiungere la possibilità di interagire con il sistema tramite altri assistenti vocali oltre ad Alexa. Lo sviluppo della Action per Google Assistant è già incominciato, ma per questioni di tempistiche è stato temporaneamente tralasciato.

Una funzionalità importante, che non è stata prevista all'interno dello sviluppo di questo progetto, è la multi-tenancy. Stack4Thing è in grado di provvedere ad esigenze come questa, perciò si dovrebbe "solamente" prevedere la multi-tenancy nella Skill Alexa, in modo che ognuno possa utilizzare indipendentemente il proprio sistema domotico. La skill avrebbe potuto funzionare in modo indipendente per ogni utente già da adesso se Alexa avesse previsto la possibilità, anche in Italia, di poter pubblicare delle skill private, ovvero Skill che prevedono un'istanza separata della stessa per ogni utente. Ma purtroppo questa funzione resta ancora non disponibile in questo paese.

Il sistema domotico qui discusso è altamente modulare, perciò il miglioramento del sistema potrebbe anche applicarsi ai dispositivi, inserendone di nuovi con hardware e software differenti.

Ringraziamenti

A conclusione di questo elaborato, vorrei dedicare una pagina a coloro che mi hanno permesso di raggiungere questo traguardo.

Ringrazio il Prof. Antonio Puliafito per avermi permesso di portare a compimento questo progetto.

Un ringraziamento particolare va all' Ing. Giovanni Merlino, per i suoi preziosi consigli e correzioni.

Grazie anche al Dott. Mauro Staiti che mi ha messo sui giusti binari nello sviluppo del progetto.

Vorrei ringraziare di cuore la mia famiglia, che mi ha sempre sostenuto nel conseguire i miei sogni.

A mia Mamma, che mi ha insegnato a puntare alto, a non accontentarmi mai, a pormi sfide sempre più grandi, perché niente è impossibile.

Al mio Papo, un'ancora in un mare in tempesta. La costante in un mondo di variabili. Perché su di lui ho sempre fatto affidamento e sempre lo farò.

A mia Sorella che, nonostante ci possano separare regioni, mari e forse un giorno continenti, saremo sempre vicini e potremo sempre contare l'uno sull'altro.

Alle mie Zie Giulia e Lucia che mi hanno insegnato ad essere sempre genuino e ad

apprezzare le cose che il mondo mi offre. Perché mi hanno dimostrato come essere Dottore nello spirito e nell'anima, non nel vestito.

A Nonno Pinuccio, che mi ha insegnato ad essere curioso, che l'età è solo un numero e che non si smette mai di imparare.

Ai miei Amici che mi hanno aiutato e supportato anche quando sono stato insopportabile.

A Marco, ormai un fratello, perché la nostra competizione mi ha dato la forza di lottare e di essere qui oggi. Continuerò a invidiarti e a starti vicino.

Appendice A

Codici nel testing

A.1 Dispositivo Luce

Di seguito il codice caricato sul dispositivo di test

```
#include <Homie.h>

#define firmwareVersion "1.0.0"
const int PIN_RELAY = LED_BUILTIN;

HomieNode lightNode("light", "Light", "switch");

bool lightOnHandler(const HomieRange& range, const String& value) {
  if (value != "true" && value != "false") return false;

  bool on = (value == "true");
  digitalWrite(PIN_RELAY, !on ? HIGH : LOW);
  lightNode.setProperty("on").send(value);
  Homie.getLogger() << "Light is " << (on ? "on" : "off") << endl;

  return true;
}

void setup() {
  Serial.begin(9600);
```

```
  Serial << endl << endl;
  pinMode(PIN_RELAY, OUTPUT);
  digitalWrite(PIN_RELAY, LOW);

  Homie_setFirmware("awesome-relay", firmwareVersion);


  ↪ lightNode.advertise("on").setName("On").setDatatype("boolean").settable(lightOnHandler);

  Homie.setup();
}

void loop() {
  Homie.loop();
}
```

Di seguito il file config.json

```
{
  "name": "light",
  "device_id": "light",
  "wifi": {
    "ssid": "Vodafone-******",
    "password": "************"
  },
  "mqtt": {
    "host": "192.168.1.5",
    "port": 1883,
    "base_topic": "homie/"
  },
  "ota": {
    "enabled": true
  },
  "settings": {
    "on": true
  }
}
```

A.2 Dispositivo Temperatura

Di seguito il codice caricato sul dispositivo di test

```
#include <Homie.h>

const int TEMPERATURE_INTERVAL = 300;

unsigned long lastTemperatureSent = 0;

HomieNode temperatureNode("temperature", "Temperature", "temperature");

void loopHandler() {
  if (millis() - lastTemperatureSent >= TEMPERATURE_INTERVAL * 1000UL ||
  ↪  lastTemperatureSent == 0) {
    float temperature = 22; // temperatura hardcoded per il test
    Homie.getLogger() << "Temperature: " << temperature << " °C" << endl;
    temperatureNode.setProperty("degrees").send(String(temperature));
    lastTemperatureSent = millis();
  }
}

void setup() {
  Serial.begin(115200);
  Serial << endl << endl;
  Homie_setFirmware("awesome-temperature", "1.0.0");
  Homie.setLoopFunction(loopHandler);

  temperatureNode.advertise("degrees").setName("Degrees")
                                      .setDatatype("float")
                                      .setUnit("ºC");

  Homie.setup();
}

void loop() {
```

```
32    Homie.loop();
33  }
```

Di seguito il file config.json

```
1   {
2     "name": "termostato",
3     "device_id": "temp",
4     "wifi": {
5       "ssid": "Vodafone-*******",
6       "password": "***********"
7     },
8     "mqtt": {
9       "host": "192.168.1.5",
10      "port": 1883,
11      "base_topic": "homie/"
12    },
13    "ota": {
14      "enabled": true
15    },
16    "settings": {
17      "degree": 22
18    }
19  }
```

Appendice B

Estratti ed esempi

B.1 Esempio di definizione degli Intent e delle Utterance in una Custom Skill

```
{
  "interactionModel": {
    "languageModel": {
      "intents": [
        {
          "name": "PlanMyTrip",
          "slots": [
            {
              "name": "travelDate",
              "type": "AMAZON.DATE",
              "samples": [
                "I am taking this trip on {travelDate}",
                "on {travelDate}",
                "{travelDate}"
              ]
            },
            {
              "name": "toCity",
              "type": "AMAZON.US_CITY",
              "samples": [
```

```
                "I'm going to {toCity}",
                "{toCity}"
              ]
            },
            {
              "name": "fromCity",
              "type": "AMAZON.US_CITY",
              "samples": [
                "{fromCity}",
                "I'm starting from {fromCity}"
              ]
            },
          ],
          "samples": [
            "{toCity}",
            "Voglio viaggiare da {fromCity} fino a {toCity} il {travelDate}"
          ]
        }
      ]
    }
  }
}
```

B.2 Export json del flusso NodeRed

```
[
    {
        "id": "471273d4.ec58ec",
        "type": "tab",
        "label": "Alexa",
        "disabled": false,
        "info": ""
    },
    {
        "id": "c7027159.de49f",
```

```
        "type": "comment",
        "z": "471273d4.ec58ec",
        "name": "Alexa-OpenHab",
        "info": "",
        "x": 120,
        "y": 60,
        "wires": []
    },
    {
        "id": "7c776476.cd589c",
        "type": "mqtt in",
        "z": "471273d4.ec58ec",
        "name": "",
        "topic": "Alexa/device",
        "qos": "1",
        "datatype": "json",
        "broker": "39a0cd5a.f8cd42",
        "x": 110,
        "y": 220,
        "wires": [
            [
                "25a820b3.7e833",
                "3b0d5e3d.927a12",
                "cfb272bd.78f6e"
            ]
        ]
    },
    {
        "id": "25a820b3.7e833",
        "type": "debug",
        "z": "471273d4.ec58ec",
        "name": "msg Alexa/device",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
```

```
        "complete": "payload",
        "targetType": "msg",
        "x": 250,
        "y": 340,
        "wires": []
    },
    {
        "id": "a9c85926.6349a8",
        "type": "openhab-v2-out",
        "z": "471273d4.ec58ec",
        "name": "ItemControl",
        "controller": "43493616.f736c8",
        "item": "ESP32",
        "topic": "ItemCommand",
        "topicType": "oh_cmd",
        "payload": "payload",
        "payloadType": "msg",
        "storeStateInFlow": true,
        "x": 630,
        "y": 140,
        "wires": []
    },
    {
        "id": "f96d5f60.44abd",
        "type": "openhab-v2-get",
        "z": "471273d4.ec58ec",
        "name": "State",
        "controller": "43493616.f736c8",
        "item": "ESP32",
        "x": 550,
        "y": 340,
        "wires": [
            [
                "301b1794.5123c8",
                "ce21cced.c58e7"
            ]
```

```
        ]
    },
    {
        "id": "3b0d5e3d.927a12",
        "type": "function",
        "z": "471273d4.ec58ec",
        "name": "Check value",
        "func": "if(msg.payload.operation==\"powerset\"){\n
        ↪ node.warn(msg.payload.value);\n
        ↪ if(msg.payload.value!='ON'&&msg.payload.value!=\"OFF\"){\n
        ↪ node.warn(\"Messaggio non valido\");\n        return null;\n    }\n
        ↪ msg.item = msg.payload.device\n    msg.payload=msg.payload.value\n
        ↪ node.warn(msg); \n}else{\n    msg = null;\n}\n\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 390,
        "y": 180,
        "wires": [
            [
                "a9c85926.6349a8"
            ]
        ]
    },
    {
        "id": "ce21cced.c58e7",
        "type": "debug",
        "z": "471273d4.ec58ec",
        "name": "State Esp",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "true",
        "targetType": "full",
        "x": 700,
        "y": 520,
```

```
        "wires": []
    },
    {
        "id": "cfb272bd.78f6e",
        "type": "function",
        "z": "471273d4.ec58ec",
        "name": "Check ReportState",
        "func": "if(msg.payload.operation==\"reportstate\"){\n    msg.item =
        ↪  msg.payload.device;\n}else{\n    msg = null;\n}\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 450,
        "y": 260,
        "wires": [
            [
                "f96d5f60.44abd"
            ]
        ]
    },
    {
        "id": "301b1794.5123c8",
        "type": "mqtt out",
        "z": "471273d4.ec58ec",
        "name": "",
        "topic": "Alexa/device/out",
        "qos": "2",
        "retain": "false",
        "broker": "39a0cd5a.f8cd42",
        "x": 760,
        "y": 380,
        "wires": []
    },
    {
        "id": "af35ba70.5e3978",
        "type": "openhab-v2-in",
        "z": "471273d4.ec58ec",
```

```
        "name": "",
        "controller": "43493616.f736c8",
        "item": "ESP32",
        "ohCompatibleTimestamp": false,
        "eventTypes": [
            "ItemStateChangedEvent"
        ],
        "outputAtStartup": true,
        "storeStateInFlow": false,
        "x": 130,
        "y": 460,
        "wires": [
            [
                "76a27e33.57a7f"
            ],
            []
        ]
    },
    {
        "id": "76a27e33.57a7f",
        "type": "function",
        "z": "471273d4.ec58ec",
        "name": "",
        "func": "var m = { \"payload\":{}};\nm.payload.item =
        ↪  msg.item;\nm.payload.state = msg.payload\nreturn m;",
        "outputs": 1,
        "noerr": 0,
        "x": 330,
        "y": 460,
        "wires": [
            [
                "ce21cced.c58e7",
                "301b1794.5123c8"
            ]
        ]
    },
```

```
    {
        "id": "39a0cd5a.f8cd42",
        "type": "mqtt-broker",
        "z": "",
        "name": "ArancinoMQTT",
        "broker": "localhost",
        "port": "1883",
        "clientid": "NodeRed",
        "usetls": false,
        "compatmode": true,
        "keepalive": "60",
        "cleansession": true,
        "birthTopic": "",
        "birthQos": "0",
        "birthPayload": "",
        "closeTopic": "",
        "closeQos": "0",
        "closePayload": "",
        "willTopic": "",
        "willQos": "0",
        "willPayload": ""
    },
    {
        "id": "43493616.f736c8",
        "type": "openhab-v2-controller",
        "z": "",
        "name": "OpenHab",
        "protocol": "http",
        "host": "localhost",
        "port": "8080",
        "path": "",
        "username": "admin",
        "password": "smartme",
        "allowRawEvents": true
    }
]
```

B.3 Richiesta JSON per l'accensione del dispositivo Luce

```
{
  "directive": {
    "header": {
      "namespace": "Alexa.PowerController",
      "name": "TurnOn",
      "payloadVersion": "3",
      "messageId": "asdasd",
      "correlationToken": "aa"
    },
    "endpoint": {
      "scope": {
        "type": "BearerToken",
        "token": "a"
      },
      "endpointId": "Luce",
      "cookie": {
        "propertyMap":
        ↪ "{\"PowerController\":{\"powerState\":{\"parameters\":{\"category\":\"LIGHT\"},\"item\":{
      }
    },
    "payload": {}
  }
}
```

B.4 Richiesta JSON dello stato del dispositivo Temperatura

```
{
    "directive": {
        "header": {
            "namespace": "Alexa",
```

```
            "name": "ReportState",
            "payloadVersion": "3",
            "messageId": "123",
            "correlationToken": "aaa"
        },
        "endpoint": {
            "scope": {
                "type": "BearerToken",
                "token": "bbb"
            },
            "endpointId": "Temperatura",
            "cookie": {
                "propertyMap":
                ↪  "{\"TemperatureSensor\":{\"temperature\":{\"parameters\":{},\"item\":{\"name\":\'
            }
        },
        "payload": {}
    }
}
```

Bibliografia

[1] *Amazon Alexa*. Nov. 2019. URL: https://it.wikipedia.org/wiki/Amazon_Alexa.

[2] Inc. Amazon.com. *Alexa documentation*. URL: https://developer.amazon.com/it/docs/ask-overviews/build-skills-with-the-alexa-skills-kit.html. (accessed: 01.12.2019).

[3] Bjoernkarmann. *bjoernkarmann/project_alias*. Gen. 2019. URL: https://github.com/bjoernkarmann/project_alias.

[4] D. Bruneo et al. «Building a Smart City Service Platform in Messina with the #SmartME Project». In: *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. Mag. 2018, pp. 343–348. DOI: 10.1109/WAINA.2018.00109.

[5] H. Chung et al. «"Alexa, Can I Trust You?"» In: *Computer* 50.9 (2017), pp. 100–104. ISSN: 1558-0814. DOI: 10.1109/MC.2017.3571053.

[6] *Documentation*. The Confederation, 2019. URL: https://developer.amazon.com/documentation.

[7] Eclipse. *eclipse/mosquitto*. Nov. 2019. URL: https://github.com/eclipse/mosquitto.

[8] *ESP32 Datasheet*. Espressif. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[9] M. Fazio et al. «Heterogeneous Sensors Become Homogeneous Things in Smart Cities». In: *2012 Sixth International Conference on Innovative Mobile and Internet*

Services in Ubiquitous Computing. Lug. 2012, pp. 775–780. DOI: 10.1109/IMIS.2012.136.

[10] Google. *App Actions — Google Developers*. URL: https://developers.google.com/assistant/app/overview.

[11] Prof. Ananda M. H. «Advanced IoT Based Integrated Home Appliances Control And Heart Beat Monitoring System.» In: *Compliance Engineering Journal* (2019).

[12] Homieiot. *homieiot/homie-esp8266*. URL: https://github.com/homieiot/homie-esp8266/tree/develop-v3.

[13] Marina Jonna. «Domotica: i 6 vantaggi che offre». In: *Panorama* (ago. 2013).

[14] *Le ultime parole famose*. URL: https://d.repubblica.it/argomenti/2011/06/07/news/ultime_parole_famose-365216/?refresh_ce.

[15] MDSLab. *MDSLab/stack4things*. URL: https://github.com/MDSLab/stack4things/blob/master/README.md.

[16] *MQTT*. Giu. 2019. URL: https://it.wikipedia.org/wiki/MQTT.

[17] Node-Red. *node-red/node-red*. URL: https://github.com/node-red/node-red.

[18] *NodeRed Documentation*. URL: https://nodered.org/docs/.

[19] OpenHAB. *OpenHab documentation*. URL: https://www.openhab.org/docs/. (accessed: 01.12.2019).

[20] Openhab. *openhab/openhab-alexa*. Dic. 2019. URL: https://github.com/openhab/openhab-alexa.

[21] Di Roberto e Roberto Moro visconti. «Internet delle cose, Networks e plusvalore della connettività». In: *Il diritto industriale* (gen. 2017), pp. 536–544.

[22] *The Homie convention*. URL: https://homieiot.github.io/.

[23] Domenico Trisciuoglio. *Introduzione alla domotica*. Tecniche nuove, 2009.

[24] Wikipedia. *Virtual assistant*. 2019. URL: https://en.wikipedia.org/wiki/Virtual_assistant. (accessed: 02.12.2019).

[25] Sam Williams. *Free as in Freedom. Richard Stallman's Crusade for Free Software.* O'Reilly Media, 2002. ISBN: 0-596-00287-4.