Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

SensorSpeak: monitoraggio e controllo vocale di reti di sensori mediante Amazon Alexa

Relatore:
Chiar.mo Prof.
Marco Di Felice

Presentata da:
Angelo De Lisa

Sessione II
Anno Accademico 2016/2017

Ai miei genitori ...

Introduzione

La possibilità di estendere l'accesso ad Internet agli *"oggetti"* ha dato il via all'era dell'*Internet of Things* (IoT). Col termine *"things"* (cose, oggetti) si sottolinea, appunto, che non si fa riferimento solamente a computer, smartphone o persone, ma soprattutto a sensori, elettrodomestici, veicoli, indumenti e altri oggetti di uso quotidiano. L'IoT rappresenta, dunque, un concetto generale in cui gli oggetti connessi ad Internet possono collezionare dati dall'ambiente esterno e poi scambiarli, processarli o utilizzarli per uno scopo specifico. La presenza pervasiva di tali oggetti intorno a noi ha permesso lo sviluppo dell'IoT in numerosi settori: domotica, trasporti, medicina, industria. Con il proliferare di nuove applicazioni e tecnologie nell'IoT, si introducono anche innovativi modi di collaborare e interagire, sia dal punto di vista degli oggetti che dell'uomo. L'integrazione della voce nei sistemi IoT offre un metodo versatile ed intuitivo per l'uomo in termini di interazione, comunicazione e controllo dei dispositivi. Infatti, la voce è la maniera naturale dell'uomo di interagire e di esprimersi ed è comprensibile il desiderio di voler estendere questo efficace metodo di comunicazione nell'interazione con le macchine. La possibilità di includere il riconoscimento vocale nei vari oggetti migliora l'usabilità e fornisce nuove capacità e modi d'uso ad una moltitudine di device connessi nei sistemi IoT.

Lo scopo di questa tesi è la realizzazione di **SensorSpeak**, un sistema di monitoraggio e controllo dei dispositivi IoT mediante l'utilizzo di comandi vocali. Attraverso la semplice interazione vocale con Alexa (lo smart personal assistant di Amazon), l'utente è in grado di monitorare e controllare i

dispositivi associati al proprio cloud. SensorSpeak, inoltre, utilizzando esclusivamente l'interazione vocale, permette di agire contemporaneamente su più device, aggregando e processando i dati provenienti da molteplici sensori oppure eseguendo specifici comandi. L'approccio utilizzato nell'implementazione del sistema estende il suo utilizzo ad un'enorme varietà di dispositivi e la sua integrazione in diversi scenari, come smart home o reti di sensori. Un aspetto importante in contesti del genere è rappresentato dalla questione energetica dei dispositivi, spesso alimentati da batterie. A tal fine, SensorSpeak implementa una funzione di predizione del tempo di carica rimanente della batteria, sfruttando un modello che descrive l'andamento del processo di scarica della batteria in questione.

SensorSpeak è stato installato e testato in una rete *6LoWPAN*, formata da una serie di *STM32 Nucleo Board*, all'interno del centro di ricerca ARCES di Bologna.

Il documento viene organizzato secondo la struttura seguente. La prima parte offre una panoramica riguardo il paradigma dell'IoT, descrivendo, inoltre, la tecnologia del controllo vocale e la sua integrazione nei sistemi dell'Internet of Things. La seconda parte illustra, in dettaglio, le fasi di progettazione e implementazione del sistema in ogni sua componente. Nell'ultima parte, infine, viene fatta una valutazione dei risultati ottenuti dall'applicazione dell'algoritmo di predizione ad uno specifico modello di batteria.

Indice

Elenco delle figure

Elenco delle tabelle

Parte I

Stato dell'arte

Capitolo 1

Internet of Things

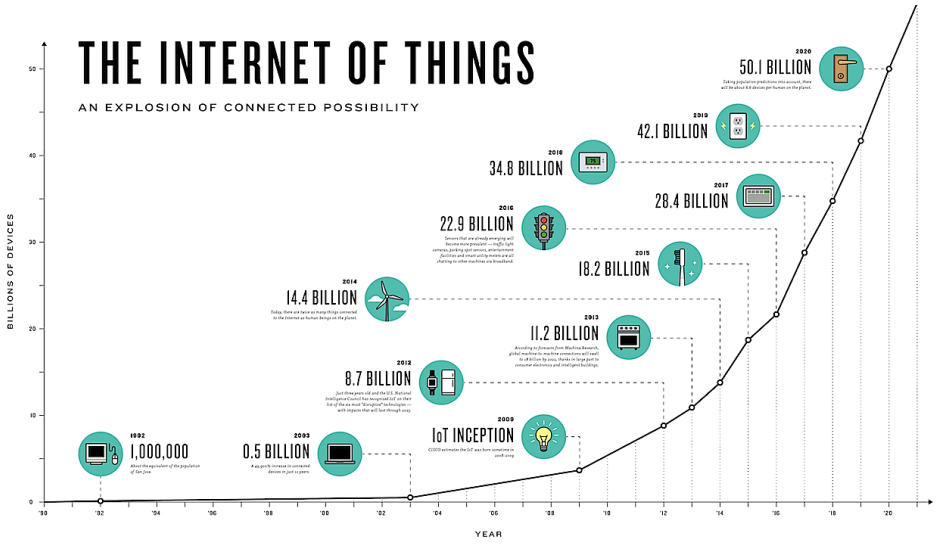

Figura 1.1: Previsione del numero di dispositivi collegati fino al 2020

Il numero di dispositivi connessi ad Internet ha ormai superato il numero di abitanti della Terra e si stima che, entro il 2020, tali device raggiungeranno quota 50 miliardi [1]. In un contesto del genere, l'*Internet of Things* sta rapidamente acquistando importanza e utilità nella vita di tutti i giorni. L'idea di base dell'IoT è la presenza pervasiva, intorno a noi, di *oggetti* in grado di interagire e di cooperare tra loro per eseguire un determinato task.

In questo capitolo, vengono descritti gli aspetti fondamentali di questo nuovo rivoluzionario paradigma.

1.1 Definizione

Il termine *Internet of Things* è stato usato per la prima volta dall'inglese Kevin Ashton nel 1999 [2], riferendosi all'utilizzo dei tag RFID (Radio Frequency IDentification) per tracciare dei prodotti in inventario. Solo nel 2005, però, l'espressione *Internet of Things* diventa ufficiale, quando la *International Telecommunication Union* pubblica il suo primo report sull'argomento. In [3], si sottolinea come il significato della parola *"Thing"* non sia limitato solamente ai tag RFID, ma comprenda un elevato numero di oggetti diversi, come sensori, attuatori o qualsiasi dispositivo in grado di interagire con il mondo esterno e di accedere ad Internet. Da allora, numerose e differenti definizioni di IoT sono state proposte.

In [4], l'Institute of Electrical and Electronic Engineers (IEEE) definisce l'IoT come: *Una rete di oggetti - ognuno dotato di sensori - collegati ad Internet.*

Una definizione più dettagliata è fornita in [5], in cui gli oggetti (*"Things"*) sono visti come partecipanti attivi nei processi informativi e sociali grazie alla capacità di interagire e comunicare tra loro e con l'ambiente esterno, scambiando dati ed informazioni raccolti dall'ambiente circostante. Inoltre, sono in grado di reagire agli eventi del mondo reale e di influenzarlo eseguendo azioni o creando servizi, senza necessariamente richiedere l'intervento dell'uomo. Dunque, consentendo nuove forme di comunicazione tra persone e oggetti, e tra gli oggetti stessi, l'IoT aggiunge una nuova dimensione al mondo dell'informazione e della comunicazione, così com'era stato per Internet.

Nello studio condotto in [6], gli autori individuano tre diversi paradigmi dell' IoT:

1. *Things oriented:* l'attenzione viene posta sugli "oggetti" e sulla ricerca di un paradigma per identificarli ed integrarli.

2. *Internet oriented:* la ricerca è volta alla creazione di una connessione efficiente tra i dispositivi tramite l'utilizzo di una versione semplificata del protocollo IP.

3. *Semantic oriented:* lo scopo è quello di rappresentare, memorizzare, estrapolare e gestire l'enorme quantità di informazioni fornita dai dispositivi IoT.

Un efficiente sviluppo dell'Internet of Things, quindi, è rappresentato dalla convergenza delle tre diverse visioni in un unico paradigma.

È possibile identificare una serie di requisiti che un sistema IoT deve supportare [7]:

- **Eterogeneità:** l'IoT è caratterizzata da una forte eterogeneità in termini di varietà di device, tecnologie, servizi e campi applicativi.

- **Scalabilità:** il crescente numero di dispositivi collegati ad un'infrastruttura globale porta il problema della scalabilità su diversi livelli, cioè: livello architetturale, livello di identificazione/indirizzamento degli oggetti, livello di comunicazione e livello di mapping oggetti/servizi.

- **Autonomia:** data la complessità e la varietà degli scenari possibili nell'IoT, gli oggetti devono possedere capacità di adattamento, di autoconfigurazione, elaborazione indipendente dei dati e reazione autonoma in base agli stimoli, al fine di minimizzare l'intervento umano.

- **Minimizzazione dei costi:** ottimizzare i costi del mantenimento del sistema IoT con una particolare attenzione al consumo energetico.

- **Qualità del servizio:** la garanzia di un'appropriata qualità del servizio è necessaria per servizi e applicazioni caratterizzati da un traffico dati real-time.

- **Sicurezza:** l'IoT deve garantire un ambiente sicuro in termini di sicurezza delle comunicazioni, autenticazione, integrità dei dati e privacy degli utenti.

1.2 Applicazioni dell'IoT

Le enormi potenzialità offerte dall'IoT hanno reso possibile lo sviluppo di applicazioni innovative in numerosi campi. Tali applicazioni hanno avuto un forte impatto sulla vita di tutti i giorni sotto molteplici punti di vista: a casa, mentre si viaggia, durante il lavoro [6].

Esempi di domini applicativi nei quali l'IoT ha portato importanti cambiamenti sono i seguenti:

- Smart Home
- Sanità
- Business e commercio
- Monitoraggio ambientale
- Trasporti

1.2.1 Smart Home

La miriade di smart device e sensori presenti in casa permette la creazione di una vasta gamma di applicazioni: spegnere tutti i dispositivi elettrici quando non sono utilizzati, impostare il riscaldamento a seconda delle nostre preferenze o del meteo, evitare incidenti domestici grazie all'uso di sistemi di allarme, e così via. Ci si immagina di poter abitare in una casa "intelligente", in cui ogni dispositivo o elettrodomestico si gestisce autonomamente e riesce a comunicare con altri componenti: l'esempio più comune è quello del frigorifero il quale si accorge che il latte è finito e comunica al supermercato di portarne una nuova confezione.

1.2.2 Sanità

Molti benefici derivanti dall'Internet of Things possono essere apportati al campo medico e sanitario. Infatti, tecnologie e sensori avanzati permettono

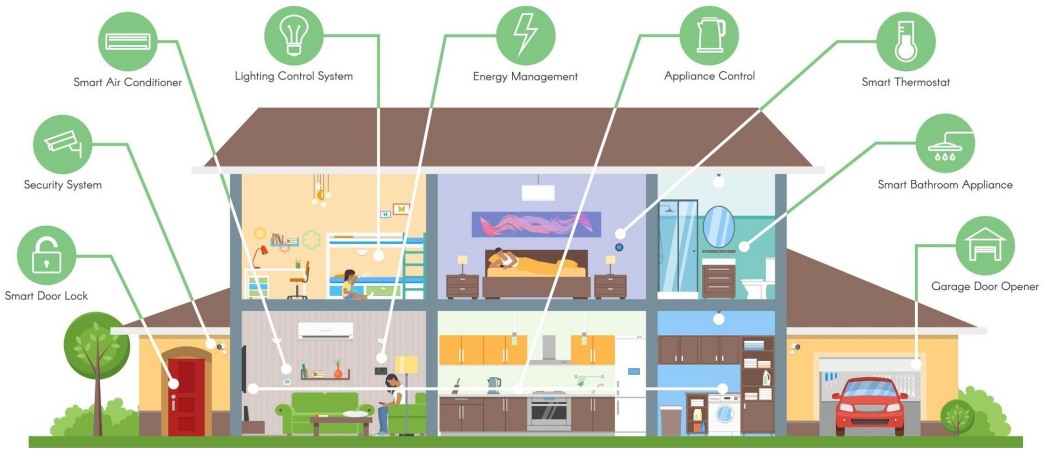

Figura 1.2: Smart Home [29]

un monitoraggio in tempo reale di parametri medici e funzioni vitali (battito cardiaco, pressione del sangue). I dati raccolti possono essere trasmessi al personale medico per effettuare immediate diagnosi e controlli sulla salute del paziente. Inoltre, monitorando specifici segnali fisiologici, i device possono essere in grado di attivare allarmi (ad esempio, in seguito alla caduta di una persona anziana), suggerire possibili visite mediche, oppure diagnosticare i primi segni di una particolare malattia (es. Parkinson) [7].

1.2.3 Business e commercio

Il modo in cui avviene lo shopping potrebbe cambiare radicalmente con l'utilizzo delle tecnologie dell'IoT. Infatti, se da un lato ci saranno dispositivi in grado di provvedere autonomamente all'acquisto dei prodotti (il frigorifero che compra il latte ne è un esempio), dall'altro, la continua raccolta di informazioni riguardo abitudini e bisogni dei clienti permetterà alle aziende di fornire servizi e prodotti sempre più personalizzati, con una conseguente probabilità di vendita maggiore. Inoltre, attraverso "scaffali intelligenti" e prodotti muniti di tag RFID, si può ridurre lo spreco di risorse: analizzando i dati forniti dai device, si inferisce qual è la reale domanda di un particolare

prodotto ed evitare sovrapproduzioni [7].

1.2.4 Monitoraggio ambientale

Combinare le informazioni ottenute dai sensori consentirà la previsione e la prevenzione delle catastrofi. L'accidentale emissione di acqua contaminata, ad esempio, potrebbe essere rilevata da un sensore che, agendo su una valvola, arresta la fuoriuscita del liquido [8]. Altri esempi di utilizzo delle tecnologie dell'IoT nel campo ambientale riguardano il monitoraggio del livello di inquinamento dell'aria, il rilevamento di incendi o l'analisi dell'attività dei vulcani.

1.2.5 Trasporti

I sensori e gli attuatori presenti lungo le strade e i binari, così come quelli disponibili su auto, treni ed autobus, possono fornire importanti informazioni ai conducenti dei veicoli al fine di garantire una migliore navigazione ed una maggiore sicurezza. Tipici esempi sono i sensori di "collision avoidance" e i sistemi di monitoraggio del trasporto di materiali pericolosi. Inoltre, l'uso delle informazioni raccolte dai sensori lungo le strade consente di pianificare viaggi aggirando il traffico e gli incidenti [6].

1.3 Sicurezza e Privacy

Data l'enorme mole di informazioni sensibili contenute nei dati raccolti e condivisi dai device, risulta necessario garantire un livello di sicurezza appropriato quando si utilizza un sistema IoT. I maggiori problemi relativi alla sicurezza riguardano l'autenticazione e l'autorizzazione [9]. L'autenticazione è di solito ottenuta tramite diversi metodi come l'utilizzo di password e ID o chiavi segrete pre-condivise. L'autorizzazione garantisce, invece, che entità estranee al sistema non riescano ad accedere e modificare i dati e ciò può essere ottenuto tramite controllo dell'accesso. A causa della eterogeneità e

della complessità degli oggetti nei sistemi IoT, le tradizioniali soluzioni di autenticazione e autorizzazione non sono applicabili. Numerose soluzioni che adoperano algoritmi di crittografia sono state proposte in letteratura (es. [10, 11]). Tuttavia, a causa della grande quantità di risorse richieste da tali algoritmi (in termini di energia e costo computazionale), le soluzioni proposte non possono essere applicate efficacemente ai sistemi IoT, costituiti prevalentemente da device dalle ridotte capacità.

Altro aspetto molto discusso in seguito alla diffusione dell'IoT è quello della privacy. Nell'ambito dell'IoT, il problema della privacy riguarda principalmente la raccolta dei dati (quali dati vengono raccolti, da chi e quando), l'uso dei dati raccolti (quali servizi sono autorizzati al trattamento di tali dati) e, infine, il salvataggio dei dati (i dati raccolti dovrebbero essere conservati solo finché è strettamente necessario) [12]. Attualmente, la questione della privacy nell'IoT è trattata solo parzialmente. Per questo motivo, la definizione di un modello generale di privacy e lo sviluppo di soluzioni che riescano a trovare un compromesso tra anonimato e operazioni di tracking, potrebbe portare ad una maggiore sicurezza per gli utenti, senza trascurare l'efficienza dei servizi forniti [13].

Capitolo 2

Controllo vocale

Data l'enorme quantità di device da gestire, sono necessari metodi di interazione uomo-macchina sempre più intuitivi ed efficaci: i progressi nel campo del riconoscimento vocale hanno permesso l'uso dei comandi vocali per l'interazione e la gestione dei device. Integrando tale tecnologia con algoritmi di intelligenza artificiale sempre più sofisticati, invece, ha portato alla nascita di assistenti personali intelligenti in grado di assistere l'uomo in molteplici contesti. Pertanto, il capitolo fornisce una panoramica dell'applicazione del controllo vocale ai sistemi IoT e una descrizione delle caratteristiche degli assistenti virtuali.

2.1 Panoramica

L'abilità di comunicare in maniera naturale con una macchina al fine di chiedere informazioni o di eseguire comandi è migliorata drasticamente negli ultimi decenni. Sebbene tale tecnologia non abbia raggiunto ancora la perfezione, l'uomo riesce ad interagire in modo ragionevole attraverso la voce con un'ampia varietà di device. Una caratteristica fondamentale dello sviluppo di tale tecnologia risiede nell'utilizzo del cloud: analizzando i dati attraverso enormi capacità computazionali, i servizi di riconoscimento vocale possono

migliorare nel tempo, grazie all'adattamento ai modelli di comunicazione e alla comprensione del linguaggio in base al contesto [14].

I vantaggi derivanti dall'utilizzo dell'interazione vocale uomo-macchina sono molteplici [23]:

- Il dialogo è il modo naturale di comunicare per l'uomo: risulta più intuitivo e semplice comunicare i comandi verbalmente.

- L'uso dei comandi vocali diventa fondamentale quando si eseguono attività che impegnano mani e occhi. Ad esempio, durante la guida, è pericoloso (e vietato dalla legge) distogliere l'attenzione per rispondere al cellulare o manovrare altri dispositivi elettronici. L'uso di comandi vocali per il compimento di task simili può evitare al conducente del veicolo di fare un incidente o di incorrere in sanzioni.

- La possibilità di utilizzare vari dispositivi tramite voce è fondamentale per persone con disabilità fisiche. Ciò permette un'autonomia e una qualità della vita migliore per i disabili.

- Risparmio economico: l'integrazione vocale potrebbe rendere svantaggioso l'utilizzo di un schermo su molti dispositivi, in quanto riduce i costi per la costruzione di quei device che non vengono utilizzati per la maggior parte del tempo. Ad esempio, potrebbe risultare più vantaggioso economicamente offrire la connessione ad un unico servizio vocale remoto, piuttosto che installare un touch screen per tutti i dispositivi che implementano la stessa funzione.

- Gli operatori vocali virtuali (es. quelli telefonici) sono un efficiente mezzo di comunicazione bidirezionale con le macchine che possono ascoltare e rispondere senza la necessità di comandi complessi.

2.2 Smart personal assistant

L'uso della voce continua a giocare un ruolo vitale per molte aziende in termini di interazione dei clienti con i propri device. Infatti, alcune importanti compagnie hanno sviluppato degli "*smart personal assistant*": software in grado di eseguire specifiche attività e di fornire informazioni o servizi interfacciandosi con un utente principalmente attraverso interazione vocale [23] (es. Alexa [18], Google Assistant [19], Siri [20], Cortana [22]). Le funzionalità fornite sono innumerevoli: dall'informazione all'intrattenimento, dalla gestione delle attività personali al controllo dei propri dispositivi, dallo shopping all'automazione della casa. Migliorando e aumentando sempre di più le capacità, tali assistenti stanno diventando essenziali nella vita di tutti i giorni. Inoltre, comunicando in maniera naturale con gli umani, essi smettono di essere visti come computer e appaiono come dei partner [24].

Di seguito, viene descritta l'architettura degli smart personal assistant e, successivamente, viene data una panoramica delle caratteristiche dei vari assistenti personali, ponendo maggiore enfasi nella descrizione di *Alexa*, essendo lo smart personal assistant utilizzato nel progetto di questa tesi.

2.2.1 Architettura

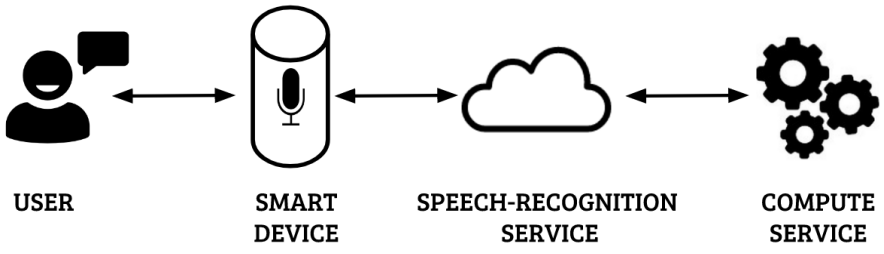

Figura 2.1: Architettura smart personal assistant

Tipicamente, il funzionamento di uno smart personal assistant è costituito dall'interazione di una serie di componenti (Figura 2.1):

- **Utente:** interagisce vocalmente con il device, di solito attivandolo con una parola chiave.
- **Smart device:** munito di microfono e connessione ad Internet, registra i comandi dell'utente in formato audio e li invia al servizio di riconoscimento vocale. Inoltre, è in grado di comunicare l'esito della richiesta in maniera vocale.
- **Speech-recognition service:** converte l'audio in testo, interpreta la richiesta dell'utente e invia il testo in un formato processabile "*machine-readable*" (es. JSON).
- **Compute service:** implementa la logica computazionale del servizio richiesto. Esso identifica quale funzione deve essere eseguita ed, eventualmente, interagisce con un web service esterno per soddisfare la richiesta dell'utente.

2.2.2 Alexa

Alexa è lo smart personal assistant sviluppato da Amazon, il quale consente agli utenti, tramite comandi vocali, di ottenere servizi o di interagire con svariati dispositivi. Alcuni esempi di funzionalità (*"skill"*) fornite da Alexa sono l'impostazione di una sveglia o un timer, la riproduzione della propria playlist musicale, ordinare una pizza, ottenere informazioni circa il traffico o il meteo, oppure gestire i dispositivi della smart home. Essendo un servizio vocale implementato nel cloud, Alexa diventa sempre più "smart" grazie all'adattamento alle preferenze personali degli utenti, all'arricchimento del proprio vocabolario e al riconoscimento di nuovi pattern del discorso [18]. Amazon, inoltre, attraverso l'*"Alexa Skill Kit"*, dà la possibilità agli sviluppatori di realizzare nuove skill, in modo tale da migliorare ed arricchire l'esperienza personale degli utenti.

Il servizio vocale di Amazon è fruibile mediante vari dispositivi, quali *Amazon Echo*, *Amazon Echo Dot* e *Amazon Show*, semplicemente pronunciando la parola chiave "*Alexa*". Tutto il codice è eseguito nel cloud, quindi

Figura 2.2: Esempi di comandi e richieste ad Alexa

sui dispositivi non è installato nulla: è necessario solamente un collegamento ad Internet per poter accedere al servizio. Tutti i device in commercio, perciò, utilizzano la stessa versione di Alexa ma, mentre i dispositivi Echo ed Echo Dot consentono un'interazione solamente vocale con l'utente, il dispositivo Amazon Show fornisce anche un supporto visivo e, di conseguenza, la possibilità di mostrare immagini e video. Tuttavia, tramite il framework *Alexa Voice Service* [25], è possibile trasformare qualsiasi device provvisto di microfono, speaker e connessione ad Internet, in un prodotto in grado di utilizzare a pieno le potenzialità di Alexa, come un normale device Echo.

Per la prima configurazione del dispositivo associato ad Alexa, è possibile utilizzare un'app disponibile sia in ambiente iOS che Android. L'applicazione può essere usata anche per installare nuove skill, controllare i device della smart home, gestire liste e timer. Essa consente, inoltre, di visualizzare immagini o il testo interpretato dal servizio vocale e di inviare un feedback ad Amazon circa la qualità del riconoscimento.

Attualmente, il servizio vocale Alexa è disponibile solamente negli USA, nel Regno Unito e in Germania permettendo, perciò, un'interazione solamente in lingua inglese o tedesca.

Alexa Skills

Alexa fornisce una serie di funzionalità già integrate nel servizio, le cosiddette *"skills"*, per esempio: riproduzione musicale, informazioni sul meteo, ricerca di un contenuto su Wikipedia, e così via. Il set di skill predefinite può essere arricchito grazie all' *Alexa Skill Kit* (ASK): una collezione di API, tool, documentazione ed esempi di codice che consente agli sviluppatori una facile implementazione di nuove skill da aggiungere ad Alexa [26]. A seconda del tipo di funzionalità che si vuole implementare, bisogna scegliere la tipologia di skill appropriata. ASK supporta la creazione di quattro diversi tipi:

1. **Smart Home Skill:** permette all'utente di controllare e gestire i device della smart home, come luci, termostati, porte. Il modello d'interazione non è gestito dallo sviluppatore ma dalle *Smart Home Skill API*. Gli intent, perciò, devono necessariamente essere processati da una AWS Lambda *Function*.

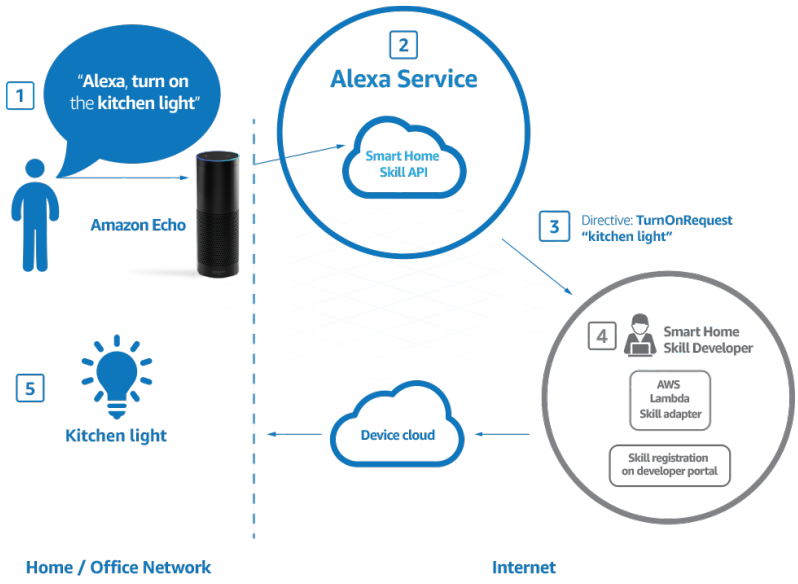

Figura 2.3: Diagramma di flusso per Smart Home Skill [28]

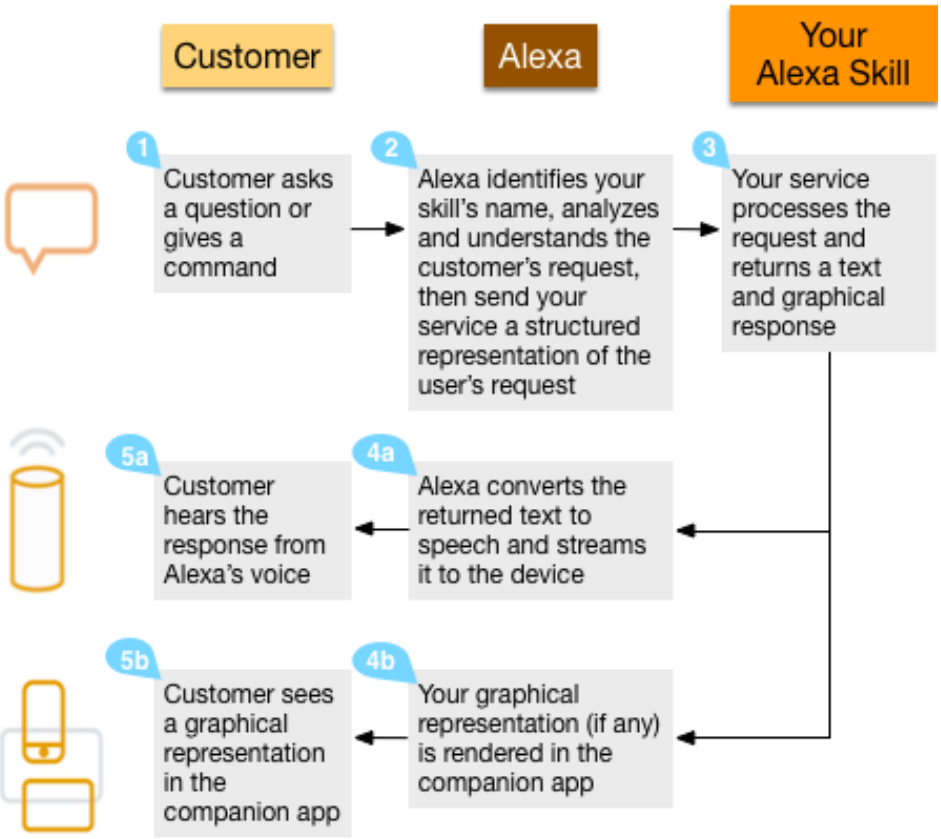

Figura 2.4: Diagramma di flusso per Custom Skill [27]

2. **Video Skill:** consente il controllo dei dispositivi video e dei servizi di streaming. L'implementazione di tale tipologia di skill è molto simile a quella Smart Home, con l'unica differenza che il modello d'interazione è definito dalle *Video Skill API*.

3. **Custom Skill:** può gestire qualsiasi tipo di richiesta, semplice o complessa. Ad esempio, effettuare un acquisto su un sito web, leggere le e-mail, controllare i propri smart device, e così via. Per l'implementazione, è necessario definire un modello d'interazione, cioè bisogna specificare quali sono le richieste che la skill può gestire (*intent*) e le

espressioni (*utterance*) che l'utente deve usare per invocare tali richieste. Un *cloud-based service*, invece, deve processare tali intent e restituire una risposta. In questo caso, il cloud-based service può essere sia un web service implementato ad-hoc per la skill, sia una funzione implementata su *Amazon Web Service (AWS) Lambda*.

4. **Flash Briefing Skill:** fornisce contenuti informativi appropriati a seconda delle preferenze dell'utente. A differenza della precedente tipologia, le interazioni vocali con l'utente sono gestite dalle *Content Skill API*. Lo sviluppatore, dunque, deve solo codificare le informazioni richieste in formato audio o in formato di testo per essere poi riprodotte da Alexa.

Per la creazione di una nuova skill che possa essere utilizzata tramite Alexa, è necessario registrarla e configurarla sull'*Amazon Developer Portal*. La configurazione di una skill prevede il completamento di alcuni step all'interno del portale:

- **Informazioni sulla skill:** definisce le informazioni di base come il nome e la tipologia di skill.
- **Modello di interazione:** configura gli intent e le utterance che possono essere usate per interagire con Alexa.
- **Configurazione:** definisce l'endpoint al quale Alexa deve collegarsi per processare le richieste dell'utente. L'endpoint può essere l'URL del web service oppure quello della Lambda Function.
- **Test:** permette di testare la skill visualizzando o ascoltando la risposta fornita da Alexa in base ai comandi ricevuti.
- **Privacy:** contiene opzioni relative alla privacy dell'utente.
- **Informazioni d'uso:** descrive come utilizzare la skill attraverso frasi esempio e descrizione delle funzionalità.

Una volta terminata tale procedura, la skill viene sottomessa ad Amazon che valuterà se risulta conforme ai "requisiti di certificazione". In caso di esito positivo, la skill diventa disponibile al download tramite l'Alexa App. Altrimenti, la skill sarà disponibile solo per l'account attraverso il quale è stata sviluppata.

Vantaggi

- È fortemente personalizzabile grazie alla possibilità di installare skill addizionali.
- Le applicazioni sviluppate possono essere eseguite su qualsiasi piattaforma e si ha il totale controllo delle modalità d'interazione.
- Ottima integrazione sia con servizi di terze parti che con dispositivi IoT e smart home.

Svantaggi

- Utilizzabile su un numero limitato di device.
- Feedback visivo limitato.
- Utilizzabile solo in lingua inglese e tedesca.

2.2.3 Google Assistant

Diversamente dagli altri assistenti, *Google Assistant*, l'assistente personale virtuale di Google, non presenta un nome femminile. Fu annunciato nel maggio del 2016 come parte dell'app di messaggistica *Allo* e dello speaker attivabile tramite voce *Google Home*. Esso consente di effettuare ricerche su Internet, schedulare eventi e timer, mostrare informazioni riguardo l'account Google dell'utente, gestire i dispositivi smart della casa, il tutto tramite comandi vocali. Inoltre, consente di ricevere i feedback riguardo le richieste sia in formato audio che visivo. Assistant è disponibile su numerosi dispositivi

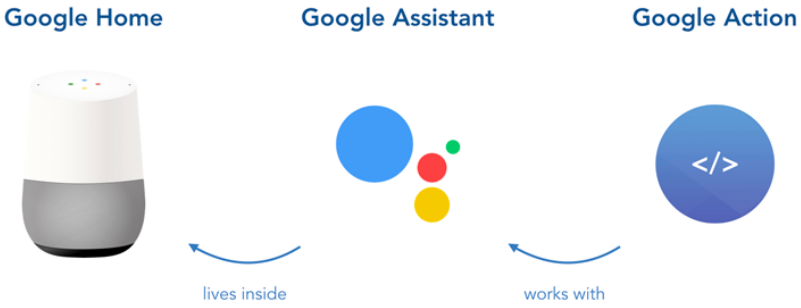

Figura 2.5: Architettura Google Assistant

Android, come smartphone, smartwatch, Tv. Inoltre, attraverso un software development kit, compagnie esterne a Google hanno la possibilità di implementare l'assistente vocale sui propri dispositivi, come apparecchi domestici o anche auto. Nel dicembre del 2016, Google ha lanciato la piattaforma per sviluppatori *"Actions on Google"*, tramite la quale migliora l'user experience di Assistant consentendo agli sviluppatori di integrare servizi esterni all'assitente virtuale. Per l'implementazione delle applicazioni, Google sfrutta la piattaforma *API.AI*. Essa permette di creare *"action"*, cioè gli step che l'applicazione deve eseguire quando una specifica richiesta è stata effettuata dall'utente. Ogni action è costituita da due componenti:

1. **Intent:** rappresenta un mapping tra ciò che dice l'utente e quale funzione deve essere invocata dal software.

2. **Fulfillment:** è un web service che processa la richiesta e fornise un'interfaccia utente.

Quando un utente richiede un'action tramite Assistant, l'interazione è divisa nei seguenti step:

1. Assistant chiede alla piattaforma Actions on Google di invocare l'applicazione appropriata per gestire la richiesta.

2. La piattaforma invia una richiesta al fulfillment dell'applicazione e riceve una risposta che passa ad Assistant.

3. Assistant mostra la risposta all'utente tramite la sua interfaccia. Inizia, quindi, la conversazione tra l'utente e l'applicazione.

4. Assistant invia l'input dell'utente all'applicazione e l'applicazione risponde direttamente ad Assistant. Questa conversazione continua finchè l'applicazione non ha raccolto tutti i dati necessari al soddisfacimento della richiesta.

Vantaggi

- Consente un'interazione audio-visiva con l'utente.
- Permette l'aggiunta di nuove funzionalità.
- Integrazione con i numerosi servizi Google.

Svantaggi

- Limitata integrazione con dispositivi della smart home.
- Numero limitato di applicazioni di terze parti preinstallate.
- Preoccupazioni riguardo la privacy dell'utente per l'accesso alle informazioni del proprio account Google.

2.2.4 Siri

Siri, uno dei primi assistenti personali sviluppati, fu inizialmente rilasciato come un'app per *iOS* nel 2010. Successivamente, venne acquistato da Apple diventando parte integrante dei prodotti di fabbrica, come *iPhone*, *Mac*, *Apple TV*. Siri supporta numerosi comandi impartiti dall'utente tramite voce come effettuare chiamate o inviare messaggi, fare ricerche in Internet

Figura 2.6: Logo di Siri

e, inoltre, è in grado di comunicare con applicazioni iOS. Tuttavia, solo in seguito al rilascio di "iOS 10" nel 2016, Apple ha permesso l'interazione tra Siri e servizi di terze parti. Gli sviluppatori, attraverso *"SiriKit"*, possono integrare Siri in applicazioni riguardanti i seguenti domini applicativi: chiamate e messaggi; pagamenti; prenotazione di ristoranti o viaggi; gestione delle componenti elettroniche dell'auto; allenamento e fitness; ricerca immagini; e gestione calendari [21]. L'applicazione sviluppata comprenderà la definizione dei seguenti step:

1. **Speech:** rappresenta il comando invocato dall'utente tramite voce; gli sviluppatori possono definire nuovi termini per aiutare Siri nell'interpretazione del comando.

2. **Intent:** interpreta il comando vocale e lo associa a qualche azione che l'applicazione può eseguire.

3. **Action:** si tratta dell'esecuzione dell'azione appropriata in base al comando invocato dall'utente.

4. **Response:** consente all'utente di confermare l'esecuzione dell'azione e rappresenta il feedback restituito all'utente in seguito alla propria richiesta.

Siri è disponibile in 36 lingue, tra cui inglese, italiano, cinese e giapponese.

Vantaggi

- Integrazione all'interno delle app iOS.
- L'installazione sui cellulari iOS ne consente l'utilizzo ovunque.
- Supporto di numerose lingue.

Svantaggi

- Integrazione limitata di servizi di terze parti.
- Non può gestire dispositivi delle smart home o device IoT in generale.
- Uso esclusivo riservato agli utenti iOS.

2.2.5 Cortana

Figura 2.7: Logo di Cortana

Cortana è un assistente personale intelligente creato da Microsoft in grado di effettuare il riconoscimento vocale dell'utente e rispondere alle sue domande usando il motore di ricerca *Bing*. Nel 2015, è stato incluso come parte integrante del sistema operativo *Windows 10*, sia per la versione desktop che mobile, ed è disponibile anche per sistemi iOS e Android. Attualmente, Cortana è utilizzabile in lingua inglese, portoghese, francese, tedesca, italiana, spagnola, cinese e giapponese.

Durante la conferenza *Microsoft Build 2017*, è stato presentato l'*Harman Kardon Invoke*, uno smart speaker con a bordo Cortana che rappresenta una vera e propria alternativa ad Amazon Echo e Google Home. Attraverso l'Invoke, Cortana potrà riprodurre musica, gestire calendari e attività, controllare il traffico e, grazie all'integrazione di Skype, effettuare chiamate. Inoltre, consentirà di controllare vocalmente i device della smart home come spegnere le luci o controllare la temperatura. L'obiettivo è quello di creare un assistente digitale che sia disponibile attraverso tutti i dispositivi dell'utente fornendo una personalizzazione sempre maggiore dei servizi e dei contenuti.

Contestualmente alla presentazione dello smart speaker, Microsoft ha anche annunciato che Cortana potrà essere arricchita di nuove funzionalità grazie a skill di terze parti. Attraverso il *Cortana Skills Kit*, gli sviluppatori possono facilmente creare esperienze personalizzate per Cortana integrando servizi esterni. Il *Cortana Skills Kit*, quindi, è una suite di strumenti che permette di creare nuove funzionalità e contenuti personalizzati accedendo alla conoscenza che Cortana possiede riguardo gli utenti, le loro preferenze e le loro esperienze. Attualmente, il Kit è in anteprima pubblica e l'unico modo per costruire una skill è quello di creare un bot che utilizza *Microsoft Bot Framework* e, successivamente, aggiungere il supporto per il riconoscimento vocale. Le caratteristiche chiave delle skill di Cortana sono:

- **Comprensione del linguaggio naturale:** grazie ad un modello di interazione vocale, le richieste dell'utente vengono filtrate ed interpretate da algoritmi di machine learning per fornire risposte personalizzate.

- **Realizzazione di conversazioni audio-visive:** oltre alla conversazione vocale, Cortana può fornire immagini, testi o altri contenuti visivi di supporto per aumentare il coinvolgimento dell'utente.

- **Esperienze personalizzate per gli utenti:** attraverso dei permessi, Cortana può accedere al profilo e alle informazioni personali dell'utente durante l'esecuzione della skill.

- **Supporto dell'intelligenza artificiale:** sfruttando i *Microsoft Cognitive Services*, si ha la possibilità di accedere ad una collezione di algoritmi di intelligenza artificiale per il linguaggio, il supporto visivo e la conoscenza.

Vantaggi

- Fornisce una user experience fortemente personalizzata.
- Essendo disponibile su numerose piattaforme (Windows 10, iOS e Android), è utilizzabile su un'enorme varietà di dispositivi.
- Consente di invocare comandi sia in forma vocale che scritta.

Svantaggi

- Integrazione di applicazioni di terze parti limitata e in fase di sperimentazione.
- Preoccupazioni riguardo la privacy dell'utente data la costante memorizzazione di dati personali (se non disattivata).
- Controllo dei dispositivi della smart home disponibile in futuro.

2.3 Controllo vocale nell'IoT

Grazie alla possibilità di fornire un'esperienza utente più flessibile e intuitiva, è inevitabile che l'integrazione del controllo vocale nei sistemi IoT pervada molteplici e diversi settori, spaziando da applicazioni per i consumatori, fino all'industria e al commercio. Nel settore dei consumatori, la voce diventerà particolarmente pervasiva nell'assistenza sanitaria, nei dispositivi wearable, Smart Home e elettronica di consumo. Nel settore commerciale, invece, esistono numerose soluzioni vocali per la gestione dell'inventario dei prodotti, il monitoraggio della salute, la logistica e la consegna (UPS/FedEx).

Infine, la tecnologia vocale è particolarmente adatta per l'uso nella robotica e nell'automazione in una serie di applicazioni industriali, come catene di montaggio, veicoli senza pilota (droni, aircraft), agricoltura e settore automobilistico [23]. In letteratura, diverse soluzioni sono state proposte in particolare nel dominio della smart home.

In [15], gli autori hanno sviluppato un sistema di controllo vocale per un *home network system*, in cui tutte le interazioni sono gestite da un agente virtuale: in questo modo, le interazioni risultano meno meccaniche e inespressive. L'agente virtuale è stato sviluppato come un Web Service utilizzando l'MMDAgent Toolkit. In [16], invece, viene presentato un sistema di home automation, il quale prevede sia un'interfaccia utente grafica che un sistema di controllo vocale. La parte di riconoscimento vocale è implementata attraverso il servizio "Microsoft Speech Recognition", il quale permette la vantaggiosa opzione di aggiungere nuove parole nel vocabolario riconosciuto dal sistema. Tuttavia, la piattaforma sviluppata consente solamente l'accensione/spegnimento dei dispositivi della casa, non permettendo di eseguire altre operazioni. L'obiettivo, in [17], è quello di creare un sistema di controllo vocale interattivo su una rete ZigBee. Applicando la libreria di riconoscimento vocale fornita da SUNPLUS, viene usato un algoritmo multi-step per identificare quale dei vari dispositivi della rete deve effettuare l'operazione corrispondente. Grazie alla suddivisione della fase di riconoscimento in più step, l'algoritmo risulta essere semplice e flessibile. Tuttavia, tale implementazione richiede una serie di step che rallentano l'esecuzione dell'operazione richiesta.

2.4 Privacy

La possibilità di utilizzare il controllo vocale nella vita quotidiana porta, inevitabilmente, una maggiore preoccupazione per la privacy degli utenti: in casa, in ufficio o in altri ambienti quotidiani, la presenza di device muniti di microfono preoccupa gli utenti circa le informazioni personali che tali

dispositivi possono registrare e diffondere. Tuttavia, è necessario fare una distinzione sulle capacità di registrazione di tali device, classificandoli in tre categorie [14]:

1. **Attivati manualmente:** l'utente preme un pulsante per accendere il microfono e cominciare a registrare e trasmettere audio nel cloud.

2. **Attivati vocalmente:** rimangono in fase di ascolto e si attivano solo in seguito alla rilevazione di una parola chiave. Da quel momento, cominciano a registrare e inviare dati.

3. **Sempre attivi:** registrano e trasmettono dati continuamente.

A seconda della tipologia, dunque, risultano differenti implicazioni riguardo la privacy degli utenti. L'abilità dei device di essere attivati tramite un comando (vocale o fisico) è un passo significativo verso l'integrazione dell'uso della voce nella vita di tutti i giorni. Dall'altro lato, però, alcuni dispositivi, come le telecamere di sicurezza, hanno la necessità di rimanere sempre attivi per svolgere la propria funzione. Considerando i benefici dei dispositivi comandati tramite voce, in parallelo con le implicazioni riguardo la privacy, portano le compagnie a dover trovare un compromesso tra le potenzialità del controllo vocale e la protezione della privacy dell'utente.

Parte II

Progettazione ed Implementazione

Capitolo 3

Progetto

Questo capitolo offre una panoramica della fase di progettazione del sistema SensorSpeak. In particolare, dopo aver definito gli obiettivi e i requisiti principali, viene descritta la struttura del sistema. Di ogni componente, quindi, vengono illustrate in dettaglio le caratteristiche e le funzionalità, oltre alla descrizione del ruolo assunto all'interno dell'architettura.

3.1 Obiettivo e requisiti

Lo scopo del progetto è la realizzazione di **SensorSpeak**, un sistema di monitoraggio e controllo di dispositivi IoT tramite comandi vocali. L'approccio utilizzato nella progettazione del sistema permette il suo utilizzo in diversi contesti, ad esempio Smart Home o Wireless Sensor Network, e con svariate tipologie di dispositivi IoT. Inoltre, SensorSpeak permette di agire non solo su uno specifico device, ma anche su più dispositivi contemporaneamente, aggregando i dati provenienti da vari sensori oppure eseguendo specifici comandi. L'interazione vocale, attualmente disponibile solo in lingua inglese, è gestita tramite il servizio Amazon Alexa mediante l'implementazione di un'opportuna skill.

Requisiti fondamentali per la realizzazione del progetto sono:

- **Scalabilità:** per garantire la facile aggiunta o rimozione dei dispositivi dal cloud.
- **Eterogeneità:** per supportare l'utilizzo di diverse tipologie di dispositivi.
- **Connessione costante:** per avere un'interazione in tempo reale con i dispositivi presenti nel cloud.
- **Semplicità di interazione:** per consentire all'utente di invocare comandi in un linguaggio discorsivo e ottenere feedback sull'operazione effettuata.

3.2 Scenario

Gli sviluppi nel campo dell'elettronica digitale e delle comunicazioni wireless ha permesso la pervasività e la diffusione di sensori e dispositivi IoT in svariati campi applicativi. La Cisco Internet Business Solutions Group [1] stima che il numero di dispositivi connessi nel 2020 sarà di circa 7 per persona. Tuttavia, questa stima considera la popolazione intera e, quindi, anche quella parte che non ha ancora accesso ad Internet. Di conseguenza, prendendo in considerazione solamente la popolazione che utilizza Internet, la stima cresce notevolmente.

Scenari classici in tale ambito sono quelli delle smart home e delle reti di sensori. Mentre nelle smart home i vari device hanno, di solito, funzioni sconnesse tra loro, nelle reti di sensori, invece, i vari nodi interagiscono e cooperano. Quindi, da un lato c'è bisogno del controllo sul singolo dispositivo, dall'altro potrebbe essere necessario intervenire su più dispositivi contemporaneamente. Lo scenario generale che si va a considerare, perciò, astrae tali circostanze permettendo al sistema di funzionare sia con device isolati che con device appartenenti ad una rete. Tale approccio permette, in aggiunta, di ottenere i dati da un singolo dispositivo oppure di aggregare i dati provenienti da diversi nodi ed analizzarli.

Si assume, inoltre, che ogni dispositivo svolga una singola funzione e possa essere identificato attraverso di essa (es. sensore della temperatura, luce). Infatti, considerando che il sistema interagisce in maniera vocale con l'utente, è necessario utilizzare nomi o espressioni vicini al linguaggio parlato quotidianamente e non abusare di termini tecnici. Con un numero sempre maggiore di device da gestire, la voce rappresenta la modalità di interazione più semplice ed intuitiva per l'uomo, essendo il suo modo naturale di interagire. I progressi nel campo del riconoscimento vocale e la nascita degli smart personal assistant consentono di non pensare più al controllo vocale solo come ad una visione futuristica ma come un'idea attuabile e innovativa.

3.3 Funzionalità

L'applicazione sviluppata fornisce una serie di funzionalità, ognuna delle quali invocabile attraverso comandi vocali intuitivi a seconda dell'operazione desiderata:

- **Aggiunta/rimozione device:** per creare un collegamento del device all'interno del cloud oppure rimuoverlo da esso.
- **Elenco di tutti i device:** per elencare tutti i dispositivi disponibili nel cloud.
- **Elenco dei device di un tipo specifico:** per elencare tutti i dispositivi di una certa tipologia (ad esempio, i sensori della temperatura) e la loro locazione.
- **Elenco dei device in una locazione specifica:** per elencare tutti i dispositivi che si trovano all'interno di una locazione (ad esempio, in ufficio).
- **Elenco dei device in uno status specifico:** per elencare tutti i dispositivi che sono accesi o spenti.

- **Accensione/spegnimento di un device:** per l'attivazione o la disattivazione momentanea di un dispositivo.

- **Accensione/spegnimento di tutti device:** per attivare o disattivare tutti i dispositivi collegati al cloud.

- **Accensione/spegnimento dei device di un tipo specifico:** per attivare o disattivare solo i dispositivi di una certa tipologia (ad esempio, tutti i sensori della temperatura).

- **Accensione/spegnimento dei device di una locazione specifica:** per attivare o disattivare solo i dispositivi situati in una certa locazione (ad esempio, tutti i dispositivi presenti in ufficio).

- **Ottenere il valore rilevato da un sensore:** per conoscere qual è il valore rilevato, ad esempio, dal sensore della temperatura.

- **Ottenere l'intervallo di aggiornamento di un device:** per conoscere ogni quanto tempo un dispositivo effettua la sua funzione.

- **Ottenere lo status di un device:** per sapere se un device è attivato o disattivato al momento.

- **Ottenere il livello di batteria di un device:** per conoscere qual è la carica rimanente della batteria. Inoltre, viene effettuata una previsione della durata rimanente della batteria.

- **Ottenere la media dei valori rilevati dai sensori di un tipo specifico:** per conoscere la media di un particolare valore in base a vari sensori (ad esempio, la temperature media delle stanze in casa).

- **Ottenere il valore massimo/minimo rilevato da un sensore:** per conoscere il valore massimo/minimo registrato in una determinata locazione (ad esempio, la temperature massima registrata in ufficio).

- **Ottenere quando è stato l'ultimo aggiornamento di un device:** per conoscere quando è stata l'ultima volta che un dispositivo si è connesso al cloud.

- **Impostare l'intervallo di aggiornamento di un device:** per definire ogni quanto tempo un dispositivo deve eseguire una determinata azione.

3.4 Architettura

La realizzazione del sistema di controllo vocale descritto in questa tesi prevede l'utilizzo e l'interazione di diverse componenti, sia hardware che software, come mostrato in Figura 3.1.

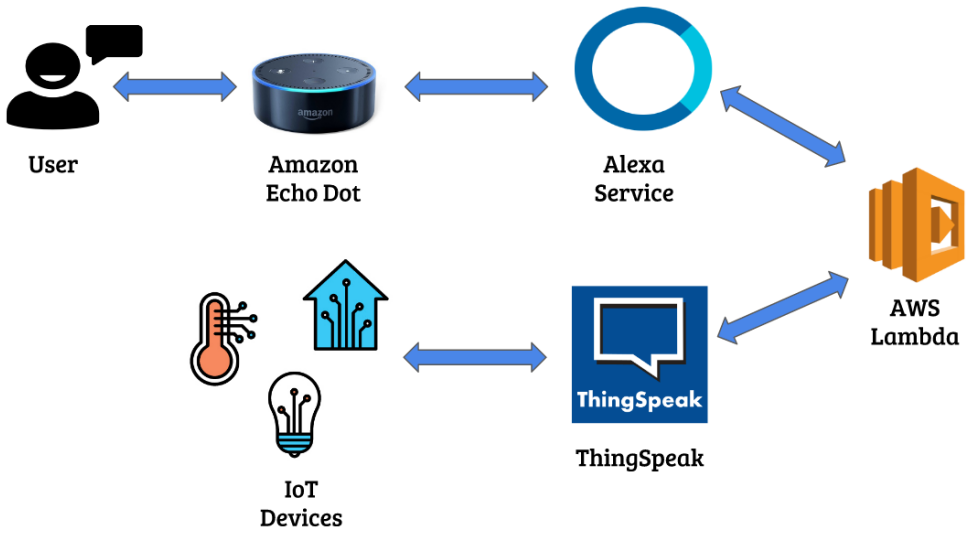

Figura 3.1: Architettura hardware e software di SensorSpeak

Di seguito, vengono descritte le varie componenti e le funzioni svolte all'interno dell'architettura proposta. I dettagli implementativi, invece, saranno forniti nel prossimo capitolo.

3.4.1 Amazon Echo Dot

L'*Amazon Echo Dot* è il dispositivo che fa da tramite tra l'utente e la logica della skill. Dotato di microfono e altoparlante, interagisce vocalmente con l'utente per raccogliere le sue richieste e, successivamente, inviare l'audio registrato all'*Alexa Service*. Inoltre, una volta processata la richiesta dell'utente, esso fornisce una risposta in maniera discorsiva circa l'esito dell'operazione richiesta. Per iniziare una conversazione con una determinata skill, è necessario, innanzitutto, attivare il dispositivo attraverso la "*wake word*" (di solito la parola "Alexa") e, successivamente, invocare la skill tramite un "*invocation name*" ("SensorSpeak" in tal caso). In questo modo, l'utente può accedere a tutte le funzionalità fornite dalla skill invocata.

3.4.2 Alexa Service

L'*Alexa Service* è il servizio di "*speech recognition*" sviluppato da Amazon che permette di tradurre in testo i comandi vocali inviati dall'utente. Inoltre, l'Alexa Service è responsabile dell'interpretazione della richiesta effettuata dall'utente e del corretto indirizzamento di tale richiesta all'endpoint designato per il suo processamento. L'interpretazione e l'indirizzamento della richiesta dipendono dalla skill invocata. È possibile, tramite l'ASK, implementare nuove skill e, di conseguenza, aggiungere nuove funzionalità ad Alexa. A tal proposito, per la realizzazione del sistema descritto in questa tesi, è stata implementata una *Custom Skill* che offre le varie funzionalità descritte sopra. Per l'implementazione di una Custom Skill, è necessario definire un modello di interazione vocale attraverso il quale è possibile mappare gli input dell'utente alle intenzioni che l'endpoint può gestire. La costruzione del modello di interazione prevede la definizione dei seguenti elementi:

- **Intent:** rappresenta un'azione che soddisfa una richiesta dell'utente; in pratica, identifica una delle funzionalità offerte dalla skill. Gli intent sono definiti in formato JSON e possono avere, eventualmente, dei parametri detti *slot*.

- **Utterance:** una frase o un'espressione che identifica uno specifico intent. Ogni intent può essere identificato da più utterance.
- **Slot:** una variabile che può assumere diversi valori a seconda del *tipo* con cui è definita. Un tipo è una lista di possibili valori che uno slot può assumere. È possibile creare nuovi tipi oppure utilizzare quelli predefiniti di Amazon.
- **Dialog Model:** una struttura che identifica gli step per una conversazione fatta di domande e risposte tra la skill e l'utente, al fine di collezionare tutte le informazioni necessarie al completamento della richiesta. Tale struttura semplifica, quindi, il modo di fornire le informazioni da parte dell'utente. Tuttavia, è un elemento opzionale nella definizione del modello d'interazione.

Figura 3.2: Esempio di invocazione di un intent

L'intent di voler accendere le luci dell'ufficio, ad esempio, potrebbe strutturarsi come mostrato in Figura 3.2.

Infine, altra funzione svolta dall'Alexa Service, è la ricezione della risposta restituita dall'endpoint e della sua traduzione *text-to-speech*, per consentire all'Echo Dot di fornire un feedback vocale all'utente. Eventualmente, la risposta può essere fornita anche all'interno dell'Alexa App, la quale permette di visualizzare testo e immagini.

3.4.3 AWS Lambda

AWS Lambda è un servizio di computazione che consente di eseguire codice senza la necessità di fornire o gestire server. È possibile eseguire il codice,

per qualsiasi tipo di applicazione o servizio backend, attraverso una potente infrastruttura di calcolo che gestisce autonomamente tutte le risorse. Attualmente, il codice può essere sviluppato in uno dei linguaggi supportati, cioè Python, Node.js, C# o Java [30]. Caricando il codice sviluppato e tutte le dipendenze necessarie, viene creata una *Lambda Function*. In particolare, la Lambda Function comunica con il servizio Alexa attraverso un meccanismo di "requeste-response" utilizzando il protocollo *HTTP*. Quando un utente interagisce con una skill, la Function riceve una richiesta in formato JSON, la quale contiene i parametri necessari al servizio per l'elaborazione e la restituzione della risposta.

Nel progetto descritto in questa tesi, la Lambda Function rappresenta la logica della skill. Infatti, riceve la richiesta formulata dall'utente in formato JSON, interpreta qual è l'intent invocato, processa la richiesta interagendo col Web Service esterno ed, infine, restituisce la risposta, sempre in formato JSON, che verrà tradotta in voce dall'Alexa Service.

3.4.4 ThingSpeak

ThingSpeak è una piattaforma open source per l'IoT che permette agli utenti di collezionare e salvare nel cloud dati prodotti da sensori e dispositivi vari. Inoltre, la piattaforma consente di analizzare e visualizzare i dati tramite MATLAB e di agire su di essi. La piattaforma mette a disposizione degli utenti delle apposite strutture chiamate "*channel*", all'interno delle quali è possibile salvare qualsiasi tipo di dato in formato testuale. Ogni channel è costituito da un identificavo, un nome, otto *field* che possono contenere i dati, più tre campi per informazioni di localizzazione. Utilizzando delle API *REST*, l'utente può creare, eliminare o aggiornare un channel, inviare nuove misurazioni effettuate dai sensori o leggere quelle già salvate. Attraverso tali API, quindi, l'utente può gestire i propri channel in ogni aspetto: l'unico requisito necessario è fornire la chiave d'accesso associata al proprio account di ThingSpeak. Sono disponibili anche delle API per il protocollo *MQTT*, le quali, però, consentono solamente l'aggiornamento dei dati di un canale.

L'aggiornamento di un canale può essere effettuata ogni 15 secondi mentre, per accedere ai dati, non si hanno vincoli di tempo [31].

Considerate le caratteristiche della piattaforma, ThingSpeak rappresenta un ideale servizio cloud tramite il quale raggruppare e gestire i dispositivi IoT dell'utente. Infatti, grazie alle API REST, è possibile automatizzare numerose funzioni e consentire alla Lambda Function di interagire con ThingSpeak, per accedere ai dati prodotti dai device oppure agire su di essi.

3.4.5 Dispositivi IoT

I dispositivi IoT sono rappresentati da tutti quei sensori, attuatori o altri tipi di device che l'utente può gestire tramite comando vocale. I requisiti necessari, che un dispositivo deve rispettare affinché possa essere gestito tramite SensorSpeak, sono essenzialmente due:

1. **Connessione ad Internet:** wired o wireless.
2. **Collegamento a ThingSpeak:** il dispositivo deve poter essere associato ad un channel nel cloud dell'utente.

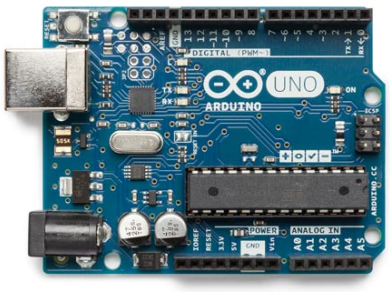

Figura 3.3: Arduino Uno

Numerosi dispositivi, dunque, possono essere utilizzati all'interno del sistema di controllo vocale. Alcuni test sono stati effettuati su diversi sensori e attuatori collegati alla board *Arduino Uno* (Figura 3.3).

Tuttavia, altre tipologie di board, anche non compatibili con Arduino, possono essere facilmente integrate nel sistema, come *Raspberry Pi*, *Intel Galileo 2*, *Arduino Nano*, *STM32 Nucleo*. Di seguito, vengono descritte le caratteristiche delle principali componenti utilizzate.

Arduino Uno

Arduino Uno è una board basata sul microcontrollore *ATmega328P* [32]. Possiede 14 pin di input/output digitali e 6 ingressi analogici attraverso i quali è possibile leggere valori in input dei sensori connessi e produrre un output su attuatori o software esterni. La board è la prima di una serie di board Arduino provviste di USB ed è il modello di riferimento della piattaforma Arduino. Le specifiche tecniche sono riassunte nella tabella seguente:

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage	6-20V (7-12V recommended)
Digital I/O Pins	14 (6 for PWM output)
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (0.5 for bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
Dimensions	68.6mm (L) x 53.4mm (W)
Weight	25 g

Tabella 3.1: Specifiche tecniche Arduino Uno [32]

Arduino Ethernet Shield

L'*Arduino Ethernet Shield* permette alla board Arduino di connettersi ad Internet e di comunicare con Web Service esterni. La shield necessita di una scheda Arduino per poter funzionare e si connette ad essa attraverso lunghi connettori che si estendono alla base della shield (Figura 3.4): tale struttura permette sia alla shield di ottenere i 5V necessari per il proprio funzionamento, sia di lasciare intatto il layout dei pin. La shield è basata sul chip ethernet *Wiznet W5100* che supporta sia TCP che UDP. Essa include, inoltre, un jack ethernet standard *RJ45*, uno slot per utilizzare una memoria SD e dei LED informativi circa lo stato della rete e delle connessioni attuali.

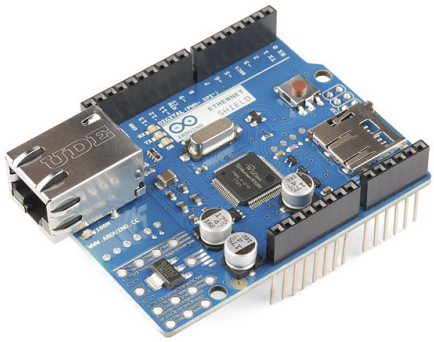

Figura 3.4: Arduino Ethernet Shield

Sensori e attuatori

I sensori e gli attuatori testati sono:

- **Fotoresistore:** il fotoresistore (o anche *Light Dependant Resistor*) è un resistore variabile: la resistenza è inversamente proporzionale alla quantità di luce che lo colpisce. Viene utilizzato, dunque, per misurare la luce rilevata nell'ambiente circostante. Il modello utilizzato è quello prodotto da *TinkerKit* (Figura 3.5), il quale è costituito oltre che dal fotoresistore, anche da un connettore a 3-pin (+5V, Data, GND) per il collegamento alla board, da un LED verde che segnala se il modulo è

alimentato correttamente e da un LED giallo la cui luminosità cambia a seconda della quantità di luce percepita. Questo modulo dà in output 5V quando il sensore non rileva luce (circuito aperto) e 0V quando, invece, è esposto a forti fonti luminose come il sole (circuito chiuso).

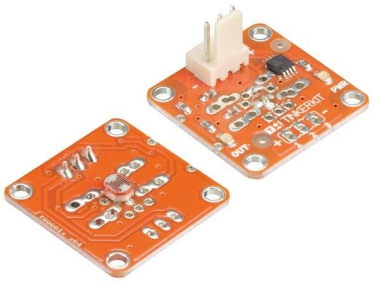

Figura 3.5: TinkerKit Photoresistor Sensor

- **DHT11:** il modulo DHT11 fornisce in output un segnale digitale proporzionale alla temperatura e all'umidità relativa rilevata dal sensore stesso. Il sensore è in grado di rilevare temperature da 0 a $+50°C$ con un'accuratezza di $\pm 2°C$ e un range di umidità da 20 a 90% con un'accuratezza di $\pm 5\%$. Il modulo si interfaccia con l'esterno tramite un connettore a 4-pin (+5V, Data, GND e uno non utilizzato) richiedendo circa 1 secondo di attesa tra due rilevamenti consecutivi. Grazie alle ridotte dimensioni e al costo esiguo, tale modulo viene spesso utilizzato in sistemi di monitoraggio ambientale.

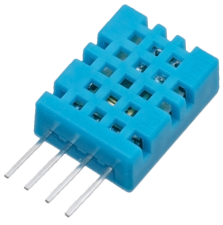

Figura 3.6: Sensore DHT11 per temperatura e umidità

- **LED:** il LED (*Light Emitting Diode*) è una fonte di luce a basso consumo ed è considerato il più semplice degli attuatori. Si accende quando è alimentato da corrente, ad esempio collegandolo ad un pin di Arduino. Il modello prodotto da *TinkerKit* fornisce, oltre al LED disponibile in vari colori, anche un connettore a 3-pin (+5V, Data, GND) ed una resistenza che garantisce al modulo la quantità di corrente ottimale per il LED.

Figura 3.7: TinkerKit Green Led da 10 mm

Capitolo 4

Implementazione

La realizzazione del sistema prevede l'integrazione di diverse componenti al fine di fornire le funzionalità descritte precedentemente. In questo capitolo, viene illustrata, appunto, la fase di implementazione del progetto, durante la quale sono state definite diverse componenti software necessarie all'interazione e all'integrazione di differenti tecnologie. In particolare, sono mostrati i dettagli implementativi della Lambda Function e del modello di interazione vocale, oltre ad un possibile esempio di sketch eseguibile su Arduino. Infine, viene descritto l'algoritmo di predizione del tempo di carica rimanente delle batterie, del quale verranno esaminati i risultati nel successivo capitolo.

4.1 Flusso di interazione

Il sistema SensorSpeak prevede l'interazione di diverse componenti per soddisfare le richieste effettuate dall'utente. Una volta associato un dispositivo al cloud, esistono sostanzialmente due process flow. Il primo è quello che permette la comunicazione tra il device e la rete cloud e prevede 2 step:

1. Il device esegue il proprio task (ad esempio rilevamento di una grandezza fisica) ed invia i dati nella rete cloud ThingSpeak.

2. Il channel ThingSpeak riceve i dati e li salva nell'apposito field.

L'altro flusso, invece, prevede l'interazione dell'utente con il sistema:

1. L'utente invoca un comando seguendo il modello di interazione previsto dalla skill.
2. L'Alexa Service interpreta la richiesta e la trasforma in formato JSON.
3. Una Lambda Function processa la richiesta comunicando con la rete cloud di ThingSpeak per ottenere i dati necessari ed, infine, restituisce una risposta in formato JSON.
4. L'Alexa Service traduce il testo in voce e comunica l'esito della richiesta all'utente.

4.2 Identificazione del dispositivo nel cloud

Per poter gestire i dispositivi, è necessario registrarli nella rete cloud dell'utente su ThingSpeak. Ogni channel su ThingSpeak è associato ad un solo device. Di conseguenza, ogni device è identificabile univocamente tramite l'ID autogenerato del channel al quale è associato. Tuttavia, durante l'interazione vocale col sistema, risulta poco pratico all'utente dover identificare i propri device tramite un ID numerico composto da diverse cifre. Per tale motivo, si rivela più semplice ed intuitivo associare dei nomi ai vari dispositivi. Tale nome viene definito dall'utente invocando la corrispondente funzionalità e viene salvato nel campo *"Nome"*, appunto, del channel appena creato. Per essere certi che l'operazione invocata venga effettuata sul dispositivo desiderato, anche il nome deve essere univoco all'interno del cloud dell'utente. A tal fine, si utilizza una coppia di valori <*locazione, tipo*> come identificativo del device:

- **Locazione:** luogo in cui il device è posizionato (es. cucina, ufficio, giardino).
- **Tipo:** indica la tipologia di device (es. sensore dell'umidità, luce, termostato).

Con questa struttura, si assume che un utente possa avere più dispositivi dello stesso tipo ma che siano posizionati in luoghi diversi. Tale strutturazione è necessaria per due motivazioni. Innanzitutto, come accennato, semplifica ed evita ambiguità nella formulazione dei comandi: per accendere la luce in cucina, invece che nel bagno, ad esempio, basterà pronunciare la frase *"turn on kitchen light"*. La seconda motivazione deriva dalle caratteristiche di Alexa: è necessario definire un insieme finito di possibili valori che l'utente può dare in input. Pertanto, la locazione e il tipo dei device sono definiti come slot all'interno delle possibili utterance. In particolare, la *locazione* utilizza la tipologia *Room* di Amazon, la quale fornisce nomi tipici di stanze di vari edifici; per il *tipo* del device, invece, è stata creata una nuova tipologia di slot, denominata *DeviceType*, nella quale sono state indicate differenti categorie di device.

Ogni channel, inoltre, presenta una struttura predefinita utile al salvataggio dei dati dei sensori e al controllo dei device in generale. Tale struttura viene generata in fase di creazione del channel e sfrutta i vari *field* messi a disposizione da ThingSpeak. In particolare, ogni channel comprende i seguenti campi:

- **Type:** identifica il tipo del device.
- **Battery:** salva le informazioni riguardo la carica della batteria.
- **State:** indica lo stato del device (es. acceso, spento).
- **Value:** colleziona i rilevamenti effettuati dai sensori o salva qualsiasi tipo di dato inviato dal device.
- **Update Interval:** denota l'intervallo di aggiornamento dei dati del device, cioè quanto tempo intercorre tra una connessione e l'altra al canale.

Organizzando i channel in questa maniera e utilizzando le API REST messe a disposizione da ThingSpeak, è possibile implementare numerose funzionalità per diverse tipologie di device.

4.3 Modello di interazione vocale

Come descritto nel capitolo precedente, il modello di interazione vocale necessita la definizione di alcuni elementi. Il primo passo è quello di identificare gli *intent* che la skill può gestire. Successivamente, bisogna definire quali sono le possibili *utterance* che permettono di invocare uno specifico intent.

Date le numerose funzionalità offerte dalla skill, risulta conveniente associare ognuna di esse ad un intent specifico. Tuttavia, avere un gran numero di funzionalità potrebbe confondere l'utente nella formulazione dei comandi. Per questo motivo, è necessario stabilire un meccanismo che consenta all'utente di indentificare facilmente i comandi da invocare in base alle proprie intenzioni. Considerando le varie funzionalità fornite, notiamo che, sostanzialmente, le azioni che il sistema può effettuare sono le seguenti:

- **Create:** creazione di un dispositivo nel cloud.
- **Remove:** rimozione di un dispositivo dal cloud.
- **List:** elenco dei dispositivi in base a particolari proprietà.
- **Get:** ottenere un valore.
- **Set:** impostazione di un valore.
- **Turn On/Off:** accensione o spegnimento dei dispositivi nel cloud.

A questo punto, risulta intuitivo schematizzare le utterance in base al tipo di azione che l'intent rappresenta. In particolare, si utilizza come prima parola di ogni utterance l'azione corrispondente all'intent al quale fa riferimento. Ad esempio, se si vogliono elencare tutti i dispositivi nel cloud, una possibile utterance potrebbe essere: *"List all devices"*.

Tutte le utterance sono definite in modo tale da rispecchiare l'intenzione dell'utente ed avvicinarsi il più possibile al linguaggio parlato. Tuttavia, data l'eterogeneità delle possibili azioni che il sistema può attuare, non è possibile definire una struttura standard per le utterance. Infatti, a seconda del tipo

di intent, possono essere richiesti parametri (slot) differenti per la corretta interpretazione. Inoltre, l'utente potrebbe erroneamente specificare solo una parte degli slot necessari al processamento della richiesta. Per questo motivo, il modello di interazione consente due modalità di invocazione di un intent:

- **Frase completa:** l'utente specifica tutti i parametri necessari all'interpretazione della richiesta.
- **Dialogo:** l'utente fornisce parte, o nessuno, dei parametri richiesti e il sistema lo guida nel completamento della richiesta.

Ad esempio, l'intent di spegnere le luci della cucina richiede due parametri, cioè il tipo di device da spegnere ("luce") e la locazione nella quale esso si trova ("cucina"). Un esempio di interazione usando la frase completa potrebbe essere il seguente:

```
Utente: "Turn off kitchen light."
Alexa:  "Well done! Kitchen light turned off."
```

Mentre, usando la forma del dialogo:

```
Utente: "Turn off."
Alexa:  "What's the type of the device?"
Utente: "Light."
Alexa:  "What's the location of the device?"
Utente: "Kitchen."
Alexa:  "Well done! Kitchen light turned off."
```

La seconda modalità risulta più lenta e schematica ma diventa molto utile quando l'utente non ha preso ancora confidenza col sistema oppure quando il sistema di riconoscimento vocale stesso non ha ben interpretato la richiesta a causa di fattori esterni (rumore di sottofondo) e quindi chiede all'utente di ripetere l'input mancante. Infine, data la criticità di alcune operazioni, come

la rimozione dal cloud di un device, il sistema può chiedere all'utente una conferma prima dell'esecuzione dell'operazione.

Nell'Appendice A sono elencati tutti gli intent implementati e, per ognuno di essi, viene fornita una descrizione dei parametri e delle utterance d'esempio.

4.4 Lambda Function

La *Lambda Function* è il codice che viene eseguito dall'Amazon Lambda Service per processare le richieste effettuate dall'utente e rappresenta la logica della skill. Il codice della Lambda Function, sviluppato nel linguaggio Python, può essere suddiviso in tre componenti principali di seguito descritte.

4.4.1 Lambda handler

Al momento della creazione della Function, bisogna specificare un *handler*, cioè una funzione all'interno del codice che il Lambda Service può invocare quando il servizio riceve una richiesta inviata da Alexa. Tale richiesta è ricevuta in formato JSON e presenta una struttura come mostrato di seguito:

```
"request": {
    "type": "IntentRequest",
    "requestId": "id",
    "intent": {
         "name": "GetValueIntent",
         "slots": {
              "location": {
                  "name": "location",
                  "value": "kitchen"
               },
               "type": {
                  "name": "type",
```

```
                "value": "humidity"
            }
        }
    },
    "locale": "en-US",
    "timestamp": "2017-09-03T15:27:39Z"
}
```

Il *Lambda handler*, quindi, verifica che la richiesta sia ben formata e individua il nome dell'intent invocato al fine di identificare qual è la funzione preposta alla gestione della richiesta. Successivamente, invoca l'*Intent handler* appropriato passando come parametro il campo *"slots"* della richiesta.

4.4.2 Intent handler

L'*Intent handler* si occupa di gestire la richiesta dell'utente. Esso interagisce con la rete cloud di ThingSpeak e fornisce i parametri necessari alla costruzione della risposta al *Response builder*. Data la moltitudine di funzionalità implementate, esiste un Intent handler per ogni tipologia di intent messa a disposizione dalla skill. Nonostante le differenze tra i vari intent, però, è possibile definire una struttura generale per tutti gli Intent handler (Algoritmo 1).

Algorithm 1 Intent handler

procedure IntentHandler(slots)
- *check* ← checkSlots(*slots*)
- **if** *check* = *False* **then return** *elicitSlot()*
- *deviceName* ← buildDeviceName(*slots*)
- *HTTPRequest* ← buildHTTPRequest(*key, deviceName, slots*)
- *HTTPResponse* ← sendRequest(*HTTPRequest*)
- *speechResponse* ← buildSpeechResponse(*HTTPResponse*)
- **return** *responseBuilder(speechResponse)*

Innanzitutto, la procedura esegue un controllo degli slot per vedere se sono stati forniti correttamente dall'utente. In caso negativo, viene attivata la modalità dialogo per completare la definizione degli slot. Una volta definiti tutti i parametri, viene creata l'API REST appropriata utilizzando il nome del device (ricavato dalla coppia di parametri <locazione, tipo>), gli slot ricevuti dalla richiesta e la chiave segreta d'accesso al cloud. A questo punto, la richiesta viene inviata al cloud di ThingSpeak che risponde con l'esito dell'operazione. La risposta viene interpretata e trasformata in una stringa che rappresenta l'output vocale fornito da Alexa. Infine, tale stringa viene passata al *Response builder*.

4.4.3 Response builder

Il *Response builder* si occupa di creare la risposta in formato JSON fornendo tutti i campi necessari ad Alexa per la traduzione in voce. Inoltre, quando previsto, la risposta contiene anche i campi necessari alla creazione di una *card* che verrà visualizzata attraverso l'Alexa App. La struttura della risposta è la seguente:

```
"response": {
  "speechletResponse": {
    "outputSpeech": {
      "text": "Humidity in the kitchen is at 23%."
    },
    "card": {
      "content": "Humidity in the kitchen is at 23%.",
      "title": "Kitchen humidity"
    },
    "reprompt": {
      "outputSpeech": {
        "text": "Humidity in the kitchen is at 23%. Ask me
                 something else!"
      }
```

```
    },
    "shouldEndSession": false
  }
}
```

Il campo *"outputSpeech"* contiene il testo che verrà pronunciato da Alexa e rappresenta, quindi, la risposta alla richiesta dell'utente. Nel campo *"reprompt"*, invece, è possibile specificare il testo da ripetere nel caso l'utente lo richieda. È buona norma ripetere la risposta fornita dal sistema, magari inserendo informazioni aggiuntive per chiarire eventuali dubbi. Nel campo *"card"* viene indicato il contenuto testuale che apparirà all'interno dell'app (Figura 4.1). Tale campo non è obbligatorio e può includere anche l'opzione per la visualizzazione di immagini. Infine, tramite il campo *"shouldEndSession"*, si indica alla skill se la sessione di interazione attuale deve essere interrotta o meno dopo aver fornito la risposta.

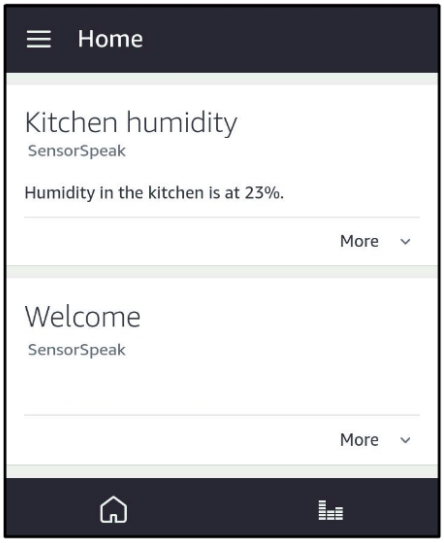

Figura 4.1: Card nell'Alexa App

4.5 Arduino sketch

I dispositivi che possono essere gestiti tramite SensorSpeak, così come le funzioni da essi implementate, sono numerosi e differenti. Per questo motivo, lo sviluppo del codice eseguito dai dispositivi dipende strettamente dagli scopi per i quali sono utilizzati e non è possibile definire una struttura standard. L'unico requisito necessario che il dispositivo deve supportare per poter essere utilizzato attraverso il sistema è la capacità di leggere o scrivere dati sul channel al quale è associato.

In questa sezione, viene riportato l'esempio di un semplice sensore della temperatura che invia i dati rilevati ad una board Arduino Uno alimentata a batteria (Algoritmo 2). Nella funzione *setup()*, vengono inizializzati il sensore della temperatura e la connessione ad Internet. Nella funzione *loop()*, invece, vengono letti il valore della temperatura e il livello della batteria. Successivamente, viene effettuata la connessione al server ThingSpeak per permettere l'aggiornamento dei dati sul channel al quale il dispositivo è associato. Infine, la board entra in modalità sleep fino al prossimo rilevamento della temperatura.

Algorithm 2 Arduino Sketch

```
procedure SETUP
    Initialize Sensor
    Initialize Internet Connection

procedure LOOP
    temp ← readSensorValue()
    battery ← readBatteryLevel()
    connect(ThingSpeakServer)
    updateChannel(temp, battery)
    sleep(interval)
```

4.6 Stima della durata della batteria

I dati inviati dai dispositivi riguardo il livello della batteria dalla quale sono alimentati possono essere utilizzati non solo per conoscere il livello attuale di carica, ma anche per effettuare una stima della durata rimanente della batteria. Infatti, ogni record salvato sul channel del dispositivo contiene, oltre ai dati inviati, anche il timestamp in cui è avvenuta la comunicazione. Da ogni channel, perciò, è possibile ottenere la coppia di valori <*timestamp, livello batteria*> e ricavare l'andamento del processo di scarica della batteria. Tuttavia, come descritto in [33], tale processo di scarica è differente a seconda della tipologia di batteria e influenzato da diversi fattori: temperatura, livello di usura, carico. Conoscere a priori il modello di scarica per ogni tipologia, quindi, permette di effettuare una stima più accurata. L'algoritmo implementato in SensorSpeak sfrutta un modello predefinito a seconda della tipologia di batteria e, combinandolo con i dati prelevati dal channel, effettua una stima del tempo rimanente di carica (Algoritmo 3).

Algorithm 3 Battery Lifetime Prediction

```
procedure PREDICT(batteryModel)
    channelData ← readChannel()
    trainingModel ← merge(batteryModel, channelData)
    remainingTime ← makePrediction(trainingModel)
    return remainingTime
```

Il modello preso in input dalla procedura è costituito da una lista di coppie <*timestamp, livello batteria*> che descrivono il modello di scarica della batteria in esame. Tale modello viene combinato con i dati attualmente disponibili sul channel per creare un appropriato modello di training che tenga in considerazione anche l'andamento attuale dell'utilizzo della batteria. La funzione di predizione prende in input tale modello ed effettua una stima del tempo rimanente. In particolare, la funzione di predizione utilizzata fa parte della libreria Python *scikit-learn* [34] e adopera la *Ridge regression* su un polinomio di grado maggiore di 1 (grado calcolato in base al modello in

input). Questa tipologia di regressione utilizza il *metodo dei minimi quadrati* combinato alla *regolarizzazione L2*. Il metodo dei minimi quadrati è una tecnica di regressione la quale permette di trovare una funzione che minimizzi la somma dei quadrati delle distanze tra i dati osservati e i valori stimati dall'approssimazione lineare. A seconda della complessità del dataset sul quale lavora, tale tecnica di regressione può avere problemi di *overfitting* (sovrastimare il valore corretto a causa di un'alta complessità del dataset) o di *underfitting* (sottostimare il valore corretto a causa della bassa complessità del dataset). L'applicazione della regolarizzazione L2 consente di introdurre informazioni addizionali per ridurre o aumentare la complessità del modello e ottenere un *fitting* migliore.

4.6.1 Altri approcci

L'algoritmo sopra descritto è stato confrontato, durante la fase di valutazione, con altri due approcci:

1. *Regressione polinomiale:* si tratta di una regressione lineare applicata ad un polinomio di grado n (con $n > 1$) [36]. Anche in questo caso è stata usata la libreria scikit-learn.

2. *Fitting polinomiale:* vengono calcolati, attraverso il metodo dei minimi quadrati, i coefficienti del polinomio di grado n (con $n > 1$) che meglio approssima i dati in input. Per questo approccio è stata usata la funzione *polyfit* della libreria *NumPy* [35].

Parte III

Valutazione

Capitolo 5

Valutazione sperimentale

Quest'ultimo capitolo analizza i risultati derivanti dall'applicazione dell'algoritmo di stima del tempo di carica rimanente ad una particolare tipologia di batteria, effettuando anche un confronto con altri due approcci. Vengono descritte, inoltre, le configurazioni dei test e le metriche utilizzate per la valutazione dell'algoritmo. I test sono stati effettuati prendendo in considerazione una sola tipologia di batteria. Si necessita, dunque, in futuro, la creazione di ulteriori modelli per validare l'algoritmo.

5.1 Contesto

Come descritto nei capitoli precedenti, SensorSpeak è in grado di comunicare ed interagire con un'ampia varietà di dispositivi diversi tra loro. Molto spesso, tali dispositivi, per assolvere una specifica funzione, sono posizionati in luoghi in cui non sempre è possibile utilizzare l'energia elettrica per alimentarli. Basti pensare, ad esempio, ad un semplice sensore della temperatura posizionato in giardino o ad un sensore di rilevamento di fumo collocato all'interno di un bosco per la prevenzione degli incendi. In situazioni del genere, può essere conveniente utilizzare una batteria per l'alimentazione in quanto permette al device sia di eseguire il proprio task, sia di poter essere spostato in base alle necessità. Tuttavia, le batterie hanno una durata limitata che

dipende dalle caratteristiche stesse della batteria (voltaggio, capacità, ecc.) e dal dispositivo al quale forniscono energia. Risulta necessario, perciò, sostituire o ricaricare la batteria una volta esausta e, conoscendo in anticipo il momento in cui tale situazione si verifica, si potrebbe evitare l'interruzione del servizio fornito dal device. L'Algoritmo 3 del capitolo precedente ha, appunto, lo scopo di stimare quanto tempo rimane alla batteria prima di esaurirsi. L'approccio sul quale si basa l'algoritmo permette di effettuare la stima su qualsiasi batteria di cui si conosca il modello di scarica. A tal proposito, l'algoritmo è stato testato su una particolare tipologia della quale è stato definito anche il modello.

5.2 Configurazione dei test

Considerati gli scenari applicativi di SensorSpeak, i test sono stati effettuati utilizzando una board Arduino Uno, provvista della Ethernet Shield, alla quale è stato collegato un sensore DHT11 (Figura 5.1). Lo sketch caricato sulla board è lo stesso descritto dall'Algoritmo 2 nella Sezione 4.6. La tipologia di batteria utilizzata per fornire l'alimentazione alla board è la batteria ricaricabile *YSD-12680* agli ioni di litio che possiede un voltaggio di 12 V in uscita e una capacità di 6800 mAh.

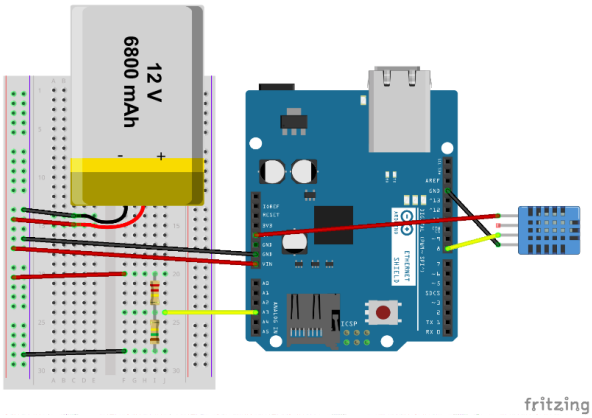

Figura 5.1: Configurazione hardware dei test

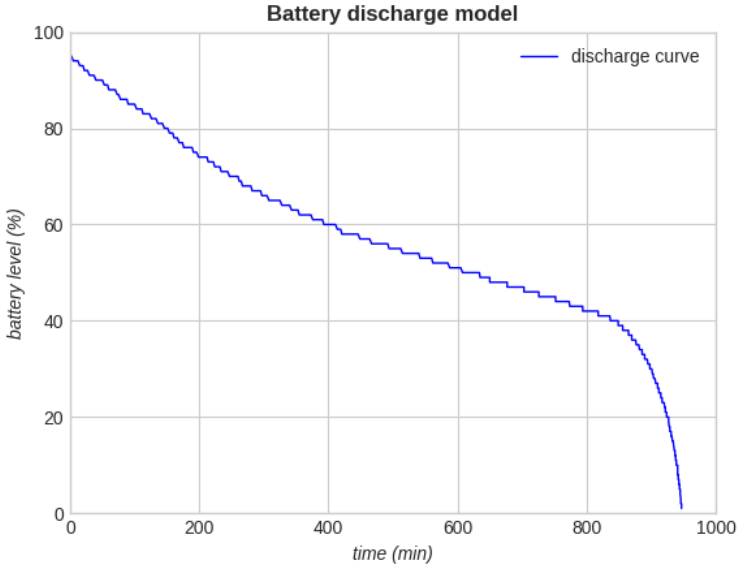

Figura 5.2: Modello di scarica della batteria

Prima di eseguire i test, sono stati effettuati dei cicli di scarica della batteria al fine di costruire il modello che verrà poi utilizzato dall'algoritmo per la stima del tempo di carica rimanente. Dato che l'algoritmo lavora con una lista di coppie <*timestamp, livello batteria*>, ad intervalli regolari sono stati effettuati i rilevamenti del livello di energia e salvati insieme al timestamp corrispondente. Da questa operazione è emerso che il modello di scarica della batteria esaminata segue l'andamento della curva descritta nella Figura 5.2. Si può osservare che il ciclo di scarica dura all'incirca 16 ore ed attraversa tre fasi:

1. Una fase iniziale di scarica veloce che va dal 100% al 60% circa.
2. Una fase centrale di scarica lenta che va dal 60% al 40% circa.
3. Una fase finale in cui si raggiunge lo 0% in maniera molto rapida.

Ogni test consiste nell'esecuzione dello sketch fino a quando la batteria non si è esaurita completamente. Tutti i test sono stati eseguiti con la stessa

configurazione hardware facendo variare, però, l'intervallo di aggiornamento dei dati inviati sul channel tra un test e l'altro. In particolare, i valori utilizzati sono stati 15, 30, 60, 90, 120, 150 e 180 secondi.

5.3 Metriche di valutazione

Data la natura dei test, un'utile metrica da valutare è rappresentata dalla differenza, in termini di tempo, tra il valore corretto e la stima fornita dall'algoritmo:

$$Error = y_{real} - y_{pred}$$

dove y_{real} rappresenta il valore reale osservato (in secondi) e y_{pred} il valore stimato dall'algoritmo (in secondi). L'obiettivo è quello di predire in anticipo il momento esatto in cui la batteria si esaurisce e, perciò, una minore differenza tra tempo reale e stima indica una maggiore accuratezza dell'algoritmo. Tuttavia, l'algoritmo può sia *sottostimare* il valore corretto, fornendo cioè un valore temporale precedente al reale momento della scarica, sia *sovrastimare*, cioè posticipare il momento in cui la batteria diventa esausta. Per tale motivo, quindi, l'errore potrebbe assumere, rispettivamente, un valore positivo o negativo. In alcuni casi, per l'analisi dei risultati, potrebbe essere più corretto prendere in considerazione il valore assoluto dell'errore misurato.

5.4 Risultati

I risultati sono stati ottenuti attraverso l'esecuzione di uno script Python che si occupa di:

1. Leggere i dati dal channel.
2. Calcolare i valori reali in base al tempo di fine del test.
3. Eseguire l'algoritmo di predizione.
4. Confrontare valori stimati con valori reali.

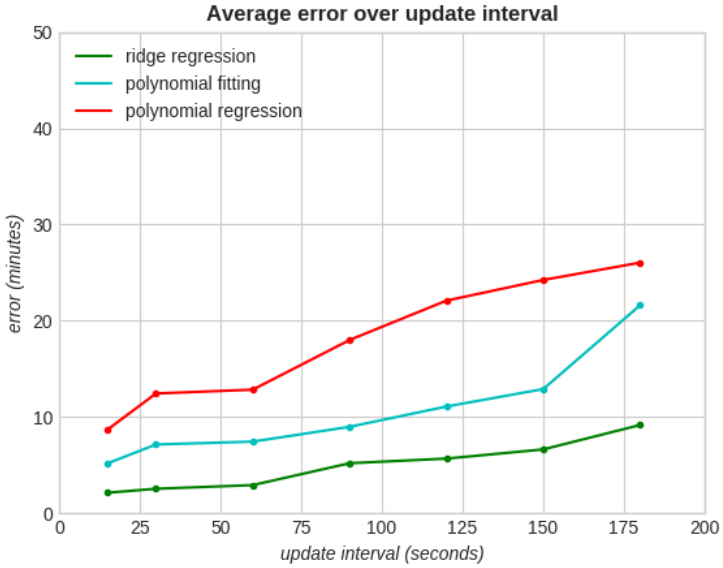

Figura 5.3: Errore medio al variare dell'intervallo di aggiornamento

Il grafico in Figura 5.3 mostra l'errore medio (in minuti) al variare dell'intervallo di aggiornamento dei dati sul channel, cioè indica quant'è, in media, la differenza in minuti tra la stima fornita dall'algoritmo e il valore reale. Dal grafico, si evince, inoltre, che all'aumentare dell'intervallo di invio dei dati, anche l'errore medio cresce. Ciò è dovuto molto probabilmente al fatto che l'algoritmo ha a disposizione un numero maggiore di dati quando gli aggiornamenti sono effettuati rapidamente. Mentre, aumentando l'intervallo di aggiornamento, i dati a disposizione sono pochi e l'algoritmo ha meno informazioni circa l'andamento attuale del processo di scarica. Per quanto riguarda il confronto fra i vari approcci testati, si nota che la regressione di tipo Ridge ottiene risultati migliori rispetto alle altre procedure.

La Figura 5.4, invece, mostra qual è l'errore massimo rilevato durante ogni test. L'andamento rispecchia i risultati mostrati dal grafico precedente e indica che la differenza massima rilevata è all'incirca di 20 minuti per la Ridge regression. Considerando la durata totale della batteria, tale errore

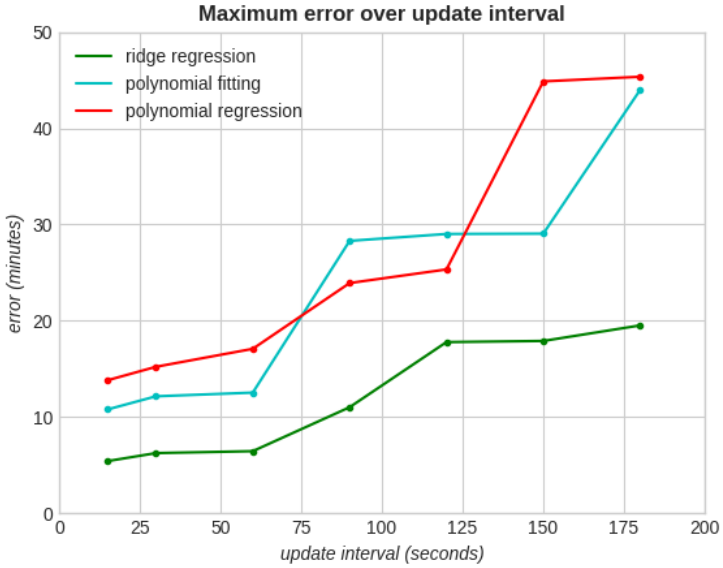

Figura 5.4: Errore massimo al variare dell'intervallo di aggiornamento

risulta poco rilevante e incisivo. Per gli altri due approcci, invece, l'errore diventa più consistente, fino ad arrivare ad un massimo di 45 minuti circa.

Tuttavia, bisogna prendere in esame anche il momento in cui tale errore si verifica, in quanto se accade quando la batteria possiede ancora un'alta percentuale di energia, l'errore risulta meno grave rispetto a quando la batteria è quasi scarica. Il grafico in Figura 5.5, appunto, mostra la distribuzione dell'errore di predizione al variare del livello di batteria. L'errore medio maggiore si presenta a ridosso del 40%-50% di carica, quando, cioè, l'andamento della curva di scarica comincia a cambiare e il livello della batteria si abbassa rapidamente. L'algoritmo non riesce a prevedere in anticipo il verificarsi di tale situazione anche a causa della differenza di consumi dovuta alla durata dell'intervallo di aggiornamento: infatti, intervalli di aggiornamento più lunghi comportano una fase di sleep maggiore per la board Arduino, con un conseguente risparmio di energia maggiore. Una volta superata tale situazione, l'errore decresce restando costantemente al di sotto dei 5 minuti.

Figura 5.5: Errore medio al variare del livello di batteria

Infine, vediamo che l'errore è maggiore all'inizio rispetto alla fine del ciclo di scarica in quanto l'algoritmo non ha ancora sufficienti informazioni riguardo l'attuale andamento dei consumi della batteria.

Come si evince dai risultati dei vari test, la regressione *Ridge* riesce ad effettuare delle stime migliori rispetto agli altri due approcci. Tuttavia, bisogna considerare che i test sono stati effettuati su un'unica tipologia di batteria. Risultano necessari, dunque, ulteriori test al fine di poter utilizzare tale procedura con differenti batterie.

Conclusioni

In questa tesi è stato descritto il sistema SensorSpeak per la gestione e il monitoraggio di dispositivi IoT mediante l'utilizzo di comandi vocali. Attraverso la semplice interazione vocale con Alexa (lo smart personal assistant di Amazon), l'utente è in grado di monitorare e controllare i dispositivi collegati nel proprio cloud. Un'importante caratteristica del sistema è che esso permette di controllare più device contemporaneamente, consentendo, ad esempio, l'aggregazione dei dati provenienti da più sensori oppure l'esecuzione di un particolare comando su molteplici dispositivi. L'approccio utilizzato nell'implementazione del sistema consente, inoltre, il suo utilizzo in diversi scenari, come smart home o reti di sensori. Data l'importanza della questione energetica in contesti del genere, è stato testato un algoritmo di predizione del tempo di carica rimanente della batteria. Prendendo in input un modello che descrive il processo di scarica della batteria che alimenta il device, l'algoritmo effettua una stima combinando tale modello con i dati inviati dal device.

SensorSpeak è stato, inoltre, installato e testato in una rete 6LoWPAN montata presso il centro di ricerca ARCES di Bologna per il monitoraggio dei sensori disposti all'interno dell'edificio.

Una serie di miglioramenti possono essere apportati a SensorSpeak:

- Aggiungere nuove funzionalità che arricchiscano l'esperienza dell'utente, consentendo di eseguire azioni più specifiche a seconda della tipologia di device.

- Estendere la compatibilità del sistema a diversi servizi di cloud hosting. Attualmente il sistema si integra unicamente con il servizio cloud fornito da ThingSpeak.

- Arricchire il set di modelli di scarica della batteria per estendere l'utilizzo dell'algoritmo alle principali batterie in commercio e migliorare l'accuratezza della stima fornita dall'algoritmo.

Appendice A

Modello di interazione vocale

Di seguito vengono elencati tutti gli *intent* implementati da SensorSpeak e le tipologie di *slot* utilizzate. Di ogni intent, inoltre, vengono definiti gli slot necessari alla formulazione della richiesta, alcune *utterance* di esempio (in lingua inglese), il comando per la modalità *dialogo*, e se è necessario fornire una conferma per l'operazione.

A.1 Slots

Type

Indica la tipologia di device: temperature, humidity, light, gas alarm.

Location

Indica il luogo in cui è posizionato il device: kitchen, office, garden.

Status

Descrive lo stato del device: on, off, standby, sleep.

Interval

Indica una durata temporale: ten minutes, seven seconds, five hours.

A.2 Intents

A.2.1 Create

CreateDeviceIntent

- **Slot:** Type, Location.
- **Utterance:** *"Create temperature sensor in the kitchen"*.
- **Modalità dialogo:** *"Create device"*.
- **Richiesta conferma:** Sì.

A.2.2 Remove

RemoveDeviceIntent

- **Slot:** Type, Location.
- **Utterance:** *"Remove kitchen temperature sensor"*.
- **Modalità dialogo:** *"Remove device"*.
- **Richiesta conferma:** Sì.

A.2.3 List

ListDevicesIntent

- **Slot:** Nessuno.
- **Utterance:** *"List all devices"*.
- **Modalità dialogo:** Non supportata.
- **Richiesta conferma:** No.

ListByLocationIntent

- **Slot:** Location.
- **Utterance:** *"List devices in the office"*.
- **Modalità dialogo:** *"List by location"*.
- **Richiesta conferma:** No.

ListByTypeIntent

- **Slot:** Type.
- **Utterance:** *"List temperature sensors"*.
- **Modalità dialogo:** *"List by type"*.
- **Richiesta conferma:** No.

ListByStatusIntent

- **Slot:** Status.
- **Utterance:** *"List devices in sleep mode"*.
- **Modalità dialogo:** *"List by status"*.
- **Richiesta conferma:** No.

A.2.4 Set

SetUpdateIntervalIntent

- **Slot:** Type, Location, Interval.
- **Utterance:** *"Set update interval of office humidity sensor to one hour"*.
- **Modalità dialogo:** *"Set update interval"*.
- **Richiesta conferma:** Sì.

A.2.5 Get

GetValueIntent

- **Slot:** Type, Location.
- **Utterance:** *"Get kitchen temperature"*.
- **Modalità dialogo:** *"Get value"*.
- **Richiesta conferma:** No.

GetStatusIntent

- **Slot:** Type, Location.
- **Utterance:** *"Get status of temperature sensor in the kitchen"*.
- **Modalità dialogo:** *"Get status"*.
- **Richiesta conferma:** No.

GetUpdateIntervalIntent

- **Slot:** Type, Location.
- **Utterance:** *"Get update interval of kitchen humidity sensor"*.
- **Modalità dialogo:** *"Get update interval"*.
- **Richiesta conferma:** No.

GetEnergyIntent

- **Slot:** Type, Location.
- **Utterance:** *"Get energy of office temperature sensor"*.
- **Modalità dialogo:** *"Get energy"*.
- **Richiesta conferma:** No.

GetLastUpdateTimeIntent

- **Slot:** Type, Location.
- **Utterance:** *"Get last update time of office humidity sensor"*.
- **Modalità dialogo:** *"Get last update time"*.
- **Richiesta conferma:** No.

GetAverageIntent

- **Slot:** Type.
- **Utterance:** *"Get average temperature"*.
- **Modalità dialogo:** *"Get average"*.
- **Richiesta conferma:** No.

GetAverageOfLocationIntent

- **Slot:** Type, Location.
- **Utterance:** *"Get average temperature in the office"*.
- **Modalità dialogo:** *"Get average in the office"*.
- **Richiesta conferma:** No.

GetMaxOfLocationIntent

- **Slot:** Type, Location.
- **Utterance:** *"Get max temperature in the kitchen"*.
- **Modalità dialogo:** *"Get maximum"*.
- **Richiesta conferma:** No.

GetMinOfLocationIntent

- **Slot:** Type, Location.
- **Utterance:** *"Get min temperature in the kitchen"*.
- **Modalità dialogo:** *"Get minimum"*.
- **Richiesta conferma:** No.

GetAllStatusIntent

- **Slot:** Nessuno
- **Utterance:** *"Get status of all devices"*.
- **Modalità dialogo:** Non supportata.
- **Richiesta conferma:** No.

GetAllEnergyIntent

- **Slot:** Nessuno
- **Utterance:** *"Get energy of all devices"*.
- **Modalità dialogo:** Non supportata.
- **Richiesta conferma:** No.

A.2.6 Turn on/off

TurnOnOffIntent

- **Slot:** Type, Location, Status.
- **Utterance:** *"Turn on kitchen temperature sensor"*.
- **Modalità dialogo:** *"Turn on"*.
- **Richiesta conferma:** No.

TurnOnOffByTypeIntent

- **Slot:** Type, Status.
- **Utterance:** *"Turn on all temperature sensors"*.
- **Modalità dialogo:** *"Turn on by type"*.
- **Richiesta conferma:** No.

TurnOnOffByLocationIntent

- **Slot:** Location, Status.
- **Utterance:** *"Turn off all devices in office"*.
- **Modalità dialogo:** *"Turn off by location"*.
- **Richiesta conferma:** No.

TurnOnOffAllIntent

- **Slot:** Status.
- **Utterance:** *"Turn on all devices"*.
- **Modalità dialogo:** Non supportata.
- **Richiesta conferma:** No.

Bibliografia

[1] D. Evans, "The internet of things, how the next evolution of the internet is changing everything", Cisco Internet Business Solutions Group (IBSG), 2011.

[2] K. Ashton, "That "Internet of Things" thing", RFiD Journal, 2009.

[3] International Telecommunication Union, "ITU Internet Report 2005: The Internet of Things", International Telecommunication Union, Geneva, 2005.

[4] IEEE - "Special Report: The Internet of Things", 2014.

[5] H. Sundmaeker, P. Guillemin, P. Friess, S. Woelfflé, "Vision and challenges for realising the Internet of Things", CERP-IoT - Cluster of European Research Projects on the Internet of Things, 2010.

[6] L. Atzori, A. Iera, G. Morabito, "The internet of things: a survey", Comp. Netw.: Int. J. Comp. Telecommun. Network. 54 (15) (2010) 2787-2805.

[7] E. Borgia, "The Internet of Things vision: Key features, applications and open issues". Computer Communications 54 (2014): 1-31.

[8] A. Bassi, G. Horn, "Internet of Things in 2020: A Roadmap for the Future", European Commission: Information Society and Media, 2008.

[9] Z. K. Zhang, M. C. Y. Cho, C. W. Wang, C. W. Hsu, C. K. Chen, and S.Shieh, "IoT security: ongoing challenges and research opportunities",

In Service-Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on (pp. 230-234). IEEE.

[10] L. Eschenauer, and V. D. Gligor, "A key-management scheme for distributed sensor networks", In Proceedings of the 9th ACM Conference on Computer and Communications Security (pp. 41-47). ACM.

[11] J. Liu, Y. Xiao, and C. L. P. Chen, "Authentication and Access Control in the Internet of Things", In IEEE 32nd International Conference on Distributed Computing Systems Workshops, June 2012.

[12] D. L. Yang, F. Liu, and Y. D. Liang, "A survey of the internet of things", In Proceedings of the 1st International Conference on E-Business Intelligence (ICEBI2010),. Atlantis Press.

[13] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges", Ad Hoc Networks, 10(7), 1497-1516.

[14] S. Gray, "Always On: Privacy Implications of Microphone-Enabled Devices", In Future of privacy forum. 2016.

[15] S. Soda, M. Nakamura, S. Matsumoto, S. Izumi, H. Kawaguchi, and M. Yoshimoto,"Implementing virtual agent as an interface for smart home voice control", In Software Engineering Conference (APSEC), 2012 19th Asia-Pacific (Vol. 1, pp. 342-345). IEEE. 2012.

[16] M. R. Kamarudin, M. A. F. M. Yusof, and H. T. Jaya, "Low cost smart home automation via microsoft speech recognition", 2013.

[17] J. K. Guo, C. L. Lu, J. Y. Chang, Y. J. Li, Y. C. Huang, F. J. Lu, and C. W. Hsu, "Interactive voice-controller applied to home automation", In Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP'09. Fifth International Conference on (pp. 828-831). IEEE. 2009.

[18] Amazon Alexa - “https://developer.amazon.com/alexa”

[19] Google Assistant - “https://assistant.google.com/”

[20] Apple Siri - “https://www.apple.com/ios/siri/”

[21] SiriKit - “https://developer.apple.com/documentation/sirikit”

[22] Microsoft Cortana - “https://support.microsoft.com/en-us/help/17214/windows-10-what-is”

[23] A. Brown, “The Role of Voice in IoT Applications”, https://www.strategyanalytics.com/, 2015.

[24] M. R. Ebling, “Can cognitive assistants disappear?”, IEEE Pervasive Computing, 15(3), 4-6. 2016.

[25] Alexa Voice Service - “https://developer.amazon.com/alexa-voice-service”

[26] Alexa Skill Kit - “https://developer.amazon.com/alexa-skills-kit”

[27] Understanding Custom Skill - “https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/overviews/understanding-custom-skills”

[28] Understanding the Smart Home Skill - “https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/overviews/understanding-the-smart-home-skill-api”

[29] Enterprise IoT Insights, “https://enterpriseiotinsights.com/20161229/channels/news/20161229channelsnewsibm-delos-health-tag31”

[30] AWS Lambda - “http://docs.aws.amazon.com/lambda/latest/dg/welcome.html”

[31] ThingSpeak - “https://thingspeak.com/”

[32] Arduino Uno - “https://store.arduino.cc/arduino-uno-rev3”

[33] H.D. Linden, “Handbook of Batteries”, 2nd ed., McGrawHill, New York 1995.

[34] Pedregosa et al., “Scikit-learn: Machine Learning in Python”, Journal of Machine Learning Research 12, pp. 2825-2830, 2011.

[35] NumPy - “https://docs.scipy.org/doc/numpy-dev/license.html”

[36] S. Raschka, “Python machine learning”, Packt Publishing Ltd, 2015.