

RELAZIONE DI PROGETTO DI
“SMART CITY E TECNOLOGIE MOBILI”

City Flooding Kit

Numero del gruppo:

66

Componenti del gruppo:

Giulia Lucchi 820828

Marco Canducci 833069

Indice

1	Introduzione	3
2	Stato dell'arte	5
3	Analisi dei requisiti	12
3.1	Glossario	12
3.2	Requisiti Funzionali	12
3.3	Requisiti Non Funzionali	14
3.4	Diagrammi dei Casi d'uso	14
3.5	User Stories	16
4	Progettazione	18
4.1	Pattern architetturali e comunicazione	19
4.1.1	Publish-Subscribe	19
4.1.2	RESTful API	20
4.2	Flooding Kit	20
4.2.1	Raspberry Pi 1 Model B, Revision 2.0	22
4.2.2	Sensore GY-BME/P 280	23
4.2.3	Sensore HC-SR04	25
4.2.4	Sensore FC-37 e YL-38	26
4.2.5	Schema di Montaggio	27
4.3	Stazione d'Allarme	28
4.3.1	Arduino Uno Rev3	29
4.3.2	Schema di Montaggio	30
4.4	Sistema Cloud	31
4.5	Applicazione Web	31
4.5.1	Mockup	32
5	Implementazione	36
5.1	Scelte tecnologiche e Linguaggi di Programmazione	36
5.1.1	AWS - Amazon Web Service	36
5.1.2	MQTT	38
5.1.3	Linguaggi di programmazione	38

5.2	Flooding Kit	38
5.2.1	Creazione della Thing	38
5.2.2	Acquisizione Dati	40
5.2.3	Comunicazione con AWS IoT Core	42
5.2.4	Creazione delle regole di archiviazione	43
5.3	Stazione d'Allarme	45
5.3.1	Creazione della Thing	45
5.3.2	Creazione regole d'allarme	46
5.3.3	Attivazione Allarme	48
5.3.4	Disattivazione Allarme	49
5.4	Applicazione Web	50
6	Testing e performance	52
6.1	Testing	52
6.2	Performance	56
7	Analisi di deployment su larga scala	58
7.1	Cliente singolo, molteplici componenti hardware	59
7.2	Molti clienti, poche componenti hardware ciascuno	60
7.3	Molti clienti, molteplici componenti hardware	60
8	Piano di lavoro	62
9	Conclusioni	65
9.1	Difficoltà incontrate	65
9.2	Sviluppi futuri	66
	Riferimenti bibliografici	67

1 Introduzione

L'inondazione è il tipo di disastro naturale più costoso del mondo. In tutto il mondo in via di sviluppo, le inondazioni possono colpire con regolarità mortale, distruggendo abitazioni, agricoltura e comunicazioni. Le nazioni sviluppate sono difficilmente immuni: le inondazioni che hanno colpito l'Europa nel 2002 costano decine di vite e miliardi di euro.[1] Oltre ai danni economici e sociali, l'alluvione porta anche gravi rischi per la salute umana e dell'ambiente. Infatti, basti pensare all'inondazione di rifiuti e di impianti per il trattamento delle acque reflue o di siti industriali che lavorano con materiale chimico, in quanto potrebbero portare all'uccisione diretta di bestiame e alla pesante contaminazione di acqua e di terreni agricoli [2]. Infine quindi è utile trovare una metodologia per prevenire questi disastri e rendere le persone più responsabili e attente.

Il progetto si colloca nel contesto dell'ideazione di *smart environment* a supporto dei sistemi di prevenzione ed allarme per disastri ambientali riguardanti, in particolare, l'esondazione dei fiumi. Lo Smart Environment è sostanzialmente identificabile con un territorio in cui l'unione di diverse infrastrutture e strumenti di monitoraggio già esistenti o in divenire collaborano per permettere a utenti diversi l'accesso ad una serie di informazioni che riguardano l'ambiente e le sue criticità. Sul versante tecnologico, l'elemento distintivo è rappresentato da una forte spinta evolutiva provocata dalla diffusione di molteplici soluzioni tecnologiche di monitoraggio, attraverso una serie di strumenti e sensori ben distribuiti sul territorio.[3]

L'obiettivo del nostro lavoro è quello di fornire un kit configurabile ad ipotetici enti pubblici, per monitorare in modo efficace le condizioni critiche che precedono un'eventuale situazione di emergenza. Tale progetto appartiene alla categoria degli *early warning systems*: sistemi di allarmistica con cui è possibile ricevere informazioni significative e tempestive in modo sistematico prima di un disastro, al fine di prendere decisioni informate ed eventualmente agire di conseguenza.

Nel dettaglio, il nostro sistema consiste in una sorta di stazione meteo specializzata per lo scopo sopra indicato. Si è visto come possa risultare efficace misurare non solo il livello del fiume, ma anche diversi dati atmosferici, per ottenere soluzioni per l'emissione di segnali d'allarme nella prevenzione delle alluvioni. Pertanto, la

nostra stazione sarà dotata dei seguenti sensori:

- ad ultrasuoni, per monitorare le variazioni del livello d'acqua nel fiume;
- di pioggia;
- di umidità, temperatura e pressione.

Per quanto riguarda il funzionamento del sistema, esso dovrà raccogliere tutti i dati dei sensori col duplice scopo di:

- permettere un'analisi real time per individuare situazioni di pericolo tramite sistemi a soglia configurabili;
- memorizzare uno storico che permetta un'eventuale futura analisi dati, a scopi statistici e/o operativi tramite tecniche più sofisticate.

Infine, dovrà esser presente una semplice interfaccia web che permetta l'accesso sia con un profilo per la configurazione/amministrazione del sistema, sia con uno per il puro monitoraggio che risulterà pubblico a chiunque.

2 Stato dell'arte

Esistono diverse macro-tipologie di soluzioni per quanto riguarda il monitoraggio delle alluvioni, le quali possono venire utilizzate sia singolarmente sia in modo combinato per ottenere risultati estremamente efficaci.

Prima di tutto, le **immagini satellitari** possono aiutare con largo anticipo gli operatori della protezione civile nel prevedere i luoghi più a rischio nei quali i fiumi potrebbero uscire dagli argini. I dati rilevati tramite immagini satellitari possono fornire modelli digitali altamente dettagliati delle aree a rischio, sui quali risulta possibile svolgere delle avanzate simulazioni di alluvioni computerizzate [1] [4]. Ottimi risultati sono stati ottenuti ad esempio elaborando in modo sinergico sia i dati ottenuti dall'osservazione di inondazioni passate, sia le immagini ad alta definizione sempre più dettagliate provenienti dall'***Earth observation systems*** [5].

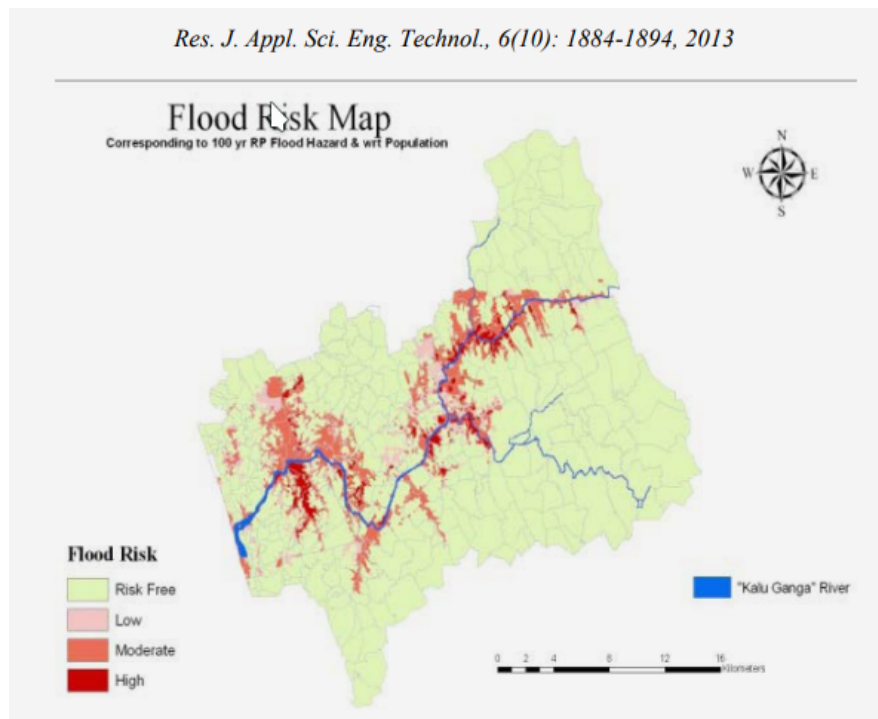


Figura 1: Mappa dei rischi alluvionali ottenuta su un periodo temporale di 100 anni sovrapponendo le mappe di rischio e di vulnerabilità con la mappa poligonale del fiume Kalu Ganga in Sri Lanka [6]

Tra gli anni '90 e ed i primi anni 2000 c'è stata un ampio sviluppo nella ricerca per l'utilizzo di **modelli idrologici** con componenti di previsione alluvionale. Questi modelli sono nella maggior parte dei casi accoppiati con sistemi di rilevazione dati da remoto e *Geographical Information Systems*. [7]

Molto interessante il fatto che sia stato dimostrato come, attraverso la mappatura di aree pericolose, i dati rilevati possano facilmente aiutare non solo la prevenzione, ma anche il monitoraggio e la valutazione dei danni addirittura in real time ad inondazione in atto [8].

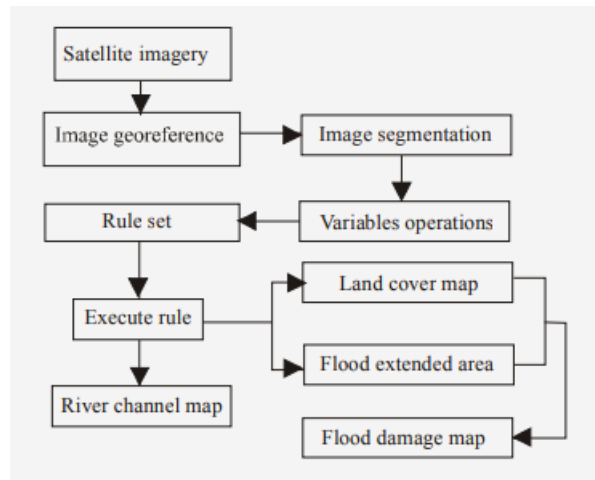


Figura 2: Diagramma di flusso che mostra i passaggi su come i dati di *remote sensing* possano essere elaborati per l'uso nella valutazione del danno da inondazione. [9]

Tralasciando gli ausili satellitari e di modellazione avanzata, si è visto nel panorama delle soluzioni commerciali a minor budget come le misure delle precipitazioni e del livello dell'acqua sui percorsi fluviali, insieme a sistemi di videosorveglianza, possano risultare un ottimo modo per controllare il rischio alluvione. Altri sensori per misurare l'umidità del suolo, la qualità dell'acqua o le condizioni meteorologiche possono aggiungere informazioni preziose [10].



Figura 3: Montaggio in loco del sistema di rilevazione ed allarme targato *Nihon Kasetsu* [10]

Altri esempi di applicazioni reali che hanno catturato la nostra attenzione durante la fase di brainstorming iniziale sono stati quelli di *Libelium* [11], *Flood Beacon* [12] e della stazione **stazione idrometrica** sul Fiume Cherio [13].



Figura 4: Waspote Plug & Sense! Smart Agriculture PRO di Libelium per la misura del livello dell'acqua e dei valori atmosferici [11].

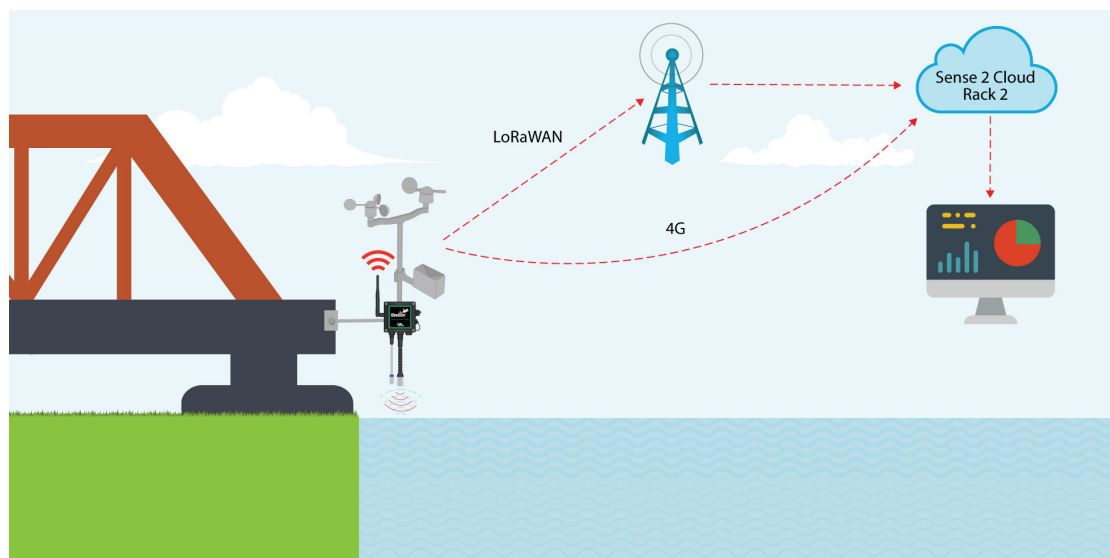


Figura 5: Schema di connessione del progetto Rack2 sviluppato in Argentina [11].

L'Autorità idrica della Provincia di Buenos Aires ha richiesto un sistema di monitoraggio permanente dell'acqua, non solo per le segnalazioni di rischio, ma anche per produrre un flusso dati soddisfacente per **il controllo e l'analisi** della variazione nel tempo del livello del fiume. A tale scopo, sono state installate quattro unità *Waspnote Plug Sense* di Libelium lungo il fiume, dotate dei seguenti sensori:

- sensore ad ultrasuoni
- sensore di temperatura, umidità e pressione
- pluviometro
- anemometro

I dati raccolti dai sensori vengono inviati ad una piattaforma cloud dove possono essere visualizzati e analizzati. La dashboard consente all'utente finale di controllare lo stato dei sensori ogni 30 minuti e ricevere avvisi di allarme nel caso in cui le soglie dei parametri vengano oltrepassate [11].

Altro progetto commerciale interessante ed estremamente particolare è quello del *Flood Beacon* [12].



Figura 6: Deploy di un *Flood Beacon* direttamente sul fiume.

I *Flood Beacon* sono utilizzati in aree ad alto rischio di alluvione come pianure alluvionali, fiumi e laghi. Possono essere ancorati ad un punto oppure lasciati galleggiare seguendo la corrente del fiume. La tecnologia a basso consumo ed il design a energia solare mantengono il Flood Beacon sempre operativo.

Flood Beacon è progettato per trasmettere dati in tempo reale come il livello dell'acqua, la posizione GPS e eventuali accelerazioni improvvise che il beacon potrebbe percepire. I dati vengono inviati al cloud per l'elaborazione e distribuiti tramite una open API gestita da *Xively*. Gli avvisi d'allarme possono quindi essere inviati alla loro applicazione proprietaria per smartphone, a stazioni di monitoraggio o a qualsiasi altro sistema si volesse sviluppare.

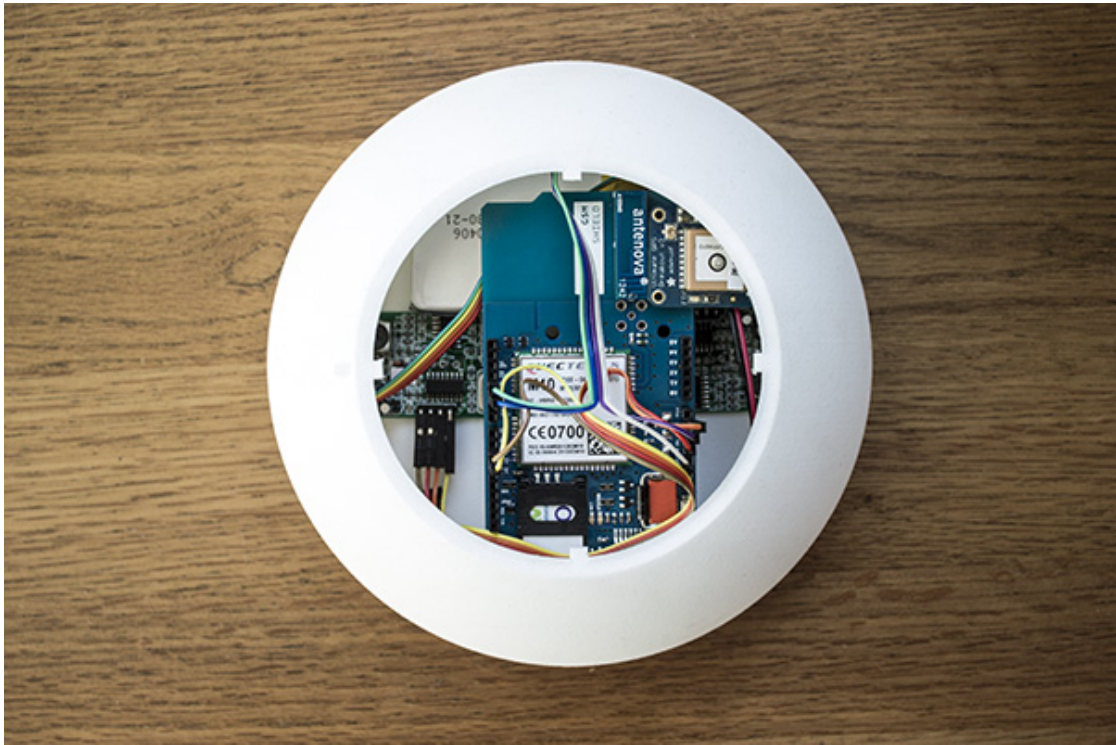


Figura 7: La componentistica interna di un *Flood Beacon*.

3 Analisi dei requisiti

3.1 Glossario

- Un **Flooding kit** è un prodotto fisico contenente una serie di sensori per il monitoraggio del meteo (temperatura, pressione, umidità, pioggia) e del livello dell'acqua nei percorsi fluviali (misurandone la variazione grazie ad un sensore ad ultrasuoni).
- Una **Stazione d'allarme** è un dispositivo fisico sul quale sono installati un pulsante di stop ed uno o più attuatori utili per segnalare uno stato di allarme. In un contesto reale tali attuatori potrebbero consistere in sirene, lampeggianti o dispositivi telematici in grado di informare in modo automatico gli enti predisposti. Nel prototipo che si intende realizzare si ritiene opportuno segnalare l'allarme tramite l'accensione di un led.
- Si entra nello **stato di allarme** quando viene innescata una regola di allarme. È possibile annullare lo stato d'allarme premendo il pulsante di stop sulla stazione d'allarme.
- Una **regola d'allarme** consiste tipicamente in un controllo a soglia configurabile, automaticamente effettuato in tempo reale sulle rilevazioni sensoriali. Allo scattare di almeno una regola di allarme riguardante un qualsiasi Flooding kit viene indetto lo stato di allarme.

3.2 Requisiti Funzionali

- L'utente dell'applicazione viene tipicamente munito di una Stazione d'allarme ed alcuni Flooding kit da posizionare lungo le zone più critiche del percorso fluviale che si desidera monitorare.
- Ogni kit deve essere geo-localizzato al momento dell'installazione assegnandogli manualmente le opportune coordinate GPS.
- Ogni kit deve essere saldamente montato sopra al percorso fluviale, con il sensore ad ultrasuoni rivolto verso il basso, perpendicolarmente alla superficie dell'acqua, ed il sensore per la rilevazione della pioggia rivolto verso l'alto.

- Ogni kit deve essere connesso alla rete Internet in modo da poter trasmettere in tempo reale i dati rilevati dai suoi sensori.
- Il sistema deve archiviare tutti i dati ricevuti dai sensori ad una certa frequenza con lo scopo di renderli disponibili per una futura analisi con tecniche di mining.
- Anche la stazione d'allarme deve essere connessa alla rete Internet, sia per essere notificata allo scattare di una regola d'allarme, sia per comunicare la cessazione dello stato di allarme in seguito alla pressione del comando di stop.
- La stazione d'allarme può essere posizionata ovunque, ricordando di rendere facilmente accessibile il suo pulsante di stop da parte di un apposito operatore.
- Il sistema deve essere accessibile tramite due diverse interfacce web: una per il monitoraggio ad accesso pubblico ed una di configurazione accessibile solo dal personale amministrativo.
- Dall'applicazione web pubblica si deve poter:
 - visualizzare su una mappa i kit installati sul territorio
 - visualizzare, per ogni kit, uno storico con tutte le informazioni ricavate dai suoi sensori
 - verificare se è in corso o meno uno stato di allarme
- Dall'interfaccia web di amministrazione si deve poter:
 - creare, modificare e cancellare regole di allarme
 - inserire le coordinate di un nuovo kit al momento della sua installazione
- La definizione di una regola d'allarme ha le seguenti possibilità:
 - per ogni sensore, per ogni kit, può esser definito un range di valori al di fuori del quale verrà innescato l'allarme
 - possono essere definite regole d'allarme che prevedano l'attivazione solamente nel caso i valori di più sensori escano contemporaneamente da determinati range di valori.

3.3 Requisiti Non Funzionali

- *Scalabilità*: il sistema nativamente è pensato per poter funzionare a livello cittadino, ma deve essere predisposto fin da subito per supportarne l'utilizzo su scala regionale o nazionale.
- *Robustezza*: il sistema deve prevedere una situazione di non disponibilità momentanea della Stazione d'allarme a causa di lavori di manutenzione o di interruzione della connessione Internet. Al ritorno *online* della stazione il suo stato deve rispecchiare quello del sistema, facendo scattare immediatamente i suoi attuatori nel caso si fosse entrati in uno stato di allarme durante la disconnessione.
- *Minimizzazione del rumore* proveniente dalle rilevazioni dei sensori.

3.4 Diagrammi dei Casi d'uso

I diagrammi dei casi d'uso vanno a riassumere tutti i requisiti funzionali sopra descritti, con lo scopo di rendere consultabile, in modo più rapido, il comportamento del sistema nel suo complesso.

Negli schemi sottostanti si vanno a rappresentare in modo conciso tutte le funzionalità offerte dall'applicazione web, dal Flooding kit e dalla Stazione d'allarme, facendo una precisa separazione tra le mansioni riservate agli utenti dotati di account amministrativo e quelle di un utente qualunque.

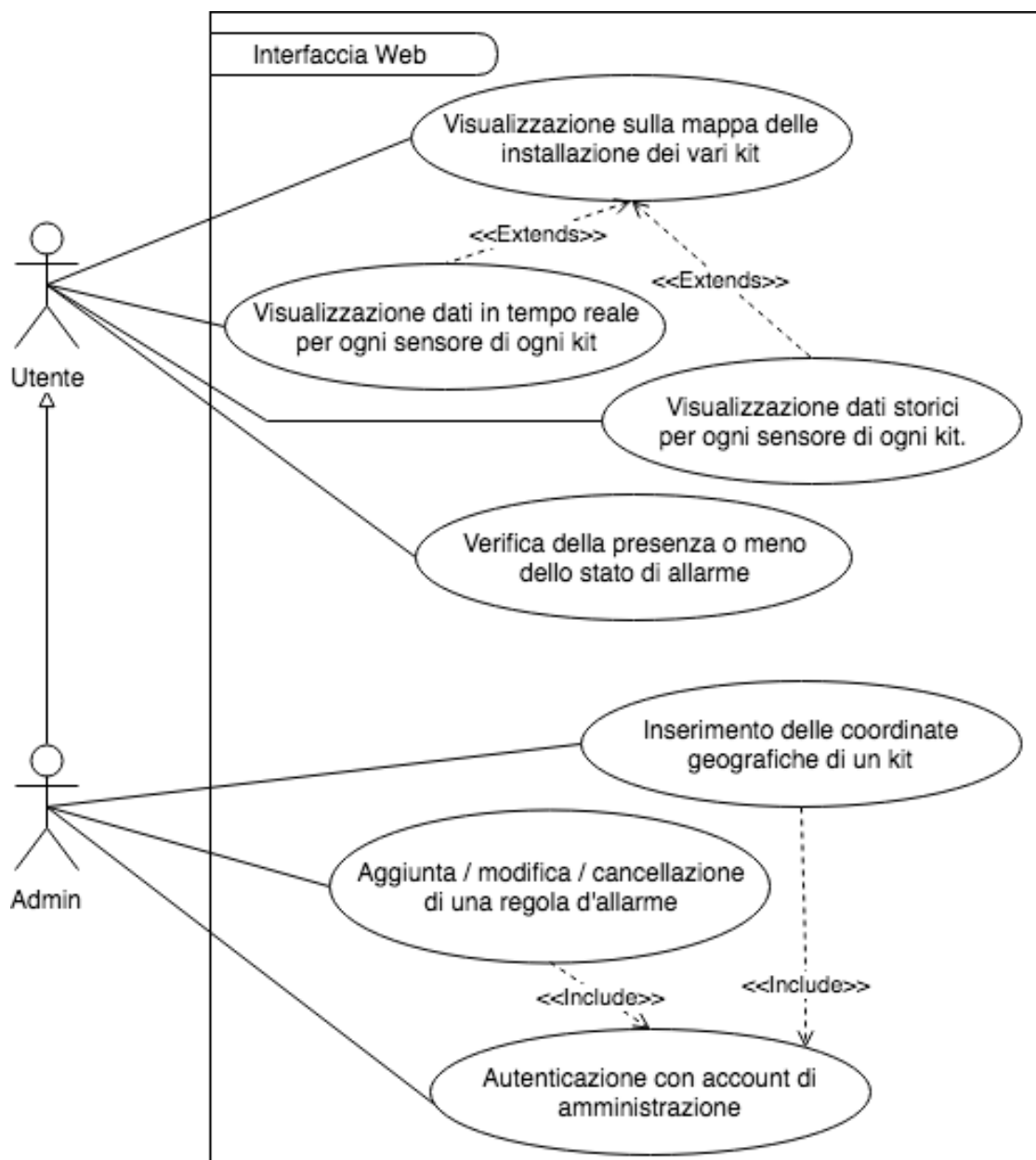


Figura 8: Diagramma dei casi d'uso per l'Applicazione Web

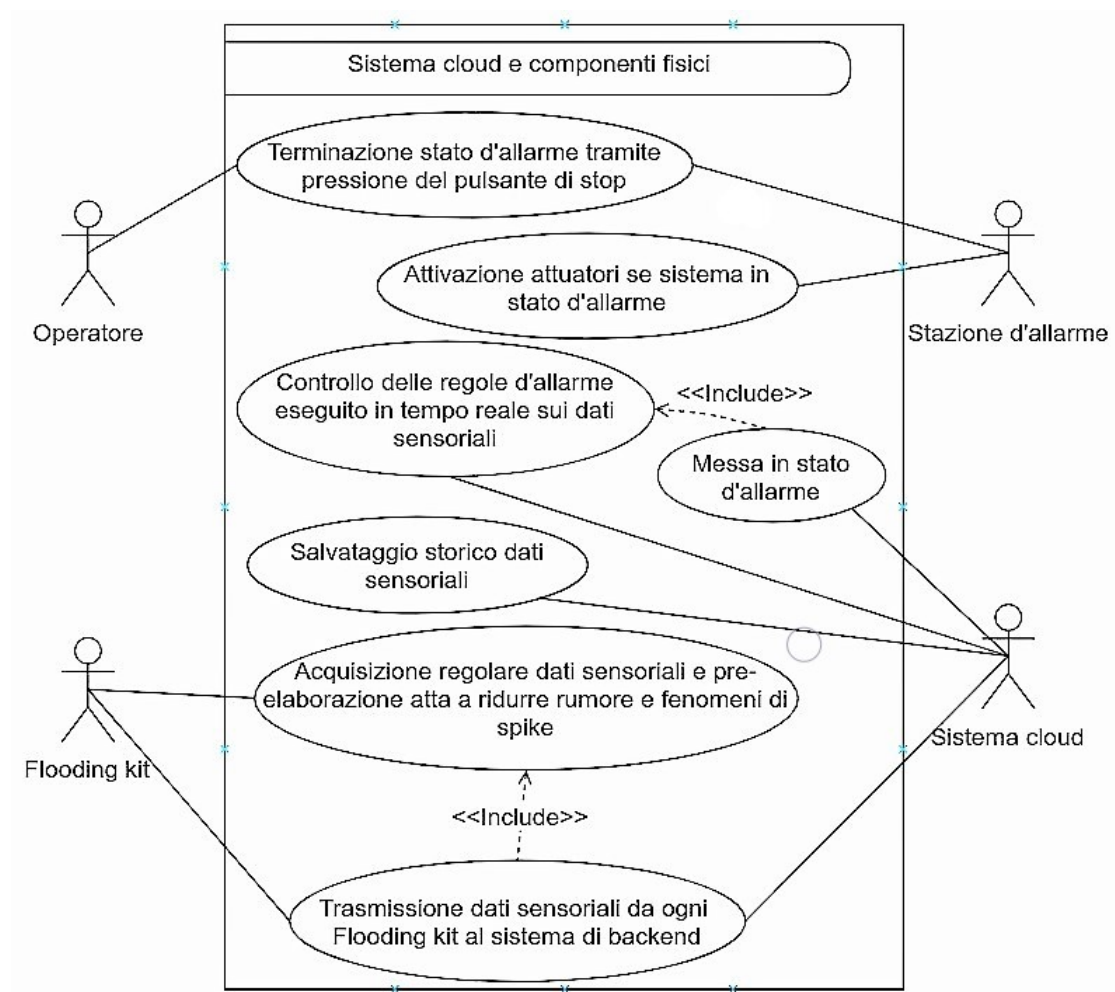


Figura 9: Diagramma dei casi d'uso per il sistema di backend ed i dispositivi fisici

3.5 User Stories

Installazione e configurazione di un "Flooding Kit"

- In veste di amministratore, mi reco in una sezione di fiume che si vuole monitorare.
- Installo un *Flooding kit* indirizzando il sensore ultrasuoni verso la superficie dell'acqua in modo perpendicolare.
- Accedo all'account di amministrazione ed imposto le coordinate geografiche del kit appena installato

- Imposto le regole d'allarme per il sensore ad ultrasuoni, definendo come valore minimo di distanza 3 metri e valore massimo di distanza 15 metri.
- Imposto ora una regola di allarme definita da una correlazione di più sensori: l'allarme dovrà scattare nel caso il sensore di prossimità sia inferiore di 5 mt e sia rilevata la presenza di pioggia.
- Accedo all'applicazione dall'interfaccia pubblica e mi assicuro sia che il kit appena installato compaia sulla mappa nella giusta posizione, sia che i sensori trasmettano correttamente i dati atmosferici e di prossimità.

Segnalazione allarme

- Il sistema monitora i sensori installati nel kit
- Il sistema rileva che la distanza del livello dell'acqua è 18 mt, maggiore del limite massimo di distanza inserito
- Il sistema invia un messaggio d'allarme alla "stazione d'allarme"
- La stazione d'allarme avvisa i cittadini limitrofi tramite una sirena e fa partire una chiamata automatica agli enti addetti

Navigazione dell'applicazione web pubblica

- Come utente generico, mi collego al portale dell'applicazione web e mi accerto che non sia segnalata una situazione d'allarme
- Seleziono sulla mappa un kit di mio interesse
- Visualizzo i valori correnti rilevati da tutti i sensori installati su quel kit
- Visualizzo i valori storici relativi all'andamento del livello dell'acqua negli ultimi due giorni
- chiudo l'applicazione

4 Progettazione

In questo capitolo, andremo a definire le scelte progettuali e l'architettura di base, andando ad individuare tutte le principali entità del nostro sistema.

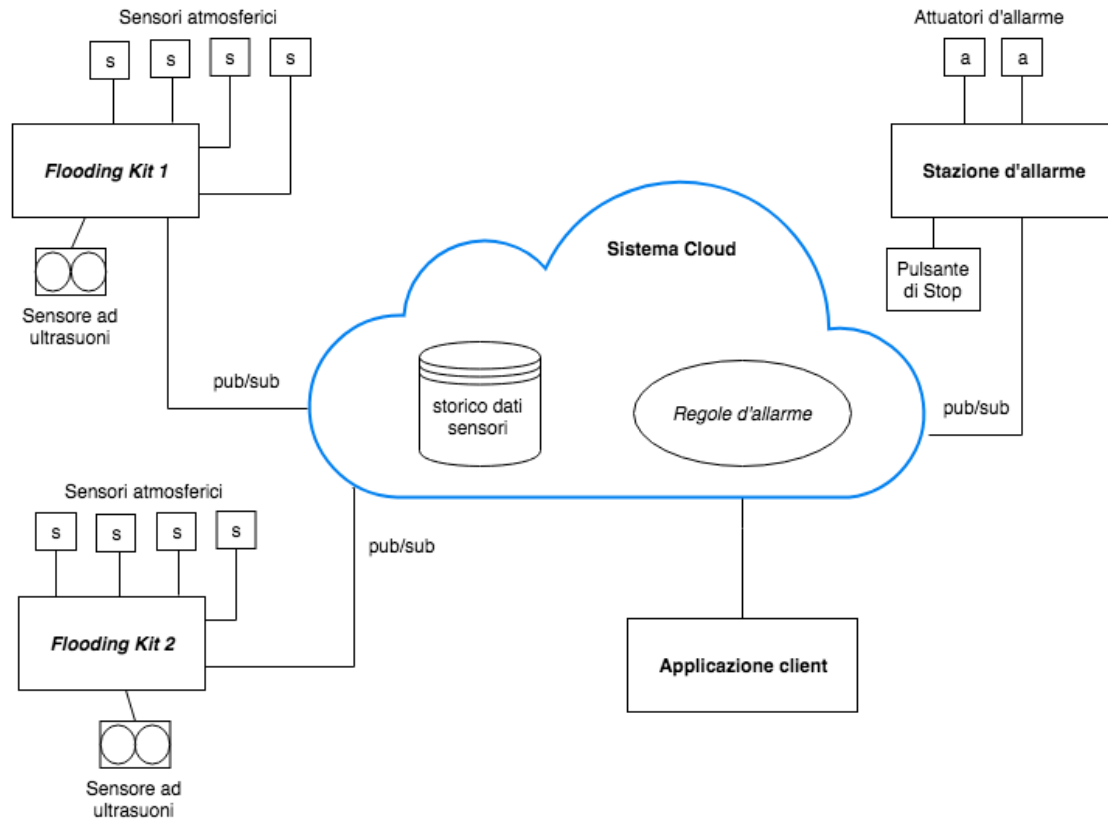


Figura 10: Architettura di base - entità principali

La figura mostra come la progettazione del sistema si basi sulla presenza di quattro entità principali che andranno esposte dettagliatamente nei sottocapitoli seguenti. Esse sono:

- **Flooding Kit:** il dispositivo fisico in grado di misurare i parametri richiesti dal sistema
- **Stazione d'allarme:** il dispositivo fisico in grado di verificare la presenza o meno di situazioni d'emergenza
- **Sistema Cloud:** la parte dell'architettura che gestisce il core del sistema

- **Applicazione Web:** l'applicazione che permette agli utenti di monitorare lo stato, attuale e storico, del Flooding Kit.

Infine, si evince che il principale pattern architetturale utilizzato per la comunicazione nel nostro sistema è di tipo Publish-Subscribe, la cui scelta è ampiamente descritta in seguito.

4.1 Pattern architetturali e comunicazione

In questa sezione andremo a definire i principali pattern architetturali utilizzati, la cui scelta è stata pensata sullo studio dei requisiti fatto precedentemente.

4.1.1 Publish-Subscribe

Il pattern architetturale *Publish-Subscribe* è un modello di messaggistica asincrona fra diversi processi e/o oggetti. I mittenti e i destinatari dei messaggi dialogano attraverso un broker che fa da intermediario. Il mittente di un messaggio, definito “publisher”, non deve essere consapevole dell’identità dei destinatari, nominati “subscriber”, ma si limita a pubblicare su un canale, chiamato “topic”, le proprie informazioni. Il subscriber quindi per ricevere i messaggi si sottoscriverà al topic di proprio interesse.

La scelta su questo tipo di modello di comunicazione è nata dalla natura stessa del progetto: creare un sistema distribuito in grado di acquisire dati in “near real-time” e di gestire una grande mole di messaggi. Questo tipo di approccio infatti si adatta facilmente all’ambiente IoT grazie anche alla buona scalabilità tipica di questo modello e all’astrazione da vincoli tecnologici e da problemi di integrazione. L’IoT si basa infatti sulla possibilità di interconnettere e integrare un numero molto elevato di dispositivi, piattaforme e software appartenenti a un vasto insieme di tecnologie eterogenee e in continua evoluzione.

Un’ulteriore caratteristica che abbiamo preso in considerazione è il disaccoppiamento. In una buona progettazione infatti, lo scopo è quello di massimizzare il disaccoppiamento, così da rendere facile la sostituzione o la modifica dei moduli del sistema durante la fase di manutenzione, in quanto [14]:

- le parti che interagiscono non hanno bisogno di conoscersi a vicenda,

- i partecipanti per comunicare non hanno bisogno di essere attivi nello stesso momento,
- non c'è la necessità di sincronizzazione, infatti la produzione e il consumo dei messaggi non avviene nel flusso di controllo principale del publisher e del subscriber.

Nel sistema in questione il modello descritto viene utilizzato in particolare per gestire:

- la comunicazione fra il Flooding Kit e il sistema Cloud per quanto riguarda le informazioni provenienti dai sensori
- l'invio degli allarmi e della causa di questi alla Stazione d'allarme e all'Applicazione Web

4.1.2 RESTful API

REST, *REpresentational State Transfer*, è uno stile architetturale per la progettazione di applicazioni in rete. L'idea è quella di utilizzare il protocollo HTTP per gestire richieste ed effettuare chiamate tra due punti. Le applicazioni basate su REST, si definiscono RESTful e utilizzano le richieste HTTP per inviare, modificare, cancellare e gestire i dati e per effettuare query.

La scelta di questo approccio è stata conseguente alla decisione di utilizzare una piattaforma cloud per lo sviluppo del core dell'applicazione, in quanto numerose API legate allo sviluppo cloud risultano di tipo RESTful. Inoltre sono stati presi in considerazione i vantaggi che possono apportare. Le RESTful API, infatti, permettono una totale separazione fra il client e il server, così da portare una maggior flessibilità per quanto riguarda un'eventuale manutenzione o modifica del sistema, ma anche una forte indipendenza dal tipo di piattaforma e dal linguaggio scelto.

4.2 Flooding Kit

Il Flooding Kit rappresenta la parte distribuita utilizzata dall'ente o dal cliente che lo acquista. All'interno del sistema, esso è il prodotto fisico munito di sensori per il monitoraggio dei fiumi con l'obiettivo di evitare il rischio d'alluvione. Esso consiste

in un dispositivo in grado di raccogliere, gestire e inviare tutte le informazioni provenienti dai sensori. La tipologia di quest'ultimi è fissa e coerente con lo scopo del sistema. Per questo, sono stati montati sul kit i seguenti sensori:

- *Sensori Atmosferici*: dopo un'attenta analisi del dominio abbiamo deciso di inserire sensori che potessero misurare temperatura, pressione barometrica, umidità e l'assenza o presenza di pioggia.
- *Sensore ad Ultrasuoni*: non misura nessun tipo di agente atmosferico, ma il suo scopo è quello di monitorare il livello dell'acqua, in quanto, come definito nei requisiti, il Flooding Kit viene posizionato sopra un fiume che potrebbe esondare e provocare un'eventuale alluvione.

Il dispositivo nativamente ha il solo compito di:

- Raccogliere i dati provenienti dai sensori posti sul dispositivo in un certo momento creando così un unico messaggio.
- Comunicare le informazioni con la piattaforma cloud posta al centro del sistema. Per fare questo utilizza il "Publish-Subscribe", il pattern architetturale sopra descritto.
- Minimizzare il rumore della misura dei sensori, in particolare per quanto riguarda il sensore ad ultrasuoni che risulta essere più sensibile ad errori di misura dovuti da cause esterne ad esso. Abbiamo deciso quindi di fare questo come media su finestra mobile di valori direttamente dal dispositivo, anziché su sistema Cloud, per effettuare meno chiamate online e ottimizzare i tempi e gli spazi.

Inoltre all'interno del sistema, come è ben visibile dalla figura precedente, possiamo trovare più Flooding Kit fisici, in quanto è possibile installare più kit distribuiti sul territorio. Per questo ad ogni Flooding Kit deve essere assegnato un identificatore univoco.

A livello fisico, il dispositivo utilizza come SoC (*System on Chip*) un "Raspberry Pi 1 Model B, Revision 2.0".

4.2.1 Raspberry Pi 1 Model B, Revision 2.0

Il Raspberry Pi 1 Model B, Revision 2.0 è un modello non molto recente della casa madre inglese Raspberry Pi Foundation

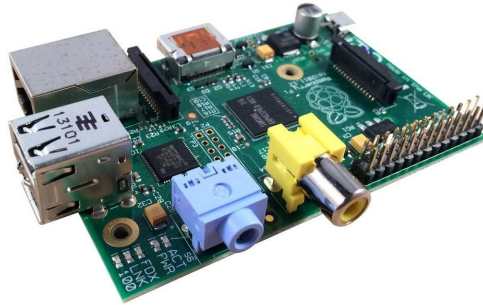


Figura 11: Raspberry Pi 1 Model B, Revision 2.0

Esso è dotato di:

- 512 MB di RAM
- porta Ethernet
- due porte USB
- due opzioni di uscita video: HDMI o composito
- jack di uscita audio da 3,5 mm
- testata GPIO a 26 pin con pin maschio, con il quale è possibile connettere una vasta gamma di sensori

Il Raspberry Pi necessita di un sistema operativo. In questo caso si è scelto di installare come sistema operativo l'ultima versione di Raspbian.

Essendo il progetto relativo all'ambito Smart City e di conseguenza anche all'ambito IoT, una delle necessità deve essere il basso costo nella realizzazione del dispositivo stesso. Per questo, ci siamo trovati di fronte ad una scelta fra due dispositivi embedded a basso costo di nostra accessibilità: Arduino e Raspberry Pi. La scelta è ricaduta sul secondo in quanto Arduino Uno non possiede una libreria crittografica, o abbastanza memoria per essere in grado di trasmettere le richieste crittografate tramite chiave privata. È stato quindi necessario utilizzare un dispositivo più potente, come appunto il Raspberry Pi.

4.2.2 Sensore GY-BME/P 280

Il sensore di Pressione GY-BME/P 280 è un sensore digitale in grado di misurare la pressione barometrica, l'umidità e la temperatura.

Il modulo può misurare secondo le seguenti caratteristiche:

- La temperatura può variare in un intervallo da -40° a 80°C . La precisione è di $\pm 1.0^{\circ}\text{C}$ è ottenuta nell'intervallo da 0° a 65°C . Al di fuori di tale intervallo, la precisione può diminuire fino a $\pm 1.5^{\circ}\text{C}$.
- L'umidità può essere misurata nell'intervallo da 0 a 100% con una precisione di $\pm 3\%$. Il sensore può misurare fino al 100% di umidità nell'intervallo di temperatura da 0° a 60°C . A temperature molto alte o basse, l'umidità misurabile massima diminuisce.
- La pressione ha un intervallo da 300 a 1100 mPa con una precisione di $\pm 1.0\text{ mPa}$. La massima precisione si ottiene nel intervallo di temperatura da 0° a 65°C .

GY-BME/P 280 si presenta come modulo a 6 pin progettato per un funzionamento a 3.3 V. I pin hanno le seguenti notazioni:

- VCC: 1.7 to 3.6V
- GND: ground
- SCL: clock (SCL / SCK) for I2C
- SDA: data (SDA / SDI) for I2C
- CSB: chip Select Bus. The default logic HIGH for I2C
- SDO: sets I2C address for I2C.

Come si può vedere dalle notazioni dei pin, il modulo supporta il protocollo I2C. Infatti il segnale in uscita è disponibile tramite bus I2C [15].

Protocollo I2C

I2C, Inter Integrated Circuit, è un protocollo di comunicazione seriale creato e sviluppato dalla Philips nel 1982. Esso è un sistema di comunicazione seriale bifilare utilizzato tra circuiti integrati. Come dice il nome stesso, si tratta di un meccanismo in grado di far comunicare diversi circuiti integrati che risiedono sullo stesso circuito stampato. Tipicamente, gli integrati che convivono su una scheda elettronica devono scambiarsi informazioni di controllo senza particolari requisiti di velocità di risposta. Per questi usi, l'adozione di un canale seriale condiviso come I2C permette di limitare notevolmente il numero di segnali elettrici da filare sullo stampato rispetto all'approccio in voga precedentemente, secondo il quale ogni integrato aveva un bus indirizzi, un bus dati e un chip select dedicato.[17]

Il protocollo permette la comunicazione di dati tra due o più dispositivi I2C utilizzando un bus a due linee bidirezionali, in cui le informazioni sono inviate serialmente :

- SDA (Serial Data): linea per i dati
- SCL (Serial Clock): linea per il clock

Oltre alle due linee principali, è presente una terza linea: la massa, comune a tutti i dispositivi. Inoltre, il bus ha due tipi di nodi: master e slave. Il nodo master è semplicemente il dispositivo che controlla il bus in un certo istante; tale dispositivo controlla il segnale di Clock e genera i segnali di *start* e di *stop*. I dispositivi slave semplicemente stanno in ascolto sul bus ricevendo dati dal master o inviandone qualora questo ne faccia loro richiesta. Nel bus possono essere presenti più dispositivi che possono svolgere la funzione master in modo mutuamente esclusivo. Tipicamente però viene usato un approccio avente un singolo master.

4.2.3 Sensore HC-SR04

Il sensore ad ultrasuoni sfrutta il concetto di suono: fenomeno causato dall'oscillazione infinitesima degli atomi presenti nel fluido o nel materiale che ci circonda che causa vibrazione; l'oscillazione provoca la propagazione di onde di pressione nell'ambiente circostante. Quindi la tecnologia ad ultrasuoni fornisce anche un metodo semplice per la misurazione di distanze [19].

Il sensore HC-SR04 è costituito da una scheda che presenta nella sua parte posteriore un sofisticato circuito elettronico e nella parte anteriore sono presenti un quarzo e due cilindri metallici, i trasduttori ad ultrasuoni. Per misurare la distanza, uno dei due trasduttori invia ultrasuoni che rimbalzano contro ad un qualunque oggetto posto di fronte ad esso, ed entrano di ritorno nell'altro cilindro. Per fare questo l' HC-SR04 presenta 4 pin, fra cui TRIG che attiva il sensore per inviare il suono ed ECHO, che riceve l'impulso di ritorno. [20]

A livello tecnico, le caratteristiche principali sono le seguenti [21]:

- Tensione di lavoro: 5 Vdc
- Corrente assorbita: >2 mA
- Frequenza di lavoro: 40 Khz
- Distanza max: 450 cm
- Distanza min: 2 cm
- Risoluzione: 3 mm
- Angolo di misura: 15°
- Ingresso: Trigger 10 us Impulso TTL
- Uscita: Echo segnale PWM TTL
- Dimensioni: 45,5 x 20,5 x 15,3 mm

4.2.4 Sensore FC-37 e YL-38

Il sensore di pioggia è uno strumento semplice per il rilevamento della pioggia. Può essere usato per verificare l'assenza o presenza di pioggia attraverso il pannello apposito e per misurare l'intensità delle precipitazioni.

Il sensore è composto da due moduli:

- *modulo FC-37*: la scheda del collettore, che consiste in un pannello che rileva o meno la presenza dell'acqua
- *modulo YL-38*: la scheda elettronica di controllo che offre l'opportunità di regolare la sensibilità con il potenziometro

Il modulo YL-38 è un modulo a 4 pin:

- VCC: 5V
- GRD: terra
- DO: output digitale
- AO: output analogico

A seconda del grado di umidità, la resistenza cambia sul pin analogico ed il valore in uscita sul pin digitale può essere 0 o 1.

La mancanza di un ADC interno al Raspberry Pi ci ha portati ad utilizzare solamente l'output digitale, in quanto non avevamo a disposizione un circuito elettronico in grado di convertire un segnale analogico con andamento continuo in una serie di valori discreti.

4.2.5 Schema di Montaggio

In seguito andremo a presentare la schematizzazione delle componenti con le relative connessioni del Flooding Kit descritto.

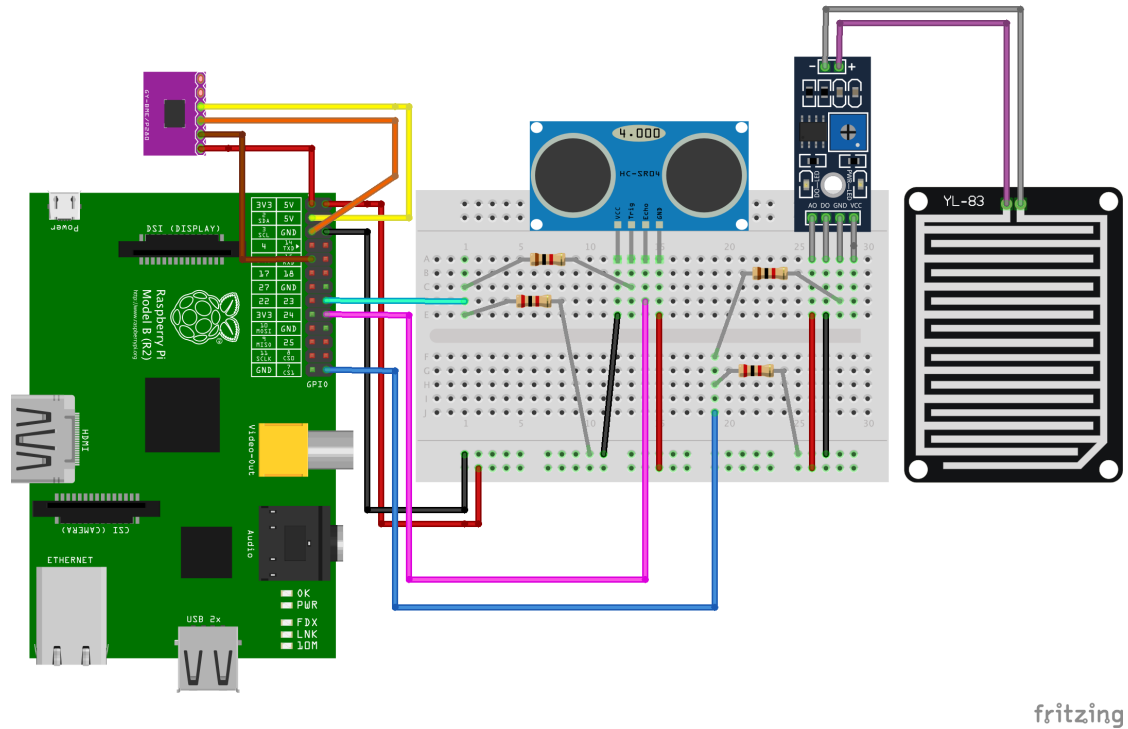


Figura 12: Circuito del Flooding Kit

Da notare la presenza sulla breadboard di due partitori di tensione identici: per ognuno di essi viene sfruttata una resistenza da $2K\Omega$ ed una da $1K\Omega$ per portare da 5V a 3.3V il segnale in ingresso ai pin del Raspberry, in quanto la versione del Raspberry utilizzata non supportava ancora il segnale in ingresso sui pin GPIO a 5V.

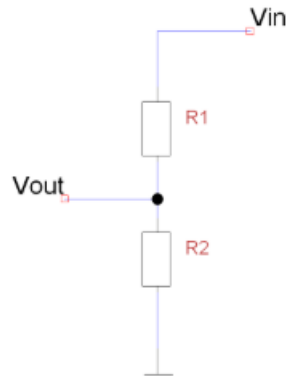


Figura 13: Partitore di tensione utilizzato con $R1 = 1K\Omega$ ed $R2 = 2K\Omega$

4.3 Stazione d'Allarme

La Stazione d'Allarme consiste in un dispositivo fisico creato con lo scopo di avvertire la città/paese dell'emergenza in atto.

Questa unità fisica ha due semplici funzionalità principali:

- Rimanere in ascolto fino a che non riceve un avviso d'allarme. Quest'ultimo verrà mostrato attraverso un led luminoso che si accende.
- Segnalare la fine della situazione d'emergenza tramite un semplicissimo pulsante che spegne il led e modifica lo stato del Flooding Kit, riflettendolo poi anche sull'applicazione Web.

La stazione d'allarme è pensata come un'unità semplice, senza nessun tipo di complessa capacità computazionale. Per questo abbiamo deciso di utilizzare l'Arduino per creare la nostra stazione. L'Arduino, infatti, si presta bene per realizzare applicazioni che riguardano il pilotaggio di sensori ed attuatori senza particolari pretese. La decisione di utilizzare Arduino piuttosto che un secondo Raspberry è derivata poi sia dal fatto che volessimo sperimentare con tecnologie diverse, sia dal fatto che ne avessimo già uno a nostra disposizione.

4.3.1 Arduino Uno Rev3

Arduino Uno è una scheda di microcontrollore basata su ATmega328P. Dispone di 14 pin di input/output digitali, 6 ingressi analogici, un cristallo di quarzo 16 MHz, una connessione USB, un jack di alimentazione, un'intestazione ICSP e un pulsante di ripristino.



Figura 14: Arduino Uno Rev3

Le specifiche tecniche sono le seguenti:

- Tensione di funzionamento: 5V
- Tensione di Alimentazione (raccomandata): 7-12V Massima
- Tensione supportata (non raccomandata): 20V
- Corrente in uscita per I/O Pin: 40 mA
- Corrente in uscita per 3.3V Pin: 50 mA
- Memoria Flash: 32 KB (ATmega328) di cui 0.5 KB usata bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Velocità di clock: 16 MHz

Infine, Arduino semplifica anche il processo di utilizzo dei microcontrollori, ma offre alcuni vantaggi interessanti rispetto ad altri sistemi [22]:

- *Economico*
- *Cross-platform* in quanto il software Arduino (IDE) funziona su sistemi operativi Windows, Macintosh OSX e Linux.
- *Ambiente di programmazione semplice e chiaro* poiché risulta facile da usare per i principianti, ma abbastanza flessibile da consentire agli utenti avanzati di trarne vantaggio
- *Software open source ed estensibile*
- *Hardware open source ed estensibile* in quanto i piani delle schede Arduino sono pubblicati sotto una licenza Creative Commons, quindi i progettisti di circuiti esperti possono creare la propria versione del modulo, estenderlo e migliorarlo

4.3.2 Schema di Montaggio

In seguito andremo a presentare la schematizzazione delle componenti con le relative connessioni della stazione d'allarme descritta.

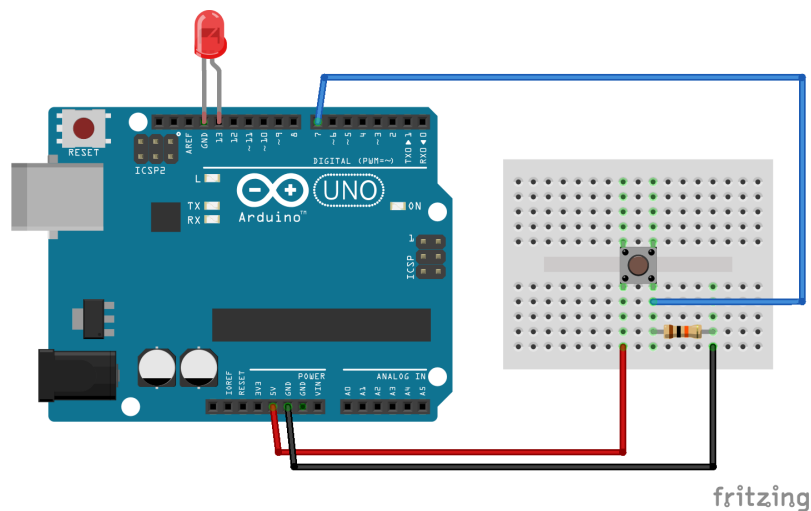


Figura 15: Circuito della stazione d'allarme

4.4 Sistema Cloud

Per tutta la gestione del nostro sistema lato back end abbiamo deciso di approcciarci al *cloud computing*. Il cloud computing è un modello per l' erogazione di servizi offerti *on demand* da un fornitore ad un cliente finale attraverso la rete Internet, a partire da un insieme di risorse preesistenti, configurabili e disponibili in remoto sotto forma di architettura distribuita.

Nel nostro sistema, il compito demandato ai servizi cloud sono i seguenti:

- configurazione e gestione dei dispositivi IoT, quali Flooding Kit e la Stazione d'allarme
- creazione di regole utili per monitorare l'ambiente tramite i sensori del Flooding Kit e per segnalare in caso di situazione d'emergenza
- archiviazione dei dati provenienti dai sensori
- hosting dell'applicazione web

Ci sono vari tipi di servizi cloud, ognuno dei quali offre diversi livelli di controllo, flessibilità e gestione. In particolare, per ogni funzionalità che abbiamo demandato al sistema cloud abbiamo usato tipologie di servizi diversi come segue:

- *SaaS - Software as a Service* per la configurazione dei dispositivi fisici e la creazione di regole per eventuali allarmi
- *PaaS - Platform as a Service* per l'hosting dell'applicazione web
- *DaaS - Data as a Service* per immagazzinare i dati dei sensori

4.5 Applicazione Web

Il sistema, come si evince dai requisiti già esposti precedentemente, deve essere accessibile pubblicamente tramite una semplice applicazione web.

L'obiettivo di questa applicazione consiste nel poter rendere partecipe un utente qualunque al monitoraggio ed alla gestione dell'ambiente. Di conseguenza, si è deciso di creare un'applicazione Web il più chiara, lineare e user-friendly possibile.

L'applicazione deve:

- Permettere la visualizzazione dello storico di un determinato kit, ma anche dei valori correnti e la posizione del kit stesso su una mappa apposita. Di conseguenza deve comunicare con il sistema Cloud in modo da riuscire a ricavare le informazioni archiviate nel database tramite RESTful API.
- Visualizzare in modo chiaro e funzionale la segnalazione di un allarme con la relativa causa. Per fare questo è necessario fare polling per verificare lo stato dei Flooding Kit installati
- Disattivare l'allarme quando l'ambiente intorno è ritornato ad una situazione di normalità, comunicando con la Stazione d'Allarme, che ha il compito di disattivare l'allarme scattato tramite l'apposito pulsante.

4.5.1 Mockup

Il mockup è una rappresentazione statica e prototipale del prodotto finale. Abbiamo deciso di utilizzare questa tecnica per riuscire a dare chiarezza e linearità alla nostra Applicazione Web in modo da poter accordarci sui vari dettagli riguardanti la rappresentazione grafica ed i contenuti. Per far questo è stato usato *Balsamiq Wireframes*: un potente strumento di prototipaggio digitale.

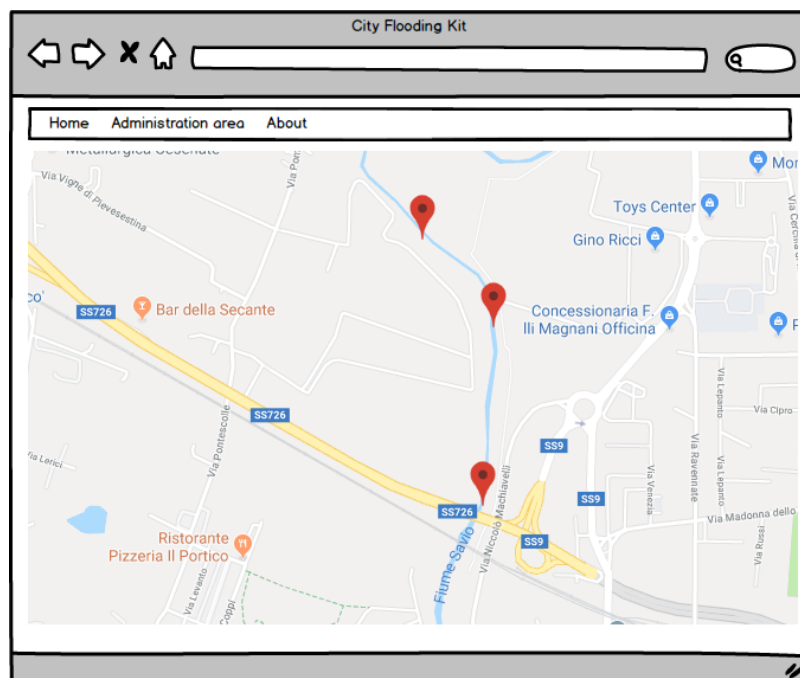


Figura 16: Application Web - home

Home

La home è composta principalmente da una cartina che mette in evidenza, in modo intuitivo e semplice, la presenza dei kit sul territorio. Ognuno di questi risulta cliccabile, indirizzandoci verso la schermata di dettaglio del kit selezionato. Infine è presente un menù in modo da poter facilmente navigare il sito e che rimarrà uguale in tutte le pagine dell'applicazione.

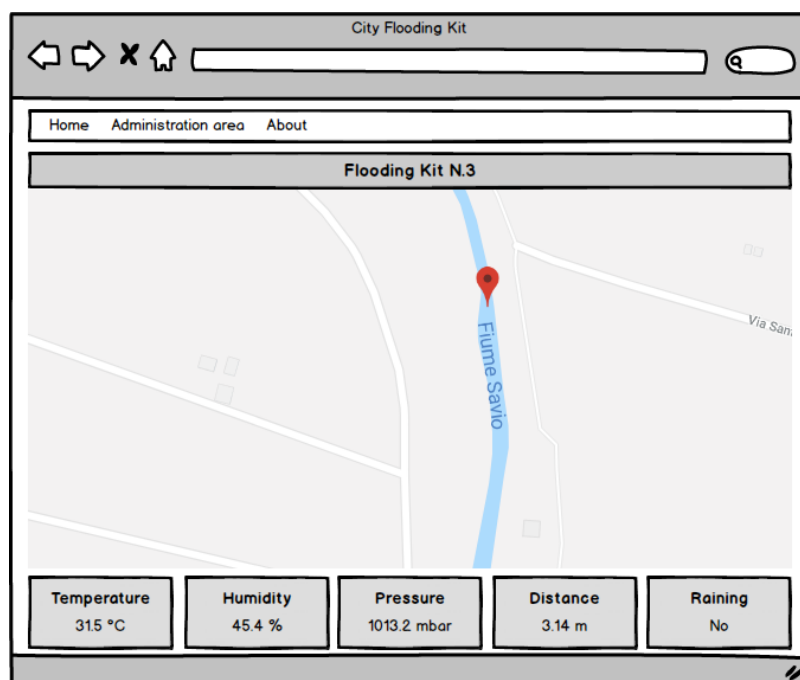


Figura 17: Application Web - kit specifico

Schermata per kit specifico

La pagina web sopra raffigurata è composta principalmente dalla cartina focalizzata sul kit selezionato e contiene tutti i dati di dettaglio per lo specifico kit. In particolare vengono mostrati la posizione ed i valori correnti di un particolare kit. Nella parte bassa troviamo dei box contenenti l'oggetto di misurazione del sensore; cliccando sul nome nel box si aprirà una schermata riportata nella figura sottostante d'esempio.

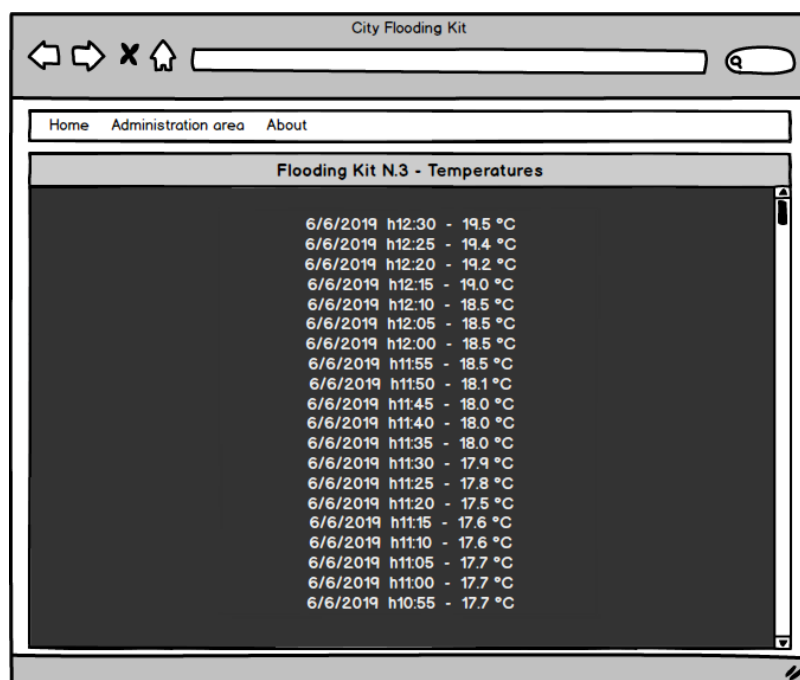


Figura 18: Application Web - history

Note

Abbiamo deciso di non progettare la sezione amministrativa del sito perché probabilmente dipendente dalle tecnologie che avremmo deciso di utilizzare durante la fase di implementazione vera e propria.

5 Implementazione

Nella fase di implementazione andremo a sviluppare e presentare le scelte tecnologiche fatte ed infine, in linea con la progettazione, svilupperemo tutte le macro aree della nostra architettura. Non sarà presente una sottosezione sul sistema cloud, poiché la sua interazione con le altre parti del sistema verrà spezzata e descritta direttamente all'interno dei restanti capitoli.

5.1 Scelte tecnologiche e Linguaggi di Programmazione

Prima di passare alla vera e propria implementazione del nostro sistema, abbiamo deciso le tecnologie e i linguaggi di programmazione da utilizzare, in modo da rendere il sistema il più idoneo possibile alla progettazione ed all'analisi dei requisiti fatta precedentemente.

5.1.1 AWS - Amazon Web Service

Amazon Web Services è la piattaforma di cloud computing di Amazon. AWS non è un servizio, bensì una intera classe di servizi offerti ed amministrati tramite una interfaccia comune: la console web di AWS, raggiungibile puntando il browser all'indirizzo: <https://console.aws.amazon.com/>

La scelta è dovuta da una preliminare conoscenza di AWS, almeno dal punto di vista teorico, ma anche dal fatto che insieme a Microsoft e Google è una delle piattaforme cloud leader nel settore. Inoltre AWS ha una copertura regionale nel mondo e una grande ricchezza di servizi offerti [23].

I servizi AWS da noi utilizzati sono stati:

- **AWS IoT Core:** come si è evince dal nome, questa piattaforma consente a dispositivi connessi di interagire in modo semplice e sicuro sia tra di loro che con altre applicazioni. AWS IoT è stato il centro del nostro sistema cloud all'interno del progetto. Esso semplifica l'utilizzo di servizi AWS da integrare, consentendo di creare applicazioni IoT che raccolgono, elaborano, analizzano e operano sui dati generati dai dispositivi connessi, senza dover gestire alcuna infrastruttura.[24]

- **AWS DynamoDB:** è un servizio di database NoSQL. Esso supporta i modelli di dati di tipo documento e di tipo chiave-valore: quello che ci serviva per il nostro sistema e che si adatta in particolare a sistemi di tipo IoT. Si tratta di un database multi-master, multi-regione e completamente gestito che offre un servizio interamente gestito, sicurezza integrata, backup, ripristino e cache in memoria per applicazioni Internet.
- **AWS Cognito:** fornisce un servizio di autenticazione, autorizzazione e gestione degli utenti. I due componenti principali di Amazon Cognito sono i *pool di utenti* e i *pool di identità*. I pool di utenti sono directory utente che forniscono opzioni di registrazione e di accesso agli utenti delle applicazioni web o mobile. I pool di identità consentono di concedere agli utenti l'accesso ad altri servizi AWS. È possibile usare i pool di identità e i pool di utenti separatamente oppure insieme.
- **AWS S3:** è uno storage di oggetti creato per memorizzare e ripristinare qualsiasi volume di dati da qualsiasi origine. È un servizio che offre un'infrastruttura di storage estremamente durevole, altamente disponibile e scalabile in modo illimitato. Fra le varie funzionalità offerte, nel nostro caso è possibile ospitare un sito Web.
- **AWS Lambda:** è un servizio di elaborazione che consente di eseguire il codice senza gestire i server o effettuarne il *provisioning*. AWS Lambda esegue il codice solo quando è necessario e si dimensiona automaticamente. Il codice dell'applicazione viene caricato sotto forma di una o più funzioni Lambda all'interno del servizio di elaborazione AWS Lambda.
- **AWS IAM:** è un servizio Web che aiuta a controllare in modo sicuro l'accesso alle risorse AWS. Utilizza IAM per controllare chi è autenticato e autorizzato per l'utilizzo di risorse. Alla creazione di un account AWS, si possiede una singola identità avente accesso completo a tutti i servizi e le risorse AWS nell'account, per poi gestire la creazione di utenti e gruppi di utenti. Nel nostro caso è stato utilizzato per configurare le *policy* e le autorizzazioni per le risorse utili nel sistema.

5.1.2 MQTT

MQTT è un protocollo ISO standard di messaggistica leggero di tipo *publish-subscribe*. La scelta nasce dal pattern architetturale scelto in fase di progettazione, dalla sua particolare rilevanza ad un contesto IoT e dall'integrazione perfetta con il servizio di AWS IoT che supporta MQTT come protocollo di comunicazione.

5.1.3 Linguaggi di programmazione

- **Python:** è stato scelto per la vera e propria implementazione del software per quanto riguarda i dispositivi Flooding Kit e la Stazione d'allarme, in quanto risulta sia facilmente integrabile con i dispositivi fisici ed AWS IoT, sia ampiamente supportato a livello di librerie per l'interfacciamento con i sensori.
- **Javascript:** è stato utilizzato Javascript come linguaggio per la creazione dell'Applicazione poiché risulta quello più comunemente usato nella programmazione Web lato client.
- **Standard Web:** abbiamo utilizzato HTML e CSS per la realizzazione della nostra applicazione web. La scelta è ricaduta su queste tecnologie in quanto il nostro obiettivo è creare un'applicazione semplice e leggera che ci permettesse di essere chiara e lineare per un utente medio.

5.2 Flooding Kit

Il Flooding Kit, come è già stato ampiamente descritto, ha lo scopo di acquisire i dati dal Raspberry per poi inviarli al cloud tramite il protocollo MQTT. Per fare questo siamo partiti dall'acquisire i dati tramite i nostri sensori e solo in un secondo momento abbiamo implementato la comunicazione con il servizio IoT di AWS.

5.2.1 Creazione della Thing

Il primo step dell'implementazione del Flooding Kit consiste nella registrazione della Thing in questione nella sezione di "Manage" sulla console di AWS IoT, con la quale viene semplificata la gestione degli oggetti connessi. Per la creazione di

questo dispositivo abbiamo deciso di non creare alcun gruppo e tipo della Thing per rendere più semplice e lineare l'amministrazione del kit, la quale potrebbe essere svolta anche da una persona non specializzata.

Dopo la creazione della Thing, è stato necessario creare un certificato ed una copia di chiavi X.509, in quanto la comunicazione tra il dispositivo e AWS IoT risulta protetta tramite questi certificati. La chiave privata ed il certificato CA root sono risultati un utile metodo di autenticazione per interfacciarsi dal dispositivo fisico al sistema cloud.

Infine abbiamo dovuto creare una *policy AWS IoT* usata per autorizzare il dispositivo ad eseguire operazioni personalizzabili, come la sottoscrizione o la pubblicazione di topic MQTT. Il dispositivo presenta il proprio certificato durante l'invio di messaggi ad AWS IoT, al quale deve essere collegata la policy in questione. In particolare la "Flooding Policy", creata da noi consiste nel documento JSON seguente:

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": "iot:*",
7       "Resource": "*"
8     },
9     {
10      "Effect": "Allow",
11      "Action": "iam:*",
12      "Resource": "*"
13    },
14    {
15      "Effect": "Allow",
16      "Action": "dynamodb:*",
17      "Resource": "*"
18    }
19  ]
20 }
```

Nel codice soprastante abbiamo in particolare definito i permessi per:

- Effettuare un qualunque tipo di operazione riguardante: l'interazioni di base

tramite protocollo MQTT, la gestione della copie Shadow e l'esecuzione di eventuali job.

- Amministrare tutta la parte relativa alla gestione degli utenti, gruppi e ruoli.
- Effettuare tutte le possibili operazioni sul database DynamoDB.

Per concretizzare i permessi descritti dalla policy risulta infine obbligatorio associarla al certificato della Thing.

Nota: In un contesto reale potrebbe risultare utile limitare sia le azioni da poter effettuare, sia le Thing su cui poterle compiere, ottenendo dei permessi di esecuzione più restrittivi e quindi più adeguati in un'ottica di security.

5.2.2 Acquisizione Dati

Vengono riportati in seguito le parti di codice più salienti per quanto riguarda l'acquisizione dei dati per ogni singolo sensore e per l'integrazione di quest'ultime.

HC-SR04

Dopo aver settato il pin di TRIGGER come output e di il pin di ECHO come input, settiamo ad HIGH il pin di TRIGGER per un nanosecondo poiché è il tempo richiesto dal sensore per attivarsi e mandare l'impulso. Dopodichè lo resettiamo a LOW.

```
1 GPIO.output(GPIO_TRIGGER, True)
2 time.sleep(0.00001)
3 GPIO.output(GPIO_TRIGGER, False)
```

Per prima cosa, ci registriamo il momento in cui il pin ECHO termina di essere al valore LOW all'interno della variabile *StartTime*. In modo analogo ripetiamo lo stesso procedimento per verificare il momento in cui il suo valore ritorna a LOW. A questo punto, calcoliamo il tempo di andata e ritorno del segnale facendo la differenza fra *StopTime* e *StartTime*.

Infine, siccome sappiamo che la velocità delle onde ad ultrasuoni è di circa 34300 cm/s, per calcolare la distanza è necessario moltiplicare tale velocità per la durata calcolata precedentemente. Il risultato poi andrà diviso per due, in quanto la distanza appena valutata tiene conto sia del percorso di andata che di ritorno.

```

1 while GPIO.input(GPIO_ECHO) == 0:
2     StartTime = time.time()
3
4 while GPIO.input(GPIO_ECHO) == 1:
5     StopTime = time.time()
6
7 TimeElapsed = StopTime - StartTime
8 distance = (TimeElapsed * 34300) / 2

```

GY-BME/P 280

Per implementazione di tale sensore, abbiamo utilizzato un codice trovato online da un repository di GitHub. Quest'ultimo è risultato affidabile, in quanto si rifà al datasheet BME280 di Bosh.[32] Ovviamente l'abbiamo reso più leggibile ed abbiamo cercato di intuirne sommariamente il funzionamento.

Sensore FC-37 e YL-38

Per l'implementazione del sensore di pioggia, abbiamo preventivamente regolato la sua sensibilità collegandolo solamente all'alimentazione e verificando che il led del modulo YL-38, associato all'uscita digitale, si accendesse e si spegnesse in modo corretto in assenza o in presenza di pioggia. Successivamente abbiamo collegato solamente il pin digitale (DO) al Raspberry verificando quindi che inumidendo la piastra dell'FC-37 il valore del pin di input del raspberry passasse da HIGH a LOW.

Riduzione del rumore

Come primo step, abbiamo deciso di rendere parametrizzabile:

- quanti valori acquisire prima di inviare una rilevazione al server
- i valori minimi da scartare fra quelli letti
- i valori massimi da scartare fra quelli letti
- quanto attendere fra una rilevazione e l'altra

```

1 READS = 10
2 LOWEST_READS_TO_DISCARD = 3

```

```

3 HIGHEST_READS_TO_DISCARD = 3
4 SECONDS_BETWEEN_READS = 1

```

Successivamente, abbiamo salvato in degli array tutti i valori letti in una singola rilevazione e li abbiamo ordinati in modo crescente. Infine abbiamo preso solamente i valori centrali dell'array per scartare eventuali spyke ed abbiamo fatto una media di questi per ottenere il valore di ogni sensore da salvare sul database.

```

1 for i in range(READS):
2     proximities[i] = readDistance()
3     temperatures[i], pressures[i], humidities[i] = readBME280All()
4     time.sleep(SECONDS_BETWEEN_READS)
5
6 proximities.sort()
7 temperatures.sort()
8 pressures.sort()
9 humidities.sort()
10
11 clean_proximities =
12     proximities[LOWEST_READS_TO_DISCARD : -
13         HIGHEST_READS_TO_DISCARD]
14 clean_temperatures =
15     temperatures[LOWEST_READS_TO_DISCARD : -
16         HIGHEST_READS_TO_DISCARD]
17 clean_pressures =
18     pressures[LOWEST_READS_TO_DISCARD : -HIGHEST_READS_TO_DISCARD]
19 clean_humidities =
20     humidities[LOWEST_READS_TO_DISCARD : -HIGHEST_READS_TO_DISCARD
21 ]

```

5.2.3 Comunicazione con AWS IoT Core

Per l'implementazione di questa porzione del Flooding Kit è stato necessario utilizzare "AWS IoT Device SDK" per Python fornita da AWS. Quest'ultima consente agli sviluppatori di scrivere script Python per utilizzare i loro dispositivi e per accedere alla piattaforma AWS IoT tramite MQTT o RESTful API. Collegando i loro dispositivi a AWS IoT, gli utenti possono lavorare in sicurezza con il broker di messaggi, le regole, la *shadow* del dispositivo e con altri servizi.

Per connettere il dispositivo ad AWS IoT è obbligatorio utilizzare una modalità di

autenticazione. Ci sono 3 tipi di autenticazione, ma quella scelta da noi in questo caso consiste nell'utilizzo delle chiavi e certificati della thing.

Raccolti tutti i dati della rilevazione, come descritto precedentemente, abbiamo infine pubblicato sul topic dedicato, denominato *flooding-kit/ponte-vecchio-kit*, i valori di ogni singolo sensore. Per fare questo abbiamo creato un apposito client con la libreria dedicata per l'utilizzo delle operazioni di sottoscrizione e pubblicazione.

Il messaggio inviato per ogni singola rilevazione ha il seguente formato:

```
1 {  
2   "measureTime": 1562839853,  
3   "proximity": 20.0,  
4   "temperature": 25.9,  
5   "humidity": 42.6,  
6   "pressure": 1011.35,  
7   "raining": false  
8 }
```

5.2.4 Creazione delle regole di archiviazione

Le regole di archiviazione sono state create tramite AWS console, in particolare grazie all'utilizzo del servizio AWS IoT. Le regole, in generale, analizzano i dati ed eseguono operazioni in base al flusso di argomenti MQTT.

Fra le varie tipologie di regole da creare, abbiamo semplicemente utilizzato la regola che permettesse in automatico di salvare i dati su una tabella DynamoDB splittando il messaggio su più colonne. Per fare questo si utilizza un linguaggio *SQL-like*, che permette di prendere tutti i valori, con gli eventuali filtri ed operazioni tipici del linguaggio SQL, tramite i topic MQTT. Per il salvataggio del singoli kit si faccia riferimento alla figura sottostante.

RULE

ENABLED

Actions

Overview

Tags

DESCRIPTION

Description

No description

Rule query statement


The source of the messages you want to process with this rule.

```
SELECT * FROM 'flooding-kit/ponte-vecchio-kit'
```

Using SQL version 2016-03-23

ACTIONS

Actions are what happens when a rule is triggered. [Learn more](#)



Split message into multiple columns of a Dyna...

ponte-vecchio-kit

Remove

Edit

Add action

ERROR ACTION

Optionally set an action that will be executed when something goes wrong with processing your rule.

Add action

Figura 19: Esempio di regola di salvataggio dei dati - AWS IoT Console

In questo caso, con l’istruzione SQL della regola andiamo a raccogliere tutti i dati sul canale *FloodingKit/ponte-vecchio-kit*, ossia il topic sul quale vengono pubblicati le informazioni del kit specifico chiamato “ponte-vecchio-kit”.

Gestione tabelle con DynamoDB.

Per gestire il sistema, considerando le operazioni da fare e i vantaggi derivati dall'utilizzo di un sistema cloud, abbiamo deciso di creare:

- una tabella “Kits” che andasse a gestire la registrazione dei vari kit di un territorio particolare, come specificato nei requisiti. Essa comprende come attributi:

- **kitId**: STRING - chiave primaria
- **latitude**: NUMBER
- **longitude**: NUMBER
- una tabella per ogni singolo Flooding Kit registrato, che raccogliesse all’interno tutti i dati relativi a tutti i sensori. Gli attributi sono stati decisi in linea con il messaggio pubblicato sul topic per il salvataggio e sono:
 - **measureTime**: NUMBER - chiave primaria
 - **proximity**: NUMBER
 - **temperature**: NUMBER
 - **humidity**: NUMBER
 - **pressure**: NUMBER
 - **raining**: BOOLEAN

Per quanto riguarda la tabella del singolo Flooding Kit, la scelta della chiave primaria è ricaduta sull’attributo “measureTime”, in quanto le rilevazioni sono fatte ad una cadenza regolare, risulta quindi impossibile che due rilevazioni posseggano lo stesso attributo (composto dal timestamp della rilevazione stessa).

5.3 Stazione d’Allarme

In questa sezione andiamo a descrivere in che modo abbiamo implementato la stazione d’allarme, seguendo in modo fedele ciò che è stato scritto nella progettazione.

5.3.1 Creazione della Thing

La creazione della thing è del tutto identica al procedimento spiegato nella sezione di “Creazione della Thing” descritta nel “Flooding Kit”. A differenza del Flooding Kit, è stato necessario assegnare un tipo alla thing (mostrato nella figura seguente) avente come attributo lo stato della shadow, utile per verificare la presenza o meno dello stato di allarme.

Type	
Q AlarmStationType	...
3 Attributes	
Attribute key	Value
Q alarm	Value of this attribute is not defined.
Q alarmReason	Value of this attribute is not defined.
Q alarmTime	Value of this attribute is not defined.

Figura 20: Tipo della thing relativa alla Stazione d'Allarme - AWS IoT Console

5.3.2 Creazione regole d'allarme

Il procedimento di creazione delle regole è uguale a quello proposto per le regole di archiviazione. La differenza si ritrova nell'utilizzo di una diversa operazione che consente di ripubblicare su un topic MQTT un messaggio arrivato da un altro topic, verificando l'avvento di una particolare situazione per la quale è opportuno far scattare l'allarme.

Nella seguente figura andiamo a vedere un esempio di regola creata per la verifica della temperatura rilevata dal Flooding Kit.

RULE

TemperatureCheck

ENABLED

Actions ▾

Overview

Tags

Description

Check if temperature is between 20 and 30 Celsius degrees.

Edit

Rule query statement

The source of the messages you want to process with this rule.


```
SELECT measureTime as state.desired.alarmTime, 'on' as state.desired.alarm,
'Temperature is between 20 and 30' as state.desired.alarmReason FROM 'flooding-
kit/#' WHERE temperature > 20 and temperature < 30
```

Edit

Using SQL version 2016-03-23

Actions

Actions are what happens when a rule is triggered. [Learn more](#)



Republish a message to an AWS IoT topic

\$\$aws/things/AlarmStation/shadow/update

Remove

Edit ▸

Add action

Figura 21: Esempio di regola di allarme per la verifica della temperatura - AWS IoT Console

Come si può notare, la ripubblicazione del messaggio è stata fatta sul topic “\$aws/things/AlarmStation/shadow/update”. Tale topic è definito da AWS un “reserved topic”: il topic predefinito sul quale passa l’aggiornamento della shadow. Anche in questo caso l’istruzione di verifica è scritta in SQL-like. In Particolare andiamo ad esaminare la seguente istruzione, nuovamente riportata causa leggibilità.

```

1 SELECT measureTime as state.desired.alarmTime,
2     'on' as state.desired.alarm,
3     'Temperature is between 20 and 30' as state.desired.
   alarmReason
4 FROM 'flooding-kit/#'
5 WHERE temperature > 20 and temperature < 30

```

Nella “select” siamo andati a ricreare lo stato della shadow nel formato supportato dalla thing della Stazione d’Allarme. L’utilizzo dell’ “as” attraverso attributi in-

47

tervallati da punti permettono di creare il messaggio in formato JSON, come serve a noi per pubblicarlo nella shadow. Inoltre nella “where” andiamo a definire la regola vera e propria, la quale verifica che i valori rientrino all’interno di un range definito.

Note: Abbiamo utilizzato questa metodologia per la creazione e aggiunta di regole tramite AWS IoT Console, in quanto permette agli utenti di aggiungere singole regole personalizzate in modo semplice e potente. Un’alternativa sarebbe stata quella di utilizzare AWS IoT Event ma risultava a nostro parere troppo avanzato per un utente non esperto.

5.3.3 Attivazione Allarme

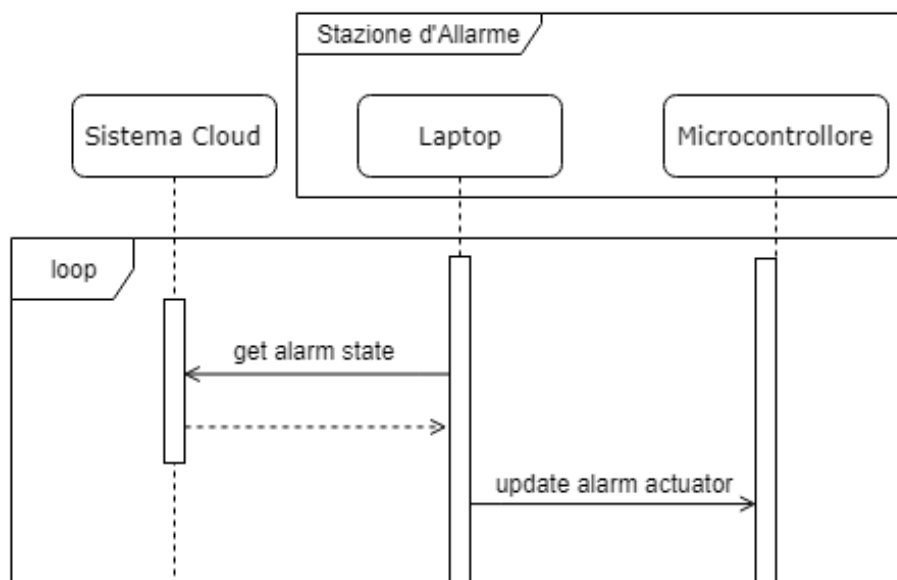


Figura 22: Diagramma di sequenza per l’ attivazione dell’allarme

Prima di passare alla vera e propria implementazione su arduino è stato necessario fare due step preliminari: aggiungere il seguente permesso nella policy Flooding Kit associata alla thing che permettesse di poter ottenere lo stato della shadow, ed inserire i certificati con l’utilizzo dell’apposita SDK per Python.

```

1 "Effect": "Allow",
2 "Action": "iot:GetThingShadow",
  
```

```

3 "Resource": ["arn:aws:iot:us-east-2:555494736670:thing
4                                     /AlarmStation"]

```

Per riuscire a far comunicare la shadow e l'arduino è stato necessario utilizzare la comunicazione seriale, in quanto attraverso uno script Python abbiamo ottenuto lo stato della shadow e in base al valore di quest'ultimo abbiamo dovuto comandare l'arduino in modo da accendere il led in caso d'emergenza. Per utilizzare la comunicazione tramite porta seriale in Python è stato necessario utilizzare la libreria Pyserial.

Nel primo step, tramite l'SDK per Python fornita da AWS, abbiamo semplicemente ottenuto lo stato della shadow tramite un apposito modulo della libreria, "AWSIoTMQTTShadowClient", avente una funzione dedicata. Ottenuto lo stato della shadow, è stato necessario verificare la presenza di una situazione d'emergenza grazie all'attributo "Alarm" del messaggio, di cui abbiamo già parlato in precedenza. A questo punto inviamo il messaggio tramite porta seriale all'arduino, il quale accende il led già configurato ponendolo ad HIGH.

5.3.4 Disattivazione Allarme

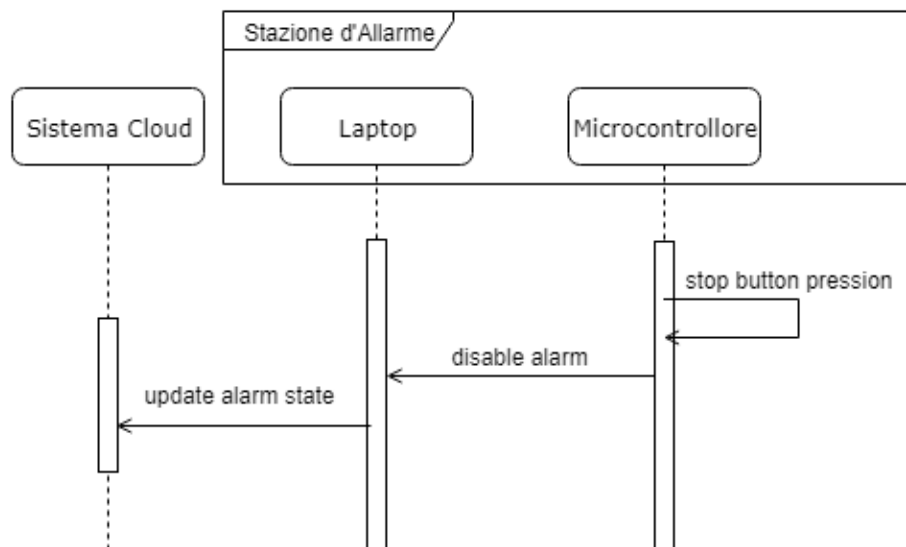


Figura 23: Diagramma di sequenza per la disattivazione dell'allarme

La disattivazione dell'allarme avviene tramite la pressione di un pulsante, connesso all'arduino. Dopo aver cliccato il pulsante, l'arduino deve comunicare e aggiornare la shadow della stazione d'allarme per impostazione l'attributo "Alarm" ad "off". Per effettuare ciò, utilizza sempre la comunicazione tramite porta seriale, come per l'attivazione dell'allarme: alla pressione del pulsante viene inviato un messaggio allo script in python, che ricevuto va a aggiornare la shadow sempre tramite la libreria "AWSIoTMQTTShadowClient" con un'apposita funzione.

5.4 Applicazione Web

L'endpoint a cui trovare il sito web è il seguente: <http://city-flooding-kit.s3-website.eu-west-3.amazonaws.com>. L'implementazione dello stile dell'applicazione web è stata fatta utilizzando uno *stylesheet bootstrap* pubblico, in quanto ci è sembrato un ottimo metodo per un veloce sviluppo prototipale.

Abbiamo utilizzato Javascript puro, sfruttando degli script importati direttamente nelle pagine web. Questi gestiscono in particolare:

Segnalazione e disattivazione dell'allarme

Per verificare o meno la presenza dello stato d'allarme, vengono effettuate regolarmente delle chiamate verso la shadow della stazione d'allarme tramite le API presenti nell'SDK per javascript fornite da AWS. Per fare questo, è necessario precedentemente autenticarsi. A differenza di come abbiamo fatto per l'implementazione del Flooding Kit, in questo caso abbiamo voluto sperimentare il servizio AWS Cognito.

visualizzazione e recupero delle informazioni

Per recuperare le informazioni relative al Flooding Kit ci siamo interfacciati al database su AWS DynamoDB tramite delle apposite RESTful API create da noi sul servizio AWS Lambda. Il codice è stato creato affidandosi ad un tutorial ufficiale di AWS IoT. In generale, quest'API non rispecchia propriamente i principi REST, poiché tutte le operazioni vengono fatte solamente attraverso una chiamata POST, che contiene al suo interno il tipo di operazione da effettuare.

```
1 var AWS = require('aws-sdk');  
2 var dynamoDB = new AWS.DynamoDB({apiVersion: '2012-08-10'});  
3
```

```

4 exports.handler = function(event, context, callback) {
5
6     var operation = event.operation;
7
8     if (event.tableName) {
9         event.payload.TableName = event.tableName;
10    }
11
12    switch (operation) {
13        case 'new':
14            dynamoDB.createTable(event.payload, callback);
15            break;
16        case 'create':
17            dynamo.put(event.payload, callback);
18            break;
19        case 'read':
20            dynamo.get(event.payload, callback);
21            break;
22        case 'update':
23            dynamo.update(event.payload, callback);
24            break;
25        case 'delete':
26            dynamo.delete(event.payload, callback);
27            break;
28        case 'list':
29            dynamo.scan(event.payload, callback);
30            break;
31        case 'echo':
32            callback(null, "Success");
33            break;
34        case 'ping':
35            callback(null, "pong");
36            break;
37        default:
38            callback('Unknown operation: ${operation}');
39    }
40 };

```

Infine, per effettuare la POST da codice javascript abbiamo deciso di utilizzare la nuova funzione di libreria *fetch*, piuttosto che la obsoleta *XMLHttpRequest*.

6 Testing e performance

Il progetto risulta essere conforme all'analisi dei requisiti: per verificare l'efficacia del sistema abbiamo effettuato appositi testing, i quali verranno esposti in seguito. Infine andremo a discutere sulla rilevazione delle performance per il nostro sistema.

6.1 Testing

Le modalità di testing utilizzati sono i seguenti:

Testing sull' acquisizione dei dati del singolo sensore

Il testing sull'acquisizione dei dati è stato fatto in modo preventivo all'invio delle informazioni sul giusto topic MQTT di AWS IoT.

Abbiamo deciso di creare, per ciascun sensore, singoli file implementati in Python per verificarne il corretto funzionamento. Tali file sono quelli che poi sono stati integrati per creare infine il messaggio da inviare al cloud per l'archiviazione. La vera e propria implementazione di questi file si può ricondurre alla parte già esposta nel capitolo di implementazione dei singoli sensori.

Infine lo scopo vero e proprio è quella di verificarne il corretto funzionamento del sensore in tutte le sue parti e la correttezza dei valori iniziali, almeno in modo approssimativo.

Testing sulla veridicità dei valori dei sensori

Al termine del progetto, si è deciso di testare se il sistema misurasse in maniera corretta i parametri ambientali. Per fare questo abbiamo utilizzato il nostro sistema in parallelo a strumenti esterni che potessero confermarci la veridicità dei valori ottenuti.

Abbiamo utilizzato come strumenti solamente un termometro e un righello, in quanto non abbiamo a disposizione mezzi in grado di misurare pressione e umidità.

Nella tabella verranno messi in confronto i valori misurati.

	Misurazione esterna al sistema	Misurazione interna al sistema
Temperatura	28.7°C	28.5°C
Distanza	10	10 cm

Per quanto riguarda la distanza abbiamo deciso di arrotondare all'unità, in quanto, essendo l'acqua influenzata dalla corrente, può subire una variazione di millimetri senza presentare un problema per la situazione ambientale.

Per quanto riguarda la pioggia abbiamo bagnato e asciugato la lastra che rileva la pioggia per verificarne la veridicità ed abbiamo testato che funziona tutto in modo corretto. Infine abbiamo deciso di prendere per veri i valori di pressione e umidità, poiché sono misurate insieme al valore di temperatura con lo stesso sensore fisico ed inoltre ci sembravano sensati in base alla nostra esperienza fisica.

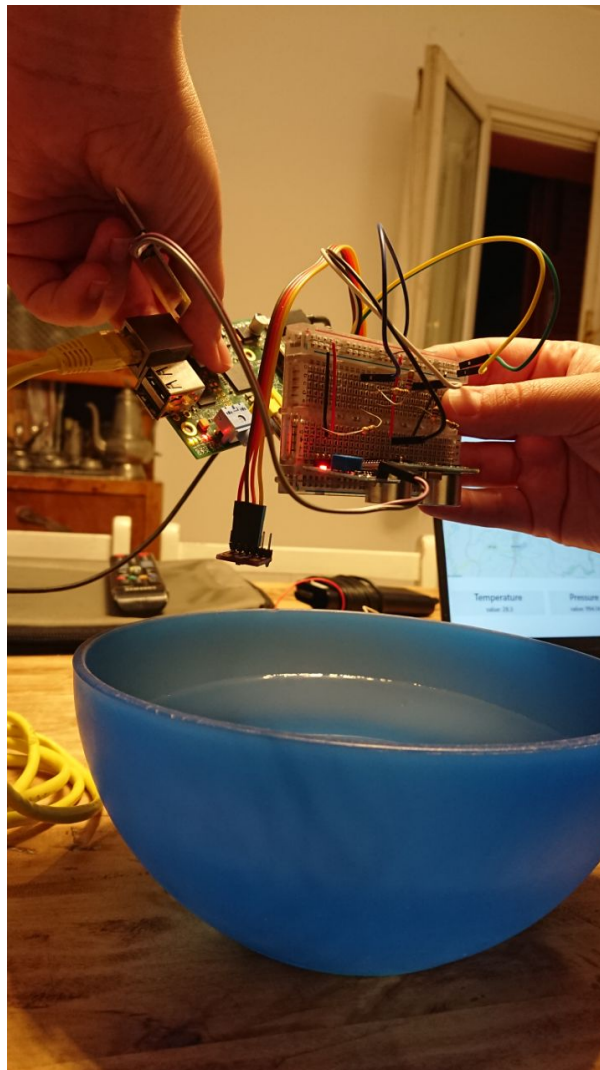


Figura 24: Verifica della misurazione della distanza su superficie acquosa

Testing sull'utilizzo di MQTT

Il testing sull'utilizzo dei topic MQTT creati e soprattutto per il funzionamento della comunicazione fra dispositivo fisico e AWS IoT è stato effettuato tramite console AWS, in quanto presenta una vera e propria sezione dedicata solamente a questo tipo di testing sia per la sottoscrizione ai vari topic sia per la pubblicazione di questi.

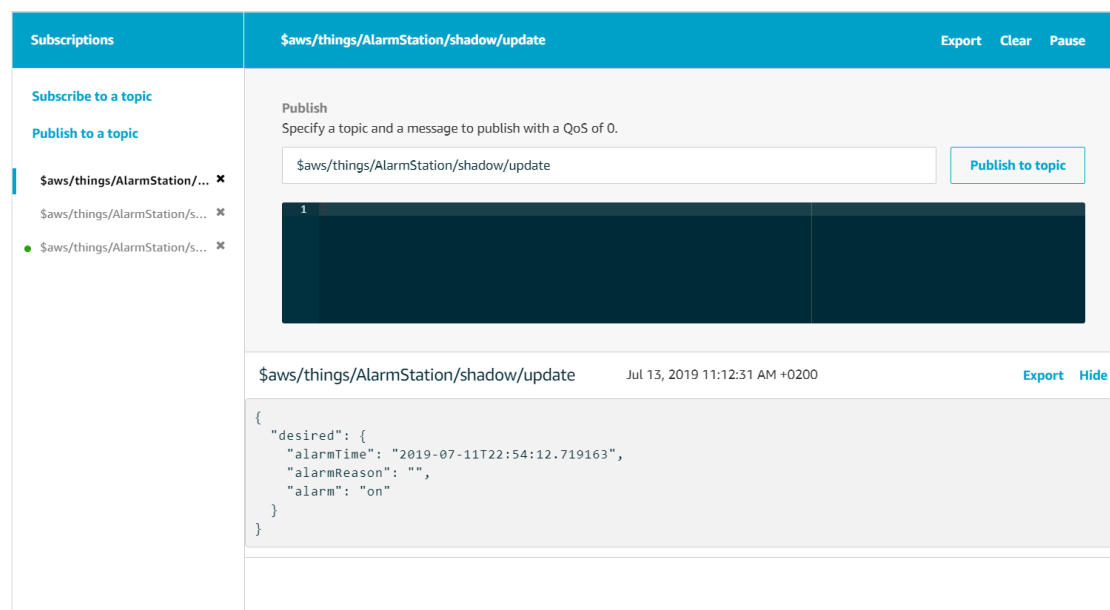


Figura 25: Verifica dell'aggiornamento della shadow

In questo caso abbiamo inviato dal sistema l'aggiornamento della shadow, verificando la ricezione in modo corretto sottoscrivendosi ai 3 canali riservati, che si trovano corrispettivamente nella colonna a sinistra dell'immagine:

- *\$aws/things/AlarmStation/shadow/update* per inviare l'update dello stato della shadow della stazione d'allarme
- *\$aws/things/AlarmStation/shadow/update/rejected* per verificare in caso d'errore quale sia il problema
- *\$aws/things/AlarmStation/shadow/update/accepted* per verificare la corretta ricezione del messaggio

Una volta ricevuto quindi il messaggio, viene mostrato sul primo topic, ma anche sul terzo. Per questo, il test ha esito positivo.

In modo analogo, abbiamo provato anche l'acquisizione dei dati dei sensori ricevuti dal Flooding Kit e tutto funziona in modo desiderato.

Testing sulla RESTful API creata

La RESTful API creata tramite AWS Lambda può essere testata direttamente da AWS Lambda all'interno della console dedicata. Quest'ultima possiede un menu nel quale è possibile inserire il body della chiamata ed eseguire la chiamata stessa. Alla fine di questo, verrà mostrato l'output della chiamata, che comprende anche i dati concreti richiesti e il numero di elementi nel risultato. Il body deve avere un particolare formato per la creazione delle chiamate, che è stato scelto in fase di creazione del RESTful API.

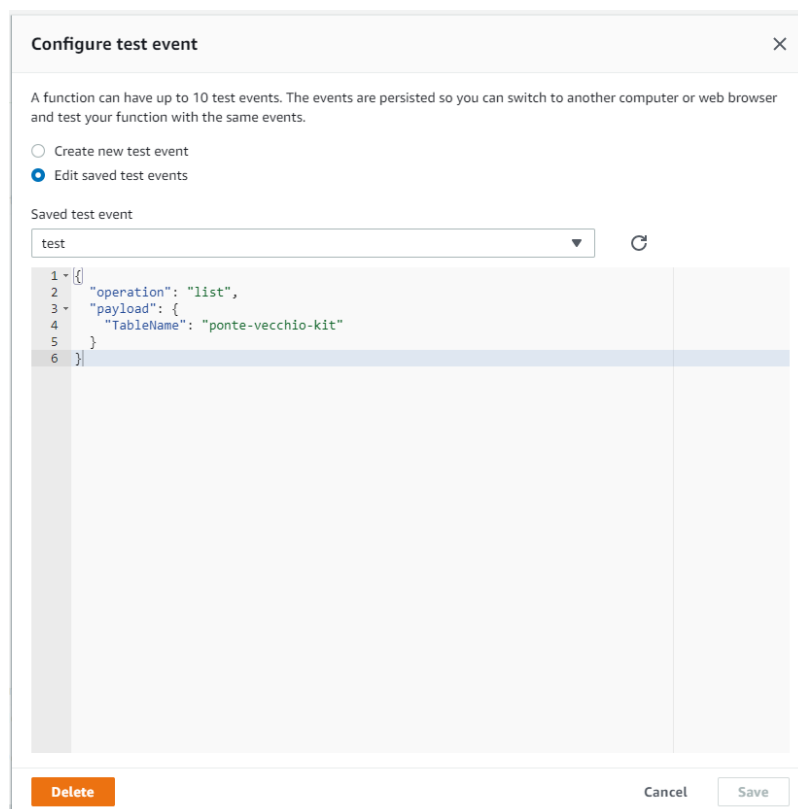


Figura 26: Esempio testing per l'operazione List - AWS Lambda Console

Ovviamente, operazione e payload sono modificati in base alla chiamata che si vuole fare, facendo riferimento a ciò che è implementato nel codice dell'API stessa. La risposta alla chiamata descritta sopra è la seguente.

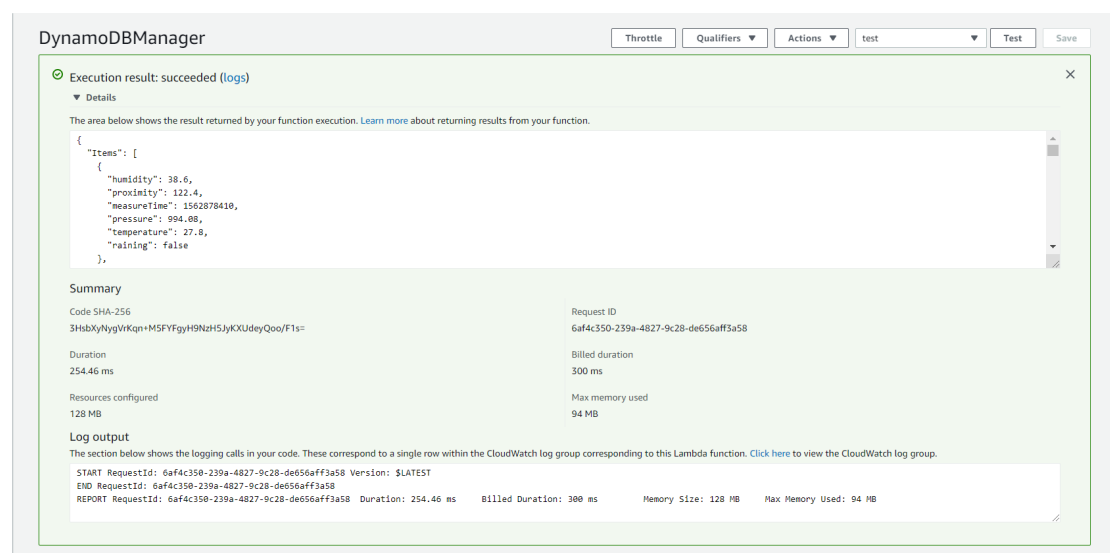


Figura 27: Esempio risultato della chiamata di List - AWS Lambda Console

Testing dell'Applicazione Web Per il testing dell'applicazione web, abbiamo fatto testare l'applicazione ai nostri familiari per valutare una user experience differente dalla nostra.

Le opinioni degli intervistati hanno toccato alcuni punti principali:

- è stato apprezzato l'utilizzo della mappa per localizzare i vari kit e l'immediatezza con il quale un utente riesce a verificare la situazione attuale;
- è stata criticata la visualizzazione dello storico che risulta troppo poco immediata ed è stato proposto l'utilizzo di un grafico per ottenere così un risultato più intuitivo e rappresentativo della situazione circoscritto ad un periodo temporale significativo. Questa critica sarà presa in considerazione per valutare gli sviluppi futuri.

6.2 Performance

Per quanto riguarda le performance, non abbiamo fatto particolari valutazioni di velocità, in quanto tutto il sistema è stato strutturato in modo tale che ogni azio-

ne fosse temporizzata secondo una necessità concreta. Per capire meglio questo, riportiamo le tempistiche delle varie attività del nostro sistema:

- La rilevazione e l'invio del messaggio contenente i dati dei sensori con annessa diminuzione del rumore viene fatta nell'ordine dei minuti. Infatti, il tempo di intervallo fra una rilevazione e l'altra è configurabile. Per il testing lo abbiamo impostato nell'ordine dei secondi, altrimenti avremmo dovuto aspettare troppo per verificare il funzionamento dell'intero sistema. Nel caso reale invece è utile impostarlo nell'ordine dei minuti, in quanto la condizione meteorologica non cambia così repentinamente.
- Il polling fatto per la verifica dello stato della shadow sia dalla stazione d'allarme sia dall'applicazione web è fatta nell'ordine dei secondi.

7 Analisi di deployment su larga scala

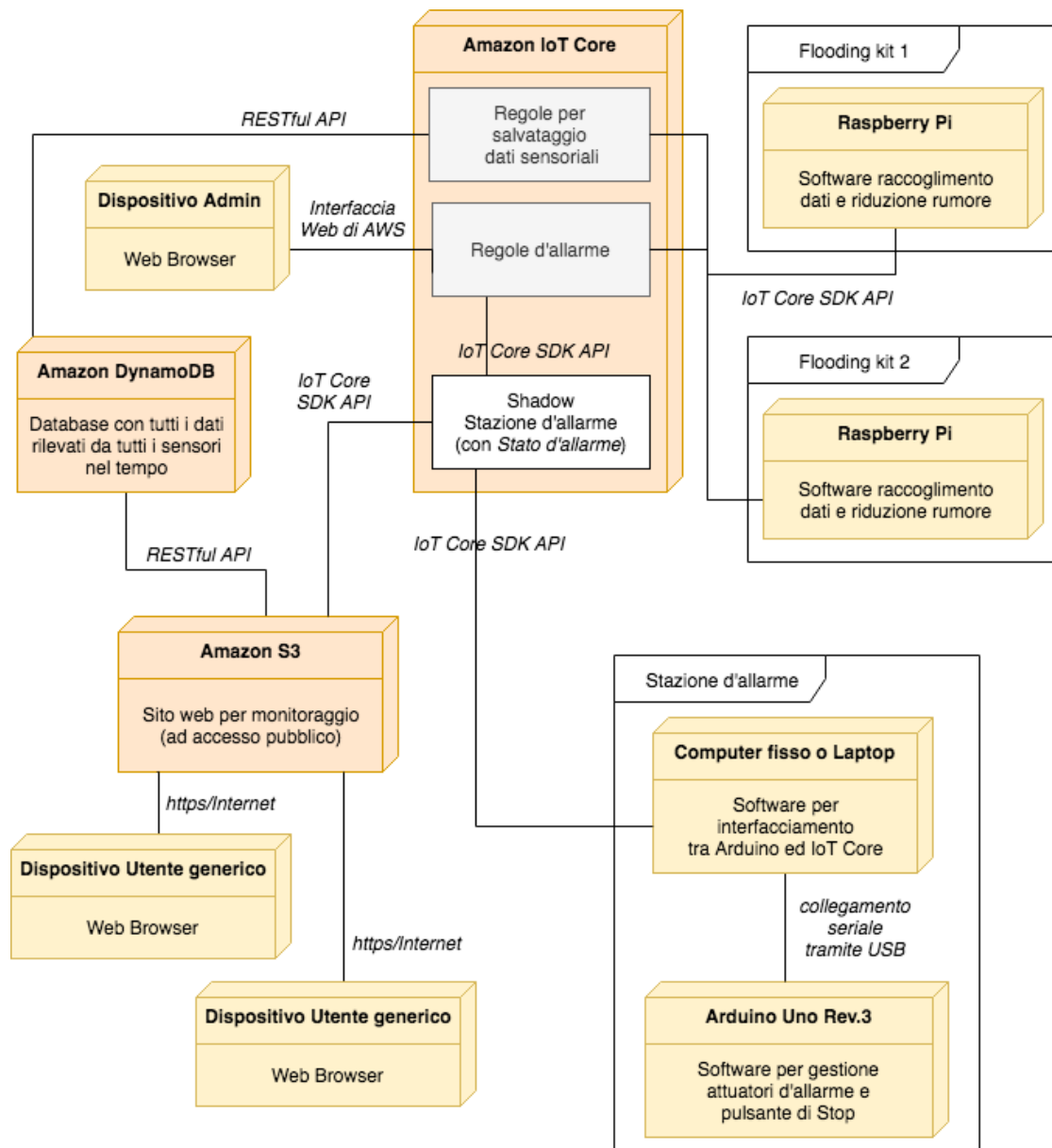


Figura 28: Diagramma di deployment per un cliente con due Flooding kit

Essendo il cuore della logica applicativa posto all'interno dell'infrastruttura cloud di AWS, il sistema risulta già particolarmente predisposto per un'eventuale adozione su larga scala. I due maggiori vantaggi ottenuti con questo tipo di approccio consistono certamente nella robustezza e nell'elasticità che offrono i contratti cloud: il modello “*pay as you go*” permette infatti di investire gradualmente sempre più fondi sul progetto, evitando di dover fare rischiosi ed ingenti investimenti a priori. Si ritiene interessante analizzare nel dettaglio i tre principali scenari nei quali risulterebbe effettivamente necessario scalare l'applicazione, specificando poi per ognuno di essi il modo in cui sarebbe opportuno far evolvere il progetto.

1. Singolo cliente con una moltitudine di Flooding kit e/o stazioni d'allarme
2. Molti clienti con pochi kit e stazioni d'allarme ciascuno
3. Molti clienti con numerosi kit e/o stazioni d'allarme

7.1 Cliente singolo, molteplici componenti hardware

Il traffico dati dai Flooding kit al sistema cloud risulterà molto più intenso a causa del maggior numero di sensori che si interfaceranno con il sistema: il numero di messaggi scambiati crescerà in modo direttamente proporzionale al numero di kit aggiunti. All'aumentare delle comunicazioni tra i kit ed il servizio AWS IoT sarà sufficiente investire all'occorrenza più fondi nel progetto: i prezzi di comunicazione sono infatti di 1\$ per milione di messaggi fino ad un totale di 1 miliardo di messaggi inviati, passando poi a 0.8\$/mln fino ad altri 4 miliardi di messaggi inviati, ed infine 0.7\$/mln oltre i 5 miliardi di messaggi [25].

Risulterà poi necessario ottenere uno *storage* molto più ampio per ospitare tutto lo storico dei dati rilevati dai sensori: ovviamente anche lo spazio di archiviazione aumenterà in modo direttamente proporzionale al numero di kit installati. In questo caso si dovrà valutare se risulterà più vantaggioso sfruttare un piano completamente *on demand* [26], oppure se potrebbe convenire una soluzione *provisioned* [27]. Converrà scegliere la prima opzione nel caso i carichi di lavoro ed il traffico proveniente dalle applicazioni web dovessero risultare ignoti. La soluzione *provisioned* potrebbe invece risultare perfetta per ottenere un ottimo controllo sui

costi nel caso siano prevedibili i requisiti di capacità.

Si potrà infine presupporre che l'interfaccia web riservata all'utenza ospite verrà utilizzata da molte più persone a causa della maggior copertura territoriale ottenibile con numerosi kit, andando quindi ad impattare sul numero di richieste effettuate verso il web server. In questo caso la soluzione più sensata sarebbe quella di sfruttare il servizio *Amazon Cloudfront* [28], il quale offre una perfetta integrazione tra *Amazon S3* [29] (sul quale viene già effettuato l'hosting dell'applicazione web di monitoraggio) ed *Amazon Elastic Load Balancing* [31], con lo scopo di garantire un'adeguata resilienza e disponibilità del sistema all'aumentare delle richieste.

7.2 Molti clienti, poche componenti hardware ciascuno

In questo caso la situazione risulta estremamente più semplice rispetto allo scenario descritto precedentemente. Sarà infatti sufficiente riproporre lo stesso sistema a clienti diversi, fornendo ognuno di essi di un differente account di amministrazione e riconfigurando tutto il sistema e tutte le regole di accesso per i diversi servizi di AWS nello stesso modo che si è già testato durante lo sviluppo di questo progetto. Nel caso si vendesse un prodotto molto simile a numerosi clienti si potrebbe considerare la possibilità di mantenere un prezzo molto competitivo sul prodotto base, per fornire poi espansioni a pagamento come personalizzazioni dell'interfaccia web ed un miglior servizio di assistenza e consulenza.

7.3 Molti clienti, molteplici componenti hardware

In questo ultimo scenario risulta naturale sfruttare entrambe le soluzioni riportate in precedenza in modo perpendicolare, riproponendo la soluzione già presa in esame per la scalabilità del singolo sistema ed applicandola a tutti i clienti che dovessero aver bisogno di aumentare le dimensioni della propria infrastruttura. Avendo un largo bacino di utenza i bisogni dei clienti saranno molto probabilmente di tipo eterogeneo: sarà quindi cruciale poter offrire anche una certa differenziazione di servizi e di fasce di prezzo.

Come ultima considerazione si vuole riflettere sul fatto che, nel caso il prezzo dei servizi di AWS o del cloud in generale dovessero diventare proibitivi, si potrebbe pensare di intraprendere diverse misure per uscire parzialmente o comunque progressivamente dalla dipendenza verso infrastrutture esterne. Vengono riportati di seguito alcuni spunti ed esempi:

- Trasferimento della logica applicativa per la gestione delle regole d'allarme direttamente all'interno dei singoli Flooding kit piuttosto che all'interno di IoT Core. Come controindicazione si renderebbe più complessa la centralizzazione delle regole d'allarme, in particolare quelle riguardanti il controllo a soglia sui valori di più sensori distribuiti su Flooding kit diversi.
- Eliminazione Shadow della Stazione d'allarme da IoT Core. Questo comporterebbe a non avere una comoda controparte digitale del componente fisico direttamente disponibile e sincronizzata particolarmente utile nei casi di manutenzione delle stazioni d'allarme o malfunzionamenti. Nulla impedisce però di costruirsi una soluzione simile in casa ed ospitarla su un proprio web server.
- Hosting dell'applicazione Web su server personali o su servizi specializzati, dotati di prezzi più competitivi rispetto ad Amazon S3. In questo caso per fare discovery dei servizi ed un eventuale *load balancing* si potrebbe pensare a soluzioni che coinvolgano Eureka e Ribbon nel contesto di applicazioni Spring Boot e Spring Cloud, oppure anche *Nginx* su una configurazione di server Heroku o server privati, piuttosto che Amazon Cloudfront.
- Trasferimento dati storici dei sensori su un database MongoDB piuttosto che su DynamoDB: le due soluzioni sono molto simili per quanto riguarda l'utilizzo che se ne farebbe nel nostro caso di studi, di conseguenza la migrazione risulterebbe semplice e naturale.

8 Piano di lavoro

Il piano di lavoro è stato suddiviso in modo equo all'interno del team di lavoro. La fase iniziale del progetto che comprende la pianificazione e progettazione del lavoro è stata portata avanti in contemporanea, aiutandosi a vicenda su tecnologie di cui non si avevano preve conoscenze, in particolare tutti i servizi utilizzati per AWS. Negli step successivi si è continuato a svolgere tutte le attività in modo più indipendente, suddividendo i compiti nel seguente modo:

- Giulia più orientata verso la parte cloud e lo sviluppo dell'applicazione web,
- Marco più orientato sulla parte hardware e relativo all'interfacciamento con il sistema cloud,

Nonostante la suddivisione dei compiti, entrambi i componenti del gruppo hanno però partecipato in modo più o meno approfondito allo sviluppo di tutte le parti del progetto, con lo scopo di imparare e sperimentare con più metodologie e tecnologie possibili, ma anche di rimanere al passo e monitorare totalmente lo stato di avanzamento del progetto.

Visto il modo di lavorare appena descritto, possiamo dire di aver seguito un approccio Agile, in quanto il focus della nostra progettazione è sempre stata la considerazione di un' ipotetico utente che utilizzerà il nostro sistema e ogni volta che ci incontravamo per portare avanti la realizzazione del sistema facevamo piccoli sprint in modo da monitorare lo stato di avanzamento e pianificare gli step successivi. Infine quindi, andando verso un approccio Agile, abbiamo utilizzato un criterio di segmentazione dei lavori che può ricondursi ad una "realizzazione evolutiva".

Il piano di lavoro adottato è stato un flusso di attività continuative dal momento dell'invio della prima email di proposta del progetto per circa un mese, dedicando al progetto almeno 4 ore al giorno da parte di entrambi i componenti, cercando di proseguire con un ritmo costante nonostante i diversi impegni personali e professionali.

Inizialmente siamo partiti andando a redigere una WBS, Work Breakdown Structure, in modo da avere un'idea omogenea, lineare e unica della pianificazione delle varie attività da svolgere. Ovviamente in questo caso non avremo il concetto di tempo, in quanto è un semplice grafo che permette di evidenziare tutte le attività in più livelli.

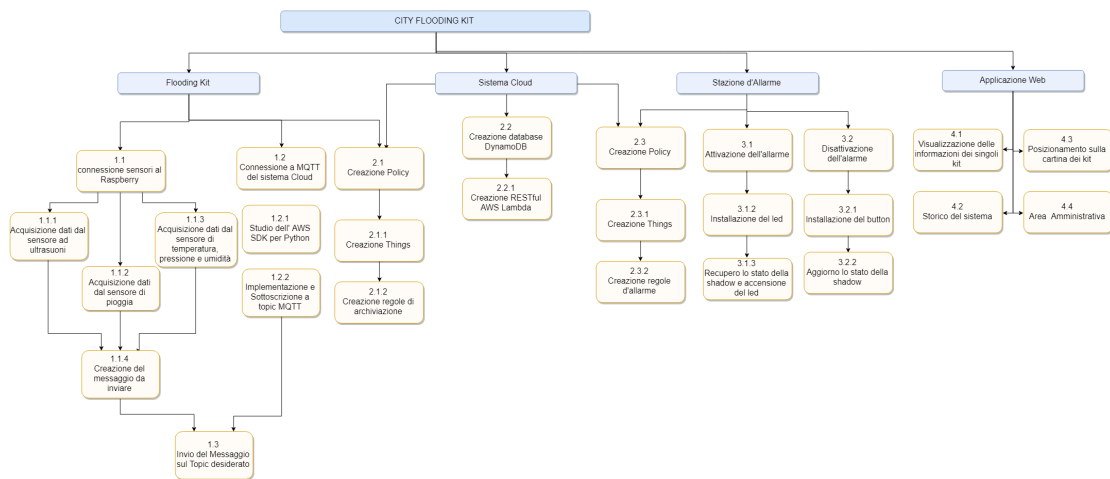


Figura 29: WBS - City Flooding Kit

Ora, partendo dalla WBS come insieme di attività, andiamo ad aggiungere il concetto di tempo creando il diagramma di Gantt che è stato utile per il monitoraggio dei tempi e per vedere lo stato di avanzamento del lavoro.

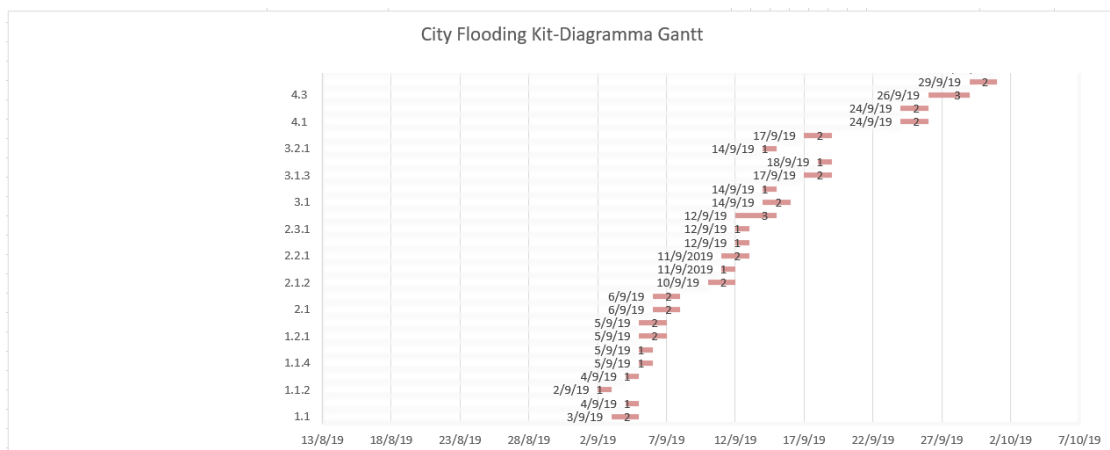


Figura 30: Diagramma di Gantt - City Flooding Kit

Ogni giorno/uomo rappresentato nel diagramma comprendeva circa 4 ore di lavoro, come abbiamo già detto sopra durante la spiegazioni sui tempi e la suddivisione del lavoro.

Come si nota dal diagramma di Gantt, il progetto in termini attività è stato terminato entro Giugno, per poi concluderlo, entro largo anticipo alla consegna,

con due settimane di refactoring, collaudo, manutenzione, testing e conclusione della documentazione di progetto.

9 Conclusioni

Il sistema finale rispecchia la proposta di progetto ed i requisiti stabiliti inizialmente, andando a risolvere con efficacia il goal concordato. Siamo riusciti nell'intento di partire dall'analisi di progetti reali [11] [12] [13] per riadattarli in un contesto tecnologico diverso, andando nel frattempo a toccare temi di sempre maggiore rilevanza come il monitoraggio e la percezione dei cambiamenti ambientali sul territorio. La cosa più interessante ai fini del corso è certamente stata la possibilità di poter sperimentare con numerose tecnologie e dispositivi fisici diversi all'interno di uno stesso progetto, andando a creare un ecosistema estremamente eterogeneo.

9.1 Difficoltà incontrate

Riportiamo di seguito le principali problematiche riscontrate durante lo sviluppo del sistema:

- Per quanto riguarda la componentistica hardware abbiamo cercato di riutilizzare il più possibile i dispositivi già in nostro possesso. Da un lato, ci è sembrato un ottimo modo sia per riportare in vita hardware vetusto, sia per simulare un contesto reale dove potessero esser presenti limitazioni di budget o di tecnologie. D'altro canto però sarebbe risultato molto più conveniente in termini di tempo investire nell'acquisto di dispositivi più recenti, o che comunque godessero di un maggior supporto sia a livello di librerie ufficiali che di community.
- Il monitoraggio ambientale è un contesto estremamente vario e che richiede un coinvolgimento multidisciplinare. In particolare, per la definizione delle regole d'allarme d'esempio sarebbe stata molto utile la presenza di un esperto del dominio da poter consultare durante il processo di sviluppo.
- L'utilizzo di AWS ci è risultato particolarmente complesso nelle fasi iniziali, in particolare per quanto riguarda la gestione dei ruoli, permessi e policy.

9.2 Sviluppi futuri

- Montaggio e testing del dispositivo in contesto reale.
- Aggiunta di ulteriori possibilità per la terminazione dello stato d'allarme: non solo tramite pulsante fisico, ma anche tramite interfaccia grafica o simili.
- Maggiore attenzione alla sicurezza del sistema, in particolare per quanto riguarda il salvataggio delle password su DynamoDB, al momento in chiaro.
- Inserimento di grafici nell'applicazione web per la visualizzazione delle rilevazioni dei sensori nel tempo.

Riferimenti bibliografici

- [1] Flood Monitoring. European Space Agency. Date of access: June 2019.
https://www.esa.int/Our_Activities/Observing_the_Earth/Securing_Our_Environment/Flood_monitoring
- [2] Research Note-Flood information systems in Europe: a survey of trans-boundary river management.
<https://search.proquest.com/docview/1943067814/6D81DD14A8654C61PQ/1?accountid=963>
- [3] SMART ENVIRONMENT «Smart City» - GREEN JOBS - "Ministero del Lavoro e delle Politiche Sociali".
https://www.cliclavoro.gov.it/Progetti/Green_Jobs/Documents/Smart_City/4_Smart%20Env
- [4] Application of Remote Sensing and Geographical Information Systems in Flood Management: A Review. Emmanuel Opolot, 2013. Ghent University, Department of Geology and Soil Science, Laboratory of Soil Science, Krijgslaan 281/S8 B-9000, Gent, BELGIUM
- [5] Group on Earth Observation: Global Flood Risk Monitoring.
<https://www.earthobservations.org/activity.php?id=94>
- [6] Samarasinghea, S.M.J.S., H.K. Nandalalb, D.P. Weliwitiyac, J.S.M. Fowzed, M.K. Hazarikad and L. Samarakoond, 2010. Application of remote sensing and GIS for flood risk analysis: A case study at Kalu-Ganga River, Sri Lanka. Int. Arch. Photogr. Remote Sens. Spatial Inform. Sci., Vol. 38, Part 8, Kyoto Japan 2010.
- [7] Chormanski, J., T. Van de Voorde, T. Deroeck, O. Batelaan and F. Canters, 2008. Improving distributed runoff prediction in urbanized catchments with remote sensing based estimates of impervious surface cover. Sensors, 8: 910-932.
- [8] Irimescu, A., G.H. Stancalie, V. Craciunescu, C. Flueraru E. Anderson, 2009. The Use of Remote Sensing and GIS Techniques in Flood Monitoring and Damage Assessment: A Study Case in Romania. Threats to Global Water Security, NATO Science for Peace and Security Series C: Environmental Security, pp: 167-177.

- [9] Uddin, K. and B. Shrestha, 2011. Assessing flood and flood damage using remote sensing: A case study from Sunsari, Nepal. Proceeding of the 3rd International Conference on Water and Flood Management.
- [10] Nihon Kasetu: Flood monitoring and early warning system.
<http://nihonkasetu.com/flood-monitoring-and-early-warning-system/>
- [11] Early flood detection and warning system in Argentina developed with Libelium sensors technology. <http://www.libelium.com/early-flood-detection-and-warning-system-in-argentina-developed-with-libelium-sensors-technology/>
- [12] FLOOD BEACON - Real time flood-level data and alerts.
<http://floodbeacon.com/>
- [13] Stazione idrometrica Fiume Cherio - Trescore Balneario (BG).
<http://www.bitlineftp.com/trescore/>
- [14] 29-esima Conferenza internazionale su Advanced Networking e workshop di applicazioni, 2015 IEEE.
<https://ieeexplore.ieee.org/document/7096250>
- [15] GY-BME280 Pressure Humidity Temperature Sensor Module.
<https://protosupplies.com/product/gy-bme280-pressure-humidity-temperature-sensor-module/>
- [16] Raspberry PI 1 model b revisione 2.0.
https://it.wikipedia.org/wiki/Raspberry_Pi#Raspberry_Pi/Pi_revisione_2
- [17] Il bus I2C.
<https://www.linux.it/~rubini/docs/i2c/i2c.html>
- [18] Protocollo di comunicazione I2C.
<http://www.crivellaro.net/wp-content/uploads/2017/03/Protocollo-di-comunicazione-I2C.pdf>
- [19] “Come funziona il sensore ad Ultrasuoni”.
<https://www.militarypedia.it/come-funziona-il-sensore-ultrasuoni/>

- [20] Misuratore di distanze ad ultrasuoni HC-SR04.
<http://www.radiofo.it/files/articoli/sku026931new.pdf>
- [21] HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi.
<https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi>
- [22] Arduino Official Page.
<https://www.arduino.cc/>
- [23] About AWS - Official site.
<https://aws.amazon.com/it/what-is-aws/>
- [24] AWS IoT Core.
<https://aws.amazon.com/it/iot-core/>
- [25] AWS IoT Core Prices.
<https://aws.amazon.com/it/iot-core/pricing>
- [26] DynamoDB ondemand prices.
<https://aws.amazon.com/dynamodb/pricing/on-demand/>
- [27] DynamoDB provisioned prices.
<https://aws.amazon.com/dynamodb/pricing/provisioned/>
- [28] Amazon Cloudfront.
<https://aws.amazon.com/it/cloudfront/>
- [29] Amazon S3.
<https://aws.amazon.com/it/s3/>
- [30] Amazon Elastic Load Balancing.
<https://aws.amazon.com/it/elasticloadbalancing/>
- [31] Netflix Eureka.
<https://spring.io/guides/gs/service-registration-and-discovery/>
- [32] BME280 Digital Humidity, Pressure and Temperature Sensor. Repository GitHub.
<https://github.com/ControlEverythingCommunity/BME280>