Embedded Systems for the Internet of Things

A.A. 2017/2018

Progetto: Smart Recycling

Gruppo: Riccardo Locci, Luca Piras

Obiettivo

Il progetto “Smart Recycling” consiste nella realizzazione del prototipo di un cassonetto dei rifiuti “intelligente”, dotato di un sensore di imaging in grado di fotografare un oggetto, applicare su di esso un processo di riconoscimento e, in base a un algoritmo decisionale implementato in Cloud, decidere se permetterne o meno lo smaltimento. L’esito della valutazione è confermato all’utente per mezzo di una serie di LED, tramite una semplice codifica basata sul colore. Un'applicazione web permette all’utente finale di monitorare in tempo reale diverse statistiche sul riconoscimento degli oggetti e di verificare manualmente le immagini inviate. Il cassonetto connesso supporta il meccanismo del digital twin, uno strumento che consente a un device fisico di essere rappresentato da una sua “replica” digitale, che ne simula lo stato.

Contesto e problematiche

Come risulta evidente dalle specifiche, il progetto è composto, a livello macroscopico, da due parti principali: quella hardware, rappresentata dal cassonetto stesso e dai sensori a esso collegati, e quella software, realizzata in Cloud (a eccezione del codice strettamente necessario al funzionamento dei sensori e degli attuatori). La parte hardware del prototipo è basata su NodeMCU, una piattaforma open-source specifica per l’IoT che sfrutta le funzionalità del microcontrollore ESP8266; le componenti Cloud sono state implementate esclusivamente sfruttando i servizi di Amazon Web Services, tra i quali Internet of Things, Lambda, S3, Dynamo DB e Rekognition.

Una delle problematiche più importanti da affrontare nel progetto in questione consiste nella comunicazione tra le componenti hardware e quelle software: nello specifico, è necessario trovare un metodo flessibile ed efficiente per spedire in Cloud le immagini acquisite dalla fotocamera. A questo scopo è stato scelto di usare il protocollo MQTT, per mezzo di un broker offerto dal servizio Internet of Things di AWS. Infatti, tale protocollo consente l’invio di immagini sotto forma di stringhe in formato base64, che possono essere facilmente riconvertite in jpeg una volta arrivate alla componente Cloud adibita alla gestione e al processing dell’immagine.

Un’altra problematica riguarda la scelta della fotocamera: infatti, molte fotocamere sviluppate per piattaforme Arduino-compatibili presenti sul mercato richiedono più pin di quelli messi a disposizione dalla board ESP8266. Per questo motivo, è stato necessario trovare un dispositivo che,

pur garantendo performance adeguate alle specifiche del progetto, permettesse agevolmente il collegamento con la board, senza richiedere dispositivi di supporto aggiuntivi come gli estensori di pin. I dettagli della fotocamera selezionata per l'implementazione del progetto verranno forniti nel prossimo paragrafo.

Per quanto riguarda la parte Cloud, la decisione progettuale più rilevante consiste nella scelta di un modo efficiente per gestire il processing dell'immagine, che si sviluppa tra la ricezione del file per mezzo di MQTT, il suo salvataggio permanente e, allo stesso tempo, l'accessibilità da parte del componente software adibito al riconoscimento dell'oggetto. Un'implementazione funzionale di questo flusso è garantita dall'integrazione di tre servizi AWS: S3, finalizzato allo storage, Rekognition, adibito al riconoscimento, e Lambda, che chiama al suo interno i due servizi appena menzionati e che svolge il ruolo di nodo di smistamento tra i sensori e il Cloud.

Le soluzioni alle problematiche appena descritte, insieme ad altre scelte progettuali e alla descrizione delle diverse componenti, verranno dettagliate nel prossimo paragrafo. La Figura 1 sintetizza le interazioni e il flusso di funzionamento del prototipo.

Scelte progettuali

Componenti Hardware

ArduCAM

La ArduCAM Mini Camera Module Shield w/ 2 MP OV2640 è una fotocamera ad alta definizione compatibile con Arduino, NodeMCU, Raspberry Pi e altre piattaforme che forniscano interfaccia SPI e I2C. Fornisce un'interfaccia hardware e la possibilità di essere integrata con librerie open source. Come anticipato nel precedente paragrafo, la scelta della fotocamera è stata guidata dall'esigenza di non superare il numero di ingressi I/O messi a disposizione dalla board ESP8266. Il modello in

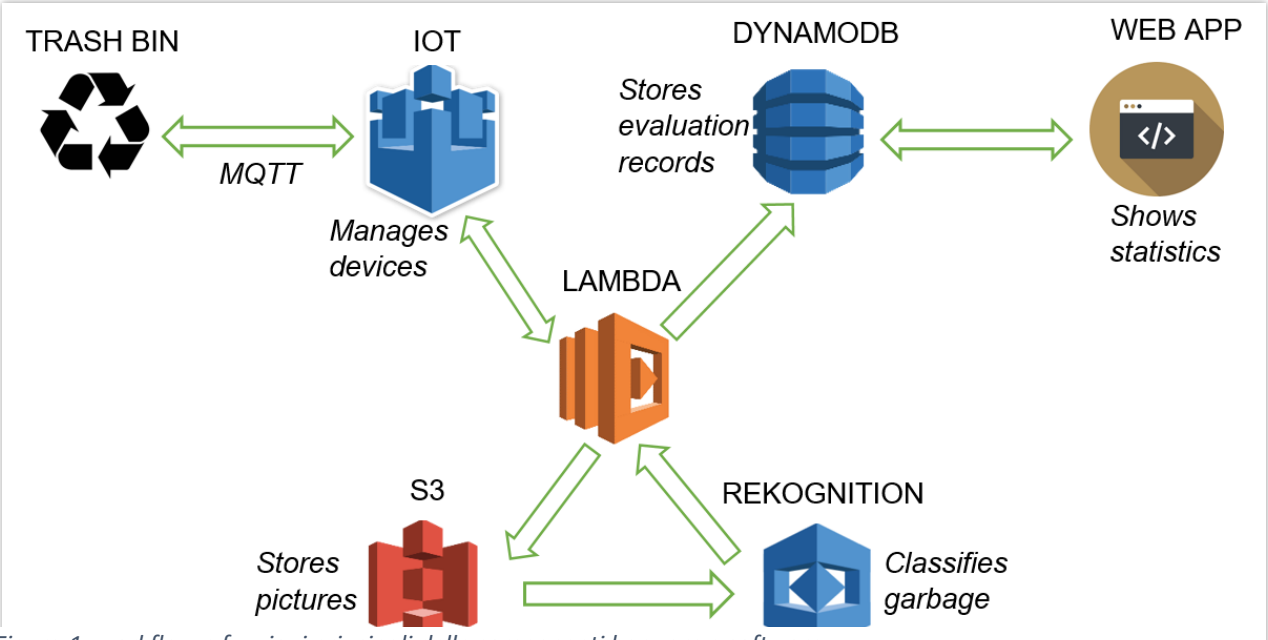


Figura 1: workflow e funzioni principali delle componenti harware e software.

questione richiede l'utilizzo di 8 pin: 6 dedicati ai dati e 2 all'alimentazione. Inoltre, le sue dimensioni e peso ridotti la rendono molto adatta ad essere applicata a dispositivi come un cassonetto dei rifiuti.

La fotocamera è attivata dalla pressione di un bottone e il suo ruolo è quello di fotografare i rifiuti che l'utente intende smaltire. L'immagine verrà successivamente inviata in Cloud per il riconoscimento.

Modulo Relay

Un modulo relay compatibile con Arduino e Raspberry viene sfruttato per alterare il circuito elettrico e permettere l'attivazione della serratura. Riceve l'input ogni volta che l'algoritmo decisionale individua un rifiuto compatibile con il cassonetto.

Serratura elettrica

La chiusura e l'apertura del cassonetto sono gestite da una serratura elettrica. Nel suo stato normale, essa rimane chiusa. Analogamente al modulo relay, viene attivata quando l'algoritmo decisionale riconosce un oggetto compatibile.

Componenti Cloud

IoT

Internet of Things è un servizio di AWS che permette di connettere i dispositivi al Cloud e fra di loro. Consente all'utente di gestire da remoto diversi sensori e di analizzare il flusso di dati proveniente da essi. Fornisce inoltre un broker MQTT, indispensabile al fine dell'invio delle immagini acquisite dalla fotocamera. Dato che le specifiche del progetto richiedevano l'utilizzo dei servizi offerti da AWS per tutte le componenti Cloud, la scelta di IoT è risultata naturale: essa è infatti la soluzione standard all'interno di Amazon per la gestione di sensori che abbiano la necessità di comunicare con del software che risieda nel Cloud.

S3

AWS S3 è un file system totalmente implementato in Cloud, in grado di salvare oggetti di diversi formati all'interno di "bucket" indirizzati da id univoci. Si è scelto di usare questo servizio in virtù della sua flessibilità nel gestire lo storage di immagini (si rimanda al codice per maggiori dettagli) e perché è uno strumento progettato specificamente per

adattarsi a scenari di Internet of Things. Nel prototipo del cassonetto, S3 è usato per salvare in modo permanente le immagini acquisite dalla fotocamera. Le nuove foto sono sempre aggiunte al medesimo bucket creato in principio e sono indirizzate in modo semplice dal loro nome.

Rekognition

AWS Rekognition è una API di Computer Vision in grado di rilevare oggetti all'interno di immagini o video. È sufficiente fornire alla funzione dedicata il file di una foto per ottenere una lista di labels, corrispondenti agli oggetti individuati al suo interno, insieme alle relative probabilità (interpretabili come valori di confidenza dell'algoritmo di riconoscimento).

Uno dei vantaggi di tale servizio è il fatto che metta a disposizione modelli pre-addestrati, già pronti all'uso, che quindi non richiedono lunghe fasi di training. Gli algoritmi di riconoscimento sfruttano tecniche avanzate allo stato dell'arte basate sul deep learning.

Nel progetto in questione, Rekognition è stato utilizzato per valutare a quale delle seguenti categorie appartiene il rifiuto che si intende smaltire: carta, vetro, plastica, organico. Di seguito viene delineato l'algoritmo decisionale responsabile di consentire o meno l'apertura del cassonetto.

Prima di tutto, viene creata una lista statica di labels per ogni categoria, usando una serie di immagini prese a campione al fine di collezionare le labels più frequenti. Ogni volta che lo script della Lambda (descritta nel seguito) riceve un'immagine acquisita dalla fotocamera, viene invocata su di essa la funzione di Rekognition, che restituisce la lista di labels individuate all'interno della foto. A questo punto, l'algoritmo controlla quante labels di ogni categoria appaiono anche fra le labels restituite da Rekognition: la categoria che registra il numero più alto di corrispondenze viene assegnata all'oggetto della foto. Rimandiamo al codice per maggiori dettagli.

DynamoDB

AWS DynamoDB è un database NoSQL che fornisce prestazioni affidabili su larga scala. È strutturato in tabelle a formato flessibile, nel senso che il numero e il tipo dei campi di ogni record non è fissato a priori. La scelta di DynamoDB è stata guidata non solo da tale elasticità nel gestire il salvataggio dei dati, ma anche dalla sua capacità di scalare qualora il progetto assumesse dimensioni progressivamente maggiori. Inoltre, questo servizio garantisce accesso ai dati a bassa latenza, che è un fattore di grande importanza negli scenari di IoT, in cui la sincronizzazione tra sensori, dispositivi e componenti software è cruciale.

Nel prototipo in questione, DynamoDB è usato per tenere traccia dei rifiuti che sono stati valutati dal cassonetto nel corso del tempo. In particolare, è stata creata una tabella con i seguenti campi:

- Item id: identificatore unico dell'oggetto;
- Classe assegnata: categoria individuata dall'algoritmo decisionale;
- Nome del file: corrisponde al nome dell'immagine salvata sul bucket S3;
- Labels: etichette individuate da Rekognition;
- Classe reale: categoria del rifiuto assegnata manualmente dall'utente finale tramite l'applicazione web.

Lambda

AWS Lambda è un servizio che consente l'esecuzione di codice sul Cloud senza la necessità di gestire un server. Il codice è attivato automaticamente da *triggers* e le risorse utilizzate dal programma sono ridimensionate di continuo in base al carico di lavoro e alla necessità.

Come evidenzia il flusso di funzionamento sintetizzato dalla Figura 1, l'architettura del progetto ha bisogno di un elemento che funga da nodo di smistamento tra le componenti hardware e quelle software e che consenta l'interazione delle diverse componenti cloud. Questo ruolo è, per l'appunto, svolto da una funzione Lambda, che nel nostro caso specifico sfrutta la libreria Python Boto3 (https://boto3.amazonaws.com/v1/documentation/api/latest/index.html), per richiamare al suo interno gli altri servizi, quali ad esempio S3 e Rekognition.

In particolare, la funzione Lambda implementata nel prototipo in questione è responsabile delle seguenti funzionalità:

- Ricezione delle immagini da IoT sotto forma di stringa in formato base64;
- Conversione delle immagini in formato jpeg;
- Salvataggio delle immagini nel bucket S3;
- Invocazione di Rekognition sulle immagini ricevute e attivazione dell'algoritmo decisionale;
- Salvataggio delle valutazioni dei rifiuti sulla tabella di DynamoDB.

Web App

Un'applicazione web (http://esit-website-bucket.s3-website-eu-west-1.amazonaws.com/) permette all'utente finale di interagire con il cassonetto. In particolare, egli può verificare manualmente la correttezza delle valutazioni svolte dall'algoritmo e può monitorare una serie di statistiche sul riconoscimento dei rifiuti:

- Percentuale di ogni categoria fra tutti i rifiuti smaltiti nel corso del tempo;
- Numero di valutazioni giornaliere;
- Accuratezza delle valutazioni.

Si è scelto di allocare il sito all'interno di un bucket S3, con il supporto di javascript e react-js per l'implementazione dell'interfaccia. Questa risulta essere la scelta più conveniente rimanendo in ambiente AWS, considerata anche la semplicità dell'applicazione richiesta.

Considerazioni finali

Uno dei fattori più importanti da tenere in considerazione al momento di valutare un prototipo, specialmente in ambito IoT, è la scalabilità, intesa come la capacità di un sistema di gestire un aumento progressivo delle proprie dimensioni o del carico di lavoro supportato (ad esempio in termini di numero di dispositivi).

Se analizziamo le componenti cloud del progetto presentato, queste sono ottimamente predisposte a scalare in modo efficace. Questo è garantito dalla piattaforma AWS, che è stata pensata e sviluppata appositamente con l'obiettivo di prestarsi a progetti di dimensione variabile e a larghissima scala. Alcune caratteristiche dei servizi impiegati nel progetto in questione rendono il prototipo particolarmente predisposto alla scalabilità. Per esempio, AWS IoT è in grado di gestire contemporaneamente migliaia di dispositivi connessi in cloud; la funzione Lambda implementata per un singolo cassonetto può essere applicata a una moltitudine di cassonetti quasi senza necessità di modifiche; S3 e DynamoDB sono servizi di storage adatti a sistemi di grandi dimensioni.

Qualora un progetto di ICT diventi anche un progetto imprenditoriale o di pubblica amministrazione, la scalabilità non è data solamente da elementi di carattere tecnico, ma anche – se non soprattutto – economico. In questo senso, si può dire che un progetto è scalabile se il costo della realizzazione e della gestione cresce in modo "lineare o meno che lineare" rispetto alle dimensioni. Anche da questo punto di vista, AWS si dimostra una scelta vincente: tutte le tariffe dei servizi sono infatti calcolate basandosi sull'uso effettivo del servizio stesso. Basta osservare, come esempio, i prezzi di alcuni dei servizi utilizzati per il progetto, per rendersi conto che le tariffe sono pensate appositamente per sistemi di grandi dimensioni. IoT costa 0,1€ per ogni 1.000 dispositivi registrati; DynamoDB costa 0,22€ per ogni GB (i primi 25 GB di ogni mese sono gratuiti), mentre S3 0,02€ per GB; Lambda concede 1 milione di richieste gratis ogni mese; Rekognition costa 0,86€ per 1.000 richieste.

La scalabilità del progetto è, naturalmente, limitata dalle componenti hardware. Tuttavia, vale una serie di considerazioni: con il supporto di dispositivi come estensori di pin I/O, una singola board ESP8266 può essere utilizzata per più di un cassonetto; un modulo relay può supportare fino a 4 serrature elettriche; il codice scritto per un dispositivo può essere facilmente modificato per gestire un gruppo di device. Tenendo in considerazione i prezzi delle componenti hardware utilizzate riscontrati sull'e-commerce di Amazon, risulta che il costo di realizzazione di un singolo cassonetto si aggira intorno ai 90-110€.

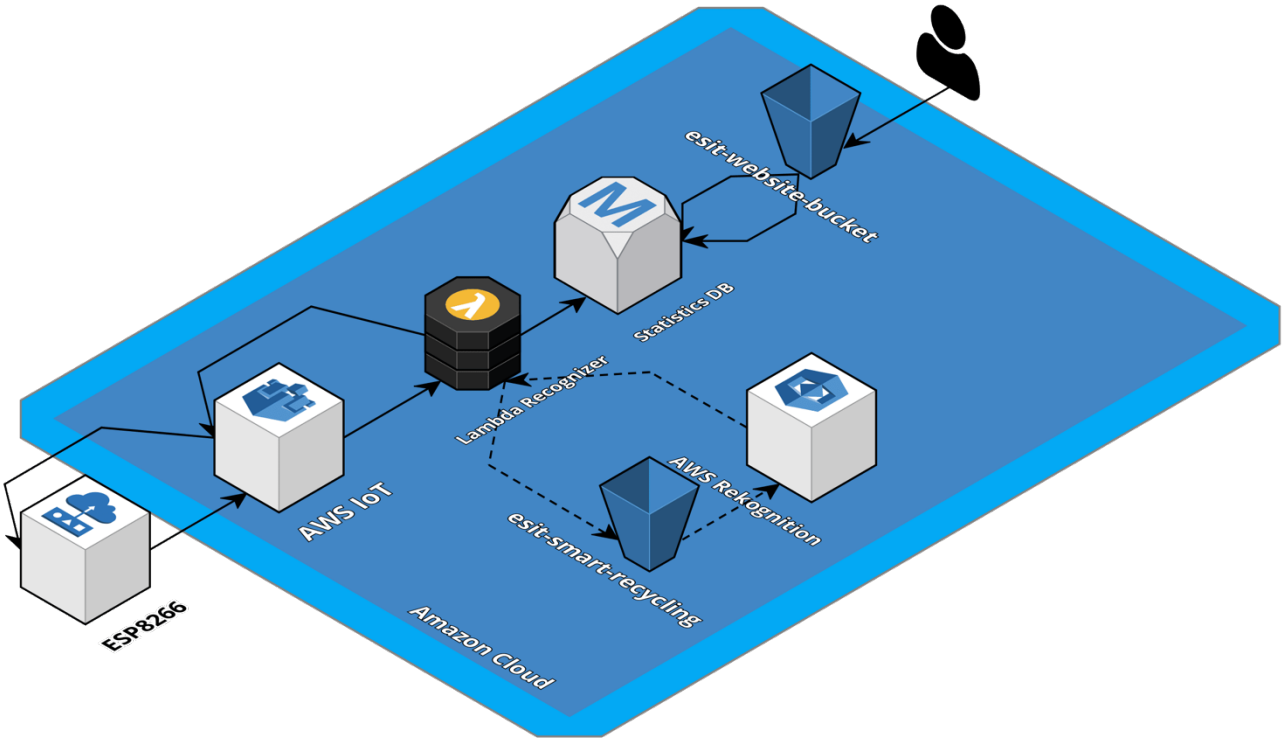
Per concludere, è evidente che, al fine di raggiungere la massima scalabilità, è cruciale progettare un'architettura che consenta l'integrazione di numerosi dispositivi sullo stesso hardware, al fine di ottimizzare l'impiego di componenti.

Uno dei limiti del prototipo è il fatto che esso sia composto da un singolo cassonetto, non permettendo in questo modo lo smaltimento di diversi generi di rifiuti. Naturalmente, questo non sarebbe ammissibile per un sistema su larga scala (ad esempio, per realizzare la gestione dei rifiuti di una città). Estendendo il progetto in questo senso, un possibile miglioramento del prototipo potrebbe prevedere anche l'inserimento di un display che dia all'utente finale feedback e indicazioni su quale sia il cassonetto appropriato per l'oggetto che egli intende smaltire.

Infine, un altro possibile sviluppo del progetto consiste nel perfezionamento dell'algoritmo decisionale, che al momento non prevede nessuna forma di apprendimento automatico o di intelligenza artificiale. Considerando la mole di dati su rifiuti, interazioni e foto provenienti da un sistema su larga scala, è facile presumere che un algoritmo che sfrutti tutte queste informazioni tramite il machine learning possa portare a una riduzione dell'errore nella classificazione dei rifiuti e a una migliore esperienza dell'utente.

Documentazione

Architettura

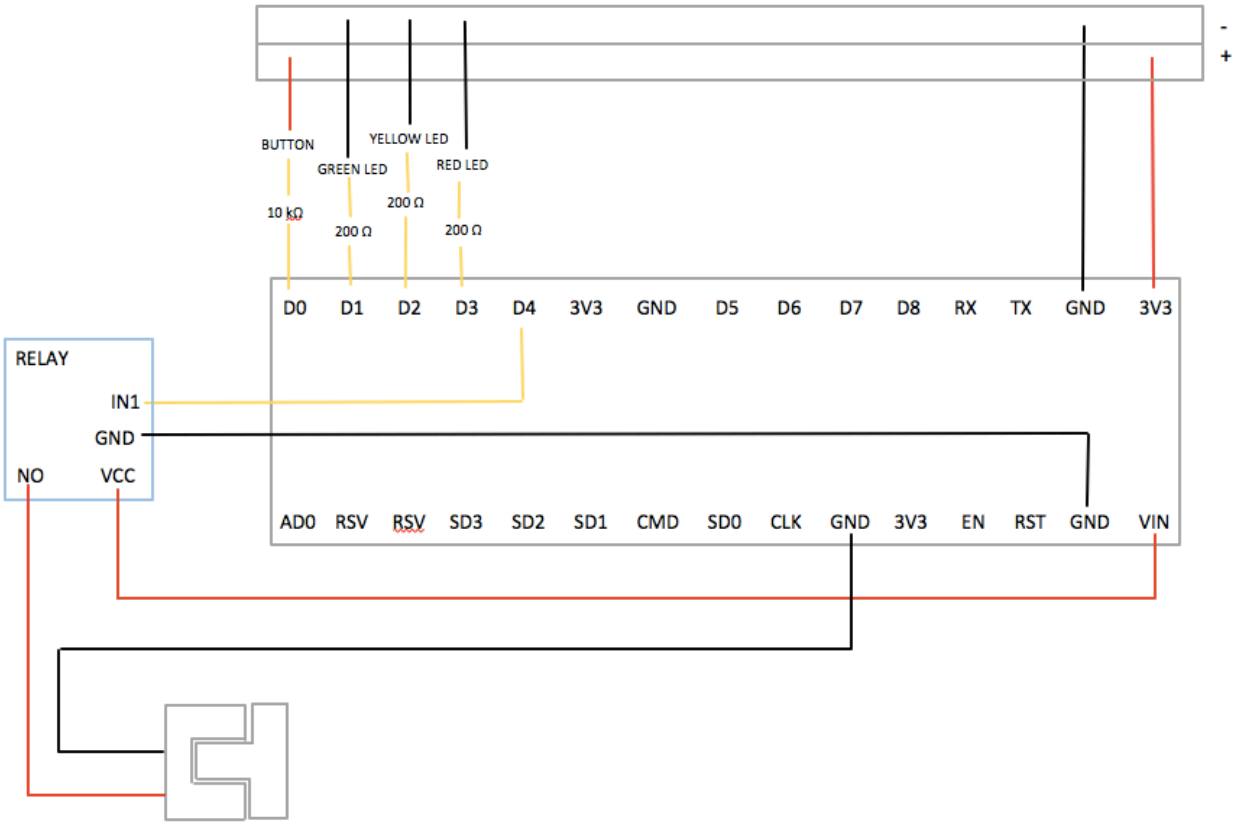


Descrizione del flusso

- La fotocamera acquisisce un'immagine del rifiuto
- L'immagine è codificata in una stringa base64 e inviata ad AWS IoT tramite protocollo MQTT
- AWS IoT contiene una regola (che attiva una AWS Lambda tramite un trigger) ogni volta che riceve dati su un topic pre-determinato sul suo broker
- AWS Lambda si appoggia ad altri due servizi, S3 e Rekognition, tramite la libreria Python boto3:
    - S3 è usato per salvare le immagini inviate dalla fotocamera
    - Rekognition preleva un'immagine da S3 ed estrae le labels relative agli oggetti individuati
- Basandosi sulle labels restituite da Rekognition, AWS Lambda determina la categoria a cui appartiene il rifiuto
- Una volta stabilita la categoria, il digital twin del sensore è aggiornato in base al risultato della valutazione

- Il risultato della computazione dell'algoritmo decisionale è salvato in un database NoSQL su AWS DynamoDB
- Infine:
    - Se il digital twin rappresenta una valutazione positiva, il LED verde si accende e la serratura si apre
    - Altrimenti, il LED rosso si accende e la serratura rimane chiusa
    - Durante la valutazione, il LED giallo lampeggia
- Dalla web app è possibile monitorare le statistiche sulle valutazioni e verificare manualmente la correttezza delle decisioni

Connessioni hardware



Componenti Hardware

- 1 bottone
- 3 LEDs (verde, giallo, rosso)
- 4 resistenze (3 * 200 Ω + 1 * 10k Ω)
- 17 connettori maschio-maschio
- 5 connettori femmina-femmina
- 1 modulo relay
- 1 serratura elettrica

Come riprodurre il progetto

AWS

N.B.: tutte le risorse create devono essere nella stessa AWS Region

DynamoDB

- Dalla console di AWS cliccare su **Services** e cercare **DynamoDB**
- Cliccare su **Create Table**
- Si può scegliere il nome che si vuole per la tabella, ma servirà ricordarlo per la creazione della funzione Lambda e per la web app (se non si vuole modificare nulla, chiamarla **Garbage_items**)
- Come **partition key** indicare **item_id** (di tipo Number)
- Cliccare **Create**

S3

- Dalla console di AWS cliccare su **Services** e cercare **S3**
- Cliccare su **Create Bucket**
- Scegliere un nome per il bucket che conserverà le immagini, ci servirà per la funzione Lambda
- Cliccare su **Create**
- Ora ci servirà un bucket che conterrà il codice della web app, perciò cliccare nuovamente su **Create Bucket**
- Scegliere un nome adeguato
- Clicca su **Next** fino ad arrivare al tab **Set Permissions**
- Sotto **Manage public permissions**, selezionare **Grant public read access to this bucket**
- Ciccare su **Next**, quindi su **Create Bucket**
- Entrare nel bucket appena creato
- Cliccare sul tab **Properties**
- Cliccare su **Enable static website hosting**
- L'endpoint che viene visualizzato sarà l'indirizzo al quale potremo trovare l'app in esecuzione
- Cliccare su **Use this bucket to host a website**
- Nella sezione **Index document** indicare **index.html** e cliccare su **Save**
- Ora, nella cartella **web_app/** dei file del progetto, aprire il file **src/App.js** e aggiornare le righe **9**, **10** , **11** e **14** con le informazioni dell'account corrente e il nome della tabella scelta, controllare inoltre che tutti i servizi creati siano sulla stessa AWS Region
    - La coppia di chiavi può essere generata cercando **IAM** tra i servizi AWS
    - Cliccare su **Users** e cercare il proprio utente (non utilizzare l'utente root)
    - Accedere al tab **Security credentials**
    - sotto **Access keys**, cliccare su **Create access key**
    - verrà scaricato un file .csv contenente le credenziali
- Per continuare è necessario avere **Node.js** installato: https://nodejs.org/en/

- Tornare alla cartella **web_app/**, e tramite il terminale lanciare il comando **npm install** all'interno della cartella
- Una volta finita l'esecuzione lanciare il comando **npm run build**
- Ora dovrebbe comparire una cartella **build**: copiare tutto il suo contenuto all'interno del bucket adibito a sito web statico
- Selezionare tutti i file nel bucket, cliccare su **Actions** e cliccare su **Make public**
- Se dal browser si apre una finestra sull'endpoint mostrato prima si dovrebbe vedere la web app in esecuzione

Lambda

- Dalla console di AWS cliccare su **Services** e cercare **Lambda**
- Cliccare su **Create function**
- Scegliere il nome che si vuole per la funzione, ma ricordarlo dato che ci servirà per AWS IoT Core
- Come **Runtime**, scegliere **Python 3.6**
- Come **Role** selezionare **Create a custom role**: il browser dovrebbe aprire una nuova finestra su AWS IAM
- Cliccare su **View Policy Document**
- Cliccare su **Edit** e copiare tutto il contenuto del file **policies/lambda_policy.json**, quindi cliccare su **Allow**
- Cliccare su **Create function**
- Comprimere tutto il contenuto della cartella **decisional_algorithm/** nei file del progetto in un file .zip
- Nella funzione Lambda, andare nella sezione **Function Code**, selezionare "Upload a zip file" e caricare il file .zip appena ottenuto
- Nella sezione **Environment variables** aggiungere tre variabili:
    - **BUCKET_NAME**: il nome del **bucket** che conterrà le immagini
    - **RESPONSE_TOPIC**: il nome del **topic MQTT** dove verrà pubblicato il risultato dell'elaborazione (il progetto utilizza **esiot_out**, se se ne vuole utilizzare un altro si dovrà modificare il codice dello sketch arduino)
    - **TABLE**: contiene il nome della tabella dynamo
- Clicca su **Save**

IoT Core

- Dalla console di AWS cliccare su **Services** e cercare **IoT Core**
- Cliccare su **Settings** e ricordare l'endpoint

- *Creazione policy per l'oggetto*
    - Cliccare su **Security** e quindi su **Policy**
    - Cliccare su **Create policy**
    - Per **Operation**, scrivere **iot:***
    - Per **ARN resource**, scrivere ***
    - Spuntare **Enable**
    - Cliccare su **Create**

- *Creazione oggetto*
    - Cliccare su **Manage** e quindi su **Things**
    - Cliccare su “Create”, quindi su **Create a single thing**
    - Scegliere un nome per l’oggetto (il progetto usa **esiot_test**, se se ne vuole utilizzare un altro si dovrà modificare il codice dello sketch arduino)
    - Cliccare su **Next**
    - Cliccare su **Create certificate**
    - Scaricare tutti i file creati, sono un **certificato**, una **chiave pubblica** e una **chiave privata**
    - Scaricare inoltre il **certificato CA**
    - Tenere da parte tutti i certificati, serviranno per il funzionamento dello sketch
    - Cliccare su **Activate**
    - Cliccare su **Attach policy**
    - Spuntare la policy creata precedentemente
    - Cliccare su **Register thing**
- *Creazione rule*
    - Cliccare su **Act**, quindi su **Create rule**
    - Scegliere un nome e una descrizione adeguata
    - Nel campo **Message source** aggiungere **SELECT * FROM esiot_in**
        - **esiot_in** è il topic usato nel progetto, può essere usato un qualsiasi altro topic ma il codice dello sketch arduino dovrà essere modificato di conseguenza
    - Nel campo **Set one or more actions** cliccare su **Add action**
    - Selezionare **Invoke a lambda function passing the message data**
    - Cliccare su **Configure action**
    - Nel campo **Function name**, cliccare su **Select** e scegliere dall’elenco la funzione Lambda creata precedentemente
    - Cliccare su **Add action**
    - Cliccare su **Create rule**

Arduino IDE

- Da **Sketch > #include libreria > Gestore librerie** installare **ArduinoJson**
    - Il progetto usa la versione 5.12.0
- Installare inoltre la libreria **PubSubClient**
- Per la gestione dei certificati sarà necessario installare un tool di gestione del file system (SPIFFS)
- Andare su https://github.com/esp8266/arduino-esp8266fs-plugin/releases/tag/0.2.0 e scaricare ESP8266FS-0.2.0.zip
- Decomprimere lo zip e mettere il contenuto all’interno della cartella tools nella root della directory di installazione di **Arduino IDE**
    - Alla fine dovrebbe esistere un path del genere: **[home_dir]\Arduino\tools\ESP8266FS\tool\esp8266fs.jar**
    - Se la cartella **tools** non esiste, è sufficiente crearla
- Riavviare **Arduino IDE**

- Se l’operazione è andata a buon fine, sotto il menù **Tools** dovrebbe comparire **ESP8266 Sketch Data Upload**
- Ora bisogna convertire i file scaricati da AWS IoT nei formati corretti, sarà necessario avere **openssl**
- Ci interessa convertire il certificato CA, il certificato dell’oggetto e la chiave privata
- Sul certificato CA usare:
  openssl x509 -in nome_file_certificatoCA -out certCA.der -outform DER
- Sul certificato dell’oggetto usare:
  openssl x509 -in nome_file_certificato -out cert.der -outform DER
- Sulla chiave privata usare:
  openssl rsa -in nome_file_chiave_privata -out private.der -outform DER
- Per ogni comando, inserire opportunamente il nome del file richiesto
- Nella cartella contenente lo sketch, creare una cartella **data** e inserire i 3 file con estensione .der
- Caricare quindi lo sketch su Arduino IDE, e cliccare su **Tools > ESP8266 Sketch Data Upload**