Luca Ceolin - 144199
Anastasia Di Carlo - 142653
Riccardo Lunardi - 142222
Sebastian Mattias Rodriguez - 144876

Relazione progetto Internet of Things
A.A. 2020/2021

Introduzione al progetto

Durante lo scorso anno, molti commercianti sono stati costretti ad avere a che fare con una serie di regole che limitava il numero di persone all'interno dei loro negozi a causa del COVID-19. Questo poteva diventare un problema in quanto, negli orari di frequentazione più alti, il conteggio delle persone può essere complicato, se non impossibile per i grandi negozi.

Partendo da questo problema, abbiamo deciso di sviluppare un sistema che permetta il conteggio di persone all'interno di un luogo.

Il nostro obiettivo è anche quello di fare in modo che il sistema che sia il più modulare e personalizzabile possibile.

Abbiamo realizzato un breve video per dimostrare il funzionamento del sistema, che è visibile a questo link: https://youtu.be/JG6iEjLWJoM.

All'interno della repository PeopleCounterIoT si trova tutto il codice sviluppato, il video dimostrativo e le stesse immagini, in definizione più alta, che abbiamo inserito in questa relazione.

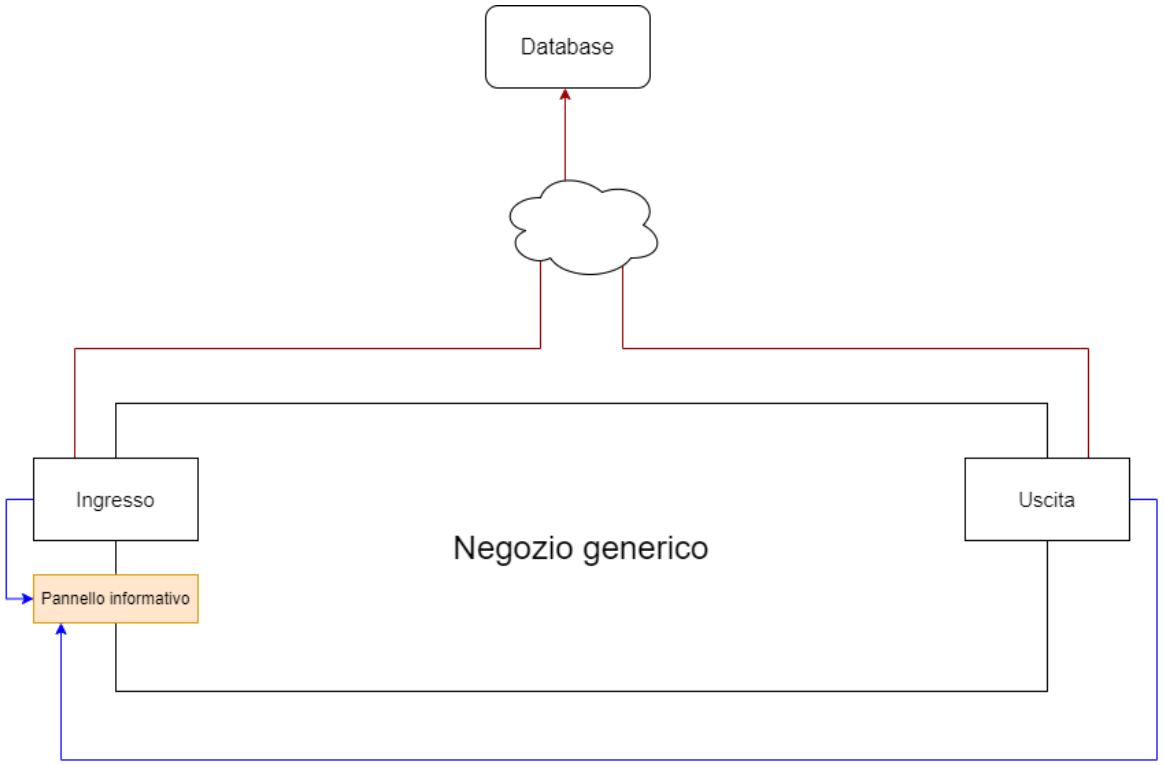
Requisiti del sistema

I requisiti che ci siamo imposti sono stati i seguenti:

- Il sistema, tramite dei dispositivi posizionati negli ingressi/uscite, ad ogni entrata/uscita di una persona, dovrà generare un dato, che conterrà il numero di persone passate.
- Il dato generato verrà inviato ad un database, il quale conterrà tutti i dati dei contapersone. L'aggregazione di questi dati permetterà di creare report sulle presenze in un certo luogo e altri tipi di statistiche.
- Il dato in via opzionale potrà essere condiviso, anche localmente, ad un sistema informativo. Quest'ultimo potrà comunicare agli utenti, ad esempio, quante persone sono già all'interno della struttura, o allertare qualcuno quando il numero di persone entrate è superiore ai limiti previsti. D'ora in poi ci riferiremo a questo sistema come *sistema informativo.*

In generale, vogliamo fare in modo che il sistema sia il più possibile indipendente dall'implementazione del contapersone e del sistema informativo, in modo tale che chiunque voglia implementare, ad esempio, il suo contapersone personalizzato, lo possa fare scrivendo meno codice possibile.

Schema di funzionamento



Lo schema rappresenta un luogo generico, nella quale sono presenti rispettivamente un ingresso ed un'uscita. È da ricordare che, in generale, in base a come funziona il contapersone, un'entrata potrebbe anche fungere da uscita.

I contapersone, al passaggio di una persona, inviano il loro dato ad un server, che può essere locale o remoto, che immagazzinerà i dati raccolti dai contapersone. Tale server sarà accessibile dagli admin per poter effettuare vari tipi di report.
Lo stesso dato verrà inviato, in questo caso, anche al sistema informativo, il quale mostrerà tramite uno schermo il numero di persone presenti all'interno del locale. Questo ultimo tipo di comunicazione avverrà sempre all'interno della rete locale, in modo da non consumare banda verso Internet.

Scelte implementative

Abbiamo deciso di implementare una nostra possibile soluzione in questa maniera:

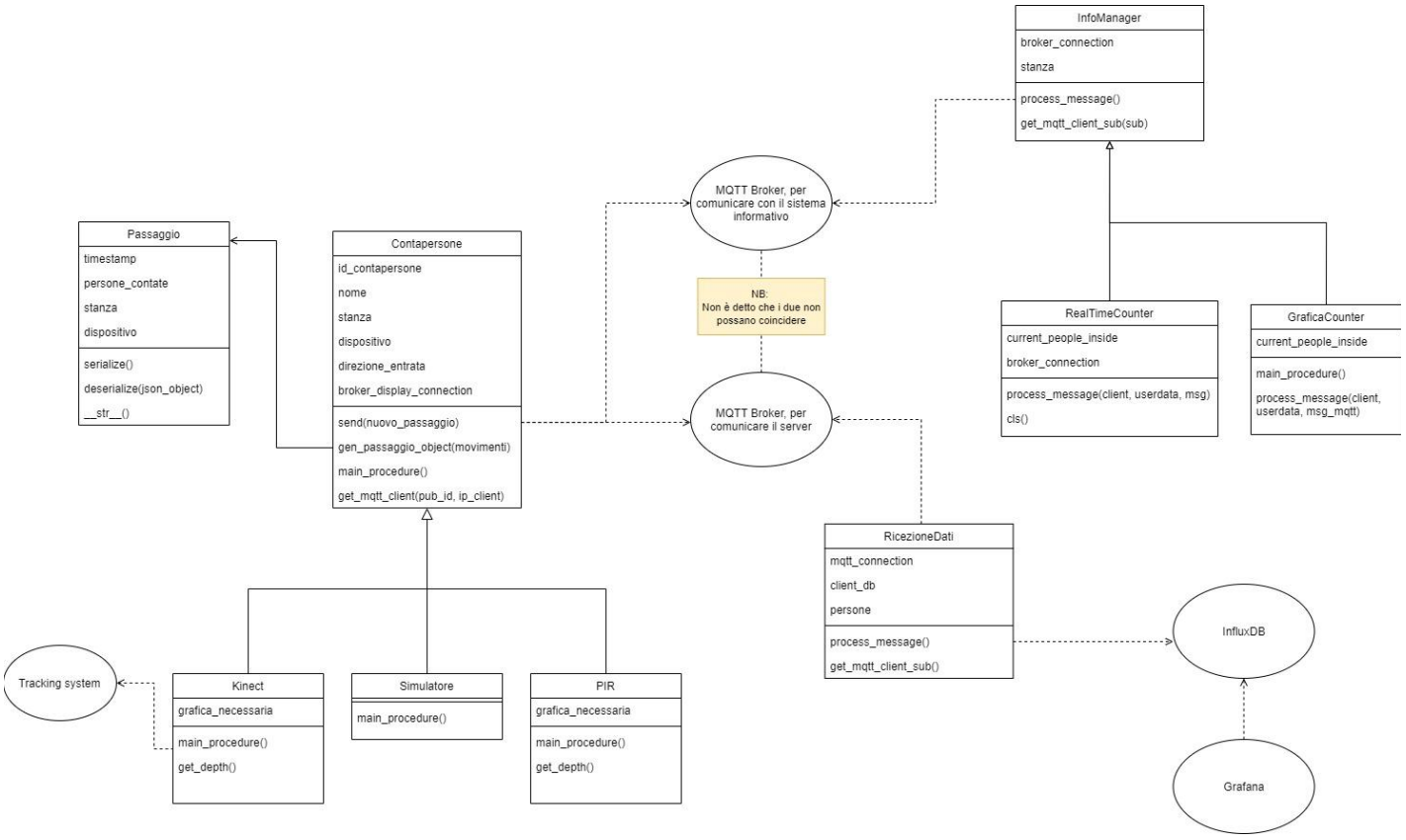
- Utilizzeremo 2 contapersone per monitorare il numero di persone all'interno di una stanza.
    - Un contapersone sarà realizzato usando un Raspberry Pi 3 Model B insieme ad un Kinect per XBox 360. Potrà essere messo in prossimità di una porta che permette sia l'entrata che l'uscita.
    - L'altro contapersone sarà realizzato con un Raspberry Pi 3 Model B ed un sensore ad infrarossi passivo (PIR), che, potendo rilevare solo il movimento,

potrà contare le persone soltanto verso una direzione, quindi o in entrata o in uscita.

Per entrambi abbiamo utilizzato un Raspberry Pi 3 Model B perché era l'unico dispositivo che avevamo a disposizione.

- Un ingresso sarà dotato di uno schermo che mostra il numero di persone all'interno della stanza. Nel nostro caso useremo un laptop.
- I dati verrano inviati ad un server, che li salverà in un database InfluxDB, con il quale successivamente sarà possibile effettuare report e statistiche tramite Grafana.
- La comunicazione tra i contapersone, il sistema informativo e il server remoto avverrà tramite il protocollo MQTT: abbiamo deciso di utilizzarlo perché, nel caso di un luogo affollato, è meno dispendioso mantenere aperta la connessione tra server e client che aprirla ogni volta con che una persona viene contata, cosa che sarebbe successa se avessimo usato HTTP.

Schema UML



Contapersone

I contapersone vengono creati implementando la classe “Contapersone”, la quale contiene già la maggior parte delle caratteristiche necessarie ad un contapersone.
L’inizializzazione avviene tramite un ID, che fa riferimento ad uno specifico contapersone. salvato nel file “configurazione.json”: questo permette di definire in modo completo i contapersone esternamente allo script Python, anche con informazioni che non sono utilizzate dagli script nella versione attuale.

Passaggio

Il passaggio di una persona in un ingresso o in un uscita è rappresentato da un oggetto Passaggio, che contiene le seguenti informazioni:

- Timestamp del passaggio
- Numero di persone contate
- Stanza a cui il contapersone fa riferimento
- Dispositivo contapersone utilizzato

La classe contiene dei metodi di deserializzazione e serializzazione, in modo da poter inviare e ricevere oggetti JSON tramite MQTT.
L’oggetto Passaggio da inviare viene creato nelle classi che implementano “Contapersone”, ma il suo invio avviene tramite lo stesso metodo ereditato chiamato “send()”.

Contapersone realizzato con Kinect

Il primo dei due contapersone, inizialmente, voleva essere realizzato tramite un Raspberry Pi Model B e una normale videocamera, che, posizionata sopra una porta, identificasse le persone in transito.
Abbiamo provato ad utilizzare una MobileNet Single Shot Detector (SSD) per la rilevazione delle persone, ma si è rivelata fallimentare, sia in quanto il Raspberry non era abbastanza potente a livello computazionale, sia perché, anche facendo elaborare il feed video ad un computer normale, le persone non venivano correttamente contate la maggior parte delle volte.
La seconda idea è stata quella di processare le immagini tramite la tecnica di “background subtraction”, che consente di rilevare movimenti nell’immagine, quando questa normalmente ha uno sfondo fisso. Il risultato non è stato soddisfacente, in quanto venivano contati erroneamente dei passaggi di persone. Se, ad esempio, per qualche istante la luce cambiava nella stanza, venivano rilevati oggetti che non erano presenti perché secondo la tecnica della background subtraction, lo sfondo fisso era cambiato e quindi c’era del movimento.
La soluzione finale è stata quella di usare un videocamera di profondità, nel nostro caso quella del Kinect 1.0. In questo modo abbiamo potuto:

- Semplificare il processo di background subtraction: lo sfondo non cambierà più per motivi esterni (es. illuminazione) perché la videocamera di profondità utilizza una propria illuminazione ad infrarossi.

- Possiamo, elaborando l'immagine con la tecnica del thresholding, isolare la profondità necessaria: in questo modo eliminiamo tutti gli oggetti che sono troppo alti o troppo bassi per essere considerati una persona, eliminando molti falsi positivi.
- Essendo la background subtraction computazionalmente meno pesante rispetto all'utilizzo della SSD, possiamo tenere l'elaborazione interna al Raspberry: in questo modo non è necessario avere un server locale per processare i dati o un feed video costante ad un server esterno: entrambi i metodi potrebbero andare ad impattare negativamente sulla banda di rete disponibile.

Per poter usufruire della videocamera di profondità tramite Python abbiamo utilizzato la libreria e i driver "freenect".
Per capire se una persona sta entrando o uscendo, dobbiamo innanzitutto rilevarla (detection) e poi tracciarne i movimenti (tracking). Dopodiché, per capire la direzione, viene fatta la media delle posizioni della persona sull'asse Y: se la posizione corrente è maggiore di quella media e l'individuo si trova nella metà superiore della vista del Kinect, allora la persona sta entrando, altrimenti se la posizione è minore di quella media e l'individuo si trova nella metà inferiore, la persona sta uscendo. Nel caso la direzione di entrata e uscita sia invertita, perché può cambiare in base all'orientamento del Kinect, è possibile impostarla dal file configurazione.json.

Detection

La detection, come già detto, viene fatta tramite la tecnica di image processing chiamata "background subtraction". Nello specifico, usiamo la seguente funzione di OpenCV:

```
cv2.createBackgroundSubtractorKNN(history=70, dist2Threshold=200, detectShadows=False)
```

La funzione utilizza l'algoritmo "K-nearest neighbours background subtraction" per rimuovere lo sfondo. I parametri sono stati scelti in base ad una serie di test che abbiamo eseguito in fase di sviluppo.
L'immagine di profondità restituita dal Kinect è un'immagine in scala di grigi, dove ogni pixel è rappresentato da un valore che va da 0 a 1024.
Il thresholding, utilizzato per eliminare falsi positivi, viene fatto sull'immagine restituita dal Kinect nel range tra 515 e 915. Il range è stato scelto dopo una serie di prove e test e ci è risultato essere il più consono, data l'altezza e l'inclinazione dello stesso.
Dopo il thresholding, utilizziamo la funzione *cv2.findContours()* per ottenere i contorni degli oggetti rilevati.
Successivamente alla detection, lo script tenta di tracciare l'oggetto rilevato tramite i contorni per capirne la direzione.

Tracking

Gli oggetti vengono tracciati tramite il "centroid tracking algorithm". Dati dei contorni, viene prima calcolato un rettangolo che contenga tale contorno, poi viene trovato il centroide, che è il punto centrale del rettangolo.

A questo punto, dati i centroidi di un frame e i centroidi del frame successivo, l'algoritmo assume che i centroidi più vicini tra loro siano correlati, cioè che siano lo stesso oggetto, ma in una posizione diversa.
Gli oggetti che rimangono per troppo tempo fermi nello stesso punto o che si muovono troppo velocemente vengono eliminati dalla lista degli oggetti in fase di tracciamento: questo viene effettuato per evitare falsi positivi.

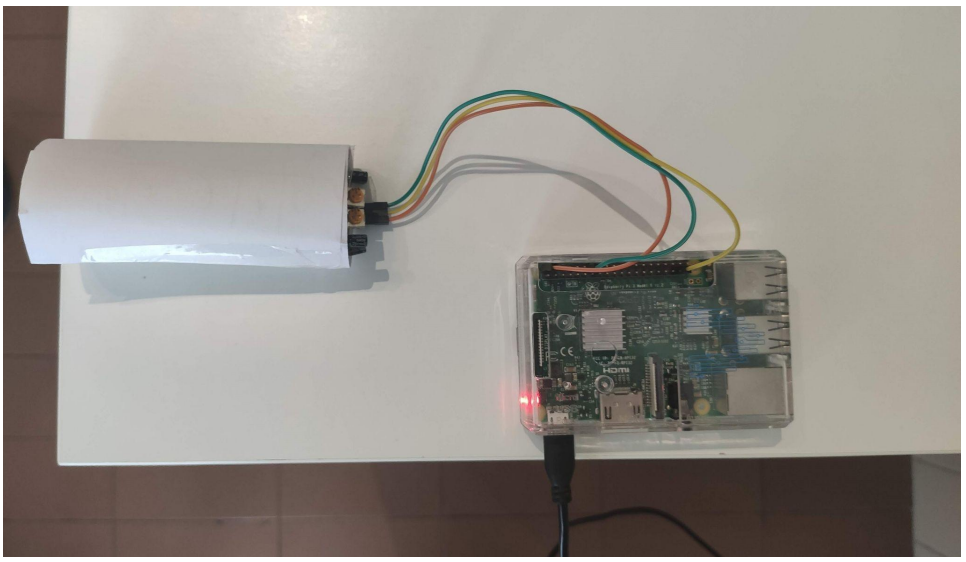
Entrambi gli algoritmi di detection e di tracking sono stati adattati da noi a partire da quelli di PyImageSearch.com:

- OpenCV People Counter
- Simple object tracking with OpenCV

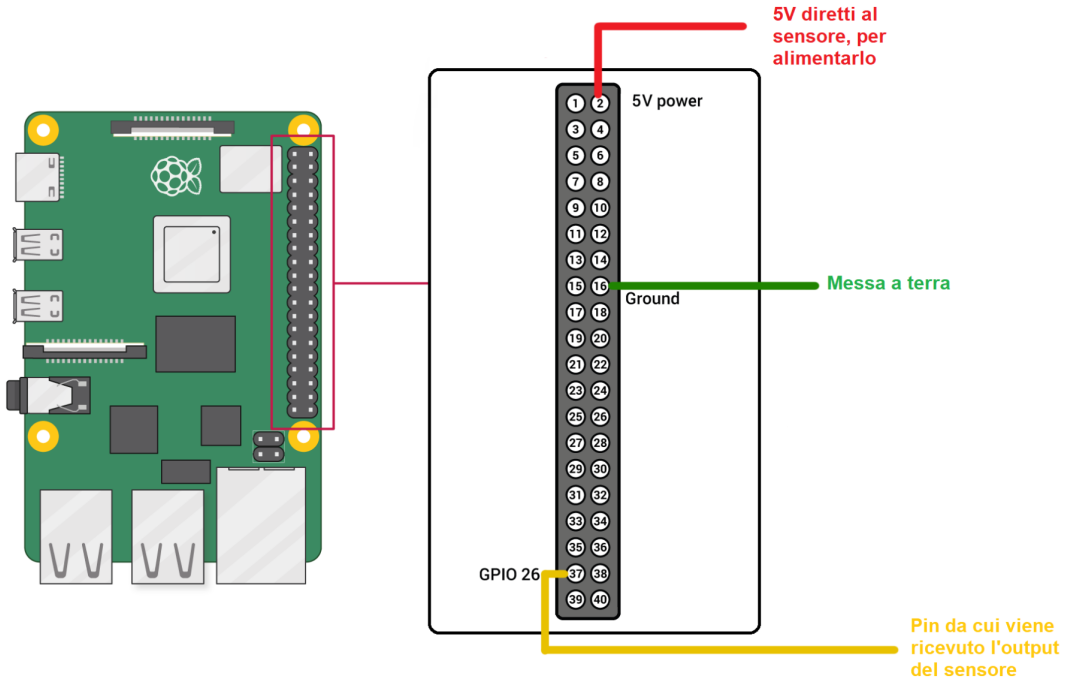
Contapersone realizzato con PIR

Un sensore PIR (Passive InfraRed) è un sensore infrarossi che rileva variazioni di radiazione infrarossa irradiata dagli oggetti nel suo campo visivo. Viene utilizzato comunemente come rilevatore di movimento.
Questo tipo di contapersone potrà essere usato solo per porte che permettono esclusivamente l'entrata o l'uscita: il sensore PIR non è in grado di capire la direzione della persona.
Il sensore da noi utilizzato è il Aukru HC-SR501, che verrà controllato tramite i GPIO del Raspberry.
Abbiamo inserito il PIR all'interno di un tubicino di carta, in modo tale da ridurre sensibilmente il campo visivo del sensore, essendo questo molto alto.
Nonostante riesca a contare correttamente le persone, probabilmente vista anche l'economicità del sensore, quando viene rilevato del movimento il PIR si blocca per qualche secondo in uno stato di eccitazione, trascurando altri eventuali passaggi di persone.

Il conteggio viene effettuato assegnando alla funzione `self.pir.when_motion` una seconda funzione, `movimento_rilevato`, che effettivamente invia al broker MQTT l'oggetto Passaggio.



Setup del Raspberry e del sensore PIR



Schema di collegamento

Realizzazione della GUI per il contapersone

InfoManager

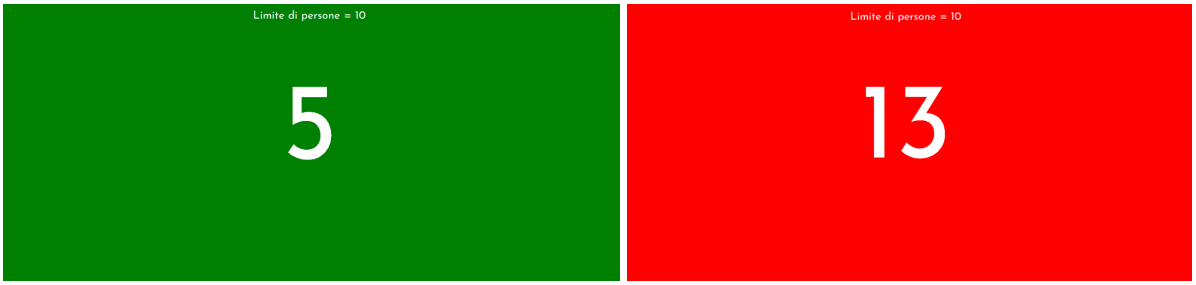
InfoManager è una classe generica che dovrebbe essere implementata da tutti gli script che vogliono realizzare un qualche tipo di contatore di persone in tempo reale da posizionare, per esempio, all'entrata della stanza. Ricordiamo che quest'ultimo fa riferimento al *sistema informativo* di cui abbiamo discusso brevemente prima. La classe gestisce solamente la connessione al broker, che deve girare nella stessa macchina su cui gira il sistema informativo: il resto è lasciato all'implementatore della classe.

Tramite InfoManager abbiamo realizzato due esempi:

- RealTimeCounter.py, che visualizza il numero di persone presenti in una stanza tramite la riga di comando;
- GraficaCounter.py, che è un contatore live più user-friendly, che comunica agli utenti, tramite colori e numeri, la situazione nella stanza e il limite di persone consentito.

GraficaCounter

Per realizzare un'interfaccia che informasse le persone che entrano in una stanza di quante ce ne fossero e quante ne potessero ancora entrare, abbiamo utilizzato una libreria di Python chiamata "Tkinter", che permette di creare interfacce grafiche. Il programma segna in tempo reale il numero di persone all'interno di una stanza e il limite di persone consentite. Lo sfondo della GUI è verde quando è possibile che una nuova persona entri, mentre diventa rosso quando nella stanza c'è un numero maggiore di persone rispetto ad un limite. Il limite può essere modificato attraverso il codice.
La componente principale è il *Canvas*, dove cui, tramite il metodo *create_text*, vengono disegnati i numeri in continuo aggiornamento. Al Canvas viene aggiunto anche il componente *Message*, che si occupa di comunicare il numero di persone massimo all'interno della stanza.



Esempi di schermate del sistema informativo

Invio dei dati al server

La comunicazione tra i contapersone e il server remoto è stata implementata con uno script in Python chiamato "RicezioneDati.py", che contiene:

- Costruttore;
- Due metodi (process_message, get_mqtt_client_sub).

Il costruttore istanzia la connessione con il database InfluxDB, il client MQTT e inizializza un dizionario vuoto, dove verranno registrate le persone presenti per ogni stanza.
La connessione al database viene fatta utilizzando il metodo `InfluxDBClient` che specifica la porta del server alla quale connettersi. Per istanziare la variabile che sfrutta i dati ricevuti dal client tramite MQTT, si utilizza un metodo implementato all'interno dello script chiamato "get_mqtt_client_sub". Il metodo ha lo scopo di inizializzare un client MQTT, con il supporto della libreria "paho-mqtt", al quale viene assegnata una funziona da eseguire se il client risulta iscritto ad uno specifico topic. La funzione eseguita, come nel caso precedente, è

implementata all'interno dello script e si chiama "process_message". Questa funzione riceve in input alcuni parametri obbligatori, tra cui *stringa_json*, che contiene il messaggio contenente i dati del contapersone. La stringa verrà convertita in formato JSON e successivamente inviata al database InfluxDB che risiede nel server remoto.

InfluxDB

InfluxDB è stato il time-series database a cui ci siamo affidati. Abbiamo deciso di utilizzarlo visto che le informazioni che raccogliamo si possono rappresentare comodamente in una serie di dati temporali provenienti da più sensori.
Per comodità, come visto a lezione, abbiamo deciso di utilizzare la versione di InfluxDB 1.8 tramite Docker. I comandi per l'installazione sono stati i seguenti:

- `docker pull influxdb:1.8`

- `docker run -d -p 8086:8086 -v influxdb:/var/lib/influxdb --name influxdb influxdb:1.8`

Per la creazione del database e per le query di test abbiamo utilizzato la funzionalità CLI di InfluxDB tramite il comando `docker exec -it influxdb influx`.

Report e statistiche su Grafana

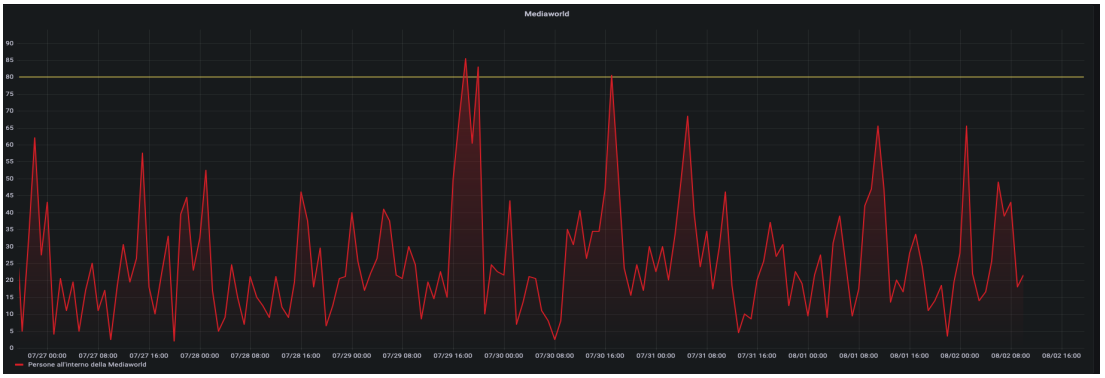
I dati utilizzati per mostrare le statistiche sul software Grafana sono stati prodotti da uno script specifico, chiamato "Simulatore.py", che inviava al server una serie di dati artificiosi provenienti da un sensore. Il dato contenente le informazioni delle persone entrate ed uscite veniva generato casualmente in un intervallo che veniva deciso arbitrariamente da noi. Sia l'installazione, che l'uso di Grafana, sono avvenuti attraverso Docker.
Come caso di studio, abbiamo preso in considerazione diverse "stanze" che possono essere presenti all'interno di un centro commerciale (come supermercati, ristoranti, negozi di elettronica/elettrodomestici).
Grafana permette di creare dashboard che si interfacciano ad un database, nel nostro caso ad InfluxDB. Ciò consente di effettuare delle query al database attraverso la GUI direttamente sul browser per creare grafici con i dati che si vogliamo esaminare. Degna di nota è anche la possibilità data da Grafana di inserire una riga orizzontale nel grafico, chiamata "Threshold". Questa riga rappresenta una sorta di "massimo" o di limite dei dati: nel nostro caso può essere impostata in modo tale da poter vedere a colpo d'occhio quali negozi non hanno rispettato il limite di persone consentito.

Per la prima serie di query, è stata fatta la media di persone presenti all'interno dei negozio ad ogni ora, successivamente distribuite negli ultimi 7 giorni:

- Query Mediaworld:

```
SELECT mean("persone_contate")
FROM "contapersone"
WHERE ("stanza" = 'Mediaworld') AND $timeFilter
GROUP BY time(1h) fill(null)
```

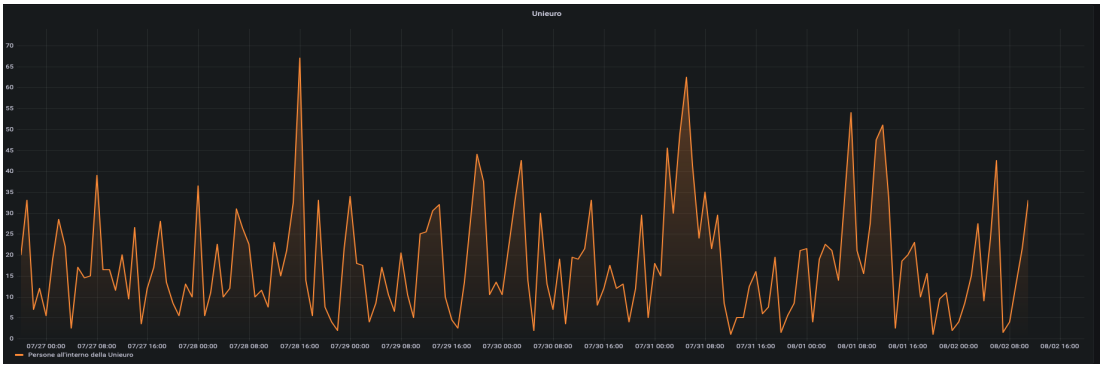
Grafico:



- Query Unieuro:

```
SELECT mean("persone_contate")
FROM "contapersone"
WHERE ("stanza" = 'Unieuro') AND $timeFilter
GROUP BY time(1h) fill(null)
```

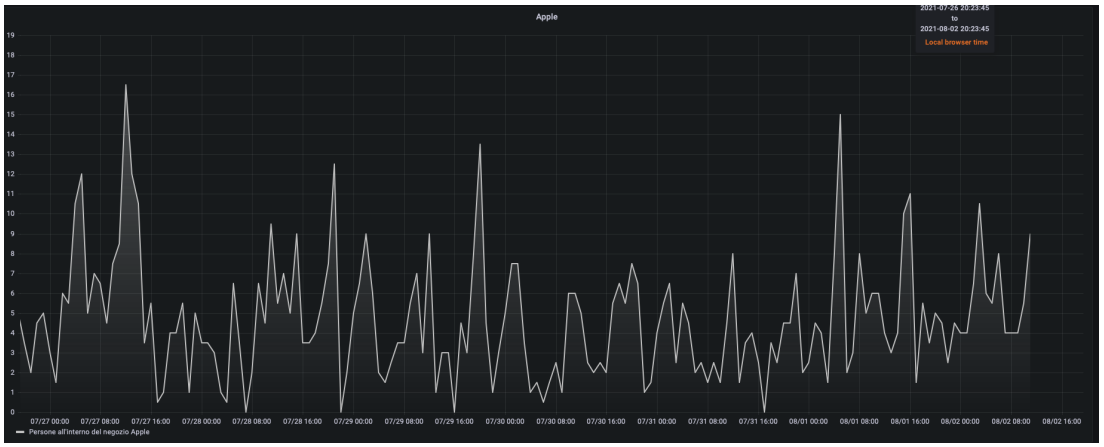
Grafico:



- Query Negozio Apple:

```
SELECT mean("persone_contate")
FROM "contapersone"
WHERE ("stanza" = 'Apple') AND $timeFilter
GROUP BY time(1h) fill(null)
```

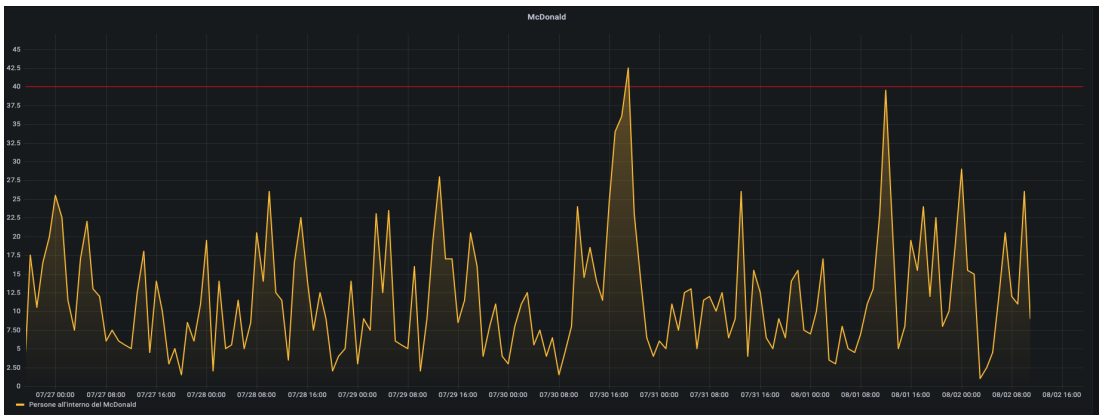
Grafico:



- Query McDonald:

```
SELECT mean("persone_contate")
FROM "contapersone"
WHERE ("stanza" = 'McDonald') AND $timeFilter
GROUP BY time(1h) fill(null)
```

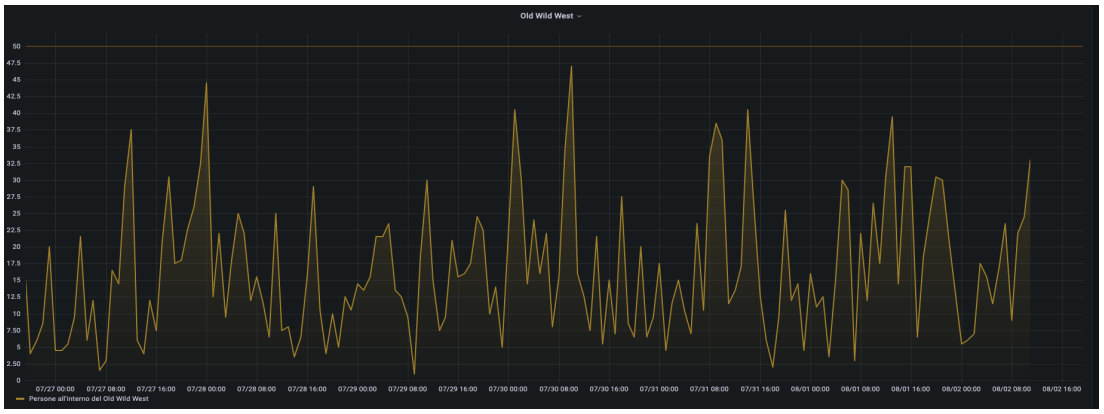
Grafico:



- Query Old Wild West:

```
SELECT mean("persone_contate")
FROM "contapersone"
WHERE ("stanza" = 'Old Wild West') AND $timeFilter
GROUP BY time(1h) fill(null)
```

Grafico:



- Query Crai:

```
SELECT mean("persone_contate")
FROM "contapersone"
WHERE ("stanza" = 'Crai') AND $timeFilter
GROUP BY time(1h) fill(null)
```

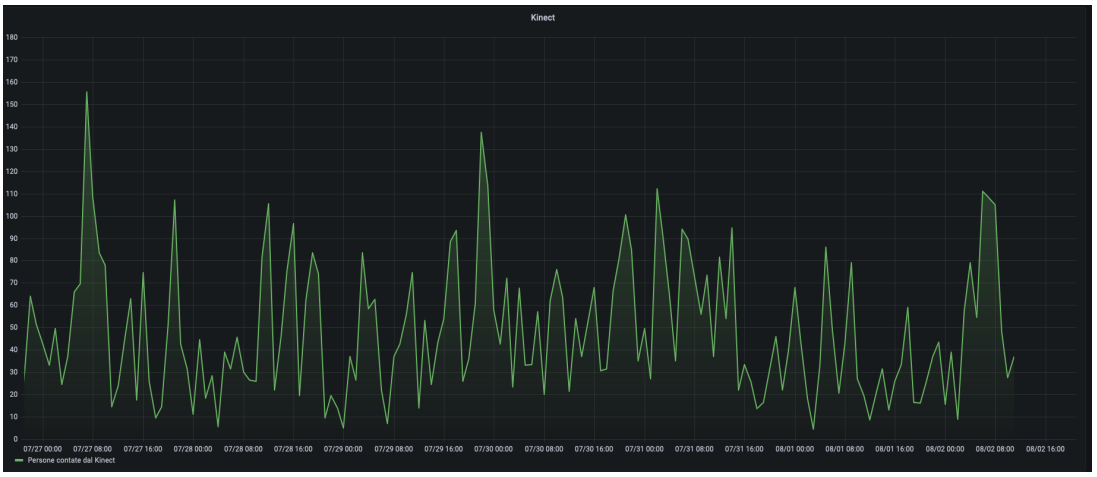
Grafico:

Per la seconda serie di query, abbiamo graficato il conteggio delle persone sia di un ipotetico Kinect, che di un PIR. Similmente alla prima serie di query, è stata fatta la media di persone presenti contate ad ogni ora e successivamente distribuite negli ultimi 7 giorni. Inoltre, abbiamo simulato una situazione nella quale il sensore PIR è soggetto a malfunzionamento dovuto ad un guasto.

- Query del conteggio delle persone effettuato dal Kinect:

```
SELECT mean("persone_contate")
FROM "contapersone"
WHERE ("dispositivo" = 'Kinect') AND $timeFilter
GROUP BY time(1h) fill(null)
```

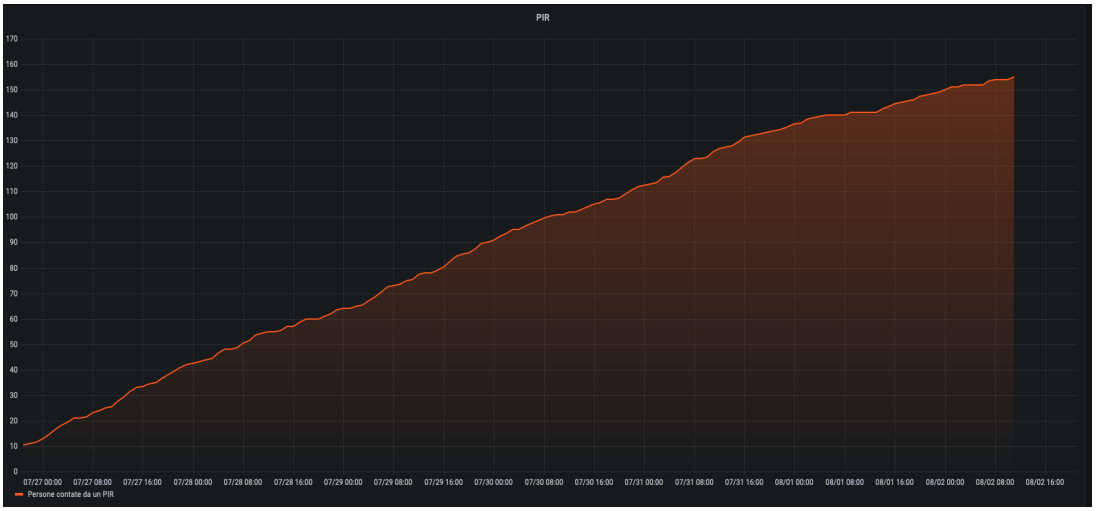
Grafico:



- Query del conteggio delle persone effettuato dal PIR:

```
SELECT mean("persone_contate")
FROM "contapersone"
WHERE ("dispositivo" = 'PIR') AND $timeFilter
GROUP BY time(1h) fill(null)
```

Grafico:

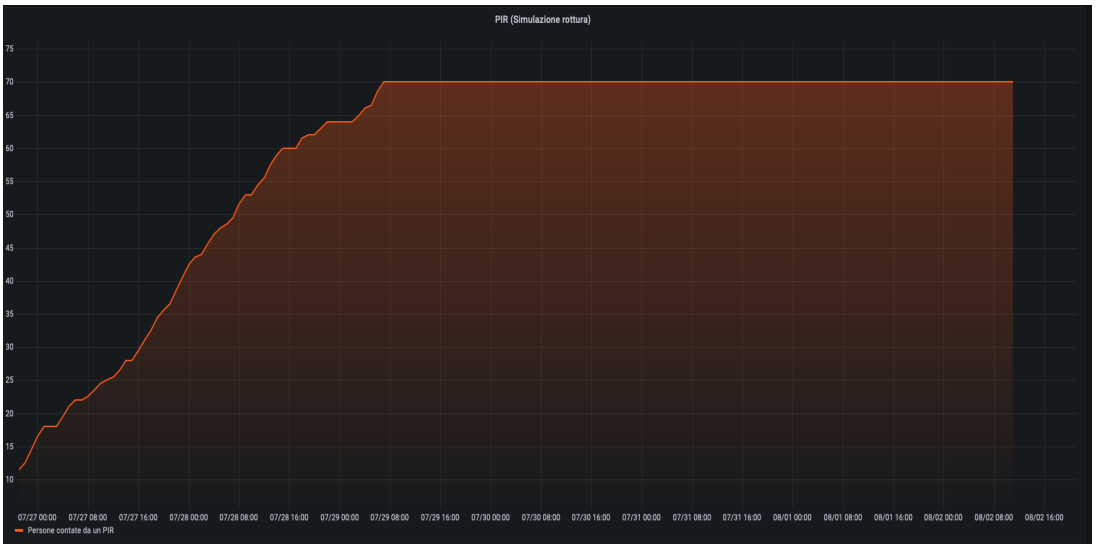


Il grafico mostra solo una crescita verso l'alto. Questo è dovuto al fatto che il PIR conta solo le persone che entrano o escono, non entrambi.

- Query del conteggio delle persone effettuato dal PIR guasto::

```
SELECT mean("persone_contate")
FROM "contapersone"
WHERE ("dispositivo" = 'PIR (Simulazione rottura)') AND $timeFilter
GROUP BY time(1h) fill(null)
```

Grafico:



In questo caso si osserva che in una determinata giornata la curva smette di crescere. Ciò ci permette di dire che il sensore sia guasto: i dati visualizzati nel tempo non cambiano mai, ma restano fissi, indicando che il PIR non ha rilevato nessuno per giorni. Un ipotetico gestore potrebbe accorgersi facilmente e relativamente velocemente del guasto, per poi intervenire tempestivamente nella sua risoluzione.

Le immagini sono visibili in maggiore definizione nella repository GitHub.

Note sul video dimostrativo

Dobbiamo fare alcune precisazioni sul video che abbiamo fatto per dimostrare il funzionamento del sistema:

- Il Kinect, in quel caso, è stato collegato ad un laptop e non al Raspberry. Siamo stati costretti ad usare un altro computer perché il Raspberry disponibile era solo uno ed era necessario per connettere il sensore PIR, visto che quest'ultimo necessitava di essere collegato ai GPIO. Questo caso a parte, tutto lo sviluppo e tutti i vari test sono stati svolti con il Kinect collegato al Raspberry.
- Nella schermata di debug del Kinect, in basso a destra, si vede che i contatori aumentano in maniera invertita: quando qualcuno esce, viene incrementato il numero di persone entrate e viceversa. Si tratta solo di bug grafico, in quanto si può vedere, dal sistema informativo, che il numero di persone totali all'interno della stanza è calcolato correttamente.

Contributi dei componenti del gruppo

Il lavoro per realizzare il progetto è stato diviso nel seguente modo:

- Luca Ceolin:
    - Test dei contapersone (Kinect e PIR).
    - Sviluppo delle classi implementative di InfoManager.py (GraficaCounter.py e RealTimeCounter.py).
- Anastasia Di Carlo:
    - Installazione e configurazione di Grafana.
    - Sviluppo di Simulatore.py e generazione dei dati.
- Riccardo Lunardi:
    - Design dell'architettura di sistema.
    - Sviluppo e test dei contapersone (Kinect e PIR).
    - Sviluppo delle classi generiche Contapersone.py, InfoManager.py e Passaggio.py.
- Sebastian Mattias Rodriguez:
    - Installazione e configurazione di InfluxDB.
    - Sviluppo e test di RicezioneDati.py.