

# Supermarket Occupancy Management System

Sistema smart per monitoraggio e predizione del livello di  
occupazione dei supermercati di una catena

---

# Sommario

01

## Introduzione

Motivazioni del progetto  
e funzionalità del sistema

02

## Architettura

Panoramica  
dell'architettura  
hardware e software

03

## Componenti

Analisi dettagliata dei  
componenti del sistema

04

## Prototipo

Implementazione delle  
funzionalità principali di SOMS

---



# 01

# Introduzione

Motivazioni del progetto e funzionalità del  
sistema

# Motivazioni (1)

I supermercati – in generale i grandi esercizi commerciali – sono molto spesso afflitti da **inefficienze** che ne limitano i potenziali profitti, in particolare:

- **Scarsa reattività e inflessibilità nell'allocazione del personale di cassa**, che può determinare tempi di servizio molto lunghi.
- **Lunghe code alle casse**, che possono generare insoddisfazione nei clienti.
- **Nessun feedback immediato e diretto sul livello di occupazione**, che permetterebbe di predirne l'evoluzione nel tempo.

Inoltre queste inefficienze sono peggiorate con la pandemia.

# Motivazioni (2)

- **SOMS** (Supermarket Occupancy Management System) è un **sistema smart, economico e altamente innovativo che usa l'Intelligenza Artificiale per monitorare e predire il livello di occupazione di supermercati o grandi esercizi commerciali**, rispondendo a tali inefficienze.
  - SOMS permette di rispondere rapidamente a variazioni nel flusso di clienti e di efficientare il servizio dei clienti, garantendone un alto livello di soddisfazione e fidelizzazione.
  - **SOMS fornisce un vantaggio concreto sulla concorrenza**, fornendo informazioni essenziali di cui molti esercizi commerciali non dispongono.
- 

# Funzionalità principali



## Monitoraggio

dell'occupazione in  
tempo reale



## Controllo

degli ingressi nel rispetto del  
distanziamento sociale



## Memorizzazione

dei dati di occupazione  
passati su cloud



## Visualizzazione

dei dati di occupazione  
passati



## Predizione

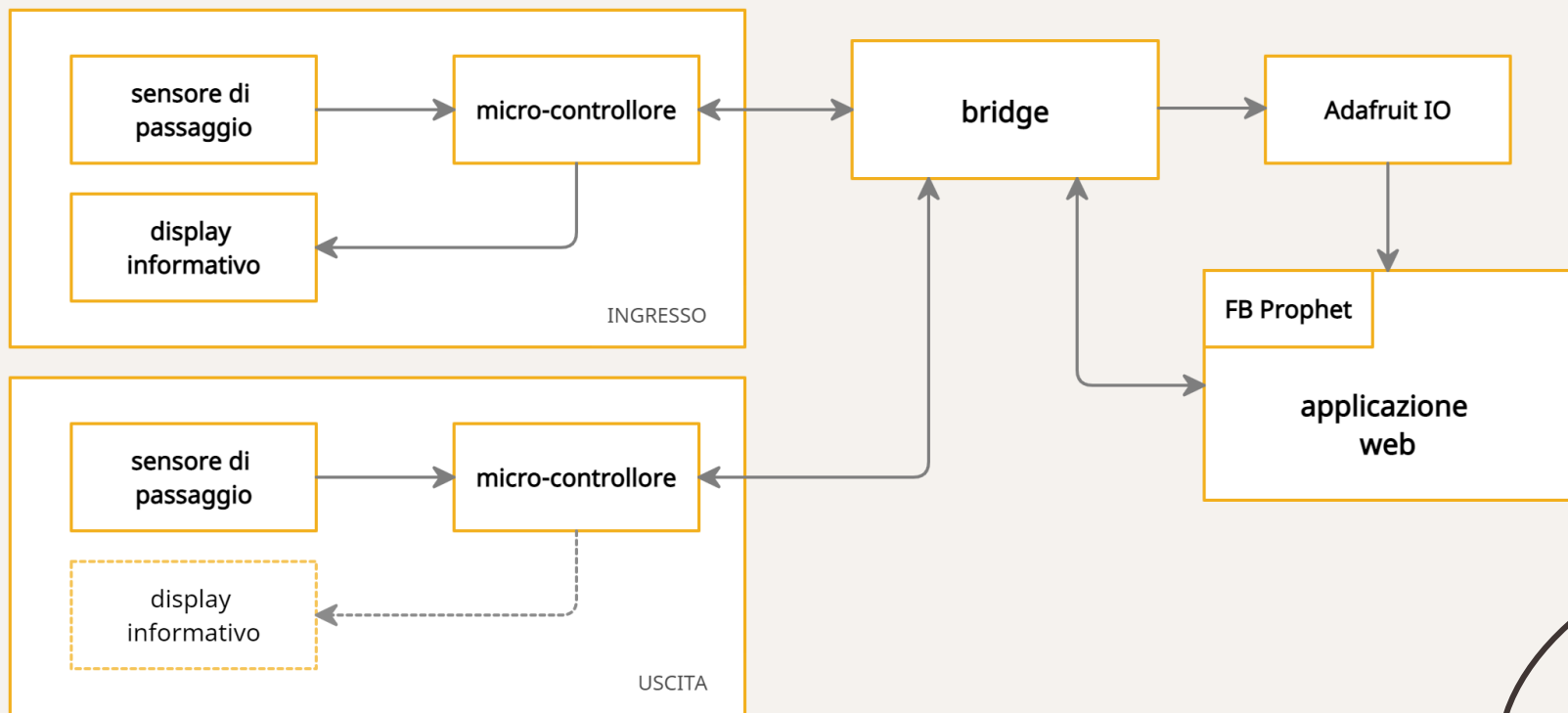
del livello di occupazione  
futuro con IA



02

# Architettura

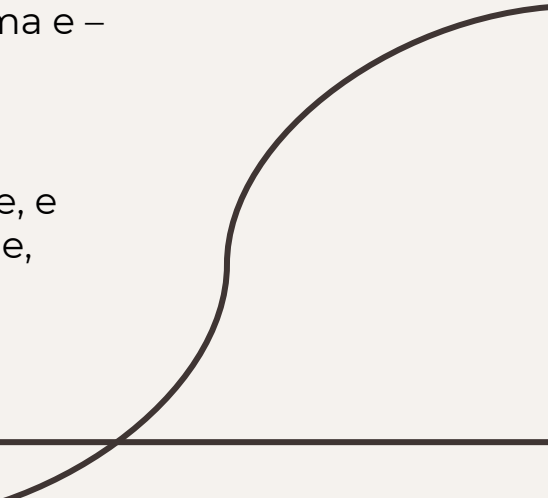
Panoramica dell'architettura hardware e  
software



*Componenti hw/sw di SOMS. Le frecce indicano il verso del flusso di dati da un componente ad un altro.*

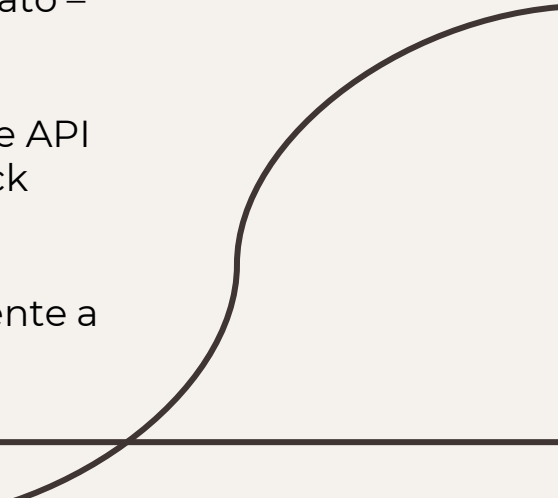


# Componenti hardware e software - sensori

- SOMS può essere esteso con molteplici sensori, configurabili come ingressi o uscite.
  - Ogni sensore può essere collegato a un display per la visualizzazione di occupazione attuale, occupazione massima e – quando disponibile – feedback relativo alla predizione per l'occupazione attuale.
  - Ogni sensore comunica gli attraversamenti rilevati al bridge, e riceve da esso aggiornamenti periodici (occupazione attuale, occupazione massima, feedback).
- 

# Componenti hardware e software - bridge


Il bridge svolge diverse funzioni essenziali:

- Raccoglie le rilevazioni dei sensori, a partire dalle quali calcola l'occupazione attuale.
  - Invia sul cloud – in particolare sul feed Adafruit IO configurato – l'occupazione attuale.
  - Recupera le predizioni relative ai prossimi giorni – mediante API esposta dall'applicazione web – ai fini di calcolare il feedback relativo alla predizione per l'occupazione attuale.
  - Invia ai sensori collegati aggiornamenti costanti relativamente a occupazione attuale, occupazione massima e feedback.
- 

# Componenti hardware e software – Adafruit IO

- L'occupazione attuale viene regolarmente caricata dal bridge su uno specifico feed Adafruit IO, che ha funzione principale di memorizzazione dei dati. Ogni feed corrisponde a una diversa sede commerciale.
- Adafruit IO è usato anche per visualizzare lo storico delle occupazioni passate, nonché avere informazioni sul funzionamento in tempo reale di SOMS (occupazione attuale, stream degli upload al feed, ecc...).
- Nel sistema finale, è fortemente consigliabile sostituire Adafruit IO con soluzione sviluppata in-house, date le sue limitazioni.

# Componenti hardware e software – applicazione web

- SOMS fornisce agli utenti un'interfaccia web semplice e intuitiva, basata su framework Flask, che consenta loro di accedere alle predizioni future di selezionati feed.
  - L'applicazione web fornisce altresì una API per visualizzare l'elenco dei feed per cui sono disponibili predizioni, generare predizioni e feedback.
  - L'applicazione web si integra con FB Prophet per generare le predizioni, addestrato con un dataset reale – pubblicamente disponibile – e processato per adattarlo al contesto di SOMS.
- 



# 03

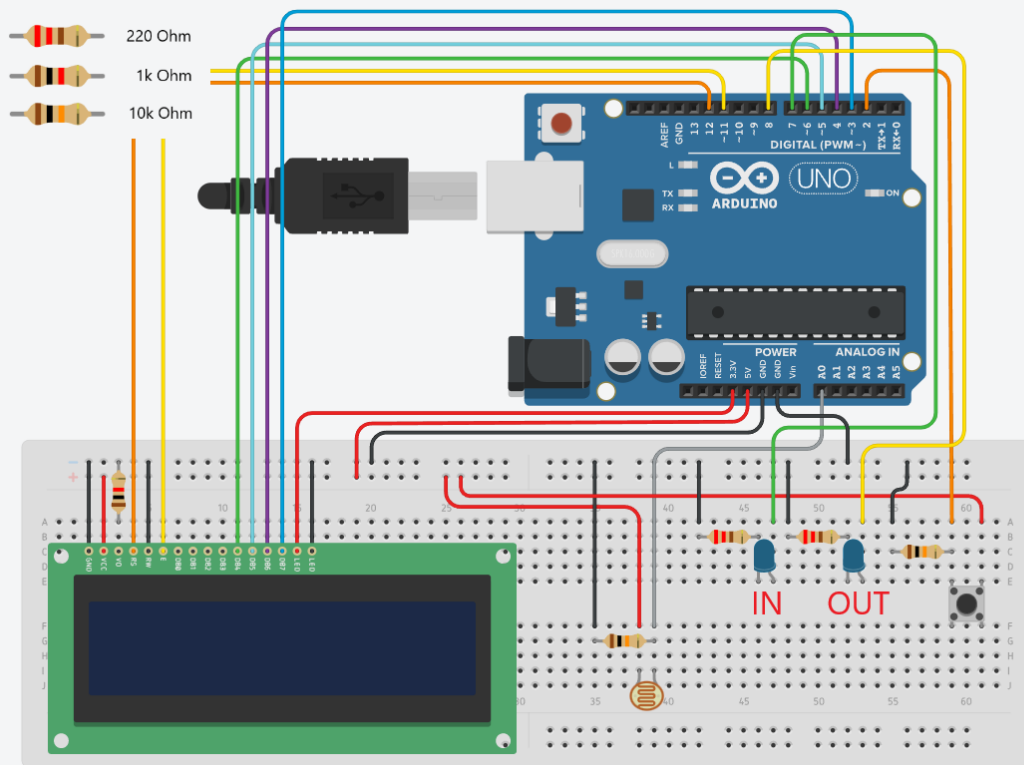
# Componenti

Analisi dettagliata dei componenti del sistema



# Analisi dettagliata dei componenti – sensori (hw)

- Ogni sensore si compone principalmente di **1 micro-controllore**, **1+ memorie**, **1 interfaccia seriale** (per il prototipo si è usata una scheda Arduino Uno), **1 coppia led-fotoresistenza** per la rilevazione degli ostacoli – idealmente da sostituire con fotocellula a infrarossi –, **1 pulsante** di selezione della modalità operativa (ingresso/uscita) e **2 led** di indicazione della modalità operativa selezionata – idealmente da sostituire con interruttore a levetta.
- Ai sensori di ingresso è collegato anche **1 display LCD** per la visualizzazione degli aggiornamenti inviati sull'interfaccia seriale dal bridge, che può essere eventualmente collegato anche ai sensori di uscita.



*Schema elettrico di un sensore di ingresso. Nello schema non è mostrato il led accoppiato alla fotoresistenza, alimentato a batteria.*

# Analisi dettagliata dei componenti – sensori (sw)

- I sensori individuano un ostacolo mediante l'interruzione di un fascio luminoso diretto dal led alla corrispondente fotoresistenza, che determina una diminuzione della luminosità media percepita rispetto alla luminosità media ambientale.
- La rilevazione di un attraversamento si ha nel momento in cui si passa dall'avere un ostacolo a non avere alcuno ostacolo – si veda l'automa a stati finiti per i dettagli. In caso di attraversamento, il sensore invia al bridge un +1 o un -1 a seconda che sia configurato come ingresso o come uscita.
- A intervalli di tempo regolari, il sensore riceve pacchetti di aggiornamento dal bridge ed effettua il refresh del display LCD – se collegato – con i nuovi dati. Si rimanda all'analisi del bridge per quanto riguarda il formato dei pacchetti di aggiornamento.

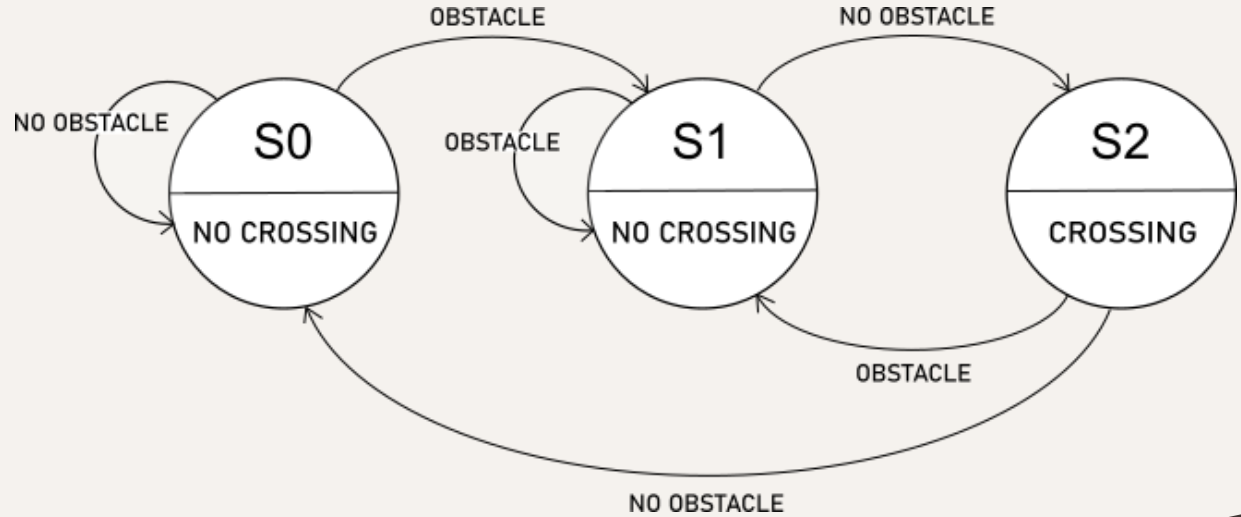


*Inputs =*  
{ OBSTACLE, NO OBSTACLE }

*Outputs =*  
{ CROSSING, NO CROSSING }

*States =*  
{ S0, S1, S2 }, con:

- S0 stato iniziale,
- S1 stato in cui è stato rilevato un ostacolo,
- S2 stato in cui l'ostacolo non è più rilevato



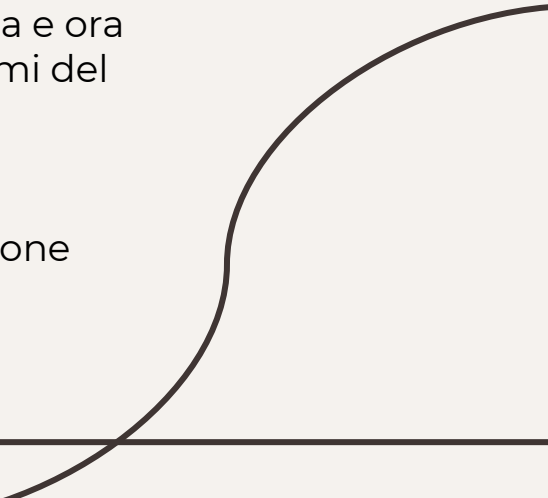
*Automa a stati finiti per la rilevazione di attraversamenti da parte dei sensori.*

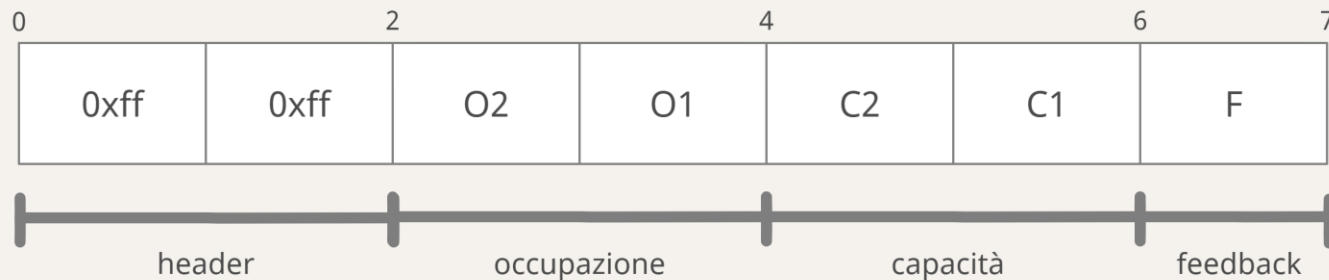
# Analisi dettagliata dei componenti – bridge (1)

- A livello hardware, si è usato un notebook con funzione di bridge, collegato ai sensori mediante USB. Nel sistema finale, si utilizzerà un SoC con funzionalità di rete. A livello software, il bridge utilizza la libreria Python esterna adafruit-io per inviare dati sul feed configurato.
- Il bridge attraversa una fase di setup iniziale, in cui sono inizializzate le connessioni seriali con i sensori, viene inizializzato un client Adafruit IO per l'invio di dati sul feed e vengono recuperate le predizioni dei prossimi 7 giorni – a partire dal giorno corrente – mediante l'API esposta dall'applicazione web.

# Analisi dettagliata dei componenti – bridge (2)

Terminata la fase di setup, il bridge esegue diverse operazioni in loop, a intervalli di tempo regolari:

- Raccoglie gli aggiornamenti dai sensori e calcola l'occupazione attuale come somma degli aggiornamenti ricevuti.
  - Calcola il feedback relativo all'occupazione predetta per data e ora correnti, confrontando l'occupazione misurata con gli estremi del range di predizioni del modello FB Prophet.
  - Compone e invia ai sensori collegati un pacchetto di aggiornamento, costituito di occupazione attuale, occupazione massima e feedback.
  - Carica l'occupazione attuale sul feed configurato.
- 



**header (2 byte):**

identifica l'inizio del pacchetto, ed è composto di due byte settati a 0xff.

**occupazine (2 byte):**

occupazione attuale in formato big endian, rappresentata da un intero nell'intervallo [0,65534].

**capacità (2 byte):**

capacità in formato big endian, rappresentata da un intero nell'intervallo [1,65534].

**feedback (1 byte):**

può assumere i valori 0x01 (interpretato come 'below range'), 0x02 ('in range'), 0x03 ('above range') o 0x04 ('no feedback').

*Formato dei pacchetti di aggiornamento inviati regolarmente dal bridge ai sensori con occupazione attuale e massima di 2 byte.*

# Analisi dettagliata dei componenti

## – applicazione web (1)

- L'applicazione web è scritta in Python e si basa su framework Flask. Fornisce sia un'interfaccia web sia un'API per la visualizzazione dell'elenco dei feed e la generazione delle predizioni. Nel caso dell'API, fornisce anche un servizio per la valutazione del feedback sulle predizioni. Si osserva che le API ritornano le risorse richieste in formato JSON.
- A livello hardware, l'applicazione viene eseguita sulla stessa macchina del bridge per comodità, in un processo separato. Ovviamente, nel sistema finale bridge e applicazione web gireranno su due macchine diverse.
- L'applicazione web adotta diverse tecniche, tra cui il caching delle predizioni, per ridurre il più possibile i tempi di gestione delle richieste.

# Analisi dettagliata dei componenti

## – applicazione web (2)

- Per la generazione delle predizioni, l'applicazione web si affida a un modello FB Prophet pre-addestrato ed esportato come file JSON.
- Tale modello è stato addestrato e sottoposto a fine-tuning utilizzando un dataset pubblicamente disponibile, ovviamente adattato al contesto in esame – il dataset è disponibile all'URL: <https://archive.ics.uci.edu/ml/datasets/CallIt2+Building+People+Counts>.
- Quando l'applicazione deve recuperare una nuova predizione, e questa non è ancora stata generata, il modello FB Prophet viene caricato dal file JSON e gli viene passato un nuovo dataframe, creato sul momento, inizializzato con i timestamp dei prossimi giorni. Una volta generata la predizione, questa viene esportata come file CSV.

# Analisi dettagliata dei componenti – API (1)

## ***Recuperare l'elenco dei feed***

È possibile recuperare l'elenco dei feed Adafruit IO per cui è stato addestrato un modello FB Prophet, con la seguente richiesta HTTP:

```
GET /api/v1/feeds
```

Il risultato è una lista di oggetti JSON nel formato:

```
{  
  "name": <nome-feed>,  
  "key": <chiave-feed>,  
  "description": <descrizione-feed>  
}
```

# Analisi dettagliata dei componenti – API (2)

## ***Recuperare una predizione***

È possibile recuperare la predizione dei prossimi *periods* giorni a partire da *from\_date* (data in formato 'YYYY-MM-DD') e a intervalli di *interval\_mins* minuti per il feed con chiave *feed\_key*, con la seguente richiesta HTTP:

```
GET /api/v1/predictions/{ feed_key }/{ from_date }/{ periods }/{ interval_mins }
```

Il risultato è un oggetto JSON nel formato:

```
{  
  "feed": <nome-feed>,  
  "periods": <periods>,  
  "timestamps": <lista-timestamps>,  
  "predictions": <lista-predizioni>,  
  "predictions_lower_bound": <lista-limiti-inferiori-predizione>,  
  "predictions_upper_bound": <lista-limiti-superiori-predizione>,  
}
```



# Analisi dettagliata dei componenti – API (3)

## ***Recuperare un feedback***

E' possibile recuperare il feedback relativo alla predizione per il feed con chiave *feed\_key* per data e ora correnti rispetto all'occupazione *occupancy*, con la seguente richiesta HTTP:

```
GET /api/v1/feedbacks/{ feed_key }/{ occupancy }
```

Il risultato è un oggetto JSON nel formato:

```
{  
  "feedback": <feedback>  
}
```

dove <feedback> può assumere i valori "in\_range", "below\_range", "above\_range" e "na" (feedback non disponibile).



04

# Prototipo

Implementazione delle funzionalità principali di  
SOMS

# Contatti e crediti

Slides a cura di:

Marco Michelini  
211728@studenti.unimore.it

**CREDITS:** This presentation template was created  
by **Slidesgo**, including icons by **Flaticon**, and  
infographics & images by **Freepik**