

Relazione Progetto IoT (Safe Lab)

Adamo Fapohunda
0000907136

Laura Bugo
0000907116

1 Introduzione

L'obiettivo del progetto è quello di realizzare un sistema per il monitoraggio di un ambiente basandosi su tecnologie IOT. In particolare vengono misurati i valori ambientali di un locale (temperatura, umidità e pressione) e il numero di persone che si trovano all'interno di esso. Ci si è posti come intento quello di implementare il sistema utilizzando sensori e schede di sviluppo a bassa potenza e basso costo.

Nella relazione che segue verranno descritti in maniera approfondita gli strumenti utilizzati (Sezione 2), l'architettura del progetto (Sezione 3), l'implementazione svolta (Sezione 4) e i risultati ottenuti (Sezione 5).

2 Environment

2.1 Hardware

Per i due obiettivi del progetto, ossia contare il numero di persone all'interno della stanza e rilevare i parametri ambientali sono stati utilizzate due single-board microcontroller NodeMCU dotate di connettività WiFi.

Per la raccolta e visualizzazione dei dati è stato utilizzato un Raspberry Pi 3, un single-board computer dotato di cpu ARM a 64 bit corredata da un hard disk drive (HDD) esterno.

2.2 Software

- Docker e Docker-compose [2]: utilizzato per distribuire il software in container isolati e che hanno la possibilità di comunicare tra loro;
- Mozilla WebThings Framework e Mozilla WebThings Gateway [8]: utilizzato per risolvere il problema della findability delle Thing connesse alla rete locale. Inoltre è in grado di gestire automaticamente la riconnessione con i devices;
- InfluxDB 2.0 [5]: utilizzato come database per il salvataggio dei dati sotto forma di time series;

- Grafana [3]; utilizzato per permettere una visualizzazione interattiva dei dati ottenuti;
- Telegram: utilizzato per notificare all'utente situazioni di alerting.

Linguaggi utilizzati:

- Arduino language (C/C++): per la programmazione delle board NodeMCU;
- Javascript/TypeScript: per la costruzione dei servizi;
- Python: per lo studio e l'implementazione di metodologie di forecasting sui dati e la costruzione del servizio di forecasting;
- Bash scripting.

3 Architettura

Come è possibile osservare in Figura 1, sono stati costruiti due smart device che sono stati implementati come Web Things. Tali Thing sono rilevate dal Gateway di Mozilla che funge da unico punto di accesso per caricare i dati su InfluxDB.

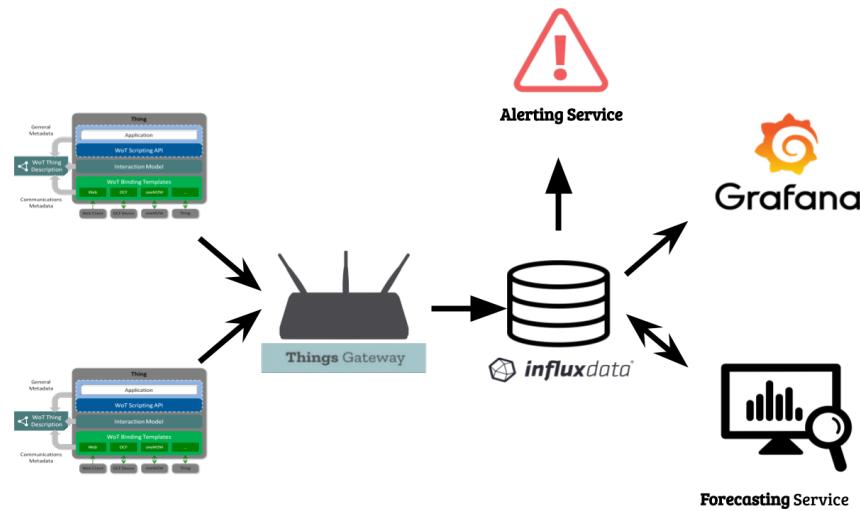


Figura 1: Architettura

I dati caricati su InfluxDB sono quindi analizzati per effettuare operazioni di forecasting sui livelli dei parametri ambientali e del numero di persone nell'ambiente. La visualizzazione dei dati è fornita attraverso le dashboard di Grafana. Inoltre è stato implementato un meccanismo di alerting per notificare l'utente attraverso Telegram nel caso in cui i parametri dell'ambiente raggiungano livelli critici.

3.1 Docker

Docker è stato utilizzato per poter mantenere i diversi servizi isolati tra loro, inoltre ci permette di garantire la riproducibilità degli ambienti di sviluppo.

Il container di Docker sono stati utilizzati per racchiudere i diversi servizi:

- Mozilla Gateway;
- InfluxDB 2.0;
- Servizio di Forecasting:
 - Writer Service;
 - Forecasting Service.
- Grafana;
- Servizio di alerting.

Tali container sono stati caricati su Raspberry Pi. Uno script di installazione permette di fare build delle immagini necessarie e gestire i servizi tramite Docker Compose. Tuttavia è necessario procurarsi i token di:

- Mozilla Gateway;
- InfluxDB;
- Telegram.

3.1.1 Problematiche riscontrate

Dato che Influx non è disponibile per architetture ARM32, per poterlo installare nel Raspberry è stato necessario installare un sistema operativo a 64 bit, in particolare Ubuntu Server 20.04. In un secondo momento è stata creata un'immagine Docker ad hoc sfruttando la release a 64 bit, tuttavia la scelta di passare a un sistema operativo a 64 bit ha comportato a difficoltà dovute a incompatibilità tra TensorFlow e architetture ARM a 64 bit.

4 Implementazione

4.1 Smart Things e WoT Things

Nelle sezioni a seguire sono presentati i dispositivi fisici costruiti e la loro rappresentazione come Web Things.

4.1.1 Room Weather

Per costruire un sistema per raccogliere i parametri ambientali di temperatura, umidità e pressione sono stati utilizzati i seguenti sensori opportunamente collegati alla board NodeMCU:

- Sensore per il rilevamento della temperatura/umidità (DHT22);
- Sensore per il rilevamento della pressione (BMP180).

Entrambi i dispositivi hardware costruiti sono stati definiti come Web Things utilizzando il WebThings Framework di Mozilla che fornisce la libreria *webthing* per Arduino IDE. Tale libreria ha permesso di definire Room Weather come una thing che possiede le properties *pressure*, *humidity* e *temperature*. Tali properties sono definite come *readOnly*. Le Thing Description sono state inoltre arricchite con un'indicazione dell'unità di misura e, dove possibile, un valore minimo e massimo.

```
1 {
2     "title": "RoomWeather",
3     "@context": "https://iot.mozilla.org/schemas",
4     "@type": [
5         "TemperatureSensor",
6         "MultiLevelSensor"
7     ],
8     "description": "Multisensor",
9     "href": "...",
10    "properties": {
11        "pressure": {
12            "title": "Atmospheric Pressure",
13            "type": "number",
14            "unit": "hPa",
15            "readOnly": true,
16            "links": [{ ... }]
17        },
18        "humidity": {
19            "title": "Humidity",
20            "type": "number",
21            "@type": "LevelProperty",
22            "unit": "percent",
23            "minimum": 0,
24            "maximum": 100,
25            "readOnly": true,
26            "links": [{ ... }]
27        },
28        "temperature": {
29            "title": "Temperature",
30            "type": "number",
31            "@type": "TemperatureProperty",
```

```

32         "unit": "degree celsius",
33         "readOnly": true,
34         "links": [...]
35     },
36     "actions": {},
37     "events": {},
38     ...
39 }
40

```

Il programma caricato sulla board tramite ArduinoIDE consente di calcolare e settare il valore delle properties WoT thing nel momento in cui avvengono delle variazioni nei parametri dei valori ambientali.

Per quanto riguarda la pressione, questa deve essere valutata tenendo in considerazione la temperatura rilevata e l'altitudine rispetto al livello del mare.

In Figura 2 è possibile vedere la visualizzazione attraverso il Mozilla Gateway del web thing Room Weather. Infine è stata costruita una custodia per contenere il device che può essere visualizzata in Figura 3.

4.1.2 People Counter

Per implementare un contatore del numero di persone in prima istanza è stata effettuata una ricerca per valutare le possibili componenti da utilizzare:

- Un sensore Grid-Eye;
- Sensori ad infrarossi passivi (PIR);

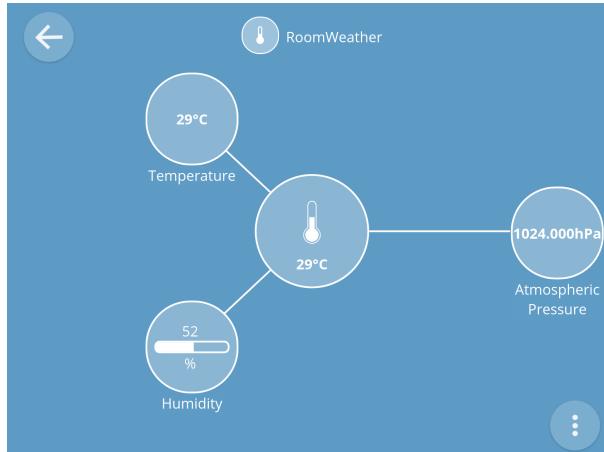
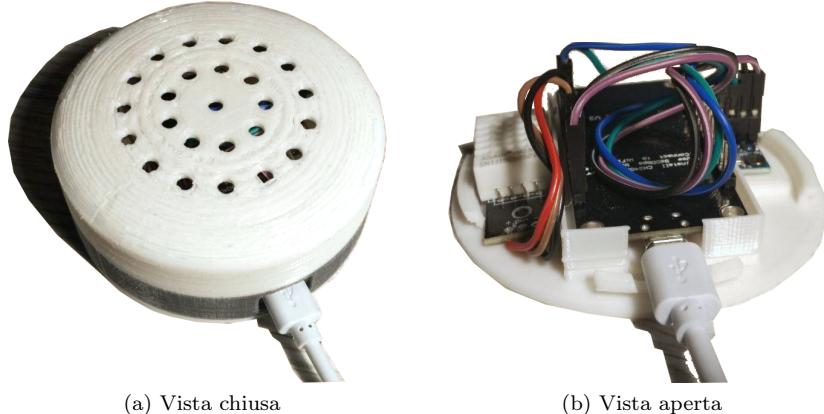


Figura 2: Visualizzazione di Room Weather come web thing attraverso il Gateway di Mozilla



(a) Vista chiusa

(b) Vista aperta

Figura 3: Viste del sensore Room Weather

- Sensori anticollisione ad infrarossi;
- Fotoresistenze abbinate a laser.

Il sensore Grid-Eye è composto da una griglia di 8x8 sensori termici che permettono di visualizzare una matrice termica dell'area verso cui il sensore è rivolto. Nonostante il potenziale l'ipotesi di utilizzarlo è stata subito scartata per ragioni di costo sia economico che computazionale infatti si sarebbe resa necessaria un'analisi di immagini per poter differenziare eventi di entrata ed uscita.

Assumendo che le entrate/uscite dalla stanza avvengano con il passaggio di una persona alla volta, è possibile passare in analisi anche le altre tipologie di sensori.

Si è passati dunque allo studio dei sensori PIR. Anche quest'ultimi sono stati scartati a causa della scarsa precisione soprattutto quando ravvicinati.

Sono infine stati realizzati due prototipi utilizzando le ultime due tecnologie passate in rassegna.

Il primo prototipo è stato realizzato utilizzando due sensori anticollisione ad infrarossi (Figura 4). Tuttavia questi venivano facilmente influenzati dalla luce solare ed inoltre si sono rivelati essere efficaci solo a brevi distanze (sotto i 20 cm).

Con poche modifiche è stato realizzato il secondo nonché ultimo prototipo utilizzando:

- Due laser
- Due fotoresistenze

L'idea di base per l'implementazione è quella di puntare i laser contro le fotoresistenze per poi controllare quale fascio venga interrotto per primo. Questo

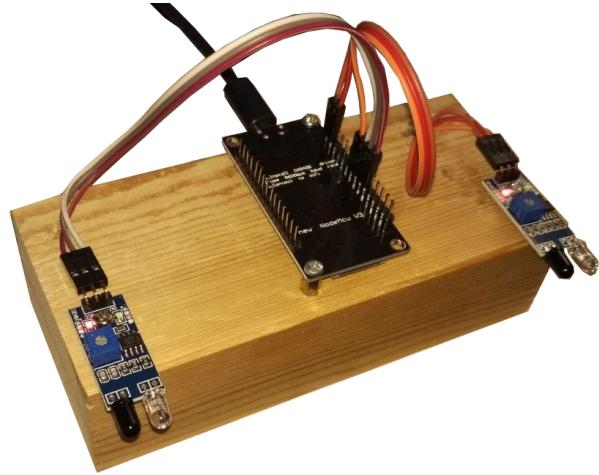


Figura 4: Prototipo per PeopleCounter con sensori anticollisione.

permette di dedurre quando il passaggio attraverso una porta sia un evento di entrata o di uscita.

Come per il caso precedente, People Counter è stato implementato come una Web Thing utilizzando il Framework fornito da Mozilla. In particolare è stato definita una thing *People Counter* con una property *peopleNum* che mantiene il numero di persone all'interno del locale. Di seguito si mostra la thing description ottenuta:

```

1  {
2      "title": "People Counter Sensors",
3      "@context": "https://iot.mozilla.org/schemas",
4      "@type": ["MotionSensor"],
5      "description": "Counter for people in a room",
6      "href": "/things/http---w25.local-things-peopleCounter",
7      "properties": {
8          "peopleNum": {
9              "title": "PeopleCounter",
10             "type": "integer",
11             "readOnly": true,
12             "links": [{ ... }]
13         }
14     },
15     "actions": {},
16     "events": {},
17     ...
18 }
```

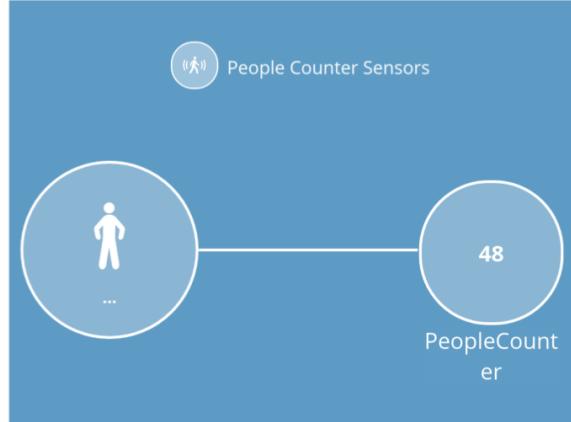


Figura 5: Visualizzazione di People Counter come web thing attraverso il Mozilla Gateway

La gestione degli accessi a livello software è stata gestita attraverso interrupt e un timer. Gli interrupt permettono di notificare quando il fascio laser viene interrotto, si valuta quindi se è già stato generato l'interrupt dell'altro laser per capire se il passaggio sia in entrata o in uscita. Il timer è stato introdotto per gestire il caso in cui venga attraversato un solo laser per poi tornare indietro. Nel momento in cui il primo fascio laser viene interrotto, viene avviato un timer entro il quale attendere un interrupt dalla seconda fotoresistenza e, allo scadere del timer, i valori vengono resettati e quindi si ritorna in uno stato in cui nessun fascio laser è stato interrotto.

In Figura 5 è possibile vedere la visualizzazione attraverso il Mozilla Gateway della web thing People Counter.

Anche in questo caso è stato costruito un prototipo di custodia per contenere i laser e le fotoresistenze come è possibile vedere in Figura 6.

4.1.3 WoT Thing aggiuntive

Inoltre sono state utilizzate altre due thing ottenute attraverso un add-on del Mozilla Gateway che permettono di monitorare le condizioni del Raspberry Pi 3. In particolare una thing permette di monitorare la temperatura della CPU e l'utilizzo della RAM. L'utilizzo di queste thing è stato rilevante perché il Raspberry Pi mostrava crash piuttosto frequenti che sono stati ricondotti a un eccessivo surriscaldamento. Per questo motivo è stato ideato un sistema di raffreddamento che ha permesso il proseguimento del lavoro. In Figura 7 si mostra il Raspberry con la ventola dedicata al raffreddamento che ha permesso di abbassare di circa 10 gradi la temperatura del sistema.

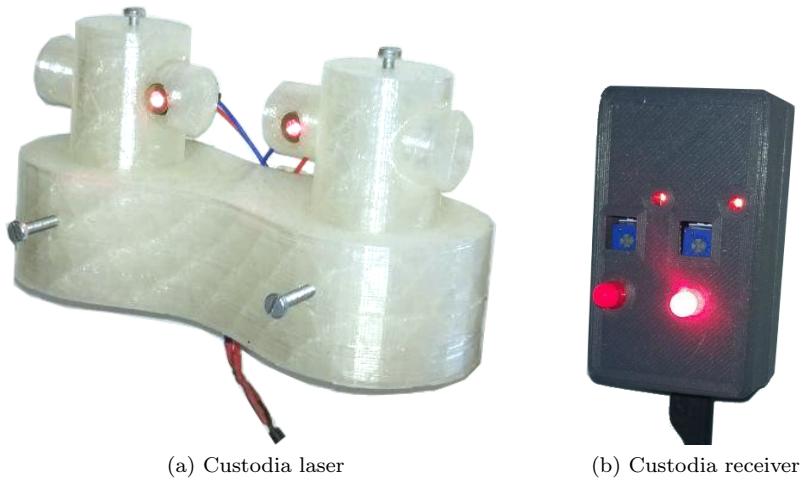


Figura 6: Viste della custodia per People Counter

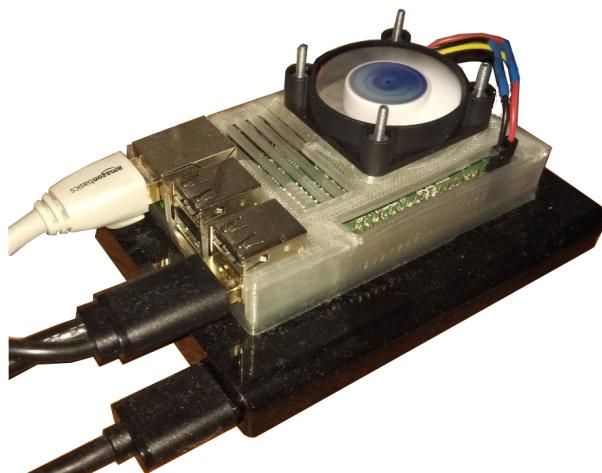


Figura 7: Raspberry Pi 3 utilizzato e ventola per il raffreddamento

4.2 Mozilla Gateway

Il Mozilla Gateway è stato utilizzato perché risolve il problema della findability delle things.

Nel momento in cui si vuole effettuare una ricerca delle thing, il gateway di Mozilla fornisce una funzione di search che utilizza il protocollo mDNS per capire quali siano i device connessi alla rete locale. Trovati tutti i device, è possibile selezionare quello da aggiungere e di questo ne viene effettuata una copia sul gateway stesso. L'indirizzo IP del device viene però sostituito con quello di ritorno del protocollo mDNS, in questo modo le thing sono esposte all'esterno attraverso il gateway su cui viene anche implementato lo strato di sicurezza.

Inoltre il Mozilla Gateway è in grado di gestire la riconnessione delle thing.

4.3 InfluxDB bridge

Per connettere il Gateway di Mozilla a InfluxDB è stato scritto un add-on partendo dall'implementazione, disponibile su GitHub, di Tim Hellhake [6] che però è stata modificata per aggiungere compatibilità con InfluxDB 2.0. Sono state inoltre aggiunte alcune funzionalità che non erano presenti nell'add-on originale, in particolare la possibilità di selezionare quali Thing salvare e la possibilità di riconnettersi automaticamente nel caso in cui la thing si sia per qualche motivo disconnessa.

L'add-on è in grado di ottenere dal Mozilla Gateway la lista dei device presenti e di eseguire un'operazione di filtraggio in modo da caricare su InfluxDB le informazioni relative solamente ad alcune thing di interesse. Per ogni dispositivo viene svolto un tentativo di aprire un websocket e, all'arrivo di messaggi da parte delle thing, viene costruito un nuovo punto da salvare su InfluxDB. I punti su InfluxDB sono salvati con il valore, il nome della property e le informazioni relative al device.

In Figura 8 è possibile visualizzare attraverso il Mozilla Gateway l'add-on costruito. L'add-on necessita di due token, quello per poter leggere dal gateway e quello per poter scrivere su InfluxDB, inoltre gli sono fornite le informazioni principali per poter scrivere sul database di Influx ossia: hostname, port, organization e bucket.

4.4 Analisi dei dati

Utilizzando Python e Jupyter Notebook è stato possibile interrogare il database di InfluxDB e analizzare il sistema migliore per fare previsioni sui dati. I dati utilizzati consistono di valori raccolti su 12 giorni e aggregati ogni 5 minuti.

4.4.1 Pulizia del Dataset

In una prima fase si utilizza il client di influxDB per richiedere i dati da analizzare ed inserirli in un Dataframe (Figura 9). Utilizzando la libreria di panadas sono state fatte alcune elaborazioni su tale Dataframe per riempire i valori nulli.

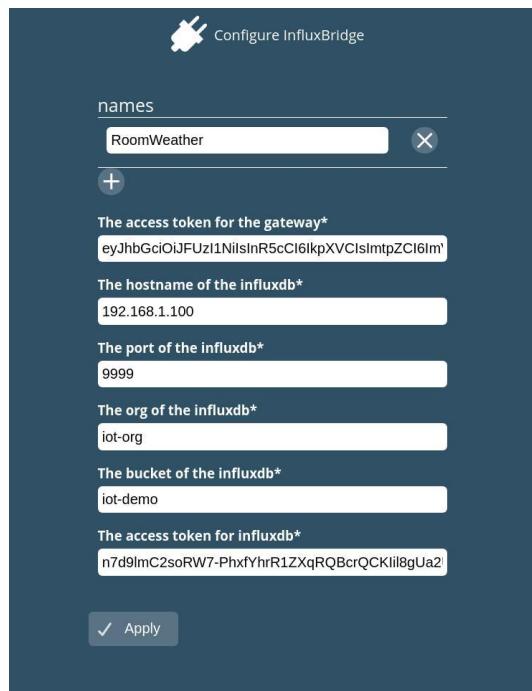


Figura 8: Visualizzazione attraverso il Mozilla Gateway dell'add-on costruito

time	temperature	humidity	pressure	people
2020-07-07 12:10:00+00:00	29.25	46.9	1019.5	0.0
2020-07-07 12:15:00+00:00	29.40	46.8	1019.5	0.0
2020-07-07 12:20:00+00:00	29.40	46.8	1019.5	0.0
2020-07-07 12:25:00+00:00	29.40	46.5	1019.5	0.0
2020-07-07 12:30:00+00:00	29.40	46.5	1019.5	0.0

Figura 9: Visualizzazione del dataframe ottenuto da InfluxDB

```

Augmented Dickey-Fuller Test on "temperature"
-----
Null Hypothesis: Data has unit root. Non-Stationary.
P-Value = 0.0047. Rejecting Null Hypothesis.
=> Series is Stationary.

Augmented Dickey-Fuller Test on "humidity"
-----
Null Hypothesis: Data has unit root. Non-Stationary.
P-Value = 0.2612. Weak evidence to reject the Null Hypothesis
=> Series is Non-Stationary.

Augmented Dickey-Fuller Test on "pressure"
-----
Null Hypothesis: Data has unit root. Non-Stationary.
P-Value = 0.7503. Weak evidence to reject the Null Hypothesis
=> Series is Non-Stationary.

Augmented Dickey-Fuller Test on "people"
-----
Null Hypothesis: Data has unit root. Non-Stationary.
P-Value = 0.0003. Rejecting Null Hypothesis.
=> Series is Stationary.

```

Figura 10: Risultati del Dickey-Fuller test su temperatura, umidità, pressione e persone

4.4.2 Stazionarietà

In un secondo momento si è svolta un'analisi per valutare se le serie ottenute fossero o meno stazionarie. Per questo viene utilizzata una funzione che svolga il test Dickey-Fuller. Dai risultati, mostrati in Figura 10, è possibile vedere che la temperatura e il numero di persone sono identificate come serie stazionarie mentre umidità e pressione sono valutate come serie non stazionarie.

4.4.3 Componenti della timeseries

Una serie temporale è costituita da tre componenti:

1. Trend: valore crescente o decrescente della serie
2. Stagionalità: cicli ripetuti nel breve termine all'interno della serie
3. Rumore: variazione casuale nella serie

Le prime due componenti sono dette sistematiche perché hanno una consistenza o ricorrenza che può essere modellata, l'ultima è detta non sistematica perché la componente del rumore della serie temporale non può essere direttamente modellata. In generale le time-series sono un'aggregazione o combinazione di queste tre componenti.

Le diverse componenti sono state estratte per ognuna delle quattro misurazioni (temperatura, umidità, pressione e numero di persone) e sono rappresentate in Figura 11. In particolare il primo grafico rappresenta il trend della serie, il secondo rappresenta il rumore, il terzo rappresenta la stagionalità e l'ultimo rappresenta la forma effettiva della serie.

4.4.4 Autocorrelazione

La correlazione di Pearson è un coefficiente compreso tra -1 e 1 che descrive una correlazione negativa o positiva rispettivamente. Dato che i fattori ambientali sono influenzati dal momento della giornata, si ipotizza una forte correlazione tra il valore attuale e quello del giorno precedente allo stesso orario. Per valutare ciò sono state utilizzate una funzione di autocorrelazione (ACF) e autocorrelazione parziale (PACF). In Figura 12 sono mostrati i grafici di ACF e PACF per i valori della temperatura.

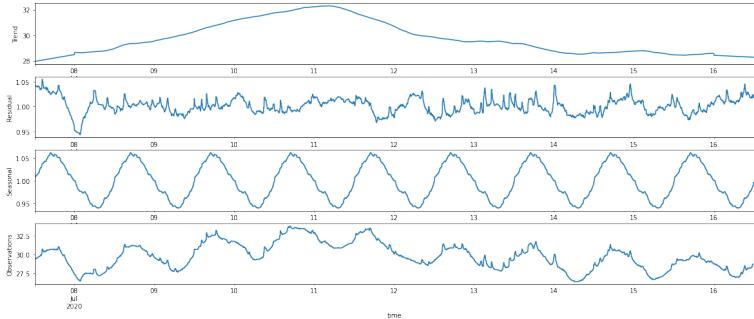
Il grafico in Figura 12a mostra la correlazione della temperatura attuale con i valori precedenti di temperatura. Risulta possibile notare che la temperatura è fortemente correlata ai valori che la precedono e che, allontanandosi nel tempo, la correlazione diminuisce. Inoltre si può osservare come la correlazione, il giorno precedente nella stessa fascia oraria, sia molto elevata.

Per valutare l'effettiva correlazione tra due punti indipendentemente dalla correlazione che deriva dai valori vicini, si utilizza il grafico PACF. Il grafico in Figura 12b ci mostra come il valore attuale di temperatura sia fortemente correlato a quello subito precedente e a quello prima. Sul grafico PACF è possibile svolgere un ragionamento analogo anche per i grafici ottenuti dall'analisi di umidità e pressione in cui il valore attuale è fortemente collegato solamente a pochi valori che lo precedono.

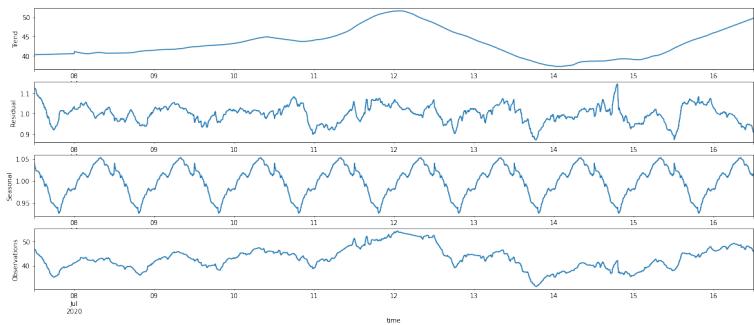
4.5 Metodi di forecasting testati

Sono stati testati diversi metodi per fare previsione sui dati ottenuti dai sensori. In particolare sono state testate le seguenti metodologie:

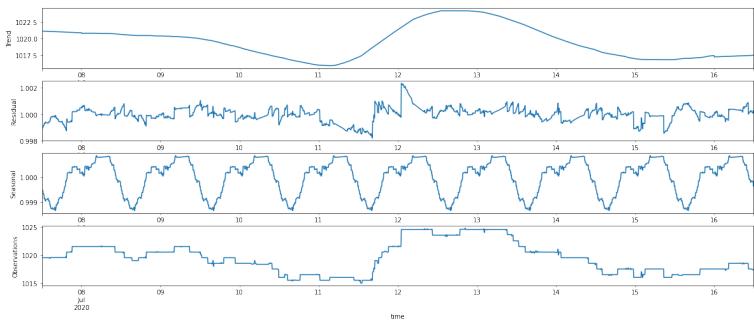
- Seasonal Autoregressive Integrated Moving-Average (SARIMA)
- Vector Autoregression Moving-Average (VARMA)
- Holt Winter's Exponential Smoothing (HWES)
- Long Short-Term Memory (LSTM)



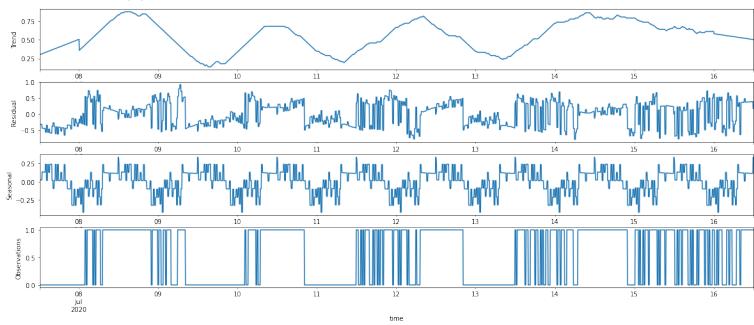
(a) Analisi delle componenti per i valori di temperatura



(b) Analisi delle componenti per i valori di umidità



(c) Analisi delle componenti per i valori di pressione



(d) Analisi delle componenti per i valori riguardo al numero di persone

Figura 11: Analisi delle componenti delle time-series

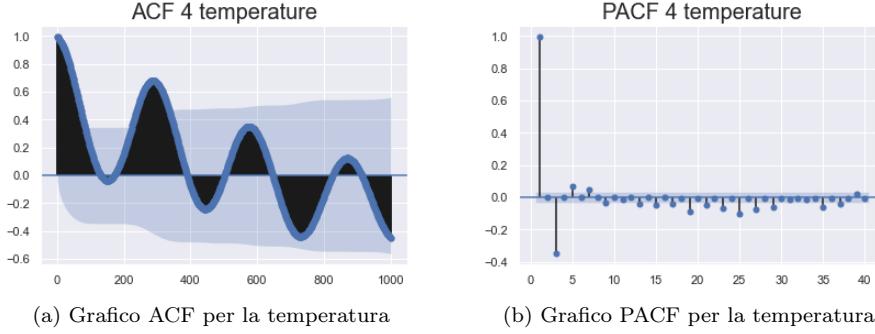


Figura 12: Grafici di correlazione per i valori di temperatura

4.5.1 SARIMA

Autoregressive Model (AR)

Il nome del modello AR deriva da autoregressione che significa utilizzare i valori precedenti della variabile per predire i valori futuri. Ci si affida a modelli di autoregressione se esiste una chiara autocorrelazione tra i dati. Il modello può essere rappresentato con parametro 1 dalla formula che segue[1]:

$$x_t = c + \Phi x_{t-1} + \epsilon_t$$

Dove x_t e x_{t-1} sono rispettivamente il valore attuale e il precedente, ϕ è una costante per cui moltiplicare il valore del lag precedente e ϵ_t è il residuo, ossia la differenza tra la predizione e il valore reale.

Analizzando il grafico PACF per la temperatura (Figura 12b) risulta possibile verificare che i valori di correlazione sono significativi fino al terzo valore, graficamente sono riconosciuti come significativi i punti fuori dall'area azzurra. Si ricerca quindi un valore di AR minore o uguale a 3. Per questo motivo sono stati testati AR(1), AR(2) e AR(3). Analizzando i coefficienti ottenuti, è possibile verificare che per tutti i modelli testati, siano tutti significativi. Si è utilizzata una funzione ausiliaria per testare se la likelihood del modello più complesso (AR(3)) fosse significativamente migliore rispetto a quella dei modelli più semplici (AR(1)). Dato che il miglioramento è significativo, si sceglie come migliore il modello AR(3).

In Figura 13 è possibile vedere i risultati ottenuti utilizzando il modello AR sui valori di temperatura, umidità e pressione. In blu sono mostrati i valori attesi mentre in rosso quelli ottenuti con la previsione del modello AR. In Tabella 1 è possibile vedere i calcoli dell'errore medio assoluto (MAE) e dell'errore medio quadrato (MSE).

I risultati ottenuti per i valori di temperature (Figura 13a) sono considerati buoni per quanto riguarda il MSE, ma si è voluto cercare un modello più complesso che permettesse di approssimare in maniera più efficace l'andamento dei valori della temperatura.

Metodo AR				
	2 h		24 h	
	MAE	MSE	MAE	MSE
temperatura	0.156	0.038	0.550	0.392
umidità	0.684	0.607	2.192	8.942
pressione	0.067	0.005	2.075	5.859

Tabella 1: MAE e MSE ottenuto dalle predizioni di 2 ore e 24 ore sui dati di temperatura, umidità e pressione utilizzando il metodo AR e un training set di 12 giorni

Metodo MA				
	2 h		24 h	
	MAE	MSE	MAE	MSE
temperatura	0.350	0.156	0.426	0.242
umidità	9.886	100.737	2.192	74.885
pressione	2.262	5.592	2.075	1.825

Tabella 2: MAE e MSE ottenuto dalle predizioni di 2 ore e 24 ore sui dati di temperatura, umidità e pressione utilizzando il metodo MA e un training set di 12 giorni

Per quanto riguarda l'umidità (Figura 13b) e la pressione (Figura 13c) sono stati svolti ragionamenti analoghi a quelli svolti sulla temperatura per trovare il parametro ottimale per il modello AR, ma per queste misurazioni l'MSE riscontrato è risultato troppo elevato per potersi considerare soddisfatti di un modello così semplice.

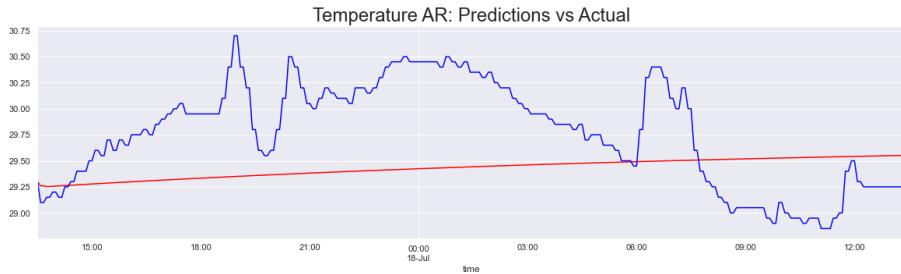
Moving Average Model (MA)

Il modello viene utilizzato nel caso in cui ci siano altri fattori oltre ai valori precedenti delle variabili che devono essere tenuti in considerazione. Pertanto, sapendo l'errore nelle previsioni precedenti, è possibile stimare in maniera migliore i nuovi valori. Questo è il motivo per cui i modelli MA incorporano i residui dei valori passati per migliorare le stime future. I modelli MA permettono di gestire in maniera più efficace gli imprevisti, per questo motivo sono noti come modelli di smoothing. Un MA di parametro 1 è definito attraverso la seguente formula[1]:

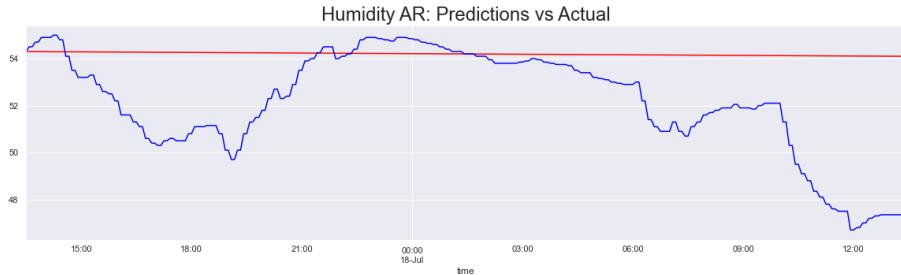
$$r_t = c + \Theta_1 \epsilon_{t-1} + \epsilon_t$$

Dove r è il valore attuale stimato, Θ_1 è il coefficiente associato al primo lag, ϵ_t e ϵ_{t-1} sono rispettivamente il valore residuo per il periodo corrente e il periodo precedente.

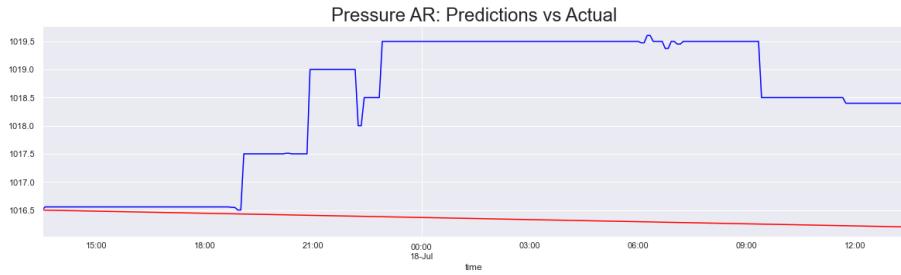
In Figura 14 è possibile vedere i risultati ottenuti applicando MA ai dati di temperatura, umidità e pressione, in blu sono mostrati i valori attesi mentre in rosso i valori predetti dal modello. In Tabella 2 sono mostrati gli errori



(a) Predizioni AR per 24 ore sui valori di temperatura



(b) Predizioni AR per 24 ore sui valori di umidità



(c) Predizioni AR per 24 ore sui valori di pressione

Figura 13: Previsioni ottenute dal modello AR applicato ai dati di 11 giorni di temperatura, umidità e pressione per l'arco di una giornata. In rosso sono mostrati i valori predetti e in blu quelli attesi.

MAE e MSE. Il MSE ottenuto è buono per i valori di temperatura, ma non per quelli di umidità e pressione. Inoltre in tutti i casi la previsione consiste fondamentalmente di una linea retta e non è in grado di seguire l'andamento dei dati.

Per questo motivo sono stati tentati approcci più complessi.

ARMA

Si è passati all'analisi del modello ARMA che permette di combinare i benefici di AR e MA. AR infatti permette di effettuare buone previsioni sui dati stazionari, ma è fallimentare nella gestione degli imprevisti, MA permette di mitigare questo side-effect. Il modello ARMA con parametri (1,1) è rappresentato dalla seguente equazione[1]:

$$r_t = c + \Phi r_{t-1} + \Theta_1 \epsilon_{t-1} + \epsilon_t$$

Dove r_t e r_{t-1} sono i valori attuale e precedente, ϵ_t e ϵ_{t-1} sono rispettivamente il valore residuo per il periodo corrente e il periodo precedente, Θ_1 e Φ sono due coefficienti che rappresentano rispettivamente quanta parte dell'errore e del valore precedente sono rilevanti nel calcolo del valore successivo.

Seguendo la linea utilizzata negli esperimenti precedenti, si è testato il modello ARMA(p,q) scegliendo p e q in base ai risultati ottenuti nell'elaborazione dei modelli AR e MA. Sono stati quindi selezionati i modelli ARMA(p,q) tali per cui tutti i parametri ottenuti dal modello fossero significativi e quindi non ci fosse rischio di overfitting. Infine è stata analizzata la log likelihood e l'information criteria (AIC) dei modelli rimasti e si è testato il modello che presentasse migliore likelihood e, contemporaneamente, AIC più bassa.

Per i valori di temperatura, umidità e pressione sono stati selezionati rispettivamente i modelli ARMA(2,1), ARMA(2,3) e ARMA(2,1). In Figura 15 si mostrano i risultati ottenuti dove in rosso sono mostrati i valori predetti e in blu i valori attesi mentre in Tabella 3 si mostrano i valori di MAE e MSE calcolati.

Per quanto riguarda la temperatura il MSE risulta sempre poco significativo mentre per le altre raccolte di dati è molto elevato. Questo può essere dovuto al fatto che la temperatura è una serie stazionaria mentre l'umidità e la pressione non sono stazionarie. Per questo motivo viene testato il modello ARIMA che dovrebbe essere più efficace nel caso di serie non stazionarie.

ARIMA

ARIMA(p,d,q) è una versione integrata del modello ARMA(p,q). Ciò significa che i dati vengono integrati d volte per raggiungere una serie stazionaria. Il modello ARIMA con parametri (1,1,1) può essere rappresentato dall'equazione che segue[1]:

$$\Delta P_t = c + \Phi \Delta P_{t-1} + \Theta_1 \epsilon_{t-1} + \epsilon_t$$

Dove $\Delta P_t = P_t - P_{t-1}$ e P_t è il valore al tempo t , ϵ_t e ϵ_{t-1} sono rispettivamente il valore residuo per il periodo corrente e il periodo precedente, Θ_1 e Φ

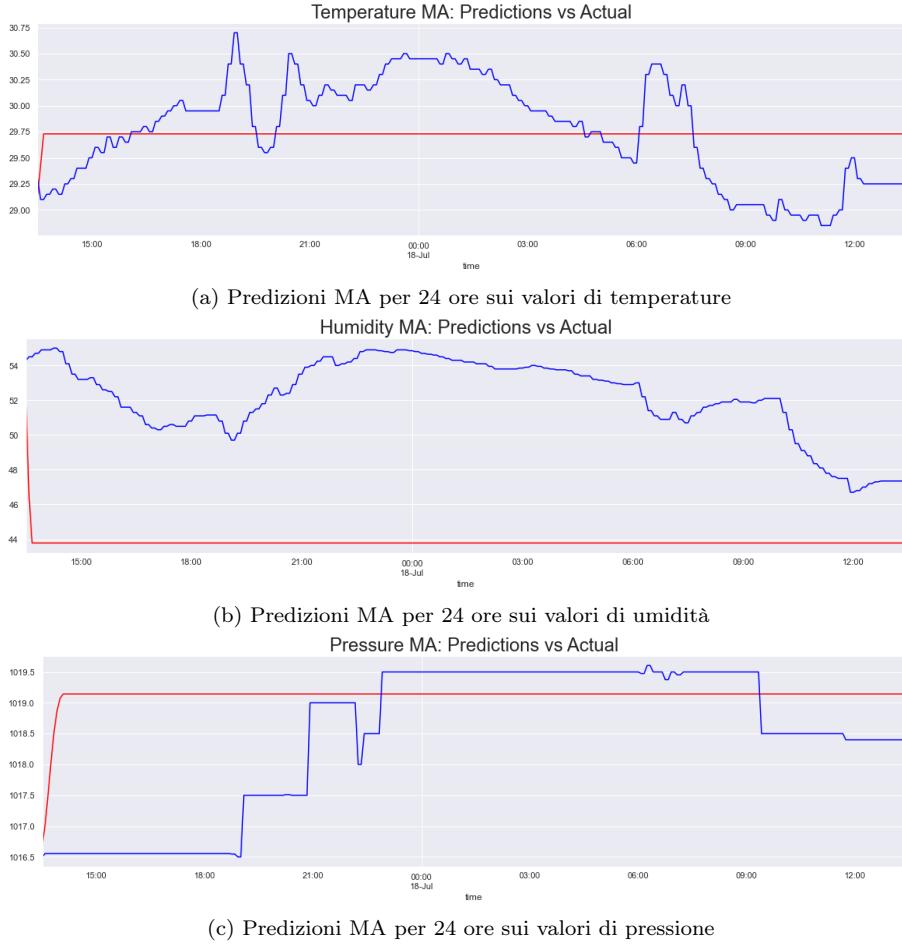
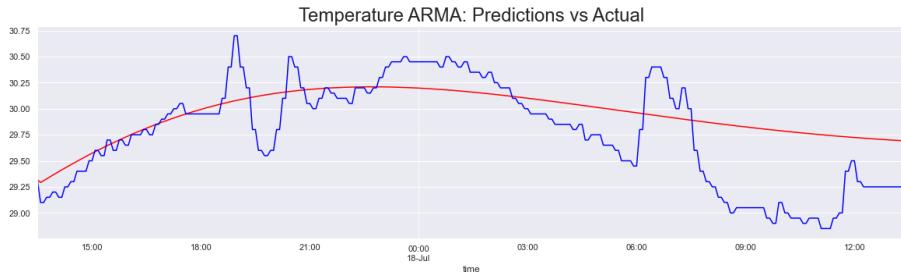


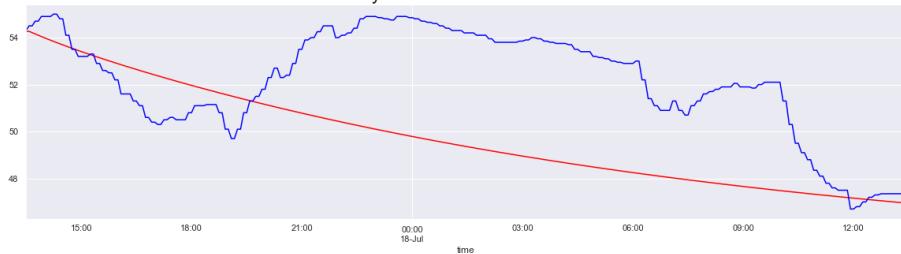
Figura 14: Previsioni ottenute dal modello MA applicato ai dati di 11 giorni di temperatura, umidità e pressione per l'arco di una giornata. In rosso sono mostrati i valori predetti e in blu quelli attesi.

Metodo ARMA				
	2 h		24 h	
	MAE	MSE	MAE	MSE
temperatura	0.129	0.022	0.301	0.155
umidità	0.495	0.410	2.837	11.465
pressione	0.024	0.001	1.160	2.062

Tabella 3: MAE e MSE ottenuto dalle previsioni di 2 ore e 24 ore sui dati di temperatura, umidità e pressione utilizzando il metodo ARMA e un training set di 12 giorni



(a) Predizioni ARMA per 24 ore sui valori di temperatura
Humidity ARMA: Predictions vs Actual



(b) Predizioni ARMA per 24 ore sui valori di umidità
Pressure ARMA: Predictions vs Actual



(c) Predizioni ARMA per 24 ore sui valori di pressione

Figura 15: Previsioni ottenute dal modello ARMA applicato ai dati di 11 giorni di temperatura, umidità e pressione per la previsione di una giornata. In rosso sono mostrati i valori predetti e in blu quelli attesi.

Metodo ARIMA				
	2 h		24 h	
	MAE	MSE	MAE	MSE
temperatura	0.171	0.047	0.603	0.516
umidità	0.694	0.667	2.340	9.883
pressione	0.053	0.003	1.924	5.109

Tabella 4: MAE e MSE ottenuto dalle predizioni di 2 ore e 24 ore sui dati di temperatura, umidità e pressione utilizzando il metodo ARIMA e un training set di 10 giorni

sono due coefficienti che rappresentano rispettivamente quanta parte dell'errore e del valore precedente sono rilevanti nel calcolo del valore successivo e c è una costante.

Come nei modelli precedenti sono stati testati diversi parametri per ARIMA(p,d,q), sono stati selezionati i modelli che presentavano tutti i coefficienti significativi e tra questi modelli è stato scelto quello con likelihood più alta e AIC più bassa. Per la temperatura, umidità e pressione sono stati scelti rispettivamente i modelli ARIMA(2,1,1), ARIMA(2,1,3) e ARIMA(2,1,1).

Per quanto riguarda la temperatura, dato che la serie è già risultata stazionaria, l'utilizzo di ARIMA non ha portato a miglioramenti nella likelihood rispetto a modelli più semplici. Anche per i modelli di umidità e pressione però non sono stati riscontrati i miglioramenti attesi. In Tabella 4 sono mostrati i valori di MAE e MSE ottenuti e in Figura 16 sono presentati i grafici ottenuti mettendo a confronto i valori attesi (in blu) e quelli predetti (in rosso).

SARIMA

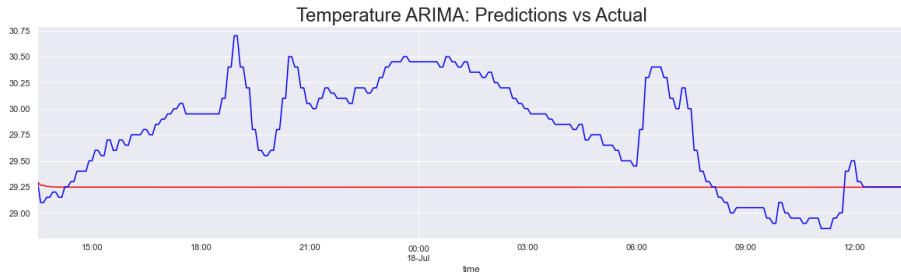
SARIMA è l'equivalente del modello ARIMA cui viene aggiunta una componente stagionale che permette di catturare pattern che non sono sempre presenti, ma occorrono periodicamente.

Sono stati svolti diversi tentativi di utilizzo del modello SARIMA, ma tale modello, anche con parametri bassi, risulta troppo complesso da calcolare rispetto ai corrispondenti modelli che non presentano stagionalità per questo motivo sono state tentate metodologie diverse.

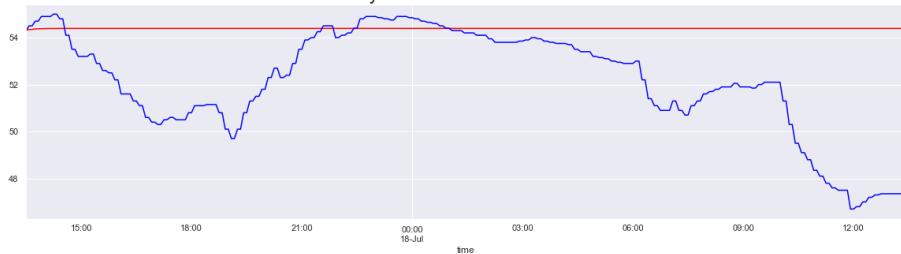
4.5.2 VARMA

Il metodo Vector Autoregression Moving-Average (VARMA) permette di generalizzare il modello ARMA su più serie temporali in maniera parallela (es. serie temporali multivariate) [4]. I parametri del modello sono analoghi a quelli utilizzati per AR(p) e MA(q).

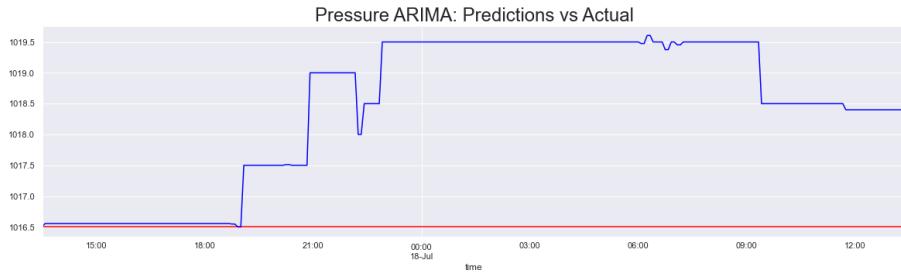
Nel nostro caso specifico è stata utilizzata una versione semplice di VARMA settando il parametro AR a 1 e quello MA a 1.



(a) Predizioni ARIMA per 24 ore sui valori di temperatura



(b) Predizioni ARIMA per 24 ore sui valori di umidità



(c) Predizioni ARIMA per 24 ore sui valori di pressione

Figura 16: Previsioni ottenute dal modello ARIMA applicato ai dati di 11 giorni di temperatura, umidità e pressione per la previsione di una giornata. In rosso sono mostrati i valori predetti e in blu quelli attesi.

Metodo VARMA				
	2 h		24 h	
	MAE	MSE	MAE	MSE
temperatura	0.186	0.040	0.933	1.596
umidità	1.981	4.506	2.308	8.391
pressione	0.880	0.831	0.731	0.892
n. persone	0.625	0.625	0.554	0.554

Tabella 5: MAE e MSE ottenuti dalle predizioni di 2 ore e di 24 ore sui dati di temperatura, umidità, pressione e numero di persone utilizzando il metodo VARMA e un training set di 12 giorni

Dato che il modello non presenta una componente di integrazione è stato necessario indurre a mano la stazionarietà delle serie integrando tutte le componenti una volta.

Il modello sembra adeguato all'utilizzo nel nostro specifico caso d'uso, ma i risultati (Tabella 5) ottenuti presentano nel caso dell'umidità un MSE molto elevato. In Figura 17 si mostrano i grafici che mettono a confronto i valori attesi (in blu) con i valori predetti (in rosso) di temperatura, umidità, pressione e numero di persone. Dato che i valori predetti non sono in grado di fornire un'approssimazione dell'andamento dei modelli, si sono tentati approcci diversi.

4.5.3 HWES

Nei metodi come ARIMA viene sviluppato un modello in cui la previsione è una somma lineare ponderata delle osservazioni e errori recenti. I metodi di Exponential Smoothing sono simili in quanto la previsione è una somma ponderata delle osservazioni passate, ma il modello utilizza esplicitamente un peso in diminuzione esponenziale per le osservazioni passate. Il modello Holt-Winters Exponential Smoothing che è stato utilizzato permette supporto per il trend e la stagionalità [7].

Dato che il modello permette di settare la componente di stagionalità, si è considerata come stagionalità il corso di una giornata che, avendo aggregato i dati in modo da avere un punto ogni 5 minuti, è settata a 288 (12 punti/ora * 24 ore). Il modello è allenato sui dati di 11 giorni e si è tentato di effettuare una previsione delle 24 ore successive.

In Figura 18 si mostrano i risultati predetti da HWES (linea rossa) confrontati con i dati reali (linea blu) mentre in Tabella 6 sono elencati il valori MAE e MSE. Il modello è stato in grado, rispetto ai precedenti, di seguire meglio l'andamento della curva dei dati reali, ma i risultati in termini di MSE non sono ancora considerati soddisfacenti.

4.5.4 LSTM

Il modello Long Short-Term Memory (LSTM) è un'architettura Recurrent Neural Network (RNN) utilizzata nel campo del deep learning. Le LSTM sono

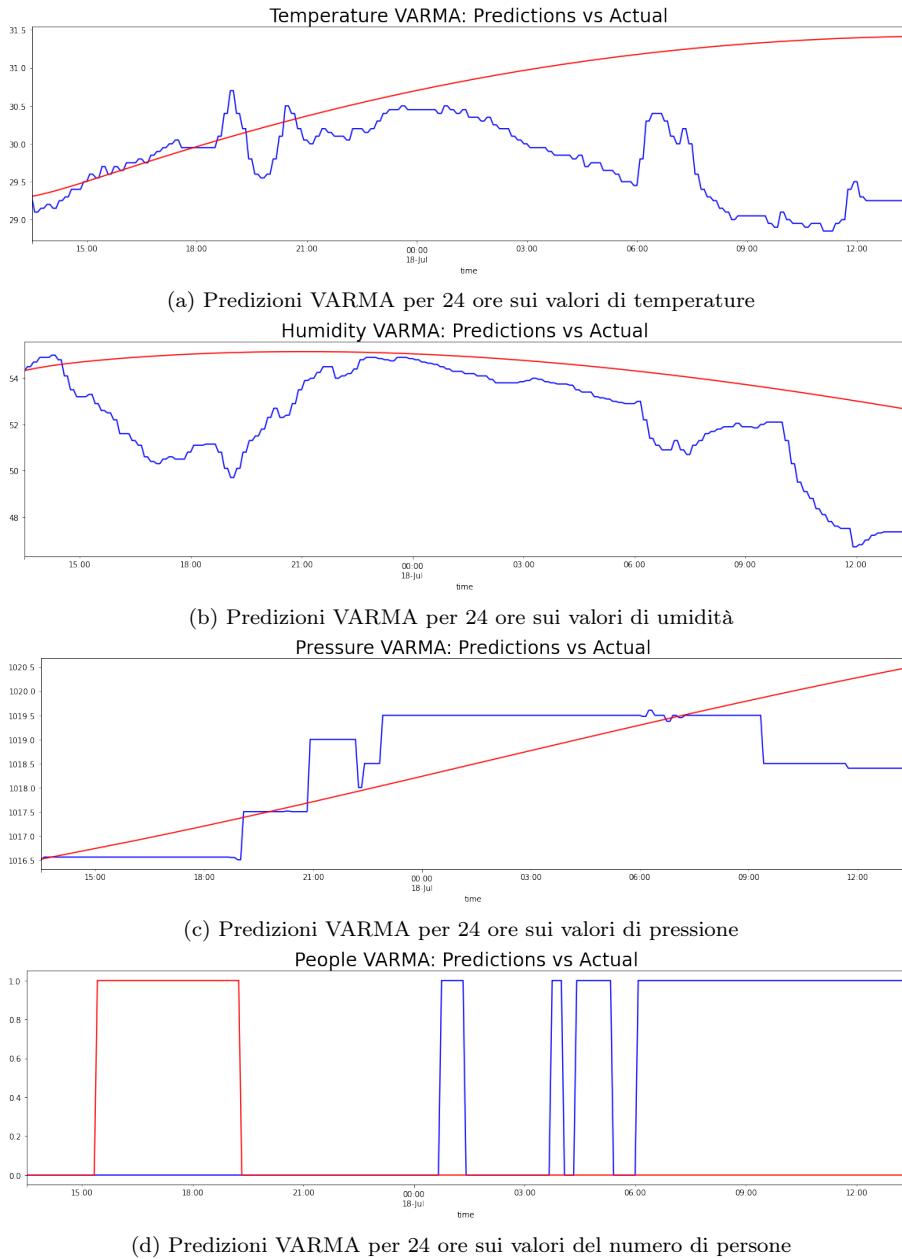


Figura 17: Previsioni ottenute dal modello VARMA applicato ai dati di 11 giorni di temperatura, umidità e pressione per la previsione di una giornata. In rosso sono mostrati i valori predetti e in blu quelli attesi.

Metodo HWES				
	2 h		24 h	
	MAE	MSE	MAE	MSE
temperatura	0.248	0.077	1.689	4.319
umidità	1.753	4.247	3.174	13.600
pressione	0.164	0.044	0.638	0.603

Tabella 6: MAE e MSE ottenuto dalle predizioni di 2 ore e 24 ore sui dati di temperatura, umidità e pressione utilizzando il metodo HWES e un training set di 12 giorni

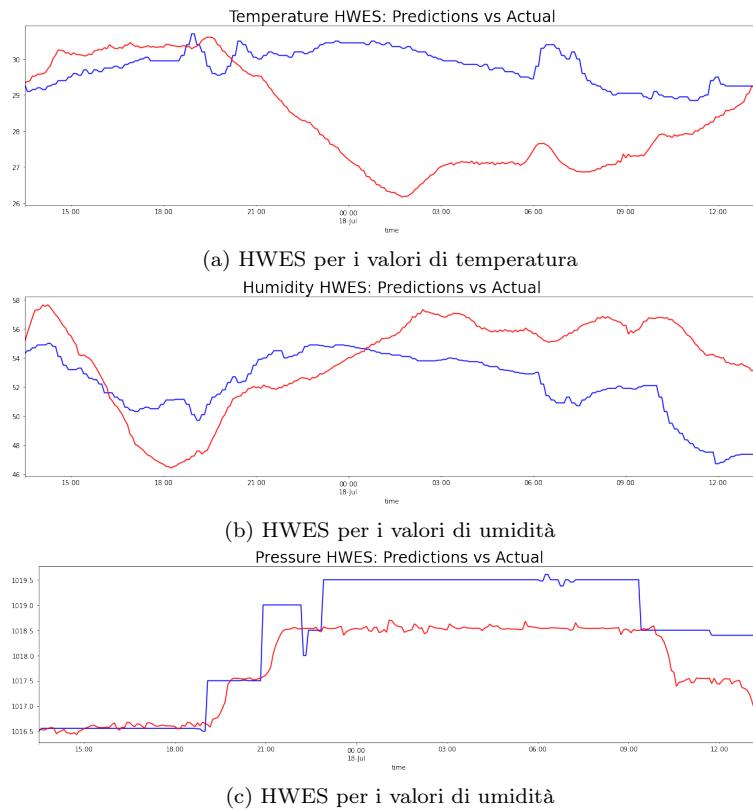


Figura 18: Previsioni ottenute dal modello HWES applicato ai dati di temperatura, umidità e pressione per l'arco di una giornata

RNN in grado di riconoscere schemi in delle sequenze di dati come le time-serie provenienti dai sensori.

Per preparare i dati per questo modello è stato necessario normalizzarli per inserirli in un intervallo $[0, 1]$. Ci si è posti come obiettivo quello di predire i risultati per le 24 ore successive ($24 \text{ ore} * 12 \text{ punti/ora} = 288 \text{ punti}$) basandosi sui valori del giorno precedente. Sia per il training set che per il validation set sono state create delle coppie in cui il primo elemento è la finestra temporale che precede il secondo e il secondo elemento è la sequenza di valori da predire. Per rendere più efficace la predizione, la finestra temporale non comprende tutti i punti dell'intervallo, ma è stato campionato solamente un punto ogni quarto d'ora e quindi le finestre temporali saranno composte da $4 * 24 = 96$ punti.

Risulta quindi possibile costruire con l'utilizzo delle librerie di tensorflow il modello di una rete neurale a tre livelli composta da due LSTM e un Dense layer finale. In un secondo momento è stato fatto quindi fitting del modello utilizzando 30 epoch.

Per quanto riguarda la previsione del numero di persone dall'interno della stanza, è stato usata una RNN a 3 livelli come quella presentata sopra, ma è stata adattata utilizzando come funzione di attivazione sigmoid che risulta molto efficace nella predizione di valori di probabilità e che quindi ci permettesse di ottenere un livello di probabilità sulla presenza o meno della persona all'interno dell'ambiente. Infine i valori ottenuti sono stati arrotondati per ottenere un valore binario.

Risulta semplice anche valutare il MSE del modello e salvarlo per poterlo utilizzare in un servizio di forecasting. In Tabella 7 è possibile vedere gli errori MAE e MSE ottenuti dal modello per una previsione di 2 e 24 ore e con una history di un giorno e di due giorni.

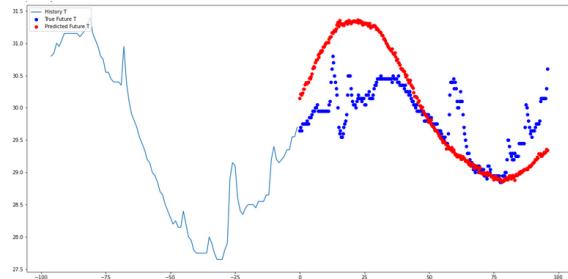
I modelli risultano meno efficaci rispetto ad altri in termini di MAE e MSE, ma si è tenuto in considerazione il fatto che, mentre gli altri modelli devo essere allenati ogni volta, questa rete può essere allenata meno spesso, per esempio una volta ogni 2/3 giorni. Inoltre, gli altri modelli hanno bisogno di dati di almeno 10 giorni e quindi, per ogni predizione, sarebbe necessario effettuare una richiesta di molti dati a InfluxDB. Questo modello invece permette di fornire risultati affidabili anche con una storia molto più breve e questo permette di effettuare query a Influx che richiedano meno tempo e un minore spostamento di dati. Infine si ritiene che, con l'aggiunta di nuovi dati, il modello LSTM possa diventare sempre più accurato nella previsione dei valori futuri.

Analizzando i risultati ottenuti dall'analisi dei modelli si è deciso quindi di utilizzare LSTM come modello per il servizio di forecasting facendo previsioni di 2 ore nel futuro.

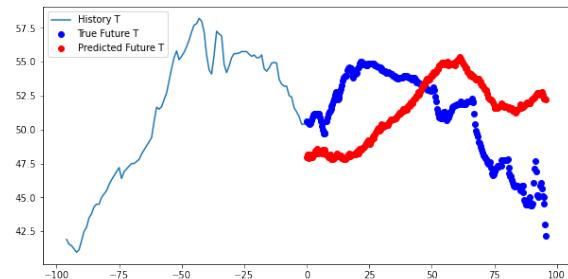
4.6 Servizio di forecasting

4.6.1 Calcolo della previsione

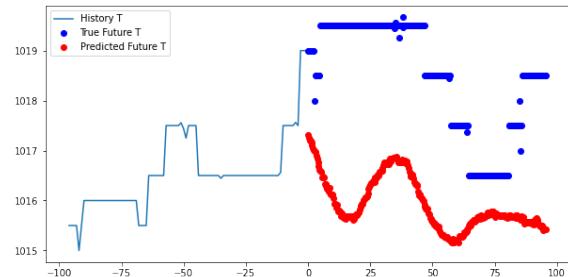
Utilizzando Python è stato costruito un servizio che si occupi di ricevere richieste di previsioni e, sfruttando i modelli per il forecasting di temperatura,



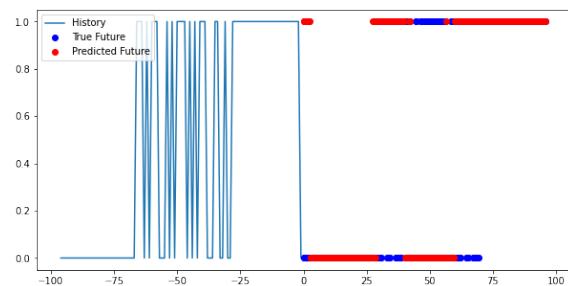
(a) Predizioni LSTM per 24 ore sui valori di temperatura



(b) Predizioni LSTM per 24 ore sui valori di umidità



(c) Predizioni LSTM per 24 ore sui valori di pressione



(d) Predizioni LSTM per 24 ore sui valori del numero di persone

Figura 19: Previsioni ottenute dal modello LSTM per 24 ore con uno storico di 24 ore per i valori di temperatura, umidità, pressione e numero di persone. I punti in rosso rappresentano i valori predetti mentre quelli in blu i valori attesi, in azzurro è rappresentata la finestra temporale utilizzata per predirre i nuovi valori

Metodo LSTM						
	1 day of history				2 days of history	
	2h prevision		24h prevision		2h prevision	
	MAE	MSE	MAE	MSE	MAE	MSE
temperatura	1.144	1.957	2.109	7.238	1.128	2.156
umidità	3.168	16.709	7.244	74.065	2.413	15.157
pressione	1.879	4.427	2.406	7.715	1.765	3.780
n persone	0.340	0.185	0.452	0.25	0.367	0.187

Tabella 7: MAE e MSE ottenuti dalle predizioni di un 2 ore e 24 ore con una history di 1 giorno e di 2 ore con una history di due giorni utilizzando il metodo LSTM e un traing set di 9 giorni

pressione, umidità e numero di persone, di calcolare la previsione per ognuna delle misurazioni per le due ore a seguire. Tale servizio utilizza come modelli i modelli ottenuti dalla rete neurale LSTM.

Il servizio deve essere in grado interrogare InfluxDB per ottenere i dati delle diverse misurazione delle ultime 24 ore. I dati ottenuti vengono normalizzati e forniti come input ai modelli salvati. Ottenuta una previsione per i valori di temperatura, umidità, pressione e numero di persone, viene creata una struttura json che è mandata in risposta alla richiesta di previsione.

4.6.2 Richiesta della previsione

Utilizzando Typescript è stato costruito un servizio, WriterService, che si occupa a intervalli regolari di:

1. Eseguire un richiesta di previsione verso il ForecastingService;
2. Elaborare la risposta ottenuta e creare un nuovo punto per ogni previsione ottenuta;
3. Caricare su InfluxDB tutti i punti contenenti le previsioni.

Dato che InfluxDB supporta caricamenti batch fino a 5000 punti, le scritture vengono effettuate con un unica chiamata al fine di non saturare di richieste il database.

4.7 Visualizzazione dei dati

La visualizzazione dei dati avviene attraverso le dashboard di Grafana. In Figura 20 sono mostrati i grafici su Grafana ottenuti interrogando InfluxDB per avere i dati, aggregati ogni 30 minuti, di temperatura, umidità, pressione e numero di persone.

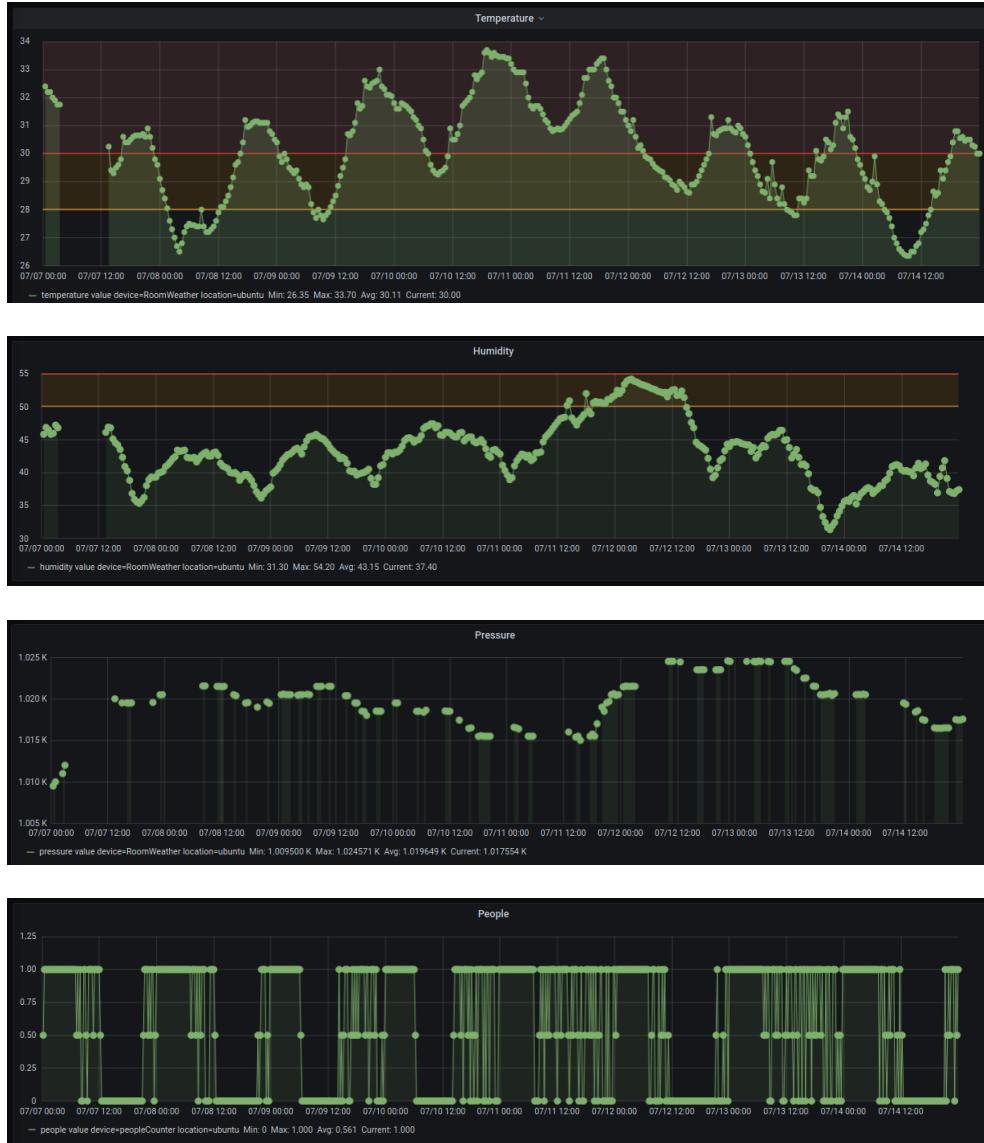


Figura 20: Visualizzazione attraverso Grafana delle dashboard di temperatura, umidità, pressione e numero di persone

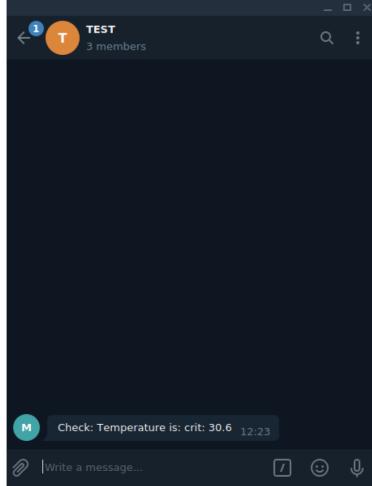


Figura 21: Bot di Telegram per la notifica di situazioni di "pericolo" nell'ambiente

4.8 Servizio di alerting

Utilizzando gli alert di InfluxDB è stato possibile costruire un servizio di alerting che permette di notificare l'utente tramite un bot di Telegram.

Su InfluxDB sono stati settati tramite script due alert che consentono di ricevere un avviso tramite il bot di Telegram quando i valori di temperatura e umidità superano una certa soglia preimpostata.

4.9 Script ausiliari

- Uno script che permette di creare un ambiente completo su InfluxDB con organizzazione, utente, buket e relativa autorizzazione;
- Uno script per la creazione degli alert su InfluxDB.

5 Risultati

Entrambe le Wot Thing costruite, RoomWeather e PeopleCounter, si sono dimostrate affidabili in fase di prototipazione e sono quindi state installate in una stanza per poter raccogliere i dati dell'ambiente.

La thing PeopleCounter è stata testata anche in contesti di luci diverse per valutare se l'influenza di luci esterne potesse creare problemi nel conteggio dei passaggi, dato che in un primo momento le fotoresistenze erano influenzate da altre luci molto forti si è deciso di applicare un filtro per la luce rossa al prototipo. Questa miglioria ha permesso di rendere il servizio efficiente in diversi contesti luminosi.

La problematica principale riscontrata consiste nel numero limitato di misurazioni che sono state prodotte, questo ha causato un'efficacia limitata nelle previsioni attraverso i metodi di forecasting.

Per quanto riguarda il modello di forecasting sui dati è stato selezionato LSTM che, nonostante non presentasse il miglior MSE, presenta grandi vantaggi in termini di efficienza nel momento in cui devono essere svolte le previsioni. I dettagli sull'analisi dell'errore calcolato per LSTM possono essere trovati nella Sezione 4.5.4.

6 Conclusioni e sviluppi futuri

Si ritiene che il sistema possa essere efficacemente utilizzato in contesti reali di monitoraggio di ambienti fornendo la possibilità di costruire a basso costo un sistema efficace e senza un eccessivo dispendio energetico. L'utilizzo di Docker e Docker Compose ha reso il sistema costruito semplice da riprodurre e aggiornare, in particolare è possibile sostituire il servizio di forecasting o aggiornare i modelli che utilizza semplicemente sostituendo un container.

Ci si propone di migliorare la rete neurale che fornisce la previsione dei risultati anche se ci si aspetta che un insieme di dati più grandi su cui fare training possa essere sufficiente per ottenere previsioni più efficaci. Inoltre si propone come sviluppo futuro quello di arricchire il servizio di forecasting con la possibilità di caricare la nuova rete LSTM allenata.

Riferimenti bibliografici

- [1] Dispense UNIPI. <http://users.dma.unipi.it/~flandoli/AUTCap4.pdf>.
- [2] Docker. <https://www.docker.com/>.
- [3] Grafana. <https://grafana.com/>.
- [4] Massimo Guidolin. Vector autoregressive moving average (varma) models.
Dispense Università Bocconi.
- [5] InfluxDB 2.0. <https://v2.docs.influxdata.com/v2.0/>.
- [6] InfluxDB bridge. <https://github.com/tim-hellhake/influxdb-bridge>.
- [7] Francesca Mazzucchi. Previsioni robuste con il lisciamento esponenziale ed il metodo di holt-winters. *Tesi di laurea di Corso di Laurea Specialistica in Scienze Statistiche*.
- [8] Mozilla WebThings. <https://iot.mozilla.org/>.