Programmazione per l'IoT - Laurea Magistrale in Informatica Applicata - DiSPeA - Università degli Studi di Urbino Carlo Bo
Data di pubblicazione xx/xx/xxxx - DOI: xxxx/xxxxxxx

Utilizzo di dispositivi "Internet of Things" e reti neurali convoluzionali per l'analisi di correlazione tra CO2 e Rumore in ambienti domestici

Matteo Marco Montanari[1]*, Emanuele Lattanzi[2]

Sommario
In questo articolo presentiamo un sistema IoT per la raccolta e l'analisi di dati sperimentali riguardanti i livelli di CO2 e rumore in ambienti domestici. Utilizziamo un microcontrollore ESP32 per la raccolta e l'invio dei dati su un server IoT e una Raspberry Pi per l'analisi di tali dati tramite reti neurali convoluzionali. Questo sistema IoT permette di avere un ridotto consumo energetico e un basso costo di realizzazione, caratteristiche che lo rendono impiegabile agevolmente anche in ambienti domestici. I risultati mostrano una correlazione tra i livelli di CO2 e rumore rilevati sperimentalmente, evidenziando l'efficacia del nostro sistema di registrare ed individuare le relazioni che intercorrono tra variabili di natura fisica, quali CO2 e rumore, in ambienti chiusi e controllati.

Keywords
Internet of Things, Neural Networks, CNN, ESP32, Raspberry Pi 3, Arduino, home monitoring

[1] *Laurea Magistrale in Informatica Applicata, Università degli Studi di Urbino Carlo Bo, Urbino, Italia*
[2] *Docente di Programmazione per l'Internet of Things, Università degli Studi di Urbino Carlo Bo, Urbino, Italia*
***Corresponding author**: m.montanari41@campus.uniurb.it

Introduzione

L'**Internet of Things (IoT)** è quella branca dell'informatica che si occupa della progettazione e sviluppo di sistemi che hanno come scopo l'interconnessione tramite Internet di oggetti provenienti dal mondo fisico ("things") dotati in particolare di sensori ed attuatori eventualmente provvisti di software di **Intelligenza Artificiale**. Sostanzialmente, l'IoT viene attuato collegando le interfacce del mondo reale a Internet: **sensori**, il cui scopo è fornire dati; e **attuatori**, che agiscono sull'ambiente circostante. Questa interconnessione tenta di colmare il divario tra il mondo fisico e quello virtuale/cibernetico, separazione che persiste sin dagli albori dell'informatica. Come esplicitato nel libro "**Internet of Things: Concepts and System Design**" [2], *l'IoT potenzia Internet aggiungendovi la dimensione del mondo reale. Con l'incorporazione dell'IoT, Internet diventa una rete di persone, informazioni, servizi e cose, essenzialmente l'Internet di tutto ("Internet of Everything").*

La qualità dell'aria e il rumore ambientale sono due fattori critici che influenzano la salute e il benessere umano. Questo studio si propone di esplorare la **correlazione** tra i livelli di CO2 e rumore in ambienti domestici utilizzando, in particolare, le **reti neurali artificiali** per modellare e prevedere le relazioni intercorrenti tra queste due variabili fisiche. In particolare, l'obiettivo principale di questo studio è realizzare questa analisi tramite l'utilizzo di **device IoT** a basso costo e ridotto consumo energetico. In genere, questi dispositivi vanno dai normali oggetti domestici a sofisticati strumenti industriali, noi ci concentreremo sui primi. Con oltre 7 miliardi di dispositivi IoT connessi oggigiorno, gli esperti si aspettano che questo numero cresca a 22 miliardi entro il 2025 [1].

Il resto dell'articolo è **organizzato** come segue: la prima parte descrive i metodi e i dispositivi utilizzati per raccogliere ed elaborare i dati, nella seconda parte vengono invece mostrati e commentati i risultati ottenuti, infine, nelle conclusioni sono esplicitati i pro e contro di questo studio assieme a un'analisi degli sviluppi futuri.

1. Metodi e dispositivi utilizzati

Il sistema da noi presentato è composto da due parti: la **raccolta dei dati** su un server IoT tramite ESP32 e l'**elaborazione** di tali dati su Raspberry Pi attraverso delle reti neurali artificiali.

1.1 Dispositivi e software utilizzati

Per la raccolta e l'analisi dei dati sono stati utilizzati i seguenti **dispositivi**:

1. **ESP32**: fa parte di una serie di microcontrollori System on a Chip (SoC) a basso costo e ridotto consumo energetico, dotati di Wi-Fi e Bluetooth integrati. ESP32 è prodotto e sviluppato da Espressif Systems [3] ed è la versione migliorata del microcontrollore ESP8266.
2. **Raspberry Pi Model 3**: è un Single Board Computer (SBC) sviluppato dalla Raspberry Pi Foundation ed è stato progettato per scopi didattici e di ricerca. Questo

dispositivo compatto esegue un sistema operativo basato su Linux chiamato **Raspbian** e offre una esperienza utente molto simile a quella di un moderno PC desktop di dimensioni e costo notevolmente ridotti [4].

3. **Sensore MH-Z19B (CO2)**: modulo per la rilevazione di gas tramite un sensore di piccole dimensioni che utilizza in particolare **infrarossi non dispersivi (NDIR)** per rilevare la presenza di CO2 nell'aria.
4. **Sensore ARD2-2238 (rumore)**: semplice dispositivo di dimensioni ridotte dotato di un microfono per rilevare l'intensità del suono. Presenta sia un'uscita digitale che un'uscita analogica. Il pin analogico emette un segnale di tensione in tempo reale proveniente dal microfono. La soglia di sensibilità del sensore può essere regolata tramite il potenziometro posizionato su di esso.

I **software** e framework necessari per il funzionamento del sistema IoT sono:

1. **Framework Arduino**: è una piattaforma di sviluppo open source realizzata principalmente per la programmazione di microcontrollori tramite i linguaggi **C/C++** in contesti didattici e di ricerca [5].
2. **InfluxDB**: è un database per serie temporali (TSDB) open source sviluppato dalla società InfluxData [6]. Viene utilizzato per l'archiviazione e il recupero di dati riguardanti serie temporali provenienti in particolare da sensori IoT. È scritto nel linguaggio di programmazione Rust.
3. **Tensorflow**: è uno dei più utilizzati framework open source per il Machine Learning e l'Intelligenza Artificiale. È sviluppato da Google e può eseguire su CPU, GPU e TPU (Tensor Processing Unit). Le API attualmente più stabili e diffuse sfruttano il linguaggio di programmazione **Python 3**. In particolare, per eseguire su Raspberry Pi Model 3 è stato necessario utilizzare una specifica versione del framework sviluppata appositamente per dispositivi embedded chiamata **Tensorflow Lite** [7].
4. **Google Colaboratory (Colab)**: è un servizio basato su tecnologia Jupyter Notebook ed è ospitato su **Google Cloud**, non richiede alcuna configurazione per l'utilizzo e fornisce accesso gratuito alle risorse di elaborazione, incluse GPU e TPU. Colab è particolarmente adatto per il Machine Learning, la **Data Science** e per scopi didattici [8].

1.2 Raccolta dati

Per la raccolta dei dati, abbiamo utilizzato un microcontrollore ESP32 equipaggiato con un sensore di CO2 (MH-Z19B) e un sensore di rumore (ARD2-2238). I dati raccolti sono stati trasmessi in tempo reale su un **server IoT** e conservati in un database per serie temporali (InfluxDB).

I dati di CO2 e rumore sono stati raccolti a intervalli regolari di 5 minuti per una durata totale di circa 5 giorni. Il microcontrollore ESP32 è stato programmato tramite il framework **Arduino** per acquisire e inviare i dati tramite Wi-Fi direttamente al server IoT.

Circuito Il circuito utilizzato per la raccolta dati è costituito da una ESP32 alimentata a 5V e collegata direttamente ai sensori di CO2 e rumore (MH-Z19B e ARD2-2238) come mostrato in Figura 1. Nel dettaglio:

1. **MH-Z19B (CO2)**: il pin V_{in} è alimentato a 5V, GND è connesso a massa. I due pin del seriale (RXD e TXD) sono connessi al seriale 2 di ESP32 (rispettivamente GPIO16 e GPIO17).
2. **ARD2-2238 (rumore)**: il pin A0 (analogico) è collegato al pin GPIO32 di ESP32. Il + è connesso a 3.3V di alimentazione e G è collegato a massa (GND). Il pin digitale (D0) non viene utilizzato.

Codice La tecnica di programmazione utilizzata per il campionamento e l'invio di dati è quella tipica dei sistemi ebbedded: il **superloop**. In sintesi, il chip ESP32 esegue all'infinito una serie di istruzioni costituite da:

1. misurazione dei valori di CO2 e rumore nella stanza;
2. invio dei dati raccolti sul server IoT;
3. attesa di 5 minuti prima di ricominciare il ciclo.

Il **codice completo** sviluppato per questo paper è disponibile al repository pubblico GitHub del primo autore [9].

Campionamento Per il campionamento dei valori di CO2 viene semplicemente letto il valore del sensore MH-Z19B a intervalli di 5 minuti tramite i pin collegati alla porta seriale n. 2 (RXD2 e TXD2). Per il campionamento del rumore, invece, assumendo una frequenza di campionamento del segnale di 22 kHz, occorre utilizzare degli accorgimenti specifici per l'esecuzione su sistemi con **poca memoria RAM** (520 kB) tipica dei sistemi embedded come ESP32 [10]. Dovendo campionare a 22 kHz per 10 secondi, sono necessari in totale 22 000 samples / sec $\times$ 10 sec = 220 000 samples. Considerando che ogni sample è un valore intero a 32 bit = 4 Byte allora sono necessari $4 \times 220\,000$ B $\approx$ 880 kB di memoria RAM per raccogliere i dati di una singola misurazione, che sono più della capienza massima della RAM disponibile su ESP32. Per questo motivo, si è deciso di calcolare la **media aritmetica** dei valori di rumore, raccolti in un intervallo di 10 secondi, direttamente sul microcontrollore ESP32 senza memorizzare di volta in volta tutti i campioni registrati. Si è predisposta dunque una variabile globale chiamata *somma* (inizializzata a 0) da utilizzare come **somma cumulativa** dei campioni ottenuti fino a quel momento. Per ottenere il valore medio di rumore ($\overline{N}$), da inviare al server IoT, occorre poi dividere la somma di tutti i campioni per il numero di samples acquisiti (220 000). Dovendo campionare a 22 kHz, il **delay** tra un sample e l'altro è di $1/22\,000 \approx 0.000\,045$ sec = 45 μ sec.

$$\overline{N} = \frac{1}{220\,000} \sum_{i=1}^{220\,000} N_i \tag{1}$$

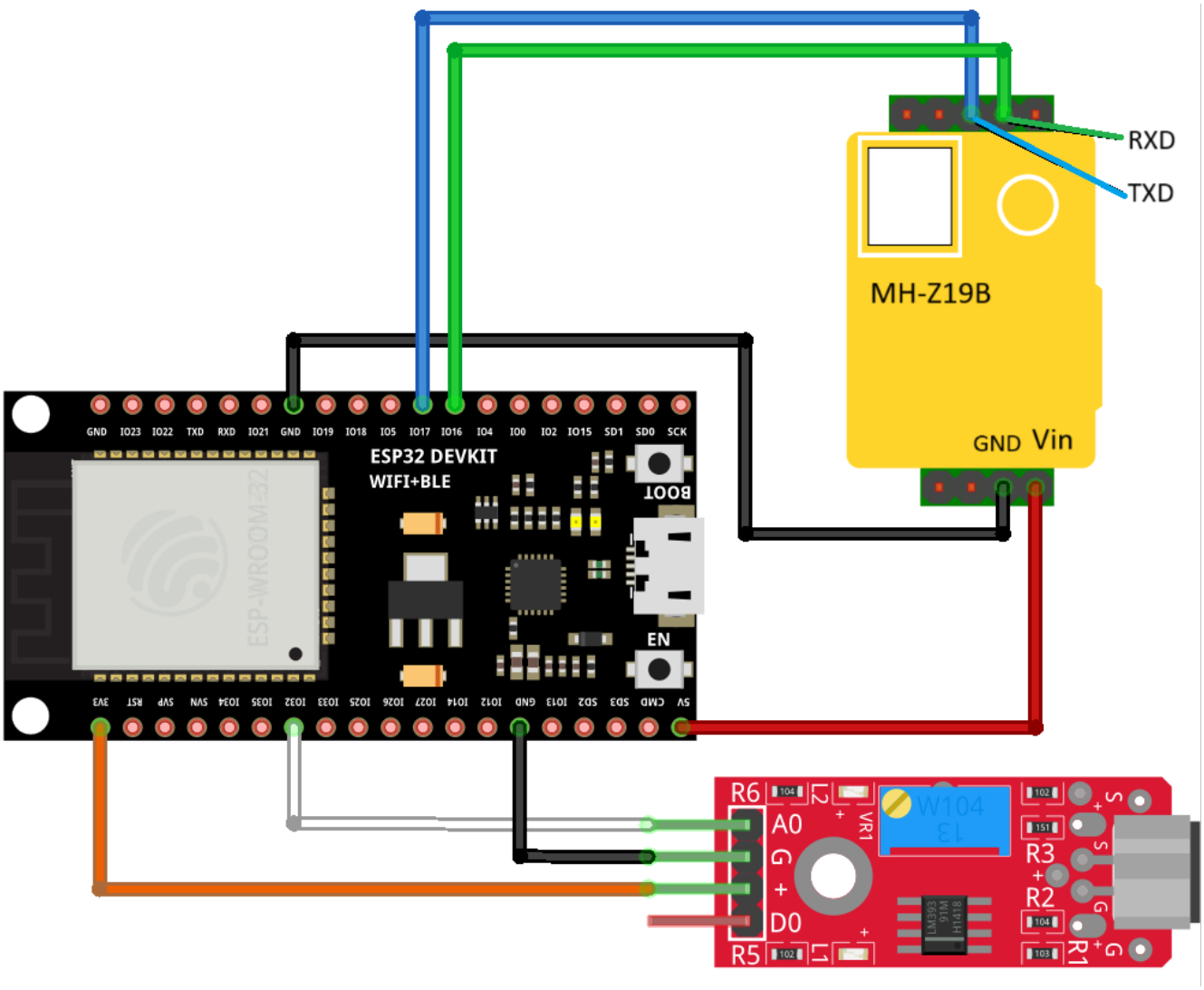

Figura 1. Circuito per la raccolta dati con ESP32, MH-Z19B e ARD2-2238

1.3 Elaborazione dei dati

Per lo studio di correlazione tra le due variabili fisiche analizzate abbiamo utilizzato un metodo di **Machine Learning** basato su reti neurali artificiali. I dati sperimentali ottenuti dai sensori sono stati utilizzati per addestrare una **rete neurale** allo scopo di individuare una relazione funzionale che rappresenti la correlazione tra le due variabili.

Definizione del Modello Per mantenere intatta la **natura temporale** dei dati raccolti, si è deciso di predire un singolo valore di una delle due variabili a partire da una sequenza di λ valori consecutivi dell'altra variabile. Assumendo quindi che il valore di CO2 sia la variabile indipendente mentre il rumore (*noise*) quella dipendente, l'obiettivo principale è stato di far apprendere alla rete neurale la **relazione funzionale** (dipendente da parametri Θ), $f_\Theta : \mathbb{R}^\lambda \mapsto \mathbb{R}$, intercorrente tra i dati raccolti sperimentalmente:

$$\text{noise} = f_\Theta(\text{CO2}^{(1)}, \ldots, \text{CO2}^{(\lambda)}) \tag{2}$$

Simmetricamente, assumendo invece che il rumore (*noise*) sia la variabile indipendente e la CO2 quella dipendente, l'**obiettivo secondario** è stato di far apprendere alla rete neurale la relazione funzionale (dipendente da parametri Φ), $h_\Phi : \mathbb{R}^\lambda \mapsto \mathbb{R}$, intercorrente tra i dati raccolti sperimentalmente:

$$\text{CO2} = h_\Phi(\text{noise}^{(1)}, \ldots, \text{noise}^{(\lambda)}) \tag{3}$$

Dataset Il dataset utilizzato per addestrare le reti neurali è composto dalla successione di tutte le coppie costituite dai valori di CO2 e rumore ottenuti sperimentalmente (al medesimo istante di tempo) tramite il sistema IoT da noi sviluppato. In simboli:

$$DataSet := \{(CO2_i, noise_i)\}_{i=1}^{N} \tag{4}$$

I dati raccolti sono stati rappresentati come serie temporali e sono mostrati in Figura 2.

Data Preprocessing Data la **natura temporale** dei dati raccolti, considerando il modello f_Θ, il dataset deve essere

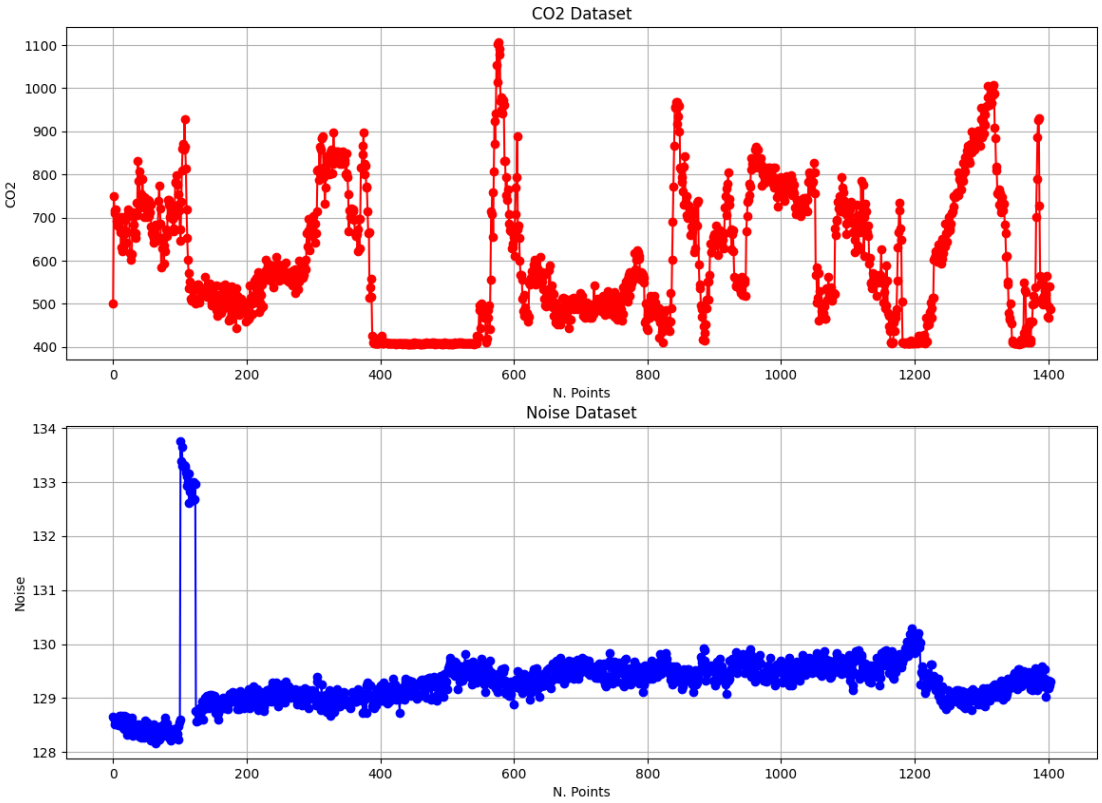

Figura 2. Dati raccolti tramite il sistema IoT da noi sviluppato.

preprocessato nel modo seguente prima di essere fornito in input alla rete neurale (se si utilizza il modello h_Φ, basta invertire $CO2_i$ con $noise_i$).

$$X_k := \{CO2_i\}_{i=k}^{k+\lambda-1} \qquad (k = 1,\dots N-\lambda) \tag{5}$$

$$y_k := [noise_i]_{i=k+\lambda} \qquad (k = 1,\dots N-\lambda) \tag{6}$$

Con $X_k \in \mathbb{R}^\lambda$ indichiamo i valori di input da fornire alla rete e con $y_k \in \mathbb{R}$ i corrispettivi **ground truth** da far predire alla rete. Inoltre, $noise_i$ e $CO2_i$ sono i valori di rumore e CO2 all'istante di tempo i-esimo, λ è la dimensione delle sequenze di input alla rete neurale e N è il numero totale di dati raccolti. Il codice scritto in linguaggio Python utilizzato per fare preprocessing è il seguente:

```
# Creiamo le sequenze di dati come input alla CNN
def create_sequences(X, y, seq_length=10):
    X_seq, y_seq = [], []
    for i in range(len(X) - seq_length):
        X_seq.append(X[i:i+seq_length])
        y_seq.append(y[i+seq_length])
    return np.array(X_seq), np.array(y_seq)

seq_length = 40  # Lunghezza delle sequenze
X_seq, y_seq = create_sequences(X, y, seq_length)
```

Addestramento della rete Pertanto, lo scopo delle reti neurali, implementate per l'analisi di correlazione tra CO2 e rumore, è quello di risolvere un problema di **regressione statistica** (non lineare): individuare una relazione funzionale (dipendente da parametri) tra variabili misurate sulla base di dati sperimentali. Questo problema, nell'ambito del Machine Learning, è un esempio di **apprendimento supervisionato** che prevede di minimizzare una opportuna funzione di costo per addestrare la rete neurale e individuare i **parametri ottimi** delle funzioni modello (Θ^*, Φ^*).
Per il modello f_Θ, in simboli:

$$\Theta^* := \arg\min_\Theta \frac{1}{N-\lambda} \sum_{k=1}^{N-\lambda} (f_\Theta(X_k) - y_k)^2 \tag{7}$$

Per il modello h_Φ, in simboli:

$$\Phi^* := \arg\min_\Phi \frac{1}{N-\lambda} \sum_{k=1}^{N-\lambda} (h_\Phi(X_k) - y_k)^2 \tag{8}$$

Dove la funzione di costo utilizzata è la **Mean Squared Error (MSE)** mentre N è il numero totale di campioni (CO2 e rumore) acquisiti durante la sperimentazione. Si ricorda che:

$$\arg\min_{x \in X} f(x) := \{x \mid \forall y : f(y) \geq f(x)\} \tag{9}$$

$$\text{MSE}(\mathbf{x}, \mathbf{y}) := \frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^2 \tag{10}$$

Pertanto, dopo l'addestramento, la rete neurale assumerà il ruolo di **modello parametrizzato** in base al valore dei dati sperimentali acquisiti: $NN^{\Theta} \equiv f_{\Theta^*}(\cdot)$ oppure $NN^{\Phi} \equiv h_{\Phi^*}(\cdot)$. Fornendo in input alla rete neurale una sequenza di λ dati ricavati sperimentalmente (di CO2 o rumore a seconda del modello utilizzato) otterremo il singolo **valore predetto** corrispondente (rumore o CO2 rispettivamente) all'istante di tempo successivo all'ultimo campione di input.

Reti neurali utilizzate Le **reti neurali convoluzionali (CNN)** sono una classe di reti neurali profonde (**deep learning**) particolarmente adatte per l'elaborazione di dati sequenziali come le serie temporali. A differenza delle classiche reti **Feed-Forward**, le CNN consentono di imparare automaticamente le features (**feature extraction**) tramite l'utilizzo di filtri convoluzionali chiamati **kernel**, che vengono applicati ripetutamente alle varie porzioni dei dati di input. Applicando più **layer convoluzionali** in cascata, è possibile estrarre features di *alto livello* a partire da caratteristiche più semplici individuate nei dati di input. Inoltre, le CNN sono reti neurali artificiali invarianti rispetto allo spazio (**Space Invariant Artificial Neural Networks - SIANN**), poiché si basano su un'architettura che condivide i kernel di convoluzione che scorrono lungo le dimensioni delle features di input, cosa che permette di risparmiare sul **numero di parametri** da addestrare. Per il nostro studio abbiamo dunque utilizzato le librerie *TensorFlow* e *Keras* per costruire due reti neurali di tipo Convolutional Neural Network (CNN) con lo scopo di individuare una relazione funzionale che rappresenti la correlazione tra le due variabili tramite un task di regressione su dati sperimentali. In simboli, un **layer convoluzionale** è rappresentato da:

$$H^{(m)} = \sigma(I * K^{(m)} + b^{(m)}) \qquad m = 1, \ldots, N_{ker} \tag{11}$$

Dove, $H^{(m)} \in \mathbb{R}^{\lambda - L + 1}$ è la **feature map** (risultato della convoluzione tra un filtro e l'input) corrispondente all'm-esimo kernel, $I \in \mathbb{R}^{\lambda}$ è il vettore di input di dimensione λ, $K^{(m)} \in \mathbb{R}^{L}$ è l'm-esimo kernel di dimensione L (un vettore contenente i pesi da dover addestrare), $b^{(m)} \in \mathbb{R}$ è il bias del filtro m, σ è la funzione di attivazione utilizzata e N_{ker} è il numero di kernel del layer. Il simbolo $*$ rappresenta l'operatore di **convoluzione** (più correttamente **cross-correlazione**) in 1D:

$$(I * K)[j] = \sum_{l=1}^{L} I[j + l] K[l] \qquad j = 0, \ldots, \lambda - L \tag{12}$$

Architetture utilizzate L'architettura che **empiricamente** risulta la migliore per l'apprendimento del modello f_{Θ}, che rappresenta la correlazione tra i valori di CO2 e rumore in base ai dati prodotti dal nostro esperimento, è la seguente (**CNN1**):

1. Layer di **Input** di dimensione $(\lambda, 1)$, con $\lambda = 40$.
2. Layer **Convoluzionale 1D** con 64 kernel di dimensione 3 e funzione di attivazione lineare.
3. Layer **Convoluzionale 1D** con 32 kernel di dimensione 2 e funzione di attivazione lineare.
4. Layer **Convoluzionale 1D** con 32 kernel di dimensione 2 e funzione di attivazione lineare.
5. Layer **Flatten** per adattare l'output degli strati convoluzionali agli strati densi finali.
6. Layer **Denso** con 30 neuroni e funzione di attivazione lineare.
7. Layer **Denso** con 1 neurone (per task di regressione) e funzione di attivazione lineare.

Di contro, l'architettura che empiricamente risulta la migliore per l'**apprendimento del modello** h_{Φ}, che rappresenta la correlazione tra i valori di rumore e CO2 in base ai dati prodotti dal nostro esperimento, è la seguente (**CNN2**):

1. Layer di **Input** di dimensione $(\lambda, 1)$, con $\lambda = 40$.
2. Layer **Convoluzionale 1D** con 64 kernel di dimensione 3 e funzione di attivazione lineare.
3. Layer **Convoluzionale 1D** con 32 kernel di dimensione 2 e funzione di attivazione lineare.
4. Layer **Flatten** per adattare l'output degli strati convoluzionali agli strati densi finali.
5. Layer **Denso** con 30 neuroni e funzione di attivazione lineare.
6. Layer **Denso** con 1 neurone (per task di regressione) e funzione di attivazione lineare.

Esecuzione su Raspberry Pi L'esecuzione dei modelli, precedentemente addestrati in Cloud tramite **Google Colaboratory**, è stata effettuata su un dispositivo embedded tipico dei sistemi IoT: **Raspberry Pi Model 3**. Il codice utilizzato per implementare le reti neurali è scritto in linguaggio **Python 3** e si basa sulle librerie di calcolo numerico: **NumPy, Matplotlib, Pandas** e sui framework di Intelligenza Artificiale e Machine Learning **TensorFlow e Tensorflow Lite**. Il codice completo è reperibile su GitHub [9].

2. Risultati

Mostriamo ora i risultati ottenuti sia nella fase di raccolta dati, attraverso il sistema IoT da noi sviluppato e testato; sia nella fase di elaborazione dati, dove abbiamo utilizzato delle reti neurali di tipo CNN per effettuare un'analisi di correlazione tra le variabili sperimentali precedentemente misurate dai sensori IoT.

Dati prodotti I dati prodotti dal sistema IoT da noi sviluppato sono mostrati in Figura 2, a dimostrazione del funzionamento del metodo e dei dispositivi da noi utilizzati allo scopo di raccogliere dati sperimentali riguardanti i livelli di CO2 e rumore in ambienti domestici.

Problema Diretto La prima rete neurale da noi sviluppata (CNN1) è stata in grado di predire i valori di rumore a partire dai dati sperimentali di CO2 immediatamente precedenti, collezionati in una finestra temporale di $\lambda = 40$ punti consecutivi. Il grafico che mostra l'**andamento dei valori di rumore** predetti ed effettivi, relativi ai dati di test, è mostrato in Figura 3 assieme all'errore assoluto. Questo grafico mostra l'effettiva capacità del nostro modello di rete neurale di **apprendere** la relazione funzionale corrispondente al modello f_Θ che individua la **correlazione** tra CO2 e rumore. Nella Figura 4 sono mostrati inoltre il grafico della distribuzione (tramite un istogramma) dell'**errore assoluto** calcolato sui dati di test e uno scatter plot che confronta i valori di rumore effettivi e predetti. In simboli (y_k sono i valori effettivi e y_k^{pred} i valori predetti dalla rete):

$$\text{Absolute Error}_k := |y_k - y_k^{pred}| \tag{13}$$

Problema Inverso Simmetricamente, considerando il problema inverso di predire i valori di CO2 a partire dai dati di rumore immediatamente precedenti, la seconda rete neurale da noi sviluppata (CNN2) non è stata così efficace a predire tali valori (si è utilizzata una finestra temporale di $\lambda = 40$ punti consecutivi). Il grafico che mostra l'**andamento dei valori di CO2** predetti ed effettivi, relativi ai dati di test, è mostrato in Figura 5 assieme all'errore assoluto. Questo grafico mostra una scarsa capacità del nostro modello di rete neurale di **apprendere** la relazione funzionale corrispondente al modello h_Φ, che dovrebbe individuare la **correlazione** tra rumore e CO2. Nella Figura 6 sono mostrati inoltre il grafico della distribuzione (tramite un istogramma) dell'**errore assoluto** calcolato sui dati di test e uno scatter plot che confronta i valori di CO2 effettivi e predetti.

Modelli a confronto La Tabella 1 mostra le due **metriche di valutazione** del modello per le due istanze di rete neurale addestrate (corrispondenti al problema diretto e inverso). MSE è la Mean Squared Error (**loss function**) mentre la **Mean Absolute Error (MAE)** è la media degli errori assoluti calcolata sui campioni di test; in simboli (y_k sono i valori effettivi e y_k^{pred} quelli predetti):

$$\text{MAE}(\mathbf{y}, \mathbf{y^{pred}}) := \frac{1}{N}\sum_{k=1}^{N} |y_k - y_k^{pred}| \tag{14}$$

Tabella 1. Metriche MSE e MAE sul test set per le due reti neurali addestrate (i dati di input non sono stati normalizzati).

NN	MSE	MAE
CNN1	0.294	0.432
CNN2	33444.141	154.797

Gli **scarsi risultati** della CNN2 sono dovuti, a nostro avviso, alla **bassa sensibilità** del sensore di rumore che ha prodotto di conseguenza dei dati che seguono un andamento pressocché lineare da cui è molto difficile ricavare i valori di CO2 ottenuti sperimentalmente, che sono invece molto variabili.

3. Conclusioni

In conclusione, il nostro sistema IoT a basso costo e ridotto consumo energetico si è rivelato un buon strumento per la rilevazione e la **raccolta di dati** sperimentali riguardanti i livelli di CO2 e rumore presenti in ambienti domestici controllati.

Anche l'analisi di tali dati, realizzata tramite delle reti neurali di tipo CNN, ha prodotto discreti risultati permettendoci di avere non solo una **analisi di correlazione** tra i valori di CO2 e rumore misurati sperimentalmente ma ci dota anche di uno **strumento di predizione** basato su tale correlazione (le reti neurali addestrate sul dataset prodotto). Tutto questo è molto importante per comprendere meglio il fenomeno fisico sottostante e per suggerire nuove tecniche di **monitoraggio** di parametri fisici in ambienti domestici tramite sistemi IoT a **impatto ridotto** e tecnologie di Intelligenza Artificiale.

Ulteriori sviluppi potrebbero riguardare l'utilizzo di altre tipologie di reti neurali per l'analisi di correlazione (nonché di altre tecniche di Machine Learning classiche), oltre la possibilità di sperimentare l'utilizzo di ulteriori dispositivi e sensori per la rilevazione di altre grandezze fisiche allo scopo di suggerire analisi di correlazione più complesse che le coinvolgano. In particolare, sembra promettente l'utilizzo delle reti neurali ricorrenti come le **Long Short Term Memory (LSTM)**, un'architettura progettata specificamente per modellare le dipendenze a lungo termine su dati sequenziali. Un iniziale tentativo in questo senso è stato fatto ma ancora non ci è stato possibile implementare tali tipologie di reti su Raspberry Pi Model 3 per motivi di **compatibilità** col framework **Tensorflow Lite**, cosa che impone lo spostamento su dispositivi diversi o l'utilizzo di hardware più recenti.

Riferimenti bibliografici

[1] Oracle Corporation. What is IoT? https://www.oracle.com/it/internet-of-things/what-is-iot/.

[2] Milan Milenkovic. *Internet of Things: Concepts and System Design*. Springer, 2020.

[3] Espressif Systems. ESP32. https://www.espressif.com/en/products/socs/esp32.

[4] Raspberry Pi. Raspberry Pi Model 3 B. https://www.raspberrypi.com/products/raspberry-pi-3-model-b/.

[5] Arduino Software. https://www.arduino.cc/en/software.

[6] Influx Data. InfluxDB. https://www.influxdata.com/.

[7] TensorFlow Lite. https://www.tensorflow.org/lite?hl=it.

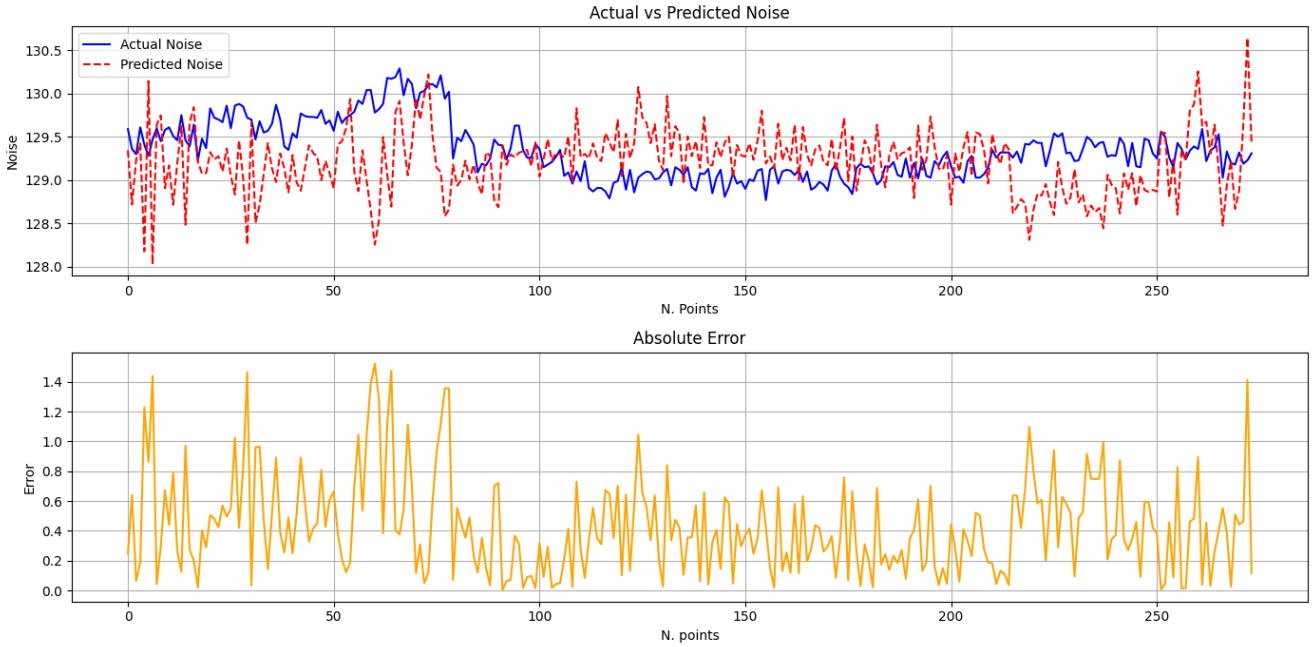

Figura 3. (**Problema diretto**) Andamento dei valori di rumore, predetti ed effettivi, relativi ai dati di test e grafico dell’errore assoluto.

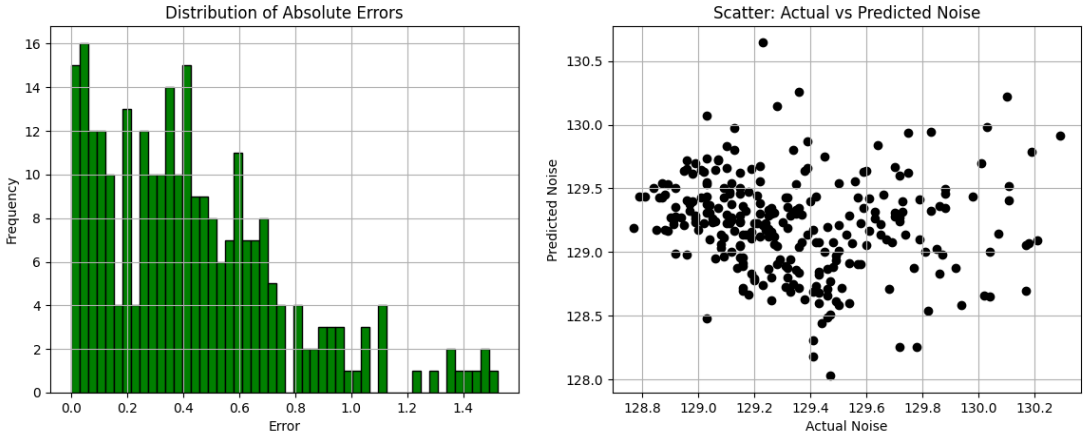

Figura 4. (**Problema diretto**) Istogramma di distribuzione dell’errore assoluto calcolato sui dati di test e uno scatter plot che confronta i valori di rumore effettivi e predetti.

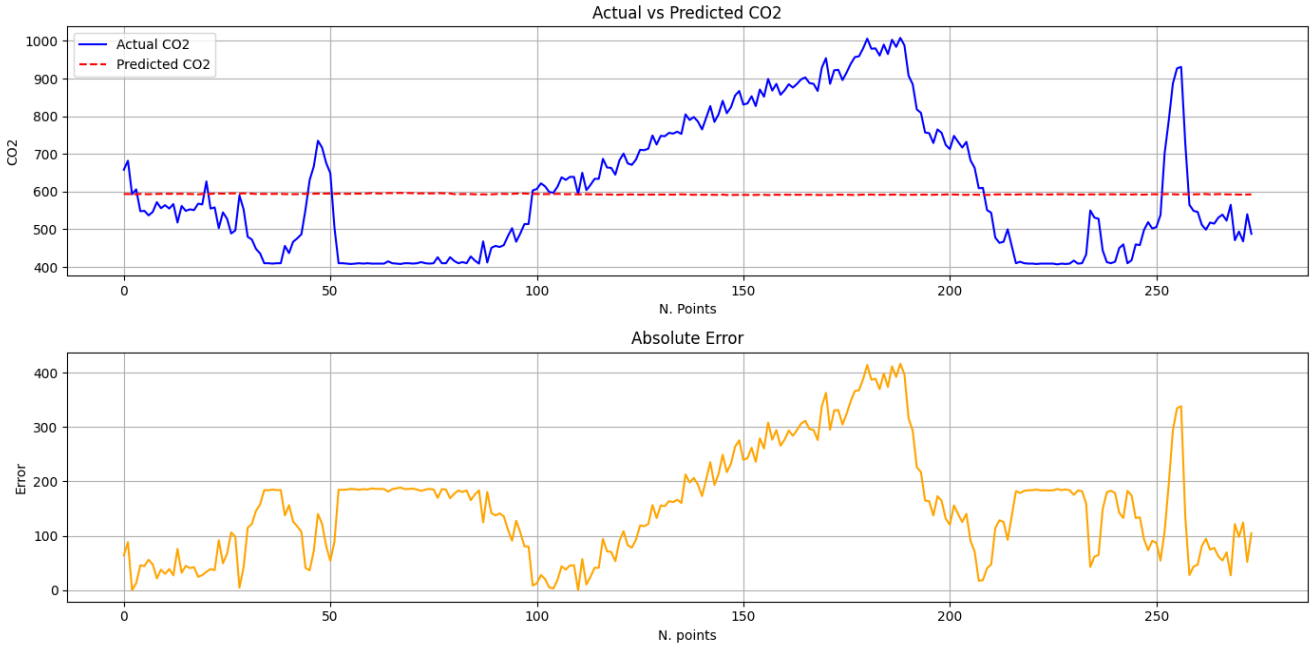

Figura 5. (**Problema inverso**) Andamento dei valori di CO2, predetti ed effettivi, relativi ai dati di test e grafico dell’errore assoluto.

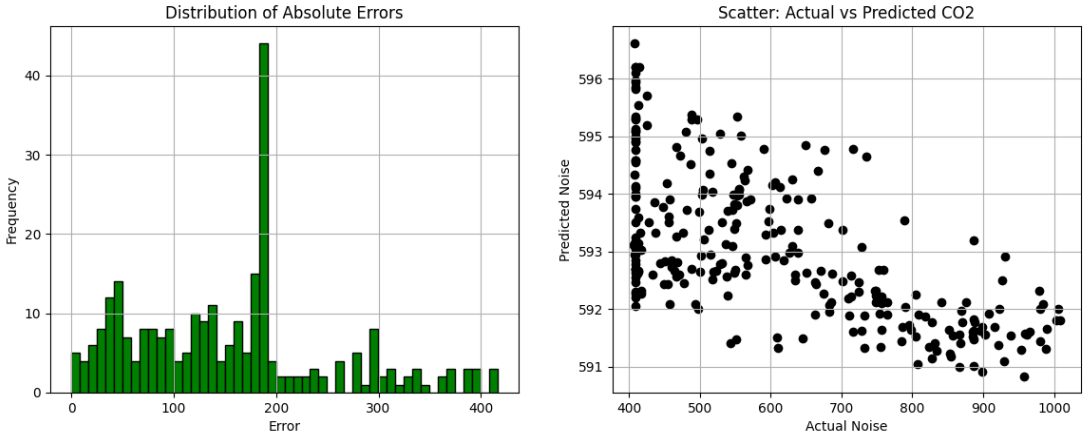

Figura 6. (**Problema inverso**) Istogramma di distribuzione dell’errore assoluto calcolato sui dati di test e uno scatter plot che confronta i valori di CO2 effettivi e predetti

[8] Google Colaboratory. https://colab.google/.

[9] Matteo Marco Montanari. GitHub source code repository. https://github.com/MatteoMarcoM/Progetto_IoT/.

[10] Espressif Systems. ESP-IDF Programming Guide. https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/memory-types.html.