Programmazione per l'IoT - Laurea Magistrale in Informatica e Innovazione Digitale - DiSPeA - Università degli Studi di Urbino Carlo Bo
Data di pubblicazione XX/XX/XXXX - DOI: xxxx/xxxxxxx

Smart Sensing con ESP32 e TensorFlow Lite: Predizione e Continuità nel Monitoraggio Ambientale

Marzio Della Bosca[1]*, Susanna Peretti[1]*, Emanuele Lattanzi[2]

Sommario
Questo progetto si colloca nell'ambito dell'"Internet of Things" (IoT), proponendo una soluzione resiliente per la raccolta di dati ambientali mediante l'impiego di dispositivi edge. L'architettura sviluppata è costituita da quattro nodi ESP32: due nodi di sensing, ciascuno dedicato al monitoraggio di una specifica grandezza fisica (temperatura e umidità), un nodo incaricato delle inferenze e un nodo Edge che oltre a fare da intermediario per i nodi che eseguono i campionamenti e il database cloud InfluxDB, gestisce la comunicazione tra i dispositivi tramite il protocollo ESP-NOW. In caso di malfunzionamento di uno dei nodi di sensing il nodo Edge attiva il nodo di predizione, che esegue reti neurali pre-addestrate (convertite tramite TensorFlow Lite) per stimare i prossimi valori. Rimane attivo finché uno o più nodi di sensing non funzionano e termina quando entrambi tornano operativi, grazie ad un messaggio inviato dal nodo Edge.

Keywords
IoT — ESP32 — ESP-NOW — InfluxDB — TensorFlowLite

[1] *Laurea Magistrale in Informatica e Innovazione Digitale, 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.dellabosca@campus.uniurb.it, s.peretti@campus.uniurb.it

Introduzione

Negli ultimi anni il monitoraggio ambientale ha assunto un ruolo sempre più centrale in molti contesti, dalla gestione urbana alla protezione degli ecosistemi, fino all'ottimizzazione di impianti agricoli e industriali. In questo contesto l'Internet of Things (IoT) si pone come una tecnologia di primaria importanza, in quanto è in grado di trasformare semplici punti di raccolta dati in nodi intelligenti e interconnessi. Grazie alla crescente disponibilità di sensori economici e a basso consumo, assieme a microcontrollori sempre più performanti e compatti è sempre più possibile realizzare reti distribuite per la sorveglianza di parametri ambientali come temperatura, umidità, pressione atmosferica, qualità dell'aria e altri indicatori climatici.

Il sistema presentato in questo progetto è composto da quattro nodi ESP32, ciascuno con un ruolo specifico: due nodi di sensing dedicati rispettivamente alla rilevazione della temperatura e dell'umidità; un nodo Edge incaricato della raccolta e inoltro dei dati verso un database cloud InfluxDB mediante tecnologia WiFi e un nodo di inferenza, attivato in presenza di anomalie nei nodi di sensing.

Il nodo Edge monitora costantemente l'attività dei sensori, e in caso di inattività prolungata invia al nodo di inferenza una sequenza contenente le ultime quattro misurazioni valide. Il nodo di inferenza, grazie a reti neurali leggere e pre-addestrate (convertite in TensorFlow Lite), stima i valori mancanti e li trasmette insieme alle proprie letture, garantendo la continuità operativa della rete.

Oltre a migliorare la resilienza del sistema questa architettura consente di esplorare l'applicazione di tecniche di machine learning lightweight su piattaforme embedded, mantenendo un equilibrio tra accuratezza, efficienza energetica e resilieanza della rete.

1. Hardware

1.1 Architettura ESP32

Il cuore del nostro sistema è rappresentato dall'ESP32 [1], un SoC (System-on-Chip) sviluppato da Espressif Systems, un chip altamente integrato, caratterizzato da elevate prestazioni, basso consumo energetico e un costo contenuto. Le principali caratteristiche tecniche dell'ESP32 sono:

- **CPU**: Dual-core Xtensa a 32 bit, fino a 240 MHz
- **Ultra-low-power (ULP)**
- **Memoria**: 520 KiB di RAM interna
- **Connettività wireless**:
  - Wi-Fi 802.11 b/g/n
  - Bluetooth v4.2 con supporto BR/EDR e BLE
- **Interfacce periferiche**:

- 34 GPIO programmabili
- 2 ADC SAR a 12 bit (fino a 18 canali)
- 2 DAC a 8 bit

Per la programmazione dell'ESP32 è stato utilizzato l'IDE di Arduino, basato sul linguaggio C++.

1.2 DHT11 Digital Sensor

Il DHT11 è un sensore digitale combinato che integra due elementi di misura, uno per la temperatura e uno per l'umidità semplificando l'interfacciamento con il microcontrollore dato che non è necessario convertire segnali analogici né applicare formule complesse per ottenere i dati.

La misura dell'umidità si basa su un componente resistivo che varia il proprio valore elettrico in funzione della quantità di vapore acqueo presente nell'aria. La temperatura invece viene rilevata tramite un termistore NTC (Negative Temperature Coefficient), ovvero una resistenza che diminuisce al crescere della temperatura. Entrambi i sensori sono collegati a un microcontrollore integrato a 8 bit, che gestisce l'elaborazione, la trasmissione dei dati e, all'occorrenza, la loro quantizzazione.

```
#include DHT.h
#define DHTTYPE DHT11
#define dht_dpin 25
DHT dht(dht_dpin, DHTTYPE);
```

La configurazione illustrata in Figura 1 evidenzia la semplicità del collegamento tra ESP32 e sensore DHT11, basato su una connessione seriale (*Single-Wire Two-Way*).

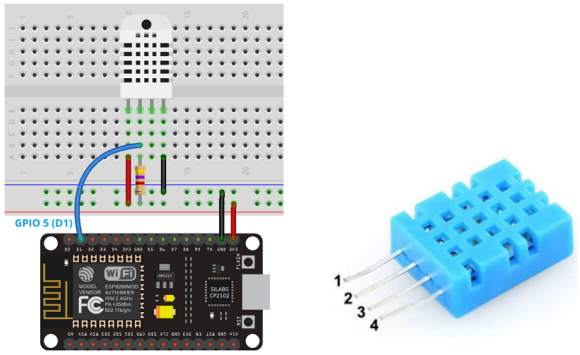

Figura 1. Schema collegamento ESP32-DHT11

Il DHT11 è dotato di quattro pin, ciascuno con una funzione specifica:

- pin 1: Vcc → (3V3)
- pin 2: Out → (GPIO25)
- pin 3: non utilizzato
- pin 4: GND → GND

2. Comunicazione

La comunicazione tra i nodi è stata implementata utilizzando il protocollo **ESP-NOW** [2], un protocollo di comunicazione sviluppato da Espressif che consente la trasmissione di dati tra dispositivi ESP32 (e compatibili). Dato che questo protocollo si basa sul supporto hardware del modulo Wi-Fi è necessario che tutti i nodi coinvolti comunichino sullo stesso canale radio per potersi riconoscere reciprocamente.

Per garantire una maggiore robustezza ogni nodo è stato progettato per connettersi inizialmente alla rete Wi-Fi locale. Questa fase di connessione ha il solo scopo di **rilevare dinamicamente il canale Wi-Fi** in uso dal router così da poterlo impostare correttamente anche per la comunicazione ESP-NOW. Tale meccanismo risulta particolarmente utile nei casi in cui il router venga riavviato e assegni un nuovo canale, potenzialmente diverso da quello usato in precedenza, in questo modo si evita la perdita di comunicazione tra i nodi in seguito a riconfigurazioni impreviste della rete.

2.1 ESP-NOW

La scelta di ESP-NOW rispetto all'uso di Wi-Fi standard è stata motivata principalmente da ragioni di **efficienza energetica** e **leggerezza del protocollo**. In particolare, ESP-NOW offre i seguenti vantaggi:

- **Basso consumo energetico**: a differenza del Wi-Fi tradizionale ESP-NOW non richiede la connessione a un Access Point.
- **Comunicazione peer-to-peer diretta**: i dispositivi comunicano direttamente tra loro senza passare da un router.
- **Pacchetti leggeri**: Possono essere trasmessi al massimo 250 byte di dati.
- **Nessuna associazione a una rete Wi-Fi**: i dispositivi possono comunicare anche senza accesso a internet o infrastrutture, rendendo il sistema più resiliente e autonomo.

ESP-NOW è molto versatile e consente comunicazioni sia unidirezionali che bidirezionali in varie configurazioni. L'accoppiamento tra i dispositivi è necessario prima della loro comunicazione. Al termine dell'accoppiamento, la connessione è sicura e peer-to-peer, senza necessità di handshake, quindi dopo aver accoppiato due dispositivi tra loro la connessione è persistente. I MAC dei dispositivi sono i seguenti:

- ESP32 Rossa → Temperatura: E4:65:B8:27:5A:2C
- ESP32 Bianca → Umidità: E4:65:B8:27:62:3C
- ESP32 Gialla → Edge: FC:B4:67:F1:52:C8

- ESP32 Rossa KO → Model: FC:B4:67:EF:F2:B8

Una scelta implementativa importante nell'utilizzare ESP-NOW è stata quella di gestire il protocollo in maniera differente sul nodo contenente i modelli rispetto agli altri nodi di sensing. Nel nodo di inferenza, infatti, la connessione ESP NOW viene resettata ogni volta che viene mandato un messaggio al nodo EDGE, ed è stata una scelta dovuta al fatto che la versione del compilatore utilizzata nel produrre il codice del nodo modello è stata la 2.0.14 a causa di un errore di inclusione da parte della libreria TensorFlowLite_ESP32.h).

Le Figure 2 e 3 mostrano che il nodo Edge riceve correttamente i primi dati dai nodi di sensing e popola i buffer. L'implementazione di buffer "a coda" gestiti in modo da contenere le ultime quattro registrazioni di temperatura e umidità si è dimostrata necessaria per permettere le inferenze su richiesta da parte del nodo Edge.

Poiché l'idea implementativa era quella di definire una rete di campionamento robusta rispetto a eventuali malfunzionamenti del nodo Edge e dei nodi di sensing, la soluzione dei buffer si è rivelata fondamentale per consentire le inferenze su richiesta. I modelli sono stati addestrati su vettori di 8 elementi della forma di input $[t_1, h_1, t_2, h_2, t_3, h_3, t_4, h_4]$, rappresentativi degli ultimi quattro campioni di temperatura e umidità. Le predizioni avvengono tramite regressione del campione successivo: il modello per l'umidità predice il valore h_5, mentre quello per la temperatura predice t_5.

Nel contesto descritto, quando il nodo Edge deve attivare il nodo "Modelli", invia direttamente un vettore contenente gli ultimi quattro campioni di temperatura e umidità in modo da inizializzare i buffer necessari per le inferenze. Durante l'esecuzione il nodo continua ad aggiornare i propri buffer, uno per l'umidità e uno per la temperatura. Questi vettori vengono aggregati attraverso l'intercalazione degli elementi prima di essere forniti in input ai modelli.

Il nodo Edge utilizza un sistema simile per tenere traccia degli ultimi campioni ricevuti dai nodi di temperatura e umidità. Quando un nodo non è raggiungibile al posto del dato mancante viene utilizzato il valore predetto dal nodo "Modelli". Il nodo Edge gestisce inoltre i valori inviati a InfluxDB e utilizzati per aggiornare i buffer dando priorità ai dati reali provenienti dai nodi di sensing rispetto a quelli predetti. In altre parole, se entrambi i nodi sensing (temperatura e umidità) non sono operativi il nodo "Modelli" viene attivato e invia le inferenze al nodo Edge che le trasmette a InfluxDB e aggiorna i buffer. Se invece almeno uno dei due nodi è attivo, l'unico dato predetto che viene usato è quello relativo alla grandezza fisica del nodo non funzionante.

Quando entrambi i nodi tornano operativi, il nodo Edge invia un messaggio al nodo "Modelli" sotto forma di booleano. Il nodo "Modelli", tramite un controllo sulla lunghezza del messaggio ricevuto (differenziando un vettore di 8 float da un singolo booleano), aggiorna un flag interno e si mette in attesa, sospendendo l'esecuzione del codice di inferenza, trasmissione e aggiornamento dei buffer.

```
21:02:13.639 -> Ricevuto da: Nodo Umidità
21:02:13.639 -> Temperatura: -1.00
21:02:13.639 -> Umidità: 39.00
21:02:13.639 -> Timestamp: 16224
21:02:13.639 ->
21:02:13.639 -> humBuffer
21:02:13.639 -> 39.0000
21:02:13.639 -> 0.0000
21:02:13.639 -> 0.0000
21:02:13.639 -> 0.0000
```

Figura 2. Acquisizione umidità

```
21:02:13.116 -> Ricevuto da: Nodo Temperatura
21:02:13.116 -> Temperatura: 31.60
21:02:13.116 -> Umidità: -1.00
21:02:13.116 -> Timestamp: 16202
21:02:13.116 ->
21:02:13.116 -> tempBuffer
21:02:13.116 -> 31.6000
21:02:13.116 -> 0.0000
21:02:13.116 -> 0.0000
21:02:13.116 -> 0.0000
```

Figura 3. Acquisizione temperatura

2.2 InfluxDB

I dati acquisiti dai nodi di sensing vengono trasmessi al server InfluxDB Cloud [3] tramite una connessione Wi-Fi privata. L'invio avviene mediante l'utilizzo di una chiave API, che consente l'accesso gratuito alla piattaforma e permette l'inserimento dei dati all'interno di un apposito *bucket*. Ogni rilevazione è organizzata secondo i campi: `host`, `humidity`, `location`, `room`, `temperature` e `timestamp`, in modo da garantire tracciabilità spaziale e temporale di ogni misura.

Il periodo di raccolta dati si è esteso per circa tre settimane, durante le quali le misurazioni sono state effettuate nella stessa zona della casa, con campionamento di umidità e temperatura ogni cinque secondi, con l'idea di fare inferenza sul valore immediatamente successivo.

Le Figure 4 e 5 mostrano rispettivamente l'andamento temporale e la distribuzione dei dati di umidità, entrambi ottenuti da InfluxDB. Lo stesso vale per la temperatura: le Figure 6 e 7 visualizzano rispettivamente la serie temporale e l'istogramma dei dati, generati durante la fase di pre-processing dei dati con la libreria Matplotlib.

3. Analisi

Una fase importante del progetto ha riguardato la selezione del modello più adatto da convertire in formato TFLite, con l'obiettivo di eseguirlo su ESP32. Le architetture valutate in questa fase sono state: una rete LSTM, una CNN e due modelli MLP.

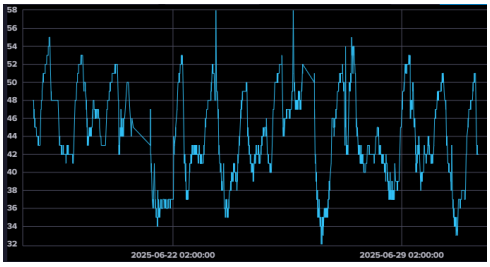

Figura 4. Andamento temporale dell'umidità

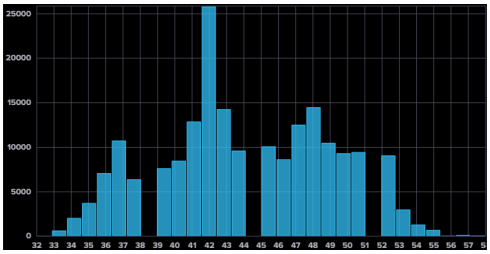

Figura 5. Istogramma dell'umidità

Figura 6. Andamento temporale della temperatura

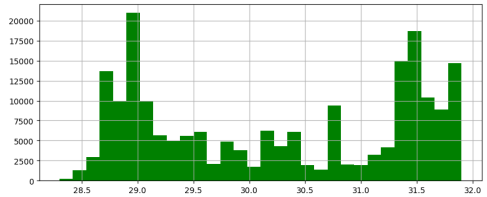

Figura 7. Istogramma della temperatura

CNN Il modello di CNN è una rete sequenziale che prende in input una sequenza di lunghezza window_size con n_features caratteristiche per passo temporale. Utilizza due layer convoluzionali 1D con filtri convoluzionali e kernel di dimensioni 3 e 2 per estrarre i pattern locali della sequenza. L'output convoluzionale viene appiattito e passato a un layer denso con 15 neuroni, seguito da un ultimo layer denso che produce la predizione finale. Infine, il modello è compilato con loss MSE e ottimizzatore Adam, adatto per problemi di regressione su dati sequenziali.

LSTM La LSTM è una rete neurale sequenziale progettata per dati temporali con due caratteristiche per ogni passo temporale, l'input ha forma (time_steps, 2), cioè sequenze di lunghezza time_steps con 2 feature ciascuna. Il layer principale è un **LSTM** con lstm_units neuroni, che cattura le dipendenze temporali della sequenza e restituisce un singolo vettore di output. Questo output viene passato a un layer denso seguito da un ulteriore layer denso con 1 neurone e attivazione lineare che produce la previsione finale. Il modello viene compilato con la funzione di perdita MSE (Mean Squared Error) e l'ottimizzatore Adam.

MLP Sono state provate diverse architetture MLP (Multy-Layer Perceptron), ma la migliore si è dimostrata la MLP_2. Questo modello è una rete sequenziale che prende in input una sequenza di lunghezza time_steps con 2 caratteristiche per passo temporale, l'input viene appiattito in un vettore monodimensionale e poi passa attraverso quattro layer densi con attivazione ReLU: i primi due con numero neuroni mlp_units e i successivi due con la metà di questi neuroni. Infine un layer denso lineare produce la predizione finale. Il modello è compilato con loss MSE e ottimizzatore Adam.

Il processo di sperimentazione è avvenuto mediante fasi di addestramento a 50 epoche secondo una divisione del dataframe di addestramento/valutazione in:

- Set di training: 60
- Set di validazione: 20
- Set di test: 20

4. Risultati

I risultati quantitativi ottenuti durante la fase di validazione sono riportati nella Tabella 1, che confronta le prestazioni in termini di *Loss* e *Mean Absolute Error* (MAE) dei modelli CNN, LSTM e MLP (in due varianti) sia sulla previsione della temperatura che dell'umidità.

Modello	Temperatura		Umidità	
	Loss	MAE	Loss	MAE
CNN	0.000415	0.0125	0.0121	0.0325
LSTM	0.0040	0.0601	0.0434	0.1701
MLP	0.0250	0.0323	0.0219	0.1059
MLP_2	0.0142	0.0783	0.0161	0.0625

Tabella 1. Prestazioni dei modelli su temperatura e umidità

Tra i modelli esplorati le reti neurali convoluzionali (CNN) hanno mostrato le migliori prestazioni sia nella stima della temperatura che dell'umidità come dimostrano i valori minimi di *loss* e *Mean Absolute Error* (MAE) ottenuti sul test set. Le Figure 8 e 9 illustrano rispettivamente le predizioni del modello CNN per la temperatura e per l'umidità. I risultati confermano l'efficacia delle CNN nel catturare pattern temporali anche con finestre di input ridotte.

Il principale obiettivo del progetto è stato rendere il sistema resiliente, in grado di garantire il monitoraggio anche in condizioni di malfunzionamento parziale. Come illustrato nelle Figure 10 e 11, nel momento in cui uno o entrambi i nodi di sensing smettono di inviare dati, l'Edge invia al nodo "Modelli" un vettore contenente le ultime quattro registrazioni valide dei parametri ambientali. Una volta ricevuto il vettore il nodo elabora le predizioni mediante i modelli neurali pre-addestrati

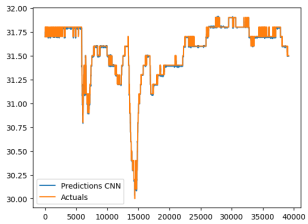

Figura 8. Predizioni della CNN per la temperatura

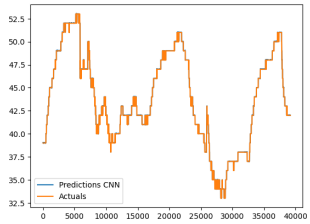

Figura 9. Predizioni della CNN per l'umidità

trasmettendo le stime al nodo Edge con cadenza regolare. Il ciclo di inferenza si interrompe esclusivamente al ricevimento di un messaggio di contenente un booleano, segnale che entrambi i nodi di sensing hanno ripreso a funzionare correttamente.

```
22:05:18.930 -> Connesso al WiFi con IP: 192.168.1.206
22:05:18.930 -> WiFi Channel: 11
22:05:18.930 -> Modelli TensorFlow Lite inizializzati correttamente!
22:05:49.124 -> Ricevuto da: FC:B4:67:F1:52:C8
22:05:49.124 -> Ricevuto vettore di 8 float → flagML = true
22:05:49.124 -> Temp: 30.70 30.70 30.70 30.70
22:05:49.124 -> Hum: 40.00 40.00 40.00 40.00
22:05:52.292 -> Temp buffer: 30.70 30.70 30.70 30.68
22:05:52.292 -> Hum buffer: 40.00 40.00 40.00 39.99
22:05:52.292 -> Messaggio inviato!
22:05:52.324 -> Send Status: Success
```

Figura 10. Avvio nodo CNNs e inzio trasmissione

```
22:07:02.296 -> Temp buffer: 30.54 30.53 30.52 30.51
22:07:02.328 -> Hum buffer: 39.94 39.93 39.93 39.92
22:07:02.328 -> Messaggio inviato!
22:07:02.328 -> Send Status: Success
22:07:06.117 -> Ricevuto da: FC:B4:67:F1:52:C8
22:07:06.117 -> Ricevuto booleano → flagML = false
```

Figura 11. Fine inferenze e trasmissione

Durante la fase di addestramento ciascun modello ha mostrato di avere circa 1.80 KB di parametri allenabili, confermando la leggerezza dell'architettura, particolarmente adatta a dispositivi embedded.

Al termine dell'addestrament, i modelli sono stati convertiti nel formato TensorFlow Lite e successivamente in codice C++ per l'integrazione su microcontrollori. Le dimensioni ottenute sono:

- `model_temp_cnn.tflite`: 5076 byte
- `model_hum_cnn.tflite`: 5076 byte
- `model_temp_cnn.cc`: 31393 byte
- `model_hum_cnn.cc`: 31391 byte

Queste dimensioni estremamente ridotte dimostrano la compatibilità dei modelli con sistemi a risorse limitate, rendendoli idonei per applicazioni edge nel contesto dell'Internet of Things (IoT).

5. Conclusioni

Grazie ai dati raccolti durante le tre settimane di monitoraggio sono stati sviluppati due modelli CNN per la previsione di temperatura e umidità, progettati per funzionare su dispositivi embedded con risorse limitate. Il sistema è stato concepito per garantire resilienza, permettendo il monitoraggio continuo anche in caso di malfunzionamenti dei sensori grazie all'uso di predizioni basate sui modelli.

Le CNN hanno mostrato migliori prestazioni rispetto ad altre architetture testate, offrendo accuratezza e compattezza ideale per l'implementazione su microcontrollori.

In conclusione, il progetto dimostra l'efficacia dell'integrazione di modelli di machine learning leggeri in soluzioni IoT, migliorando l'affidabilità del monitoraggio ambientale in scenari reali.

Riferimenti bibliografici

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

[2] Espressif Systems. ESP-NOW. `https://www.espressif.com/en/solutions/low-power-solutions/esp-now`.

[3] InfluxDB. `https://www.influxdata.com/`.