Agenzia nazionale per le nuove tecnologie,
l'energia e lo sviluppo economico sostenibile

MINISTERO DELLO SVILUPPO ECONOMICO

Sviluppo di sensori intelligenti e moduli di interfacciamento per il recupero dati per applicazioni Smart Home

A. Laudani, F. Riganti Fulginei, G. M. Lozito, M. Botticelli

ROMA TRE
UNIVERSITÀ DEGLI STUDI

Report RdS/PAR2016/004

SVILUPPO DI SENSORI INTELLIGGENTI E MODULI DI INTERFACCIAMENTO PER IL RCUPERO DATI PER APPLICAZIONI SMART HOME

A. Laudani, F. Riganti Fulginei, G. M. Lozito, M. Botticelli (Università di Roma Tre, Dipartimento di Ingegneria)

Settembre 2017

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Annuale di Realizzazione 2016
Area: Efficienza energetica e risparmio di energia negli usi finali elettrici e interazione con altri vettori energetici
Progetto: D.6 Sviluppo di un modello integrato di smart district urbano
Obiettivo: b. Sistemi e servizi smart per edifici
Responsabile del Progetto: Claudia Meloni, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione "Sviluppo di sensori intelligenti e moduli di interfacciamento per il recupero dati per applicazioni Smart Home"

Responsabile scientifico ENEA: Francesco Romanello
Responsabile scientifico Università degli Studi Roma Tre: Antonino Laudani

Indice

Sommario

Il presente documento descrive il lavoro di analisi preliminare e progettazione per lo sviluppo di sensori intelligenti e moduli di interfacciamento per il recupero dati per applicazioni Smart Home. Il documento è strutturato in due parti. La prima descrive lo sviluppo di un sensore virtuale per la misura della CO2 in ambienti chiusi a partire da variabili misurabili. In particolare, il problema della misura indiretta della CO2 viene prima affrontato dal punto di vista analitico, per determinare la fattibilità della misura e le grandezze ad essa correlate, sulla base di una serie di dati sperimentalmente acquisiti. Successivamente, viene proposta una soluzione basata su reti neurali dinamiche, per l'implementazione a scatola nera delle equazioni di stato descriventi l'evoluzione temporale della CO2. Infine, una possibile implementazione hardware su microcontrollore viene proposta. La seconda descrive lo studio di fattibilità di un sistema di sensori acustici e il loro interfacciamento per il recupero dati. Il lavoro parte da una analisi di mercato dei sensori acustici disponibili, al fine di individuare dei sensori in grado di fornire una risposta in frequenza di interesse per il riconoscimento di suoni tipici in ambito abitativo. Successivamente, viene proposta una soluzione completa di sensoristica, elaborazione del segnale, interfacciamento con un database di raccolta dati e possibile alimentazione dell'intero sistema tramite *energy harvesting*. Entrambe le sezioni sono corredate di codici sorgente sviluppati all'interno del lavoro.

1 Introduzione

L'oggetto del contratto ENEA–UNIROMA3 sul tema "Sviluppo di sensori intelligenti e moduli di interfacciamento per il recupero dati per applicazioni Smart Home" è un'attività di ricerca, che si propone da una parte lo sviluppo di sensori virtuali di CO2 per applicazioni Smart Home e dall'altro lo sviluppo di moduli di interfacciamento per il recupero dati dai sensori distribuiti in un ambiente Smart Home. Nell'attività di studio è prevista anche lo studio di fattibilità di sensori acustici. In particolare le attività previste nel contratto si articolano nei seguenti punti:

A) definizione delle variabili misurabili e disponibili per la valutazione della percentuale di CO2 in ambienti confinati e definiti;

B) messa a punto e training di reti neurali per un sensore virtuale di CO2 e sua implementazione hardware nell'ambito di un sistema di smart home;

C) analisi di fattibilità di sensori acustici wireless per applicazioni di smart home;

D) sviluppo di moduli di interfacciamento e gestione di dati di una rete di smart home.

Nei paragrafi successivi verranno discussi le attività effettivamente svolte ed i risultati ottenuti per i punti precedentemente elencati.

2 Descrizione delle attività svolte e risultati

2.1 Definizione delle variabili misurabili e disponibili per la valutazione della percentuale di CO2 in ambienti confinati e definiti

Come già detto in precedenza lo scopo fondamentale di questa ricerca, di seguito descritta, è lo sviluppo di un sensore virtuale di CO2 per ambienti confinati. Per sensore virtuale si intende uno strumento in grado di effettuare una stima di una grandezza sulla base di una misura indiretta, ossia attraverso la misura di altre grandezze che risultano in qualche maniera collegate con la grandezza che si intende misurare. Chiaramente ciò risulta semplice tutte le volte in cui il legame tra la grandezza da misurare virtualmente e le grandezze misurate effettivamente è ben noto perché legato a fenomeni fisici ben definiti e conosciuti. Ad esempio attraverso la cosiddetta legge di Ohm, sappiamo che è possibile misurare virtualmente il valore di una resistenza R attraverso la misura della corrente I che la attraversa e della tensione V ai suoi capi effettuando il semplice rapporto R=V/I. Non essendo noto a priori un legame tra la presenza di CO2 ed altri parametri misurabili, lo scopo di questa attività di ricerca è stata la formulazione di una serie di modelli a scatola nera basati su sistemi di intelligenza artificiale, tramite i quali fosse possibile stimare la percentuale di anidride carbonica nell'aria a partire da letture istantanee di grandezze ambientali facilmente misurabili e/o disponibili. La prima fase dell'attività di ricerca è stata, quindi, proprio l'individuazione di quali grandezze fossero correlate in maniera determinante con l'andamento dell'anidride carbonica all'interno di un ambiente confinato [1-3]. Chiaramente i livelli di CO2 vengono ad essere modificati in un ambiente confinato a causa di fenomeni in parte indipendenti dall'uomo ed in parte dipendenti dalla presenza dell'uomo. Tra le grandezze non dipendenti dall'uomo c'è il ricircolo dell'aria dovuto a porte o infissi aperti, nonché ai sistemi di circolazione artificiale e di ricambio dell'aria. Questi ultimi sono stati esclusi dall'analisi presupponendo che, solitamente, non sono disponibili nelle comuni abitazioni civili e, qualora presenti e funzionanti, dovrebbero garantire il mantenimento di condizioni di salubrità dell'aria (ossia valori accettabili di CO2, per cui è vana la misura) [4-6]. Tra gli elementi antropici, ci sono sicuramente le alterazioni dovuto al respiro (ossia legate al numero di persone presenti nell'ambiente) ed alle attività connesse alla presenza umana che tendano ad alterare le condizioni di umidità e temperatura dell'aria. Ci sarebbe anche da considerare eventualmente la presenza di piante a foglia stretta o larga, che con l'attività di fotosintesi alterano anch'esse la concentrazione di CO2. In questa fase si è trascurato comunque il contributo relativo alla presenza/assenza di piante. La prima fase dell'attività ha riguardato quindi un'approfondita ricerca bibliografica relativa a

questo tipo di modellamento e la verifica delle disponibilità alcune grandezze, in quanto facilmente misurabili. In questa fase sono state selezionate e, quindi, sono state oggetto di nelle campagne di misura, la temperatura, l'umidità relativa, il numero di persone nel locale e lo stato di apertura o di chiusura di porte e finestre. La lettura di queste grandezze può avvenire tramite sensori dedicati (almeno per la temperatura e l'umidità), ampiamente diffusi negli ambienti domestici e che quindi non richiedono modifiche o costi aggiuntivi nei comuni schemi di gestione di una smart home, o con sensori specifici (numero di persone, porte e finestre aperte chiuse) [7-9]. Nel caso specifico queste ultime tre grandezze sono state registrate in un archivio di eventi (entrate-uscita persona, apertura-chiusura porte e finestre) che poi è stato aggiunto ai dati raccolti attraverso data-log dai sensori di temperatura e umidità. Chiaramente i modelli matematici da implementare devono permettere di stimare con una certa accuratezza il livello di CO2 a partire da queste grandezze, ed essendo basati su black box richiedono, nella loro messa a punto, il training e la validazione tramite confronto con delle misure dirette di CO2. Queste sono state ottenute tramite un sensore dedicato presente nel locale utilizzato come locale di test.

2.1.1 Analisi dei dati disponibili

Diverse acquisizioni ambientali sono state fatte nei locali dell'ENEA a Casaccia dall'Ing. Romanello ed in particolare all'interno ad un ufficio di circa 15 metri quadri ossia circa 40/45 metri cubi. Tale dati sono stati forniti per la successiva analisi al gruppo di ricerca dell'Università degli Studi di Roma Tre coordinato dal Prof. Antonino Laudani. I dati consistono in una serie di acquisizioni sincrone, spaziate ad intervalli di uno o di cinque minuti, delle seguenti grandezze:

- **HUM:** Umidità relativa
- **TMP:** Temperatura media del locale
- **PP:** Numero di persone presenti nel locale
- **DOR:** Stato di apertura e chiusura porta
- **WIN:** Stato di apertura e chiusura finestra
- **CO2:** Valore di CO2 presente nel locale

I dati acquisiti vengono organizzati in sequenze giornaliere della durata di 24 ore. Ad ogni giornata corrispondono quindi 288 campioni, nel caso di campionamento a cinque minuti poi usato, delle sei grandezze elencate.
Di seguito nelle figure 2.1 – 2.6 riportiamo dei grafici relativi alle varie grandezze misurate per un arco temporale complessivo di 6 giornate.

2.1.1.1 HUM - Umidità relativa

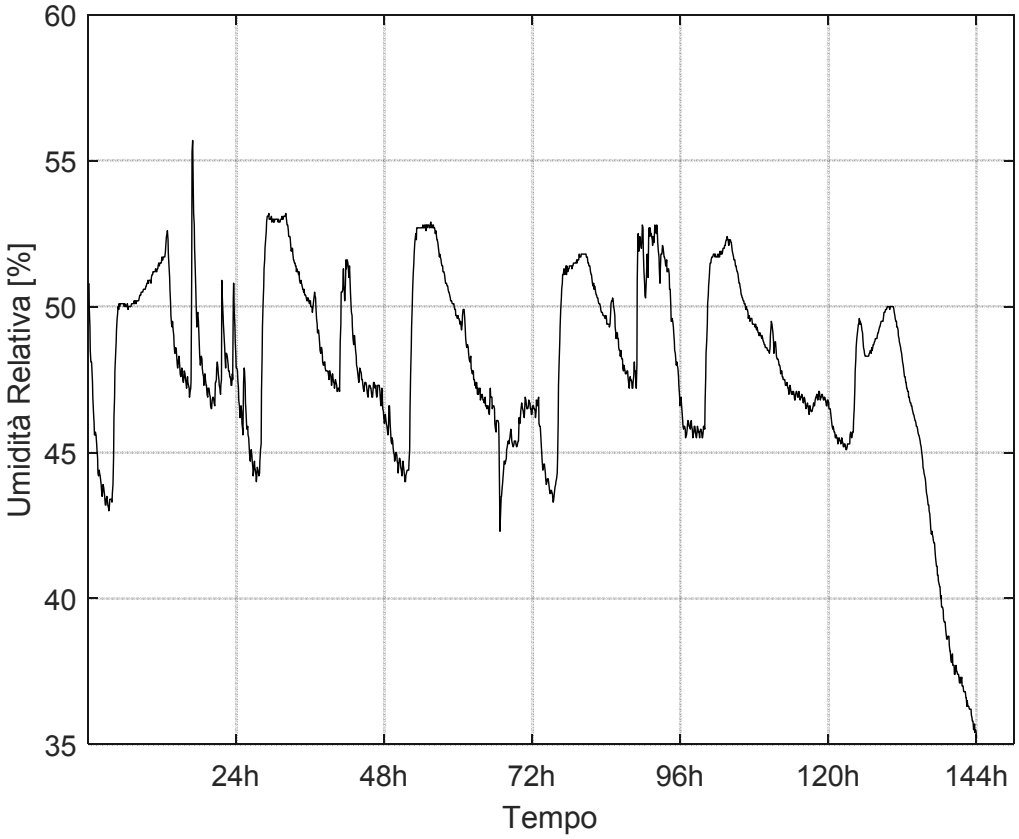

Fig. 2.1 – Andamento temporale dell'umidità relativa misurata nell'ambiente di test considerato

Il valore dell'umidità relativa nei giorni 2, 3, 4 e 5 è debolmente ripetitivo. Nel sesto giorno è presente un comportamento anomalo, che non può essere però spiegato dai dati a nostra disposizione.

Statistica	Valore
Massimo	55.700
Minimo	35.200
Media	47.984
Varianza	11.639

2.1.1.2 TMP – Temperatura

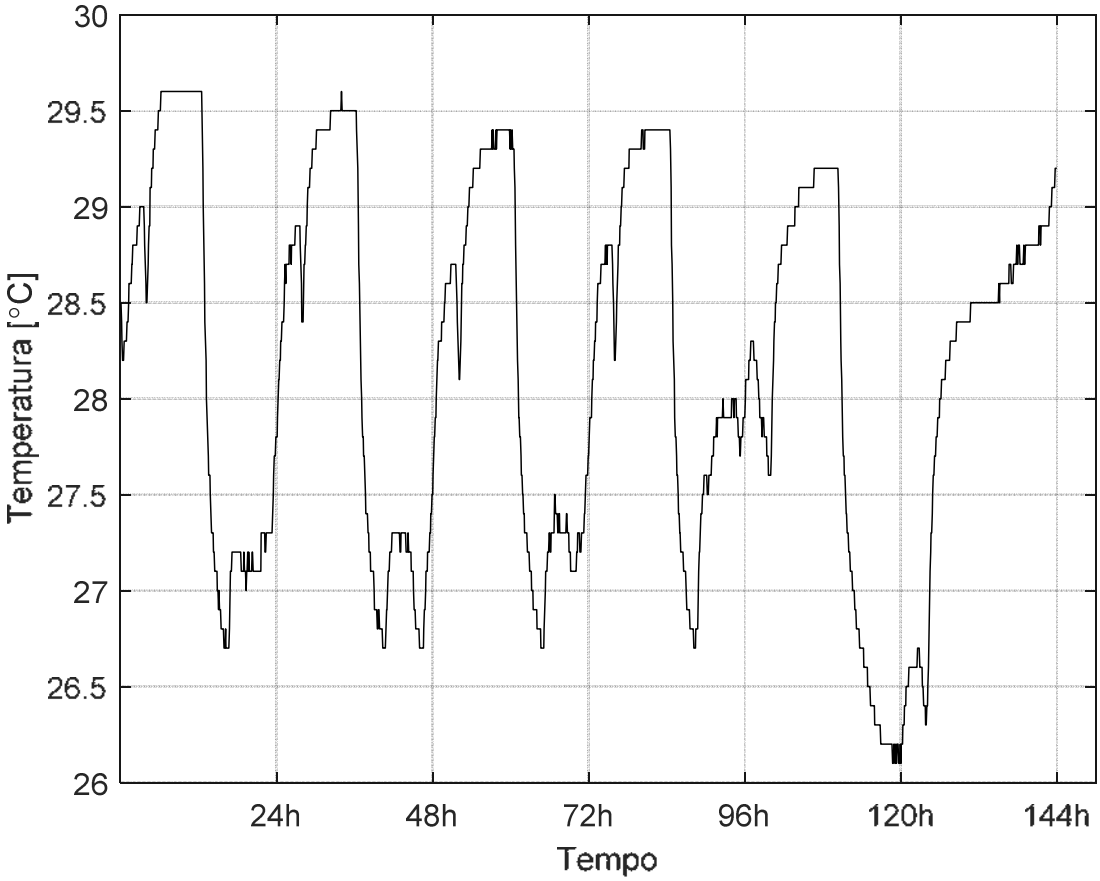

Fig. 2.2 – Andamento temporale della temperatura misurata nell'ambiente di test considerato

La temperatura risulta stabile ed abbastanza ripetitiva nei primi quattro giorni. C'è una lieve anomalia nel quinto giorno e una forte anomalia nel sesto giorno. Dalla rappresentazione grafica risulta evidente la forte quantizzazione della grandezza in esame a causa della bassa varianza.

Statistica	Valore
Massimo	29.600
Minimo	26.100
Media	28.168
Varianza	0.997

2.1.1.3 PP - Persone nel locale

Il numero di persone nel locale è lievemente periodico nelle giornate 1, 2 e 4. La assenza di persone nelle giornate 5 e 6 può spiegare parzialmente le anomalie presenti nella temperatura e nell'umidità nelle stesse giornate. La grandezza in questo caso è completamente quantizzata per cui non è significativo svilupparne una statistica vera e propria in termini di valore medio/varianza.

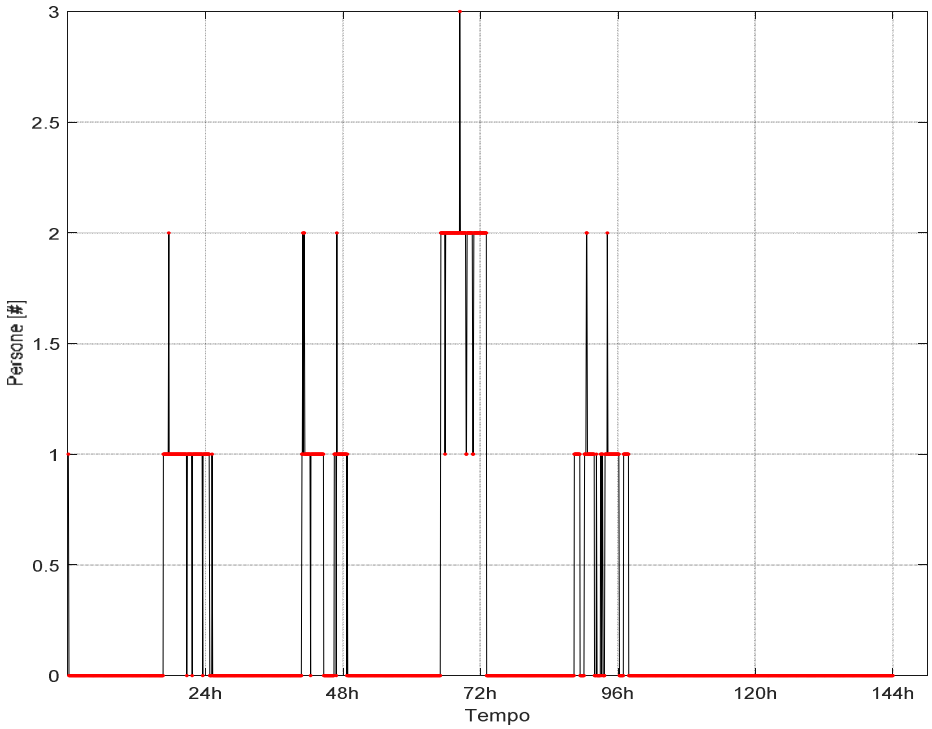

Fig. 2.3 – Andamento temporale del numero di persone presenti nell'ambiente di test considerato

2.1.1.4 DOR - Apertura e Chiusura Porta

Lo stato di apertura e chiusura porta è privo di periodicità notevoli. La codifica numerica scelta per lo stato di apertura e chiusura è di +1 per la porta aperta e 0 per la porta chiusa. Da un confronto con il dato relativo al numero di persone si evince che il locale, in assenza di personale, è mantenuto con la porta chiusa. Inoltre gli intervalli di tempo in cui risulta aperta la porta coincidono quasi sempre con l'entrata e l'uscita delle persone ed eventualmente in occasione di tali eventi la porta rimane aperta per tempi un po' più lunghi del solito. Non si presentano però casi in cui l'apertura della porta duri per tempi maggiori di qualche decina di minuti.

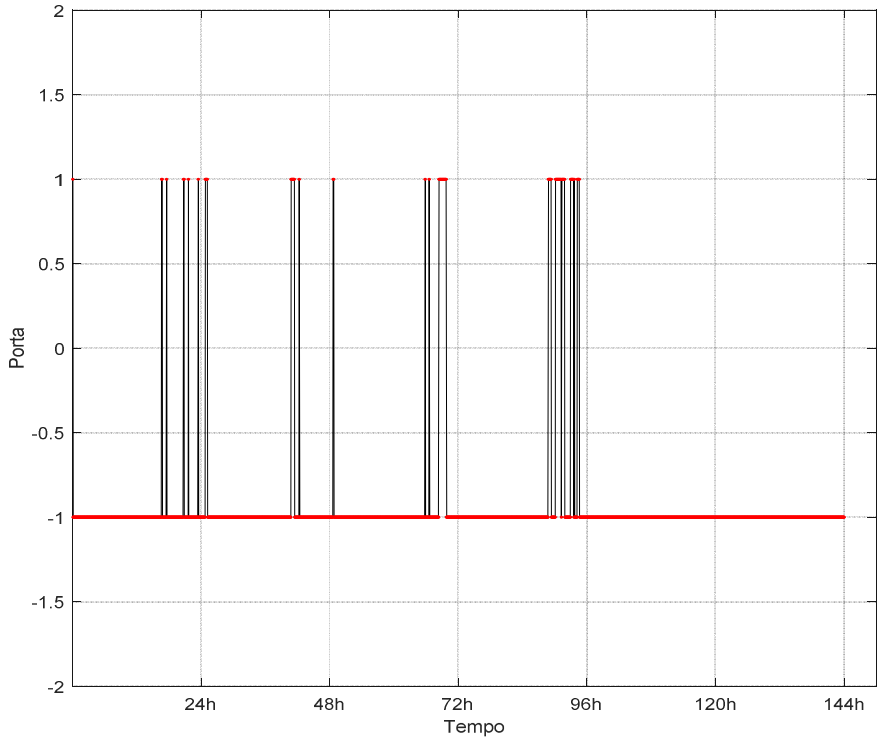

Fig. 2.4 – Andamento temporale dello stato della porta (0 = chiusa, 1 = aperta) nell'ambiente di test considerato

2.1.1.5 WIN - Apertura e Chiusura Finestra

Analogamente allo stato della porta, anche per la finestra non si notano periodicità notevoli. La codifica è questa volta di tipo +1 per finestra aperta e -1 per finestra chiusa. La apertura della finestra è un evento molto raro nel set di dati fornito. Anche per questo caso, come in precedenza per il numero di persone nel locale e per l'apertura e chiusura della porta, non è significativo sviluppare una statistica in termini di valore medio e di varianza.

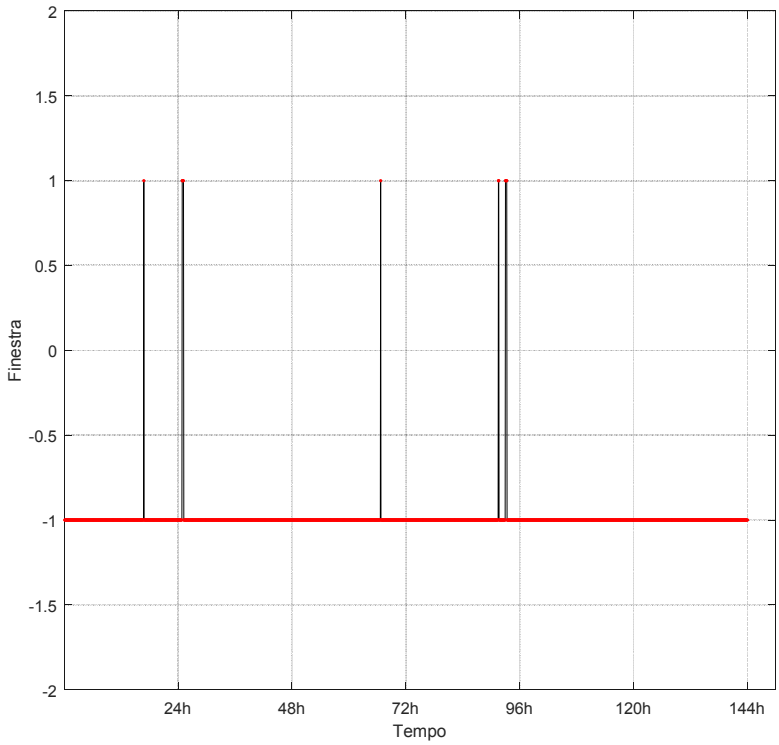

Fig. 2.5 – Andamento temporale dello stato della finestra (-1 = chiusa, 1 = aperta) nell'ambiente di test considerato

2.1.1.6 CO2 - Valore di CO2 presente nel locale

Il valore di CO2 presente nel locale presenta una serie di picchi che sono altamente correlabili, temporalmente, al numero di persone presenti all'interno del locale. Si può notare un valore minimo al quale si attesta la CO2 in condizioni di locale privo di persone. Tenendo conto che il locale è privo di persone durante la notte e che in queste condizioni la CO2 decresce in maniera quasi esponenziale per attestarsi ai valori minimi intorno 400 ppm, possiamo immaginare che vi sia in condizioni naturali, per questo sistema/ambiente, una dinamica di diminuzione dell'anidride carbonica che segua un tale andamento decrescente.

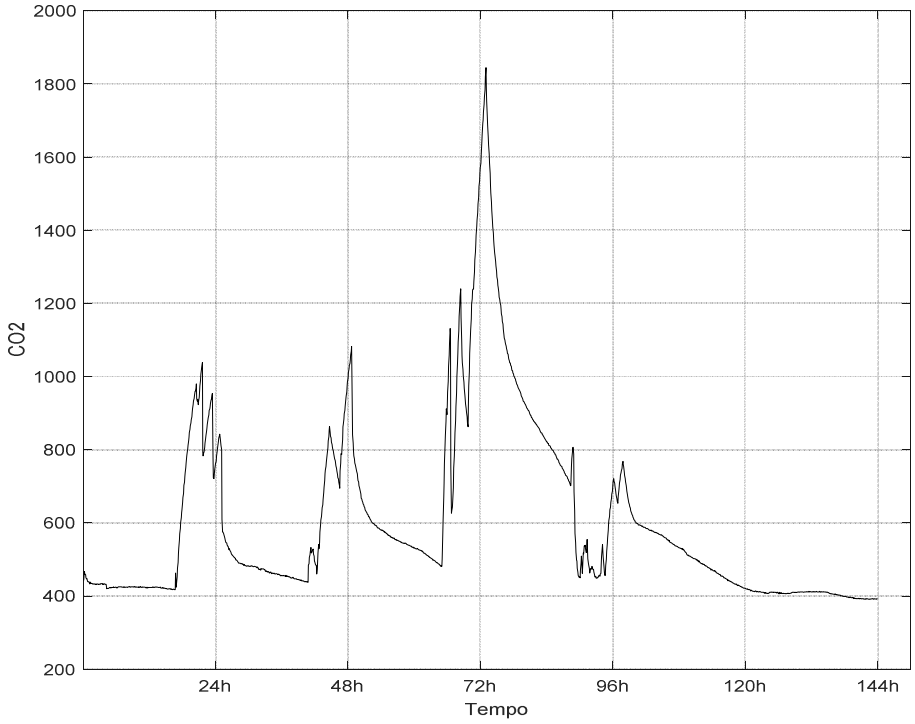

Fig. 2.6 – Andamento temporale della CO2 misurata nell'ambiente di test considerato

Statistica	Valore
Massimo	1845.00
Minimo	391.00
Media	606.182
Varianza	63693.256

Tutti i segnali considerati si sono rilevati sufficientemente correlati con l'andamento della CO2 e perciò sono stati utilizzati nella messa a punto del modello neurale descritto nel paragrafo successivo.

2.2 *Messa a punto e training di reti neurali per un sensore virtuale di CO2 e sua implementazione hardware nell'ambito di un sistema di Smart Home*

Per effettuare la stima della CO2 è necessario valutare se esista un legame funzionale tra le grandezze misurate direttamente (TMP, HUM, PP, DOR, WIN) e la grandezza da stimare. Il legame funzionale che si vorrebbe ottenere è del tipo:

$$CO2 = f(TMP, HUM, PP, DOR, WIN)$$

Ciò significa che per ogni punto del dominio a cinque dimensioni {TMP, HUM, PP, DOR, WIN} deve corrispondere un singolo punto del codominio CO2. Da una analisi preliminare è emerso che questo legame funzionale non esiste, essendo presenti situazioni nelle quali la stessa quintupla {TMP, HUM, PP, DOR, WIN} è legata a diversi valori di CO2.

Analizzando i dati si rileva che 837 (su quasi 1800) campionamenti presentano quintuple simili {TMP, HUM, PP, DOR, WIN} con diversi valori di {CO2}, vedi figura 2.7

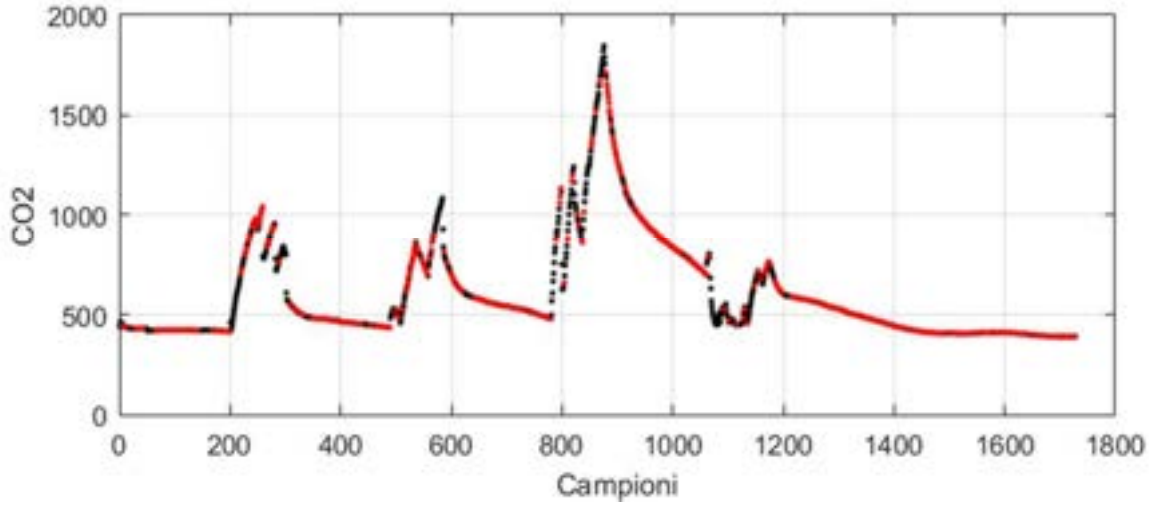

Fig. 2.7 – Andamento temporale della CO2 e campioni (in rosso) che presentano valori analoghi delle grandezze di input considerate

È evidente quindi che la CO2 non può essere espressa mediante un semplice modello statico, ma è necessario considerare alcune grandezze dal punto di vista dinamico. Due possibili soluzioni sono state indagate.

La prima consiste nel considerare, come ingressi addizionali del sistema, le repliche (ritardate) degli ingressi passati. In questo modo è possibile considerare una relazione funzionale non solo con la grandezza istantanea, ma con la sua derivata e il suo valore integrale.

La seconda consiste nel modellare il sistema tramite un'equazione di stato, andando quindi a considerare l'evoluzione della CO2 nel tempo a partire da un valore noto. Questa seconda soluzione richiede ovviamente un valore di partenza noto, che potrebbe essere proprio il minimo di CO2 cui si attesta il locale durante le ore notturne.

Dopo diversi tentativi di implementazione di modelli black box di intelligenza artificiale con ambedue le soluzioni, questa seconda soluzione si è rilevata più efficace ed è quindi quella che è stata scelta per il successivo approfondimento e l'implementazione hardware.

In particolare la black box è stata realizzata tramite una rete neurale, ossia una delle implementazioni più diffuse e generali tra i sistemi di softcomputing ed intelligenza artificiale. La rete neurale utilizzata ricade nelle cosiddette reti ricorrenti ed in particolare nella famiglia delle NARX (Nonlinear autoregressive with external input). Queste reti sono ampiamente usate per descrivere modelli dinamici complessi a piacere in quanto è possibile dimensionarne opportunamente lo stato ed il numero di neuroni in ragione della complessità del sistema.

Nel caso specifico attraverso le procedure di messa punto di reti neurali appositamente sviluppate dal gruppo di ricerca dell'Università Roma Tre [10-11] è stato trovato un ottimo compromesso tra la complessità della rete (importante per la sua successiva implementazione su microcontrollore [12-16]) ed l'accuratezza della previsione.

In particolare si è trovato attraverso una procedura intensa di trial and error che una rete con 2 ritardi nell'uscita e 5 neuroni nello strato nascosto non lineare permetteva di ottenere buone prestazioni sia con il training sia con il validation set (ossia su campioni non conosciuti durante il training). Per le varie fasi della messa a punto di questa rete è stato utilizzato Matlab e gli algoritmi di training proprietari basati su metodi di minimizzazione di problemi ai minimi quadrati, come il Levemberg-Marquadt
Nella figura 2.8 è mostrato la rappresentazione data da Matlab a questo tipo di rete. In particolare è possibile distinguere gli ingressi x(t) che sono le 5 variabili umidità, temperatura, numero di persone, stato della porta e stato della finestra e l'uscita retroazionata (con due ritardi, ossia due campioni ritardati vengono riproposti all'ingresso come input)

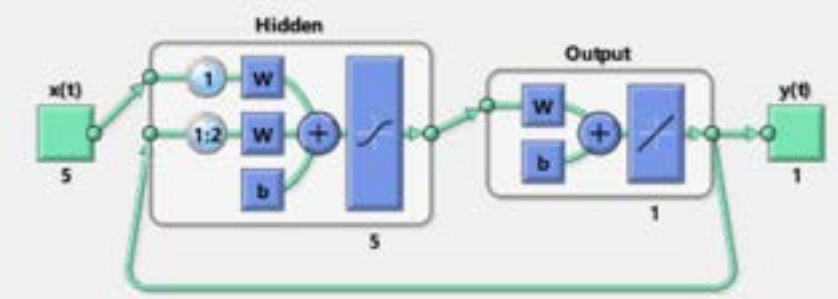

Fig. 2.8 – Schematizzazione Matlab della rete NARX considerata.

Lo strato nascosto (hidden), come già detto, è costituito da 5 neuroni e lo stato d'uscita da 1 solo neurone lineare. Complessivamente abbiamo 5x5 pesi sinaptici per gli ingressi, 2x5 pesi sinaptici per l'uscita retroazionata, 5 bias per lo strato nascosto e 5x1 pesi per lo strato d'uscita ed 1 solo bias per questo stato: quindi abbiamo in totale 25+10+5+5+1=46 grandezze che ci caratterizzano in maniera totale la black box neurale così costruita. Le funzioni di attivazione scelta è la tansig.
L'errore relativo ottenuto con il sistema neurale così progettati è mediamente basso ed, in particolare, nella figura 2.9 è mostrato l'andamento dell'istogramma dell'errore, ossia il numero di campioni per cui l'errore relativo vale il valore riportato sull'asse delle ascisse. Come è evidente in un gran numero di casi l'errore relativo si mantiene ben al di sotto dell'1 % e nel 90% dei casi al di sotto del 2%. Volendo calcolare i parametri statistici di tale errore è possibile vedere che in valore assoluto vale mediamente l'1% e che la deviazione standard è anch'essa dell'1%.

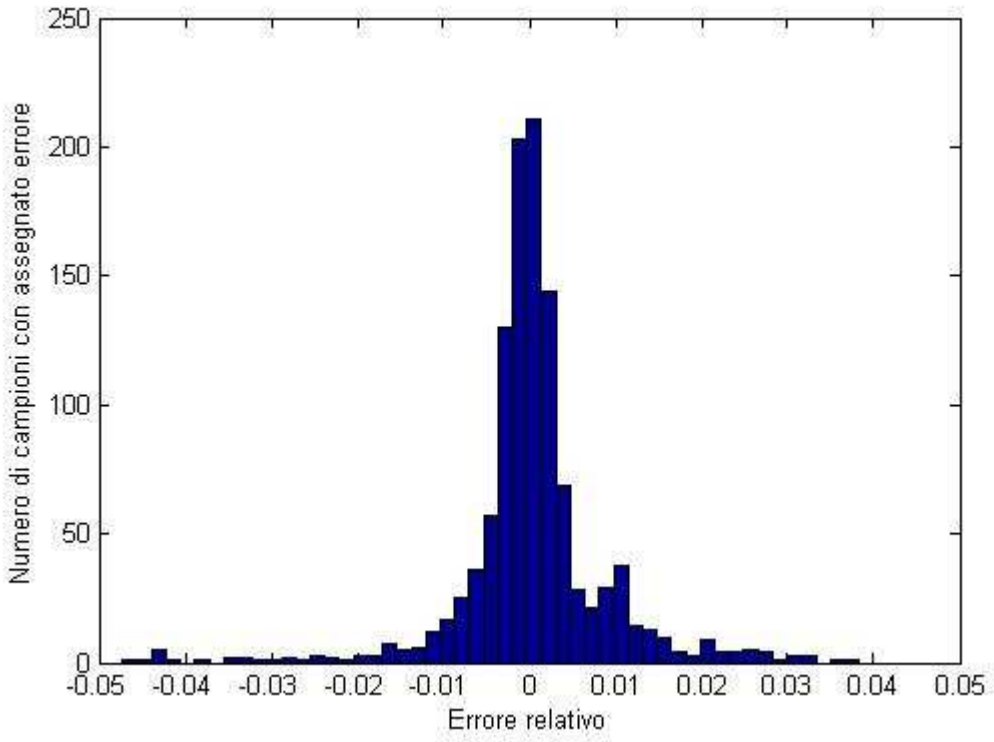

Fig. 2.9 – Istogramma dell'errore

Per validare il sistema abbiamo concatenato dati provenienti da giorni diversi cercando di verificare una certa continuità nella concatenazione. Questa chiaramente non risulta mai perfetta e ciò ha introdotto errori di predizione. Nonostante ciò il sistema neurale così trovato, che ricordiamo essere un sistema a memoria e quindi che teoricamente non risponderà agli stessi ingressi alla stessa maniera dipendendo dallo stato, riesce a correggere sufficientemente bene queste discontinuità come mostrato in fig. 2.10

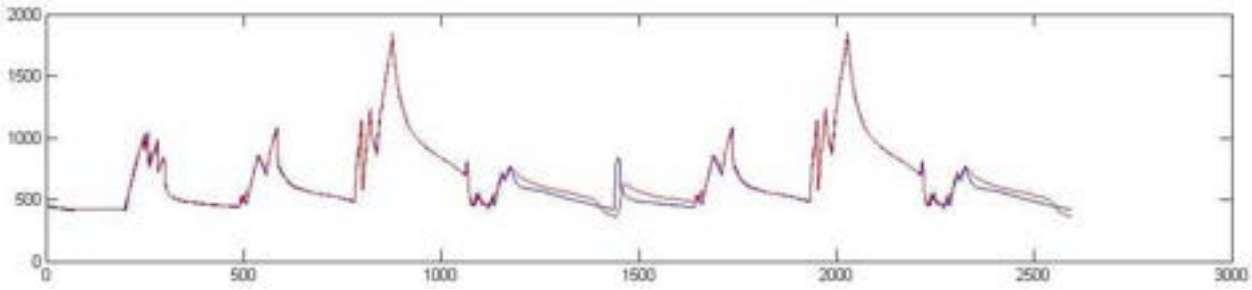

Fig. 2.10 – Andamento della CO2 nella fase di validazione della rete neurale dopo il concatenamento dei campioni dovuti a giornate diverse

Questa correzione può essere ancora attribuita al fatto che determinate condizioni si verificano in questo sistema più o meno alla stessa maniera durante la notte, quando il livello di CO2 si abbassa fino ad arrivare a 400 ppm

2.2.1 Implementazione su MATLAB

Di seguito è presentato il codice matlab utilizzato per il training della rete neurale.

```
clear; clc; close all;

load DataPerDay.mat

day_in{1} = day1_in;
day_in{2} = day2_in;
day_in{3} = day3_in;
day_in{4} = day4_in;
day_in{5} = day5_in;
day_in{6} = day6_in;

day_out{1} = day1_out;
day_out{2} = day2_out;
day_out{3} = day3_out;
day_out{4} = day4_out;
day_out{5} = day5_out;
day_out{6} = day6_out;


% Dati in forma di vettori riga:
% UMIDITA
% TEMPERATURA
% PERSONE
% PORTA (-1 chiusa, +1 aperta)
% FINESTRA (-1 chiusa, +1 aperta)

% SCELTA DATI DI INGRESSO RETE
% 1 UMIDITA
% 2 TEMPERATURA
% 3 PERSONE
% 4 PORTA (-1 chiusa, +1 aperta)
% 5 FINESTRA (-1 chiusa, +1 aperta)

DATA = [1:5];

% DIVISIONE GIORNI IN TRAINING E VALIDAZIONE

TRAIN_DAYS = [1,2,3,4];
VAL_DAYS = [1,2,3,4,5,2,3,4,5];

net_in_mat = [];
net_out_mat = [];

net_in_v_mat = [];
net_out_v_mat = [];

for ct_t = TRAIN_DAYS
    net_in_mat = [net_in_mat,  day_in{ct_t}(DATA,:)];
    net_out_mat = [net_out_mat,  day_out{ct_t}];
end

for ct_t = VAL_DAYS

    net_in_v_mat = [net_in_v_mat,  day_in{ct_t}(DATA,:)];
    net_out_v_mat = [net_out_v_mat,  day_out{ct_t}];
end

% VALIDAZIONE A CAMPIONI ALTERNI (sovrascrive VAL_DAYS!)
% ######################################################
```

```
%
% dispari = 1:2:length(net_in_mat);
% pari = dispari + 1;
%
% net_in_v_mat = net_in_mat(:,pari);
% net_out_v_mat = net_out_mat(:,pari);
%
% net_in_mat = net_in_mat(:,dispari);
% net_out_mat = net_out_mat(:,dispari);

% ###########################################################


net_in =    num2cell(net_in_mat ,1);
net_out =   num2cell(net_out_mat);

net_in_v =    num2cell(net_in_v_mat,1);
net_out_v =   num2cell(net_out_v_mat);


% ###################################################
% ########## SCELTA TIPO DI RETE DA USARE ##########
% ###################################################


net = narxnet(1,1:2,5);
net = closeloop(net);
net.divideFcn = '';
net.trainParam.epochs = 2000;
net = train(net,net_in,net_out);


% NORMAL OUTPUT
subplot(2,1,1)
plot(cell2mat(net_out)); hold on; plot(cell2mat(net(net_in)),'r')

subplot(2,1,2)
plot(cell2mat(net_out_v)); hold on; plot(cell2mat(net(net_in_v)),'r')
```

2.2.2 Implementazione su scheda a microcontrollore Tiva TM4C1294

Dopo lo sviluppo dei modelli matematici, la fase successiva del lavoro ha coinvolto lo studio di piattaforme di sviluppo adatte alla implementazione hardware del suddetto modello al fine di creare uno strumento di misura vero e proprio. A questo scopo, sono state studiate piattaforme a microcontrollore ad alte prestazioni in grado di interfacciarsi con la sensoristica già presente nell'ambiente sotto esame, e in grado di essere facilmente interrogate tramite interfaccia seriale e/o ethernet. Per lo sviluppo del codice in C, e per il test del prototipo, è stata usata una scheda di sviluppo a microcontrollore della Texas Instruments Tiva C Series TM4C. Questa scheda di sviluppo è dotata di un microprocessore di tipo ARM Cortex M4-F ad alte prestazioni. La scheda è un ottimo compromesso tra prestazioni, versatilità, costi e dimensioni. Il microcontrollore TM4C1294NCPTDI è al centro della scheda di sviluppo. Si tratta di un microcontrollore a 32 bit con massima frequenza di clock pari a 120MHz. La memoria è particolarmente estesa, essendo presenti 1Mb di Flash, una SRAM di 256kB e 6kB di EEPROM direttamente sul chip. È inoltre presente una ROM interna contenete le procedure in Assembler per il controllo delle periferiche, tramite la quale è possibile ridurre notevolmente le dimensioni del codice in fase di programmazione. Il microcontrollore è dotato di una interfaccia completa 10/100 Ethernet MAC + PHY, interfaccia USB H/D/O e controllori dedicati per le principali interfacce seriali per il collegamento con ulteriori sensori integrati. La conversione analogico-digitale è possibile tramite due ADC a 12 bit (2MS/s) collegati tramite sequenziatori ad alte prestazioni per il monitoraggio periodico della tensione su 24 pin. La scheda è dotata di numerose sorgenti di clock, per fornire una soluzione idonea a qualsiasi necessità dell'utente. Il microcontrollore dispone internamente di un PIOSC (Precision Internal Oscillator) da 16 MHz, tramite il quale è possibile avviare il MCU senza necessità di alcun componente esterno. Una volta avviato il microcontrollore, è possibile configurarlo per poter essere alimentato da un oscillatore esterno, il MOSC (Main OSCillator) che può sfruttare un clock o un cristallo esterno. Questo è necessario per l'uso di frequenze di clock più elevate (è presente un PLL demoltiplicato interno) e soprattutto per interfacciare in maniera sincrona il MCU ad altre periferiche. È presente anche un ingresso da 30 kHz per le modalità di deep-sleep e risparmio energetico. Infine è presente un modulo clock da 32,768 Hz che viene demoltiplicato al fine di ottenere un *real time clock* con impulsi ogni secondo, utile per funzioni di calendario e scheduling.

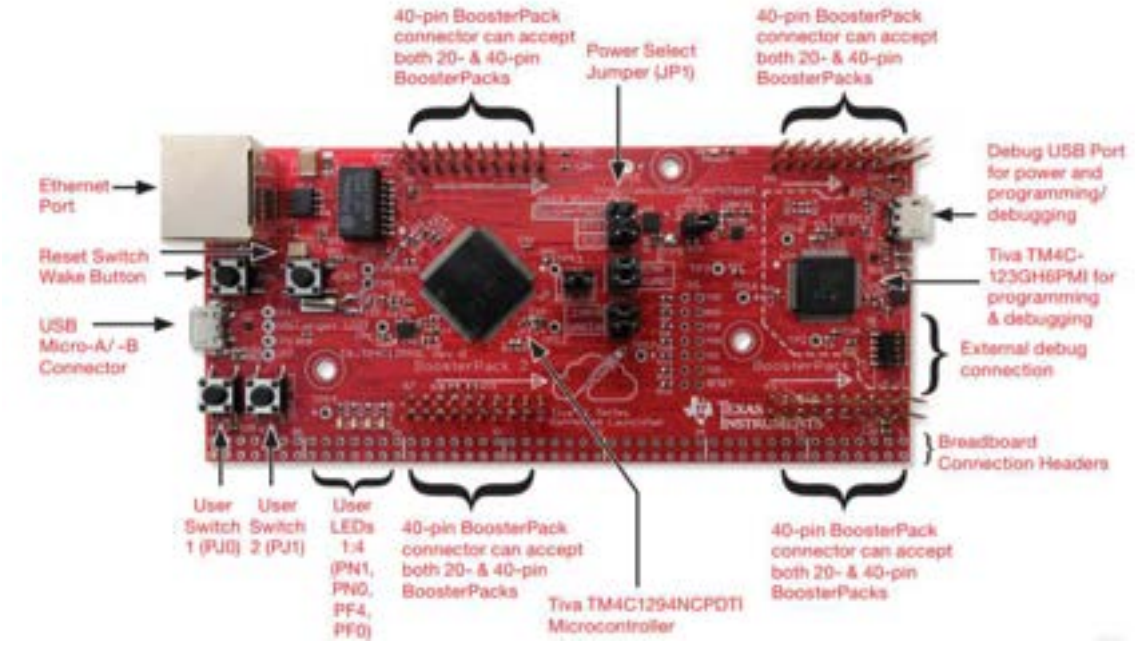

Fig. 2.11 – scheda a microcontrollore della Texas Instruments Tiva TM4C1294

Il microcontrollore può essere interfacciato con sensori di vario genere tramite diverse modalità. Per sistemi di domotica può essere utile l'uso della interfaccia Ethernet (una versione leggermente più avanzata della

stessa scheda integra la connettività WiFi). Per la lettura diretta di sensori può essere usata l'interfaccia ADC (lettura analogica) o le numerose interfacce seriali (lettura digitale).

2.2.2.1 ADC

Il microcontrollore presenta due moduli ADC (ADC0 e ADC1), ognuno dei quali opera con una risoluzione a 12 bit ed è indipendente dall'altro. Entrambi si interfacciano ai PIN del dispositivo tramite 12 canali indipendenti ed è possibile effettuare la digitalizzazione di più segnali contemporaneamente da diversi canali. Questo è reso possibile dai *sequencer*, quattro per ogni ADC, che determinano l'ordine di acquisizione dai diversi canali. Il limite di acquisizione dei singoli ADC è di 1MS/s, tuttavia grazie al controllo di fase hardware implementato, è possibile programmare gli ADC per operare in modalità complementare ed ottenere una frequenza di campionamento doppia (2MS/s). Il convertitore è anche dotato di un *hardware averager*, tramite il quale è possibile effettuare una media dei campioni acquisiti, rendendo più stabile una acquisizione rumorosa al costo di ridurre la frequenza di campionamento. L'implementazione hardware permette di azzerare i tempi necessari per il calcolo della media. Il convertitore è di tipo SAR (*Successive Approximation Register*) ed è la scelta più comune di convertitore ADC presente in ambito integrato, permettendo di ottenere una alta precisione (nel caso del microcontrollore della Texas, 2 bit maggiore rispetto alla media dei microcontrollori della stessa fascia) a velocità accettabili. Il sistema di Sample & Hold è interamente gestito dall'ADC e sono presenti, nel manuale, delle tabelle di configurazione per la massima frequenza di campionamento in funzione della resistenza di uscita della sorgente di tensione da convertire.

2.2.2.2 GPIO

Per l'interfacciamento digitale di sensori il microcontrollore è dotato di una interfaccia GPIO (General Purpose Input Output). Ogni porta GPIO è una periferica a se stante, alimentata (dal clock) autonomamente. Sono presenti 15 diverse porte (A-Q), per un totale di 90 pin, distribuiti variabilmente nelle diverse porte. I PIN possono essere configurati come ingresso digitale, ingresso analogico o uscita digitale. La massima tensione di ingresso è 3.3V. Internamente, la periferica è connessa sull'Advanced High Performance Bus, ciò significa che le scritture e le letture possono essere effettuate ogni ciclo di clock. Inoltre, la scrittura e la lettura sulle porte avviene tramite «bit-banding», una tecnica tramite la quale è possibile effettuare la modifica dello stato dei singoli PIN di una porta come operazione atomica (normalmente è necessaria una operazione di lettura, una di modifica e una successiva di scrittura). La periferica GPIO svolge il ruolo di interfaccia verso l'esterno per tutte le periferiche interne al microcontrollore, ed è dotata di un sistema di multiplazione tramite il quale è possibile scegliere per lo stesso PIN diverse funzioni. Le caratteristiche analogiche dei pin possono essere configurate. (e.g. corrente di drive/sink da 2 a 18mA). Direttamente alla GPIO sono connessi i controller per le interfacce dedicate ai principali sistemi di comunicazione seriale: USB, UART, I2C, SPI, CAN e 1-Wire.

2.2.2.3 Power Saving Mode

Il sistema è particolarmente efficiente dal punto di vista energetico, ed è in grado di entrare in modalità di risparmio energetico a bassissimi consumi (nell'ordine dei µA). I sistemi di risveglio da queste modalità possono essere di tipo temporale (e.g. il raggiungimento di un valore prestabilito di un timer) o tramite segnali esterni ricevuti su particolari pin del dispositivo adibiti al risveglio. La flessibilità del sistema di alimentazione è particolarmente importante per dispositivi di monitoraggio alimentati a batteria.

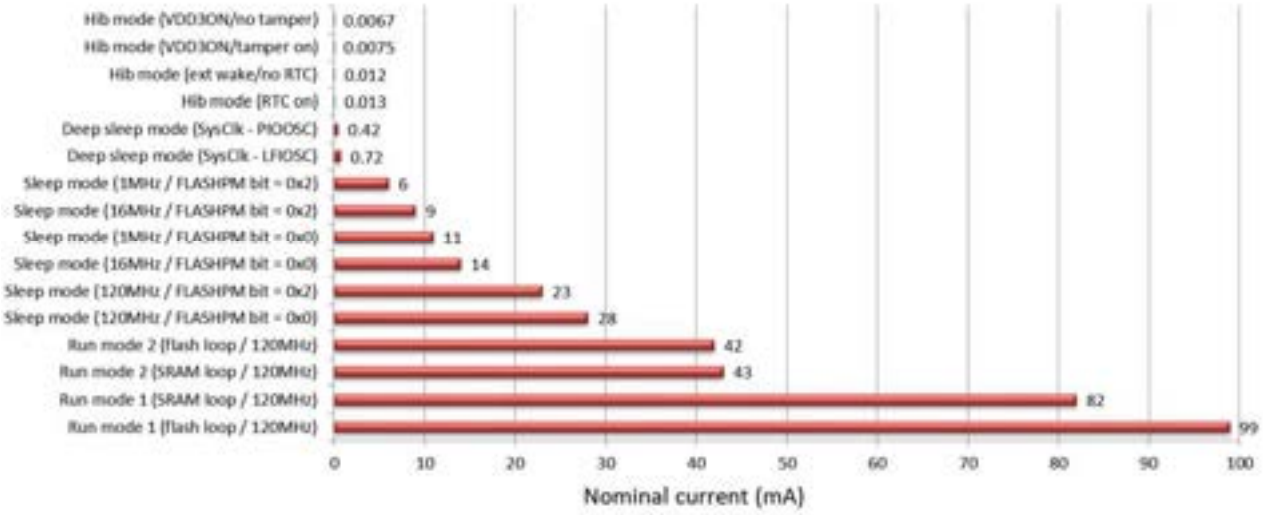

Fig. 2.12 – scheda a microcontrollore della Texas Instruments Tiva TM4C1294

2.2.2.4 Floating Point Unit

Nel caso in esame il microcontrollore dovrà effettuare una elaborazione locale dei dati mediante modelli matematici. In generale le capacità computazionali di un microcontrollore *general purpose* come quello usato per questo progetto non sono elevate (o quantomeno, non elevate quanto quelle di un DSP). Tuttavia il microcontrollore montato su questa scheda di sviluppo è dotato di una unità di calcolo Floating Point (ovvero in virgola mobile) dedicato in grado di rendere più veloci le operazioni di calcolo. Le operazioni supportate sono di somma con segno, moltiplicazione, divisione, moltiplicazione ed accumulo e radice quadrata. Nel caso la velocità ottenuta con questo genere di precisione non fosse sufficiente, sono disponibili delle librerie gratuite della Texas Instruments (*IQ math* Library) tramite le quali è possibile effettuare la conversione di un algoritmo floating-point in versione fixed-point, in grado di fornire velocità estremamente più elevate con una accettabile riduzione della precisione di calcolo.

2.2.2.5 Timer

Per la temporizzazione, il microcontrollore è dotato di registri incrementali (timer) collegabili al clock di sistema. Il microcontrollore è dotato di 8 timer(s) a 32/16 bit. I timer possono essere usati sia in modalità 32 bit (concatenati) o 16 bit (splittati). Quando non splittati, le due metà sono sempre identificate con le lettere A e B. E' possibile configurare la periferica per ottenere diverse funzioni: Contatore one-shot / periodico, Input-Edge-Count, Input-Edge-Time, Modulo PWM, Real Time Clock.
I timer risultano particolarmente utili nella sincronizzazione del campionamento ad opera dei moduli ADC. È possibile infatti impostare una connessione hardware che avvii la conversione dell'ADC al riempimento di un timer.

2.2.2.6 NVIC (Nested Vector Interrupt Controller)

Il controller per gli interrupt è necessario per la gestione delle eccezioni all'interno del microcontrollore. Il microprocessore, in assenza di eccezioni, esegue le operazioni in maniera sequenziale. Nel caso sia presente una eccezione, l'esecuzione sequenziale del codice viene interrotta per entrare in una nuova sezione di codice atta alla gestione delle eccezione. Gli interrupt sono delle eccezioni programmate che possono provenire da un segnale esterno connesso su un pin, dall'esaurimento di un timer, dalla conclusione di una conversione ADC e simili. Il sistema NVIC permette di assegnare a ciascuna eccezione un grado di priorità, in modo da istruire il microcontrollore alla esecuzione delle routine di interrupt più urgenti e demandare ad un momento successivo la gestione delle eccezioni meno urgenti. La programmazione di questa periferica è versatile ed è possibile definire le priorità e le routine sia in fase di programmazione (compile-time) sia in fase di esecuzione (run-time).

2.2.2.7 Code Composer Studio

La programmazione del dispositivo può essere effettuata mediante diversi IDE e toolchain disponibili sia gratuitamente sia a pagamento. In particolare il software Code Composer Studio, fornito dalla Texas Instruments, fornisce un ambiente di programmazione (derivato da Eclipse) completamente funzionale per la programmazione, il debug e il test di codici di qualsiasi dimensione. Va sottolineato che il software è gratuito solo per l'uso su schede di sviluppo. Non è infatti possibile usarlo per la programmazione diretta del microcontrollore tramite, per esempio, interfaccia JTAG. Il software CCS comprende una sezione di programmazione, con un editor di linguaggio C/C++, e una sezione di debug/test, preconfigurata per l'accesso diretto alla memoria e alle variabili e l'impostazione di un numero elevato di breakpoints software. Il microcontrollore è corredato di una ampia libreria precompilata (TivaWare) per il controllo di tutte le periferiche e le funzionalità presenti. La libreria permette una programmazione ad alto livello del microcontrollore senza necessità di accesso ai registri di memoria. La libreria è inoltre caricata a bordo della ROM del microcontrollore. Questo permette di ridurre le dimensioni del codice in fase di programmazione.

2.3 Analisi di fattibilità di sensori acustici wireless per applicazioni di smart home

Lo scopo di questa attività è lo studio di fattibilità di sensori acustici a basso costo per applicazioni di smart home con una eventuale configurazione in ambienti confinati. L'impiego di tali sensori deve avere caratteristiche smart, ossia i segnali acquisiti devono essere elaborati per consentire la segnalazione di allarmi o comandi, oppure per la localizzazione e il conteggio delle persone in un ambiente chiuso. Altre funzionalità eventualmente utili sono il rilevamento di rumori noti (ad esempio legati a specifici elettrodomestici posizionati in punti noti, ecc.)

A tal fine è necessario disporre in uno stesso ambiente confinato di più sensori dello stesso tipo, che quindi devono risultare:

- piccoli e poco invadenti in termini di impatto visivo sulle persone (a tal fine ne deve essere anche considerata l'integrabilità all'interno di soprammobili o strutture esistenti, quali prese elettriche lampade, ecc.);
- a basso costo;
- wireless o facilmente collegabili con moduli wireless dedicati.
- direzionali o omnidirezionali secondo specifiche esigenze di localizzazione e/o di monitoraggio di alcuni elettrodomestici.

La prima fase dello sviluppo di questa attività ha quindi riguardato un'analisi di mercato sui dispositivi esistenti, considerando le loro caratteristiche elettriche e le loro prestazioni acustiche. Il risultato di questa fase è che difficilmente si trovano sul mercato, se non a costi non congrui con gli scopi proposti, sensori acustici in grado di soddisfare le esigenze di una smart home. Infatti la maggior parte dei sensori acustici disponibili sul mercato o sono pensati per applicazioni industriali, come ad esempio il monitoraggio dei sistemi meccanici e quindi presentano bande non utili per i segnali tipicamente presenti nelle nostre abitazioni, ossia segnali che rientrano nella banda udibile dell'orecchio umano, o sono pensati per applicazioni all'aperto (come ad esempio il monitoraggio del traffico) e quindi presentano caratteristiche geometriche e costi non adeguati per la nostra applicazione.

Oltretutto nessuno dei prodotti presente sul mercato è pensato per applicazioni di smart home, e non si presenta accoppiato ad un sistema di elaborazione a costo congruo per l'estrazione di features interessanti per l'analisi del segnale acustico in ambienti confinati, con occupanti umani. Per questo motivo si è arrivati alla conclusione che si rende necessaria la progettazione ad hoc di un sensore acustico intelligente per questo tipo di applicazioni.

Nella successiva fase dall'analisi di fattibilità del sensore si è quindi passati ad affrontare la stesura di uno schema di massima per la creazione di sensori acustici intelligenti, in grado di soddisfare le specifiche richieste.

Questi sensori smart devono essere costituiti da una parte dal sensore acustici vero e proprio e dall'altra dal sistema di pre-elaborazione e trasmissione del segnale acquisito, compresa la sua alimentazione. Tale soluzione sebbene più complicata dal punto di vista realizzativo, rispetto a quella di un dispositivo già esistente sul mercato, è, allo stesso tempo, innovativa e flessibile e meglio si adatta ad essere integrata in un

ambiente smart home. Si presta inoltre allo sviluppo di nuovi dispositivi brevettabili e direttamente inseribili sul mercato.
Chiaramente questo dovrà comportare allo stesso tempo una successiva attività di ricerca e sviluppo relativa sia alla parte dei sensori acustici sia e alla parte dei sistemi di elaborazione, trasmissione e via dicendo. Dal punto di vista dello studio di fattibilità questa attività di ricerca ha quindi affrontato da una parte un'indagine sui sensori acustici veri e propri e dall'altro le schede di acquisizione/trasmissione wireless ad essi collegati.
Riguardo i sensori acustici si è considerata la possibilità di utilizzo di una capsula microfonica a condensatore o magnetica o piezo-elettrica, e la necessità di preamplificazione o meno del segnale. Partendo dalla condizione che la banda udibile dall' orecchio umano si attesta in un range di 20 [Hz] a 20000 [Hz] è necessario lavorare su una capsula che lavori all' interno di questo intervallo. Il compromesso migliore in termini di costi e prestazioni è dato da una capsula a condensatore ossia un device che sfrutta l'effetto capacitivo per trasformare le onde acustiche in aria in tensioni. Il funzionamento di un microfono a condensatore si basa sulla variazione di tensione ai capi di un condensatore in cui un' armatura è fissa, l' altra è costituita dalla membrana del microfono stesso, a cui è fornita una quantità di carica Q=C*V. Le due lamine che compongono la capsula sono sollecitabili dalla variazione di pressione che un suono provoca nel mezzo di propagazione (aria). Il microfono a condensatore può esser considerato il migliore perché più sensibile ai transienti (rapida variazione dell' ampiezza del segnale) e alle sollecitazioni; tuttavia la capsula non ha un grande rendimento per cui viene aggiunto un preamplificatore atto ad alzare il livello del segnale generato. Questi preamplificatori possono essere già integrati in un dispositivo completo o realizzati a parte, sono disponibili diverse soluzioni in commercio e non sono critiche per costi di realizzazione.
Dal punto di vista delle caratteristiche direttive del sensore acustico bisognerà poi considerare la figura polare del sensore così costituito ossia la rappresentazione grafica su di un piano della sensibilità di un microfono in funzione della direzione di provenienza di un segnale; può essere:

- omnidirezionale (la risposta è uguale in tutte le radiazioni);
- bidirezionale (usato negli studi radiofonici ed il diagramma di radiazione assume una forma ad "8");
- unidirezionale (ha una sensibilità che varia a seconda della direzione di provenienza del suono);

Per il progetto in esame solo capsule/microfoni omnidirezionali o unidirezionali devono essere considerate. Il costo di questi tipi di sensore comunque non è molto alto (qualche euro) per cui la scelta progettuale non è gravosa e spesso i diversi dispositivi hanno caratteristiche simili.
Per quanto riguarda invece la parte relativa al sistema di elaborazione si deve considerare la possibilità di utilizzo di un scheda contenente un microcontrollore con wireless integrato. Una tale soluzione rende semplice lo sviluppo e soprattutto l'integrazione con sistemi di gestione di una molteplicità di sensori in maniera semplice. Dall'altro canto è bene valutare il posizionamento dei sensori all'interno dei singoli locali. La possibilità di comunicare usando z-wave o zigbee permette di connettere in una rete maglia più sensori tra loro. Questo vantaggio (ossia l'utilizzo di diverse tipologie di trasmissione) può essere risolto in maniera modulare affidando a due sistemi diversi l'acquisizione/elaborazione, da una parte, e la trasmissione dall'altra. Questa soluzione si presta meglio all'integrazione ed è quindi quella suggerita per la realizzazione finale. Va valutata anche 'l'intelligenza' di un tale sistema, ovvero se affidare a tale microcontrollore l'estrazione di features del segnale acustico o meno. Infatti non è detto che l'elaborazione debba essere affidata necessariamente ad microcontrollore per ogni microfono, quanto piuttosto ad un DSP. La scelta del microcontrollore può ricadere su board sviluppate con dispositivi AVR/Atmel come Arduino o su sistemi della Texas Instruments, che è ben nota per lo sviluppo di sistemi a DSP, adatti per l'elaborazione dei suoni, ma anche su sistemi della Microchip, che sviluppa sia microcontrollori che DSP, nonché schede che contengono entrambi i device. Un esempio di un sistema di sviluppo valutabile per la successiva fase di ricerca è *MPLAB Starter Kit for dsPIC DSC*, costituito da DSC dsPIC33FJ256GP506 a 40 MIPS, con memoria flash da 256 KB, SRAM da 16 KB, Memoria flash seriale da 4 Mbit a bordo scheda, CODEC audio da 16/24/32 bit con una frequenza di campionamento massima di 48 kHz, Circuiti di riproduzione e rilevamento audio dai costi contenuti grazie all'uso di un ADC a 12 bit e segnali audio PWM, Ingressi di linea e microfono con guadagno di ingresso regolabile, Amplificatore per cuffie da 100 mW con controllo digitale del volume, interruttori utente, 3 LED utente, Sensore di temperatura. Chiaramente prodotti simili sono disponibili sul mercato e sono centinaia le aziende che li producono. Bisogna però anche considerare che l'ingegnerizzazione della soluzione finale può presentare caratteristiche di vario genere che esulano anche da questo genere di prodotti. Complessivamente lo schema del sistema che si dovrebbe andare a realizzare per garantire le prestazioni desiderate sarà simile a quello mostrato in figura 2.13. Oltre le parti fondamentali già descritte (capsula, preamplificatore, DSP/MCU per l'elaborazione e sistema di gestione) è bene considerare la

possibilità di sfruttare un sistema di alimentazione che sfrutti l'energy harvesting, in modo da realizzare un sensore eventualmente alimentato a batteria, che recuperi dall'ambiente esterno energia attraverso ad esempio un pannellino fotovoltaico. Questo rappresenta un piccolo costo aggiuntivo dell'ordine di 4 o 5 euro, ma renderebbe il sensore realmente applicabile in qualsiasi ambiente o posizione della stanza in maniera da ridurre al minimo l'invasività dell'applicazione. Chiaramente si dovrà utilizzare un pannello in silicio amorfo o a film sottile che è più sensibile alla banda delle luci artificiali rispetto a pannelli in silicio monocristallino. Questo permette al meglio lo sfruttamento del sensore così costruito in ambienti Indoor. Inoltre per ridurre i consumi si attiverà la trasmissione del segnale solo quando necessaria e per fare questo dovranno essere sfruttate le caratteristiche di intelligenza locale dell'intero sistema così costituito. Le scelte progettuali proposte sono tutte realizzabili e si prestano anche ad una diversificazione del prodotto complessivo (con o senza energy harvesting).

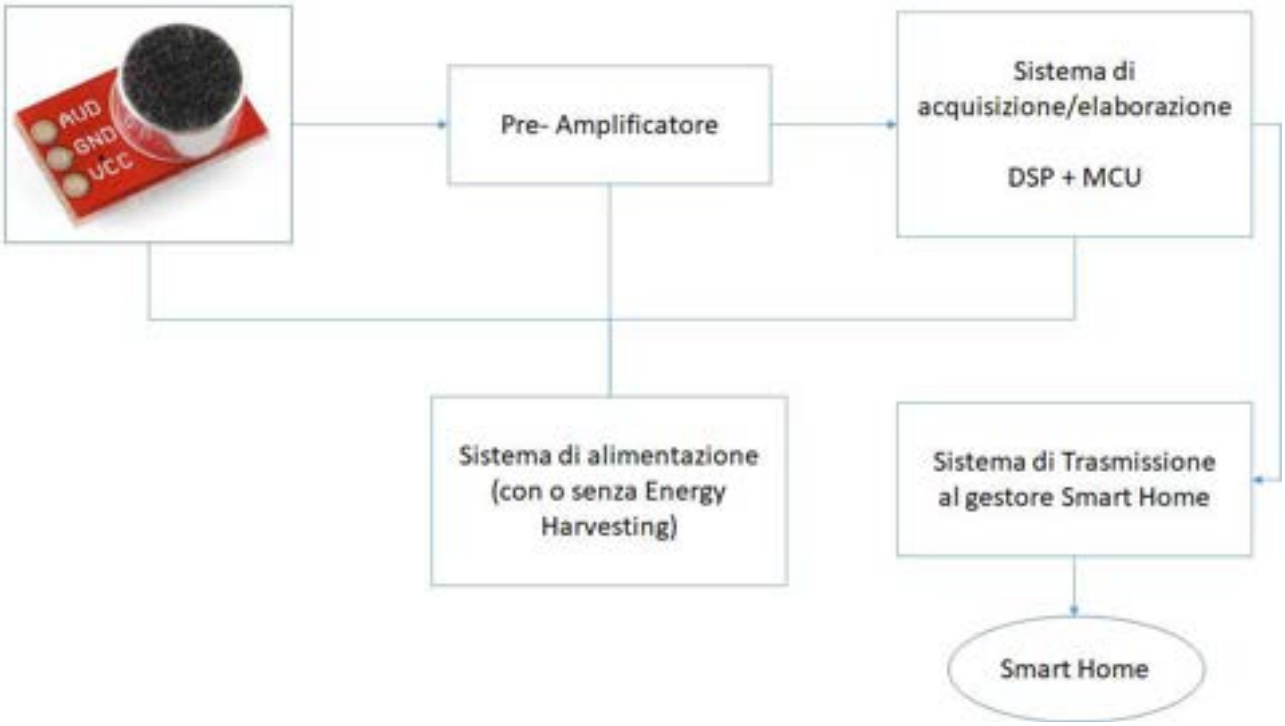

Fig. 2.13 – Schema del sensore acustico

2.4 Sviluppo di moduli di interfacciamento e gestione di dati di una rete di Smart Home

Questa attività si pone come obiettivo quello di recuperare i dati provenienti dai sensori installati nella rete di Smart Homes monitorate per la sperimentazione. Tali dati infatti, venivano collezionati in un DB proprietario del sistema Apio, al quale per motivi di privacy, essendo condiviso con altri utenti, non è concesso l'accesso diretto.
Si è quindi reso necessario lo sviluppo di moduli di interfacciamento tra quest'ultimo DB proprietario e una piattaforma Enea di archiviazione dei dati.
A tal fine è stato sviluppato un web service per fare il data exchange tra DB Apio (che accentra i dati provenienti dai vari sensori) e la piattaforma che dovrà prelevare i dati. Sono state in particolare messe a disposizione delle rotte che è possibile richiamare tramite web service. Con delle chiamate di tipo "GET", utilizzando specifici parametri, è stato infatti possibile avere accesso ai dati. I dati ritornati con queste chiamate, sono restituiti su file di tipo JSON e di essi è stato quindi possibile, grazie ad un parsing del file e

successive query di tipo insert, effettuarne la storicizzazzione. Si è quindi creato un database Enea contenente tutti i dati provenienti dalle abitazioni monitorate, che predispone i dati per successive elaborazioni sintetiche (es. valori medi giornalieri).

2.4.1 Sviluppo del Web Service

Per interfacciare il Database Apio con una piattaforma Enea di archiviazione dei dati, è stato sviluppato un Web Service di tipo RESTful. Tale servizio, è custode di un insieme di risorse sulle quali un client può chiedere le operazioni canoniche del protocollo HTTP (GET, POST, PUT, PATCH, DELETE). Esso infatti è un sistema software progettato per supportare un'interazione tra applicazioni, utilizzando le tecnologie e gli standard Web. L'approccio REST (REpresentational State Transfer) è ispirato ai principi architetturali tipici del Web e si concentra sulla descrizione di risorse, sul modo di individuarle nel Web e sul modo di trasferirle da una macchina all'altra e si è pertanto rilevato appropriato per effettuare il data exchange tra DB Apio, che accentra i dati degli oggetti e le varie piattaforme che dovranno prelevare i dati.
Sarà quindi possibile utilizzare delle API per svolgere delle operazioni all'interno di un database installato su un server connesso online:

- GET questo ci permette di richiedere al server un determinato set di dati;
- POST la possibilità di creare un nuovo oggetto all'interno del database;
- PUT con questo possiamo modificare o sostituire completamente un oggetto già esistente;
- DELETE ovvero possiamo, da remoto, cancellare un oggetto contenuti all'interno del database al quale siamo collegati.

Tramite le API, saremo quindi in grado di fare tutte queste operazioni senza la necessità di dover accedere direttamente al database Apio.

2.4.2 Modello dati Apio

Occorre fare alcune premesse sul modello dei dati Apio, ciò è infatti necessario per capire come poter richiamare tali dati tramite Web Service.
Nel modello dati Apio gli oggetti sono formati hanno un UUID (Universally unique identifier), che rappresenta un identificativo univoco universale per ciascun dato.
A sua volta, l' UUID è la composizione di ObjectID e APIOID, seguendo la seguente composizione: ObjectID_APIOID.
Si avrà quindi: UUID: ObjectID_APIOID
L'ObjectID identifica un oggetto all'interno di un sistema, ovvero ciò corrisponde in questo caso all'ID identificativo di uno specifico sensore all'interno di una determinata abitazione.
L'APIOID identifica invece un sistema, quindi nel nostro caso l'Energy Box relativo ad una specifica abitazione, dove il sensore è posizionato.
Ad esempio prendiamo il gateway ENEA3 (facente parte delle abitazioni monitorate per la sperimentazione e appartenete all'utente 3), al suo interno avremo diversi sensori configurati, relativi all'abitazione dell'utente 3, che sono identificati da diversi objectID, mentre il sistema ENEA3 è identificato da un APIOID.
Supponiamo che esista un sensore Energy Meter all'interno dell'abitazione dell'utente 3 e che esso abbia ObjectID: 21.
Supponiamo inoltre che il sistema ENEA3 abbia ApioID: 27466e1d-a9f6-4cde-a043-d25abebda469. Allora l'UUID che mi identifica il sensore Energy Meter all'interno dell'abitazione dell'utente3 sarà: 21_27466e1d-a9f6-4cde-a043-d25abebda469.
Esiste anche un ulteriore identificativo: il varID. Quest'ultimo è rappresentato dalle diverse proprietà che un oggetto può avere (ad esempio varID: power).

2.4.3 Come richiamare le Api

Abbiamo visto come le RESTful Api sono un insieme di tecniche che ci permettono di collegarci a un server esterno ed eseguire alcune operazioni sui dati che esso contiene.
Nel nostro caso ciò ci servirà, come già detto, per prelevare i dati delle abitazioni monitorate raccolti nel DB Apio e farli confluire verso una piattaforma Enea di archiviazione.

A tal fine abbiamo quindi effettuato, tramite il servizio realizzato che abbiamo descritto, chiamate di tipo "GET", utilizzando specifici parametri.
L'endpoint, ossia l'Url del servizio che viene esposto, è il seguente: https://www.apio.cloud/api .
Ad ogni richiesta via API, è necessario loggarsi all'interno del sistema che vogliamo utilizzare.
Nel contesto di una transazione, viene utilizzato il metodo "basic auth" (basic access authentication), che è un metodo per fornire credenziali di accesso tra client e server preservando così la sicurezza dei dai. L'accesso ai dati è quindi protetto da username e password riservate.
Il nome della rotta per richiamare i dati provenienti dai vari sensori nelle abitazioni monitorate, è chiamata "rawSensorData".
Le richieste di tipo GET che è possibile effettuare tramite web service, sono così costituite:
enpoint/nomeRotta/UUID/varID/timestampStart/timestampEnd
Passando le opzioni appropriate, è quindi possibile ottenere i dati da un determinato sensore, anche filtrando per data e ora (opzionale).
Per l'estrazione automatica dei dati tramite le suddette rote, è stata sviluppata un'apposita classe Java "getData", il quale codice verrà riportato in seguito.
Il risultato di queste chiamate, sono dei file JSON. In particolare viene ritornato un array di oggetti JSON, ciascuno contenente un dato e avente la seguente struttura:


```
[{
  "propertyName":"power",
  "varValue":"127.163",
  "varDataTime":"2017-03-03T17:42:44.196Z"
},
{
  "propertyName":"power",
  "varValue":"153.132",
  "varDataTime":"2017-03-03T17:47:21.198Z"
}]
```


Enea3 -EnergyMeter-power

Sensor	EnergyBox	propertyName	varValue	varDataTime
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	421,985	2017-03-03T17:30:55.617Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	420,567	2017-03-03T17:31:00.600Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	427,154	2017-03-03T17:42:14.305Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	427,592	2017-03-03T17:42:19.190Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	427,669	2017-03-03T17:42:24.183Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	428,499	2017-03-03T17:42:29.196Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	428,293	2017-03-03T17:42:34.187Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	423,531	2017-03-03T17:42:39.287Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	424,195	2017-03-03T17:42:44.196Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	423,33	2017-03-03T17:42:49.197Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	429,69	2017-03-03T17:42:54.187Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	422,974	2017-03-03T17:42:59.223Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	423,025	2017-03-03T17:43:04.200Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	422,871	2017-03-03T17:43:09.187Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	423,345	2017-03-03T17:43:14.191Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	422,871	2017-03-03T17:43:19.186Z
21	27466e1d-a9f6-4cde-a043-d25abebda469	power	423,397	2017-03-03T17:43:24.217Z

Fig. 2.14 – Esempio file csv contenente i dati estratti con una deteminata chiamata

Una volta effettuato il parsing dei file JSON ritornati, i dati di ciascuna chiamata sono stati riportati sul file csv, uno per ogni chiamata.
Dove “propertyName” indica il nome della proprietà misurata dallo specifico sensore a cui il dato si riferisce; “varValue" indica il valore della misura effettuata e “varDataTime" indica il timestamp di acquisizione del dato.
A tal proposito, per fare il parsing dei file JSON ritornati, è stata implementata una classe Java di supporto “jsonStructure”, che non fa altro che descrivere la struttura dei file JSON e quindi richiamarle i vari campi all’occorrenza.
In figura vediamo un esempio di esportazione dei dati su file csv, per quanto riguarda l’Energy Box ENEA3, sensore Energy Meter, proprietà power (tali parametri sono stati passati con una chiamata di tipo “GET” customizzata come descritto precedentemente).
Vediamo come i file sono stati organizzati in colonne e riportano la seguente struttura:
Sensor-EnergyBox-propertyName-varValue-varDataTime
Tali identificativi di colonna, rappresentano rispettivamente: ObjectID, APIOID, varID, valore misurato, timestamp relativo alla misurazione.
E’ chiaro come in questo modo sia possibile effettuare l’import dei dati all’interno di un Data Base. Infatti è stato realizzato un DB di storage in MySql e presenta la seguente struttura:
Id Energy Box - Nome sensore - Timestamp – Property- Valore

id	ID_EB	Nome_sensore	Timestamp	Property	Valore
369586	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 07:15:00	Somma delle Energie nel quarto...	296.154
369587	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 07:00:00	Media POWER [W]	1244.045
369588	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 07:00:00	Somma delle Energie nel quarto...	328.635
369589	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 06:45:00	Media POWER [W]	1244.034
369590	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 06:45:00	Somma delle Energie nel quarto...	293.385
369591	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 06:30:00	Media POWER [W]	1243.995
369592	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 06:30:00	Somma delle Energie nel quarto...	330.004
369593	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 06:15:00	Media POWER [W]	1243.958
369594	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 06:15:00	Somma delle Energie nel quarto...	291.985
369595	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 06:00:00	Media POWER [W]	1243.941
369596	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 06:00:00	Somma delle Energie nel quarto...	328.953
369597	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 05:45:00	Media POWER [W]	1243.909
369598	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 05:45:00	Somma delle Energie nel quarto...	293.01
369599	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 05:30:00	Media POWER [W]	1243.865
369600	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 05:30:00	Somma delle Energie nel quarto...	331.006
369601	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 05:15:00	Media POWER [W]	1243.824
369602	97a2538f-388e-4240-9c24-74218022e599_ENEA 7(...	21_Consumo Generale	2017-07-23 05:15:00	Somma delle Energie nel quarto...	290.917

Fig. 2.15 – Esempio di visualizzazione del DB

Notiamo come i campi del DB siano equivalenti a quelli riportati nei file csv.
I dati puntuali raccolti dai vari Energy Box disposti nelle abitazioni, sono quindi stati raccolti e storicizzati per elaborazioni successive.

2.4.4 Parti di codice sviluppato (classe “getData”)

```
public class getData {

        public static String sendRequest(String apiName, int objID, String
apioID, String property, String user, String password, boolean returnResponse) {

            URL url = null;

            String tmpUrl = "";
```

```
            try {
                //tmpUrl = "https://www.apio.cloud/api/" + apiName + "/" + objID
+ "_" + apioID + "/" + property + "/" + "2017-03-05T23:00:00.000Z/2017-03-
15T11:49:41.240Z";
                tmpUrl = "https://www.apio.cloud/api/" + apiName + "/" + objID +
"_" + apioID + "/" + property;
                url = new URL(tmpUrl);
                System.out.println(tmpUrl);
            } catch (MalformedURLException e) {
                System.out.println("MalformedUrlException: " + e.getMessage());
                e.printStackTrace();
                return "-1";
            }
            HttpsURLConnection uc = null;
            try {
                uc = (HttpsURLConnection)url.openConnection();
            } catch (Exception e) {
                //System.out.println("IOException: " + e.getMessage());
                //e.printStackTrace();
                return "-12";
            }

            String userpass = user + ":" + password;
            String basicAuth = "Basic " +
javax.xml.bind.DatatypeConverter.printBase64Binary(userpass.getBytes());
            uc.setRequestProperty("Authorization", basicAuth);
            InputStream is = null;
            try {
                is = uc.getInputStream();
            } catch (IOException e) {
```

```
                //System.out.println("IOException: " + e.getMessage());
                //e.printStackTrace();
                System.out.println("Errore del server remoto");
                return "-13";
            }
            if (returnResponse) {
                BufferedReader buffReader = new BufferedReader(new
InputStreamReader(is));
                StringBuffer response = new StringBuffer();

                String line = null;
                try {
                    line = buffReader.readLine();
                } catch (IOException e) {
                    e.printStackTrace();
                    return "-1";
                }
                while (line != null) {
                    response.append(line);
                    response.append('\n');
                    try {
                        line = buffReader.readLine();
                    } catch (IOException e) {
                        System.out.println(" IOException: " + e.getMessage());
                        e.printStackTrace();
                        return "-14";
                    }
                }
                try {
                    buffReader.close();
```

```
                } catch (IOException e) {
                    e.printStackTrace();
                    return "-15";
                }
                System.out.println("Response: " + response.toString());
                return response.toString();
            }
            return "0";
        }
        public static int logToFile(String objID, String apioID, String jsonObj)
throws FileNotFoundException {
                Gson gson = new Gson();
                Type collectionType = new
TypeToken<Collection<jsonStructure>>(){}.getType();
                Collection<jsonStructure> jsonObjColl = gson.fromJson(jsonObj,
collectionType);

                PrintStream ps = new PrintStream( new
FileOutputStream("R:\\Desktop\\tuoFile.csv",true) );
                for (jsonStructure j : jsonObjColl){
                        ps.println(objID + "," + apioID + "," +
j.getPropertyName() + "," + j.getvarValue() + "," + j.getDataTime());
                }
                ps.close();
                return 0;
        }

        public static void initializeFile() throws FileNotFoundException {
                PrintStream ps = new PrintStream( new
FileOutputStream("R:\\Desktop\\tuoFile.csv") );
```

```
ps.println("Sensor,EnergyBox,propertyName,varValue,varDataTime");
                ps.close();
        }

        public static void main(String[] args) throws FileNotFoundException,
IOException {
                try {
                        initializeFile();
                        for (int i=17;i <= 17;i++){
                                String retJson = sendRequest("rawSensorData", i,
"97a2538f-388e-4240-9c24-74218022e599", "power", "User", "Password", true);
                                if(retJson != "-13"){
                                        logToFile(String.valueOf(i),"27466e1d-
a9f6-4cde-a043-d25abebda469",retJson);
                                }
                        }
        } catch (Exception e) {
            e.printStackTrace();
        }
}
}
```

2.4.5 Parti di codice sviluppato (classe ““jsonStructure”)

```
public class jsonStructure {
      private String propertyName;
      private String varValue;
      private String varDataTime;
```

```
	public jsonStructure(String pPropertyName, String pVarValue, String
pVarDataTime) {
		this.propertyName = pPropertyName;
		this.varValue = pVarValue;
		this.varDataTime = pVarDataTime;
	}

	public String getPropertyName(){
		return this.propertyName;
	}

	public String getvarValue(){
		return this.varValue;
	}

	public String getDataTime(){
		return this.varDataTime;
	}
}
```

3 Conclusioni

Il lavoro presentato ha riguardato lo sviluppo di sensori intelligenti e moduli di interfacciamento per il recupero dati per applicazioni Smart Home. Lo studio ha riguardato in prima istanza lo sviluppo di un sensore virtuale di anidride carbonica (CO2). Il sensore virtuale misura indirettamente la CO2 a partire da acquisizioni di grandezze ad essa correlate e da un modello matematico. Da analisi preliminari sui dati, è emersa la necessità di usare un modello dinamico caratterizzato da equazioni di stato. Senza perdere di generalità, il modello è stato implementato sfruttando delle reti neurali auto regressive (NARX) capaci di rappresentare un sistema dinamico di tipo ingresso-stato-uscita. Le reti sono state implementate in ambiente Matlab ed addestrate e testate con successo su dati sperimentali acquisiti nei locali dell'ENEA a Casaccia, in un ufficio di circa 15 metri quadri ossia circa 40/45 metri cubi. È stata poi studiata una piattaforma a microcontrollore adatta alla implementazione integrata della suddetta rete neurale, e al suo interfacciamento con un sistema di sensori adeguato. In seconda istanza, è stato effettuato uno studio di fattibilità per l'implementazione di un sensore wireless acustico per applicazioni di Smart Home. Sono stati studiati i dispositivi attualmente presenti sul mercato, e in particolare, la loro applicabilità ad un sistema di Smart Home in relazione alle loro caratteristiche tecniche. Sono stati poi analizzati gli stadi di pre-condizionamento del segnale e la successiva digitalizzazione. Infine, sono stati studiati i moduli di interfacciamento e di gestione dati di una rete Smart Home con supporto ad un database remoto.

4 Riferimenti bibliografici

[1] BHATTACHARYA, Sayantani; SRIDEVI, S.; PITCHIAH, R. Indoor air quality monitoring using wireless sensor network. In: *Sensing Technology (ICST), 2012 Sixth International Conference on*. IEEE, 2012. p. 422-427.

[2] KIM, Jong-Jin; JUNG, Sung Kwon; KIM, Jeong Tai. Wireless monitoring of indoor air quality by a sensor network. *Indoor and built Environment*, 2010, 19.1: 145-150.

[3] SAAD, Shaharil Mad, et al. Implementation of index for real-time monitoring indoor air quality system. In: *Electronic Design (ICED), 2014 2nd International Conference on*. IEEE, 2014. p. 53-57.

[4] ZIN, Mohd Hanif Mohd; ISMAIL, Sharifah Norkhadijah Syed; KARUPPIAH, Karmegam. Smart Co2 Detector Prototype Development to Enhance the Efficiency of Ventilation System in a Building. *Iranian Journal of Public Health*, 2016, 45.1: 93.

[5] ZHOU, Jiaqing; KIM, Chang Nyung. Numerical investigation of indoor CO2 concentration distribution in an apartment. *Indoor and Built Environment*, 2011, 20.1: 91-100.

[6] Wang, H., Xie, L., Liu, S., & Xu, J. (2016, June). A model-based control of CO2 concentration in multi-zone ACB air-conditioning systems. In *Control and Automation (ICCA), 2016 12th IEEE International Conference on* (pp. 467-472). IEEE.

[7] HAN, Zhenyu; GAO, Robert X.; FAN, Zhaoyan. Occupancy and indoor environment quality sensing for smart buildings. In: *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*. IEEE, 2012. p. 882-887

[8] Jin, M., Bekiaris-Liberis, N., Weekly, K., Spanos, C., & Bayen, A. (2015). Sensing by proxy: Occupancy detection based on indoor CO2 concentration. *UBICOMM 2015*, *14*.

[9] Nematchoua, M. K., Tchinda, R., Orosa, J. A., & Roshan, G. (2014). Study of dioxide carbon concentration and indoor àir quality in some buildings in the equatorial region of Cameroon (Yaoundé).

[10] Laudani, A., Lozito, G. M., Fulginei, F. R., & Salvini, A. (2015). On training efficiency and computational costs of a feed forward neural network: a review. Computational intelligence and neuroscience, 2015, 83.

[11] Riganti Fulginei, F., Laudani, A., Salvini, A., & Parodi, M. (2013). Automatic and parallel optimized learning for neural networks performing MIMO applications. Advances in Electrical and Computer Engineering, 13(1), 3-12.

[12] Lozito, G. M., Laudani, A., Fulginei, F. R., & Salvini, A. (2014). FPGA implementations of feed forward neural network by using floating point hardware accelerators. Advances in Electrical and Electronic Engineering, 12(1), 30.

[13] Laudani, A., Lozito, G. M., Fulginei, F. R., & Salvini, A. (2014, March). An efficient architecture for floating point based MISO neural neworks on FPGA. In Computer Modelling and Simulation (UKSim), 2014 UKSim-AMSS 16th International Conference on (pp. 12-17). IEEE.

[14] Laudani, A., Fulginei, F. R., Salvini, A., Lozito, G. M., & Mancilla-David, F. (2014, June). Implementation of a neural MPPT algorithm on a low-cost 8-bit microcontroller. In Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2014 International Symposium on (pp. 977-981). IEEE.

[15] Lozito, G. M., Bozzoli, L., & Salvini, A. (2014, September). Microcontroller based maximum power point tracking through FCC and MLP neural networks. In Education and Research Conference (EDERC), 2014 6th European Embedded Design in (pp. 207-211). IEEE.

[16] Lozito, G. M., Schmid, M., Conforto, S., Fulginei, F. R., & Bibbo, D. (2015). A Neural Network Embedded System for Real-time Estimation of Muscle Forces. Procedia Computer Science, 51, 60-69.

Curricula Autori

A. Laudani, F. Riganti Fulginei, G.M. Lozito (ESTLab)

Il laboratorio universitario ESTLab (Electrical Science and Technology Laboratory) è un laboratorio di ricercatori e professori universitari afferenti al Dipartimento di Ingegneria dell'Università degli Studi Roma Tre. Il laboratorio sviluppa software e dispositivi per industrie e laboratori di ricerca. I temi di maggior interesse sono le tecnologie fotovoltaiche, l'energy harvesting, i dispositivi magnetici, microcontrollori/FPGA, sensori. Il laboratorio vanta esperienze pluriennali nell'ambito del calcolo scientifico, degli algoritmi di ottimizzazione, del calcolo parallelo, delle simulazioni di modelli e del soft-computing.

Martina Botticelli ha conseguito la Laurea Specialistica in Ingegneria Informatica presso l'Università di Roma Tre. Dottoranda di Ricerca sull'analisi e sviluppo di metodologie di diagnosi, telecontrollo e gestione energetica di Smart Building Networks, in ambito residenziale e terziario.