UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea in Ingegneria Elettronica

Studio, sviluppo e validazione di un sistema a microcontrollore per l'acquisizione di dati di temperatura da sensori eterogenei

Study, design and validation of a microcontroller-based system for temperature data acquisition from heterogeneous sensors

Relatore:

Prof.ssa Susanna Spinsante

Tesi di Laurea di:

Kevin Di Leo

Correlatore:

Dott. Ing. Gianluca Ciattaglia

A.A. **2022** / **2023**

Indice

Introduzione

Il presente lavoro di tesi tratta dello studio, sviluppo e validazione di un sistema a microcontrollore per l'acquisizione di dati di temperatura da sensori eterogenei, ed è stato svolto in tre fasi: una prima fase è stata quella dello studio del materiale fornito, ovvero i tre sensori di temperatura e le tre schede a microcontrollore, per individuare la migliore configurazione hardware funzionale allo scopo.

Per quanto riguarda i sensori per la misura di temperatura, due di questi presentano la stessa tecnologia (infrarosso) mentre il terzo è l'unico funzionante a contatto. Le tre schede di sviluppo individuate per la scelta hanno caratteristiche simili, con lievi differenze tra di esse. La fase iniziale è consistita nel capire quale tra le tre schede potesse meglio interfacciarsi con i tre sensori simultaneamente ed acquisire valori di temperatura.

La seconda fase invece, è consistita nello sviluppo del codice necessario a far funzionare il sistema a microcontrollore. Il sistema, infatti, oltre che acquisire i valori di temperatura dai tre sensori, deve anche lavorare come data logger e quindi salvare i valori acquisiti. Inoltre, il codice doveva essere tale da calcolare alcuni parametri statistici come la deviazione standard campionaria e la media dei valori complessivamente acquisiti in ogni sessione di misura.

La terza fase è stata quella sperimentale. Gli esperimenti svolti sono stati effettuati in due diverse modalità: la prima con una misurazione di temperatura costante, la seconda con una misurazione di temperatura variabile. Inoltre, sono stati sviluppati due codici in MATLAB per elaborare i risultati ottenuti dalle prove e poterne analizzare e graficare i risultati.

Il progetto è stato di tipo sperimentale, quindi la teoria si è intrecciata con la pratica. Quest'ultima è stata presente in gran parte del progetto, dallo sviluppo del codice alle prove di misura finali.

Tramite quest'ultime si sono potute inoltre evidenziare le differenze tra i tre sensori: seppur due lavorassero con la tecnologia infrarosso, comunque in fase di prova hanno generato risultati diversi. Analogamente per quanto riguarda il sensore di contatto.

La tesi è stata organizzata in tre capitoli. Il primo capitolo riporta brevi accenni teorici riguardante il progetto: il principio di funzionamento dei sensori infrarosso e di contatto, la modalità di collegamento utilizzata tra schede e sensori e altri concetti utili per avere un quadro della componentistica usata e delle sue caratteristiche.

Il secondo capitolo è dedicato alla descrizione dei sensori e delle schede a microcontrollore, e dei parametri con cui si è arrivati a scegliere la scheda con cui poi si è lavorato. Inoltre, si descrive il codice sviluppato per il sistema di acquisizione e per far lavorare la scheda come data logger.

Il terzo capitolo descrive le tre prove che sono state svolte e commenta i risultati che sono stati ottenuti, riportando e analizzando anche i relativi grafici.

Il lavoro svolto può essere ampliato in futuro aggiungendo al sistema per esempio altre tipologie di sensori come quelli per il rilevamento del gas, della pressione o altri parametri fisici e rendere quindi il sistema sviluppato dedicato non solo alla misura di temperatura ma di più grandezze fisiche.

Capitolo 1
Richiami teorici

1.1 Richiami di teoria delle misure

Misurare una grandezza vuol dire metterla in relazione con una grandezza campione; tutte le grandezze fisiche sono misurabili: peso, lunghezza, forza, temperatura. Le sette grandezze fisiche fondamentali sono racchiuse in quello che viene definito il *Sistema Internazionale di unità di misura (SI)*.

1.1.1 Il sistema internazionale di unità di misura

Abbreviato come **SI** [1], il Sistema Internazionale delle unità di misura è un insieme nel quale sono state introdotte e definite le sette unità di misura cosiddette fondamentali. Tutte le restanti unità di misura per grandezze fisiche non contenute nel SI sono definite come derivate. Per ogni unità di misura fondamentale è definito un simbolo che la rappresenta.

Grandezza fisica	**Unità di misura**	**Simbolo**
Lunghezza	metro	m
Tempo	secondo	s
Massa	kilogrammo	kg
Intensità di corrente	ampere	A
Temperatura	kelvin	K
Quantità di materia	mole	mol
Intensità luminosa	candela	cd

Tabella 1.1: Unità di misura fondamentali del SI.

Come si può notare in Tabella 1.1, i simboli in maiuscolo (A per Ampere e K per Kelvin) sono quelli che rappresentano cognomi di scienziati che hanno contribuito all'unità di misura che rappresentano.

Come detto precedentemente, le grandezze non contenute nella tabella sono dette derivate, per esempio la velocità. Essa è definita come il rapporto tra lunghezza e tempo con unità di misura i metri al secondo (*m/s*). Da queste sette grandezze fondamentali si possono derivare le unità di misura di qualsiasi grandezza.

A volte si ha bisogno di multipli o sottomultipli di tali grandezze: in tabella se ne riportano alcuni.

Prefisso	Fattore di conversione
mega	10^6
kilo	10^3
deca	10^1
deci	10^{-1}
milli	10^{-3}
micro	10^{-6}

Tabella 1.2: Multipli e sottomultipli delle grandezze.

1.1.2 Tipi di grandezze

Una grandezza può essere principalmente di tre tipi [2]:

- **Grandezze razionali**: sono quelle esprimibili tramite un numero di tipo razionale.
- **Grandezze numerali**: sono espresse da numeri interi positivi, per esempio: il numero di persone che studiano ingegneria in un certo quartiere.
- **Grandezze complesse**: rientrano in questo sottoinsieme le tipologie di grandezze del tipo vettoriale o tensoriale, per esempio.

1.1.3 Metodi di misura

Quando bisogna misurare un certo valore, possono essere implementati due metodi [2]:

- **Metodo diretto**: in questo caso, la misurazione avviene confrontando direttamente l'elemento in esame con il campione dell'unità di misura di riferimento. Per esempio, misurare una lunghezza di un oggetto direttamente con un righello.
- **Metodo indiretto**: la misurazione avviene tramite la misurazione di altre grandezze, come per il caso del volume del cubo ottenuto a partire dalla misurazione del lato (tramite metodo diretto), sfruttando opportuni legami funzionali.

1.1.4 Indici di variabilità

Le misurazioni sono sempre soggette a variabilità più o meno aleatoria. Ci sono casi in cui questa variabilità può essere più accentuata e casi in cui tende a esserlo meno.

Si pensi di voler misurare la variazione di temperatura all'interno di una stanza in un certo range temporale: difficilmente si avranno grandi variazioni tra un valore ed un altro.

Se invece, sempre all'interno della stessa stanza, viene acceso un camino o una stufa, si potrà notare come i valori aumenteranno man mano che scorre il tempo.

Per poter capire in che modo in valori, o ancor meglio, le variabili che sono state misurate tendono a distribuirsi nel corso del tempo si possono usare gli indicatori statistici.

L'indicatore statistico più diffuso e di più facile applicazione è sicuramente la media. Ma a volte, in ambito ingegneristico, la media potrebbe risultate insufficiente per poter analizzare con completezza il modo in cui una variabile tende a evolvere nel tempo.

Per questo si usano gli indicatori statistici variabili come la **varianza** e la **deviazione standard** (scarto quadratico medio).

Quando si effettuano delle misure e se ne calcola la media, essa rappresenta una sorta di baricentro delle nostre misure [3]. Seppur utile, non fornisce un'idea di come i dati tendano a disporsi nel corso del tempo. La deviazione standard e la varianza forniscono degli indicatori di quanto queste misure tendono ad allontanarsi dalla media [3].

Infatti, se si esegue la differenza tra la variabile aleatoria (come può essere una misurazione di temperatura) e il valore medio di tutte le variabili misurate, si ha una misura dello scostamento. Elevando al quadrato e sommando tutti gli scostamenti, dividendo per il numero di misure, si ottiene la varianza (**1.1**):

$$\sigma^2_x = \frac{\sum_{i=1}^{N}(x_i - \bar{X})^2}{N} \quad (1.1)$$

Per ragioni di comodità, si preferisce usare in alcune situazioni la deviazione standard (**1.2**): essa non è altro che la radice quadrata della varianza:

$$\sigma_{x=} = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{X})^2}{N}} \quad (1.2)$$

La deviazione standard è legata alla varianza, essendo quest'ultima il quadrato della prima. Entrambe forniscono una misura della dispersione che una certa variabile ha intorno al valore medio assunto da essa.

Piccoli valori della deviazione standard indica che i valori misurati si discostano poco dal valore medio, viceversa, valori maggiori indicano che le variabili tendono ad allontanarsi maggiormente dal valore medio (quindi dal baricentro della misura).

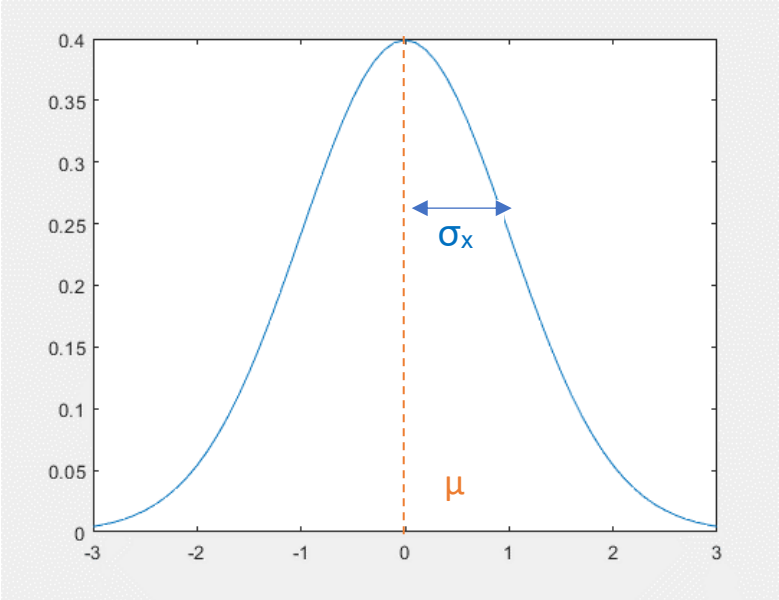


Figura 1.1: Gaussiana generata in MATLAB, in evidenza valore medio e deviazione standard.

In **Figura 1.1** un esempio di distribuzione probabilistica (la Gaussiana), dove si può visualizzare un valore di deviazione standard e il valore medio (μ), che cade proprio al centro della curva.

1.2 Sensori e il loro funzionamento

Nel mondo di oggigiorno si è circondati da sensori. Basti pensare alla sensoristica presente negli smartphone, nelle automobili, negli antifurto e in tanti altri esempi. Un sensore da solo non può fare molto senza un dispositivo che ne interpetri i valori acquisiti e ne monitori il funzionamento: ovvero un microcontrollore o un dispositivo analogo.

Ogni qualvolta è presente un sensore, esso sarà in comunicazione con un microcontrollore (o una CPU) che ne acquisirà i dati e ne farà le dovute elaborazioni.

In questa sezione si farà una breve panoramica su cosa è un sensore in generale, per poi concentrarsi sulla sensoristica per la misurazione della temperatura.

1.2.1 Sensori

Un sensore è un dispositivo che riesce a percepire variazioni fisiche di una certa grandezza, fornendo un output proporzionale a quest'ultime.

Generalmente si commette l'errore di considerare sensori e trasduttori come la stessa cosa: il trasduttore trasforma la grandezza fisica in una grandezza elettrica proporzionale, il sensore non è detto che abbia un output in tensione o in corrente.

Dagli ultimi anni del secolo scorso, la crescita tecnica in ambito sensoristico è stata rapida ed esponenziale. Il motivo di ciò è dovuto soprattutto alla sempre più spinta miniaturizzazione dell'elettronica, che permette di avere circuiti integrati e sensori sempre più piccoli [4].

La risposta alla domanda di come funziona un sensore non è univoca. Esistono svariate tipologie di sensori, ognuno dedicato alla misurazione di una certa grandezza fisica. Il loro funzionamento dipende proprio da ciò che dovranno misurare.

In **Figura 1.2**[1] un esempio di sensore per il rilevamento di gas.

Figura 1.2: Sensore di gas.

L'output di un sensore può essere generalmente in tensione, in corrente o in resistenza. Inoltre, si può avere un sensore digitale che comunica inviando pacchetti di bit oppure sensori con uscita analogica.

Lavorando con sensori di tipo analogico si avrà la necessita di interporre tra essi e il microcontrollore un convertitore analogico digitale, in modo tale che i segnali generati dal sensore possano essere acquisiti dal microcontrollore stesso (o l'acquisitore che si è scelto di usare). Nel prossimo paragrafo si vedrà come un segnale in uscita da un sensore abbia bisogno di essere dapprima processato e solo successivamente acquisito per l'utilizzo.

1.2.2 Condizionamento del segnale

Il sensore, in una rete di acquisizione dati, è l'elemento che interagisce con il mondo circostante: sarà sollecitato da degli stimoli esterni che gli permetteranno di fornire un output proporzionale ad essi.

Si ipotizzi che l'output prodotto dal sensore sia analogico e in tensione. Se lo si fornisce direttamente in ingresso al microcontrollore, quest'ultimo (nel caso non abbia un convertitore ADC integrato) non riuscirà ad interpetrare i valori di tensione che rappresentano per esempio una temperatura.

[1] Figura 1.2 tratta https://www.robotstore.it/Modulo-Sensore-rilevatore-di-Gas-e-Fumo-MQ-2

Sarà necessario un convertitore analogico-digitale che trasformi il segnale del sensore in un segnale interpretabile dal microcontrollore (si ricorda che un microcontrollore lavora con segnali digitali).

Bisogna fare una precisazione: schede a microcontrollore come Arduino UNO, Raspberry Pi Pico o anche microcontrollori come il PIC16F887, dispongono già a bordo di un convertitore analogico digitale. Un convertitore analogico digitale esterno servirà solamente in quelle schede di sviluppo in cui non ne è presente uno a bordo.

Quindi, quando successivamente si parlerà di conversione analogico-digitale si intenderà quella eseguita dal convertitore interno della scheda a microcontrollore che si è scelto di usare.

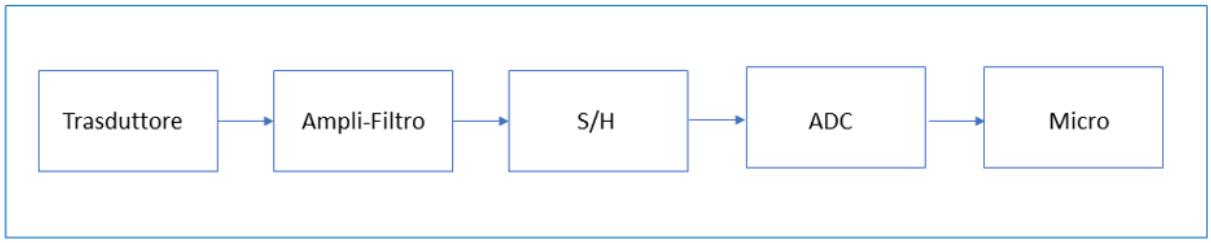


Figura 1.3: Catena di acquisizione.

In **Figura 1.3** è mostrato lo schema di una classica catena di acquisizione di un segnale: una temperatura, una luminosità o un rumore vengono acquisiti e processati fino a diventare un segnale usufruibile dall'utente. La maggior parte delle volte i passaggi fondamentali sono la riduzione del rumore attraverso dei filtraggi, l'amplificazione e la conversione analogico – digitale.

Si è parlato di amplificazione poiché la maggior parte dei sensori fornisce livelli di tensione o analogamente di corrente molto bassi, nell'ordine dei µV o µA, rispettivamente. Tramite un circuito di amplificazione è possibile riportarli a livelli di ampiezza accettabili per la lavorazione stessa del segnale negli stadi successivi.

1.2.3 Radiazioni infrarosso

Due dei tre sensori adoperati nel progetto sfruttano la tecnologia infrarosso per la rilevazione della temperatura: l'AMG8833 e MLX90614, di cui si avrà modo di parlare in modo specifico più avanti. Entrambi i sensori basano il loro funzionamento su un principio fisico (*Legge di Stefan-Boltzmann*).

Prima di descrivere come funzioni un sensore infrarosso è bene capire cosa sia una radiazione infrarosso, anche detta radiazione IR.

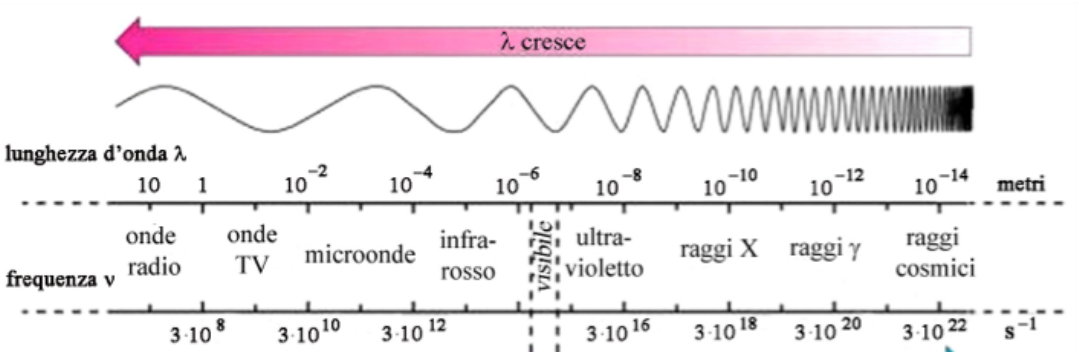


Figura 1.4: spettro elettromagnetico.

Sono definite radiazioni infrarosso, tutte quelle radiazioni elettromagnetiche che presentano una lunghezza d'onda tra 1 *mm* e 0,7 *μm* [5]. Come si può notare dallo spettro in **Figura [2] 1.4**, la banda infrarosso si trova tra le microonde e la banda visibile; il nome deriva dal fatto che si estende fino a toccare la prima radiazione visibile, che è proprio quella relativa al colore rosso [5].

La banda infrarosso è ampia: per questo di solito si tende a suddividerla in quattro zone principalmente [5]:

- **Infrarosso vicino**: 0,7 a 10 *μm*
- **Infrarosso medio**: 10 a 50 *μm*
- **Infrarosso lontano**: 50 a 300 *μm*
- **Infrarosso estremo**: 300 *μm* a 1 *mm*

1.2.4 Legge di Stefan-Boltzmann

I sensori infrarosso basano il loro principio di funzionamento sulla **Legge di Stefan-Boltzmann**. Dalla fisica è noto che la temperatura di un corpo è dovuta al moto di agitazione delle molecole che lo costituiscono. Ogni molecola è costituita da molteplici atomi, che a loro volta sono formati da un nucleo (protoni e neutroni) e da elettroni che orbitano intorno ad esso.

Come noto, una carica elettrica in movimento genera un campo elettrico variabile, che a sua volta genera un campo magnetico variabile: ovvero un campo elettromagnetico. Quando quest'ultimo è associato a calore si parla di radiazione termica. La maggior parte delle radiazioni termiche si trova proprio in corrispondenza delle radiazioni infrarosso (**Figura [3]1.5).**

[2] Figura 1.4 tratta da https://ibseedintorni.com/2015/01/04/esplorare-lo-spettro-elettromagnetico/
[3] Figura 1.5 tratta da https://unirc.it/documentazione/materiale_didattico/1467_2015_404_23323.pdf

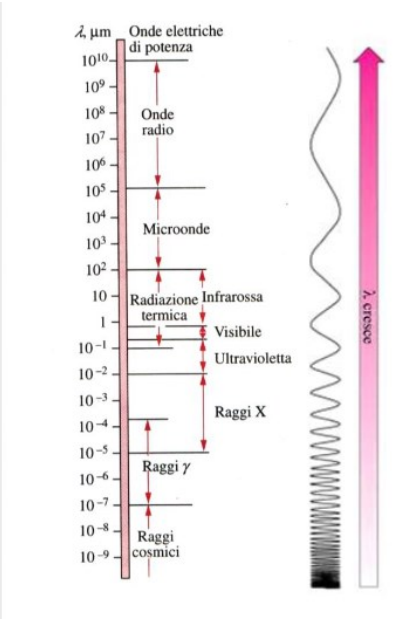


Figura 1.5: Localizzazione radiazione termica.

Un'onda elettromagnetica può essere descritta dalla sua lunghezza d'onda λ e dalla sua frequenza ν tramite il legame che intercorre con la velocità della luce c **(1.3)** [6]

$$\lambda = \frac{c}{\nu} \quad (1.3)$$

Tra i risultati notevoli riguardanti le radiazioni termica si trova la Legge di Planck **(1.4)** [6]

$$W_\lambda = \frac{\varepsilon(\lambda) C1}{\pi \lambda \hat{} 5 (e \hat{} (\frac{C2}{\lambda T}) - 1)} \quad (1.4)$$

Effettuando la derivata prima di (1.4) e uguagliandola a zero, si ottiene la Legge di Wien (**1.5)** [6]

$$\lambda_m = \frac{2898}{T} \quad (1.5)$$

Essendo la temperatura dovuta al moto di infinite molecole, ognuna con sua una energia cinetica e con una differente radiazione termica (e quindi infinite lunghezza d'onda), la legge di Wien permette di calcolare la lunghezza d'onda più probabile a cui vibra una particella che presenta una certa temperatura *T* [6].

Una radiazione termica ha uno spettro illimitato, però naturalmente un sensore potrà lavorare solamente in un determinato range di frequenze.

La legge di Stefan Boltzmann permette di avere un'approsimazione della potenza totale di radiazione termica irradiata in una determinata banda **(1.6)** [6]

$$\Phi_{bo} = A \varepsilon \sigma T^4 \quad (1.6)$$

Dove A è un fattore geometrico, T la temperatura, ε l'emissività e σ la costante di Stefan-Boltzmann pari a 5.67×10^{-8} W/m^2 K^4.

1.2.5 Sensori infrarosso

Ogni oggetto che presenta una temperatura sopra lo zero assoluto emana una certa quantità di radiazione termica, la cui intensità è governata dalla Legge di Stefan-Boltzmann (1.6).

Il componente base di un sensore IR è il cosiddetto *sensing plate*, un piccolo strato di materiale che presenta un'alta emissività nello spettro di frequenze considerato (si ricorda che la radiazione termica emanata da un qualsiasi oggetto presenta infinite lunghezze d'onda, un sensore IR lavorerà solo in un range delle medio-alte IR, ovvero quello dove è contenuta gran parte del calore).

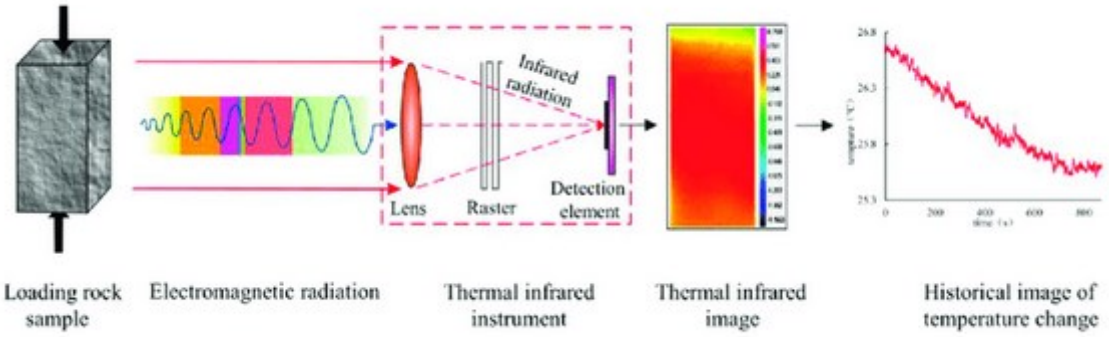


Figura 1.6: Esempio di come funziona il rilevamento di una radiazione termica.

L'oggetto di cui si vuol misurare la temperatura avrà una sua temperatura T ed una certa emissività ε; come noto irradierà un certo flusso di radiazione termica, la cui intensità è data dalle Legge di Stefan Boltzmann. Parte di questa radiazione incontrerà il *sensing plate* del sensore, anch'esso con una sua temperatura T_s ed una certa emissività ε_s. Si evince quindi, che il flusso che si avrà tra sensore ed oggetto potrà essere calcolato dalla (1.6) come [6]:

$$\Phi = A\varepsilon\ \varepsilon_s\ \sigma(T^4 - T_s^4) \quad (1.7)$$

Se si prende in mano un sensore IR qualsiasi, per esempio l'MLX90614, si potrà notare la presenza di una piccola lente ottica: essa serve proprio ad indirizzare la maggior quantità di flusso termico sul *sensing plate*.

La variazione di temperatura subita dal *sensing plate* dovrà essere convertita in un valore elettrico da un trasduttore apposito, per esempio un sensore thermopile (termopila in italiano).

Si possono riassumere gli step di funzionamento in questo modo: un oggetto emana una certa radiazione infrarosso, quest'ultima impatta sulla membrana sensibile che si riscalda o si raffredda, questa variazione di temperatura verrà misurata da un apposito sensore che la convertirà in un valore elettrico che potrà successivamente essere elaborato.

Una termopila consiste principalmente in una serie di termocoppie unite alla membrana sensibile. Ogni singola termocoppia produce bassi valori di tensione (una decina di microVolt per grado centigrado di gradiente di temperatura tra giunzione calda e fredda della termocoppia), per questo se ne mettono molteplici per aumentare il fattore di conversione del calore in tensione. La tensione che si viene a generare grazie a questa serie di termocoppie è data dalla **(1.8)** [6]**:**

$$V_{out}=\alpha\Delta T \quad (1.8)$$

Dove $\alpha\Delta T$ è un fattore dovuto alla presenza delle molteplici termocoppie. Questo valore di tensione naturalmente sarà processato tramite un opportuno circuito di condizionamento (per esempio in **Figura 1.7** è presente lo schema a blocchi del MLX90614).

Nel caso invece di sensori come l'AMG8833, che restuisticono una matrice di valori, il funzionamento è analogo solamente che i sensori thermopile sono molteplici e alloggiati in una matrice 8x8. Questo permette di aumentare la risoluzione a 64 punti di misura, a differenza del MLX90614 che ha una risoluzione di un solo punto. In **Figura [4] 1.6** un esempio di rilevazione di radiazione termica.

1.2.6 Esempio pratico di funzionamento sensore IR: MLX90614

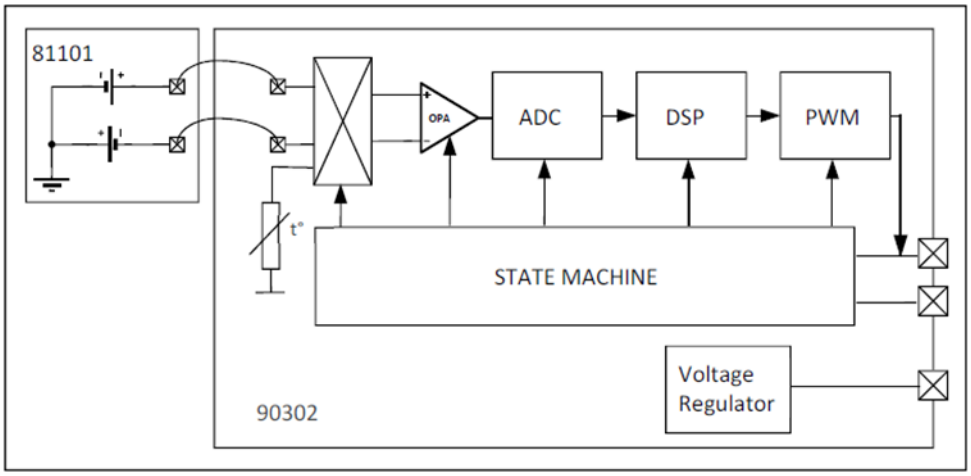


Figura 1.7: Schema a blocchi MLX90614.

[4] Figura 1.6 tratta da https://www.researchgate.net/figure/Schematic-diagram-of-thermal-infrared-radiation-detection-principle-According-to-Cai_fig1_363805858

Un esempio pratico di come funziona un sensore IR può essere fornito dal sensore **MLX90614** stesso. Esso è formato da due chip: il sensore ad infrarosso **MLX81101** e il chip DSP **ASSP 90302** (**Figura[5] 1.7**).

La macchina a stati interna è il cuore del **MLX90614**: calcola la temperatura (basandosi sull'output del sensore 81101) e invia i dati sui relativi bus di comunicazione.

L'output del 81101 viene amplificato da un amplificatore chopper a basso rumore, per poi essere convertito da un modulatore Sigma Delta in un flusso unico di bit; questo flusso viene dato in ingresso al DSP per il processamento digitale. Dei filtri FIR e IIR eliminano il rumore fuori banda, cercando di ridurre la banda del segnale a quella utile; il risultato del filtraggio è disponibile in una RAM interna [7].

Entrambe le temperature (ambientale e oggetto) calcolate hanno una risoluzione di 0.01 gradi centigradi, permettendo cosi misure relativamente precise.

1.2.7 Sensori RTD e termoresistori

Il terzo sensore, il DS18B20 è l'unico tra i tre che non sfrutta la tecnologia infrarosso, essendo un termistore. In questo paragrafo si descriveranno altre due tipologie di sensoristica sfruttata per la misura di temperatura: i sensori RTD e i termoresistori (NTC e PTC).

I sensori RTD (*Resistive Temperature Detectors*) basano il loro funzionamento sulla dipendenza della resistenza elettrica di alcuni metalli con la temperatura (per esempio il platino). I sensori RTD hanno coefficienti di temperatura positivi: all'aumentare della temperatura, aumenta la resistenza elettrica. La resistenza elettrica R_T, che si trova ad una certa temperatura T, può essere calcola tramite la relazione **(1.9)** (per temperature che vanno da 0 a 630° C) [6]:

$$R_t=R_0(1+AT+BT^2) \quad (1.9)$$

Dove R_0 è la resistenza del metallo ad una temperatura di riferimento, le costanti A, B dipendono dal metallo (la maggior parte delle volte il platino) che costituisce il sensore. Si utilizza specialmente il platino grazie alle sue caratteristiche di stabilità e durabilità: si ricorda infatti che il platino è un metallo nobile e quindi resistente a corrosione e ossidazione. Se le temperature raggiungono valori superiori a 600° C viene utilizzato il tungsteno.

I termistori, invece, si suddividono in PTC (*Positive Temperature Coefficient*) e NTC (*Negative Temperature Coefficient*); i primi presentano una resistenza elettrica che aumenta con la temperatura, i secondi una resistenza elettrica che diminuisce all'aumentare della temperatura. Sono generalmente costruiti con metalli semiconduttori (germanio o silicio).

[5] Figura 1.7 tratta da https://www.sparkfun.com/datasheets/Sensors/Temperature/MLX90614_rev001.pdf

La funzione di trasferimento di un termistore risulta più complessa rispetto a quella di un sensore RTD: essa è non lineare e dipende molto dal sensore stesso. Un'equazione polinomiale che permette di esprimere il comportamento di un termistore è la seguente **(1.10)** [6]:

$$\ln(R_t) = A_0 + \frac{A_1}{T} + \frac{A_2}{T^2} + \frac{A_3}{T^3} \quad (1.10)$$

Dalla (1.10) sono state tratte tre funzioni di trasferimento: il cosiddetto modello semplice, il modello di Fraden e il modello di Steinhart e Hart.

Il modello semplice è il più immediato: si tratta di semplificare la (1.10) e riscriverla come la **(1.11)** [6]**:**

$$\ln(R_t) = A + \frac{\beta}{T} \quad (1.11)$$

Beta è una costante che dipende dal materiale con cui è costruito il termoresistore, espressa in gradi Kelvin; A è una costante adimensionale.

Un effetto da non trascurare nei termistori è l'auto riscaldamento: quando la corrente scorre tramite il sensore, per effetto Joule, esso si riscalderà portando a errori di misura essendo il sensore più caldo dell'ambiente circostante.

Dopo un certo tempo denominato τ_T, il sensore si porterà da una temperatura pari a quella ambientale T_a ad una temperatura superiore Ts.

Il $\Delta T = T_s - T_a$ si può calcolare come riportato in **(1.12)** [6]**:**

$$\Delta T = \frac{P}{\delta}\left(1 - e^{-\frac{\delta}{C}t}\right) \quad (1.12)$$

Dove P è la potenza del generatore che alimenta il sensore, δ è il fattore di dissipazione e C la capacità termica del termoresistore. Un modo per ridurre, e al limite annullare, l'auto riscaldamento è quello di usare un generatore a bassa potenza e alte resistenze di limitazione, in modo tale da rendere lo scarto di temperatura quasi trascurabile (in **Figura[6] 1.8** un semplice esempio di uso di un NTC).

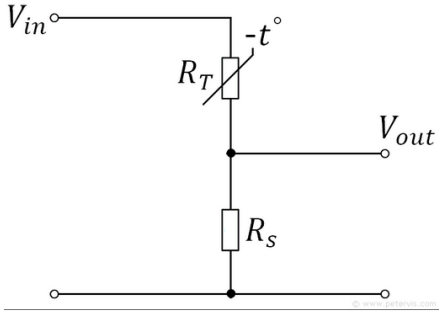


Figura 1.8: Partitore resistivo di misura con NTC.

[6] Figura 1.8 tratta da https://www.petervis.com/electronics%20guides/calculators/thermistor/thermistor.html

1.3 Schede a microcontrollore e I^2C

In questa sezione si descriverà in modo generale cosa è una scheda a microcontrollore e di come possa essere interfacciata con un dispositivo esterno come un sensore, per esempio, tramite una delle principali modalità di comunicazione seriale sincrona: I^2C.

1.3.1 Microcontrollore

Un **microcontrollore** è un microcalcolatore integrato su un singolo chip; al suo interno è presente tutto l'hardware necessario per poter funzionare: memorie RAM, ROM ed EEPROM, GPIO, TIMER, convertitori analogico digitali e le periferiche di comunicazione come I^2C o SPI.

Seppur a livello prestazionale una CPU supera di gran lunga un microcontrollore, essi sono utili quando si lavora in tutte quelle situazioni dove non si ha bisogno di grandi potenze di calcolo. Trovano largo utilizzo nei sistemi elettronici di uso dedicato: piccoli dispositivi come per esempio termometri digitali, frigoriferi, lavatrici. In **Figura[7] 1.9** un esempio di microcontrollore: **PIC16F887**.

Figura 1.9: PIC16F887.

Questi dispositivi possono essere programmati tramite un linguaggio ad alto livello come per esempio il *C*, oppure tramite un linguaggio a basso livello come *l'Assembly*.

Una scheda a microcontrollore agevola il lavoro del programmatore avendo montato a bordo elementi che agevolano lo sviluppo del firmware stesso.

Ci sono vari esempi di schede a microcontrollore: Arduino, Raspberry Pi Pico, STM32. Non bisogna incorrere però nell'errore di considerare il microcontrollore a bordo con il nome della scheda: nel caso di Arduino non è esso stesso il microcontrollore ma l'ATmega328P; Arduino è solamente il nome della scheda di sviluppo.

[7] Figura 1.8 tratta da https://www.microchip.com/en-us/product/PIC16F887

1.3.2 Comunicazione

Due sensori su tre che sono adoperati nel progetto sfruttano una comunicazione seriale sincrona denominata I^2C. Prima di tutto bisogna chiarire cosa si intende con comunicazione nel senso più generale del termine. Ogni qualvolta avviene un trasferimento dati tra due o più periferiche sta avvenendo una comunicazione; che possa essere tra due PC, tra un PC e un microcontrollore o anche tra due microcontrollori stessi il discorso non cambia.

Ciò che viene comunicato sono *byte* e la comunicazione può avvenire in due modi: seriale e parallelo. La comunicazione seriale ha il vantaggio di avere un solo bus di comunicazione con meno fili tra le periferiche con il prezzo però di poter inviare un solo bit alla volta per ogni ciclo di clock. Il parallelo dualmente dispone di un filo per ogni bit, per esempio, comunicando tra due dispositivi a 8-bit disporremo di otto fili che trasportano ognuno un bit; ciò lo rende molto più veloce di una comunicazione seriale.

Ma anche qui c'è un prezzo da pagare: all'aumentare della complessità dell'architettura aumentano di pari passo i numeri di fili con cui si comunica; basti pensare ad un sistema a 64-bit. In questo paragrafo si parlerà di seriale principalmente, essendo l'I^2C un protocollo che implementa questa modalità di comunicazione.

Si è detto protocollo poiché l'I^2C è proprio questo: essendo la comunicazione seriale più complessa della parallela, inviando un bit alla volta e non un byte contemporaneamente come nella parallela, si ha il problema di dover ricostruire in modo corretto la parola al ricevitore. Tramite i protocolli di comunicazione, che possono essere I^2C ma anche altri come SPI, si hanno degli standard con cui si riesce a ricostruire in ricezione ciò che è stato inviato su unico filo.

1.3.3 Protocollo I^2C

Un bus di questo tipo è formato da due linee: SDA, SCL. Sulla prima viaggeranno i dati, sulla seconda il clock che sincronizza i dispositivi in comunicazione tra loro. Infatti, possono essere collegati più di due dispositivi: un dispositivo assumerà il ruolo di master ed uno o più il ruolo di slave. Il master è colui che emetterà il segnale di clock e quindi assume il comando della comunicazione, lo slave si sincronizzerà su tale clock.

Non bisogna commettere l'errore di pensare che lo slave possa solo ricevere: uno slave può anche trasmettere al master, ma lo farà sempre rispettando il clock che il master impone. Dal punto di vista circuitale un collegamento tipico dell'I2C è quello mostrato in **Figura[8] 1.10**.

[8] Figura 1.10 tratta da https://learn.adafruit.com/working-with-i2c-devices/pull-up-resistors

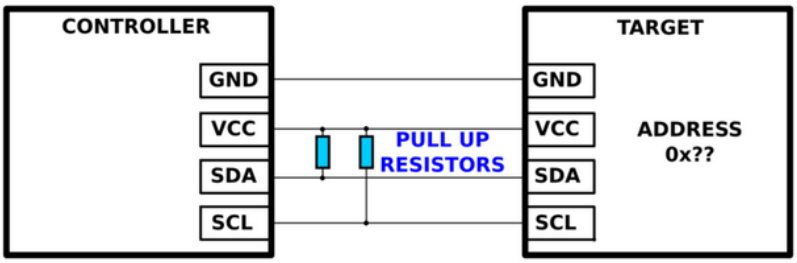


Figura 1.10: collegamento tipico I²C.

La presenza dei resistori di pull-up serve a fornire alla linea uno livello logico stabile (**Vcc**) nello stato di IDLE (ovvero quando la linea è inattiva). L' I²C ha sette bit per l'indirizzamento e quindi 128 dispositivi collegabili, anche detti nodi, su unico bus. Bisogna precisare che sedici di questi indirizzi sono riservati, quindi sono 112 quelli realmente usabili.

1.3.4 Temporizzazione dell'I²C

Quando avviene un trasferimento dati ciò che avviene può essere riassunto in questo modo: **Start condition**, nella quale la linea **SDA** passa da uno stato alto (IDLE) ad uno stato basso mentre il clock (**SCL**) è tenuto alto.

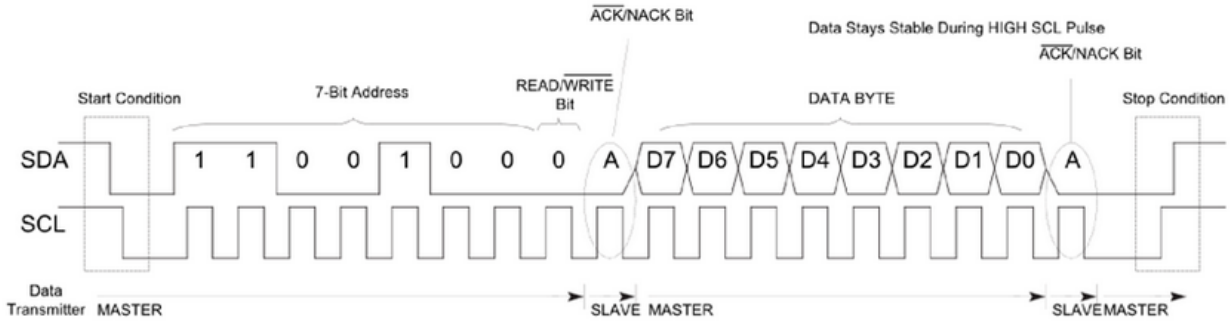


Figura 1.11: Temporizzazione I²C.

Il trasferimento termina con una **Stop condition** nella quale SDA passa da uno stato basso ad uno alto, mentre SCL viene mantenuta sempre alta.

Tra le due condizioni i dati vengono inviati in pacchetti di bytes. I dati vengono inviati lungo la linea SDA solamente quando SCL è basso; quando SCL=HIGH la linea SDA deve rimanere stabile in modo tale da poter essere letta dal ricevente.

Se quindi si ha per esempio un master che vuole trasmettere un bit=1, le azioni che dovrà fare sono: SCL=0; SDA=1 successivamente SCL=1 ovvero i dati sono inviati al ricevente e di nuovo SCL=0 per impostare il nuovo bit sulla linea SDA.

Dopo aver un inviato l'intero byte viene inviato il bit di **acknowledge** il quale può essere un ACK o NACK. Se ACK (1) vuol dire che la trasmissione è andata a buon fine, se NACK (0) c'è stato qualche errore in trasmissione.

Il primo pacchetto ad essere inviato è quello contenente l'indirizzo (7-bit) del nodo a cui si vuole comunicare; l'ottavo bit del byte è di lettura/scrittura. Permette di capire se si vuole leggere il contenuto del nodo indirizzato o scriverci. Successivamente il ricevitore invia un bit di **acknowledge**, forzando bassa la linea SDA. Ciò permette al dispositivo che trasmette di capire che il pacchetto di byte contenente l'indirizzo è arrivato correttamente al ricevitore. Successivamente inizia la trasmissione vera e propria dei pacchetti di byte nelle modalità descritte sopra.

In **Figura[9] 1.11** è possibile visualizzare la temporizzazione della linea SDA e SCL.

[9] Figura 1.11 tratta da https://www.analog.com/en/technical-articles/i2c-primer-what-is-i2c-part-1.html

Capitolo 2
Analisi del materiale e sviluppo del codice

2.1 I sensori adoperati

Scopo di questo progetto di tirocinio è quello di effettuare misure simultanee di temperature tramite l'uso di tre sensori che presentano tecnologie differenti: si hanno due sensori infrarosso e un sensore di contatto digitale. L'eterogeneità dei tre ha l'obiettivo proprio di mettere in risalto le differenze che possono presentarsi nella misurazione di temperatura.

Prima di iniziare le varie prove di misura e lo sviluppo del codice, è stato eseguito un lavoro preliminare: infatti oltre ai tre sensori, i quali sono *l'AMG8833*, *MLX90614* e il *DS18B20*, sono state fornite tre schede di sviluppo: *Arduino UNO, Arduino Portenta* e *Raspberry Pi Pico*. Questa prima parte del lavoro ha riguardato proprio lo studio dei datasheet e delle caratteristiche di schede e sensori al fine di poter valutare quale tra le tre schede fornite potesse avere le migliori caratteristiche per il fine del progetto.

2.1.1 Adafruit AMG8833 IR Thermal Camera FeatherWing

L'AMG8833, sensore infrarosso prodotto dalla Panasonic [8], restituisce una matrice 8x8 di valori di temperatura (che corrispondo ad una risoluzione di 64 pixels).

La serie 8833 ha una tensione operativa di 3.3 V. Come detto precedentemente, il sensore è montato su una scheda (una breakout board) della Adafruit che comunica tramite I^2C. Purtroppo, in rete non si trova documentazione specifica della scheda in questione; infatti, anche consultando la scheda prodotto sul sito dell'azienda [9] si viene rimandati ad una documentazione di una scheda che monta sempre lo stesso sensore Panasonic e che riporta anche la stessa nomenclatura AMG8833.

Seppur le due schede riportano lo stesso nome e montano lo stesso sensore si hanno delle differenze. Naturalmente, come si nota in **Figura[10] 2.1**, presentano due pinot differenti seppur i pin principali siano quattro e i restanti siano per agevolare il montaggio su breadboard. Infatti, i pin utili sono **SDA**, **SCL** e i due pin di alimentazione **3.3V** e **GND** che sono presenti su entrambe le schede.

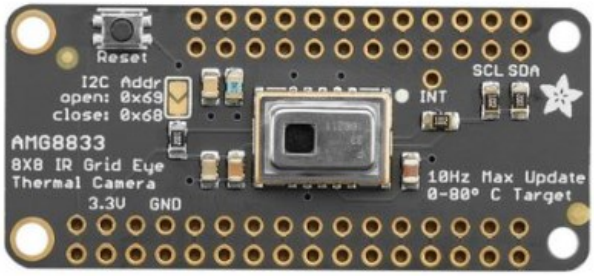


Figura 2.1: Scheda in dotazione.

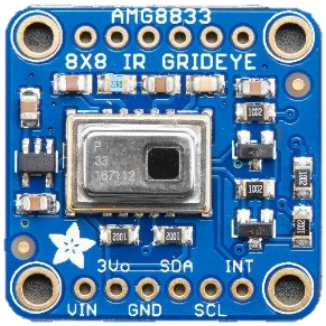


Figura 2.2: Scheda presente nella documentazione.

In **Figura[11] 2.2** si può notare un pin denominato 3Vo da dove è possibile prevelare una tensione di 3.3 V da un regolatore di tensione montato sulla scheda. Infatti, come detto precedentemente, il sensore lavora a 3.3 V, il regolatore permette quindi di poter usare anche tensioni fino a 5 V che verranno convertite alla tensione di lavoro del sensore. Sulla scheda a disposizione, quella presente nell'immagine di sinistra da un'indagine visiva sembra non sia presente un regolatore di tensione e quindi bisogna prestare attenzione quando si collega il sensore ad un dispositivo esterno come, per esempio, un Arduino UNO il quale presenta le uscite I^2C a 5 V.

In questo caso, per evitare di rovinare la componentistica, bisogna interporre tra scheda e un sensore un logic level converter, in modo tale da adattare i 5 V di Arduino ai 3.3V del sensore.

2.1.2 MLX90614 DCC IR Array

Il secondo sensore in dotazione è un **MLX90614** [7] anch'esso infrarosso, ma che a differenza del precedente restituisce una misura puntuale di temperatura. A bordo della breakout board è montata una termocamera che fornisce un campo di ripresa (FOV) di trentacinque gradi.

[10] Figura 2.1 tratta da https://www.robot-italy.com/it/adafruit-amg8833-ir-thermal-camera-featherwing.html
[11] Figura 2.2 tratta da https://learn.adafruit.com/adafruit-amg8833-8x8-thermal-camera-sensor/overview

Il sensore può lavorare sia a 5 V che a 3.3 V disponendo internamente di un livellatore di tensione. In **Figura[12] 2.3** è possibile vedere come si presenta la scheda.

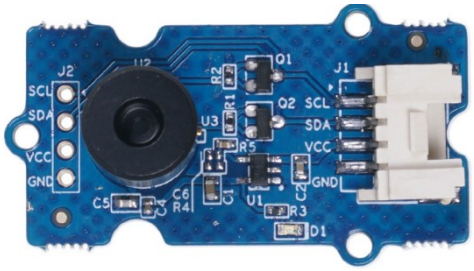


Figura 2.3: MLX90614.

Il pinout è molto semplice, sono presenti solamente quattro pin: le due linee dell'I^2C e i due pin di alimentazione.

Consultando il datasheet [7], è possibile leggere come il sensore venga già fornito calibrato, e può lavorare in due differenti modalità: 40°C-125°C per misure ambientali e -70°C-380°C per misure di oggetti.

Come detto precedentemente la differenza sostanziale tra questo sensore e l'AMG8833 è nella modalità in cui viene restituita la misura: nel caso del MLX90614 viene fornito un valore puntuale di temperatura; nel caso dell'AMG8833 viene restituita una matrice 8x8 di valori di temperatura e ciò permette quindi di avere una misura di temperatura in un'area coperta di 64 pixels. Il funzionamento dei sensori si basa sempre su quello di un sensore infrarosso generale, che si è già avuto modo di descrivere nel **paragrafo 1.2.5**.

2.1.3 Sensore di temperatura DS18B20

Questo sensore è presente sul mercato sotto due forme: uno in un package TO92, comunemente usato per i transistor, e un secondo come sonda impermeabilizzata da un cavo in PVC. Fornisce valore di temperatura tra 9 e 12 bit in gradi Celsius; la risoluzione può essere impostata in fase di sviluppo codice tramite le funzioni fornite dalla libreria apposita del sensore; inoltre supporta sia tensioni a 5 V che 3.3 V.

Presenta tre cavi: rosso per l'alimentazione, nero per la massa e il giallo per l'uscita, come si può notare in **Figura[13] 2.4**.

[12] Figura 2.3 tratta da https://www.kiwi-electronics.com/en/grove-single-point-infrared-thermometer-mlx90614-dcc-with-35-fov-10768
[13] Figura 2.4 tratta da https://www.robotstore.it/Sensore-di-temperatura-waterproof-DS18B20

Figura 2.4: Sensore DS18B20.

L'uscita è un bus dati di tipo 1-wire, che andrà collegato su uno qualunque dei pin digitali del microcontrollore in uso. Per effettuare il collegamento si avrà bisogno di un resistore da 4.7 kΩ tra il pin di alimentazione e di comunicazione; servirà come resistore di pull-up.

2.1.4 Protocollo 1-Wire

L'ultimo sensore descritto, il **DS18B20** è un sensore digitale che comunica tramite un unico cavo. Esso sfrutta il protocollo seriale 1-Wire sviluppato dalla **Dallas Semiconductor**. Come si può vedere dalla **Figura**[14] **2.5** si ha un dispositivo Master che controlla uno o più dispositivi Slave collegati tramite un unico filo.

Ogni dispositivo Slave presenta un indirizzo a 64-bit che permette di essere identificato dal Master in fase di comunicazione [10].

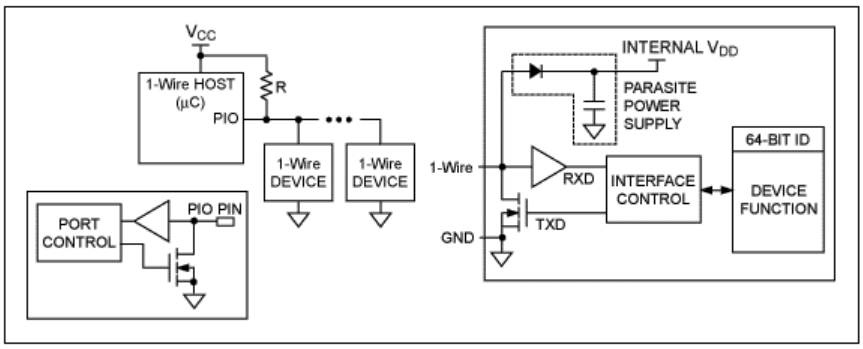


Figura 2.5: Protocollo 1-Wire.

[14] Figura 2.5 tratta da https://www.analog.com/en/technical-articles/guide-to-1wire-communication.html

È presente un resistore di pull-up che permette di impostare un livello di tensione durante lo stato IDLE della linea; la comunicazione è tipo di half-duplex; i due dispositivi possono lavorare sia come ricevitori che come trasmettitori ma uno alla volta [10].

Come si può vedere dalla Figura 2.5, il più delle volte, i dispositivi collegati lungo la linea 1-Wire non hanno bisogno di alimentazione esterna: essi sfruttano la capacità parassita (PARASITE POWER SUPPLY) per alimentarsi.

2.2 Le schede a microcontrollore

Come detto nella sezione 2.1, il lavoro preliminare è consistito nello scegliere, tra le tre schede a microcontrollore, quella che avrebbe potuto meglio soddisfare l'obiettivo di avere misure simultanee ed elaborazioni dati riguardanti la temperatura.

La scelta è caduta sul Raspberry Pi Pico, il quale tra le tre, presenta le migliori caratteristiche hardware. Di seguito si riporta una descrizione generale delle tre schede.

2.2.1 Arduino UNO

L'Arduino UNO (**Figura[15] 2.6**) è una scheda di prototipazione estremamente diffusa. La scheda si basa sul microcontrollore **ATmega328P,** prodotto dalla Atmel. Presenta un'architettura Harvard modificata a 8-bit con un set di istruzioni di tipo RISC.

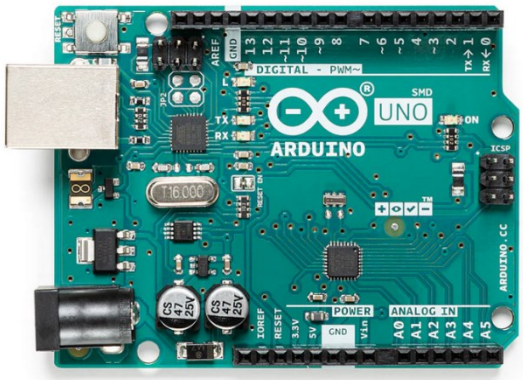


Figura 2.6: Arduino UNO Rev3 SMD in dotazione.

[15] Figura 2.6 tratta da https://store.arduino.cc/products/arduino-uno-rev3-smd

La scheda presenta 14 pin I/O digitali tra cui 6 configurabili come PWM e 6 input analogici. In tabella 2.1 si riportano alcune informazioni (le più utili) sulla scheda, tratte direttamente dal sito stesso della casa produttrice [11].

Si è evitato di riportare informazioni come peso o lunghezza non essendo strettamente necessari ai fini di questo progetto.

In Tabella 2.1 [11] le principali caratteristiche tecniche della scheda:

Microcontrollore	ATmega328P
Tensione di lavoro	5V
Pin digitali	14 (6 usabili come PWM)
Input analogici	6
Corrente pin digitali	20 mA
GND	3 pin
I2C	1x
Corrente per pin a 3.3V	50 mA
Velocità di clock	16 MHz

Tabella 2.1: Caratteristiche Arduino UNO.

La programmazione dell'Arduino UNO (linguaggio simil C) avviene principalmente sfruttando il software fornito dalla casa stessa [12].

In **Figura 2.7** uno screenshot dell'IDE.

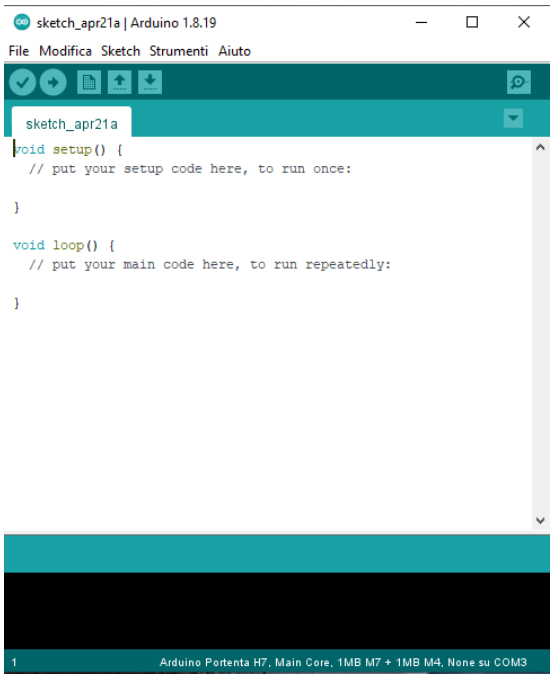


Figura 2.7: IDE per programmare Arduino.

2.2.2 Arduino Portenta H7

Rispetto al precedente, il Portenta presenta una struttura hardware più articolata. Dotato di due cores paralleli, il sistema si basa sul microcontrollore **STM32H747** che include un Cortex M7 a 480 MHz e un Cortex M4 a 240 MHz, entrambi su architettura ARM a 32-bit.

A differenza dell'Arduino UNO, il Portenta si inserisce in una fascia più professionale e per usi industriali. Come per il precedente, la programmazione avviene sempre tramite l'IDE andando a selezionare la scheda Portenta H7 nel menù apposito di selezione schede.

In **Figura**[16] **2.8** il Portenta H7 in dotazione.

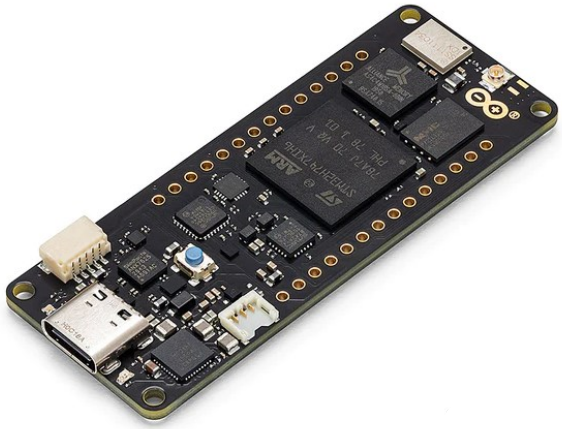


Figura 2.8: Portenta H7.

In **Tabella 2.2**, si riportano le caratteristiche principali della scheda.

Microcontrollore	STM32H747
Tensione di lavoro	3.3 V
Pin digitali	84 (10 usabili come PWM)
Input analogici	8
Corrente pin digitali	8 mA
GND	2pin
I2C	3x
Velocità di clock	M7: 480 MHz, M4: 240 MHz

Tabella 2.2: Caratteristiche generali Portenta H7.

[16] Figura 2.8 tratta da https://store.arduino.cc/products/portenta-h7

2.2.3 Raspberry Pi Pico

A differenze delle due schede a microcontrollore precedentemente descritte, quest'ultima è prodotta dalla Raspberry Pi.

Microcontrollore dalle dimensioni ridotte (21mm x 51mm), la scheda si basa sul RP2040 costituito da un Cortex M0+ a 32-bit.

Presenta 26 pin GPIO, di cui tre collegati ad un convertitore ADC presente sulla scheda e quindi utilizzabili come input analogici. In più presenta un sensore di temperatura già integrato. In Tabella 2.3 vengono riassunte le principali caratteristiche.

Microcontrollore	RP2040 (Cortex M0+)
GPIO	26 di cui 3 settabili come analogici
Input Voltage	1.8 -5.5 V DC
Tensione di lavoro	3.3 V
I2C	2x
GND	8 pin

Tabella 2.3 Caratteristiche principali del Pico.

La programmazione può avvenire in C/C++ o MicroPython, utilizzando diversi ambienti di sviluppo. In questo progetto si è scelto di usare l'ambiente Mu [13].

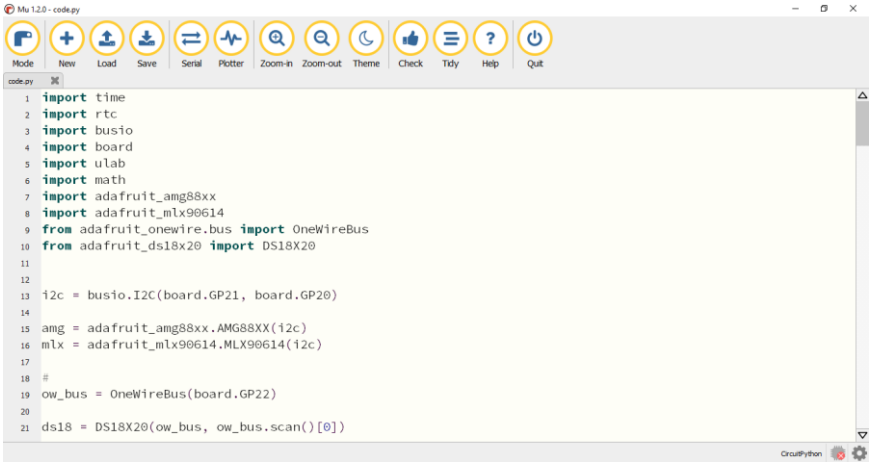


Figura 2.9: Ambiente Mu 1.2.0.

In **Figura 2.9** è possibile osservare l'ambiente di programmazione che si è sfruttato nel progetto. La programmazione del codice è avvenuta sfruttando il linguaggio CircuitPython, un linguaggio simile al classico MicroPython usabile sulle schede Raspberry.

2.3 Scelta del microcontrollore

Il punto più importante di questa parte preliminare, oltre allo studio dei vari sensori e di come essi debbano essere interfacciati, è stato quello di capire quale scheda a microcontrollore potesse soddisfare al meglio l'obiettivo finale.

Per capire ciò, si sono testati, sia singolarmente, che contemporaneamente i tre sensori con ognuna delle tre schede facendo girare semplici programmi che facessero acquisire la temperatura. Per eseguire questi test preliminari si sono usati i codici forniti direttamente dalle librerie stesse dei sensori; l'obiettivo principale in questa fase era quello di capire come interfacciare le periferiche. Il codice vero e proprio è stato sviluppato successivamente.

2.3.1 Test Arduino UNO

Si è partiti dapprima testando i tre sensori singolarmente per capire al meglio il loro funzionamento e come andassero interfacciati con il microcontrollore. Per ogni sensore si è importata la libreria necessaria all'interfacciamento: per l'MLX90614 si è usata la libreria fornita dalla Adafruit: ***Adafruit_MLX90614.h***, così come per l'AMG8833 si è usata la libreria ***Adafruit_AMG88xx.h***. Per il sensore di contatto, il DS18B20, si è fatto uso di due librerie: ***DallasTemperature*** e la ***OneWire***.

Non c'è stato bisogno di scaricare dalla rete alcuna libreria poiché si è fatto uso del gestore librerie presente nell'IDE di Arduino. Come detto in precedenza, per l'Arduino UNO e il sensore AMG8833 si è fatto uso di un logic level converter per livellare i livelli di tensione del microcontrollore a 3.3 V in modo tale da interfacciarlo con il sensore. I tre codici sono perlopiù quelli base forniti dalla libreria come esempio.

Figura 2.10: Esempio pixel_test.

Una volta installate le librerie necessarie per ognuno dei tre sensori, è possibile trovare degli esempi nella cartella esempi sotto la voce *File*: in **Figura 2.10** l'esempio usato per testare l'AMG8833 (*pixel_test*).

Invece in **Figura 2.11** è mostrato l'output fornito in seriale dal sensore MLX90614.

```
Ambient = 20.55*C       Object = 24.81*C
Ambient = 68.99*F       Object = 76.66*F

Ambient = 20.55*C       Object = 19.07*C
Ambient = 68.99*F       Object = 66.33*F

Ambient = 20.59*C       Object = 21.21*C
Ambient = 69.06*F       Object = 70.18*F
```

Figura 2.11: Lettura valori MLX90614.

Tutti e tre sensori, infatti, forniscono un output sul monitor seriale dell'IDE.

Dopo aver testato i tre sensori singolarmente, è stato condotto un test con tutti e tre contemporaneamente (si sono semplicemente uniti i tre programmi di esempio). In **Figura 2.12** una foto dei collegamenti dell'Arduino con i tre sensori.

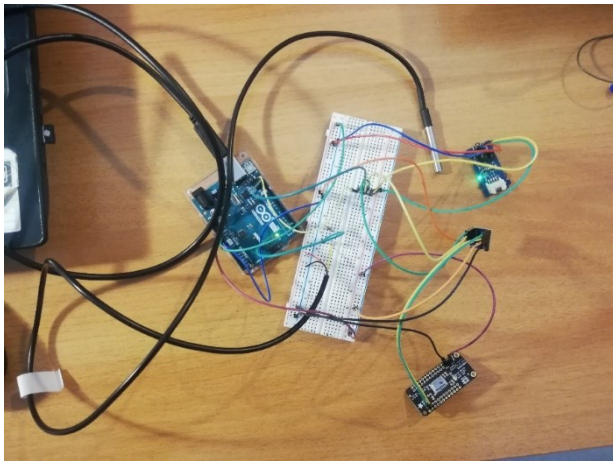


Figura 2.12: Arduino e i vari sensori.

2.3.2 Test Arduino Portenta H7

Purtroppo, la fase di test con il Portenta non è andata come ci si aspettava. Se la scheda non ha dato problemi nell'interfacciarsi con il sensore AMG8833, con il sensore MLX90614 non riusciva a fornire misure esatte di temperatura.

Una volta collegato il Portenta e l'MLX90614 insieme, sul monitor seriale veniva stampato un output dove veniva fornita la temperatura di 1037 gradi centigradi costanti, come se il sensore rilevasse una temperatura tale da mandarlo in overflow.

Si sono provate varie soluzioni: cambiare la velocità dell'I^2C, impostare manualmente l'indirizzo del sensore, fornire un'alimentazione esterna costante ma nulla ha risolto il problema. Le cause potrebbero essere dovute ad un problema di comunicazione seriale tra il sensore e il Portenta.

Si esclude naturalmente l'ipotesi che il sensore non funzioni, non avendo riscontrato problemi in fase di test con l'Arduino UNO e il Raspberry Pi Pico.

2.3.3 Test Raspberry Pi Pico

Per la programmazione del Pico si è scelto di installare il CircuitPython [14], linguaggio simile al MicroPython. Il CircuitPython è ottimo per scopi educativi avendo a disposizione molte librerie utili nell'interfacciamento con periferiche esterne.

Il Pico viene fornito con un drive denominato *RP1/RP2*. Per installare il CircuitPython bisognerà andare in modalità bootloader (tenendo premuto il pulsante sul Pico stesso). Una volta fatto ciò basterà trascinare il file CircuitPython precedentemente scaricato sul drive del Pico e aspettare che il nuovo drive venga installato con il nome ***CIRCUITPY***.

Come già accennato precedentemente, per la programmazione in CircuitPython si farà uso dell'IDE Mu, ma nulla esclude la possibilità di utilizzare altri ambienti di programmazione. In **Figura[17] 2.13** l'installazione del nuovo drive.

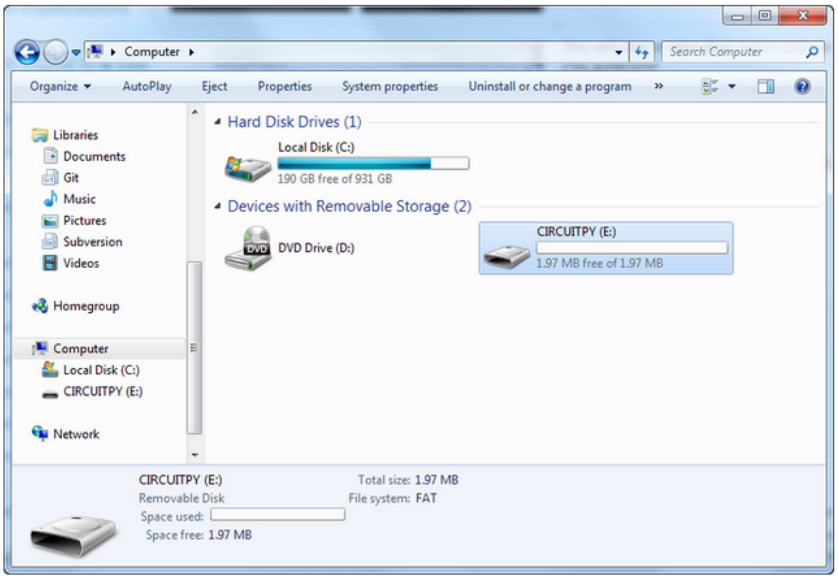


Figura 2.13: Drive del Pico una volta installato il CircuitPython.

[17] Figura 2.13 tratta da https://learn.adafruit.com/welcome-to-circuitpython/installing-circuitpython

Come per la fase di test dell'Arduino Uno, con il Pico si è scelto la stessa modalità: dapprima si sono testati i tre sensori singolarmente, successivamente in contemporanea. Tutti i moduli necessari all'interfacciamento sono stati direttamente scaricati da un bundle fornito dalla Adafruit [15].

In **Figura 2.14** si riporta una lettura seriale dei tre sensori insieme

```
C:18.69
Ambient = 19.59*C       Object = 18.49*C
Ambient = 67.26*F       Object = 65.28*F

[19.25, 18.50, 18.50, 19.00, 18.00, 18.25, 18.75, 18.50,
18.25, 18.50, 18.25, 18.75, 18.25, 18.50, 18.50, 19.00,
18.25, 18.25, 19.00, 19.00, 18.25, 18.00, 18.25, 18.75,
18.00, 18.00, 18.75, 18.75, 18.00, 18.00, 18.50, 18.50,
18.25, 17.75, 18.50, 19.00, 18.75, 18.25, 18.00, 18.75,
18.50, 18.00, 18.75, 18.25, 18.00, 18.75, 18.25, 18.25,
17.50, 18.00, 17.50, 18.50, 18.00, 18.25, 18.50, 18.75,
17.75, 18.50, 18.25, 18.50, 18.00, 18.25, 18.75, 18.50,
]
```

Figura 2.14: Lettura rispettivamente partendo dall'alto del DS18B20-MLX90614-AMG8833.

2.3.4 Scelta della scheda

Fase finale di questo lavoro preventivo è stata la scelta della scheda. Naturalmente per il problema rilevatosi con il sensore MLX90614, la scheda Portenta è stata esclusa limitando la scelta tra l'Arduino UNO e la Raspberry Pi Pico.

Come si può vedere in **Tabella 2.4** il Pico batte l'Arduino per caratteristiche hardware; c'era da aspettarselo essendo la prima molto più recente rispetto la seconda. Inoltre, con il Pico non si ha bisogno di un logic level converter per l'interfacciamento con l'AMG8833: questo semplifica e riduce l'ingombro dei vari collegamenti.

Naturalmente la scelta poteva ricadere anche sull'Arduino UNO; vien da sé però che il prezzo competitivo, il linguaggio più intuitivo e le caratteristiche hardware migliori hanno spinto verso il Pico, portando alla scelta di quest'ultimo.

	Raspberry Pi Pico	Arduino
Tensione di lavoro	3.3 V	5V
Pin	26 x GPIO (di cui 3 analogici) 2 x I2C 2 x UART 2 x SPI	14 pin digitali 6 Pin analogici 1 x I2C 1 x SPI 6 x canali PWM
Linguaggio	C/C++ MicroPython	Java, C, C++
Prezzo	Circa 4 euro	Circa 25 euro
Processore	Chip RP2040 con due core ARM Cortex-M0+ a 48MHz espandibili fino a 133MHz	Chip Atmel modello ATMega328 a 16 MHz
Memoria	264KB SRAM 2MB memoria Flash	2KB RAM 32KB memoria Flash 1KB memoria EEPROM

Tabella 2.4: Confronto tra le due schede.

2.4 Codice per l'acquisizione della temperatura

Parte centrale del progetto è lo sviluppo del codice; esso principalmente dovrà: acquisire i valori di temperatura dei tre sensori, eseguire calcoli statistici (deviazione standard campionaria) e permettere al Pico di lavorare come data logger. Come già accennato, bisognerà prima installare le varie librerie (all'interno della cartella *lib* del CIRCUITPY) che permetteranno l'interfacciamento con i tre sensori.

La parte iniziale del programma serve ad importare i vari moduli che serviranno nella stesura del codice: nella pagina successiva, in **Figura 2.15** si riportano le prime righe di codice.

```
import time
import rtc
import busio
import board
import ulab
import math
import adafruit_amg88xx
import adafruit_mlx90614
from adafruit_onewire.bus import OneWireBus
from adafruit_ds18x20 import DS18X20
```

Figura 2.15: Importazione dei moduli utili.

Successivamente vi è la parte di inizializzazione delle varie variabili ed oggetti che serviranno in fase di progetto (**Figura 2.16**)

```
11
12 i2c = busio.I2C(board.GP21, board.GP20)
13 amg = adafruit_amg88xx.AMG88XX(i2c)
14 mlx = adafruit_mlx90614.MLX90614(i2c)
15 ow_bus = OneWireBus(board.GP22)
16 ds18 = DS18X20(ow_bus, ow_bus.scan()[0])
17
18 valoriAMG=[]
19 conta=0
20 colonna=0
21 sottrazione=0
22 quadrato=0
23 risultati=[]
24 S=0
25 ####
26 r = rtc.RTC()
27 r.datetime = time.struct_time((2023, 4, 30, 10, 53, 00, 0, -1, -1))
28 ########
```

Figura 2.16: Inizializzazione oggetti e variabili.

L'istruzione in riga 12 crea l'oggetto ***i2c***: esso permette di usare le due linee SDA ed SCL del Pico: GP21 (SCL) e GP20 (SDA). Le righe 13-14 servono a creare i relativi oggetti ***amg***, ***mlx****:* tramite essi si potrà accedere ai vari metodi della classe del relativo sensore. La riga 15 imposta il bus OneWire sul pin GP22 (dove è collegato il DS18B20); analogamente alle righe 13-14, la riga 16 crea l'oggetto ***ds18***.

Le restanti riga servono a creare le varie variabili che serviranno nel loop principale: le righe 26-27 servono per la stampa dei vari timestamp.

Essi servono per inizializzare il modulo RTC: Real Time Clock che si userà per poter scrivere sul file i vari timestamp.

Purtroppo, la data e l'ora da cui far partire il conteggio devono essere settati manualmente via codice.

Il Pico non dispone di un orologio di sistema che riesce a mantenersi sincronizzato anche dopo aver tolto l'alimentazione della scheda: ogni qualvolta si volesse far ripartire il conteggio dopo aver tolto l'alimentazione il Pico partirebbe dalla data relativa al 1-01-2020. Per ovviare a questo inconveniente occorre settare manualmente la data e l'ora di inizio a cui sincronizzare l'orologio interno: in questa maniera una volta avviato il programma, il tempo scorrerà a partire dai parametri temporali impostati

Il loop principale è racchiuso all'interno di un comando *try-except:* infatti il loop principale girerà non appena il Pico verrà collegato come data logger. Nel caso si colleghi il Pico in modalità standard verrà gestito il codice presente all'interno del except.

```
29 try:
30
31     with open("/temperature.txt", "a") as datalog:
32
33
34         while True:
35             if conta==0:
36                 #Titolo formattazione testo
37                 datalog.write('Sensore  Temperatura    Ora   Deviazione Campionaria\n')
38                 conta=1
39
40             x=ulab.numpy.array(amg.pixels,dtype=ulab.numpy.float)
41
42             #calcolo media matrice amg.pixels
43             media=ulab.numpy.mean(x)
44             datalog.write('AMG ')
45             datalog.write('      {0:0.2f} '.format(round(media,2)))
46
```

Figura 2.17: Righe 29-45.

Collegato il Pico come data logger (successivamente verrà spiegato come), viene aperto un file denominato *temperature.txt* in modalità append; esso verrà aperto sul drive stesso del Pico (CIRCUITPY).

L'intero programma viene eseguito in loop all'interno del ciclo iterativo *while*. Il ciclo if presente in riga 35 serve a stampare sul file aperto i titoli delle colonne dove andranno scritti i vari parametri. Naturalmente esso deve essere eseguito solamente una volta, da qui il controllo tramite ciclo condizionale. In riga 40 è presente un'istruzione del modulo *ulab*: il sensore AMG8833, come già detto, restituisce la matrice 8x8 di valori come *amg.pixels*.

Questa matrice però viene ritornata come stringa: tramite l'istruzione in riga 35 viene convertita in una matrice di valori float e ne viene calcolato il valore medio (riga 43).

Successivamente, si stampa sotto la colonna ***Sensori*** la tipologia di sensore e sotto la colonna ***Temperatura*** la media arrotondata a due cifre decimali calcolata precedentemente.

```
48             #acquisione tempo matrice
49             current_time = r.datetime
50             ora=current_time.tm_hour
51             minuti=current_time.tm_min
52             secondi=current_time.tm_sec
53
54             datalog.write('      {hour}:'.format(hour=ora))
55             datalog.write('{minu}:'.format(minu=minuti))
56             datalog.write('{sec} '.format(sec=secondi))
57
```

Figura 2.18: Scrittura del timestamp.

Tramite l'istruzione *r.datetime* è possibile scrivere il timestamp relativo all' acquisizione del sensore AMG8833: vengono prelevati l'ora, i minuti e i secondi e successivamente vengono stampati in modo tale da trovarsi sotto la colonna ***Ora***.

Scrivere sul file i timestamp è utile a capire quanto tempo passi tra la lettura di un sensore ed un'altra: infatti, se i due sensori AMG8833 e MLX90614 sono quasi istantanei nella lettura, il DS18B20 è un sensore di contatto e quindi più lento nell'acquisizione. Sul relativo datasheet si può leggere come, impostando una risoluzione di 12-bit, il sensore abbiamo bisogno di 750 *ms* per acquisire il valore.

```
64          for row in amg.pixels:
65              for temp in row:
66                  #trasformo in float elemento matrice che è una stringa
67                  a=float(temp)
68                  valoriAMG.append(a)
69                  sottrazione=a-media
70                  quadrato=math.pow(sottrazione,2)
71                  risultati.append(quadrato)
72
73          somma=sum(risultati,0)
74          N=63
75          div=somma/N
76          S=math.sqrt(div)
77          risultati.clear()
78          valoriAMG.clear()
79          datalog.write('       {0:0.2f}\n'.format(S))
```

Figura 2.19: Calcolo della deviazione standard campionaria.

Le righe 64-79 calcolano e stampano il valore della deviazione standard campionaria. Si inizia con un doppio ciclo *for* per poter scorrere all'interno della matrice *amg.pixels*.

Come già detto, gli elementi della matrice vengono forniti come valori stringa, per questo si convertono in valori float per poter eseguire su di essi i dovuti calcoli matematici. Successivamente si implementa la formula (**Figura[18] 2.20**) e si scrive il valore incolonnandolo sotto la colonna **Deviazione campionaria**.

Come si può notare è simile al calcolo della deviazione standard (**paragrafo 1.1.4**) ma normalizzata a N-1 campioni.

$$S=\sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1}}$$

Figura 2.20: Deviazione standard campionaria.

[18] Figura tratta da https://paolapozzolo.it/deviazione-standard/

La parte restante del codice esegue le stesse operazioni già illustrate per l'AMG8833, escluso il calcolo della deviazione standard campionaria.

```
81          datalog.write('MLX ')
82
83          datalog.write('      {0:0.2f} '.format(round(mlx.object_temperature,2)))
84          #timestamp MLX
85
86          current_time = r.datetime
87          ora=current_time.tm_hour
88          minuti=current_time.tm_min
89          secondi=current_time.tm_sec
90
91          datalog.write('      {hour}:'.format(hour=ora))
92          datalog.write('{minu}:'.format(minu=minuti))
93          datalog.write('{sec}\n'.format(sec=secondi))
```

Figura 2.21: Calcolo temperatura e timestamp per MLX90614.

Le righe 81-93 stampano la temperatura acquisita dal sensore MLX90614 e l'ora di tale acquisizione. Analogamente le righe 95-105 per il DS18B20 (**Figura 2.22)**.

```
95          datalog.write('DS18X20 ')
96          datalog.write('  {0:0.2f}'.format(round(ds18.temperature,2)))
97          #timestamp DS18X20
98          current_time = r.datetime
99          ora=current_time.tm_hour
100         minuti=current_time.tm_min
101         secondi=current_time.tm_sec
102
103         datalog.write('       {hour}:'.format(hour=ora))
104         datalog.write('{minu}:'.format(minu=minuti))
105         datalog.write('{sec}\n'.format(sec=secondi))
106
107         ########Fine lettura#######
108
109         datalog.flush();
110         time.sleep(3)
```

Figura 2.22: Calcolo temperatura e timestamp per DS18B20.

Tramite il *time.sleep(3)* si ritarda di tre secondi un'acquisizione (complessiva dei tre sensori) rispetto un'altra.

La restante parte di codice è quella gestita dall'*except*: in caso di filesystem non scrivibile oppure se pieno si viene rimandati in queste righe di codice. In questo caso si è solo impostata una variabile delay nel caso si volesse far lampeggiare un eventuale led di segnalazione.

```
118 except OSError as e:  #Filesystem non scrivibile
119     delay = 0.5
120     if e.args[0] == 28:  # Filesystem pieno
121         delay = 0.25
122
```

Figura 2.23: Parte finale del codice.

2.4.1 File testuale

Come già detto, una volta collegato il Pico come data logger, inizierà l'acquisizione dati e la relativa scrittura sul file testuale aperto via codice sul drive CIRCUITPY. Su come far funzionare il Pico come data logger si tornerà più avanti, in **Figura 2.24** si può vedere un esempio di acquisizione dati.

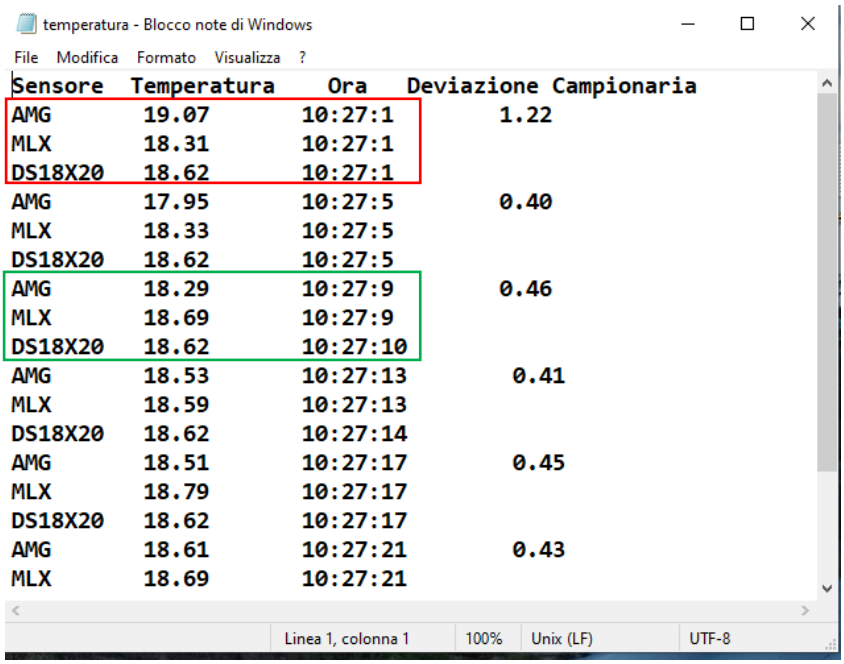


Figura 2.24: File testuale contenente i dati.

La scrittura sul file è avvenuta in modo tale da seguire una formattazione ben precisa, incolonnando ogni valore sotto la relativa colonna; in questo modo si può importare il file testuale in Excel o MATLAB per eseguire le relative analisi e grafici.

Si può notare come, per il DS18B20, a volte l'acquisizione è simultanea ai due sensori infrarosso (riquadro in rosso) altre volte ritarda di un secondo (riquadro in verde).

C'era da aspettarselo, essendo un sensore di contatto e lavorando alla massima risoluzione (12-bit), impiega circa 750 *ms* per poter acquisire il valore, a differenza dei sensori infrarosso che sono molto più veloci.

2.4.2 Raspberry Pi Pico come data logger

Come si è avuto modo di leggere, si è usato il Pico come data logger, ovvero come un collezionatore di dati; in questo caso valor, di temperatura e i relativi istanti di acquisizione.

Per poter usare il Pico come data logger [16], bisogna prima di tutto caricare sul drive un file denominato *boot.py*.

Infatti, la scheda non può agire sul filesystem del CIRCUITPY contemporaneamente al computer: i file presenti sul drive possono essere modificati o solo dal PC (banalmente quando si modifica e si carica un codice) oppure solamente dal Pico (quando si apre il file dove caricare i valori di temperatura).

È possibile far lavorare il Pico come data logger in due modalità: sfruttando un collegamento esterno oppure modificando volta per volta il file *boot.py*. In questo caso si è scelta la prima modalità per la sua facilità di implementazione.

Si carica sul drive CIRCUITPY un file boot.py (**Figura 2.25**) che contiene una particolare istruzione che permette al Pico di agire sul filesystem solamente se il pin GP0 del Pico è collegato a massa. Questo breve codice può essere trovato all'indirizzo [16].

Come si nota in **Figura 2.25**, l'istruzione contenuta in riga 11 indica che il computer non può più modificare i file contenuti sul drive CIRCUITPY se il pin GP0 è collegato a massa. Per far tornare la gestione del file system al computer basterà staccare prima l'USB e successivamente il cavo che collega GP0 a GND e ricollegare il Pico senza questo collegamento, per tornare a modificare e caricare codice sul Pico.

```
1  import board
2  import digitalio
3  import storage
4
5  write_pin = digitalio.DigitalInOut(board.GP0)
6  write_pin.direction = digitalio.Direction.INPUT
7  write_pin.pull = digitalio.Pull.UP
8
9  # If write pin is connected to ground on start-up, CircuitPython can write to CIRCUITPY filesystem.
10 if not write_pin.value:
11     storage.remount("/", readonly=False)# Write your code here :-)
12
```

Figura 2.25: File boot.py.

Se si tentasse di modificare il file che si è creato, quando la scheda è collegata come data logger, si perderanno alcuni o tutti i file contenuti sul drive CIRCUITPY e si dovrà reinstallare CircuitPython da capo.

Bisognerà tenere premuto il tasto BOOTSEL del PICO e contemporaneamente collegare l'USB al PC. Si aprirà il drive pulito del Pico, e basterà reinstallare il CIRCUITPY seguendo le istruzioni già spiegate precedentemente (**paragrafo 2.3.3**).

Quindi, una volta caricato il file boot.py sul drive del Pico, basterà staccare e ricollegare la scheda con un jumper che collega GP0 a massa per poter raccogliere dati e scriverli su un eventuale file di testo; si ricorda che in questa modalità la gestione del filesystem del Pico è lasciata a esso stesso: modificare o caricare file porterebbe alla corruzione del drive del Pico.

La procedura può essere quindi riassunta in questo modo:

1. Si carica il programma con il Pico collegato normalmente e successivamente si stacca l'USB che collega Pico e PC
2. Si collega un jumper tra GP0 e massa, e lo si ricollega al PC
3. Si aspetta il tempo scelto per collezionare dati
4. Si stacca dapprima il collegamento USB, successivamente il jumper e si ricollega il Pico normalmente per poter aprire il file contenente i dati presente sul CIRCUITPY o per lavorare normalmente (scrivere e caricare codice).

In **Figura 2.26** un esempio di collegamento tra i sensori e il Pico e il jumper.

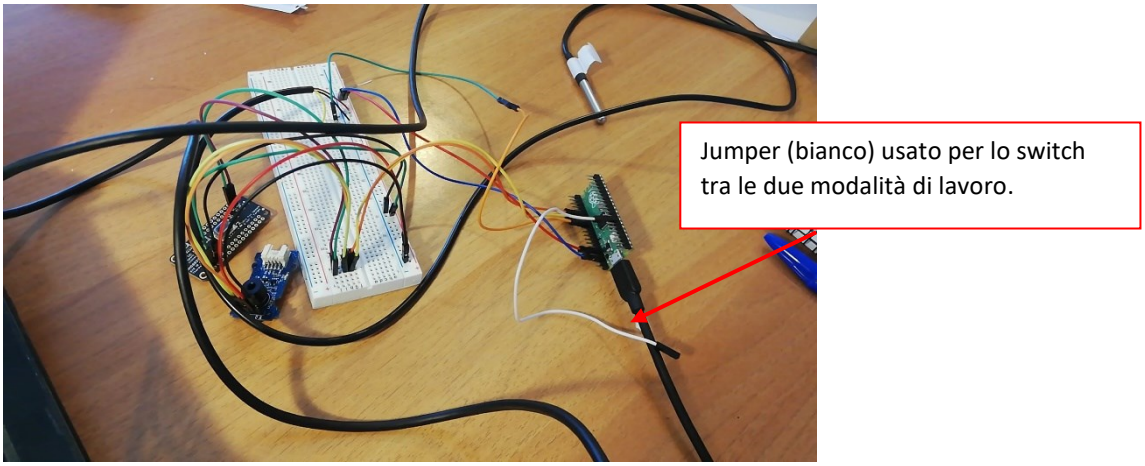


Figura 2.26: Pico con i vari sensori.

Capitolo 3
Prove sperimentali

3.1 Esperimento di misura con temperatura costante

Il primo esperimento svolto è stato quello di una prova di misura con una temperatura costante, nello specifico quella all'interno di una stanza. I risultati ottenuti sono stati successivamente processati tramite MATLAB.

3.1.1 Svolgimento della prova

La sessione è consistita in una prova di misurazione della durata di circa cinquanta minuti all'interno di una stanza; si è supposto che la temperatura all'interno di essa non subisca bruschi cambiamenti nell'arco di tempo della misurazione. Il codice utilizzato è quello che si è avuto modo di vedere nel **paragrafo 2.4**.

Dopo aver impostato la frequenza di lettura dei tre sensori a tre secondi, si è aspettato il tempo predisposto alla campagna di acquisizione. Successivamente si sono raccolti i valori presenti sul file testuale e si sono importati in MATLAB.

3.1.2 Realizzazione grafici tramite MATLAB

Il plot dei risultati è stato realizzato tramite MATLAB: nello specifico, il codice si occupa di realizzare i grafici temperatura-campioni ed una tabella che si avrà modo di descrivere successivamente.

Come già detto, i risultati delle misurazioni sono contenuti all'interno del file che si è aperto in fase di acquisizione dati **(Figura 3.1).** Dopo averli importati nel workspace di MATLAB (tramite il tool Import Data presente nella scheda Home), si è sviluppato il codice necessario al plot dei risultati (come si può vedere in **Figura 3.2**).

La colonna delle temperature è stata importata come un vettore colonna sul workspace (Temperatura) così come la deviazione standard campionaria calcolata per il sensore AMG8833.

In **Figura 3.3** si può visualizzare il workspace del programma.

Prova5maggio - Copia - Blocco note di Windows

File Modifica Formato Visualizza ?

Sensore	Temperatura	Ora	Deviazione Campionaria
AMG	19.06	11:45:1	1.30
MLX	18.91	11:45:1	
DS18X20	19.69	11:45:1	
AMG	18.32	11:45:5	1.37
MLX	18.69	11:45:5	
DS18X20	19.69	11:45:5	
AMG	18.67	11:45:9	1.29
MLX	18.49	11:45:9	
DS18X20	19.69	11:45:10	
AMG	25.07	11:45:13	3.36
MLX	18.81	11:45:13	
DS18X20	19.69	11:45:14	
AMG	19.07	11:45:17	1.76
MLX	18.63	11:45:17	
DS18X20	19.69	11:45:17	
AMG	19.18	11:45:21	1.80
MLX	18.87	11:45:21	
DS18X20	19.69	11:45:21	
AMG	19.07	11:45:25	1.71
MLX	18.59	11:45:25	
DS18X20	19.69	11:45:25	
AMG	19.00	11:45:29	1.54
MLX	18.81	11:45:29	
DS18X20	19.69	11:45:30	
AMG	19.00	11:45:33	1.46
MLX	18.69	11:45:33	
DS18X20	19.69	11:45:34	
AMG	18.81	11:45:37	1.54

Figura 3.1: File contenente le misurazioni.

```
1   %%Grafici Prova 5 Maggio
2   %Il vettore Temperatura(importato dalla colonna relativa del file testuale) contiene le temperature
3   %% AMG fino a 731 // MLX da 732 fino a 1462 // DS18B20 da 1463 fino a 2193
4 - AMG=Temperatura(1:731);
5 - Campioni=1:731;
6 - MLX=Temperatura(732:1462);
7 - DS=Temperatura(1463:2193);
8 - figure;plot(Campioni,AMG,Campioni,Deviazione);legend("Temperatura AMG","Deviazione STD campionaria");title("Temperatura -
9 - figure;plot(MLX);title("Andamento temperatura MLX90614");axis([1 800 16 26]);ylabel("Temperatura (°C)");xlabel("Campione")
10 - figure;plot(DS);title("Andamento temperatura DS18B20");axis([1 800 16 26]);ylabel("Temperatura (°C)");xlabel("Campione");
11  %%Plot tabella
12 - clc;
13 - Data=[mean(AMG) mean(MLX) mean(DS)
14        mean(Deviazione) std(MLX) std(DS)
15                         ]
16 - VarNames = {'AMG8833', 'MLX90614', 'DS18B20'};
17 - t=uitable(figure,'data',Data,'columnname',VarNames)
18 - t.RowName={'Media','DevStdC'}
19
```

Figura 3.2 Codice MATLAB per l'elaborazione dei risultati.

Il vettore *Temperatura* contiene l'intera colonna delle temperature misurate: si selezionano quindi le porzioni del vettore corrispondenti al sensore (riga 4-6-7).

Successivamente si graficano i risultati (riga 8-9-10).

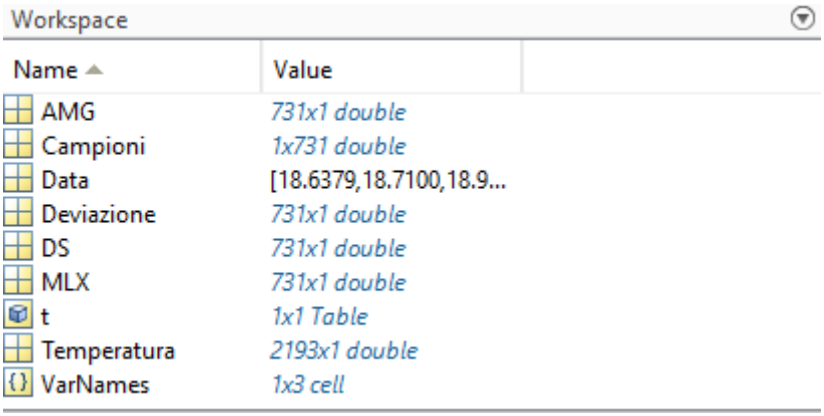


Figura 3.3: Workspace del programma.

In Figura 3.3 si possono visualizzare le variabili utilizzate dal programma tra cui anche il vettore *Deviazione*, il quale contiene le deviazioni calcolate per il sensore AMG8833.

3.1.3 Commento dei risultati ottenuti

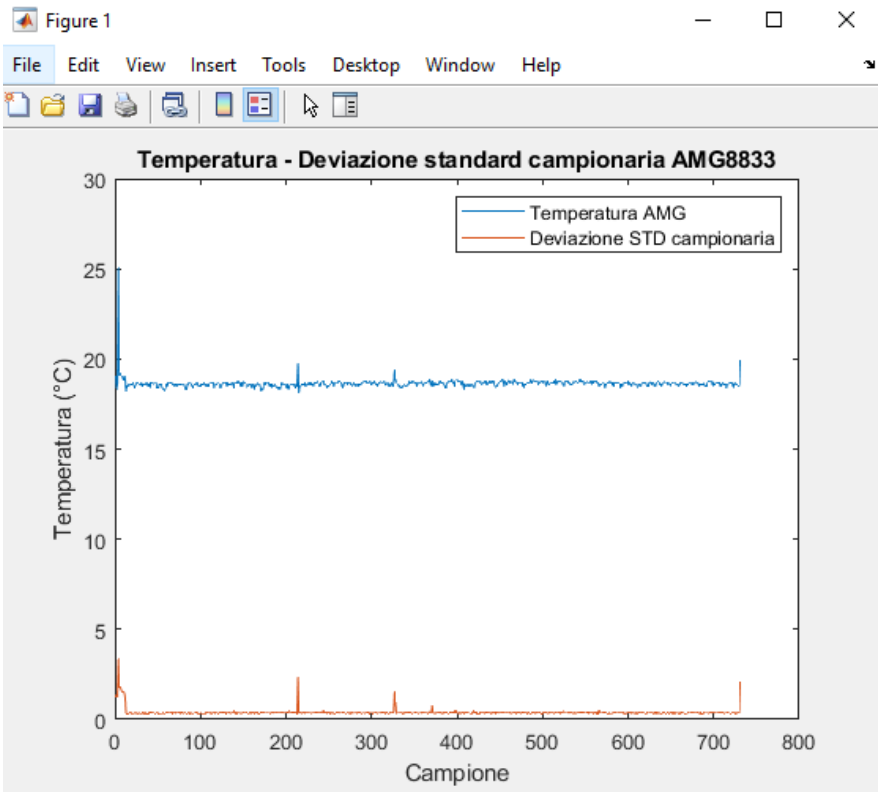


Figura 3.4: Temperatura misurata dal sensore AMG8833 e deviazione standard campionaria.

Come si può vedere in **Figura 3.4**, sono presenti due grafici: in blu l'andamento della temperatura del sensore AMG8833, in arancione quello della sua deviazione standard campionaria calcolata campione per campione, ovvero sugli insiemi di 64 punti di misura acquisiti dal sensore.

I campioni sono presenti sull'asse delle ascisse: essendo ogni acquisizione dati effettuata con un ritardo di tre secondi, tra un campione e il successivo passa un tempo di circa tre o quattro secondi (contando il ritardo intrinseco del sensore DS18B20).

Seppur nel complesso la temperatura si possa ritenere relativamente costante, sono presenti alcuni picchi. Il primo lo si può vedere nei primi istanti di misurazione: si può considerare la sua presenza dovuta all'avvio del sensore.

Gli altri due picchi si trovano in corrispondenza del campione 214 e del campione 327; il primo è presente anche nel caso del grafico del sensore MLX90614 (**Figura 3.5**) e corrisponde a circa 19.8 gradi centigradi. Si può pensare quindi a qualche cambiamento di temperatura all'interno della stanza; il secondo picco (19.4 gradi centigradi) risulta solamente nel grafico dell'AMG8833. Ciò può essere dovuto a del rumore elettronico che ha portato il sensore a rilevare un picco anomalo di temperatura.

Il secondo grafico riguarda l'andamento di temperatura del sensore MLX90614: come già detto si può notare un picco di circa 19.4°C a circa 214 campioni, così come esso era stato rilevato nell'andamento dell'AMG8833. Il grafico si mantiene in un range di temperatura relativamente ristretto.

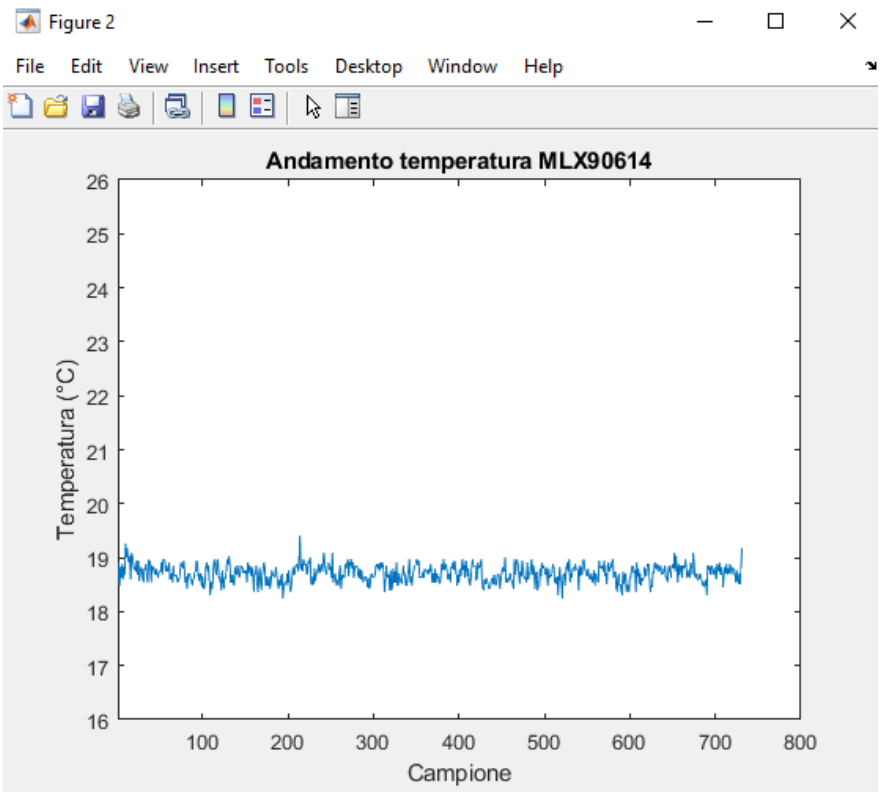


Figura 3.5: Temperatura misurata dal sensore MLX90614.

L’ultimo grafico, in **Figura 3.6**, riguarda il termoresistore DS18B20:

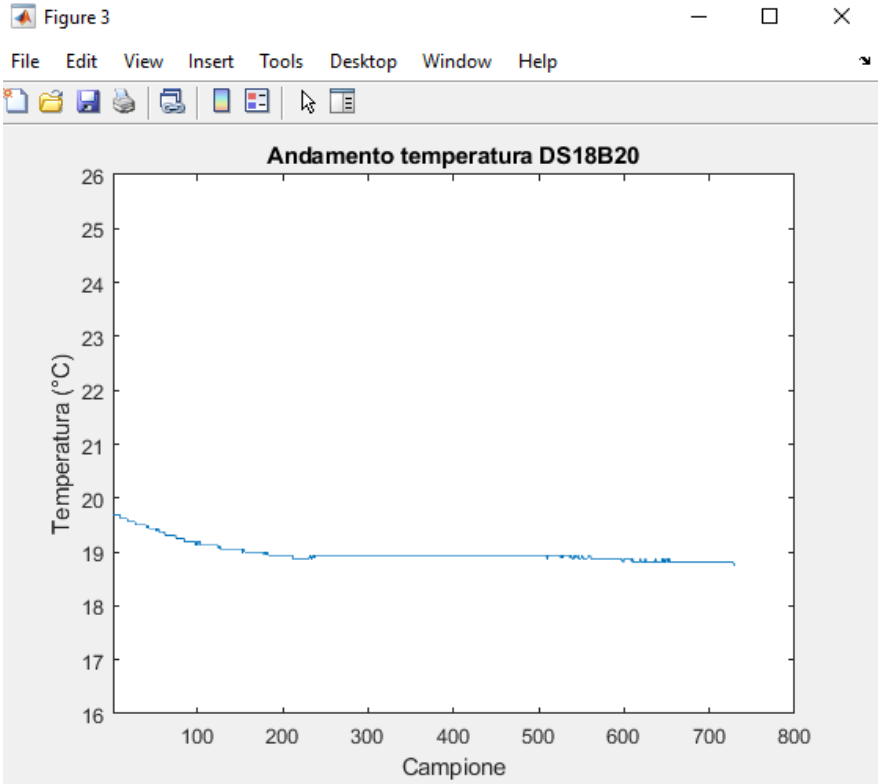


Figura 3.6: Temperatura misurata dal sensore DS18B20.

La differenza sostanziale rispetto ai due grafici precedentemente analizzati la si può notare in base a ciò che accade nei primi istanti di misura (tra il campione 0 e il campione 100).

Il DS18B20 è un sensore di contatto, quindi necessita di un certo tempo per portarsi all’equilibrio termico con l’ambiente circostante; in questo caso impiega 100 campioni per stabilizzarsi, corrispondenti ad un tempo di circa 300 secondi (5 minuti).

Tra il campione 100 e 200 la discesa del valore misurato di temperatura è meno brusca ed a tratti costanti; successivamente l’andamento si stabilizza come nel caso dei due sensori infrarosso.

Tornando al codice MATLAB, le righe 13-18 servono a creare la tabella presente in **Figura 3.7:**

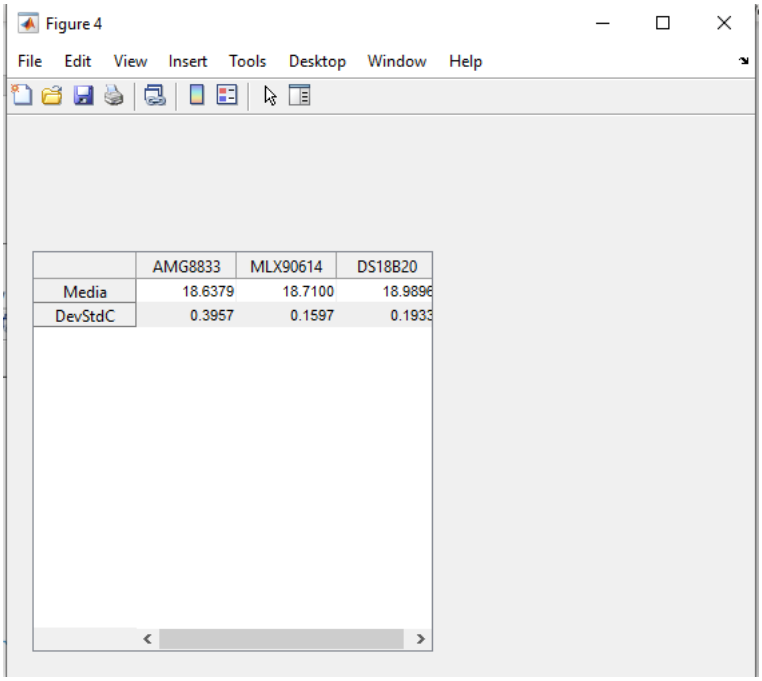


Figura 3.7: Indicatori statistici (Media- Deviazione standard campionaria).

Per quanto riguarda il sensore AMG8833, il programma calcola la media della matrice 8x8 e la fornisce sotto la colonna delle temperature: nella riga *Media* della tabella è presente la media di queste medie. Per gli altri due sensori è presente invece la media delle varie temperature puntuali rilevate.

Le tre medie tendono ad essere molto simili, anche per quanto riguarda le cifre decimali: essendo un test di temperatura ambientale (senza fattori esterni che la potessero modificare) difficilmente ci si poteva aspettare un risultato diverso. Per quando riguarda la deviazione standard campionaria, il sensore AMG8833 è quello che presenta il valore più grande: ciò permette di capire come i suoi valori tendano a discostarsi maggiormente dal valore medio di quanto facciano gli altri due sensori.

3.2 Esperimento di misura con temperatura variabile

Il secondo esperimento è consistito in una misura di temperatura variabile, anche in questo caso si riporteranno, così come per la prova precedente, i risultati ottenuti.

3.2.1 Svolgimento della prova

Scopo principale di questa prova è stato quello di analizzare il comportamento dei tre sensori durante la misura di una temperatura variabile. Nello specifico si è scelto di far bollire dell'acqua contenuta all'interno di una pentola e successivamente, come si può vedere in **Figura 3.8**, misurare la temperatura dell'acqua contenuta al suo interno.

Figura 3.8: Setup di misura.

Dopo aver aspettato circa un'ora di tempo, si sono ottenuti i risultati che si andranno a commentare nel prossimo paragrafo. Si evita di riportare il codice MATLAB, essendo quest'ultimo identico a quello già utilizzato per l'esperimento precedente (**Paragrafo 3.1).**

3.2.2 Commento dei risultati ottenuti

Il primo risultato che si andrà a commentare è quello ottenuto dal sensore AMG8833, di cui si può vedere il grafico dell'andamento di temperatura in **Figura 3.9** (pagina successiva).

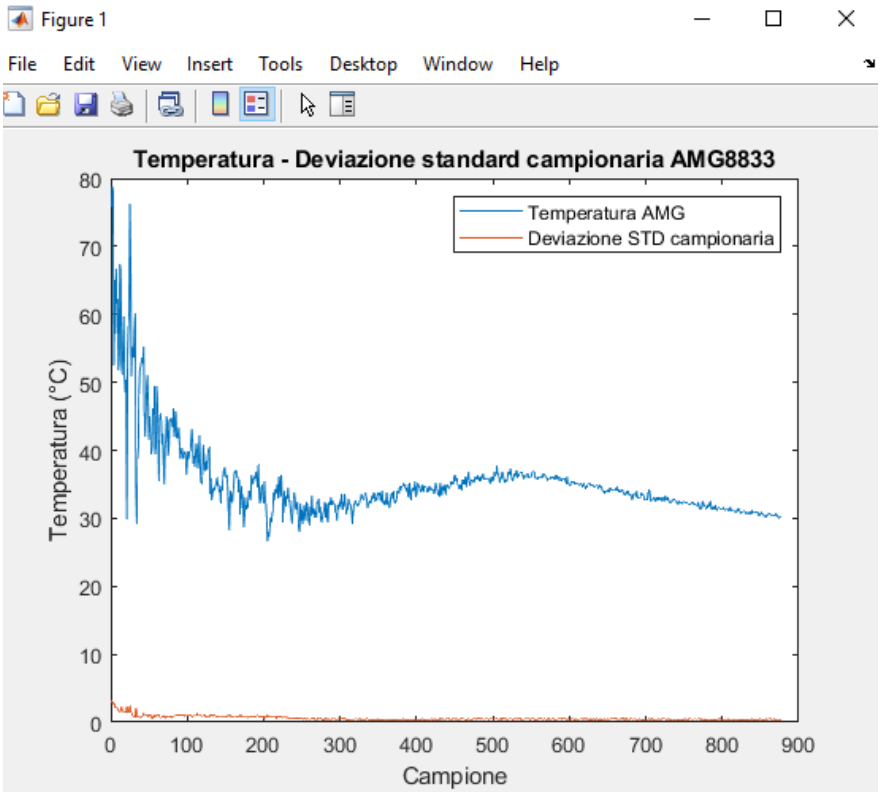


Figura 3.9: Temperatura misura dal sensore AMG8833 e deviazione standard campionaria.

Anche qui, si ha l'andamento della temperatura e della deviazione standard campionaria graficati insieme. Come si può vedere, è presente un picco di temperatura tra il campione 0 e il campione 100: nei primi istanti di raffreddamento della pentola, essa emanava del vapore che può aver fatto rilevare questo picco di temperatura andando ad alterare il funzionamento del sensore.

Inoltre, tra i campioni (circa) 450 e 600 è presente una risalita di temperatura: il vapore condensato sulla finestra del sensore molto probabilmente ha fatto rilevare un rialzo di temperatura anomalo. Per quanto riguarda invece il sensore MLX90614, l'andamento è discendente; analogamente per il DS18B20.

Quest'ultimo, tra i tre, è quello che presenta una curva di discesa "meno spigolosa": il motivo di ciò si può ricondurre al fatto che essendo direttamente a contatto con l'acqua e non avendo problemi dovuti all'ottica del sensore e del vapore acqueo, la misura risulta più risoluta e assume una forma che ricorda quello di un'esponenziale decrescente.

Nella pagina successiva sono riportati i grafici per il sensore MLX90614 e per il DS18B20.

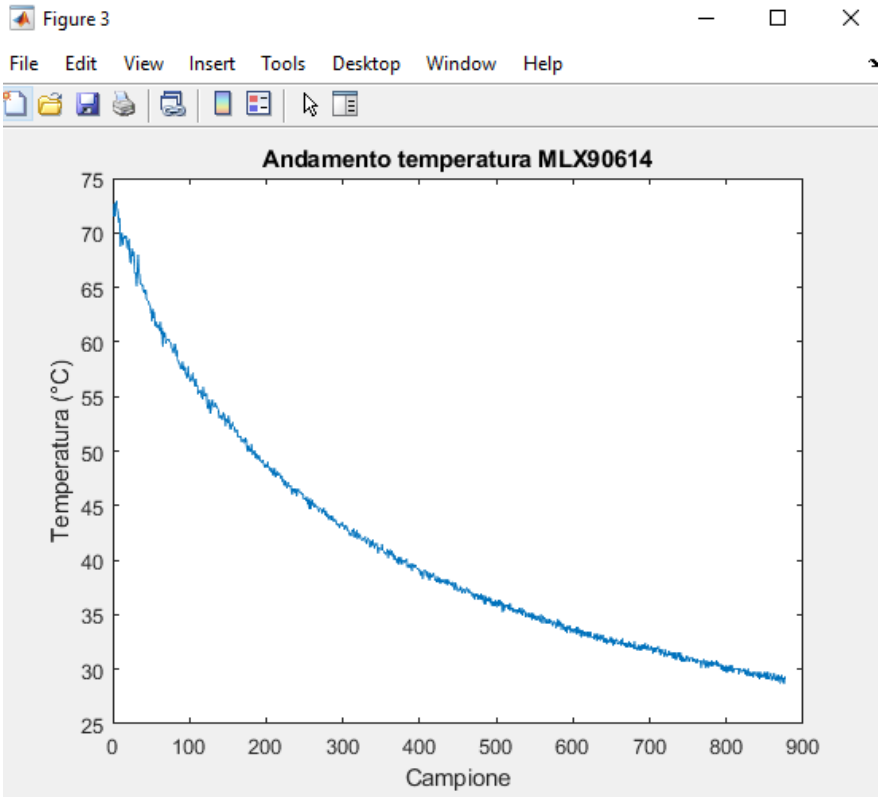


Figura 3.10: Andamento valori acquisiti con sensore MLX90614.

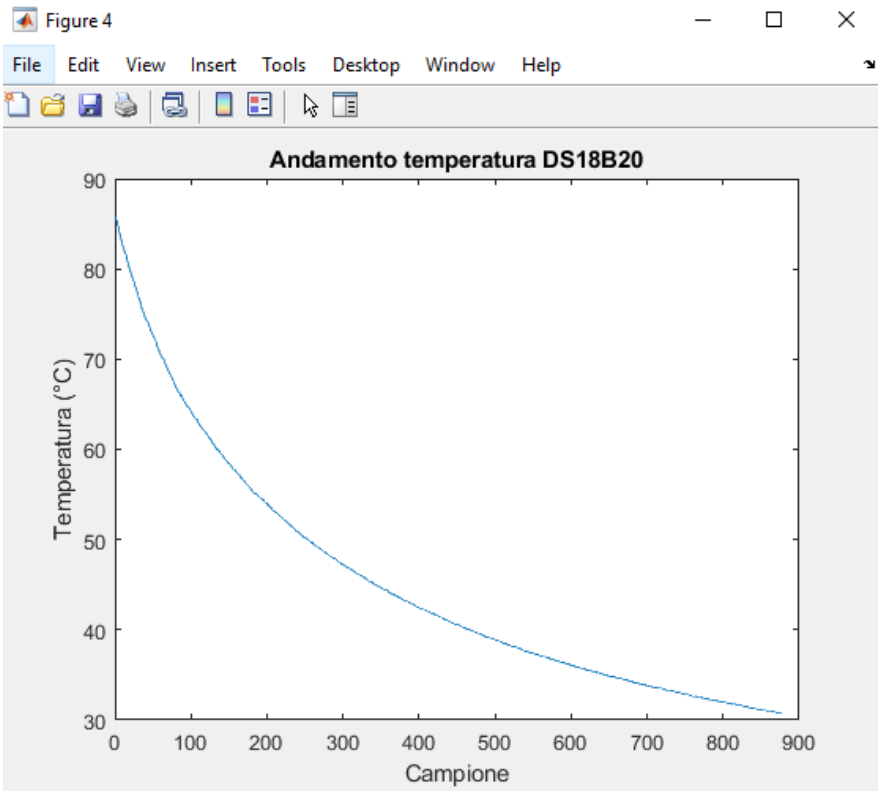


Figura 3.11: Andamento valori acquisiti con sensore DS18B20.

Infine, si riporta in **Figura 3.12** la tabella contenente i vari indicatori statistici, analogamente a quando fatto per la prova di misura a temperatura costante.

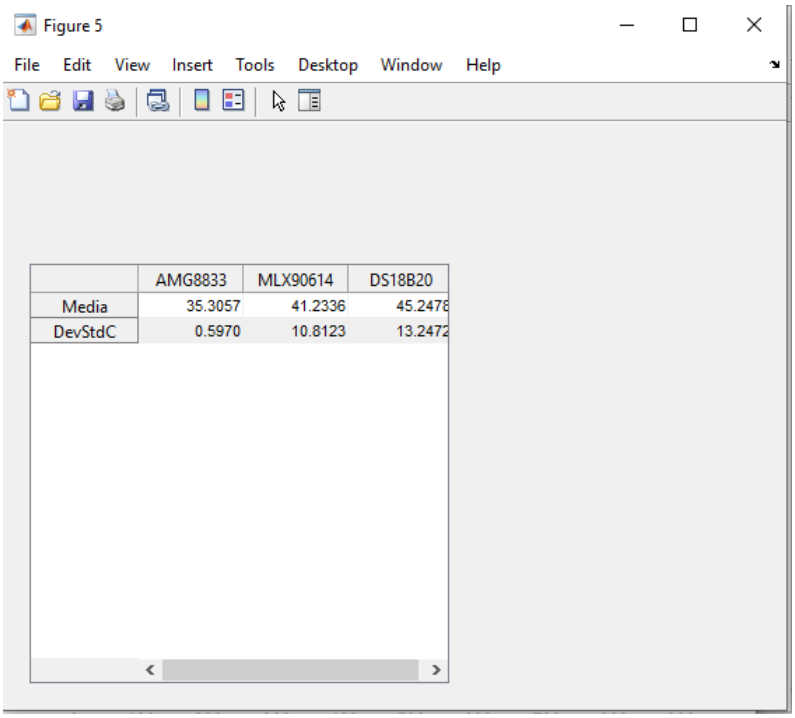


Figura 3.12: Tabella (Media-Deviazione standard campionaria).

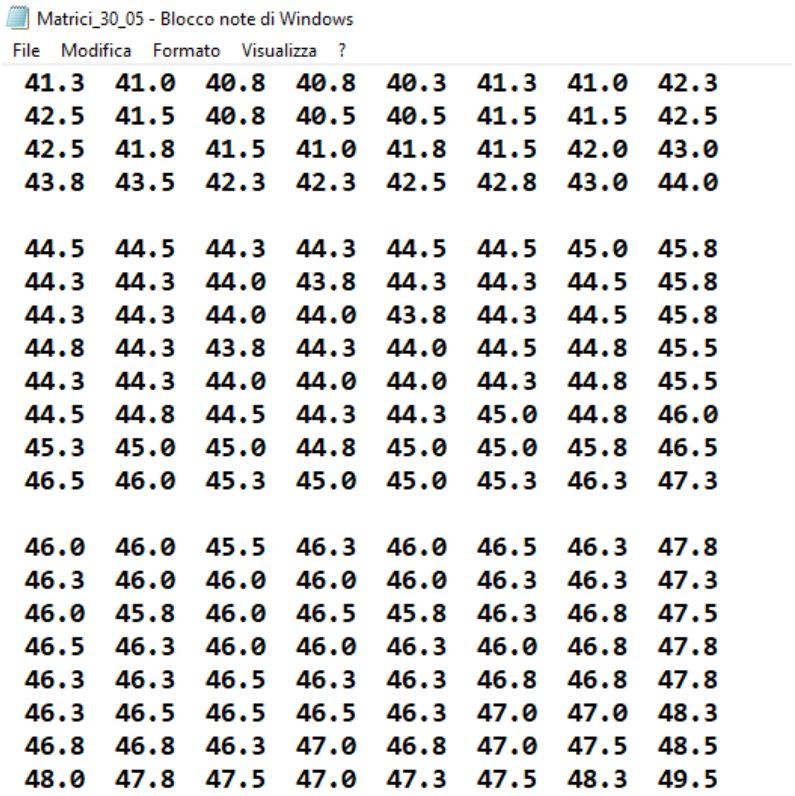
L'AMG8833 ha una media più bassa rispetto a quella degli altri due sensori, dovuta alla risalita di temperatura che si è già avuto modo di spiegare. La finestra ottica del sensore AMG8833 è relativamente più esposta al vapore a quella del MLX90614, che ha una lente protetta da un opportuno involucro e tende quindi a essere meno incline alla condensa del vapore acqueo.

3.3 Analisi delle matrici del sensore AMG8833

L'ultimo esperimento è stato uguale a quello descritto precedentemente (misura di temperatura variabile) con lo stesso set di misura. L'unica differenza è che in questo caso ci si è focalizzati sul sensore AMG8833 e si è abbreviata la durata dell'esperimento a venti minuti.

3.3.1 Svolgimento della prova

Come già accennato nel paragrafo precedente lo svolgimento della prova è stato identico al caso precedente, in questo caso però non si sono raccolte tutte e tre le misurazioni dei tre sensori ma solamente quelle del AMG8833. Inoltre, a differenza dei casi precedenti, si sono salvate sul file testuale non le medie delle matrici ottenute ma le matrici stesse. In **Figura 3.13** si possono vedere alcune matrici salvate durante la prova.


Matrici_30_05 - Blocco note di Windows

File Modifica Formato Visualizza ?

```
41.3  41.0  40.8  40.8  40.3  41.3  41.0  42.3
42.5  41.5  40.8  40.5  40.5  41.5  41.5  42.5
42.5  41.8  41.5  41.0  41.8  41.5  42.0  43.0
43.8  43.5  42.3  42.3  42.5  42.8  43.0  44.0

44.5  44.5  44.3  44.3  44.5  44.5  45.0  45.8
44.3  44.3  44.0  43.8  44.3  44.3  44.5  45.8
44.3  44.3  44.0  44.0  43.8  44.3  44.5  45.8
44.8  44.3  43.8  44.3  44.0  44.5  44.8  45.5
44.3  44.3  44.0  44.0  44.0  44.3  44.8  45.5
44.5  44.8  44.5  44.3  44.3  45.0  44.8  46.0
45.3  45.0  45.0  44.8  45.0  45.0  45.8  46.5
46.5  46.0  45.3  45.0  45.0  45.3  46.3  47.3

46.0  46.0  45.5  46.3  46.0  46.5  46.3  47.8
46.3  46.0  46.0  46.0  46.0  46.3  46.3  47.3
46.0  45.8  46.0  46.5  45.8  46.3  46.8  47.5
46.5  46.3  46.0  46.0  46.3  46.0  46.8  47.8
46.3  46.3  46.5  46.3  46.3  46.8  46.8  47.8
46.3  46.5  46.5  46.5  46.3  47.0  47.0  48.3
46.8  46.8  46.3  47.0  46.8  47.0  47.5  48.5
48.0  47.8  47.5  47.0  47.3  47.5  48.3  49.5
```

Figura 3.13: File contenente le misurazioni dell'AMG8833.

Alla fine dell'esperimento, si è importato il file testuale in MATLAB per le successive elaborazioni.

3.3.2 Programma MATLAB per l'elaborazione dei risultati

Il programma MATLAB che si andrà adesso a commentare dovrà effettuare un'analisi mirata sulle matrici ottenute dall'esperimento; nella fattispecie ciò che verrà fatto può essere riassunto con quanto segue:

- per ogni riga di tutte le N matrici, calcolare la media e la deviazione standard, analogamente per quanto riguarda le colonne.
- creare grafici in cui in ognuno di essi si mostra l'andamento del valore medio su ogni riga, per tutte le matrici acquisite. Analogamente per quanto riguarda le colonne.

La prima parte del programma ha lo scopo di estrapolare dal file testuale importato in MATLAB le singole matrici e salvarle all'interno di una struttura di tipo Cell Array denominata in questo caso *matrici.*

```
matrici={};
medie_righe={};
deviazioni_colonne={};
medie_colonne={};
deviazioni_righe={};

a=zeros(8,8);
righe=0;
colonne=0;
elementi=0;
index=0;
```

Figura 3.14: Dichiarazione variabili.

Per far ciò si sfrutta la struttura iterativa del ciclo *for*, come si può vedere in **Figura 3.15**. Il file testuale che è stato importato è composto da 2904 righe; tramite un doppio *for* si itera all'interno del file testuale; quest'ultimo può essere visto come una matrice composta da 2904 righe e 8 colonne.

Tramite una matrice temporanea (8x8) denominata *a,* si salva ogni singolo elemento della matrice 2904x8, questo fino a quando il prodotto righe per colonna non risulta 64: ciò vuol dire che è stata estratta una delle N matrici fornite dal sensore. Ogni singola matrice 8x8 viene salvata sulla struttura *matrici,* la quale conterrà tutte le N matrici misurate.

```
for i=1:2904
    %tramite questo for scorro all'interno del file importato
    righe=righe+1;
    colonne=0;
    for j=1:8
        colonne=colonne+1;

        a(righe,colonne)=Matrici3005(i,j);
    end
    p=righe*colonne;
    if p==64
        index=index+1;
        %aggiungo la matrice 8x8 a matrici (cell array)
        matrici{index}=a;
        righe=0;
        colonne=0;
    end
end
```

Figura 3.15: Estrazione delle singole matrici 8x8 dal file testuale importato.

La parte successiva del programma calcola le medie e le deviazioni standard, nelle modalità descritte nella pagina precedente.

```
36 -    mediaRiga=zeros(1,8);
37 -    deviazioneRiga=zeros(1,8);
38
39 -    for i=1:363
40          %i indica la matrice che sto analizzando
41 -        matrice=matrici{i};
42 -        for k=1:8
43              %prelevo riga k-esima e inserisco in un vettore denominato riga
44 -            riga=matrice(k,:);
45              %calcolo valore medio riga e deviazione e inserisco in una delle 8
46              %celle
47 -            mediaRiga(k)=mean(riga);
48 -            deviazioneRiga(k)=std(riga);
49 -        end
50          %dopo aver calcolato le 8 medie e le 8 deviazioni, salvo gli array nel
51          %cell-array
52 -        medie_righe{i}=mediaRiga;
53 -        deviazioni_righe{i}=deviazioneRiga;
54 -    end
```

Figura 3.16: Calcolo delle medie e deviazioni standard per ogni singola riga.

Le matrici estratte sono in totale 363: tramite un *for* si itera all'interno del Cell Array *matrici* (si ricorda che ogni sua cella contiene una matrice 8x8). La i-esima matrice viene salvata all'interna di una variabile temporanea chiamata *matrice.*

Il secondo *for* (riga 42) permette di estrarre la singola riga; successivamente si calcola la media e la deviazione della k-esima riga estratta e si salvano tali valori all'interno dei vettori *mediaRiga* e *deviazioniRiga* (entrambi aventi 8 celle).

Ogni k-esima cella di questi due vettori contiene rispettivamente la media e la deviazione standard della k-esima riga; i vettori, quindi, conterranno complessivamente 8 medie e 8 deviazioni.

```
58      %stesso discorso per quanto riguarda le colonne
59 -    mediaColonna=zeros(1,8);
60 -    deviazioneColonna=zeros(1,8);
61
62 -    for i=1:363
63          %i indica la matrice che sto analizzando
64 -        matrice=matrici{i};
65 -        for k=1:8
66              %prelevo colonna k-esima
67 -            colonna=matrice(:,k);
68              %calcolo valore medio colonna e deviazione
69 -            mediaColonna(k)=mean(colonna);
70 -            deviazioneColonna(k)=std(colonna);
71 -        end
72          %dopo aver calcolato le 8 medie e le 8 deviazioni, carico gli array nel
73          %cell-array: ogni array contiene 8 medie (una per riga)
74 -        medie_colonne{i}=mediaColonna;
75 -        deviazioni_colonne{i}=deviazioneColonna;
76 -    end
```

Figura 3.17: Procedimento analogo per le colonne.

Questi due vettori vengono salvati caricandoli all'interno di due Cell Array (riga 52 e riga 53); successivamente il ciclo ricomincia andando a fare lo stesso e identico procedimento per le matrici successive. In **Figura 3.17** il procedimento è analogo ma questa volta riferito alle colonne.

```
82      %Prima riga
83      %il vettore conterrà tutte le medie riferito alla rigal di tutte e 363 le
84      %matrici
85 -    medie_primaRiga=zeros(1,363);
86      %vettore di passaggio
87 -    vettore=zeros(1,8);
88 -    for i=1:363
89          %medie_righe(i) contiene il vettore delle 8 medie della i-esima matrice
90          %devo preleveare dagli array il primo elemento (1)
91         %medie_righe{i}=vettore;
92 -       vettore=medie_righe{i};
93         %la cella 1 di vettore conterrà la media della prima riga della i-esima
94         %matrice, la cella 2 la media della seconda riga della i-esima matrice
95
96 -       medie_primaRiga(i)=vettore(1);
97         %il vettore medie_primaRiga conterrà nella cella 1 la media della prima
98         %della matrice 1, nella cella 2 la media della prima riga della matrice
99         %2
100 -   end
```

Figura 3.18: Estrazione di tutte le medie riferite alle prime righe.

Per poter graficare tutte le medie delle N (N=363) matrici, riferite ad ogni singola riga, si procede come riportato in **Figura 3.18.** Si crea (riga 85) un vettore che conterrà tutte le medie delle prime righe delle matrici. Sempre tramite un ciclo *for* si itera all'interno delle 363 celle della struttura *medie_righe* e si preleva l'i-simo vettore (riga 92). Da questo vettore, volendo avere la media della prima riga, si preleva la prima cella.

Lo stesso procedimento vale anche per le righe successive, con l'unica differenza che dalla variabile *vettore* si preleva la riga corrispondente.

Dopo avere estratto tutte le medie delle matrici, corrispondenti a ciascuna riga, si procede con il plot dei risultati (**Figura 3.19**). Come si può vedere, sono stati creati 8 vettori, ognuno contenente le 363 medie corrispondenti alla rispettiva riga.

```
153      %%Plot risultati RIGHE
154 -    figure;plot(medie_primaRiga);title("Medie delle prime righe delle matrici");ylabel("Temperatura (°C)");xlabel("Campione"
155 -    figure;plot(medie_secondaRiga);title("Medie delle seconde righe delle matrici");ylabel("Temperatura (°C)");xlabel("Campi
156 -    figure;plot(medie_terzaRiga);title("Medie delle terze righe delle matrici");ylabel("Temperatura (°C)");xlabel("Campione"
157 -    figure;plot(medie_quartaRiga);title("Medie delle quarte righe delle matrici");ylabel("Temperatura (°C)");xlabel("Campion
158 -    figure;plot(medie_quintaRiga);title("Medie delle quinte righe delle matrici");ylabel("Temperatura (°C)");xlabel("Campion
159 -    figure;plot(medie_sestaRiga);title("Medie delle seste righe delle matrici");ylabel("Temperatura (°C)");xlabel("Campione"
160 -    figure;plot(medie_settimaRiga);title("Medie delle settime righe delle matrici");ylabel("Temperatura (°C)");xlabel("Campi
161 -    figure;plot(medie_ottavaRiga);title("Medie delle ottave righe delle matrici");ylabel("Temperatura (°C)");xlabel("Campion
```

Figura 3.19: Plot dei risultati per quanto riguarda le medie corrispondenti alle righe.

Il resto del codice è identico a quello mostrato in Figura 3.18 e Figura 3.19, solamente riferito alle colonne.

3.3.2 Commento dei risultati ottenuti

Nelle pagine successive sono mostrati i sedici grafici ottenuti tramite il programma MATLAB di cui si è parlato nel paragrafo precedente.

I primi otto grafici riguardano le medie riferite alle righe. Anche se a prima vista i risultati possono avere un andamento molto simile tra loro, ci sono comunque delle differenze. Alcuni grafici riportano un valore medio più basso, altri invece un valore medio più alto. Ciò naturalmente dipende da come i valori sul bordo e quelli centrali della matrice tendono a essere più caldi o più freddi.

Infatti, alcune matrici hanno dei valori sui bordi più caldi, mentre sul centro tendono a essere più freddi. Altre invece l'esatto contrario. Ciò dipende da come la temperatura dell'area di 64 pixels dell'oggetto inquadrato, in questo caso l'acqua che si raffredda, vari durante il corso dell'esperimento. Se si prende quindi una delle medie delle matrici a caso (valore sull'asse delle ascisse) e si confrontano gli otto grafici tra loro, si può vedere come i valori siano molto simili tra loro. Questo permette quindi di concludere come i valori sulle diverse righe tendono a disporsi in modo omogeneo, portando quindi ad avere delle medie simili tra le diverse righe.

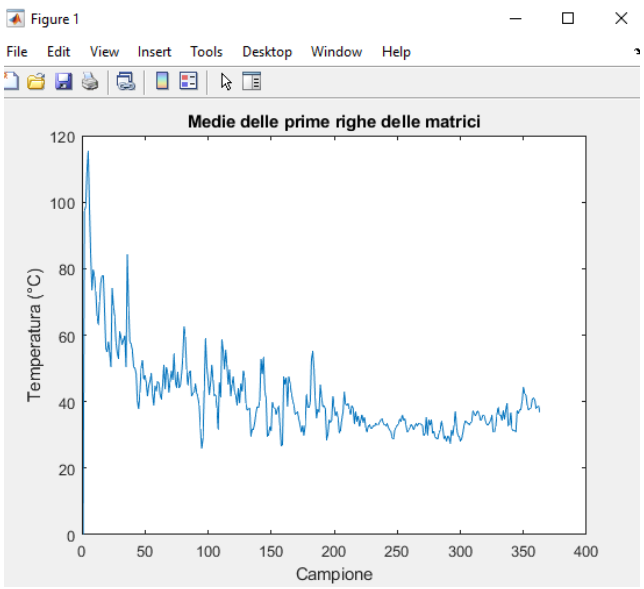


Figura 3.20: Medie delle prime righe delle N matrici.

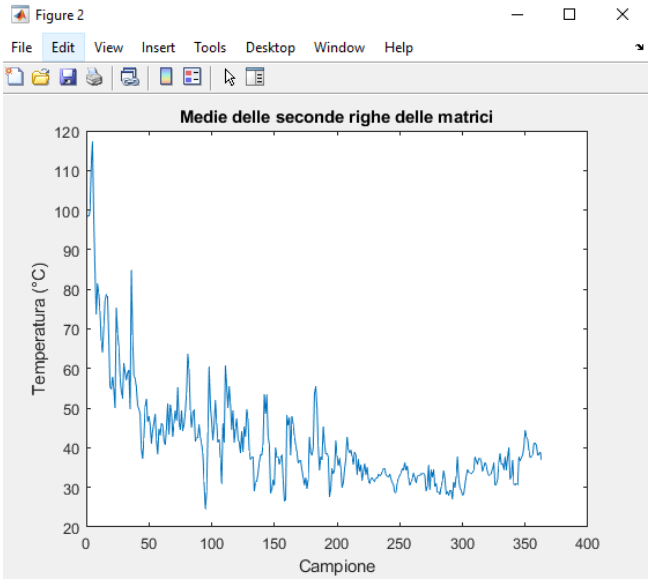


Figura 3.21: Medie delle seconde righe delle N matrici.

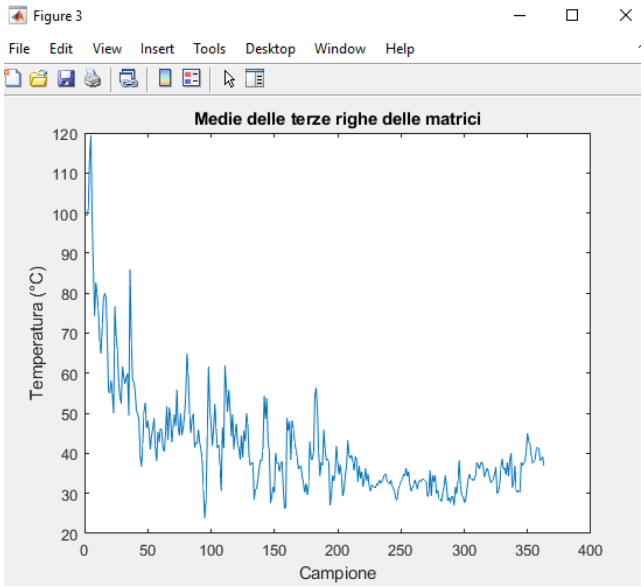


Figura 3.22: Medie delle terze righe delle N matrici.

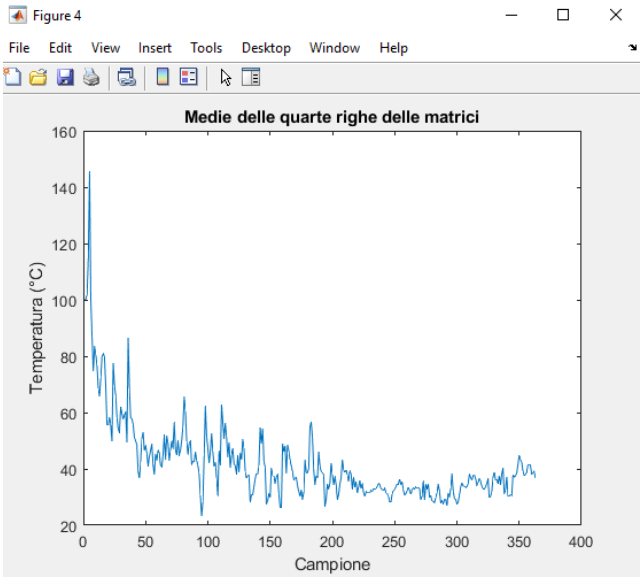


Figura 3.23: Medie delle quarte righe delle N matrici.

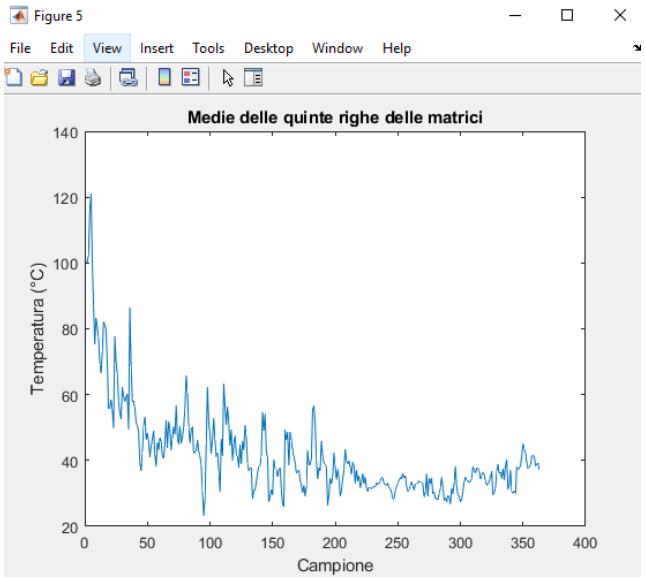


Figura 3.24: Medie delle quinte righe delle N matrici.

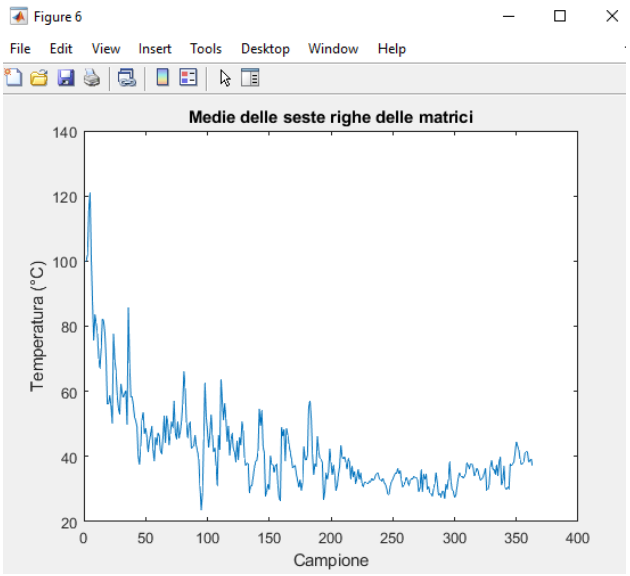


Figura 3.25: Medie delle seste righe delle N matrici.

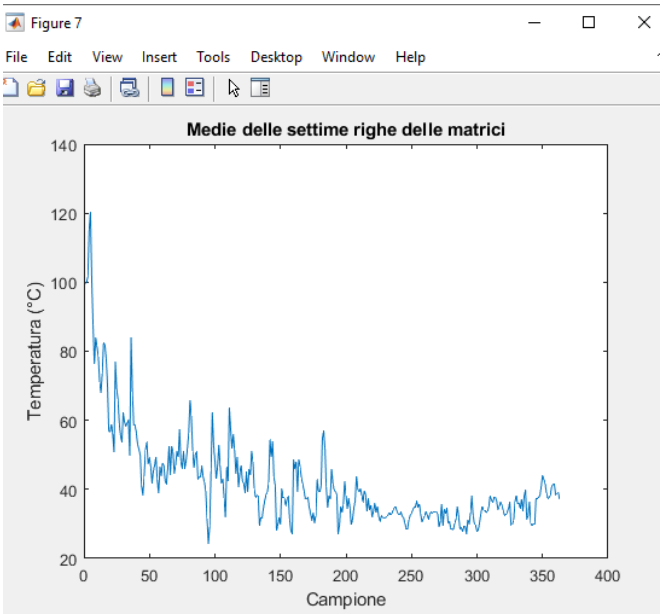


Figura 3.26: Medie delle settime righe delle N matrici.

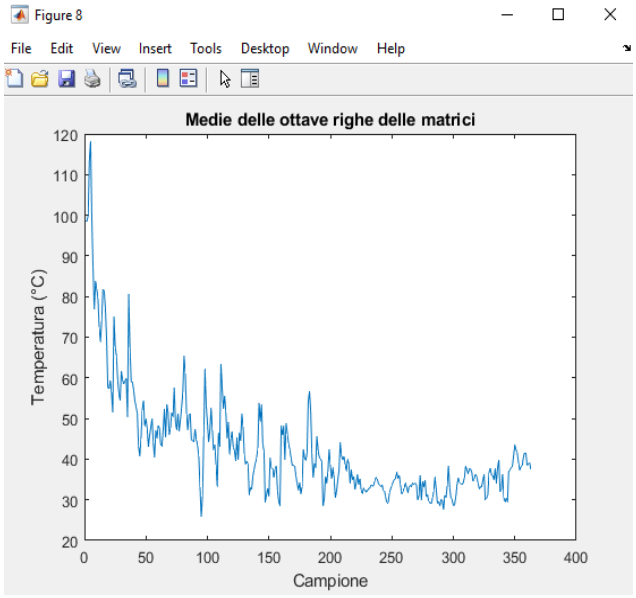


Figura 3.27: Medie delle ottave righe delle N matrici.

Di seguito si riportano i risultati per quanto riguarda il calcolo delle medie delle colonne.

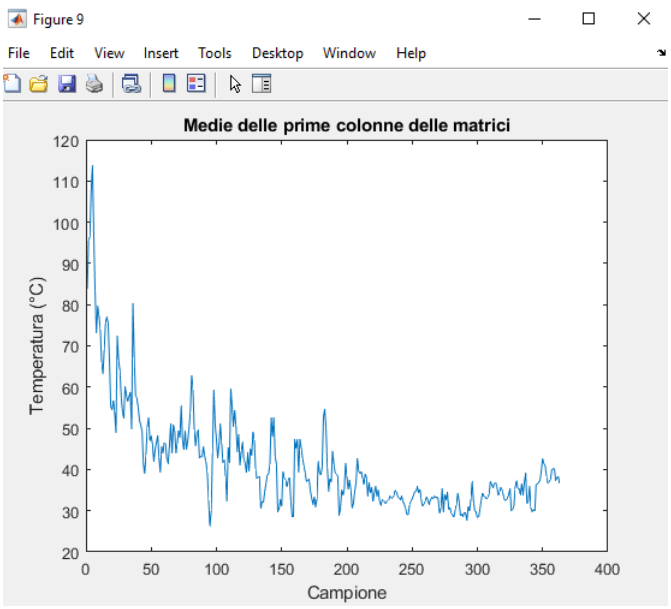


Figura 3.28: Medie delle prime colonne delle N matrici.

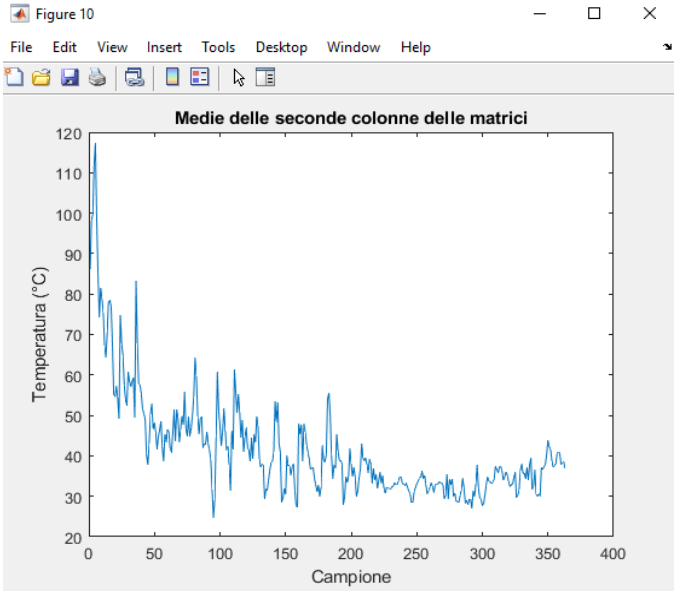


Figura 3.29: Medie delle seconde colonne delle N matrici.

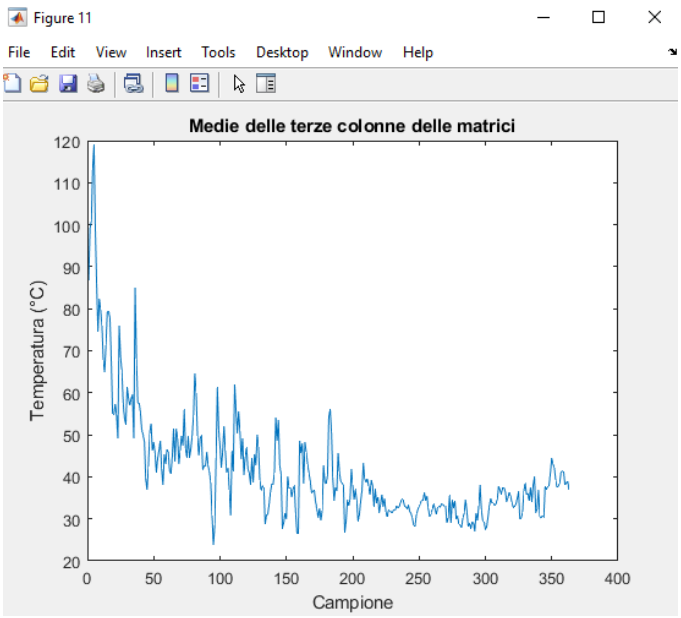


Figura 3.30: Medie delle terze colonne delle N matrici.

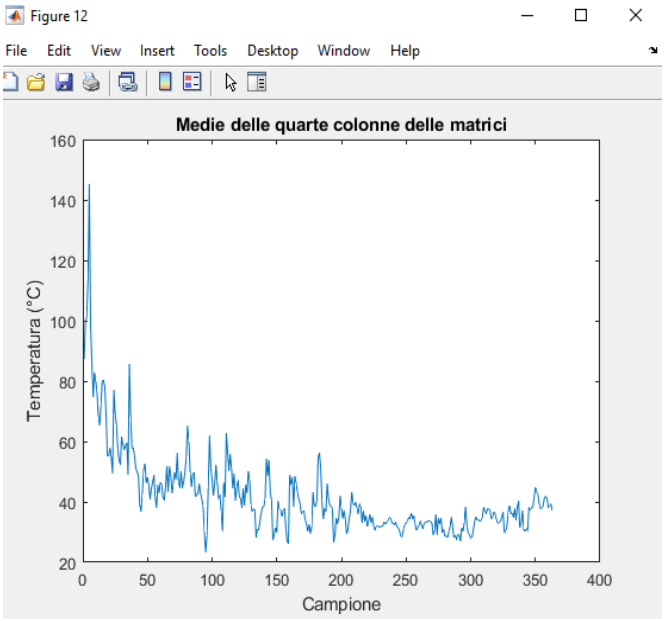


Figura 3.31: Medie delle quarte colonne delle N matrici.

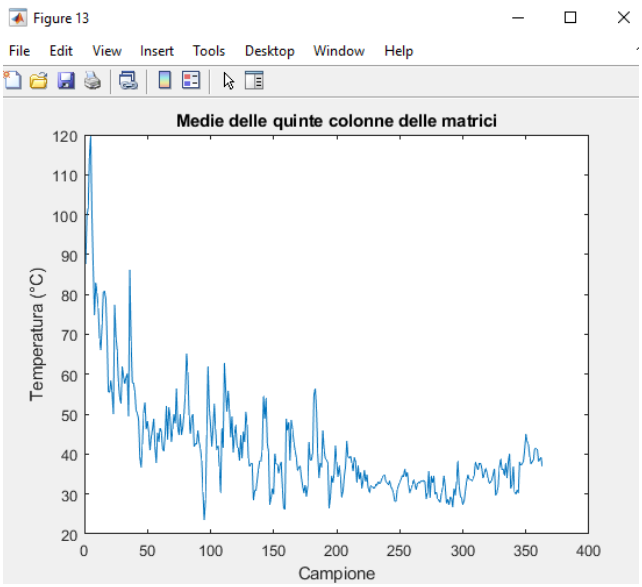


Figura 3.32: Medie delle quinte colonne delle N matrici.

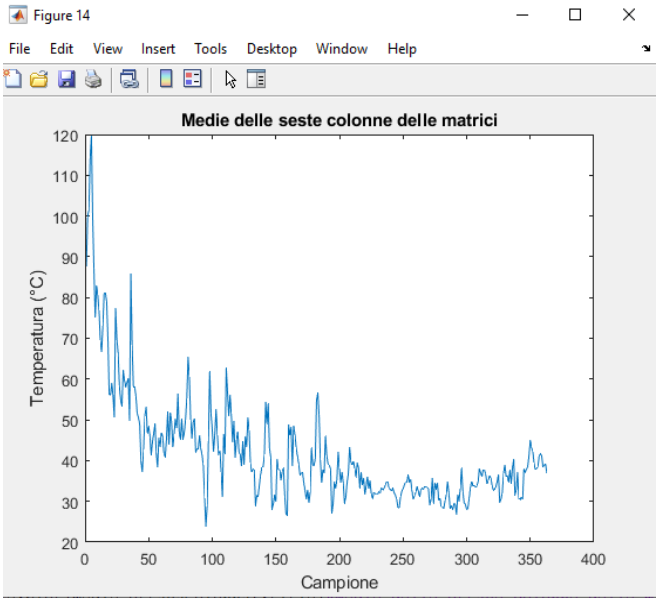


Figura 3.33: Medie delle seste colonne delle N matrici.

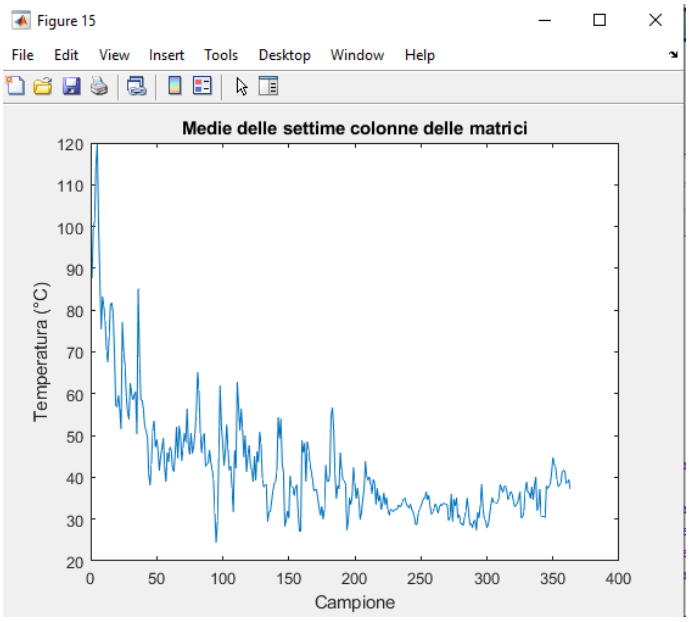


Figura 3.34: Medie delle settime colonne delle N matrici.

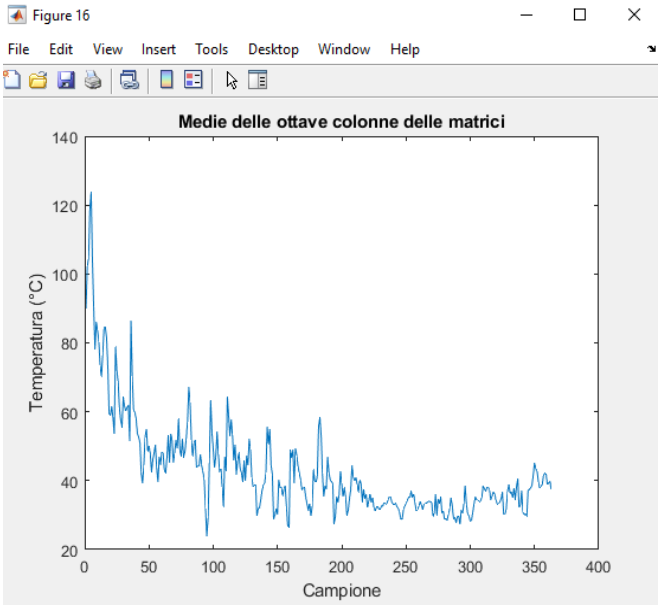


Figura 3.35: Medie delle ottave colonne delle N matrici.

Le considerazioni che si possono fare sulle medie ottenute dalle colonne sono analoghe a ciò che è stato detto per le righe. L'unica differenza che si può riscontrare è che nel caso delle colonne il valore medio degli otto differenti grafici tende ad essere lievemente più simile.

3.4 Considerazioni finali sul comportamento dei sensori

In conclusione, tramite questi tre esperimenti si è avuto modo di poter constatare il diverso funzionamento che i tre sensori hanno avuto in due situazioni di utilizzo: temperatura variabile e temperatura fissa.

Nel caso dell'esperimento riguardante la prova di misurazione di temperatura costante, si è avuto modo di vedere come i risultati ottenuti siano, seppur con le giuste variazioni, simili tra loro.

Quando invece si è considerato come caso di utilizzo quello di una misurazione di temperatura variabile, le differenze dei tre sensori sono risaltate maggiormente. Si è avuto modo di constatare come il sensore AMG8833, rispetto al MLX90614, sia il più sensibile a fattori esterni come la condensa. Il DS18B20 invece, essendo un sensore di contatto è quello che ha riportato delle misurazioni con variazioni meno spigolose.

Naturalmente queste prove non sono esaustive nell'evidenziare tutte le differenze che i tre sensori riportano, ma solamente una parte riferita agli esperimenti che sono stati svolti. Gli esperimenti e le situazioni di utilizzo dei sensori di temperatura sono molteplici e sarebbe necessario condurre ulteriori test per un'analisi esaustiva del comportamento dei sensori considerati.

Conclusioni

Si è cercato di presentare i capitoli del presente documento di Tesi in modo tale da seguire il percorso logico effettuato durante la sua stesura. Una fase iniziale di studio e richiami teorici riguardo il materiale che si stava per andare a utilizzare, una seconda parte di studio di quest'ultimo e il successivo lavoro di progetto vero e proprio.

Essendo stato un progetto di tesi sperimentale, non sono mancate le occasioni di poter mettere in pratica ciò che si è studiato. Questi mesi di tirocinio e la relativa tesi sono stati sia impegnativi ma anche gratificanti. Aver potuto costruire un sistema, sicuramente ampliabile e migliorabile, che potesse misurare la temperatura da tre sensori eterogenei è stato sicuramente appagante.

Il percorso è stato oltretutto istruttivo. Si è avuto modo di ampliare le proprie conoscenze su argomenti come sistemi embedded e sensoristica. Su quest'ultimo ambito soprattutto, le conoscenze sono aumentate, ampliando così il bagaglio culturale tecnico che si è avuto modo già di costruire nel corso della Laurea triennale.

Ciò non solo nella parte più elettronica del progetto ma anche in quella di sviluppo del codice. Si è avuto modo di imparare un nuovo linguaggio di programmazione e averne avuto un riscontro pratico sulla scheda di sviluppo. Tramite lo sviluppo finale dei grafici si è avuto modo di consolidare le conoscenze acquisite in corsi precedenti sull' analisi dei segnali tramite linguaggio MATLAB.

Naturalmente, come in tutti i progetti, le difficoltà non sono mancate. A volte problemi semplici come un cavetto che dava falso contatto, altre volte più complessi come la mancata lettura di temperatura del sensore a causa di errori di codice. Ma è giusto che sia così, l'importante è essere riusciti a raggiungere l'obiettivo finale, ovvero, la realizzazione di un sistema a microcontrollore che potesse acquisire e raccogliere valori di temperatura da tre sensori eterogenei.

Come già accennato nell'introduzione, sicuramente il sistema è ampliabile a diverse tecnologie di sensori. Ma si può essere soddisfatti di ciò che è stato raggiunto, essendo una buona base da cui partire per sviluppi futuri. Un sistema di questo tipo ha illimitati ambiti di utilizzo, non solo per quanto riguarda il controllo della temperatura. Si può realizzare un sistema per acquisire la temperatura, la pressione, l'umidità e altre grandezze fisiche sfruttando come base di partenza ciò che è stata descritto nel presente documento.

In definitiva posso ritenermi molto soddisfatto, sia dei risultati raggiunti, ma anche delle conoscenze acquisite che potranno sicuramente tornarmi utili in futuro non solo dal punto di vista didattico ma anche lavorativo.

Sitografia-Bibliografia

[1] «Sistema internazionale,» [Online]. Available: https://www.treccani.it/enciclopedia/sistema-internazionale/.

[2] C. Landi, N. Pasquino e A. Baccigalupi. [Online]. Available: https://www.docenti.unina.it/webdocenti-be/allegati/materiale-didattico/289389.

[3] P. Pozzolo, «Deviazione standard: definizione e significato,» [Online]. Available: https://paolapozzolo.it/deviazione-standard/.

[4] «sensore,» [Online]. Available: https://www.treccani.it/enciclopedia/sensore.

[5] «Infrarosso,» [Online]. Available: https://www.treccani.it/enciclopedia/infrarosso/.

[6] J. Fraden, Handbook of Modern Sensors: Physics,Designs, and Applications, San Diego, CA, USA: Springer, Fifth edition, 2016.

[7] «Datasheet for MLX90614,» [Online]. Available: https://www.melexis.com/en/documents/documentation/datasheets/datasheet-mlx90614.

[8] «AMG8833 : Infrared Array Sensor Grid-EYE,» [Online]. Available: https://industrial.panasonic.com/ww/products/pt/grid-eye/models/AMG8833.

[9] «Adafruit AMG8833 IR Thermal Camera FeatherWing,» [Online]. Available: https://www.adafruit.com/product/3622.

[10] «Overview of 1-Wire Technology and Its Use,» [Online]. Available: https://www.analog.com/en/technical-articles/guide-to-1wire-communication.html.

[11] «Arduino Uno Rev3 SMD,» [Online]. Available: https://store.arduino.cc/products/arduino-uno-rev3-smd.

[12] «Previous IDE Relases,» [Online]. Available: https://www.arduino.cc/en/software.

[13] «Code With Mu,» [Online]. Available: https://codewith.mu/.

[14] «CircuitPython,» [Online]. Available: https://circuitpython.org/.

[15] «Libraries,» [Online]. Available: https://circuitpython.org/libraries.

[16] «Data Logger,» [Online]. Available: https://learn.adafruit.com/getting-started-with-raspberry-pi-pico-circuitpython/data-logger.