UNIVERSITÀ
POLITECNICA
DELLE MARCHE

UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Corso di Laurea Triennale in Ingegneria Elettronica

Implementazione e test di un sistema multisensore per l'acquisizione di immagini RGB e termiche su scheda embedded
Implementation and testing of an embedded multisensor system for the acquisition of RGB and thermal images

Tesi di Laurea di:
Davide Capuano

Relatore: Chiar.ma
Prof.ssa **Susanna Spinsante**

Correlatore:
Dott. Ing. **Gianluca Ciattaglia**

Anno Accademico 2023/2024

Abstract

In questo studio viene sviluppato un sistema integrato che combina un sensore RGB e un sensore termico a infrarossi per acquisire simultaneamente dati visivi e termici. L'obiettivo principale é valutare l'efficacia dell'integrazione di questi due tipi di sensori, analizzando come le informazioni termiche possano arricchire i dati visivi ottenuti dal sensore RGB. La piattaforma hardware utilizzata si basa su una scheda Portenta, che consente l'elaborazione in tempo reale di immagini e dati termici. Le misurazioni vengono effettuate a diverse distanze dal sensore per verificare l'impatto della distanza sulla precisione e stabilitá delle letture termiche.

L'analisi si concentra su due parametri principali: la temperatura media degli oggetti individuati nelle immagini RGB e la variabilitá delle misurazioni, quantificata tramite la deviazione standard. Questi parametri permettono di comprendere quanto siano precise e affidabili le misurazioni fornite dal sensore termico in diverse condizioni sperimentali. Inoltre, viene calcolata l'incertezza di tipo A, per stimare il livello di fiducia associato alla media delle misurazioni e valutare la qualitá complessiva dei dati acquisiti.

L'obiettivo finale é generare una mappa termica dettagliata che correli le informazioni visive e termiche, migliorando la comprensione delle caratteristiche degli oggetti rilevati.

Questa integrazione tra sensori visivi e termici rappresenta un importante passo avanti verso la creazione di sensori aumentati, in grado di fornire una visione multidimensionale piú accurata delle scene inquadrate. Le potenziali applicazioni di questa tecnologia spaziano dalla diagnostica medica, dove i dati termici possono contribuire a rilevare anomalie fisiologiche, all'automazione industriale, alla sicurezza e alla sorveglianza. In ambito medico, l'aggiunta di informazioni termiche potrebbe migliorare la diagnosi o il monitoraggio di patologie, mentre in contesti di sicurezza, un sistema che combina immagini RGB e dati termici potrebbe consentire un'analisi piú dettagliata e informata di oggetti o persone in movimento.

L'approccio adottato in questo lavoro si propone di fornire un quadro preliminare sull'affidabilitá e potenzialitá della combinazione di sensori RGB e termici. Sebbene

la fase sperimentale si concentri sulla valutazione della precisione e stabilitá delle misurazioni in funzione della distanza, lo studio pone le basi per futuri sviluppi in cui questi sensori potranno essere ulteriormente potenziati attraverso l'uso di algoritmi di intelligenza artificiale, migliorando la capacitá di analizzare e interpretare le scene in tempo reale con maggiore profonditá e precisione.

In sintesi, questo progetto rappresenta un primo passo verso l'integrazione di tecnologie avanzate che uniscono visione e percezione termica, con prospettive future di sviluppo in settori chiave come la robotica, l'Internet delle Cose (IoT) e i sistemi di monitoraggio intelligente.

Abstract

This study presents the development of an integrated system combining an RGB sensor and an infrared thermal sensor to simultaneously capture visual and thermal data. The main objective is to assess the effectiveness of this integration by analyzing how thermal information can enhance the visual data obtained from the RGB sensor. The hardware platform is based on the Portenta board, enabling real-time processing of both images and thermal data. Measurements are performed at various distances from the sensor to evaluate how distance impacts the accuracy and stability of the thermal readings.

The analysis focuses on two key parameters: the average temperature of the objects detected in the RGB images and the variability of the measurements, quantified using the standard deviation. These parameters help understanding how precise and reliable the thermal readings are under different experimental conditions. Additionally, Type A uncertainty is calculated to estimate the confidence level associated with the average measurements and to evaluate the overall quality of the collected data.

The data is processed using edge detection algorithms, which precisely identify the contours of objects or faces in the RGB images, followed by the analysis of thermal data for the selected pixels. The final goal is to generate a detailed thermal map that correlates visual and thermal information, improving the understanding of the characteristics of the detected objects.

This integration of visual and thermal sensors represents a significant step forward toward the creation of advanced sensors capable of providing a more accurate, multidimensional view of the scenes being captured. Potential applications of this technology range from medical diagnostics, where thermal data can help detect physiological anomalies, to industrial automation, security, and surveillance. In medical contexts, the inclusion of thermal information could enhance diagnosis or monitoring of pathological conditions. Meanwhile, in security settings, a system combining RGB images and thermal data could enable a more detailed and informed analysis of moving objects or individuals.

The approach adopted in this work aims to provide a preliminary assessment of the reliability and potential of combining RGB and thermal sensors. Although the experimental phase focuses on evaluating the precision and stability of the measurements based on distance, this study lays the groundwork for future developments where these sensors can be further enhanced through artificial intelligence algorithms, improving their ability to analyze and interpret scenes in real time with greater depth and accuracy.

In conclusion, this project represents a first step toward the integration of advanced technologies that combine vision and thermal perception, with promising future developments in key fields such as robotics, the Internet of Things (IoT), and intelligent monitoring systems.

Indice

Capitolo 1

Panoramica della tesi

Questo capitolo ha l'intento di guidare il lettore, fornendo una spiegazione dettagliata di ció che é stato progettato in relazione all'obiettivo stabilito.

1.1 Introduzione

Uno dei principali problemi che si presentano nell'ambito dell'elettronica é l'interfacciamento tra i microcontrollori e l'essere umano. Questa interazione é fondamentale per lo sviluppo di dispositivi elettronici intelligenti e reattivi. Per facilitare questa connessione sono state sviluppate due categorie principali di componenti: gli attuatori e i sensori.

Gli attuatori svolgono un ruolo cruciale trasformando segnali elettrici in output fisici. Questi output possono manifestarsi in diverse forme, come il movimento meccanico di un motore, l'emissione di luce da un LED o la generazione di calore attraverso un elemento riscaldante. In sostanza, gli attuatori agiscono come intermediari che permettono di convertire le istruzioni digitali emesse dai microcontrollori in azioni tangibili e percepibili nell'ambiente.

D'altra parte, i sensori svolgono la funzione opposta. Questi dispositivi sono progettati per captare grandezze fisiche dall'ambiente circostante, come temperatura, pressione, luminositá e movimento. Una volta rilevate, queste grandezze fisiche vengono convertite in segnali elettrici, quali tensioni, correnti o variazioni di resistenza, che

possono essere interpretati dai microcontrollori. In questo modo i sensori permettono agli apparecchi elettronici di percepire il mondo esterno e reagire di conseguenza.
In sintesi, gli attuatori e i sensori sono componenti essenziali che consentono un'interazione efficace tra i microcontrollori e l'ambiente, facilitando la creazione di sistemi elettronici che possano adattarsi e rispondere in modo intelligente alle condizioni esterne. Questa sinergia é fondamentale per lo sviluppo di tecnologie avanzate, come l'Internet delle Cose (IoT), i robot autonomi e i sistemi di automazione domestica.

1.2 Analisi del problema

Negli ultimi anni, la crescente richiesta di tecnologie avanzate per il monitoraggio senza contatto ha spinto lo sviluppo di sensori sempre piú sofisticati, capaci di acquisire informazioni dettagliate e precise da diverse fonti. Tra questi, i sensori RGB e i sensori termici sono strumenti chiave utilizzati in una vasta gamma di applicazioni, dall'automazione industriale alla diagnostica medica, fino all'ambito della sicurezza e sorveglianza. I sensori RGB, con la loro capacitá di catturare immagini anche ad alta risoluzione e di distinguere le componenti cromatiche, sono fondamentali per l'analisi visiva di oggetti o persone, mentre i sensori termici, sensibili alla radiazione infrarossa, permettono di rilevare la temperatura senza contatto diretto.
L'integrazione di questi due tipi di sensori offre una prospettiva affascinante per la creazione di sistemi di rilevamento "aumentati", in grado di combinare informazioni visive e termiche per fornire un'analisi piú completa e accurata delle scene inquadrate. Un potenziale ambito di applicazione di questa tecnologia é quello medico, dove l'aggiunta dell'informazione termica a immagini RGB potrebbe migliorare la diagnosi o il monitoraggio di condizioni patologiche. In un contesto industriale o di sicurezza, un "sensore RGB aumentato" potrebbe offrire una valutazione non solo dell'aspetto visivo di un oggetto o persona, ma anche delle sue proprietá termiche, consentendo decisioni piú informate e tempestive.
In questo lavoro ci proponiamo di sviluppare una piattaforma di acquisizione che combini un sensore RGB e un sensore di temperatura senza contatto, implementando

un sistema su una scheda Portenta. L'obiettivo principale é quello di valutare le prestazioni del sensore termico in combinazione con il sensore RGB, al fine di analizzare se e come le informazioni fornite dal sensore termico possano "arricchire" quelle offerte dal sensore RGB. Questo lavoro rappresenta un passo preliminare per la futura realizzazione di un "sensore RGB aumentato" dotato di capacitá di analisi avanzata tramite algoritmi di intelligenza artificiale, anche se questa seconda fase non verrá trattata nel presente studio.
Nella fase sperimentale si effettueranno acquisizioni simultanee di dati RGB e valori di temperatura, con lo scopo di elaborare successivamente i dati in Matlab. L'elaborazione consisterá in una edge detection sulle immagini RGB, per identificare i pixel corrispondenti a oggetti o volti, e nella conseguente analisi dei dati termici solo per i pixel selezionati. In particolare, si calcoleranno il valore medio della temperatura e la dispersione dei dati termici relativi ai pixel individuati, ripetendo il processo per diverse condizioni sperimentali. Infine, l'analisi permetterá di valutare la stabilitá e la coerenza dei dati acquisiti, variando condizioni quali la distanza del soggetto dalla camera.
Questo lavoro intende quindi fornire un quadro preliminare sull'affidabilitá e la potenzialitá dell'integrazione di sensori RGB e termici per future applicazioni, creando le basi per lo sviluppo di sensori "aumentati" capaci di analisi multidimensionali piú profonde e accurate.

1.3 Soluzione proposta

Per lo sviluppo del sistema di misura della temperatura, si é scelto di utilizzare una piattaforma hardware che integra un microcontrollore con una fotocamera RGB e un sensore a infrarossi, consentendo l'acquisizione simultanea di dati visivi e termici. Il sensore a infrarossi rileva la temperatura della scena inquadrata, mentre la fotocamera RGB cattura immagini che verranno successivamente analizzate per identificare oggetti o aree specifiche di interesse.

Il sistema, una volta ottenuti i dati visivi e termici, si concentra sull'elaborazione della temperatura media dei pixel che corrispondono agli oggetti individuati nelle immagini. Questo permette di associare i dati visivi a una mappa termica dettagliata, evidenziando la correlazione tra le caratteristiche dell'immagine e le temperature rilevate in quella stessa area. La media delle temperature calcolata su questi pixel rappresenta una prima valutazione della scena in termini termici.
Un aspetto cruciale dell'analisi é la valutazione della variabilitá e dell'affidabilitá delle misurazioni di temperatura. In questo contesto un indicatore importante é la deviazione standard, una misura che quantifica la dispersione dei dati intorno alla media. Essa fornisce informazioni chiave sulla stabilitá delle misure: una deviazione standard elevata indica che i valori delle temperature variano significativamente rispetto alla media, suggerendo una maggiore dispersione o incertezza nei dati, mentre una deviazione standard bassa suggerisce misurazioni piú uniformi e stabili. La formula matematica utilizzata per calcolare la deviazione standard é:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2}$$

In cui N rappresenta il numero totale delle misurazioni effettuate, x_i sono i singoli valori delle misure ti temperatura e μ é la media calcolata su i singoli campioni.
Questa formula descrive la distanza media di ciascun valore x_i dalla media μ consentendo di valutare quanto le temperature differiscano dal valore centrale. L'utilizzo della deviazione standard é essenziale in questo tipo di analisi, in quanto fornisce una misura diretta della precisione dei dati raccolti e permette di identificare eventuali variazioni significative all'interno del set di misure.
Oltre alla deviazione standard é importante considerare l'incertezza di tipo A, che rappresenta una stima quantitativa dell'incertezza statistica basata sull'analisi delle misurazioni ripetute. Questa incertezza viene calcolata quando si ha a disposizione un numero sufficiente di dati sperimentali e fornisce un'indicazione dell'affidabilitá della media delle misure. La formula utilizzata per l'incertezza di tipo A é:

$$u_A = \frac{\sigma}{\sqrt{N}}$$

In questo caso, l'incertezza di tipo A si basa sulla deviazione standard e sul numero di misurazioni N. Essa riflette il grado di confidenza che si puó avere nella media delle misure come rappresentazione del valore reale. Se N é sufficientemente grande l'incertezza di tipo A tende a ridursi, suggerendo che la media delle misure é sempre piú vicina al valore reale della grandezza fisica che si sta valutando.

Entrambi questi concetti, la deviazione standard e l'incertezza di tipo A, sono fondamentali nell'analisi dei risultati di un esperimento di misura. La deviazione standard permette di valutare quanto le misure siano precise, indicando la loro variabilitá attorno alla media, mentre l'incertezza di tipo A fornisce una stima della fiducia che si puó avere sulla media delle misurazioni, offrendo una misura quantitativa della qualitá del risultato sperimentale.

In un contesto sperimentale la combinazione di queste due grandezze consente di ottenere un quadro chiaro della precisione e dell'affidabilitá delle misurazioni. Ad esempio, se si osserva una bassa deviazione standard accompagnata da una ridotta incertezza di tipo A, si puó concludere che il sistema di misura fornisce dati stabili e precisi. Al contrario, se entrambe le misure sono elevate, é probabile che siano presenti errori o variabili non controllate che influenzano i risultati.

L'analisi statistica dei dati, tramite questi strumenti, diventa quindi essenziale non solo per interpretare correttamente i risultati, ma anche per migliorare la qualitá delle misurazioni e ridurre gli errori associati.

Per ottenere un'analisi piú completa e precisa é fondamentale eseguire le misurazioni a distanze diverse dal sensore. Questo approccio consente di valutare come la distanza tra l'oggetto inquadrato e il sensore influenzi i dati raccolti, sia in termini di temperatura media che di variabilitá delle misurazioni. Infatti, in un sistema di misura basato su sensori termici senza contatto, la distanza puó influenzare significativamente la precisione delle letture termiche, poiché fattori come la risoluzione del sensore, l'angolo di campo e la dispersione dei raggi infrarossi variano con la distanza.

L'esecuzione di misurazioni a diverse distanze consente quindi di stabilire le condizioni ottimali per l'utilizzo del sistema, identificando la distanza alla quale il sensore fornisce i dati piú stabili e precisi. Inoltre, tale approccio permette di valutare la robustezza

del sistema, testando come esso reagisca in situazioni di variabilitá nelle condizioni sperimentali.

Capitolo 2

Strumenti di sviluppo hardware e software

Si continua con la descrizione dei diversi componenti hardware utilizzati e dei software impiegati per programmare l'intero sistema. Si inizia con un'introduzione ai sistemi embedded, che costituiscono il nucleo del progetto, e alla loro composizione. Successivamente, si approfondiscono le due schede Arduino, i sensori utilizzati, ovvero il sensore AMG8833 e il MLX90614, e infine si esaminano gli strumenti software adottati, tra cui Arduino IDE e MatLab.

2.1 Software e hardware utilizzati

La scheda Arduino Portenta H7 funge da unitá principale di elaborazione, con la funzione di raccogliere ed elaborare i dati provenienti dalle schede secondarie, in particolare dalla vision shield e dal sensore AMG8833: la prima fornisce al sistema la capacitá di visione artificiale mentre la seconda realizza una mappatura termica degli oggetti. Entrambi i sensori con le relative schede vengono utilizzate per acquisire immagini degli oggetti usati nelle prove da studiare nei test sperimentali eseguiti.
Una volta che la vision shield e il sensore hanno acquisito le immagini desiderate, tramite la comunicazione seriale esse vengono inviate all'ambiente software Matlab cosí che sará possibile elaborarle e analizzare i dati raccolti.

2.2 Sistema embedded

Arduino Portenta H7 é un sistema embedded dotato di due processori Arm Cortex ST (Cortex-M4 e Cortex-M7) che eseguono Mbed OS. Quest'ultimo é un sistema operativo embedded open source progettato specificamente per l'Internet of Things, in grado di operare in tempo reale (RTOS), progettato specificamente per i microcontrollori e per eseguire applicazioni IoT a basso consumo energetico.
I sistemi embedded sono costituiti da una combinazione di hardware e software ideata per portare a termine compiti ben specifici; i due funzionano sempre come elementi o parti di un sistema completo e da ció deriva il termine "embedded", dall'inglese incorporato.
Questi tipi di sistemi possono essere considerati come calcolatori economici e a basso consumo energetico i quali vengono integrati in altri sistemi destinati alla gestione di interfacce meccaniche o elettriche.
Tuttavia questa definizione potrebbe risultare generica e sviante poiché il numero dei dispositivi che rientrano in questa categoria é estremamente ampio e comprende una grande varietá di elementi, sia per quanto riguarda l'architettura hardware che il software.
Essendo sistemi di elaborazione, possono variare da sistemi caratterizzati dall'assenza di interfaccia utente, come nei dispositivi concepiti per operare un unico compito senza necessitá di interazioni con l'utente che non richiedono controlli da parte di quest'ultimo, fino a tecnologie complesse dotate di interfacce utente grafiche avanzate, come nei dispositivi mobili.
La complessitá varia da un singolo microcontrollore a configurazioni piú articolate con diversi processori e periferiche collegate.
I sistemi embedded presentano diverse complessitá, ma generalmente sono composti da tre elementi fondamentali:

- Hardware: La parte hardware dei sistemi embedded si basa su microprocessori e microcontrollori.

- Software e firmware: il software utilizzato nei sistemi embedded puó variare in termini di complessitá; tuttavia, i microcontrollori di livello industriale e i sistemi embedded IoT operano solitamente con software molto semplici che richiedono una quantitá minima di memoria.

- Sistema operativo in tempo reale (RTOS): un sistema operativo in tempo reale é ideato per eseguire applicazioni in tempo reale che elaborano i dati man mano che vengono ricevuti, solitamente senza ritardi di buffer. Non tutti i sistemi embedded includono un RTOS. Questi tipo di sistemi operativi in tempo reale regolano il funzionamento monitorando l'esecuzione del programma e stabilendo regole durante il processo.

La CPU del sistema embedded é generalmente un microprocessore o un microcontrollore. La principale differenza tra i due é che un microprocessore rappresenta il cuore di un computer o di un dispositivo mobile, ma necessita l'integrazione di altri componenti, come le periferiche di input e output per poter operare.
A differenza dei sopra citati microprocessori, i microcontrollori non necessitano di componenti aggiuntivi, poiché contengono giá tutto il necessario al loro interno e possono funzionare senza l'ausilio di elementi esterni. Un microcontrollore é un circuito integrato compatto progettato per controllare una specifica operazione, definita in fase di progettazione, all'interno di un sistema embedded
Questi dispositivi includono un processore, uno oscillatore di clock, il circuito di reset, memoria e periferiche di ingresso/uscita (I/O) su un singolo chip peró hanno una capacitá di calcolo limitata. I microcontrollori sono presenti in una vasta gamma di dispositivi destinati all'utilizzo di tutti i giorni come, elettrodomestici, veicoli, dispositivi medici e di telecomunicazione, sistemi di sicurezza e industriali e strumenti di rete. Si tratta essenzialmente di semplici strumenti progettati per controllare alcune funzioni di un sistema piú grande.
Ció che rende i microcontrollori ideali nei sistemi embedded sono la potenza di elaborazione, l'elevato numero di linee di I/O, la disponibilitá di diverse periferiche,

le dimensioni compatte, la comoditá dell'ambiente di sviluppo e la possbilitá di programmazione ripetibile e illimitata.
I sistemi embedded, una volta ricevuti i dati attraverso le periferiche di I/O, li elaborano tramite il loro processore centrale, immagazzinando le informazioni temporanee ricevute nella memoria dati. Il processore accede a questa memoria e utilizza le istruzioni memorizzate nella sua program memory per interpretare ed applicare i dati in arrivo.
Le porte di comunicazione fornite con i sistemi embedded consentono la trasmissione e le ricezione di dati tra il processore e i dispositivi periferici, che possono includere altri sistemi embedded, o anche un ambiente esterno. Queste operazioni avvengono utilizzando un protocollo di comunicazione definito in base al tipo di dispositivo utilizzato.[1][2]

2.3 Arduino

Il sistema scelto per il progetto é la piattaforma Arduino. Si tratta di un microcontrollore open-source, composta da una famiglia di schede elettroniche progettate per la realizzazione di prototipi personalizzati, scelti in base a requisiti di prestazione come velocitá, memoria e capacitá di connessione wireless.
La piattaforma Arduino é stata concepita per facilitare la prototipazione rapida, consentendo anche a chi non ha esperienza in elettronica o programmazione di utilizzarla, grazie all'ampia disponibilitá di documentazione e tutorial.

2.3.1 Arduino Portenta H7

Per questo progetto é stata selezionata quella che meglio soddisfa i requisiti di prestazione e funzionalitá, ovvero l'Arduino Portenta H7 mostrato in figura 2.1. Questa scheda é stata progettata per affrontare progetti complessi che richiedono un'elevata capacitá di elaborazione, con particolare attenzione al settore dell'automazione e dell'industria.

L'Arduino Portenta H7 si distingue per la capacitá di essere programmato con linguaggi di alto livello, offrendo la possibilitá di eseguire simultaneamente codice Arduino compilato e MicroPython. Questa scheda sfrutta un meccanismo di Remote Procedure Call (RPC), che consente la comunicazione tra i due core del microcontrollore senza interruzioni, permettendo di invocare funzioni sull'altro processore in modo fluido e continuo. Questa caratteristica consente un vero multitasking, ovvero la possibilitá di eseguire due script separati o integrati per applicazioni e funzioni diverse in parallelo.

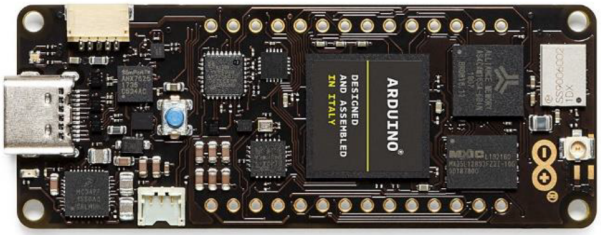

Figura 2.1: Arduino Portenta H7

Grazie alla sua architettura, la scheda Portenta H7 garantisce operazioni a bassa latenza, ottimizzando la velocitá di risposta del sistema. Questo é possibile grazie ai suoi due potenti processori: il microcontrollore STM32H747XI dual-core, che include un Cortex M7 operante a 480 MHz e un Cortex M4 a 240 MHz. Questi processori sono in grado di eseguire attivitá di alto livello e operazioni in tempo reale in parallelo, rendendo la scheda ideale per applicazioni che richiedono un'elevata potenza di calcolo e una gestione efficiente delle risorse.

In termini di connettivitá, l'Arduino Portenta H7 é equipaggiato con un modulo dual WiFi Murata 1DX e supporta la tecnologia Bluetooth 5.1, offrendo cosí opzioni avanzate di comunicazione wireless, fondamentali per l'integrazione in sistemi IoT e applicazioni di automazione remota.

In sintesi, l'Arduino Portenta H7 rappresenta una soluzione altamente versatile e potente, capace di soddisfare esigenze avanzate di elaborazione e comunicazione, il che la rende la scelta ideale per progetti che richiedono prestazioni elevate e integrazione in sistemi complessi.[3][4]

2.3.2 Portenta Vision Shield

Affinché si possano rilevare gli oggetti si necessita di fornire alla scheda una capacitá di visione attraverso la Vision Shield, raffigurata in figura 2.2 progettata per Arduino Portenta H7, dotato del modulo camera

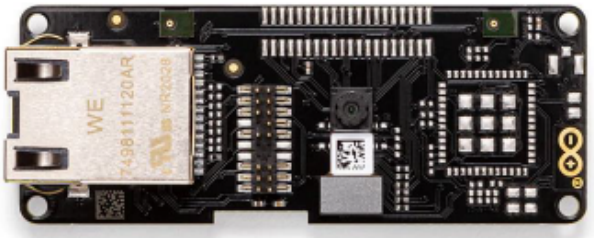

Figura 2.2: Portenta Vision Shield

La Portenta Vision Shield é un'espansione progettata per Portenta H7 che aggiunge capacitá avanzate di visione artificiale e connettivitá. Collegando la Vision Shield a Portenta H7, diventa possibile eseguire applicazioni di computer vision integrate direttamente sulla scheda, oltre a fornire opzioni di connettivitá aggiuntiva, sia tramite rete wireless che Ethernet. Questo consente alla scheda di interfacciarsi con infrastrutture locali o servizi cloud, e di attivare automaticamente il sistema in risposta al rilevamento di suoni o eventi specifici.

Portenta Vision Shield é dotata del modulo camera Himax HM-01B0, un sensore di immagine CMOS progettato specificamente per applicazioni di intelligenza artificiale che richiedono basse potenze e risoluzioni contenute. Questo modulo é capace di fornire immagini monocromatiche con una risoluzione massima di 320×320 pixel, dove ogni pixel rappresenta un valore di intensitá luminosa compreso tra 0 e 255, consentendo variazioni di luminositá tra il bianco e il nero.

In aggiunta alla cattura delle immagini, la Vision Shield offre diverse opzioni di connettivitá, rendendola particolarmente versatile per vari scenari applicativi. Per quanto riguarda la connettivitá wireless, é dotata di supporto Wi-Fi, che permette di connettersi a reti senza fili. Questo é particolarmente utile per applicazioni che richiedono

l'invio in tempo reale dei dati o il monitoraggio remoto. La scheda é anche dotata di funzionalitá Bluetooth, facilitando la comunicazione a corto raggio con dispositivi come smartphone e altri sensori.
Per applicazioni che necessitano di coprire ampie aree geografiche senza fare affidamento su infrastrutture di rete complesse, la Vision Shield é equipaggiata con supporto per LoRa (Long Range), una tecnologia che consente comunicazioni a lungo raggio e a basso consumo energetico. Inoltre, la scheda offre un'interfaccia Ethernet per connessioni cablate, caratterizzate da alta velocitá e stabilitá, ideale per situazioni in cui la connessione affidabile é fondamentale.
Un'altra funzionalitá significativa della Vision Shield é la presenza di un microfono integrato, che consente la cattura e l'elaborazione del suono. Questo microfono é di tipo MEMS (Micro-Electro-Mechanical System), noto per la sua compattezza e qualitá audio superiore, rendendolo adatto per applicazioni in cui la registrazione del suono é cruciale.
Infine, la scheda é dotata di uno slot per schede SD, che permette la memorizzazione semplice e veloce di dati come immagini, registrazioni audio e informazioni provenienti da altri sensori. Questa capacitá di archiviazione rende la Portenta Vision Shield un dispositivo versatile, in grado di gestire una varietá di dati e applicazioni in tempo reale.[5][6]

2.4 AMG8833

Il sensore Panasonic AMG 8833 8×8 gridEYE, rappresentato in figura2.3 é uno dei sensori IR termici 8×8 che il mercato offre ed é in grado di misurare temperature comprese tra 0 °C e 80 °C con un errore massimo assoluto di ±2.5 °C in campo visivo di 60 ° fino ad una distanza massima di 7 m.

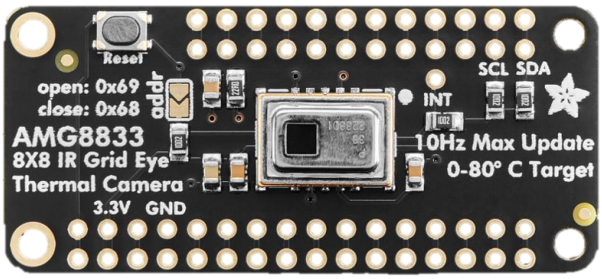

Figura 2.3: AMG8833

Il componente AMG8833 é una telecamera termica compatta e avanzata, facilmente integrabile, composta da una matrice di sensori infrarossi 8×8. Questi sensori rilevano la quantitá di raggi IR emessi da un oggetto e inviano i dati ad Arduino Portenta H7 tramite la comunicazione I2C.

Essendo un sensore a termopila a infrarossi, l'AMG8833 misura la radiazione termica proveniente da superfici e oggetti, che emettono infrarossi proporzionali alla loro temperatura, se questa é superiore allo zero assoluto. L'integrato converte questa radiazione in segnali elettrici, i quali corrispondono alla temperatura percepita. Ogni pixel del sensore cattura le radiazioni provenienti da un'area specifica del suo campo visivo, utilizzando la tecnologia delle termopile per la conversione in segnali elettrici. I valori raccolti da ciascun pixel vengono elaborati e successivamente visualizzati come una mappa termica attraverso il software MATLAB.[7][8]

La comunicazione I2C é un protocollo di comunicazione seriale bidirezionale ideata per permettere a piú dispositivi elettronici di dialogare tra loro utilizzando solo due fili, la Serial Data Line (SDL) per la trasmissione dei dati e il Serial Clock Line (SCL) per trasmettere il clock utile al sincronismo. Il protocollo é strutturato secondo l'architettura master-slave, il primo controlla il bus, invia il segnale di clock e avvia la comunicazione mentre gli slave sono quei dispositivi che rispondono alle richieste del primo.

Ogni slave é caratterizzato da un indirizzo unico il quale viene utilizzato dal master per identificare con quale dispositivo comunicare, successivamente invia un messaggio

con l'indirizzo del destinatario seguito dai dati da trasferire.
Il master inizia la comunicazione inviando una condizione di start, la quale avvisa tutti i dispositivi collegati che la trasmissione sta per iniziare, successivamente invia l'indirizzo univoco del dispositivo con il quale vuole comunicare seguito da un bit che specifica se vuole leggere o scrivere.
Il dispositivo di slave caratterizzato dall'indirizzo sopra citato risponde al master con un segnale di acknowledge (ACK) per confermare che ha ricevuto la richiesta, successivamente il master invia o riceve i dati dal dispositivo di slave, con ogni byte di dati seguito da un bit di ACK, al termine della comunicazione il dispositivo di master invia una condizione di stop per terminare la comunicazione.

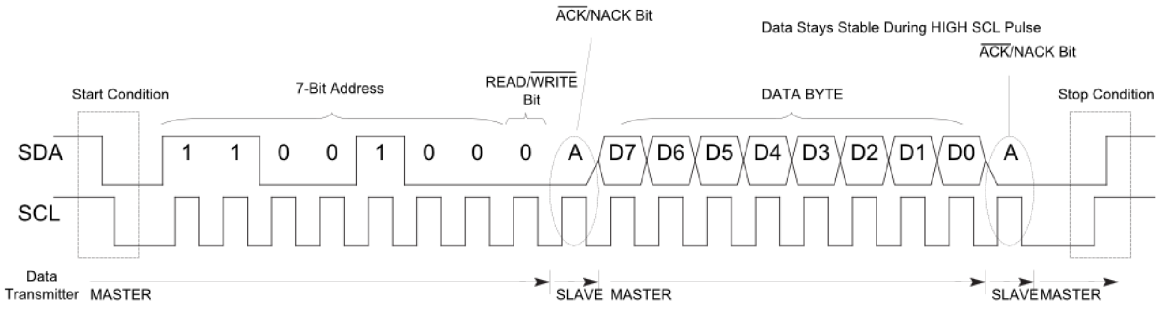

Figura 2.4: Comunicazione I2C

2.5 Software impiegati

In questa sezione vengono illustrati i software utilizzati al fine della realizzazione del sistema di acquisizione e analisi delle immagini. In particolare, sono stati impiegati Arduino IDE e MATLAB, ciascuno con un ruolo specifico nel flusso di sviluppo del sistema.

2.5.1 Arduino IDE

Arduino IDE é stato utilizzato per caricare il programma sul microcontrollore Portenta H7, dotato della Vision Shield e del sensore termico AMG8833. Grazie a questo ambiente di sviluppo é possibile gestire l'interfaccia hardware e acquisire immagini visive e termiche. La Vision Shield consente di catturare fotografie mentre il sensore AMG8833 rileva una mappa termica a bassa risoluzione, entrambi fondamentali per il processo di riconoscimento dei parametri biometrici.

Arduino IDE (Integrated Development Environment) é un ambiente di sviluppo open source utilizzato per programmare le schede elettroniche della famiglia Arduino, quale Portenta H7, e compatibili. Tale ambiente é progettato per essere intuitivo e semplice, rendendo cosí accessibile la programmazione di microcontrollori.

Una delle caratteristiche chiave di questo software é la sua capacitá di compilare il codice e caricarlo direttamente sulla scheda Arduino, l'IDE trasforma il codice sorgente in un formato eseguibile per il microcontrollore e lo carica sulla scheda attraverso una connessione seriale.

L'IDE utilizza una versione semplificata del linguaggio di programmazione C/C++, con una serie di librerie predefinite le quali facilitano l'interazione con i componenti hardware come per esempio la Vision Shield o il sensore IR AMG8833. Di seguito in figura2.5 viene riportato il diagramma di flusso del codice Arduino utilizzato, il quale verrá spiegato nel dettaglio nel capitolo successivo:

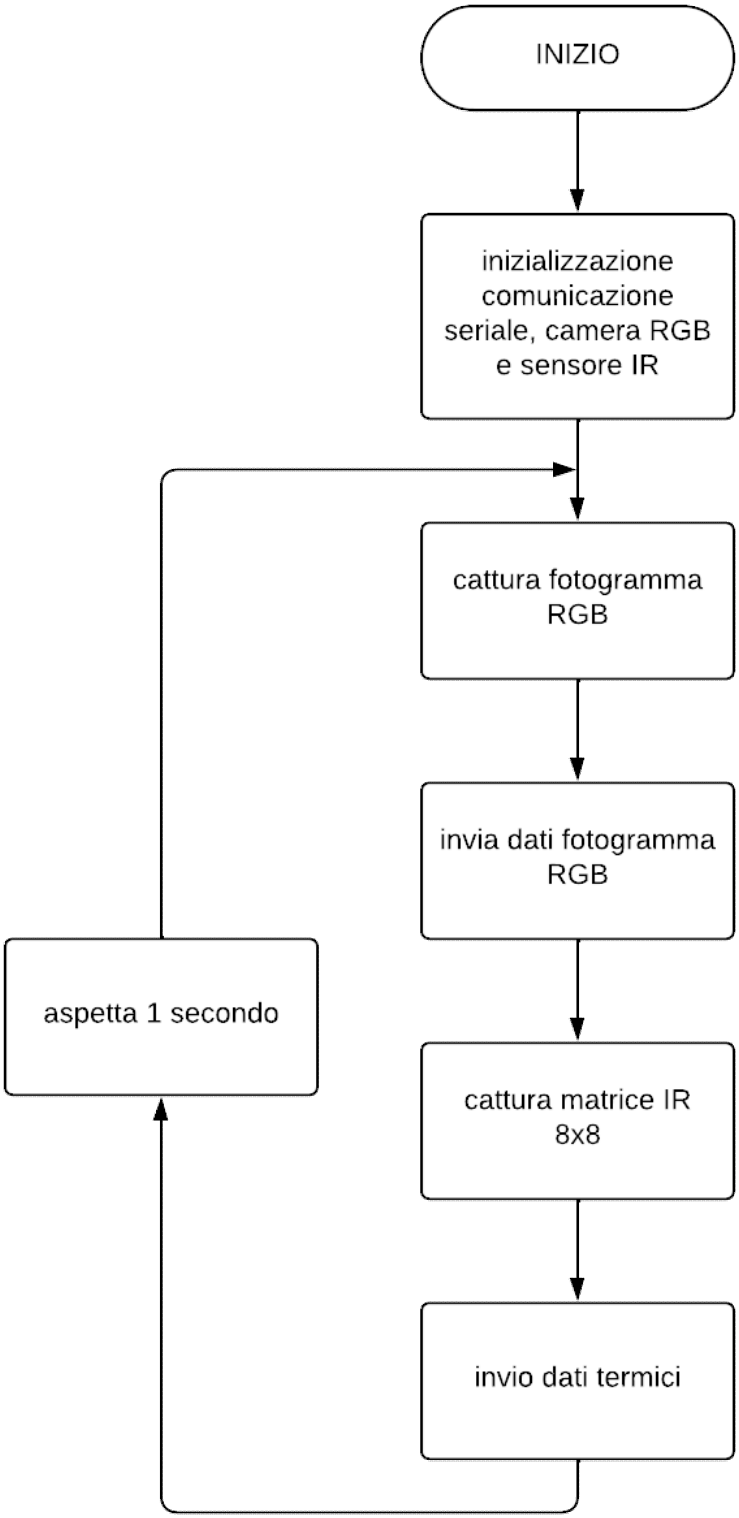

Figura 2.5: Diagramma di flusso Arduino

2.5.2 MATLAB

MATLAB, acronimo di "Matrix Laboratory", é un ambiente di programmazione e un linguaggio di alto livello creato da MathWorks, progettato principalmente per facilitare calcoli matematici complessi. Questo software é particolarmente rinomato per la sua capacitá di eseguire elaborazioni numeriche avanzate, manipolare matrici, risolvere sistemi di equazioni, e analizzare dati in modo efficiente.

Oltre alle sue potenti funzionalitá numeriche, MATLAB si distingue anche per l'eccellente supporto alla visualizzazione grafica, permettendo di generare grafici 2D e 3D in maniera semplice e personalizzabile. Queste caratteristiche lo rendono uno strumento ideale per la creazione di modelli matematici, simulazioni e applicazioni ingegneristiche complesse.

La versatilitá di MATLAB si manifesta anche nella sua capacitá di gestire una vasta gamma di discipline e ambiti scientifici. Viene utilizzato in campi che spaziano dall'ingegneria elettronica e meccanica all'intelligenza artificiale, fino a settori come la fisica, la biologia computazionale e la finanza quantitativa. Questo perché MATLAB consente agli utenti di scrivere algoritmi complessi con una sintassi relativamente semplice, riducendo cosí i tempi di sviluppo e semplificando la gestione di grandi quantitá di dati.

In questa ricerca, MATLAB é stato impiegato per l'elaborazione e l'analisi delle immagini, utilizzando il toolbox Image Processing. Questa sezione del software fornisce una serie di funzioni dedicate alla manipolazione delle immagini.

Oltre all'elaborazione delle immagini, MATLAB é stato utilizzato per l'analisi dei dati raccolti durante l'esperimento, in particolare per il calcolo della deviazione standard e dell'incertezza di tipo 'A' delle misurazioni.

Di seguito in figura2.6 viene riportato il diagramma di flusso del codice MATLAB utilizzato, il quale verrá spiegato nel dettaglio nel capitolo successivo:

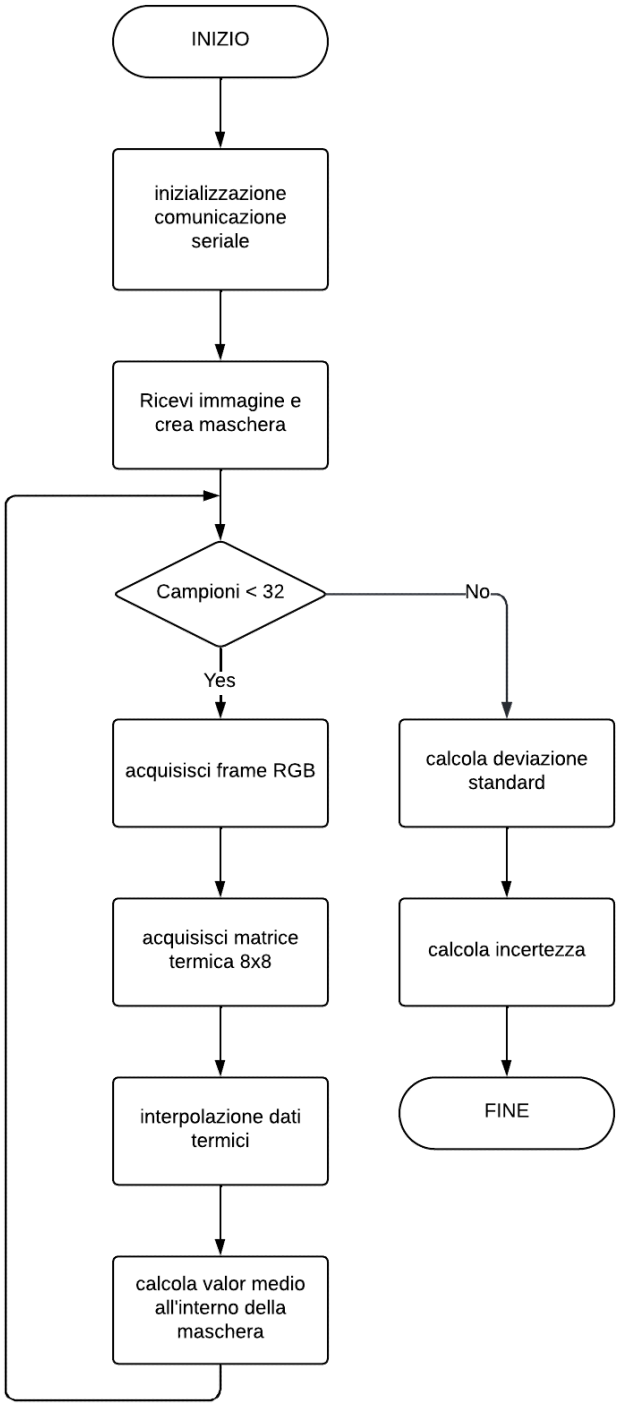

Figura 2.6: Diagramma di flusso MATLAB

Capitolo 3

Progetto ed implementazione

In questo capitolo verranno descritti in dettaglio i passaggi fondamentali che hanno portato alla realizzazione del progetto, suddivisi in fasi sequenziali e ben strutturate. L'obiettivo é quello di guidare il lettore attraverso l'intero flusso di lavoro, a partire dalla raccolta del dataset, passando per l'elaborazione e l'analisi dei dati, fino alla visualizzazione finale dei risultati.

La prima e cruciale fase del progetto é rappresentata dalla creazione di un dataset di immagini. Questa operazione costituisce il punto di partenza per l'intero studio e richiede l'acquisizione di un numero sufficiente di immagini di alta qualitá, che siano rappresentative delle condizioni studiate. Per la raccolta delle immagini, é stato utilizzato un sistema di acquisizione specifico, basato su sensori termici e visivi descritti nel capitolo precedente, il cui output é stato successivamente elaborato. Le immagini sono state accuratamente selezionate e organizzate in un formato adatto per l'elaborazione successiva.

Una volta completata la fase di acquisizione delle immagini, il focus si é spostato sulla scrittura del codice per la gestione e l'organizzazione del dataset. Questa fase é fondamentale per garantire che tutti i dati raccolti siano gestiti in modo efficace ed efficiente. Il codice scritto in ambiente MATLAB ha permesso di automatizzare il processo di lettura, salvataggio e organizzazione delle immagini acquisite, nonché di eseguire operazioni di preprocessing, come il ridimensionamento e la normalizzazione delle immagini.

Una volta completato il processo di acquisizione e calcolo dei parametri, l'attenzione si é spostata sulla visualizzazione dei dati raccolti. MATLAB, grazie alla sua ampia gamma di strumenti grafici, ha facilitato la creazione di rappresentazioni visive dei dati, come mappe termiche, grafici di distribuzione e istogrammi delle temperature rilevate. Queste rappresentazioni sono state fondamentali per l'interpretazione e la comprensione dei risultati, poiché hanno permesso di individuare rapidamente tendenze e anomalie nelle misurazioni.

3.1 Configurazione del modello

Per poter effettuare correttamente le misurazioni é necessario configurare il sistema collegando il microcontrollore Portenta H7 alla Vision Shield e al sensore infrarosso AMG8833. La connessione tra la Portenta H7 e la Vision Shield é particolarmente agevole grazie alla natura modulare del design di entrambe le schede. Queste si collegano tramite gli appositi connettori ad alta densitá, che permettono non solo il trasferimento di dati, ma forniscono anche l'alimentazione necessaria per il funzionamento della Vision Shield, il tutto senza richiedere cavi o alimentazioni esterne aggiuntive.

Diversa é la situazione quando si tratta di collegare il sensore a infrarossi AMG8833, che utilizza il protocollo di comunicazione I2C. In questo caso, é necessario effettuare una connessione manuale tra i pin del sensore e i relativi pin della Portenta H7, l'alimentazione del sensore deve essere collegata al pin 3.3 V della Portenta H7, mentre il pin GND del modulo IR va connesso al rispettivo connettore di terra della scheda. Per quanto riguarda i segnali di comunicazione, il pin SDA dell'AMG8833 va connesso al pin D14 della Portenta H7, e il pin SCL del sensore termico deve essere collegato al D15 del microcontrollore, come mostrato in figura3.1.

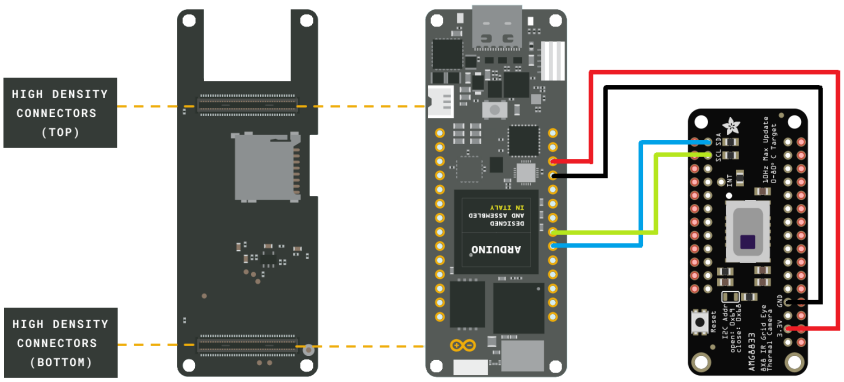

Figura 3.1: Collegamenti modello

Il passo successivo del progetto consiste nell'allineare i due sensori in modo che possano catturare l'immagine dell'oggetto simultaneamente dalla stessa posizione. Tuttavia, questa operazione si presenta come un compito difficile, se non addirittura impossibile, poiché i sensori occupano fisicamente dello spazio e non possono essere sovrapposti perfettamente.

Per affrontare questa sfida abbiamo due opzioni principali da considerare. La prima opzione prevede di posizionare la Vision Shield direttamente sopra il sensore AMG8833. Questa configurazione, peró, comporta una distanza verticale di circa 18 mm tra i due sensori, il che potrebbe limitare l'efficacia dell'allineamento.

La seconda opzione consiste nel disporre i sensori uno accanto all'altro, riducendo in questo modo la distanza tra di essi a circa 7 mm. Questa soluzione é stata preferita, poiché una distanza minore tra i sensori dovrebbe facilitare una migliore sincronizzazione nella cattura dell'immagine dell'oggetto. Tuttavia questa scelta presenta un piccolo inconveniente: l'immagine termica acquisita dal sensore dovrá essere ruotata di 90 ° per poterla allineare correttamente con l'immagine catturata dall'altro sensore. In sintesi, anche se la disposizione affiancata dei sensori presenta un vantaggio in termini di distanza ridotta, richiederá un'ulteriore fase di elaborazione dell'immagine per garantire una rappresentazione accurata e coerente dei dati termici acquisiti.

Questa fase di rotazione delle immagini é un passaggio fondamentale per integrare le informazioni provenienti dai due sensori in modo efficace e ottenere risultati ottimali. La figura3.2 mostra la configurazione reale delle schede utilizzata per le misurazioni.

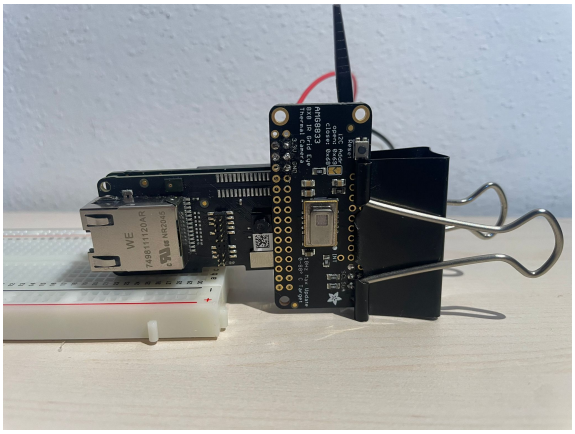

Figura 3.2: Disposizione schede

3.2 Arduino e MATLAB

Per la realizzazione del dataset di immagini, come giá detto, si utilizza la scheda Portenta H7 insieme alla Vision Shield e al sensore infrarossi AMG8833, dopo aver collegato la scheda al computer si apre l'IDE Arduino e si carica il seguente codice:

```
#include <himax.h>
#include <camera.h>
#include <Adafruit_AMG88xx.h>

HM01B0 himax; // istanza un oggetto himax per gestire il
   sensore HM01B0
Camera cam(himax); // istanza un oggetto cam, che e'la
   fotocamera associata al sensore HM01B0
```

```
FrameBuffer fb(320, 320, 1); // crea un buffer di
   dimensioni 320x320 associando 1 byte per pixel
Adafruit_AMG88xx amg; // istanza un oggetto amg per
   gestire il senosre AMG8833
float pixels[AMG88xx_PIXEL_ARRAY_SIZE]; // creo un array
   contente 64 dati di tipo float al fine di contenere le
   temperature misurate

void setup() {
  Serial.begin(921600);
  while(!Serial);
  cam.begin(CAMERA_R320x320, CAMERA_GRAYSCALE,30); //
     inizializzo il sensore HM10B0 con una risoluzione 320
     x320 pixel, in scala di grigi a 30 fps
  while(!amg.begin()){ // controllo che la comunicazione
     con il senosre AMG8833 sia pronta all'uso
  Serial.println("inizializzazione AMG8833 non riuscita");
  }
}

void loop() {
  if(cam.grabFrame(fb, 1000)==0){
    Serial.println("IMG");
    for(int i = 0; i < cam.frameSize(); i++){ // invio
       tramite seriale il buffer fb
      Serial.write(fb.getBuffer()[i]); // invio un comando
         newline per terminare la comunicazione
    }
    Serial.println();
  }
```

```
  amg.readPixels(pixels); // il senosre AMG8833 salva i
     dati istantanei della temperatura nell'array pixels
    Serial.println("IR");
    for(int i = 0; i < 64; i++){ // tramite seriale invio
       tramite seriale un pixel alla volta separato da una
        virgola
    Serial.print(pixels[i]);
    if(i < 63){
      Serial.print(",");
    }
  }
  Serial.print('\n'); // invio un comando di newline per
     terminare la comunicazione
delay(1000);
}
```

Questo codice consente ad Arduino Portenta H7 di interfacciarsi con il computer, in particolare con MATLAB, tramite la comunicazione seriale. Attraverso questa connessione, vengono trasmessi i dati necessari per raggiungere l'obiettivo, ovvero i fotogrammi RGB acquisiti dal sensore HM01B0 e le mappe termiche generate dal sensore AMG8833.
Per iniziare, il codice include le librerie

```
<himax.h> , <camera.h> e <Adafruit_AMG88xx.h>
```

che semplificano la configurazione e l'uso dei sensori impiegati.
Segue poi la completa definizione ed inizializzazione di tutte quelle variabili, dati, e buffer che saranno utilizzate dal programma nelle fasi successive:

```
HM01B0 himax
```

crea un'istanza dell'oggetto himax, responsabile della gestione del sensore RGB,

```
Camera cam
```

imposta un buffer di 320×320 pixel, con ciascun pixel memorizzato in un byte, per conservare l'immagine catturata;

```
FrameBuffer fb(320, 320, 1)
```

definisce un buffer di dimensione 320×320 pixel, ognuno dei quali occupa un solo byte, che viene utilizzato per memorizzare l'immagine catturata dalla fotocamera,

```
Adafruit_AMG88xx amg
```

crea l'oggetto amg che gestisce il sensore termico AMG8833; infine,

```
float pixels[AMG88xx_PIXEL_ARRAY_SIZE]
```

dichiara un array di 64 elementi float, destinato a memorizzare i valori di temperatura raccolti dal sensore IR.
Segue la sezione

```
void setup(){
    ...
}
```

dove vengono configurati la comunicazione seriale, la fotocamera e il sensore AMG8833. La comunicazione seriale viene impostata a una velocitá di 921600 baud, cosí da garantire l'invio dell'immagine in meno di un secondo dal microcontrollore al computer. La fotocamera é configurata per acquisire immagini in scala di grigi con una risoluzione di 320×320 pixel e una frequenza di 30 frame al secondo. Infine, l'inizializzazione del sensore AMG8833 verifica la stabilitá della comunicazione I2C per assicurarsi che il dispositivo sia pronto all'uso.
La porzione di codice che segue viene eseguita ripetutamente e contiene due sezioni principali: la prima compie l'acquisizione e la trasmissione dell'immagine RGB e la seconda ha il compito di acquisire e trasmettere i dati di temperatura,

```
void loop() {
  if(cam.grabFrame(fb, 1000)==0){
    Serial.println("IMG");
    for(int i = 0; i < cam.frameSize(); i++){
```

```
      Serial.write(fb.getBuffer()[i]);
    }
    Serial.println();
  }
  amg.readPixels(pixels);
    Serial.println("IR");
    for(int i = 0; i < 64; i++){
    Serial.print(pixels[i]);
    if(i < 63){
      Serial.print(",");
    }
  }
  Serial.print('\n');
delay(1000);
}
```

La trasmissione dell'immagine RGB é preceduta dalla stringa "IMG", che permette al codice MATLAB di riconoscere che i dati in arrivo sono relativi all'immagine, successivamente, i byte contenuti nel buffer fb vengono inviati.
Analogamente i dati termici, sono anticipati dalla stringa "IR", segnalando a MATLAB che riceverá le informazioni sulla temperatura, con ogni valore separato da una virgola.
Infine, per prevenire il sovraccarico del buffer di comunicazione seriale e garantire che i dati vengano completamente trasferiti, viene inserita una pausa di un secondo prima di iniziare una nuova acquisizione di frame e dati termici. [9]
Dopo aver caricato il codice Arduino sul microcontrollore Portenta H7, é necessario aprire MATLAB sul computer ed eseguire il seguente file:

```
clc;
clear all;
close all;
```

```
% Configurazione della comunicazione seriale
port = "COM5";  % Imposta la tua porta COM5
baudrate = 921600;
serialObj = serialport(port, baudrate);  % Crea oggetto
   per la porta seriale

% Impostazioni dell'immagine
img_width = 320;
img_height = 320;
ir_width = 8;
ir_height = 8;
img_size = img_width * img_height;  % Numero totale di
   pixel
ir_size = ir_height*ir_width;
new_ir_size = [320,320];
img_data = uint8(zeros(1, img_size));  % Buffer per i dati
    dell'immagine
ir_data = zeros(1,ir_size);
j=0;
flush(serialObj);  % Pulisci il buffer seriale

data = readline(serialObj);  % Leggi un dato per avviare
   il processo
while ~contains(data, "IMG")
    data = readline(serialObj);  % Continua a leggere fino
        a trovare "IMG"
end
% Leggi l'immagine dalla seriale
img_data = read(serialObj, img_size, "uint8");
```

```
img = reshape(img_data, [320, 320]);
img = img';  % Trasposta per correggere l'orientamento

% Visualizza la prima immagine per disegnare la maschera
figure(1);
title("Disegna la maschera sulla prima immagine");
imshow(img, []);
h = drawpolygon('LineWidth', 2, 'Color', 'yellow');  %
   Crea la maschera manualmente
mask_filled = createMask(h);  % Crea la maschera una sola
   volta
ir_average_data = [];
while j ~= 32
    data = readline(serialObj);
    if contains(data,"IMG")
        % Leggi l'immagine dalla seriale
        img_data = read(serialObj, img_size, "uint8");
        % Converti i dati letti in un array 2D (immagine)
        img = reshape(img_data, [320,320]);
        img = img';  % Trasposta per correggere l'
           orientamento
        % Visualizza l'immagine
        figure(2)
        imshow(img, []);
        title(sprintf('Immagine %d', j));
        % Salva l'immagine come file PNG
        filename = sprintf('immagine_%03d.png', j);
        saveas(gcf, filename);
        disp(['Immagine salvata: ', filename]);
        pause(0.1);
```

```
end
if contains(data,"IR")
    ir_data=readline(serialObj);
    ir_data = str2double(split(ir_data,","));
    ir_data = reshape(ir_data,[ir_width,ir_height]);

    [xOld, yOld] = meshgrid(1:size(ir_data, 2), 1:size
       (ir_data, 1));  % Griglia 8x8
    [xNew, yNew] = meshgrid(linspace(1, size(ir_data,
       2), new_ir_size(2)),linspace(1, size(ir_data,
       1), new_ir_size(1)));     % Griglia 320x320
    ir_data_interpolated = interp2(xOld, yOld, ir_data
       , xNew, yNew, 'linear');
    figure(3);
    imagesc(ir_data_interpolated);
    colormap();
    colorbar;
    title('Mappa termica AMG8833');
    filename = sprintf('mappa_termica_%03d.png', j);
    saveas(gcf,filename);
    pause(0.1); % Metti in pausa per aggiornare la
       visualizzazione

    % sovrappongo la maschera alla mappa termica
    ir_mask = ir_data_interpolated.*mask_filled;
    ir_non_zero = ir_mask(ir_mask~=0);
    ir_average = mean(ir_non_zero);
    ir_average_data = [ir_average_data,ir_average];
    j=j+1;
end
```

```
end
figure(4);
histogram(ir_average_data,16);
xlabel('temperatura');
ylabel('frequenza');
ir_dev = std(ir_average_data);
ir_inc_a = ir_dev/sqrt(length(ir_average_data));
```

Questo codice consente a MATLAB di ricevere immagini RGB e mappe termiche dalla scheda Portenta H7, salvarle, e successivamente analizzarle per ottenere informazioni utili dalle misurazioni.
Nella prima parte dello script, viene inizializzata la comunicazione seriale con il microcontrollore attraverso le seguenti istruzioni:

```
port = "COM5";
baudrate = 921600;
serialObj = serialport(port, baudrate);
```

Queste righe specificano la porta seriale utilizzata e la velocitá di trasmissione dei dati. Successivamente, viene creato l'oggetto "serialObj" che gestisce la connessione tra MATLAB e la scheda Portenta H7.
Si passa poi alla configurazione delle dimensioni sia delle immagini che delle mappe termiche:

```
img_width = 320;
img_height = 320;
ir_width = 8;
ir_height = 8;
img_size = img_width * img_height;
ir_size = ir_height*ir_width;
new_ir_size = [320,320];
```

le prime due istruzioni definiscono le dimensioni dell'immagine RGB acquisita dal sensore, 320 pixel in larghezza e 320 in altezza. I comandi successivi specificano la

risoluzione delle mappe termiche inviate dal sensore AMG8833 il quale genera una griglia 8×8 pixel. Le variabili img_size e ir_size rappresentano il numero totale di pixel delle immagini e delle mappe, infine si precisa la dimensione finale della mappa termica interpolata, la quale verrá ridimensionata a 320×320 per poter essere messa a confronto con l'immagine RGB. A seguire vengono creati i buffer per memorizzare i dati acquisiti:

```
img_data = uint8(zeros(1, img_size));
ir_data = zeros(1,ir_size);
j=0;
```

Il buffer img_data, di dimensione 1ximg_size, é inizialmente vuoto e serve per immagazzinare i dati relativi all'immagine RGB in formato byte. Allo stesso modo, ir_data é un array inizialmente vuoto che memorizzerá i dati della mappa termica a 8×8 pixel. La variabile j viene inizializzata a zero per contare quante immagini e mappe termiche verranno ricevute durante l'esecuzione.
Successivamente si ripulisce il buffer della comunicazione seriale cosí che al suo interno non siano contenuti dati riguardanti le trasmissioni precedenti i quali non sono stati ancora letti, assicurandosi che solo i nuovi dati vengano letti e processati, attraverso:

```
flush(serialObj);
```

In seguito il codice entra in un ciclo che legge i dati provenienti dalla seriale fino a quando non riceve la stringa "IMG" la quale indica l'inizio della trasmissione di un'immagine RGB da parte di Arduino Portenta H7:

```
data = readline(serialObj);
while ~contains(data, "IMG")
    data = readline(serialObj);
end

img_data = read(serialObj, img_size, "uint8");
img = reshape(img_data, [320, 320]);
img = img';
```

```
figure(1);
title("Disegna la maschera sulla prima immagine");
imshow(img, []);
h = drawpolygon('LineWidth', 2, 'Color', 'yellow');
mask_filled = createMask(h);
```

Dopo aver rilevato la stringa, il file legge i dati relativi all'immagine e li salva all'interno del buffer img_data. I dati vengono riorganizzati in una matrice bidimensionale di 320×320 pixel e trasposti per correggere l'orientamento dell'immagine. A questo punto, la prima immagine viene visualizzata e viene chiesto all'utente di disegnare una maschera su di essa, la quale verrá utilizzata successivamente nell'analisi dei dati. Dopo di ció lo script entra in un loop che verrá eseguito 32 volte:

```
data = readline(serialObj);
    if contains(data,"IMG")
        img_data = read(serialObj, img_size, "uint8");
        (immagine)
        img = reshape(img_data, [320,320]);
        img = img';
        figure(2)
        imshow(img, []);
        title(sprintf('Immagine %d', j));
        filename = sprintf('immagine_%03d.png', j);
        saveas(gcf, filename);
        disp(['Immagine salvata: ', filename]);
        pause(0.1);
    end
```

Tale porzione di codice legge i dati ricevuti in ingresso, se riceve la stringa "IMG" allora il file legge i dati relativi all'immagine e li salva all'interno del buffer img_data, il quale viene poi elaborato come nel caso della creazione della maschera, ma a differenza

del caso precedente, qui l'immagine dopo essere stata visualizzata viene salvata come file ".png".

```
if contains(data,"IR")
    ir_data=readline(serialObj);
    ir_data = str2double(split(ir_data,","));
    ir_data = reshape(ir_data,[ir_width,ir_height]);
    [xOld, yOld] = meshgrid(1:size(ir_data, 2), 1:size(
       ir_data, 1));
    [xNew, yNew] = meshgrid(linspace(1, size(ir_data, 2),
       new_ir_size(2)),linspace(1, size(ir_data, 1),
       new_ir_size(1)));
    ir_data_interpolated = interp2(xOld, yOld, ir_data,
       xNew, yNew, 'linear');
    figure(3);
    imagesc(ir_data_interpolated);
    colormap();
    colorbar;
    title('Mappa termica AMG8833');
    filename = sprintf('mappa_termica_%03d.png', j);
    saveas(gcf,filename);
    pause(0.1);
    ir_mask = ir_data_interpolated.*mask_filled;
    ir_non_zero = ir_mask(ir_mask~=0);
    ir_average = mean(ir_non_zero);
    ir_average_data = [ir_average_data,ir_average];
    j=j+1;
end
```

Dopo aver salvato l'immagine RGB, il programma riceve un input e, se l'input é esattamente la stringa "IR", inizia a leggere i dati provenienti dalla porta seriale.

Questi dati contengono informazioni sulla temperatura, organizzate sotto forma di stringa, in cui i valori sono separati da virgole. Una volta acquisita la stringa, essa viene suddivisa in singoli elementi, i quali vengono successivamente convertiti da caratteri a numeri in formato double. Analogamente a quanto fatto per l'immagine RGB, i dati della stringa vengono ridimensionati e organizzati in una matrice di dimensioni 8×8.

Successivamente, la matrice appena creata viene sottoposta a un processo di interpolazione, che la trasforma in una matrice 320×320. Interpolare una mappa termica significa prendere una mappa termica con bassa risoluzione e pochi dettagli e migliorarne la risoluzione visiva e la quantitá di dettagli. Questo avviene riempiendo automaticamente gli spazi tra i punti della matrice originale, in modo che la mappa diventi piú fluida e dettagliata.

Questa interpolazione viene eseguita con l'istruzione MATLAB "interp2", che utilizza l'interpolazione bilineare. Il valore di ogni punto intermedio viene stimato come una media pesata dei valori circostanti, ipotizzando che la variazione tra questi punti sia lineare. Dopo l'interpolazione, la nuova griglia termica viene visualizzata sullo schermo e salvata in formato ".png".

A questo punto, inizia la fase di analisi vera e propria. Viene applicata una maschera alla mappa termica appena salvata, limitando l'analisi ai valori di temperatura all'interno della maschera, che corrispondono all'oggetto d'interesse. Su questi valori si calcola la temperatura media e il risultato viene memorizzato in un array precedentemente preparato.

Infine, il programma visualizza un grafico a istogramma che rappresenta la distribuzione delle temperature medie calcolate. L'istogramma mostra la frequenza con cui compaiono i valori di temperatura nei vari intervalli misurati. Inoltre, vengono calcolate la deviazione standard e l'incertezza di tipo A, che forniscono informazioni sulla variabilitá dei dati e sull'affidabilitá delle misurazioni [10] [11].

```
figure(4);
histogram(ir_average_data,16);
xlabel('temperatura');
```

```
ylabel('frequenza');
ir_dev = std(ir_average_data);
ir_inc_a = ir_dev/sqrt(length(ir_average_data));
```

3.3 Esecuzione

Per valutare l'accuratezza delle misurazioni, é stata eseguita una serie di test utilizzando varie distanze e differenti condizioni termiche. Nella prima serie di prove, l'oggetto era mantenuto a una temperatura inferiore rispetto all'ambiente circostante, mentre nella seconda, la temperatura dell'oggetto superava quella dell'ambiente. Queste prove vengono eseguite in condizioni variabili proprio per valutare l'affidabilitá delle misure e determinare quale sia la distanza ottimale affinché il sistema operi con la massima efficienza. Le distanze testate includono:

- 15 mm
- 20 mm
- 25 mm
- 30 mm
- 35 mm
- 40 mm

In fase di avvio del programma MATLAB, questo acquisisce un fotogramma che servirá per le successive analisi. All'utente viene chiesto di tracciare manualmente una maschera sull'immagine acquisita, come mostrato in figura3.3, la quale costituirá l'area di interesse per l'elaborazione dei dati. Lo scopo é verificare come il sistema risponda alle diverse distanze, analizzando la qualitá delle misure raccolte in corrispondenza di ciascuna di esse. In questo modo, é possibile identificare la distanza alla quale il sistema fornisce le prestazioni piú precise e affidabili, migliorando cosí l'efficienza del processo complessivo.

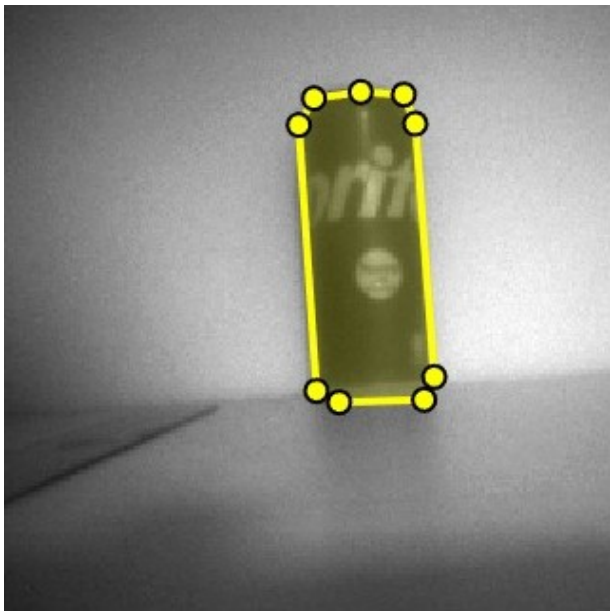

Figura 3.3: Immagine MATLAB creazione della maschera

Una volta completate le misurazioni a una specifica distanza, la procedura prevede di incrementare progressivamente la distanza e ripetere l'intera misurazione per ciascun intervallo successivo. Questo processo viene eseguito per ogni distanza prestabilita, al fine di raccogliere dati che possano essere confrontati e analizzati, permettendo di verificare l'affidabilitá delle misure ottenute alle diverse distanze.

Per garantire l'accuratezza e la validitá dei dati raccolti, é stato impiegato anche un dispositivo di controllo aggiuntivo. Nello specifico é stato utilizzato il termometro a infrarossi FLUKE 62 MAX, uno strumento noto per la sua precisione nel rilevamento delle temperature a distanza. L'uso di questo termometro ha permesso di confrontare i valori misurati con quelli rilevati dal sistema, fornendo un ulteriore livello di conferma sull'affidabilitá dei risultati ottenuti.

In questo modo, non solo si valuta la capacitá del sistema di lavorare in modo ottimale a diverse distanze, ma si garantisce anche che le misurazioni siano coerenti e verificabili attraverso un metodo di rilevamento esterno e indipendente. Il confronto con i dati forniti dal termometro IR consente di identificare eventuali discrepanze o errori, migliorando ulteriormente l'affidabilitá complessiva del processo di misurazione

Per fornire una comprensione piú chiara delle analisi eseguite nel presente lavoro, vengono riportate di seguito due immagini esemplificative: una catturata dalla fotocamera RGB, mostrata in figura3.4, e una mappa termica generata dai dati del sensore infrarossi, rappresentata in figura3.5. Queste immagini rappresentano visivamente il processo di acquisizione e fusione delle informazioni visive e termiche, che costituiscono la base della successiva analisi.

L'immagine RGB permette di identificare chiaramente le caratteristiche visive della scena, come i contorni e le forme degli oggetti o dei volti. Questo tipo di informazione é essenziale per individuare le aree di interesse, che verranno successivamente correlate con le letture di temperatura.

La mappa termica, invece, visualizza le temperature registrate dal sensore a infrarossi su scala cromatica, dove i colori piú caldi indicano temperature piú elevate, mentre i colori piú freddi corrispondono a temperature piú basse. Questa rappresentazione permette di identificare facilmente i punti di maggiore e minore emissione termica all'interno della scena.

Queste due immagini combinate offrono una visione integrata della realtá fisica e termica dell'oggetto inquadrato, dimostrando come i dati visivi e termici possano essere utilizzati insieme per ottenere una comprensione piú ricca delle caratteristiche della scena.

Figura 3.4: Immagine HM01B0, distanza: 25 cm

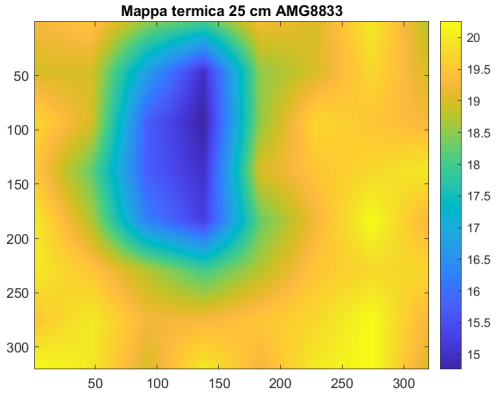

Figura 3.5: Mappa termica interpolata, distanza: 25 cm

Capitolo 4

Risultati Sperimentali

In questo capitolo vengono presentati in dettaglio i dati raccolti attraverso il sistema di misurazione integrato, che combina una fotocamera RGB con un sensore termico a infrarossi. Verranno analizzati i risultati ottenuti a partire dalle misurazioni effettuate a diverse distanze, confrontando le variazioni nella temperatura media, la dispersione dei valori e l'incertezza associata alle misure ripetute.

4.1 Dati misurati

Attraverso una serie di tabelle e istogrammi i dati raccolti verranno organizzati e illustrati in modo tale da evidenziare i principali trend e caratteristiche emersi dalle analisi.
Questi strumenti visuali aiutano a rendere piú comprensibile la distribuzione dei valori, la loro variabilitá e la precisione complessiva del sistema.
Gli istogrammi, in particolare, forniranno una rappresentazione grafica immediata delle frequenze dei valori misurati, permettendo di osservare facilmente come si distribuiscono le temperature registrate dal sensore e di identificare eventuali anomalie o dispersioni significative nei dati.
Per ogni set di misurazioni eseguite a differenti distanze dal sensore verranno riportate le temperature medie calcolate sui pixel corrispondenti agli oggetti o alle aree di interesse all'interno delle immagini.

Le tabelle presenteranno, per ciascuna misurazione, il valore medio della temperatura, la deviazione standard e l'incertezza di tipo A, evidenziando le differenze tra le misurazioni effettuate a varie distanze.

Nel contesto delle misurazioni a distanze diverse, l'analisi della deviazione standard ci permetterá di capire come la distanza dal sensore influenzi la precisione delle misurazioni.

É ragionevole aspettarsi che a distanze maggiori si possa osservare una maggiore dispersione dei dati, poiché i sensori termici tendono a perdere precisione man mano che l'oggetto si allontana.

Gli istogrammi riportati nelle figure 4.1 e 4.2 si riferiscono al primo set di misurazioni, quelle in cui l'oggetto si trova ad una temperatura minore di quella ambientale.

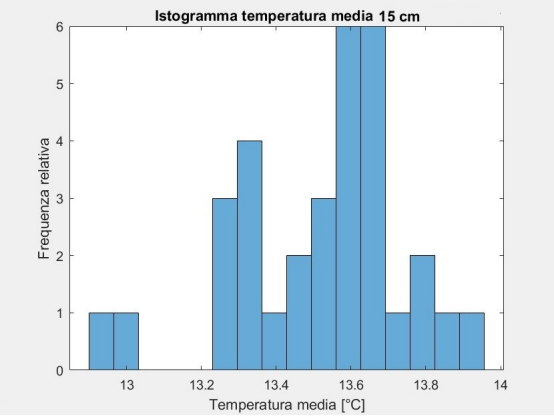

Figura 4.1: Istogramma temperatura media, distanza: 15 cm

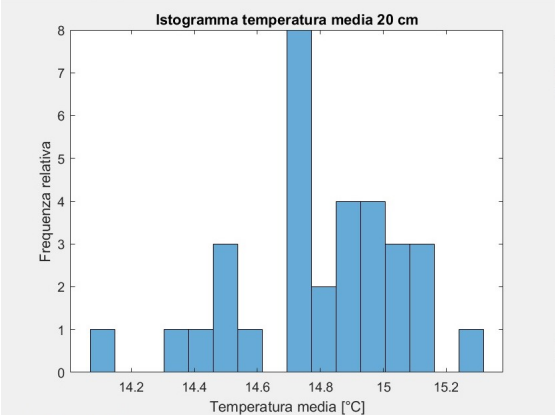

Figura 4.2: Istogramma temperatura media, distanza: 20 cm

In figura 4.1 l'istogramma delle misurazioni effettuate a 15 cm di distanza mostra un picco della temperatura attorno a 13.6 °C con una distribuzione concentrata, che indica una bassa variabilitá delle misurazioni.

Aumentando la distanza a 20 cm, come mostrato in figura 4.2, si osserva un aumento della variabilitá delle misure e un picco di temperatura che si sposta verso 14.6 °C.

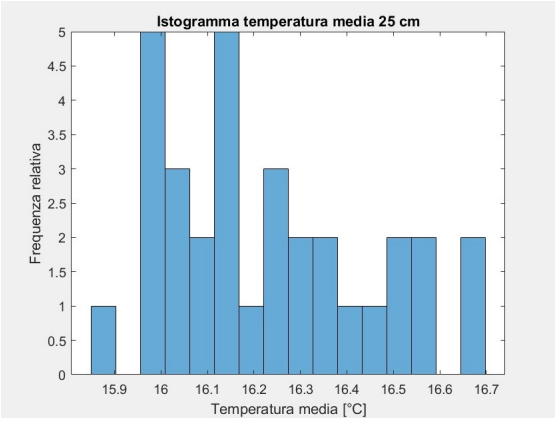

Figura 4.3: Istogramma temperatura media, distanza: 25 cm

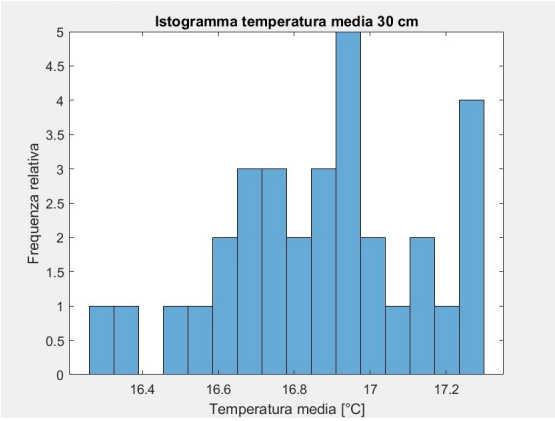

Figura 4.4: Istogramma temperatura media, distanza: 30 cm

In figura 4.3, che rappresenta le misurazioni a 25 cm, la temperatura media si stabilizza attorno a 16.1 °C e la distribuzione si allarga ulteriormente rispetto alle distanze precedenti, segnalando una maggiore dispersione delle misure. L'istogramma delle misurazioni effettuate a 30 cm, riportato in figura 4.4, evidenzia un picco intorno a 16.8 °C e una distribuzione piú ampia, segno di un'ulteriore crescita della variabilitá rispetto ai dati precedenti.

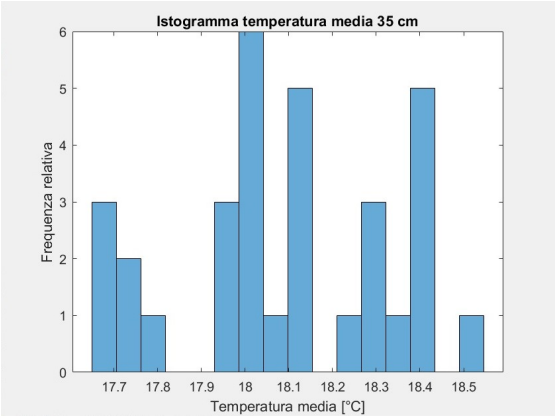

Figura 4.5: Istogramma temperatura media, distanza: 35 cm

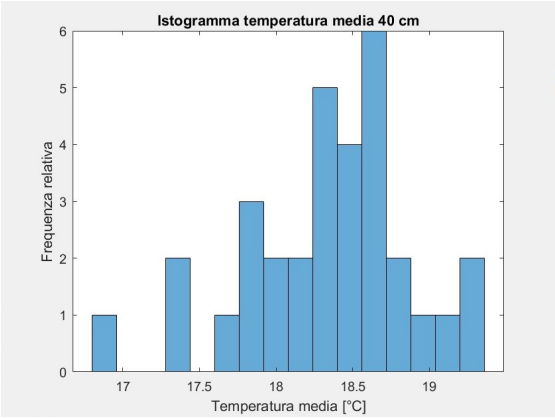

Figura 4.6: Istogramma temperatura media, distanza: 40 cm

Infine, nelle misurazioni a distanze maggiori, come si vede in figura 4.5 a 35 cm, la temperatura media si concentra su 18.1 °C e la distribuzione si amplia ancora di piú. Nella figura 4.6, relativa alla distanza di 40 cm, si registra il picco a 18.9 °C con la

distribuzione piú larga tra tutte le misurazioni, evidenziando una crescente incertezza e dispersione nei dati rilevati con l'aumento della distanza.

Nel grafico mostrato in figura 4.7, é possibile osservare la correlazione tra la temperatura media misurata e l'incertezza in relazione alla distanza che separa l'oggetto inquadrato dal sensore. Man mano che la distanza aumenta, emerge un chiaro trend: si registra un incremento dell'incertezza nelle misurazioni, mentre la temperatura media rilevata tende ad aumentare.

Questo innalzamento della temperatura misurata é, in parte, il risultato dell'aumento della temperatura dell'oggetto fino a raggiungere l'equilibrio termico con l'ambiente circostante. Quando un oggetto é piú lontano dal sensore, le letture possono essere influenzate da fattori esterni come la dispersione del calore e le variazioni di temperatura dell'aria circostante poiché l'area catturata e misurata dal sensore é maggiore.

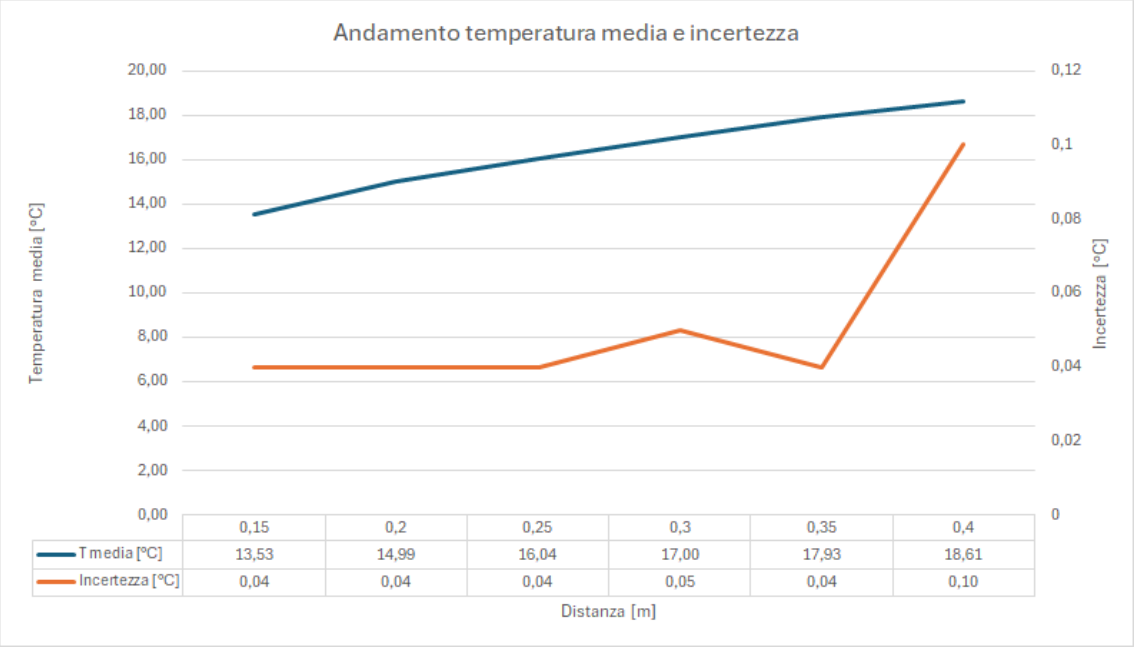

Figura 4.7: Andamento temperatura media e incertezza

Gli istogrammi riportati nelle figure 4.8 e 4.9 si riferiscono al secondo set di misurazioni, quelle in cui l'oggetto si trova ad una temperatura maggiore di quella ambientale.

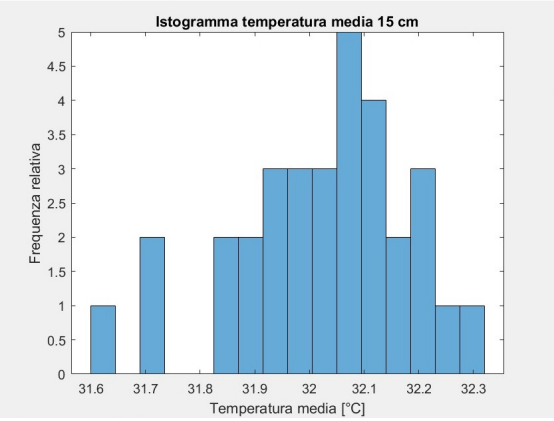

Figura 4.8: Istogramma temperatura media, distanza: 15 cm

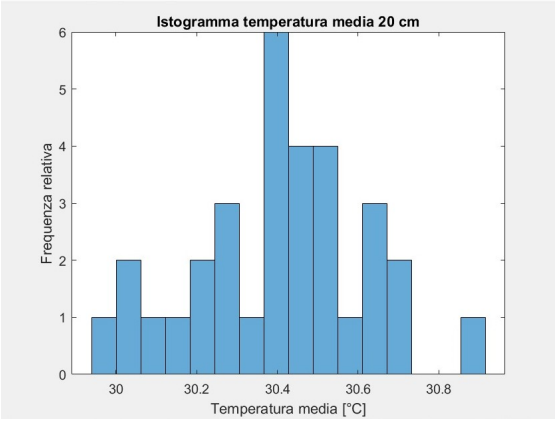

Figura 4.9: Istogramma temperatura media, distanza: 20 cm

In figura 4.8, si osserva un picco della distribuzione attorno ai 32 °C, con una certa ampiezza, che evidenzia una variabilitá nelle misurazioni. Successivamente, nella figura 4.9, la temperatura media si stabilizza intorno ai 30.4 °C, con una concentrazione piú marcata tra i 30 °C e i 30.8 °C, suggerendo una distribuzione piú omogenea.

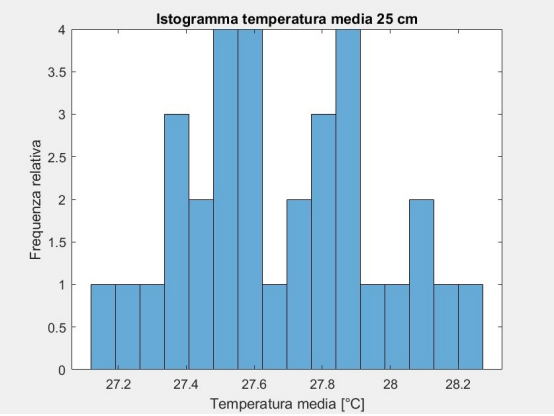

Figura 4.10: Istogramma temperatura media, distanza: 25 cm

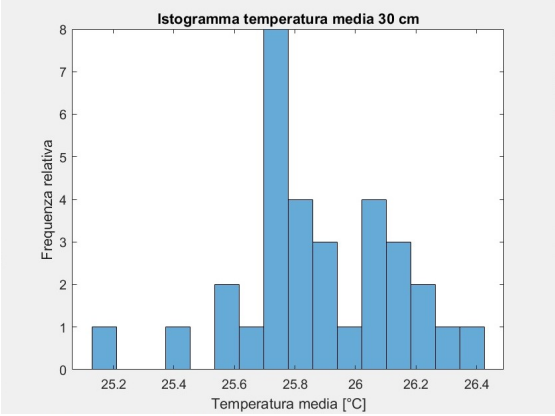

Figura 4.11: Istogramma temperatura media, distanza: 30 cm

Nella figura 4.10, invece, la distribuzione risulta leggermente piú dispersa, con valori medi che si concentrano tra 24.7 °C e 28 °C, indicando un ulteriore abbassamento delle

temperature rispetto alle precedenti misurazioni. Nell'istogramma riportato in figura 4.11, si osserva un ulteriore calo della temperatura media, con un picco significativo intorno ai 25.6 °C. La distribuzione appare piú ristretta rispetto a quelle rilevate a distanze inferiori.

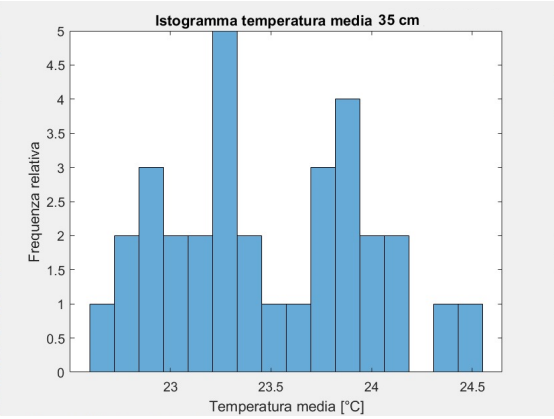

Figura 4.12: Istogramma temperatura media, distanza: 35 cm

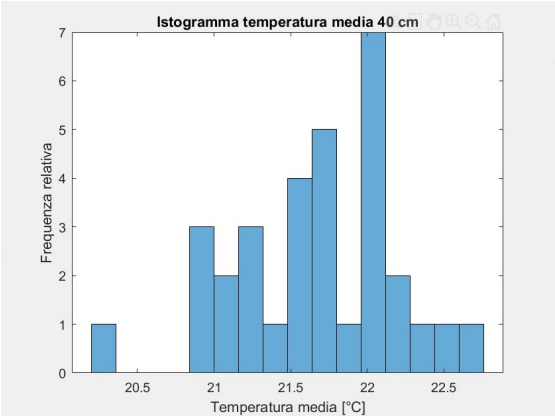

Figura 4.13: Istogramma temperatura media, distanza: 40 cm

Passando all'istogramma relativo alla distanza di 35 cm, illustrato in figura 4.12, la temperatura media presenta un picco attorno ai 25 °C, ma la distribuzione risulta essere piú ampia rispetto a quella precedente. Infine, nell'ultima misurazione, rappresentata in figura 4.13, la temperatura media si attesta su 21.6 °C e la distribuzione risulta la maggiormente ampia tra tutte le misurazioni effettuate, segnalando un ulteriore incremento della variabilitá.

Nel grafico mostrato in figura 4.14, é possibile osservare la correlazione tra la temperatura media misurata e l'incertezza in relazione alla distanza che separa l'oggetto inquadrato dal sensore. Man mano che la distanza aumenta, emerge un chiaro trend: si registra un incremento dell'incertezza nelle misurazioni, mentre la temperatura media tende a diminuire.

Questa diminuzione della temperatura misurata é, in parte, il risultato dell'abbassamento della temperatura dell'oggetto fino a raggiungere l'equilibrio termico con l'ambiente circostante. Quando un oggetto é piú lontano dal sensore, le letture possono essere influenzate da fattori esterni come la dispersione del calore e le variazioni di temperatura dell'aria circostante.

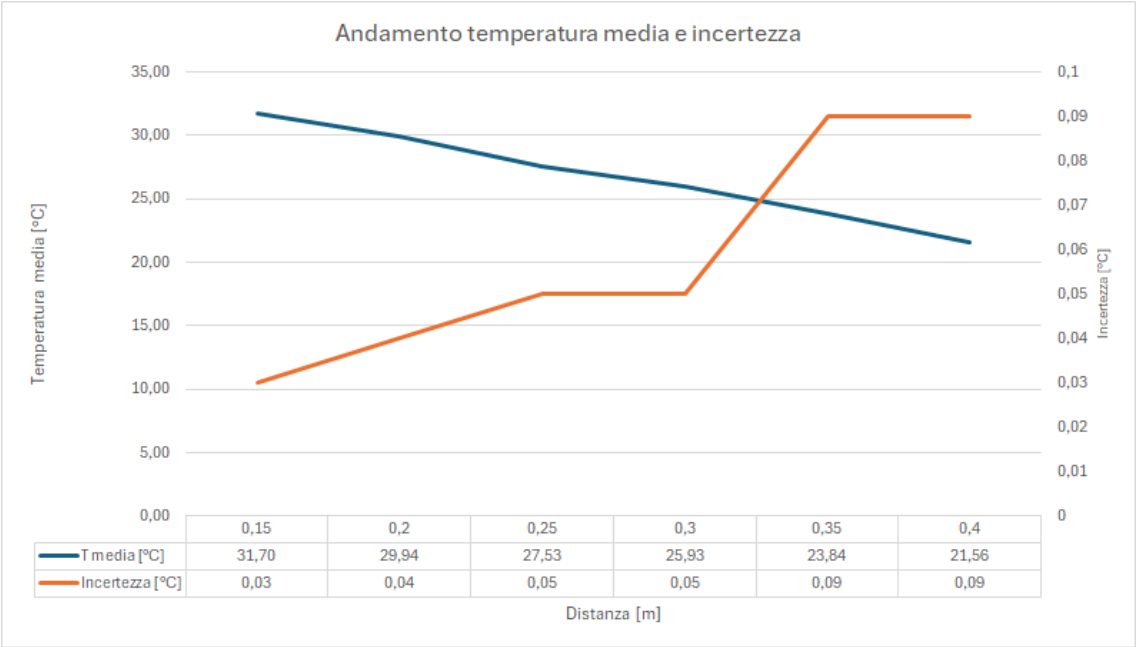

Figura 4.14: Andamento temperatura media e incertezza

Capitolo 5

Conclusioni

Nel capitolo seguente verranno esaminati in dettaglio sia le conclusioni che si possono trarre dall'analisi e dalle sperimentazioni fin qui condotte, che i possibili sviluppi futuri di questo progetto. Sará illustrato come le tecnologie e le metodologie utilizzate possano essere ulteriormente migliorate e adattate per ottenere risultati piú avanzati, non solo dal punto di vista tecnico, ma anche in termini di applicabilitá in settori specifici come la diagnostica medica, l'automazione industriale e la sicurezza.

5.1 Conclusioni

I grafici 4.7 e 4.14, i quali illustrano la relazione tra temperatura media, incertezza e distanza, rivelano un trend significativo: l'affidabilitá delle misurazioni tende a diminuire all'aumentare della distanza. Quando il sensore é posizionato piú lontano dall'oggetto inquadrato, deve acquisire dati su un'area maggiormente estesa. Questo implica il rischio di includere fonti di calore non correlate o di essere influenzati da interferenze ambientali, che possono alterare le letture.
Questo fenomeno non solo complica il processo di misurazione, ma porta anche a un incremento dell'incertezza associata. É quindi fondamentale riconoscere che la distanza rappresenta una variabile critica per garantire la precisione e l'affidabilitá delle misurazioni termiche mediante sensori senza contatto.

L'aumento dell'incertezza nelle misurazioni puó essere attribuito a una serie di fattori. In primo luogo, é possibile che l'ambiente di misurazione non sia ottimale, influenzando negativamente la precisione dei dati acquisiti. Le variabili ambientali, come l'umiditá, la presenza di ostacoli fisici e le condizioni di illuminazione, possono alterare il comportamento dei sensori di temperatura senza contatto, portando a letture errate e imprecise.
Inoltre, é essenziale considerare gli errori che possono verificarsi durante il processo di interpolazione. L'interpolazione viene spesso utilizzata per ottenere valori medi o per colmare lacune nei dati. Tuttavia, piccole imprecisioni nei dati originali possono accumularsi, amplificando l'incertezza complessiva. Pertanto, per migliorare la qualitá delle misurazioni, diventa cruciale ottimizzare le condizioni ambientali durante la raccolta dei dati e perfezionare i metodi di interpolazione adottati.
Un'analisi approfondita di questi fattori non solo faciliterá una comprensione piú chiara delle origini dell'incertezza, ma fornirá anche spunti preziosi per l'ottimizzazione delle future implementazioni del sistema di rilevamento. Questi miglioramenti sono fondamentali per garantire che il sistema possa fornire misurazioni piú accurate e affidabili, contribuendo cosí a risultati piú significativi e utili nelle applicazioni pratiche.

5.2 Sviluppi futuri

Un primo possibile sviluppo futuro consiste nella realizzazione di una scheda che integri entrambi i sensori, l'HM01B0 per la parte visiva e l'AMG8833 per la rilevazione termica. Questa soluzione permetterebbe di ridurre al minimo la distanza tra i due sensori, consentendo loro di inquadrare con precisione lo stesso oggetto, migliorando l'allineamento tra le immagini visive e termiche. L'integrazione diretta in un'unica piattaforma hardware garantirebbe una maggiore coerenza nelle misurazioni e semplificherebbe il processo di sincronizzazione dei dati tra i due dispositivi.
Un secondo sviluppo potrebbe concentrarsi sull'adozione di sensori termici di qualitá superiore, caratterizzati da una maggiore uniformitá in termini di angolo di visione

e risoluzione, per ridurre l'errore generato dall'interpolazione dei dati. Utilizzando, ad esempio, sensori IR come il MLX90640, che offre una risoluzione di 32×24 pixel, o il FLIR Lepton 3.5, con una risoluzione piú elevata di 160×120 pixel, sarebbe possibile ottenere una maggiore accuratezza nelle misurazioni termiche, migliorando significativamente la qualitá dei dati e riducendo al minimo le distorsioni legate alle differenze tra i sensori.

Un ulteriore sviluppo potrebbe essere l'integrazione di algoritmi di machine learning direttamente sulla scheda Portenta H7. Questo consentirebbe di elaborare i dati raccolti dai sensori in tempo reale, migliorando la capacitá del sistema di riconoscere pattern e caratteristiche rilevanti in maniera autonoma. Implementare soluzioni di intelligenza artificiale permetterebbe non solo di migliorare l'accuratezza delle analisi, ma anche di automatizzare il processo decisionale, rendendo il sistema piú intelligente e adattabile a scenari complessi. Questo approccio potrebbe aprire la strada a nuove applicazioni avanzate, soprattutto in ambiti come la sorveglianza, l'assistenza medica e l'automazione industriale.

Bibliografia

[1] HEAVY.AI. (n.d.) Embedded Systems. [Online]. Available: https://www.heavy.ai/technical-glossary/embedded-systems

[2] Margaret Rouse. (n.d.) Embedded System. TechTarget IoT Agenda. [Online]. Available: https://www.techtarget.com/iotagenda/definition/embedded-system

[3] Arduino. (n.d.) Portenta H7 Documentation. [Online]. Available: https://docs.arduino.cc/hardware/portenta-h7/

[4] ——. (n.d.) Arduino Portenta H7. [Online]. Available: https://store.arduino.cc/products/portenta-h7

[5] ——. (n.d.) Vision Shield Documentation. [Online]. Available: https://docs.arduino.cc/hardware/portenta-vision-shield/

[6] ——. (n.d.) Arduino Portenta Vision Shield - Ethernet. [Online]. Available: https://store.arduino.cc/products/arduino-portenta-vision-shield-ethernet?selectedStore=eu

[7] Adafruit. (n.d.) Adafruit AMG8833 IR Thermal Camera Breakout. [Online]. Available: https://www.adafruit.com/product/3622#technical-details

[8] Panasonic. (2017) Grid-EYE Specifications (Reference). [Online]. Available: https://cdn-learn.adafruit.com/assets/assets/000/043/261/original/Grid-EYE_SPECIFICATIONS%28Reference%29.pdf?1498680225

[9] Arduino. (2024) Arduino Libraries. [Online]. Available: https://docs.arduino.cc/libraries/

[10] MathWorks. (n.d.) Serial Function Documentation. [Online]. Available: https://it.mathworks.com/help/matlab/ref/serial.html

[11] ——. (n.d.) Image Processing Toolbox - MATLAB. [Online]. Available: https://it.mathworks.com/products/image-processing.html