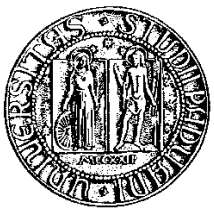


Università degli Studi di Padova

Facoltà di Ingegneria

Corso di Laurea Triennale in Ingegneria Meccatronica

Sistema di sensing IR contactless per la rilevazione di principi d'incendio elettronici

TESI DI LAUREA TRIENNALE

Relatore: Prof. Alessandro Sona

Laureandi: Riccardo Abbate, Tiziano Costa, Filippo Da Pra, Nicolò Ghiotto

Sommario

Nell'ambito dei sistemi elettronici, il progressivo sviluppo delle tecnologie richiede una gestione sempre più complessa delle potenze termiche sviluppate.

Viene proposto un modello in scala ridotta funzionante che rappresenta un possibile sistema industriale di sicurezza, da implementare in ambienti a rischio elevato di incendio causato da guasti elettrici ed elettronici, come le sale server o i quadri elettrici industriali.

Il sistema utilizza sensori a infrarossi IR per identificare e localizzare principi di incendio, con la possibilità di rispondere prontamente con dispositivi antincendio integrati. La mobilità del sistema, controllata da una scheda Arduino Nano 33 BLE e servomotori, permette una copertura panoramica dell'area frontale. A differenza dei tradizionali sistemi fissi, questo offre costi ridotti, rilevazione rapida e intervento estremamente localizzato.

Autori: Riccardo Abbate, Tiziano Costa, Filippo Da Pra, Nicolò Ghiotto

Indice

Introduzione

Nell'industria moderna, per unione di elettronica e internet, diventa sempre più comune la presenza di sistemi elettronici in luoghi difficilmente raggiungibili, sia in località remote, sia in prossimità di macchinari in funzionamento. L'evoluzione dell'elettronica, la miniaturizzazione e l'aumento di potenza su area hanno portato ad una elevata generazione di calore che necessita di opportuna dissipazione. Tuttavia il calore può essere tale che in situazioni estreme vi sia innesco di fiamme dell'isolante o del componente. In un impianto industriale, l'elettronica è tra le parti meno resistenti, perciò un apparato non protetto o protetto da sistemi antincendio tradizionali può venire facilmente danneggiato dalle fiamme o dall'elevata quantità di acqua usata per estinguerle.

Solo in Italia, nell'anno 2021 sono stati richiesti interventi al Corpo Nazionale dei Vigili del Fuoco per 11129 incendi legati a cause elettriche, di cui 4122 per quadri elettrici parti di impianti elettrici, e 727 in locali quadri elettrici domestici [1]. Questi dati evidenziano come sia tuttora presente un elevato numero di impianti non protetti o protetti in modo inadeguato.

In questa tesi viene presentato un modello di sistema autonomo di rilevamento di principi d'incendio che permetta di effettuare interventi rapidi e localizzati, in modo da estinguere quanto prima la fiamma, e limitare i danni all'elettronica presente. Inoltre, viene proposta una raccolta di scenari atti a stabilire una velocità di intervento e valutare la precisione di puntamento della testa del robot. Una possibile applicazione pratica del sistema è il suo utilizzo nel caso specifico di server o armadi per apparecchiatura elettronica.

La funzionalità ed efficacia del modello sono state testate per diverse distanze e con diverse richieste di precisione, ed è risultato un tempo di puntamento inferiore ai 4 secondi nel 97% dei casi.

Il progetto si inserisce in una fascia di prezzo bassa per poter coprire una fetta di mercato in cui prezzi superiori non permetterebbero un ritorno dell' investimento, ma in cui una protezione antincendio è preferibile, se non necessaria. Inoltre, mira alla semplicità costruttiva e di montaggio, in modo da evitare lavori invasivi nell'armadio o nella stanza e da permettere agli operatori stessi di montarlo nei propri sistemi. Possibili applicazioni possono essere la protezione di armadi adibiti a server, inverter, quadri elettrici, e in generale locazioni ad alta densità di componentistica elettrica e/o elettronica.

Ipotizzando:

- di aver accesso all'armadio e di poter apportare modifiche;
- che i circuiti elettronici siano disposti in parallelo all'interno dell'armadio;
- di poter utilizzare acqua nebulizzata;
- di poter sostituire la parte di circuito interessata dalle fiamme e fare un intervento di pulizia e recupero su quella interessata dall'acqua,

è sufficiente concentrarsi sul corretto puntamento e distanziamento dalla fiamma, in quanto fattori che più incidono sulla capacità di estinzione [2].

Per il modello sono stati utilizzati sensori infrarossi contactless, schede di condizionamento, un microcontrollore Arduino Nano 33 BLE e servomotori, elementi facilmente reperibili ed economici, ma che consentono un funzionamento efficace.

In questa tesi verranno presentati in dettaglio gli elementi del sistema antincendio, in particolare i sensori, il sistema di sensing, gli aspetti realizzativi del modello e le performance ottenute.

Capitolo 1

Tecnologie utilizzate

In questo capitolo vengono presentati i sensori di fiamma utilizzati, con focus particolare sul fotodiodo e circuito di condizionamento del segnale. Inoltre, vengono riportate nozioni essenziali sulla teoria della radiazione infrarossa nel caso generico e nel caso specifico di fiamme su circuiti elettrici.

1.1 Sensore di fiamma KY-026

Il sensore di fiamma KY-026, riportato in figura 1.1, è in grado di captare i segnali infrarossi emessi da un corpo caldo.

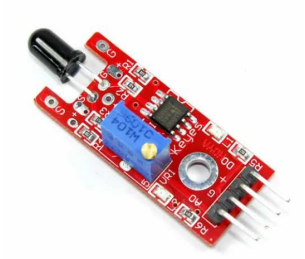


Figura 1.1: Il sensore reale [3]

É essenzialmente composto da:

- 1 ricevitore infrarosso LED da 5mm;
- 1 comparatore differenziale doppio LM393;
- 1 potenziometro a trimmer 3296W;
- 6 resistori;
- 2 indicatori LED;

- 4 pin maschio per connettori ad intaglio.

Sul piedino digitale di uscita D0 viene fornito un valore logico HIGH se viene rilevata una fiamma, LOW in caso contrario. Inoltre, mediante il PIN A0, viene fornita un'uscita analogica proporzionale all'intensità della radiazione luminosa rilevata. Tramite il potenziometro a trimmer è possibile regolare la sensibilità del sensore.

Infine, il led L1 indica la presenza della tensione di alimentazione, mentre il led L2 indica il superamento della soglia impostata.

1.1.1 Ricevitore infrarosso LED da 5mm

Il funzionamento di un ricevitore infrarosso è basato sulla capacità di alcuni materiali di assorbire la radiazione infrarossa e di generare una piccola quantità di corrente elettrica. Quando il ricevitore viene esposto alla radiazione, i fotodiodi presenti all'interno del componente convertono la radiazione in una corrente elettrica proporzionale all'intensità del segnale. Il sensore scelto è in grado di rilevare lunghezze d'onda tra $0,76 - 1,10\mu m$, ossia è ottimizzato per la lunghezza d'onda di luce infrarossa. Il suo cono di visione è di circa 60° rispetto alla direzione longitudinale, motivo per cui si è scelto di utilizzare più sensori contemporaneamente montandoli lungo una semicirconferenza, in modo tale da aumentare il range di visione del sistema.

Come si può notare da figura 1.2 sono presenti due terminali da collegare in uscita alla scheda, anodo e catodo, mentre la testa è formata da un involucro plastico nero, trasparente alla luce infrarossa, che può quindi raggiungere il fotodiodo collocato al suo interno. Alcuni parametri:

- diametro fisico del ricevitore: $5mm$;
- massimo valore di tensione inversa (polarità invertita): $5V$;
- picco di lunghezza d'onda di risposta: $940nm$.

Figura 1.2: Ricevitore infrarosso [4]

Fotodiodo

Il fotodiodo è una giunzione P-N esposta alla radiazione luminosa, in questo caso specifico attraverso una copertura trasparente agli infrarossi. Il fotodiodo utilizzato funziona in polarizzazione inversa, come è possibile osservare in figura 1.3, ossia il catodo viene collegato al terminale negativo di alimentazione, l'anodo a quello positivo. Alimentando il diodo in questo modo si genererà quindi una regione di carica spaziale: maggiore è la polarizzazione e più ampia sarà questa area.

Quando la radiazione luminosa impatta la regione di carica spaziale ionizza gli atomi generando delle coppie lacuna-elettrone. Di conseguenza, il campo elettrico esistente, dovuto alla polarizzazione inversa farà muovere le lacune verso l'anodo e gli elettroni verso il catodo, creando così una corrente fotoelettrica direttamente proporzionale all'intensità luminosa.

Se il diodo non viene illuminato, questa regione avrà pochi portatori di carica liberi e sarà presente solamente una corrente causata dalla ionizzazione termoeccitata, chiamata "corrente di buio". Ipotizzando un comportamento ideale, tale corrente dovrebbe essere nulla. Essa è proporzionale alla temperatura del diodo e in caso di livelli di luce molto bassi potrebbe nascondere la corrente fotoelettrica.

Di seguito una rappresentazione schematica del funzionamento del fotodiodo in figura 1.3.

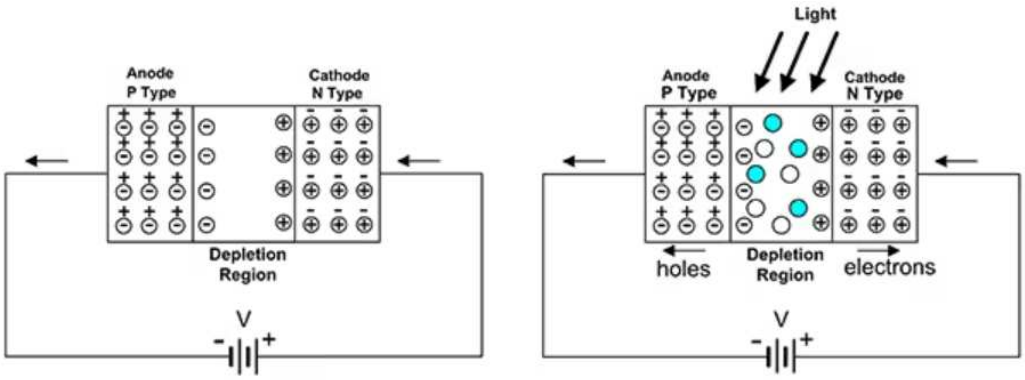


Figura 1.3: Fotodiodo polarizzazione inversa [4]

1.1.2 Comparatore differenziale doppio LM393

Viene proposto di seguito in figura 1.4 lo schema circuitale interno del comparatore differenziale doppio LM393.

L'LM393 è un componente elettronico formato da due comparatori di tensione indipendenti, progettati per operare con un'unica tensione di alimentazione. Ciò significa che invece di avere due comparatori separati, si ha un unico componente che fornisce due unità funzionali.

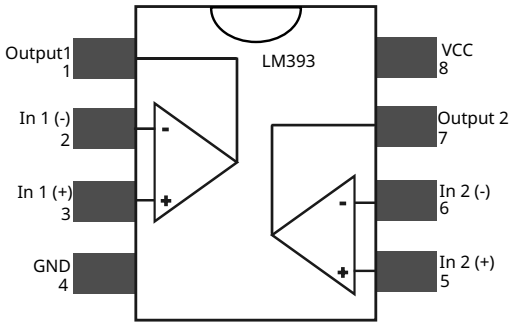


Figura 1.4: Schema comparatore LM393

Avere un comparatore differenziale doppio porta diversi vantaggi:

- efficienza nello spazio: considerando l'occupazione fisica dei componenti, si nota che avere due comparatori all'interno dello stesso chip richiede meno spazio rispetto a utilizzarne due separati;
- risparmio di costo: l'impiego di un singolo chip che contiene due comparatori potrebbe comportare un costo inferiore rispetto all'acquisto di due componenti distinti. Questo può essere un vantaggio economico nella progettazione di circuiti;
- coerenza e precisione: poiché entrambi i comparatori sono presenti sullo stesso chip, tendono ad avere caratteristiche simili e sono meno soggetti a variazioni rispetto a due componenti separati. Questo contribuisce a garantire una maggiore uniformità e precisione nelle prestazioni dei comparatori;
- semplicità di progettazione: l'utilizzo di un comparatore doppio semplifica la progettazione del circuito, riducendo la complessità del cablaggio e semplificando le connessioni tra i comparatori e altri componenti.

Dalla tabella 1.1, si osserva come tensione di offset in ingresso, potenza dissipata e tempi di risposta abbiano valori molto bassi, rendendo il comparatore una scelta valida.

Tabella 1.1: Tabella con le specifiche di LM393

Specifiche	LM393	Unità di misura
Tensione di alimentazione	2 a 30	V
Totale corrente di alimentazione	1 a 2, 5	mA
Tensione di offset in ingresso	± 4	mV
Dissipazione massima di potenza	570	mW
Tempi di risposta	1.3	μs

1.2 Schematica dei collegamenti nel sensore

La figura 1.5, riportata di seguito, rappresenta i collegamenti elettrici tra i componenti del sensore.

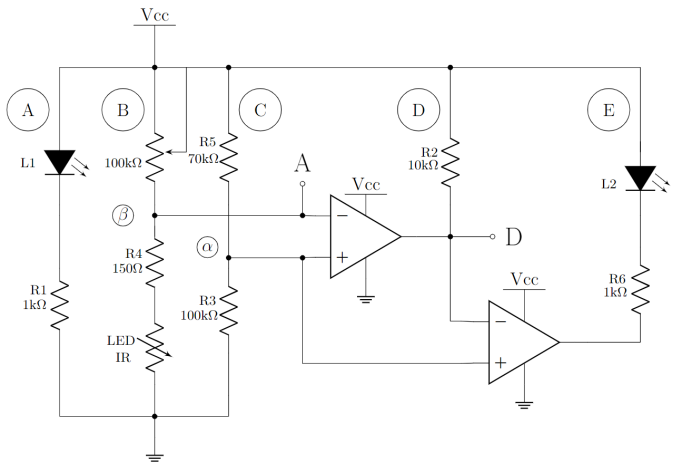


Figura 1.5: Schematica dei collegamenti elettrici nel sensore

Il circuito è alimentato dalla tensione Vcc nella parte superiore e collegato a GND (terra) nella parte inferiore. Inoltre, vale la pena notare che entrambi gli amplificatori operazionali presenti nel circuito seguono lo stesso schema di alimentazione, cioè sono anch'essi collegati a Vcc e GND. I numeri posti su di essi rappresentano gli ingressi/uscite come descritto in fig. 1.4. Partendo dal ramo A si osserva il LED L1 in serie con la resistenza R1. In questo modo L1 si accenderà in relazione alla corretta alimentazione del circuito. Spostandosi verso destra si trova il ramo B, composto da un potenziometro di resistenza variabile (regolabile tramite l'avvitamento in senso orario o antiorario), dalla resistenza R4 e da un fotodiodo, schematizzato nel circuito come una resistenza variabile. Descrivere il fotodiodo in questo modo consente di comprendere più agevolmente il suo comportamento in risposta all'illuminazione. Si può affermare, infatti, che maggiore è la luce incidente e maggiori saranno le coppie elettroni-lacune generati nel fotodiodo. Ciò causa un aumento della conduttività e conseguentemente una diminuzione della resistività. All'aumentare della radiazione incidente il diodo risulterà meno resistivo. Va, però, sottolineato che si tratta di una semplificazione concettuale per comprendere meglio il comportamento del fotodiodo all'interno del circuito. Per il vero comportamento del fotodiodo si faccia riferimento a quanto è stato descritto nel cap. 1.1.1. Il ramo B viene collegato all'ingresso invertente del primo amplificatore operazionale e all'uscita analogica. Si osserva che, sostanzialmente, si realizza un partitore di tensione con due resistenze variabili, il potenziometro e il fotodiodo. Il ramo C è anche esso composto da un partitore di tensione, con la differenza che è formato

da resistenze di valore fisso R5 e R3; sarà poi collegato agli ingressi non invertenti di entrambi gli amplificatori operazionali. Nel ramo D si trova la resistenza R2 connessa all'uscita dell'amplificatore 1. Il valore di tensione in uscita dall'amplificatore 1 corrisponderà all'uscita digitale, collegata anche all'ingresso invertente dell'amplificatore 2. Infine, il ramo E è posto tra l'alimentazione e l'uscita del secondo amplificatore. In esso sono collegati la resistenza R6 e il LED L2, con quest'ultimo che indica lo stato dell'uscita digitale.

1.2.1 Comportamento del circuito

Si consideri una tensione di alimentazione Vcc con valore compreso tra $3,3V$ e $5V$, in questo caso il LED L1 risulterà acceso. Nel ramo C, grazie al partitore di tensione, il nodo α (collegato a entrambi gli ingressi non invertenti degli amplificatori) avrà potenziale pari a:

$$V_\alpha = V_{CC} \cdot R_3/(R_3 + R_5) = V_{CC} \cdot 100/170 \approx 0.6 \cdot V_{CC} \tag{1.1}$$

Osservando il ramo B, invece, la tensione nel nodo β varia in base alla resistenza del potenziometro e del fotodiodo. Tale tensione corrisponde al valore dell'uscita analogica e all'ingresso invertente dell'amplificatore 1.
Si analizzano le cadute di tensione causate dal fotodiodo:

- caduta di tensione piccola se viene rilevata una fiamma, quindi uscita analogica più bassa;
- caduta di tensione più grande se non viene rilevata una fiamma, quindi uscita analogica più alta.

Se presente una fiamma, il potenziometro va regolato in modo tale da imporre una tensione nel nodo β minore di $0,6 \cdot V_{CC}$. In questo caso, l'output del primo comparatore sarà la tensione di alimentazione V_{CC} (HIGH), quindi anche l'uscita digitale sarà alta. Nel secondo comparatore, di conseguenza, si avrà una tensione pari a V_{CC} nell'ingresso invertente e una tensione pari a $\approx 0.6 \cdot V_{CC}$ nell'ingresso non invertente, l'uscita sarà quindi GND (LOW). Nel nodo E, ai capi di L2, ci sarà una differenza di potenziale che causerà l'accensione del LED. Se non è presente una fiamma, il potenziale sul nodo α deve essere più alto di $0,6 \cdot V_{CC}$, in questo modo l'uscita del primo comparatore sarà GND (LOW), quindi l'uscita digitale sarà GND. Questa tensione sarà presente anche nell'ingresso invertente del secondo comparatore, quindi l'uscita risulterà V_{CC}. Di conseguenza, il LED L2 risulterà spento. Si riporta in fig.1.6 un grafico dell'andamento dei valori in uscita analogico e digitale, e nella tabella 1.2 un riassunto del comportamento della scheda.

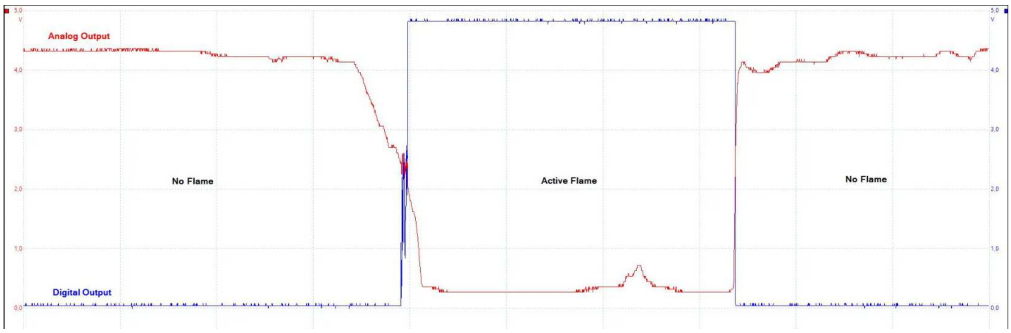


Figura 1.6: Segnali analogico e digitale in uscita dalla scheda [5]

Tabella 1.2: Riassunto comportamento scheda

Presenza fiamma	Intensità radiazione infrarossa	Output analogico	Output digitale	LED2
Sì	Alta	Valore basso	V_{CC}	ON
No	Bassa	Valore alto	GND	OFF

1.3 Teoria sulla lunghezza d'onda infrarossa

La radiazione termica costituisce uno dei modi elementari di trasmissione dell'energia termica. Essa ha la peculiarità di non necessitare di un mezzo materiale potendo avvenire anche attraverso il vuoto. Ogni corpo materiale è in grado di emettere e assorbire energia sotto forma di onde elettromagnetiche, questo permette quindi un trasferimento di energia. L'energia irradiata aumenta in modo molto significativo all'aumentare della temperatura. Un parametro fondamentale per lo studio delle onde elettromagnetiche è la lunghezza d'onda λ, essa permette di fare una suddivisione in gruppi delle onde. Questa grandezza può essere calcolata a partire dalla formula:

$$c = \lambda \cdot f \tag{1.2}$$

dove c rappresenta la velocità di propagazione in m/s ed f la frequenza in Hz dell'onda. In condizioni di vuoto la velocità di propagazione ha un valore noto e fisso che vale:

$$c_0 = 2,9979 \cdot 10^8 m/s$$

Nel caso in esame, il mezzo di trasmissione è l'aria, per la quale è nota l'emissività. Alle lunghezze d'onda infrarosse è possibile considerare c_0 come parametro senza introdurre errori significativi. Questa conclusione si deduce osservando i valori determinati dallo studio [6].

La radiazione termica è caratterizzata da lunghezze d'onda che variano tra 10^{-1} e $10^2 \mu m$.

Dalla figura 1.7 si osserva come il campo operativo sia situato nel campo delle onde infrarosse con lunghezza d'onda appena superiore al campo del visibile. La zona a infrarosso è a lunghezza d'onda maggiore di quella visibile, con un range

di $700nm \div 1mm$ o, invertendo 1.2, in frequenza $300GHz \div 430THz$. A partire da questi valori viene applicata la legge di Wien, che consente di ricavare la lunghezza d'onda per cui è massima l'emissione di un corpo nero ad una certa temperatura, $\lambda_{\max} = b/T$, con $b = 2,8977685mK$.

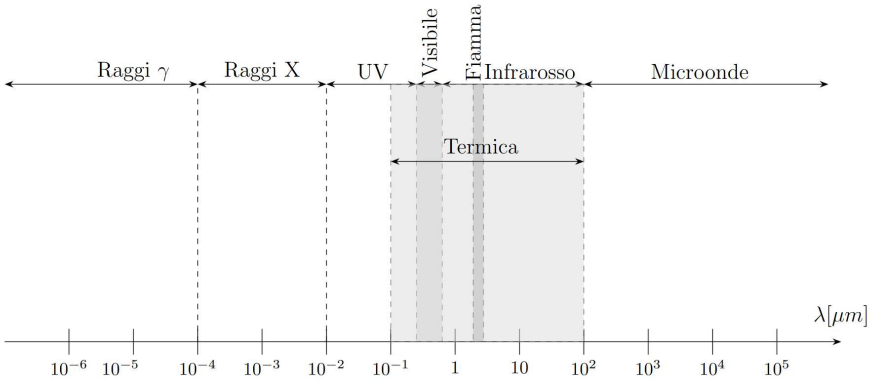


Figura 1.7: Lunghezze d'onda

Nel caso di una fiamma che investe cavi elettrici, solitamente le temperature sono comprese nell'intervallo $500K \div 1000K$ [7], per cui la lunghezza d'onda di massima emissione di un corpo nero a quella temperatura è nell'intervallo $2.9\mu m \div 5.8\mu m$, ossia la massima emissione si ha per lunghezze d'onda di radiazione infrarossa.

Tuttavia, nessun corpo può emettere più di un corpo nero, per normalizzare l'emissione dei corpi reali si utilizza l'emissività globale emisferica $\epsilon = \frac{E(T)}{(E)_n(T)}$. L'emissività dipende con proporzionalità diretta dalla temperatura, dal grado di ossidazione e dalla presenza di isolante. Per fornire un ordine di grandezza, nel caso del rame ossidato si ha $\epsilon > 0.61$ per $T > 200°C$ [8].

Alle temperature di fiamma considerate, il rame si ossida e l'isolamento presente carbonizza, perciò è corretto aspettarsi emissività più elevate di 0.61, anche sensibilmente.

Capitolo 2

Sistema progettato

In questo capitolo viene presentato lo schema a blocchi del sistema, il principio di funzionamento e i principali blocchi che lo compongono: servomotore e scheda Arduino.

2.1 Schema a blocchi

In figura 2.1 viene riportato lo schema a blocchi del sistema.

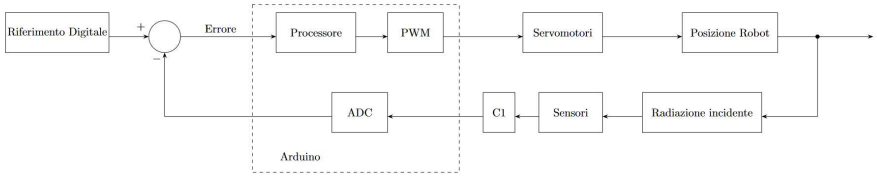


Figura 2.1: Schema a blocchi sistema

Quando il sistema viene alimentato, l'Arduino comanda ai servomotori la rotazione da effettuare in modo da muovere le varie parti del robot, così che la testa possa fare un ampio movimento.

Sulla testa sono montati a corona 5 sensori infrarossi, i cui output analogici vengono letti dall'ADC integrato nel microcontrollore Arduino.

Dall'analisi di questi valori di tensione convertiti a valori digitali è possibile determinare la presenza di fiamma, ed è inoltre possibile valutarne la posizione relativa, se è stata fatta una opportuna calibrazione.

Quando la fiamma è individuata, l'Arduino è programmato per comandare i servomotori in modo tale da puntare la testa in direzione diretta verso la fiamma e posizionarla ad una certa distanza da essa.

2.2 Servomotore

Il servomotore MG90S microservo riportato in figura 2.2 è costituito da:

- motore in corrente continua da 4.8V;
- albero di uscita;
- meccanismo di demoltiplica;
- circuito di controllo;
- potenziometro.

Figura 2.2: Servomotore MG90S microservo [9]

Lo scopo del servomotore è sostanzialmente quello di imprimere rotazione al perno d'uscita in modo da raggiungere e mantenere stabilmente la posizione desiderata. Per realizzare ciò, quindi, al motore in corrente continua viene collegato un albero allo scopo di trasmettere la sua rotazione al perno d'uscita. Il motore è collegato anche ad un meccanismo di demoltiplica costituito da ingranaggi in metallo, utilizzato per ridurre il numero di giri dell'albero motore e aumentarne sensibilmente di conseguenza la coppia in fase di rotazione. Nel servomotore, inoltre, è presente un potenziometro, sottoposto anch'esso a rotazione essendo vincolato all'albero del motore DC, che consente di conoscere precisamente la posizione raggiunta dall'albero. Il potenziometro nel servomotore, quindi, è configurato come un partitore di tensione e fornisce al circuito di controllo un valore di tensione che varia in funzione della posizione dell'albero di uscita del motore.

Nel circuito di controllo pertanto confluiscono i segnali mediante tre fili di collegamento, contraddistinti da colori diversi per distinguerli facilmente: il marrone si collega alla massa; il rosso si collega alla tensione di alimentazione, in questo caso 4.8V; l'arancione conduce il segnale PWM (Pulse Width Modulation).

2.2.1 PWM

Il segnale PWM consente di variare il periodo di accensione e di spegnimento del motore e di controllarne quindi la posizione. L'obiettivo del circuito di controllo sarà quindi quello di mettere in relazione il valore di tensione proveniente dal potenziometro con la temporizzazione degli impulsi digitali in ingresso, generando un segnale di errore nel momento in cui sia necessario correggere la tensione da fornire al motore DC e di conseguenza la posizione. Il valore del segnale di errore generato sarà direttamente proporzionale alla differenza tra la posizione del potenziometro e la temporizzazione definita dal segnale in ingresso. Viene applicato al motore un segnale ponderato in modo tale da compensare l'errore. Nel momento in cui il valore della tensione del potenziometro e la temporizzazione degli impulsi digitali coincidono, il segnale di errore viene annullato e il motore si ferma nella posizione desiderata. Il servomotore MG90S microservo è contraddistinto da una rotazione complessiva di 180 gradi, 90 per direzione. Il periodo del segnale PWM è 20ms, la frequenza associata è dunque pari a 50Hz, mentre la regolazione dell'angolo avviene con impulsi compresi fra 1 e 2ms. In particolare la posizione "0", che è quella centrale, è associata ad un impulso pari a 1.5ms con un duty cycle di 0.075. Quando il perno è ruotato completamente verso destra l'impulso è pari a 2ms, mentre nella posizione opposta è di 1ms con duty cycle pari rispettivamente a 0.05 e 0.1. Valori intermedi di duty cycle consentono di riprodurre posizioni intermedie. La velocità di rotazione dichiarata è di 60 gradi in 0.1s.

Il servomotore è inoltre contraddistinto da una dead band width (larghezza di banda morta) di $5\mu s$, in questo modo si evita che il servomotore oscilli continuamente attorno alla posizione richiesta, a causa di variazioni molto piccole nel duty cycle.

2.3 Arduino Nano 33 BLE

L'Arduino Nano 33 BLE è uno dei modelli della famiglia di microcontrollori della Arduino e rappresenta un importante progresso nel campo della prototipazione elettronica.

La scheda è disegnata e prodotta in Italia appositamente per applicazioni di automazione low-cost a bassa potenza. Presenta sensori integrati che non sono stati utilizzati nel progetto, in particolare un IMU (Inertial Measurement Unit) a 9 assi, un giroscopio e un magnetometro a 3 assi. Viene proposta in figura 2.3 una immagine della scheda L'Arduino Nano 33 BLE, che utilizza il microcontrollore nRF52840 della Nordic Semiconductors.

2.3.1 Microcontrollore nRF52840

Il microcontrollore nRF52840 è basato su un'architettura ARM Cortex-M4F, dotato di FPU (Floating Point Unit) per i calcoli in virgola mobile a singola

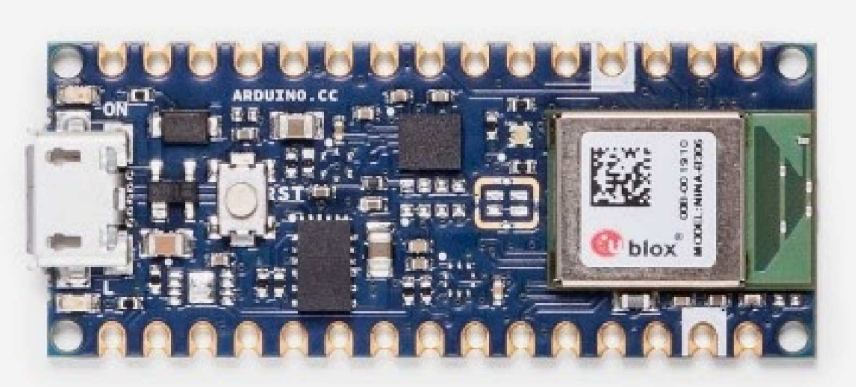


Figura 2.3: Arduino Nano 33 BLE [10]

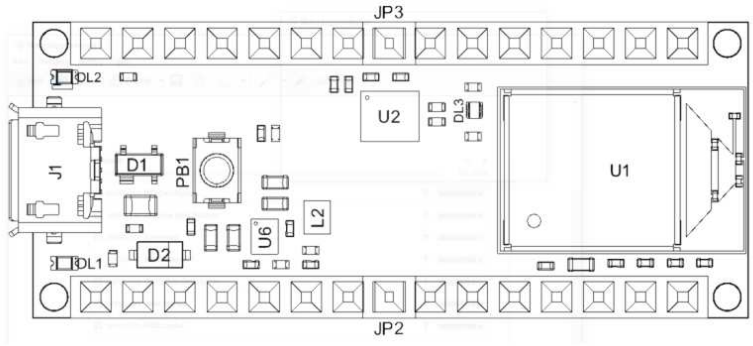
precisione, una architettura a 32 bit e clock da 64 MHz.

Le ottime capacità di calcolo a ridotti consumi energetici (212 punti CoreMark, 64 CoreMark/mA a 3 V) lo rendono adatto per l'automazione a bassa potenza. La scheda è stata scelta proprio perché grazie a questo microcontrollore è in grado di gestire applicazioni più complesse rispetto ai predecessori.

Alle motivazioni della scelta si aggiunge la presenza di una memoria dinamica da 256 KB e una memoria flash da 1 MB, necessarie per gestire i vettori di dati provenienti dall'ADC e le successive elaborazioni. Presenta, inoltre, connettività Bluetooth 5.0 Low Energy (BLE) integrata a $2.4GHz$.

2.3.2 Caratteristiche tecniche

In figura 2.4 viene riportata un'immagine delle componenti presenti sulla scheda. Segue l'elenco completo.



Board topology Top

Ref.	Description	Ref.	Description
U1	NINA-B306 Module Bluetooth® Low Energy 5.0 Module	U6	MP2322GQH Step Down Converter
U2	LSM9DS1TR Sensor IMU	PB1	IT-1185AP1C-160G-GTR Push button
DL1	Led L	DL2	Led Power

Figura 2.4: componenti installate sulla scheda [10]

- Microcontrollore: nRF52840
- Architettura: ARM Cortex-M4F
- Clock: 64 MHz
- Memoria flash: 1 MB
- Memoria dinamica: 256 KB
- NINA-B306 BLE
- LSM9DS1 IMU
- 14 pin digitali I/O (5 PWM)
- 8 pin analogici I/O
- 1 UART, 1 SPI, 1 I2C
- 1 LED integrato di accensione
- 1 LED integrato giallo al pin 13
- 1 pulsante di reset
- Alimentazione da USB: 5 V
- Alimentazione da pin Vin: 5 V - 18 V
- Dimensioni: 45 x 18 mm

2.3.3 Output digitali

Sono stati scelti i pin digitali D3, D5, D6 per il controllo in PWM attraverso il cavo di segnale (arancione) di ciascun servomotore.

2.3.4 ADC integrato

L'Arduino Nano 33 BLE utilizza un microcontrollore nRF52840 che dispone di un ADC a 12 bit integrato.

L'ADC, o Convertitore Analogico-Digitale, è un dispositivo elettronico che converte un segnale analogico di tensione in un valore digitale. Questa conversione è fondamentale in molte applicazioni, dato che i microcontrollori operano solo con segnali digitali.

Caratteristiche dell'ADC

Le principali caratteristiche da considerare dell'ADC integrato sono:

- risoluzione: determina il numero di valori discreti che un ADC può produrre. In questo caso, l'ADC a 12 bit ha $2^{12} = 4096$ valori discreti possibili, quindi avrà come uscita un valore tra 0 e 4095 (compresi);
- velocità di campionamento: indica quanti campioni al secondo l'ADC può acquisire e convertire. L'ADC utilizzato raggiunge i 200 kS/s (kilo campioni al secondo). Questa alta velocità di campionamento consente all'Arduino Nano 33 BLE di leggere segnali analogici ad alta frequenza e da molteplici input.

Utilizzo

Per utilizzare l'ADC sull'Arduino Nano 33 BLE, si può fare uso della funzione `analogRead()`. Questa funzione restituirà un valore tra 0 e 4095 che rappresenta la tensione letta sul pin specificato.

La scheda presenta 8 pin analogici I/O, organizzati come segue:

- A0 - A3, utilizzabili come input analogici o digitali;
- A4 - A5, utilizzabili come input analogici o per la comunicazione I2C;
- A6 - A7, utilizzabili come input analogici o digitali.

Sono stati scelti gli input analogici A0, A1, A2, A3 e A6 per la lettura sequenziale dei segnali provenienti dai sensori di fiamma.

Capitolo 3

Realizzazione del sistema

In questo capitolo vengono esaminati i punti salienti del codice che regola il funzionamento del sistema, viene illustrata la schematica dei collegamenti elettrici e sono descritti i collegamenti meccanici volti alla realizzazione della struttura.

3.1 Programma Arduino

L'ambiente di sviluppo scelto per la programmazione è Arduino IDE 2.2.1, molto diffuso e ben documentato, oltre che nativamente compatibile con la scheda utilizzata.

A differenza di altre tipologie di microcontrollori, Arduino Nano 33 BLE è programmabile in linguaggio C++. Ciò permette la programmazione ad oggetti e l'utilizzo di librerie. Tali strutture permettono di scrivere codice più pulito e ordinato che sfrutta in modo efficiente le risorse hardware a disposizione.

Si riporta di seguito, commentato sezione per sezione, il codice che permette il funzionamento del robot.

robot.ino La prima riga di codice è l'inclusione della libreria `Servo.h`, che permette di utilizzare e controllare i servomotori:

Listing 3.1: Include

```
#include <Servo.h>
```

Seguono le direttive di preprocessore per la definizione dei pin utilizzati per i servomotori e per i sensori. Sono definiti anche tutti i parametri utilizzati dal programma per determinare la velocità di movimento dei servomotori e la precisione del puntamento e del distanziamento dalla fiamma.

Listing 3.2: Define

```
#define BASE_PIN 3
```

```
#define TORSO_PIN 5
#define HEAD_PIN 6
#define SENSOR_PINS {A0, A1, A2, A3, A6}
#define NUM_SENSORS 5
#define BATCH_SIZE 16
#define NUMBER_OF_BITS 4
#define TARGET_DELTA 0.2
#define SEARCH_DELTA 0.1
#define MOE_HEAD 100
#define MOE 200
#define ANGLE_MARGIN 20
#define MAX_ANGLE 180.0
#define MIN_ANGLE 0.0
#define SENSOR_HIGH_THRESHOLD 3800
#define SENSOR_LOW_THRESHOLD 2000
#define SENSOR_TARGET_VALUE 1500
#define SENSOR_MARGIN 200
#define ATM_MULTIPLIER 10
#define ATM_ADDED 600
#define LARGE_SWEEP 3
#define SMALL_SWEEP 1
#define COUNTDOWN_DELTA 5
#define COUNTDOWN_ACTIVATION 500
#define COUNTDOWN_DECAY_MARGIN 50
#define COUNTDOWN_FAST_DECAY 20
#define ESCAPE_ANGLE_MAX 95
#define ESCAPE_ANGLE_MIN 85
```

Si definiscono quindi le variabili globali utilizzate dal programma.

Listing 3.3: Dichiarazione variabili

```
Servo base;
Servo torso;
Servo head;

bool intargeting = false;
u_char sensors[] = SENSOR_PINS;
double baseAngle=90.0, torsoAngle=90.0, headAngle=90.0;
signed char headDirection = -1, baseDirection = 1;
u_int countdown =  0;

int samplingMatrix[NUM_SENSORS][BATCH_SIZE];
int avgValues[NUM_SENSORS];
double maxSoftRange = 90.0 + ANGLE_MARGIN;
double minSoftRange = 90.0 - ANGLE_MARGIN;
double adaptiveTargetDelta = TARGET_DELTA;
```

setup() La prima funzione definita è `setup()`, la quale viene eseguita una sola volta all'accensione della scheda.

Listing 3.4: setup()

```
void setup() {

    base.attach(BASE_PIN);
    torso.attach(TORSO_PIN);
    head.attach(HEAD_PIN);
    writeAngles();
    analogReadResolution(12);
}
```

La funzione inizializza i servomotori, li posiziona in uno stato di partenza e imposta la risoluzione di lettura ADC dei pin analogici a 12 bit.

loop() La funzione `loop()` viene eseguita ciclicamente.

Listing 3.5: loop()

```
void loop() {

    ADCSampling();
    calculateAverage();
    modeSelection();
}
```

Loop() esegue ripetutamente fino allo spegnimento dell'Arduino le funzioni `ADCSampling()` (pag.19), `calculateAverage()` (pag.20) e `modeSelection()` (pag.28).

ADC_sampling() La funzione `ADCSampling()` gestisce gli input analogici dai sensori.

Listing 3.6: ADC_sampling()

```
void ADCSampling() {

    for (u_char j = 0; j < NUM_SENSORS; j++) {
        for (u_char i = 0; i < BATCH_SIZE; i++) {

            samplingMatrix[j][i] = analogRead(sensors[j]);
        }
    }
}
```

Esegue le seguenti operazioni:

- per ogni sensore

  - per ogni campione del batch
    - acquisisce il valore campionato e convertito dall'ADC del sensore
    - salva il valore acquisito nella matrice `samplingMatrix`

calculateAverage() La funzione `calculateAverage()` calcola la media dei valori raccolti dai sensori.

Listing 3.7: calculateAverage()

```
void calculateAverage() {

    for (u_char n = 0; n < NUM_SENSORS; n++) {
        int tempAvg = 0;
        for (u_char i = 0; i < BATCH_SIZE; i++) {
            tempAvg += samplingMatrix[n][i];
        }
        avgValues[n] = tempAvg >> NUMBER_OF_BITS;
    }
}
```

Esegue le seguenti operazioni:

- per ogni sensore
  - inizializza una variabile temporanea a 0
  - per ogni valore convertito
    - somma il valore convertito alla variabile temporanea
  - calcola la media dei valori eseguendo un bitshift verso destra (divisione più efficiente) e la salva nella variabile `avgValues`

isSensorLowest() La funzione `isSensorLowest()` verifica se il sensore passato come parametro è il sensore con il valore più basso, e che quindi percepisce più luce infrarossa.

Listing 3.8: isSensorLowest()

```
bool isSensorLowest(int sensorIndex) {

    for (u_char i = 0; i < NUM_SENSORS; i++) {
        if (i != sensorIndex && avgValues[sensorIndex] +
           SENSOR_MARGIN > avgValues[i]) {
            return false;
        }
    }
    return true;
}
```

Esegue le seguenti operazioni:

- per ogni sensore
  - se il sensore non è il sensore passato come parametro e il valore del sensore passato come parametro è maggiore del valore del sensore corrente
    - restituisce `false`
- restituisce `true`

areAllSensorsAboveThreshold() La funzione `areAllSensorsAboveThreshold()` verifica se tutti i sensori sono sopra la soglia passata come parametro.

Listing 3.9: areAllSensorsAboveThreshold()

```
bool areAllSensorsAboveThreshold(int threshold) {

    for(u_char i = 0; i < NUM_SENSORS; i++) {
      if(avgValues[i] < threshold) {
          return false;
      }
    }
    return true;
}
```

Esegue le seguenti operazioni:

- per ogni sensore
  - se il valore del sensore è minore della soglia passata come parametro
    - restituisce `false`
- restituisce `true`

isAnySensorBelowThreshold() La funzione `isAnySensorBelowThreshold()` verifica se almeno un sensore è sotto la soglia passata come parametro.

Listing 3.10: isAnySensorBelowThreshold()

```
bool isAnySensorBelowThreshold(int threshold) {

    for (u_char i = 0; i < NUM_SENSORS; i++) {
        if (avgValues[i] < threshold) {
            return true;
        }
    }
    return false;
}
```

Risulta molto simile alla `areAllSensorsAboveThreshold()` (pag.21), ma con i `return()` scambiati per realizzare la logica inversa.

angleToMicroseconds() La funzione `angleToMicroseconds()` converte l'angolo passato come parametro in microsecondi di impulso PWM.

Listing 3.11: angleToMicroseconds()

```
int angleToMicroseconds(double angle) {

    return static_cast<int>(angle * ATM_MULTIPLIER +
        ATM_ADDED);
}
```

La funzione moltiplica l'angolo passato come parametro per il fattore ATM_MULTIPLIER, somma il risultato al fattore ATM_ADDED, lo converte in intero e lo restituisce.

writeAngles() La funzione `writeAngles()` esegue il posizionamento angolare dei servomotori.

Listing 3.12: writeAngles()

```
void writeAngles() {

    baseAngle = constrain(baseAngle, MIN_ANGLE, MAX_ANGLE);
    torsoAngle = constrain(torsoAngle, MIN_ANGLE, MAX_ANGLE)
        ;
    headAngle = constrain(headAngle, MIN_ANGLE, MAX_ANGLE);

    base.writeMicroseconds(angleToMicroseconds(baseAngle));
    torso.writeMicroseconds(angleToMicroseconds(torsoAngle))
        ;
    head.writeMicroseconds(angleToMicroseconds(headAngle));
}
```

writeAngles() limita gli angoli dei servomotori tra i 0 e 180 gradi, li converte in microsecondi di impulso PWM e li genera sui rispettivi pin.

calculateTorsoAngle() La funzione `calculateTorsoAngle()` calcola l'angolo del torso come differenza tra 180 e l'angolo della base durante la ricerca.

Listing 3.13: calculateTorsoAngle()

```
void calculateTorsoAngle() {

    torsoAngle = MAX_ANGLE - baseAngle;
}
```

pointingHead() La funzione `pointingHead()` esegue il puntamento della testa verso la fiamma.

Listing 3.14: pointingHead()

```
void pointingHead() {

    int diff = 0;

    if (avgValues[2] > avgValues[1] || avgValues[2] >
       avgValues[3]) {

        diff = avgValues[0] + avgValues[1] - (avgValues[3] +
            avgValues[4]);

        if (diff > MOE_HEAD) {
            headAngle += (avgValues[0] - avgValues[1] >
               MOE_HEAD) ? LARGE_SWEEP : SMALL_SWEEP;
        } else if (diff < -MOE_HEAD) {
            headAngle -= (avgValues[4] - avgValues[3] >
               MOE_HEAD) ? LARGE_SWEEP : SMALL_SWEEP;
        }
    } else {

        diff = avgValues[1] - avgValues[3];

        if (diff > MOE_HEAD/4) {
            headAngle += adaptiveTargetDelta;
        } else if (diff < -MOE_HEAD/4) {
            headAngle -= adaptiveTargetDelta;
        }
    }
    headAngle = constrain(headAngle, MIN_ANGLE, MAX_ANGLE);
    head.write(headAngle);
}
```

Esegue le seguenti operazioni:

- se il sensore centrale è maggiore dei sensori adiacenti
  - calcola la differenza tra la somma dei valori dei sensori a destra e la somma dei valori dei sensori a sinistra
  - se la differenza è maggiore di MOE_HEAD
    - se la differenza tra i sensori a destra è maggiore di MOE_HEAD
      - incrementa l'angolo della testa di LARGE_SWEEP o di SMALL_SWEEP a seconda della differenza tra il valore del sensore esterno e quello interno rispetto a di MOE_HEAD
    - se la differenza tra i sensori a destra è minore di -MOE_HEAD

            - decrementa l'angolo con la stessa logica precedente ma tra i sensori a sinistra
        - altrimenti
            - calcola la differenza tra i valori dei sensori interni
            - se la differenza è maggiore di MOE_HEAD/4
                - incrementa l'angolo della testa di adaptiveTargetDelta
            - se la differenza è minore di -MOE_HEAD/4
                - decrementa l'angolo della testa di adaptiveTargetDelta
- limita l'angolo della testa tra 0 e 180 gradi
- genera la PWM sul pin della testa

adjustTorso() La funzione `adjustTorso()` esegue l'aggiustamento dell'angolo del torso.

Listing 3.15: adjustTorso()

```
void adjustTorso(double delta) {

    torsoAngle += delta;
    writeAngles();
    countdown += COUNTDOWN_DELTA;
}
```

La funzione esegue le seguenti operazioni:

- incrementa l'angolo del torso di delta
- chiama la funzione `writeAngles()` (pag.22)
- incrementa il countdown di COUNTDOWN_DELTA

headROMCompensation() La funzione `headROMCompensation()` esegue un aggiustamento del torso se l'angolo della testa va fuori dai limiti meccanici.

Listing 3.16: headROMCompensation()

```
void headROMCompensation() {

    if (isSensorLowest(0) && headAngle == MIN_ANGLE) {
        adjustTorso(-2 * adaptiveTargetDelta);
    }

    if (isSensorLowest(4) && headAngle == MAX_ANGLE) {
        adjustTorso(2 * adaptiveTargetDelta);
    }
}
```

Esegue le seguenti operazioni:

- se il sensore più a destra è il sensore con il valore minimo e l'angolo della testa è MIN_ANGLE
  - chiama la funzione `adjustTorso()` (pag.24) con parametro -2 volte adaptiveTargetDelta
- se il sensore più a sinistra è il sensore con il valore minimo e l'angolo della testa è MAX_ANGLE
  - chiama la funzione `adjustTorso()` (pag.24) con parametro 2 volte adaptiveTargetDelta

adjustTorsoBasedOnHeadAngle() La funzione `adjustTorsoBasedOnHeadAngle()` esegue l'aggiustamento dell'angolo del torso in base all'angolo della testa.

Listing 3.17: adjustTorsoBasedOnHeadAngle()

```
void adjustTorsoBasedOnHeadAngle() {

    if (headAngle > maxSoftRange) {
        torsoAngle += adaptiveTargetDelta;
    } else if (headAngle < minSoftRange) {
        torsoAngle -= adaptiveTargetDelta;
    }
}
```

Esegue le seguenti operazioni:

- se l'angolo della testa è maggiore di maxSoftRange
  - incrementa l'angolo del torso di adaptiveTargetDelta
- altrimenti se l'angolo della testa è minore di minSoftRange
  - decrementa l'angolo del torso di adaptiveTargetDelta

adjustBodyAndTorso() La funzione `adjustBodyAndTorso()` esegue l'aggiustamento degli angoli della base e del torso.

Listing 3.18: adjustBodyAndTorso()

```
void adjustBodyAndTorso() {

    if (headAngle < minSoftRange || headAngle > maxSoftRange
        || torsoAngle < minSoftRange || torsoAngle >
       maxSoftRange) {
        int directionMultiplier = (avgValues[2] >
           SENSOR_TARGET_VALUE) ? 1 : -1;
```

```
        if (torsoAngle < minSoftRange || torsoAngle >
            maxSoftRange) {
            baseAngle += (torsoAngle < minSoftRange ? -1 :
                1) * directionMultiplier *
                adaptiveTargetDelta;
            torsoAngle += (torsoAngle < minSoftRange ? 0.5 :
                 -0.5) * directionMultiplier *
                adaptiveTargetDelta;
        } else {
            baseAngle += (baseAngle <= 90 ? 1 : -1) *
                directionMultiplier * adaptiveTargetDelta;
        }

        if (baseAngle <= MIN_ANGLE || baseAngle >= MAX_ANGLE
            ) {
            baseAngle = (baseAngle <= MIN_ANGLE ?
                ESCAPE_ANGLE_MAX : ESCAPE_ANGLE_MIN);
            torsoAngle = (baseAngle <= MIN_ANGLE ?
                ESCAPE_ANGLE_MIN : ESCAPE_ANGLE_MAX);
            headAngle = (baseAngle <= MIN_ANGLE ? MAX_ANGLE
                : MIN_ANGLE);
            writeAngles();
        }
    }
}
```

Esegue le seguenti operazioni:

- se l'angolo della testa è minore di minSoftRange o maggiore di maxSoftRange o l'angolo del torso è minore di minSoftRange o maggiore di maxSoftRange
  - inizializza una variabile directionMultiplier a 1 se il valore del sensore centrale è maggiore di SENSOR_TARGET_VALUE, altrimenti a -1
  - se l'angolo del torso è minore di minSoftRange o maggiore di maxSoftRange
    - incrementa l'angolo della base di -directionMultiplier per adaptiveTargetDelta se l'angolo del torso è minore di minSoftRange, altrimenti di directionMultiplier per adaptiveTargetDelta
    - incrementa l'angolo del torso di mezzo directionMultiplier per adaptiveTargetDelta se l'angolo del torso è minore di minSoftRange, altrimenti di meno mezzo directionMultiplier per adaptiveTargetDelta
  - altrimenti

        * incrementa l'angolo della base di directionMultiplier per adaptiveTargetDelta se l'angolo della base è minore o uguale a 90, altrimenti di -directionMultiplier per adaptiveTargetDelta
    - se l'angolo della base è minore o uguale a MIN_ANGLE o maggiore o uguale a MAX_ANGLE
        * se l'angolo della base è minore o uguale a MIN_ANGLE
            · l'angolo della base diventa ESCAPE_ANGLE_MAX
            · l'angolo del torso diventa ESCAPE_ANGLE_MIN
            · l'angolo della testa diventa MAX_ANGLE
        * altrimenti
            · l'angolo della base diventa ESCAPE_ANGLE_MIN
            · l'angolo del torso diventa ESCAPE_ANGLE_MAX
            · l'angolo della testa diventa MIN_ANGLE
        * chiama la funzione `writeAngles()` (pag.22)

adjustDistanceFromTarget() La funzione `adjustDistanceFromTarget()` esegue l'aggiustamento della distanza dal bersaglio.

Listing 3.19: adjustDistanceFromTarget()

```
void adjustDistanceFromTarget() {

    if (abs(avgValues[2] - SENSOR_TARGET_VALUE) > MOE) {
        adjustTorsoBasedOnHeadAngle();
        adjustBodyAndTorso();
        digitalWrite(13, LOW);
    } else {
        digitalWrite(13, HIGH);
    }
    writeAngles();
}
```

Esegue le seguenti operazioni:

- se il valore assoluto della differenza tra il valore del sensore centrale e SENSOR_TARGET_VALUE è maggiore di MOE
    - chiama la funzione `adjustTorsoBasedOnHeadAngle()` (pag.25)
    - chiama la funzione `adjustBodyAndTorso()` (pag.25)
    - spegne il LED integrato
- altrimenti
    - accende il LED integrato
- chiama la funzione `writeAngles()` (pag.22)

calculateAdaptiveTargetDelta() La funzione `calculateAdaptiveTargetDelta()` calcola il delta adattivo del bersaglio.

Listing 3.20: calculateAdaptiveTargetDelta()

```
void calculateAdaptiveTargetDelta() {

    adaptiveTargetDelta = TARGET_DELTA * (abs(avgValues[2] -
        SENSOR_TARGET_VALUE)) / 1000;
}
```

modeSelection() La funzione `modeSelection()` seleziona la modalità di funzionamento del robot.

Listing 3.21: modeSelection()

```
void modeSelection() {

    if (isAnySensorBelowThreshold(SENSOR_LOW_THRESHOLD))
       countdown = COUNTDOWN_ACTIVATION;
    if (areAllSensorsAboveThreshold(SENSOR_HIGH_THRESHOLD)
       && countdown > COUNTDOWN_DECAY_MARGIN) {
       countdown -= COUNTDOWN_FAST_DECAY;
    }

    if (countdown > 0) {countdown--; targeting();} else {
       searching();}
}
```

Esegue le seguenti operazioni:

- se almeno un sensore è sotto la soglia SENSOR_LOW_THRESHOLD
  - il countdown diventa COUNTDOWN_ACTIVATION
- se tutti i sensori sono sopra la soglia SENSOR_HIGH_THRESHOLD e il countdown è maggiore di COUNTDOWN_DECAY_MARGIN
  - il countdown diminuisce di COUNTDOWN_FAST_DECAY
- se il countdown è maggiore di 0
  - il countdown diminuisce di 1
  - chiama la funzione `targeting()` (pag.29)
- altrimenti
  - chiama la funzione `searching()` (pag.29)

searching() La funzione `searching()` esegue la ricerca della presenza di un principio d'incendio.

Listing 3.22: searching()

```
void searching() {

    if (headAngle >= MAX_ANGLE || headAngle <= MIN_ANGLE) {
        headDirection *= -1;
    }

    if (baseAngle >= MAX_ANGLE || baseAngle <= MIN_ANGLE) {
        baseDirection *= -1;
    }
    headAngle += headDirection * SEARCH_DELTA * 3;
    baseAngle += baseDirection * SEARCH_DELTA;
    calculateTorsoAngle();
    writeAngles();
}
```

Esegue le seguenti operazioni:

- se l'angolo della testa è maggiore o uguale a MAX_ANGLE o minore o uguale a MIN_ANGLE
  - inverte la direzione della testa
- se l'angolo della base è maggiore o uguale a MAX_ANGLE o minore o uguale a MIN_ANGLE
  - inverte la direzione della base
- incrementa l'angolo della testa di headDirection per 3 volte SEARCH_DELTA
- incrementa l'angolo della base di baseDirection per SEARCH_DELTA
- chiama la funzione `calculateTorsoAngle()` (pag.22)
- chiama la funzione `writeAngles()` (pag.22)

targeting() La funzione `targeting()` esegue il puntamento e distanziamento dalla fiamma.

Listing 3.23: targeting()

```
void targeting() {

    calculateAdaptiveTargetDelta();
    headROMCompensation();
    pointingHead();
```

```
    if (isSensorLowest(2)) {
        countdown += COUNTDOWN_DELTA;
        adjustDistanceFromTarget();
    }
}
```

Esegue le seguenti operazioni:

- chiama la funzione `calculateAdaptiveTargetDelta()` (pag.28)
- chiama la funzione `headROMCompensation()` (pag.24)
- chiama la funzione `pointingHead()` (pag.23)
- se il sensore centrale è il sensore con il valore minimo
  - incrementa il countdown di COUNTDOWN_DELTA
  - chiama la funzione `adjustDistanceFromTarget()` (pag.27)

3.2 Schematica del circuito

In figura 3.1 viene presentato lo schema elettrico del circuito completo, realizzato utilizzando fritzing 1.0.0, un software open source per la realizzazione di schemi elettrici e circuiti stampati.

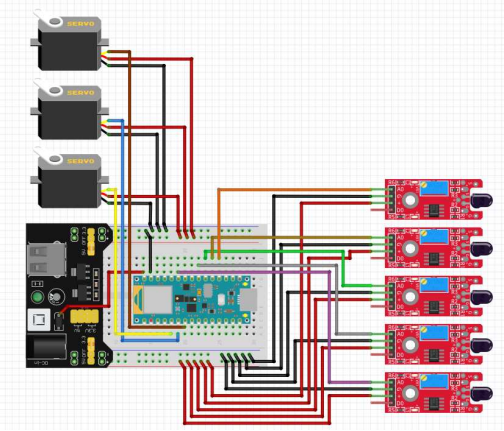


Figura 3.1: Schematica del circuito

3.2.1 Cablaggio

I cavi sono stati rappresentati seguendo i seguenti criteri:

- rosso: alimentazione;

- nero: massa (GND);
- altri colori: segnale.

Per distinguere tra loro i cavi di segnale e migliorare la leggibilità del circuito, ognuno di essi di essi è stato rappresentato un colore diverso.

Come detto precedentemente, i sensori sono collegati al GND, all'alimentazione esterna a 3.3V e ai pin analogici A0, A1, A2, A3 e A6. Anche i servomotori sono collegati al GND (comune a tutto il circuito) e all'alimentazione esterna a 5 V, mentre i pin di segnale sono collegati ai pin digitali 3, 5 e 6. Lo stesso Arduino è alimentato dalla scheda di alimentazione attraverso i pin VIN e GND, collegati rispettivamente al polo negativo del diodo SMD montato sulla scheda di alimentazione (per proteggere la scheda da eventuali inversioni di polarità) e al GND.

3.3 La struttura

La struttura del robot, riportata nelle figure 3.2 e 3.3, è costituita da:

- 3 servomotori MG90S microservo;
- 5 sensori di fiamma KY-026;
- Arduino Nano 33 BLE;
- scheda di alimentazione.

La struttura del robot è caratterizzata da tre servomotori MG90S microservo. Il primo di questi è fissato a terra ed è responsabile della rotazione di base della struttura. Il secondo servomotore è collegato al primo mediante un braccio di plastica rigida ed è caratterizzato da una rotazione opposta al primo allo scopo di garantirne l'allineamento durante la ricerca. Il terzo servomotore, invece, è fissato in posizione lineare davanti al secondo e consente la regolazione della rotazione della testa. I servomotori sono collegati all' Arduino mediante dei cavi dotati di prolunga. Sulla testa del robot sono posizionati i 5 sensori, disposti a corona al fine di minimizzare le problematiche di parallasse che sarebbero sorte in caso di disposizione lineare. I sensori, inoltre, essendo vincolati alla testa, sono messi in rotazione dalle componenti sopra citate durante la fase di ricerca di un principio d'incendio. In particolare, i sensori sono costituiti da un diodo e da una scheda di amplificazione. Il diodo è posizionato sulla testa del robot in modo da rilevare correttamente la radiazione luminosa ed è collegato alla scheda di amplificazione mediante dei cavi gialli e verdi. Anche i sensori sono collegati all'Arduino mediante dei cavi. Tutti i collegamenti elettrici sono realizzati mediante una saldatura con stagno e sono visibili nel dettaglio mediante lo schema fritzing proposto precedentemente. Infine, è presente una scheda di alimentazione per adattare i valori di tensione provenienti dall'alimentatore a quelli richiesti

dai servomotori e dai sensori pari rispettivamente a 5 e 3.3V. In particolare, la tensione uscente dall'alimentatore è pari a circa 12V che diventano poi 11.09V a seguito del passaggio attraverso un diodo che provoca un abbassamento di tensione. Differentemente dai servomotori e dai sensori che necessitano della scheda di alimentazione per adattare i valori di tensione, l'Arduino converte internamente la tensione da 11.09V a 3.3V.

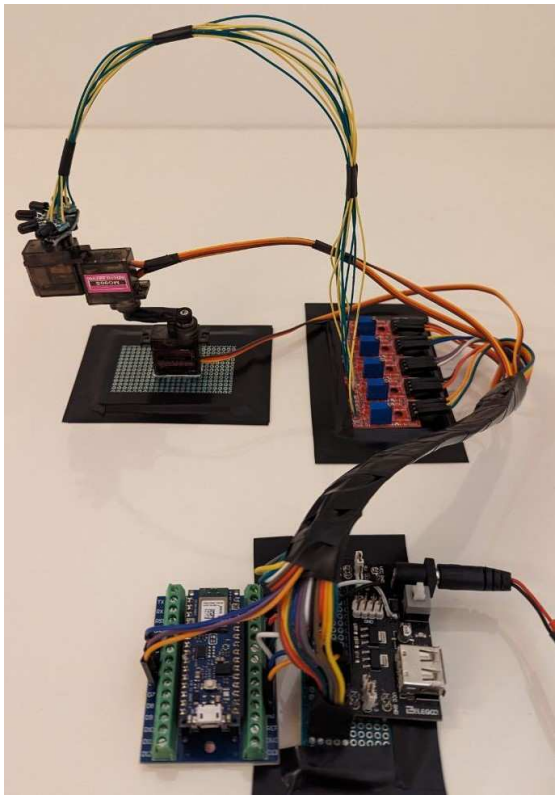


Figura 3.2: Immagine della struttura

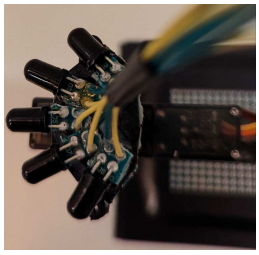


Figura 3.3: Immagine della corona di fotodiodi

Capitolo 4

Validazione

In questo capitolo vengono presentati i metodi di calibrazione dei sensori.

4.1 Calibrazione dei sensori

Per la procedura di calibrazione viene preso in considerazione inizialmente il funzionamento del singolo sensore per poi analizzare successivamente il funzionamento del robot nella sua interezza e testarne le performance. È stato realizzato appositamente il seguente codice Arduino per il singolo sensore, che viene riportato di seguito, con l'intento di valutare l'andamento di alcuni parametri al variare della distanza dallo strumento di ricezione e di conseguenza di calibrare correttamente i sensori. Viene posta particolare attenzione alla tensione d'uscita del sensore ed al valore convertito dall'ADC.

Listing 4.1: Include

```
int val;
void setup() {
  Serial.begin(9600);
  analogReadResolution(12);
}
void loop() {
int sum = 0;
for(int i = 0; i < 1024; i++) {
  sum += analogRead(A1);
}
  val = sum >> 10;
  serialOutput();
}
void serialOutput() {
      Serial.print("val");
      Serial.print(":");
      Serial.print(val);
      Serial.print(",");
```

```
        Serial.println();
}
```

4.1.1 Tensione d'uscita del sensore e valore convertito

I due grafici riportati di seguito, in figura 4.1 e 4.2, rappresentano un esempio del comportamento dei due parametri nel tempo ad una distanza dalla sorgente luminosa pari a $10cm$.

Figura 4.1: Valore convertito dall'ADC

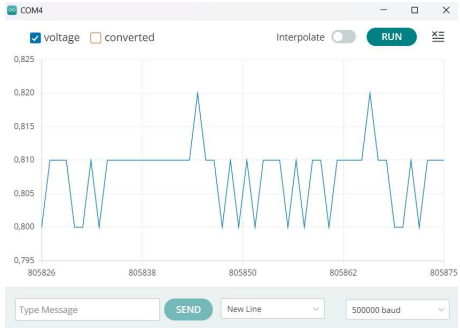


Figura 4.2: Tensione d'uscita del sensore

Nel caso in esame, i due parametri visibili nel monitor seriale dell'Arduino sono la tensione d'uscita del sensore e il valore convertito dall'ADC. Il voltaggio di uscita si differenzia dal valore convertito esclusivamente per un fattore moltiplicativo che ne adatta il range. Essendo l'ADC a 12 bit, il valore convertito sarà un numero intero compreso tra 0 e 4095, mentre il voltaggio di uscita sarà un numero decimale compreso fra 0 e $3.3V$ con una risoluzione pari a $0.01V$. Da qui in avanti sarà quindi esclusivamente il valore convertito poichè esso presenta una risoluzione migliore rispetto alla tensione d'uscita in rapporto al range complessivo.

4.1.2 Elementi di disturbo nelle misurazioni e valore convertito

Osservando il grafico del valore convertito nel monitor seriale dell'Arduino si nota come vi siano delle oscillazioni importanti al variare del tempo. Ciò è dovuto a molteplici fattori, come la presenza di rumore nel sistema di sensing e l'influenza di eventuali agenti esterni. In particolare, il rumore del sensore risulta essere un elemento indesiderato in quanto va a sovrapporsi al segnale utile causando una diminuzione dell'accuratezza della misura.

Tra gli agenti esterni che tendono ad influenzare anche in maniera significativa la misurazione dell'intensità luminosa vi è la presenza di correnti d'aria e di eventuali raggi luminosi indesiderati che potrebbero essere rilevati dal sistema di sensing. Infine l'oscillazione del valore d'uscita può essere giustificata dal fatto

che per calibrare i sensori e valutare le performance del robot è stata utilizzata una candela, che è caratterizzata da un'intensità luminosa variante leggermente nel tempo ed influenzata in maniera non trascurabile dagli agenti esterni di cui sopra. Essendo gli errori citati di tipo casuale è possibile attenuarne gli effetti: ripetendo più volte la misurazione e facendo la media dei risultati, si ottiene un valore più vicino a quello atteso per ragioni probabilistiche. Si è pertanto utilizzata la media di 1024 valori convertiti.

4.2 Procedura di calibrazione

All'inizio della procedura di calibrazione è stato impostato il potenziometro a dei valori noti in modo da comprendere il comportamento del singolo sensore al variare della distanza dalla sorgente luminosa, per poi raggiungere le condizioni desiderate, compatibili con le esigenze del robot. In particolare, è molto importante che il robot sia in grado di puntare le radiazioni luminose fino ad una certa distanza, fissata a $15cm$, per escludere elementi di disturbo come la luce solare.

4.2.1 Potenziometro $95k\Omega$

Si è impostato il potenziometro al valore massimo di resistenza, pari a circa $95k\Omega$. Ponendo il valore convertito dall'ADC sulle ordinate e la distanza in mm sulle ascisse, si è ottenuto il grafico in figura 4.3, i cui dati sono riportati nella tabella 4.1.

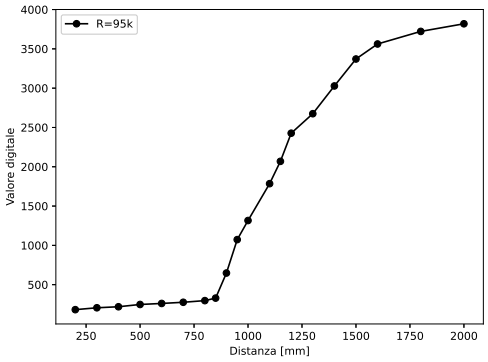


Figura 4.3: Caratteristica del sensore con potenziometro a $95k\Omega$

Si nota come il grafico sia caratterizzato da 3 zone principali: la prima a comportamento lineare caratterizzata da una pendenza bassa fino a circa $85cm$; la seconda, anch'essa sostanzialmente lineare, con pendenza notevolmente più accentuata; ed infine un'ultima fase oltre i $150cm$ caratterizzata da un assestarsi del valore convertito verso il suo valore massimo.

Distanza (mm)	200	300	400	500	600	700	800	850
	900	950	1000	1100	1150	1200	1300	1400
	1500	1600	1800	2000				
Valore convertito	182	206	219	248	261	276	297	329
	648	1073	1316	1785	2069	2427	2675	3028
	3371	3562	3722	3819				

Tabella 4.1: Valori e distanza con potenziometro a $95k\Omega$

4.2.2 Potenziometro $50k\Omega$

Successivamente, si è ripetuta la medesima procedura anche per il valore di $50k\Omega$, ottenendo una caratteristica del sensore molto simile, contraddistinta dalle medesime zone. La differenza è data esclusivamente dall'avvicinarsi del flesso a valori più bassi di distanza della fiamma e da un comportamento complessivamente traslato. Ora non si trova più a $85cm$ ma a circa $35cm$.

Le misurazioni effettuate costruiscono la caratteristica del sensore, riportata in figura 4.4.

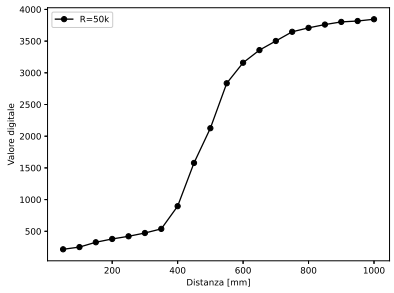


Figura 4.4: Caratteristica del sensore con potenziometro a $50k\Omega$

Viene proposta in seguito la tabella dei valori convertiti 4.2.

Distanza (mm)	50	100	150	200	250	300	350	400
	450	500	550	600	650	700	750	800
	850	900	950	1000				
Valore convertito	216	251	327	379	421	473	538	896
	1578	2126	2835	3159	3358	3502	3647	3709
	3761	3803	3817	3845				

Tabella 4.2: Valori e distanza con potenziometro a $50k\Omega$

Si osserva come sia possibile servirsi della prima zona a comportamento lineare per rilevare le sorgenti luminose desiderate, sfruttando poi la rapida crescita del valore convertito nella seconda zona per escludere le fonti di luce più lontane e di disturbo.

4.2.3 Potenziometro $5k\Omega$

Infine, per valutare il comportamento del sensore anche a resistenze minori, si è impostato il potenziometro ad un valore di circa $5k\Omega$, vicino al limite minimo, in modo da discostarsi come precedentemente di $45k\Omega$ dal valore intermedio di $50k\Omega$. Il grafico ottenuto, come si osserva in figura 4.5, è sostanzialmente il medesimo, con un flesso ora ancora più accentuato e traslato a sinistra. La caratteristica ottenuta si avvicina ora notevolmente alle esigenze dei sensori del robot. In seguito viene proposta la tabella 4.3 dei valori convertiti.

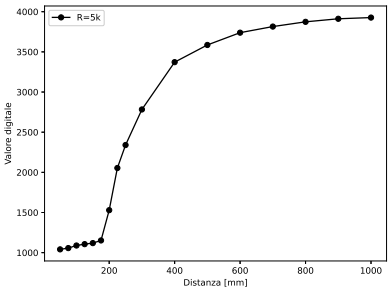


Figura 4.5: Caratteristica del sensore con potenziometro a $5k\Omega$

Distanza (mm)	50	75	100	125	150	175	200	225
	250	300	400	500	600	700	800	900
	1000							
Valore convertito	1041	1057	1089	1106	1119	1152	1529	2054
	2340	2783	3372	3586	3739	3815	3874	3912
	3927							

Tabella 4.3: Valori e distanza con potenziometro a $5k\Omega$

4.2.4 Potenziometro 0Ω

Ponendo invece il potenziometro a 0Ω si ottiene sostanzialmente un partitore di tensione con un cortocircuito. Di conseguenza la tensione analogica di uscita dal sensore è molto vicina ai $3,3V$. In questo caso non viene proposto il grafico in quanto la presenza o meno della fiamma è indistinguibile, anche indipendentemente dalla distanza.

Nella fase di programmazione del codice Arduino, si utilizzerà il valore convertito di 1500 come riferimento digitale a cui avvicinarsi. Mantenendo un certo margine di sicurezza per contrastare le eventuali oscillazioni della fiamma ed elementi di disturbo, è possibile fissare un valore di soglia a circa 2000 per stabilire la quota al di sotto della quale il robot passa dalla fase di ricerca alla fase di puntamento. Successivamente il robot potrà puntare precisamente e avvicinarsi

alla fiamma fino a raggiungere un valore pari a 1500 dal sensore centrale. Calibrando correttamente i sensori, il robot sarà in grado di individuare, puntare e distanziare la fiamma ad valore prossimo ai 15cm.

4.3 Setup

In figura 4.6 vengono infine riportate alcune immagini del setup utilizzato nella rilevazione dei dati sul singolo sensore.

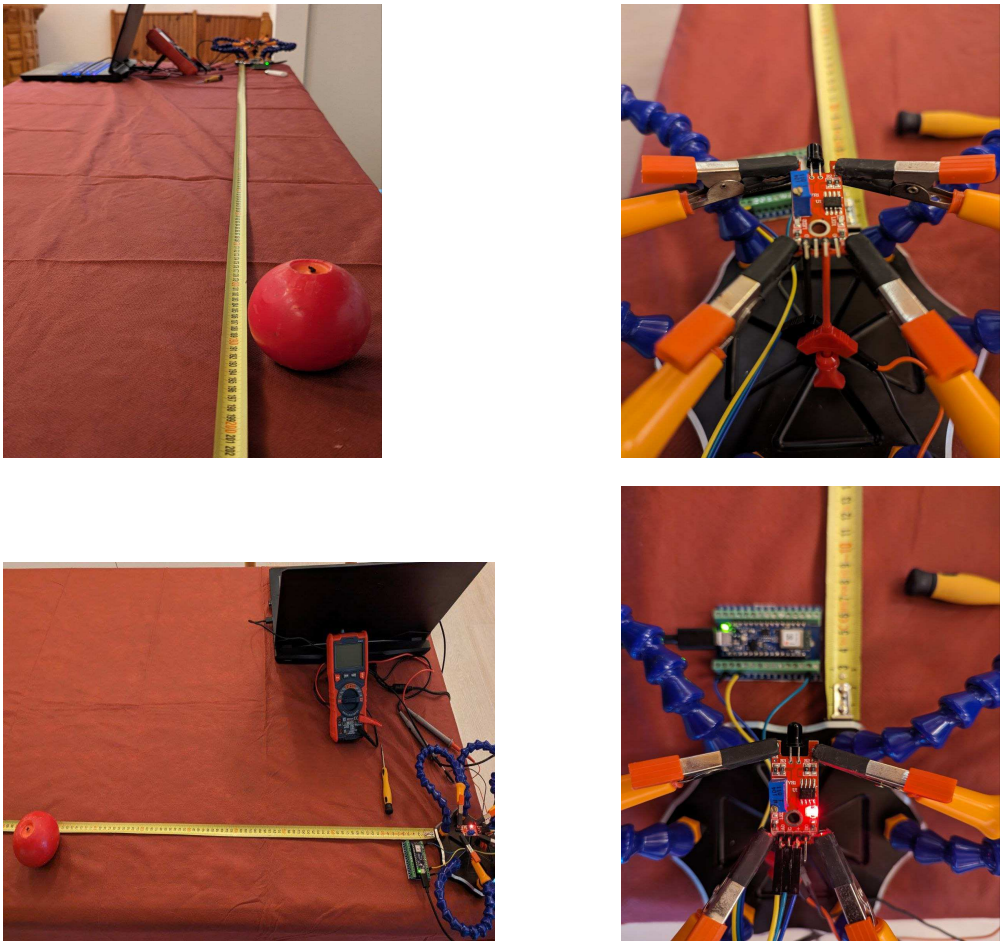


Figura 4.6: Foto del setup

4.4 Calibrazione dinamica

Avendo studiato il comportamento dei sensori è possibile ora procedere con una calibrazione "dinamica". Inizialmente è fondamentale fissare la fiamma, prodotta

da una candela, alla distanza desiderata di 15cm dalla testa del robot cercando di mantenerne l'intensità più costante possibile. In particolare, è molto importante centrarla perfettamente ponendo tutti i 3 servomotori alla posizione intermedia di 90 gradi.

Per la calibrazione, il robot viene avviato in fase di ricerca, con lo scopo di rilevare i valori convertiti medi nel proprio spazio di lavoro.

Vengono memorizzati i valori medi minimi di ognuno dei sensori, separatamente. Nel caso in cui venga rilevato un valore medio minore del precedente, il valore medio minimo precedente viene sostituito. I potenziometri dei 5 sensori sono impostati inizialmente a 5$k\Omega$ con valori che vengono progressivamente aggiustati in modo dinamico. La seguente figura 4.7 propone il grafico Serial Plotter dei valori convertiti di tutti i 5 sensori nelle condizioni iniziali, quindi con potenziometro fissato a 5$k\Omega$.

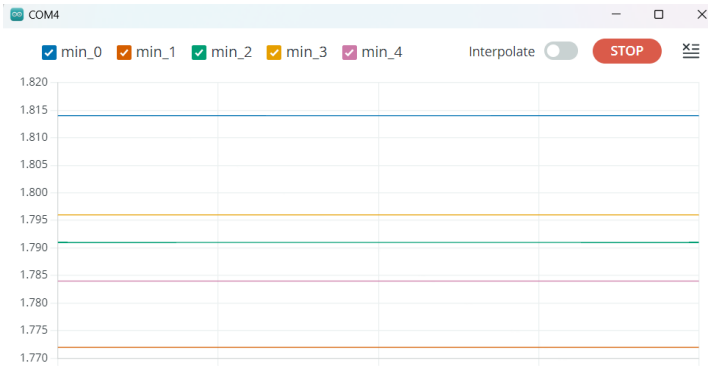


Figura 4.7: Condizioni iniziali della calibrazione dinamica

L'obiettivo, come stabilito in precedenza, è ora quello di portare tutti i valori convertiti medi minimi a circa 1500. Per fare ciò, andranno modificate le impostazioni dei potenziometri, abbassando le resistenze fino ad ottenere il risultato corretto. Con l'avvicinarsi a 1500 la calibrazione diventa sempre più delicata, in quanto piccole variazioni del potenziometro in quell'intorno portano ad ingenti variazioni nell'uscita del sensore. Viene riportato in figura 4.8 il risultato finale del processo di calibrazione.

I valori delle resistenze dei potenziometri dei sensori, indicate da destra verso sinistra, quindi A0, A1, A2, A3, A6 sono rispettivamente 2163Ω, 1007Ω, 935Ω, 1357Ω, 1840Ω. Si nota come in questo caso le resistenze siano minori rispetto al caso rilevato nelle misurazioni a singolo sensore. Ciò accade perché, mentre il robot è in movimento, è presente anche un carico sull'alimentatore dato dai servomotori. Ne deriva che la tensione reale che arriva ai sensori è minore, andando di conseguenza ad influenzare il valore di resistenza del potenziometro che dovrà essere minore a parità di valore in uscita dal sensore. Si osserva anche come le resistenze siano diverse tra un sensore e l'altro. Ciò deriva dalla diversa collocazione dei sensori sulla testa, in quanto i sensori più esterni sono rivolti

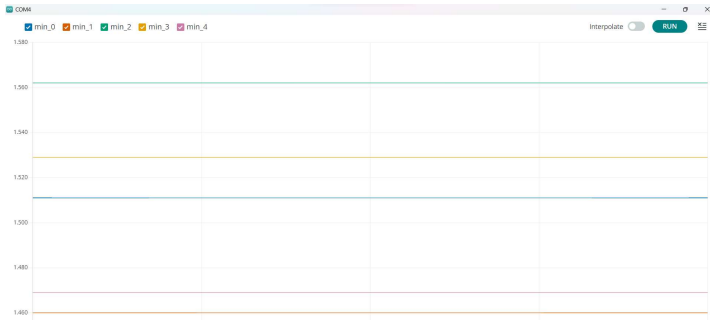


Figura 4.8: Valori finali della calibrazione dinamica

maggiormente verso avanti rispetto alla direzione radiale dal centro della testa. La procedura di calibrazione è quindi cruciale in quanto consente di regolare i sensori durante una simulazione del funzionamento effettivo del robot.

4.5 Test delle performance

4.5.1 Test dei tempi

Data la funzione antincendio del robot, il parametro sicuramente più rilevante è quello relativo al tempo di individuazione del principio d'incendio. Si è pertanto sviluppata una procedura facilmente replicabile per misurare il tempo impiegato dal robot per passare dalla fase di ricerca a quella di puntamento della fiamma.

La prima fase della procedura di test prevede l'accensione del robot in fase di ricerca. Esso si muove ricercando un principio d'incendio nel proprio spazio di lavoro. Si procede quindi all'accensione della fiamma della candela. Per ottenere delle misure di tempo più precise si utilizza una barriera ABS opaco che impedisca la visuale della fiamma al robot fino all'inizio della misurazione.

Rimossa la barriera, dopo un intervallo di tempo di ricerca, puntamento e distanziamento si accende un led giallo sull'Arduino. Il led è programmato per accendersi nel momento il cui il valore convertito dall'ADC in uscita dal sensore centrale si trova in un intorno di 1500. Pertanto, affinché il valore si collochi in questa fascia, è necessario eseguire questa procedura con la testa del robot posizionata ad una distanza pari a $15cm$ dalla candela, come si è osservato anche durante la procedura di calibrazione. Le misure possono essere effettuate anche ponendo la fiamma ad una distanza inferiore o leggermente superiore, in quanto il robot è stato programmato per posizionarsi alla giusta distanza anche ritirandosi, fino a quando possibile, evitando conflitti strutturali.

Vengono realizzate numerose misurazioni posizionando la candela nello spazio di lavoro del robot, simile ad una corona di semicirconferenza con raggio massimo vicino a $15cm$. Può quindi essere calcolato il tempo minimo, massimo e medio impiegato dal robot per individuare la fiamma, dopo aver raccolto i

dati temporali di molteplici test. Dato che il tempo necessario è dell'ordine di grandezza del secondo, per realizzare delle misurazioni più accurate possibili, è stato realizzato un video per ciascuna rilevazione in modo da contare i frame trascorsi da quando la barriera viene spostata al momento in cui si accende il led dell'Arduino. Sapendo che le riprese effettuate sono caratterizzate da 30 frame al secondo, l'intervallo minimo è pari a circa $0.03s$. In questo modo si ottiene una misurazione molto accurata, replicabile difficilmente con l'ausilio di un cronometro.

Il grafico in figura 4.9 mostra i risultati di 33 test, inseriti in ordine cronologico, con il numero del test sulle ascisse ed il tempo in secondi sulle ordinate. In particolare il grafico deve essere suddiviso in due parti, in quanto le prime 21 misurazioni sono state eseguite con intorno di ± 200, mentre negli ultimi 12 test l'intorno è pari a ± 100.

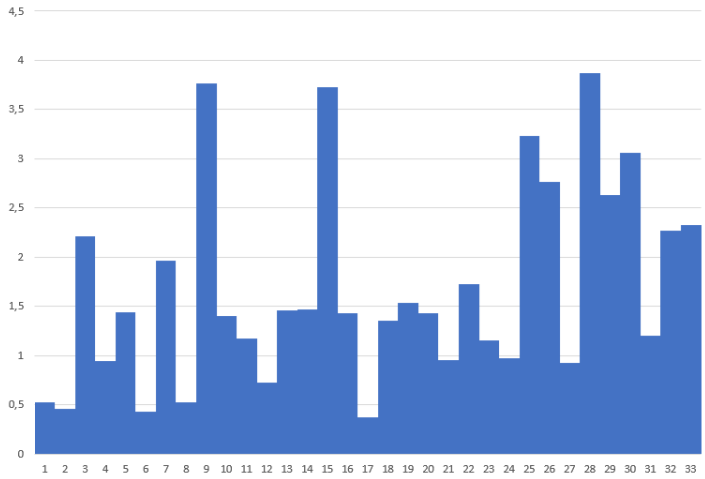


Figura 4.9: Grafico dei test dei tempi

La conseguenza di questo cambiamento risalta chiaramente nel tempo richiesto per il puntamento, che risulta essere maggiore nel secondo caso dal momento che il valore convertito richiede più tempo a stabilizzarsi in un intervallo più ristretto. In particolare per rendere più ripetibili le misure dei tempi, il led giallo è stato considerato acceso quando è rimasto stabile per almeno 0.5 secondi.

I tempi risultanti sono molto variabili tra loro. Ciò dipende dalla vicinanza alla fiamma del robot al momento del sollevamento della barriera. Un ulteriore causa di errore da non sottovalutare è senz'altro quello dell'oscillazione della fiamma, ad esempio a causa dello spostamento d'aria determinato dal sollevamento della lastra. Considerando le prime 21 misurazioni, il tempo medio di rilevazione, puntamento e distanziamento dalla fiamma risulta essere pari a 1.40 secondi, mentre il tempo massimo a 3.77 secondi. Considerando le rimanenti misurazioni, il tempo medio è pari a 2.18 secondi, ed il tempo massimo di 3.87 secondi. Questi dati, riportati nella tabella 4.4, dimostrano come il robot pre-

senti una buona reattività e sia in grado di individuare la fiamma in un tempo piuttosto breve.

Numero test	1	2	3	4	5	6	7	8	9
	10	11	12	13	14	15	16	17	18
	19	20	21	22	23	24	25	26	27
	28	29	30	31	32	33			
Tempo (s) ±0.03	0.53	0.46	2.21	0.95	1.44	0.43	1.97	0.53	3.77
	1.40	1.17	0.73	1.46	1.47	3.73	1.43	0.37	1.36
	1.54	1.43	0.96	1.73	1.16	0.97	3.23	2.77	0.93
	3.87	2.63	3.06	1.20	2.27	2.33			

Tabella 4.4: Risultati ottenuti dal test dei tempi

4.5.2 Test dell'affidabilità

Un'altra caratteristica fondamentale per un robot antincendio è certamente l'affidabilità, che può essere rilevata con una procedura identica alla precedente. In particolare su 34 test, 33 dei quali considerati nelle misurazioni dei tempi, in un solo test il robot antincendio non ha concluso l'operazione correttamente. Da ciò è possibile stimare come l'affidabilità sia pari al $97,06\%$. Questo parametro consente di considerare questo robot affidabile, requisito fondamentale per un robot con una funzione di rilevazione di principi d'incendio.

Conclusioni

Dai risultati di 34 test si evince la capacità del modello di puntare e distanziarsi dalle fiamme in modo rapido e preciso, in un tempo inferiore ai 4 secondi nel 97% dei casi. I dati ottenuti dai test sono coerenti con quanto atteso, per cui il progetto si ritiene aver avuto successo con risultati soddisfacenti.

Il modello presenta ancora ridotta mobilità tridimensionale e problemi di precisione in ambienti in cui filtri molta luce solare siano presenti ostacoli tra il sistema elettrico e i sensori. Di seguito si intende fornire un approfondimento su alcuni dei possibili sviluppi futuri del modello.

Agente estinguente L'acqua nebulizzata con additivi, rilasciata da sprinkler simili a quello in figura C .2, ha diversi vantaggi, tra i quali un costo e impatto ambientale ridotto comparato ad alternative come CO2, argonite, HFC-227ea o HFC-225. La doppia azione di raffreddamento dell'ambiente e soffocamento della fiamma la rendono un'ottima scelta come agente estinguente. Tuttavia, gli ugelli richiedono frequente manutenzione a causa della presenza di elementi in soluzione nell'acqua, come sali minerali e gli additivi stessi.

Sensore di sovratemperatura L'utilizzo di sensori di sovratemperatura ad infrarossi, scartato per ragioni di economicità, permetterebbe una previsione del possibile incendio causato da un surriscaldamento, e quindi una maggiore reattività del sistema.

Meccanismo reale Una possibile applicazione reale del modello dovrebbe mantenere o migliorare libertà di movimento, rapidità, ingombro, economicità e semplicità d'installazione. Il meccanismo di tipo CoreXY, solitamente utilizzato in CNC e stampanti 3D viene azionato da due soli motori. Come da figura C .1, utilizza l'intreccio di cinghie per gestire autonomamente i motori e muovere i sensori, disposti a doppio o triplo ventaglio in modo da coprire una ancora maggiore superficie, e un ugello con cui spruzzare l'acqua. Potrebbe essere montato frontalmente su una porta d'armadio elettrico, garantendo la funzionalità desiderata con ingombro ridotto.

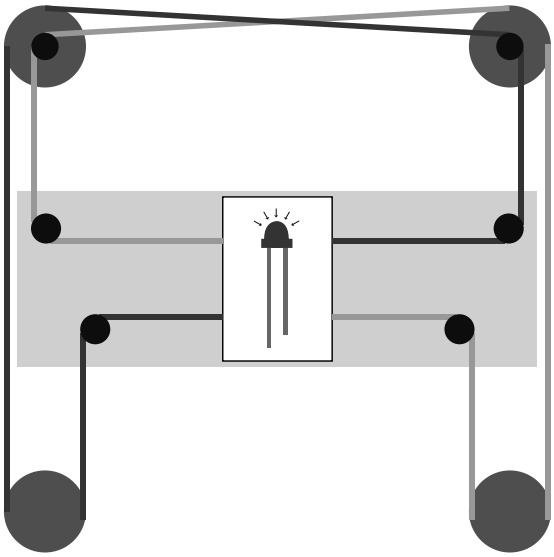


Figura C .1: Meccanismo CoreXY

Figura C .2: Sprinkler [11]

Bibliografia

[1] "Annuario statistico dei vigili del fuoco." `https://www.vigilfuoco.it/`.

[2] J. Mawhinney and B. Taber, "Findings of experiments using water mist for fire suppression in an electronic equipment room," in *Proceedings: Halon Alternatives Technical Working Conference, Albuquerque, New Mexico*, p. 15, 1996.

[3] "Flame sensor module ky026." `https://win.adrirobot.it/`.

[4] "Basic of photodiodes." `https://www.digikey.it/`.

[5] "Ky-026 flame sensor tutorial for arduino, esp8266 and esp32." `https://diyi0t.com/`.

[6] M. Höpfner, M. Milz, S. Buehler, J. Orphal, and G. Stiller, "The natural greenhouse effect of atmospheric oxygen (o2) and nitrogen (n2)," *Geophysical Research Letters*, vol. 39, no. 10.

[7] O. Keski-Rahkonen, J. Mangs, and A. Turtola, "Ignition of and fire spread on cables and electronic components," 1999.

[8] V. Babrauskas, *Electrical Fires*, pp. 662–704. New York, NY: Springer New York, 2016.

[9] "Servomotore." `https://win.adrirobot.it/`.

[10] "Arduino documentation." `https://docs.arduino.cc/hardware/nano-33-ble`.

[11] "Immagine esemplificativa sprinkler." `https://educationbusinessuk.net/`.

[12] A. S. Matteo Bertocco, *Misure elettroniche*. 2010.

[13] T. N. B. Richard C. Jaeger, *Microelettronica*. McGraw-Hill Education, 2013.

[14] A. C. Cesare Bonaccina and L. Mattarolo, *Trasmissione del calore*. Coop Libraria Editrice Università di Padova, terza ed., 2014.

[15] B.-H. C. JiaZheng Lu, Y. S. Ping Liang, and Z. Fang, "Experimental evaluation of protecting high-voltage electrical transformer using water mist with and without additives," *Fire Technology*, no. 55, pp. 1671–1690, 2019.

[16] G. Kirchhoff, "Über das verhältnis zwischen dem emissionsvermögen und dem absorptionsvermögen der körper für wärme und licht," *Annalen der Physik und Chemie*, 1960.

[17] "Fritzing." `https://fritzing.org/`.

[18] "Meccanismo corexy." `https://corexy.com`.

[19] "Microcontrollore." `https://it.infocenter.nordicsemi.com/`.

[20] T. Rakib and M. R. Sarkar, "Design and fabrication of an autonomous fire fighting robot with multisensor fire detection using pid controller," in *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, pp. 909–914, IEEE, 2016.

[21] A. Gaur, A. Singh, A. Kumar, K. S. Kulkarni, S. Lala, K. Kapoor, V. Srivastava, A. Kumar, and S. C. Mukhopadhyay, "Fire sensing technologies: A review," *IEEE Sensors Journal*, vol. 19, no. 9, pp. 3191–3202, 2019.

[22] F. W. Mowrer and M. G. Pecht, "Exploratory research on nonthermal damage to electronics from fires and fire-suppression agents," in *Annual Reliability and Maintainability Symposium 1995 Proceedings*, pp. 1–6, IEEE, 1995.

[23] V. Babrauskas, "Research on electrical fires: The state of the art," *Fire Saf. Sci*, vol. 9, pp. 3–18, 2008.

[24] "Inhibition of thermal runaway in lithium-ion batteries by fine water mist containing a low-conductivity compound additive," *Journal of Cleaner Production*, vol. 340, p. 130841, 2022.

[25] K. Notarianni and W. Rinkinen, "Protection of data processing equipment with fine water sprays:," 1 1994.

[26] J. Liu, G. Liao, P. Li, W. Fan, and Q. Lu, "Progress in research and application of water mist fire suppression technology," *Chinese Science Bulletin*, vol. 48, pp. 718–725, 2003.

Elenco delle tabelle

Elenco delle figure