

# UNIVERSITÀ DEGLI STUDI DI CATANIA

Dipartimento di Ingegneria Elettrica, Elettronica e Informatica

Corso di Laurea in Ingegneria Informatica


---


***Andrea Arciprete***


ELABORATO FINALE

# Progetto e realizzazione di un sistema di orientamento per robot bioispirati basato sulla fonotassi

TESI DI LAUREA


**RELATORE**

*Prof. Paolo Pietro Arena*

***CORRELATORE***

*Prof. Luca Patanè*


---

*ANNO ACCADEMICO 2018-2019*

# Abstract

Oggigiorno esistono diversi settori di utilizzo della cosiddetta localizzazione sonora. Questo concetto non viene utilizzato soltanto per localizzare una persona che sta parlando o uno speaker dal quale proviene un determinato suono, ma può anche essere utilizzato per rilevare una persona bisognosa di aiuto. Questo progetto non è da considerare a sé stante ma va inquadrato in un contesto più generale in cui esso può essere utilizzato e magari migliorato con l'aggiunta della parte motoria che consentirebbe al robot non solo di ruotare verso la direzione dalla quale proviene il suono ma anche di muoversi e dirigersi verso la persona che, ad esempio, sta richiedendo aiuto. Se si riuscisse a ridurre la dimensione del robot al minimo e, inoltre, si aggiungesse la parte motoria esso, ad esempio, potrebbe essere inviato in località che hanno subito delle calamità naturali o eventi catastrofici. Perché, in fondo, secondo la mia opinione: la robotica non va intesa come una "semplice" macchina bensì come uno strumento che non sostituisce l'uomo ma che semplicemente lo affianca e lo aiuta nelle situazioni più difficili. Una fonte di ispirazione è data dal mondo degli animali, ed in particolare dai grilli, parecchio studiati in tale ambito, in quanto utilizzano gli stimoli sonori sia a scopi riproduttivi, che per fuggire da eventuali predatori. È nota infatti la fonotassi del grillo, ovvero la capacità di quest'ultimo di dirigersi verso individui di sesso differente a scopo riproduttivo. Un metodo per localizzare la posizione di una sorgente sonora, analogamente al principio su cui si basa la fonotassi del grillo, è usare diversi microfoni per registrare la differenza tra gli istanti di tempo in cui ciascun microfono rileva lo stesso suono. Utilizzando questa tecnica che prende il nome di Time Difference of Arrival (TDOA) e la trigonometria è possibile calcolare la posizione della sorgente sonora. L'obiettivo di questa tesi è di indagare su come determinare la posizione di una sorgente sonora utilizzando non la TDOA ma una tecnica che si basa sul calcolo della differenza (errore) tra il valore dell'ampiezza del suono che arriva ad un microfono e lo stesso valore che viene registrato da un altro microfono. In pratica, in real time viene calcolata la differenza tra due valori di ampiezza: quello rilevato dal microfono destro e quello rilevato dal microfono sinistro e in base al risultato viene effettuato il calcolo della posizione. Al fine di testare il progetto è stata costruita una piattaforma su cui sono stati montati i 3 microfoni che sono stati utilizzati per acquisire il suono proveniente dalla sorgente sonora. Inoltre, sempre su di essa è stato montato un sensore ad ultrasuoni che dà la possibilità di misurare la distanza tra il robot e un oggetto qualunque e, quindi, di realizzare anche un Sonar che consente di visualizzare tutti gli oggetti presenti nel raggio d'azione del robot. Infine, è stato utilizzato anche un sensore a infrarossi avente lo scopo di acquisire il segnale proveniente da un telecomando.

Ho deciso di includere anche questa feature nel progetto in modo da poter selezionare la modalità di funzionamento del robot che può essere: sonar o localizzazione sonora. Se si trova in modalità Sonar non fa altro che rilevare tutti gli oggetti che si trovano nel raggio d'azione del robot senza acquisire nessuna sorgente sonora, se è impostato in modalità localizzazione sonora, esso non fa altro che acquisire il suono proveniente dalla sorgente sonora e ruotare verso la direzione da cui proviene il suono.

# Ringraziamenti

Desidererei ringraziare il Prof. Paolo Pietro Arena e il Prof. Luca Patanè per avermi dato la possibilità di cimentarmi con lo sviluppo di questo progetto. Inizialmente, sembrava tutto molto più grande di me, al di fuori della mia portata, delle mie conoscenze, ma con il tempo ho iniziato ad appassionarmi sempre di più al campo della robotica, soprattutto in virtù delle lezioni tenute dal Prof. Arena che mi hanno portato a scegliere e svolgere una tesi che mi ha permesso di migliorarmi tantissimo, di avere ancora più autostima in me stesso e di imparare tantissime cose sull'ambito della robotica, anche se ancora sono solo all'inizio. Io penso che lo sfogo migliore per un ingegnere o per un aspirante tale sia toccare con mano o comunque vedere che ciò che è stato progettato nella teoria poi può essere effettivamente realizzato e funzionante. Questa tesi mi ha dato la possibilità effettiva di realizzare un sistema che prima era stato immaginato soltanto teoricamente ma che ora è realtà, ecco perché desidero ringraziare i professori per avermi dato questa possibilità che mi ha fatto crescere molto.

L'altra persona che desidero ringraziare prima di ogni cosa si chiama Daniela Chiavetta, colei che mi ha sempre capito, che mi ha sempre assistito in tutto, che c'è sempre stata, anche per un semplice supporto morale, colei che mi ha allietato le giornate intense di studio che sembrava non finissero mai. Questi tre anni molto pesanti sono stati resi più leggeri grazie alla presenza di questa persona che denominare "speciale" è dir poco. Se ho raggiunto questo livello, se ho superato tutte queste materie è anche merito suo, anche perché affrontare tutte le materie, tutti gli esami, condividere l'ansia da esami è una cosa da non sottovalutare in quanto mi ha permesso di dare il meglio di me stesso; ecco perché desidererei ringraziarla con tutto il mio cuore. Una corsa è certa, "niente e nessuno" ci dividerà.

Le altre persone che desidererei ringraziare sono i miei amici, in particolare il mio saldatore di fiducia Cristian Barbagallo, i miei colleghi tra i quali è doveroso citare Giuseppe Caramagna e Bartolomeo Caruso, ma soprattutto la mia famiglia che c'è sempre stata nel momento del bisogno. Un ringraziamento particolare va ai miei fratelli e ai miei genitori che in questi tre anni hanno fatto tantissimi sacrifici per far sì che io li affrontassi nel migliore dei modi. Infine, un ringraziamento va a me stesso per il fatto di non essermi mai arreso nonostante le difficoltà, di non aver mai mollato e di averci sempre creduto anche quando mi sono state voltate le spalle. Nella vita bisogna crederci sempre, non bisogna arrendersi mai perché con costanza e dedizione è possibile raggiungere qualsiasi obiettivo, anche quello che sembra impossibile da raggiungere.

# Indice

# Capitolo 1

# Introduzione

Questo lavoro di tesi ha per argomento principale il progetto e la realizzazione di un dispositivo destinato a piattaforme robotiche ai fini dell'orientamento in un ambiente soggetto a stimoli sonori. A scopo introduttivo si vuol dare una prospettiva di tipo bioispirato, cioè facente riferimento a come il mondo animale, ed in particolare gli insetti utilizzano le fonti sonore per poter compiere azioni di sopravvivenza o di riproduzione. In particolare, la fonotassi, cioè la capacità di riconoscere e dirigersi verso particolari sorgenti sonore, viene spesso identificata come un riflesso primario, cioè in grado di salvare la vita o poterla trasmettere alle generazioni successive. Infatti, facendo particolare riferimento al grillo, questo utilizza la fonotassi sia per dirigersi verso individui dell'altro sesso a scopo riproduttivo (è ben noto il canto del grillo che altro non è che una sillaba complessa e quindi nettamente riconoscibile generata a scopo puramente attrattivo) o per salvare la vita. Infatti, sempre a proposito del grillo, gli ultrasuoni emessi dai pipistrelli (noti predatori dei grilli) sono avvertiti in modo istintivo come stimoli da cui rifuggire. Nell'ambito di una attività internazionale il gruppo di Biorobotica della nostra Università circa una decina di anni or sono si è occupato a fondo del problema, contribuendo al progetto ed alla realizzazione di una rete neurale ispirata alla struttura neurale del grillo, che fungesse da sistema di orientamento su piattaforme robotiche. Allo scopo era stato appositamente progettato e realizzato un complesso circuito a partire dalle caratteristiche fisiche del grillo che fu testato con successo su una piattaforma mobile [0]. Lo scopo di questa tesi è quello di pervenire al medesimo risultato di orientamento, ma utilizzando architetture moderne, a basso costo e che utilizzino algoritmi di filtraggio la cui complessità non costituisca più un collo di bottiglia ai fini dell'implementazione su microcontrollori di fascia economica. Naturalmente il problema è stato notevolmente semplificato per quanto riguarda il tipo di segnale da processare, ma questo non costituisce alcun problema i fini della implementazione su robot. I risultati ottenuti mostrano come l'approccio seguito possa essere facilmente implementato su piattaforme mobili.

## 1.1 Motivazione

Nei campi della robotica e dell'intelligenza artificiale, la modalità sensoriale più popolare ad essere incorporata nei sistemi è la visione; fino a poco tempo fa l'udito (ascolto) non ha giocato un ruolo molto importante nella ricerca di sistemi intelligenti. Sono stati fatti pochi tentativi di incorporazione del sound processing in un robot autonomo. Tuttavia, il suono fornisce una ricca fonte di informazione: molti animali si affidano alla localizzazione e ad altri aspetti percettivi uditivi per sopravvivere; la parola e l'udito sono i principali mezzi di comunicazione per gli esseri umani, la domanda sorge spontanea: perché non provare a riprodurre questi mezzi anche artificialmente ed incorporarli all'interno di un robot?

In un certo senso la capacità di ascoltare delle sorgenti sonore da parte di un robot è più difficile di quanto non lo sia la visione. La grande differenza consiste nel fatto che, a differenza degli occhi, le orecchie non ricevono direttamente informazioni spaziali dall'ambiente circostante. Il sistema uditivo si basa soprattutto sull'elaborazione di dati sensoriali grezzi per estrarre segnali acustici e derivare indirettamente informazioni spaziali.

Nonostante queste complessità, è importante trarre vantaggio da ambo le proprietà uditive e visive per estrarre funzionalità e informazioni dall'ambiente circostante, cosa che sarebbe più complessa qualora si considerasse soltanto l'una o l'altra proprietà singolarmente [1].

A tal fine, è fondamentale avere un sistema integrato in grado di accoppiare le due differenti modalità sensoriali, come audizione e visione. Il lavoro presentato in questa tesi mira a realizzare un sistema che possa acquisire una sorgente sonora, elaborarla così da determinare la posizione della sorgente sonora stessa. In un certo senso, questa tesi focalizza l'attenzione su uno dei due aspetti sensoriali ossia l'audizione.

## 1.2 Obiettivo e Contenuti

Lo scopo della tesi è attuare la localizzazione sonora studiando la possibilità di implementarla all'interno di un robot in modo da rilevare la posizione della sorgente sonora stessa. L'obiettivo principale del progetto è rispondere alle seguenti domande:

- Quanto accuratamente può essere fatta la localizzazione di una sorgente sonora, calcolando l'angolo e la distanza da essa, utilizzando dei componenti a basso costo come ad esempio dei microfoni e Arduino Mega?
- Con quanta accuratezza possono essere effettuate delle misurazioni da parte di un sensore ad ultrasuoni, che consente la realizzazione di un Sonar?

Questo progetto è focalizzato sulla localizzazione di una sorgente sonora, esso non prevede il rilevamento della voce umana o di una canzone, o quantomeno lo prevede ma non viene garantita alcuna certezza circa l'accuratezza della misurazione dell'angolo e della distanza, anche perché gli unici suoni che vengono seguiti sono quelli aventi frequenza fondamentale pari a 1kHz.

In questa tesi, viene preso in considerazione l'utilizzo di una sorgente sonora in grado di generare solo ed esclusivamente un suono impulsivo o un suono che rimane stazionario nel tempo. Le variazioni di velocità del suono dovute a variazioni di pressione e temperatura sono trascurate.

Per quanto concerne i contenuti:

**Capitolo 2:** Esso si prefigge di descrivere teoricamente i componenti che verranno adoperati all'interno del progetto. Ovviamente non si può non iniziare facendo un breve accenno al suono, visto che l'intero progetto si basa sull'acquisizione di una sorgente sonora. Sempre all'interno di questo capitolo verrà data una breve descrizione di cosa sia un microfono, delle varie tecniche di localizzazione esistenti e del PWM, che viene utilizzato per comandare il servomotore.

**Capitolo 3:** Pur non avendo implementato la tecnica di localizzazione, denominata TDOA, ho deciso di dedicarle un capitolo a parte in quanto è una delle tecniche più utilizzate in assoluto; inoltre, avendo a disposizione una gamma di microfoni molto performante sarebbe possibile implementarla in modo da localizzare correttamente una sorgente sonora. Questo progetto ha lo scopo di perseguire un obiettivo: realizzare un sistema in grado di seguire una sorgente sonora utilizzando componenti a basso costo, dunque non è stato conveniente implementare questa tecnica di localizzazione che si basa sulla ITD, poiché, altrimenti vi sarebbe stata una lievitazione dei costi, con conseguente allontanamento dall'obiettivo che questo progetto vuole perseguire.

**Capitolo 4:** Esso prende il nome di *system design e implementazione* in quanto è incentrato su due aspetti: hardware e software. In questo capitolo verranno descritti tutti i sensori che sono stati utilizzati per lo sviluppo del progetto: sensore a ultrasuoni, microfoni, sensore IR e, inoltre, verrà data anche una breve descrizione del servomotore, elemento imprescindibile per il progetto in quanto consente al robot di effettuare la rotazione. Riferendomi sempre a questo capitolo, come detto sopra, ho preso anche in considerazione l'aspetto software di cui ho messo in evidenza il concetto di frequenza di campionamento che, purtroppo, rappresenta una grande limitazione per l'Arduino. Inoltre, ho messo in risalto anche l'aspetto della rilevazione del suono e della sua elaborazione mediante il calcolo della FFT e l'aspetto del calcolo dell'angolo di rotazione del servomotore. Infine, ho descritto come è stato realizzato il Sonar

mediante l'utilizzo del software Processing 3 e come, lato software, sono stati intercettati e decodificati i segnali provenienti dal telecomando.

**Capitolo 5:** Questo capitolo rappresenta la parte conclusiva della tesi in cui ho messo in risalto i risultati a cui sono giunto, una discussione finale, le conclusioni ed eventuali progetti futuri.

## 1.3 Metodo

L'idea che sta alla base del progetto è di realizzare una piattaforma su cui verranno montati i vari sensori. Questa piattaforma, a sua volta, verrà montata proprio sul servomotore che consentirà di effettuare la rotazione. In pratica ciò che alla fine dovrà verificarsi è che in seguito alla rilevazione della sorgente sonora da parte dei microfoni l'intera piattaforma che giace sopra il servomotore dovrà effettuare una rotazione in modo da puntare verso la direzione dalla quale proviene il suono. Per attuare l'esperimento ho deciso di utilizzare 3 microfoni. Il primo è stato montato sul lato sinistro della piattaforma, il secondo sul lato destro e il terzo nella parte centrale. I due microfoni laterali si trovano ad una distanza di circa 50 cm; ho scelto questa distanza in quanto facendo diversi test ho notato che è la distanza ottimale per ottenere una misura abbastanza precisa. Sempre sulla parte centrale è stato montato il sensore ad ultrasuoni che mi ha consentito di realizzare il Sonar e il sensore ad infrarossi che mi ha consentito di intercettare i segnali emessi da un telecomando. Per quanto concerne l'aspetto software, come detto nell'Abstract, ho preferito non implementare la TDOA per attuare la localizzazione sonora. Quest'ultima è stata implementata utilizzando un'altra tecnica che consiste nel calcolare l'errore, ossia la differenza tra le ampiezze rilevate dal microfono destro e sinistro e in funzione di questa effettuare una rotazione che verrà arrestata non appena il microfono centrale avrà rilevato un valore di ampiezza maggiore di quello rilevato dai microfoni: destro e sinistro. Questa tecnica verrà descritta in maniera più esaustiva nel Capitolo 4, con l'ausilio del codice.

## 1.4 Progetti Correlati

Sono stati necessari anni di ricerca nel campo della localizzazione sonora a causa della sua complessità in presenza di ambienti con riverbero o rumorosi [2]. Nella maggior parte dei progetti già sviluppati vengono utilizzati quattro microfoni per localizzare una sorgente sonora. Ciò nonostante è possibile usarne soltanto tre per ottenere sia la direzione sia la distanza dal suono utilizzando la TDOA e la trigonometria [3]. Esistono anche progetti in cui con soli due microfoni si riesce ad ottenere una localizzazione della sorgente sonora con un grado di errore minimo. Inoltre, sono stati realizzati anche dei veicoli in grado non solo di ruotare verso la direzione da cui proviene il suono ma anche di dirigersi fisicamente verso la sorgente sonora [4]. In fase di progetto è stato deciso di utilizzare tre microfoni in modo da trovare il miglior compromesso tra accuratezza delle misurazioni e costo. Il punto focale su cui ruota tutta la tesi è sviluppare un sistema in grado di seguire una sorgente sonora utilizzando componenti a basso costo; è questo l'obiettivo che mi sono prefissato sin dall'inizio.

# Capitolo 2

# Teoria

Questo capitolo descriverà quali teorie vengono applicate per dare una comprensione generale della parte dimostrativa del progetto.

## 2.1 Suono

Il suono è la sensazione data dalla vibrazione di un corpo in oscillazione. Tale vibrazione, che si propaga nell'aria o in un altro mezzo elastico, raggiunge l'apparato uditivo dell'orecchio che, tramite un complesso meccanismo interno, crea una sensazione "uditiva" correlata alla natura della vibrazione; in particolar modo la membrana timpanica subendo variazioni di pressione entra in vibrazione. Le oscillazioni sono spostamenti delle particelle intorno alla posizione di riposo e lungo la direzione di propagazione dell'onda; gli spostamenti sono provocati da movimenti vibratori, provenienti da un determinato oggetto, denominato sorgente del suono, il quale trasmette il proprio movimento alle particelle adiacenti grazie alle proprietà meccaniche del mezzo; le particelle, a loro volta, iniziando ad oscillare, trasmettono il movimento alle altre particelle vicine e queste a loro volta ad altre ancora, provocando una variazione locale della pressione; in questo modo, un semplice movimento vibratorio si propaga meccanicamente originando un'onda sonora (o onda acustica), che è pertanto un’onda longitudinale. Si ha un'onda longitudinale quando le particelle del mezzo in cui si propaga l'onda oscillano lungo la direzione di propagazione. Le onde meccaniche longitudinali sono anche denominate “onde di pressione”. Un suono può essere visualizzato come un’onda sinusoidale con una certa frequenza, che indica il numero di oscillazioni al secondo [5].
In figura 2.1 possono essere osservate due differenti onde sonore sinusoidali.

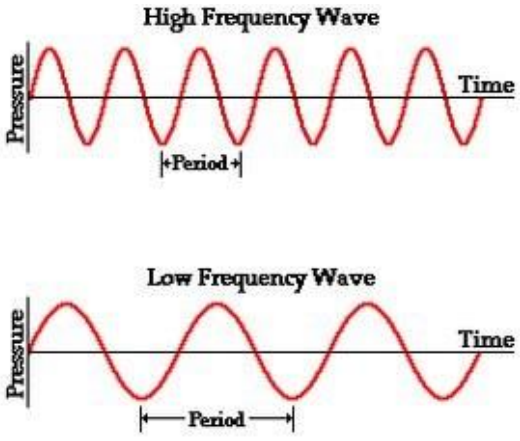


Figura 2.1: Due onde sonore sinusoidali differenti [6].

Il primo diagramma mostra un'onda sonora avente una frequenza più alta di quella dell'onda sonora mostrata nel secondo diagramma poiché il periodo è più breve. È importante notare che le due onde sonore hanno la stessa ampiezza. Per un essere umano l'intervallo di frequenze udibile è compreso tra i 20 Hz e i 20000 Hz.

## 2.2 Microfono

In linee generali, un microfono è costituito da una sottile membrana chiamata diaframma che vibra in presenza di un suono. L'ampiezza della vibrazione viene successivamente convertita in un segnale elettrico tramite una bobina e un magnete. Il segnale elettrico può, quindi, essere interpretato dal microcontrollore per rilevare il livello di ampiezza del suono. Quando è presente un suono più forte, la membrana vibra con un'ampiezza maggiore dando luogo ad un segnale elettrico la cui ampiezza è più elevata [7].
La figura 2.2 mostra la sezione trasversale di un microfono.

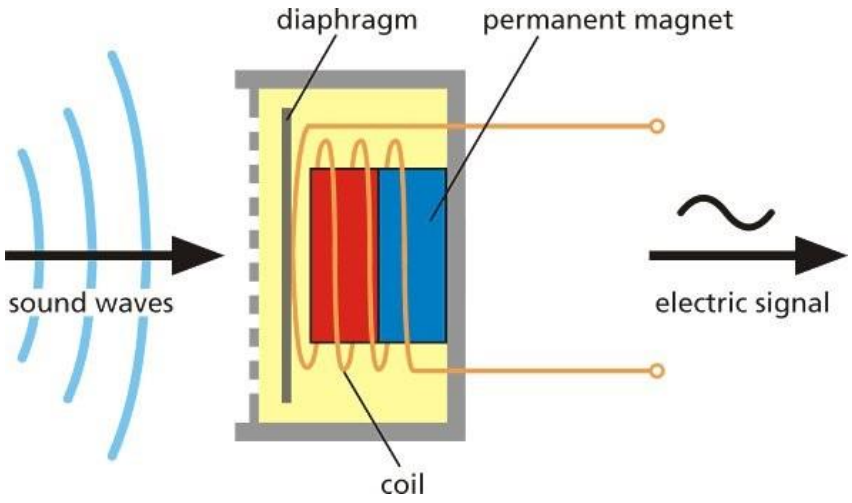


Figura 2.2: Sezione trasversale di un microfono [8].

## 2.3 PWM: Pulse Width Modulation

Un segnale digitale è un segnale discreto che può valere zero o uno, dove zero corrisponde al segnale basso e uno al segnale alto. Un segnale analogico è un segnale continuo che può assumere tutti i valori dell'insieme dei numeri reali.

In altre parole, può assumere infiniti valori. Al fine di avere un solo canale di uscita sia per un segnale analogico sia per un segnale digitale può essere utilizzato il Pulse Width Modulation (PWM) [9].

In elettronica e telecomunicazioni la modulazione a larghezza di impulso (o PWM, acronimo del corrispettivo inglese pulse-width modulation) è un tipo di modulazione digitale che permette di ottenere una tensione media variabile dipendente dal rapporto tra la durata dell'impulso positivo e di quello negativo (duty cycle). Allo stesso modo, è fortemente utilizzato per protocolli di comunicazione in cui l'informazione è codificata sotto forma di durata nel tempo di ciascun impulso. Grazie ai moderni microcontrollori, è possibile attivare o disattivare un interruttore ad alta frequenza e allo stesso modo rilevare lo stato e il periodo di un impulso. La durata di ciascun impulso può essere espressa in rapporto al periodo tra due impulsi successivi, implicando il concetto di ciclo di lavoro. Un ciclo di lavoro utile pari a 0% indica un impulso di durata nulla, in pratica assenza di segnale, mentre un valore del 100% indica che l'impulso termina nel momento in cui ha inizio il successivo. Un segnale di clock è a volte utilizzato per determinare la posizione degli impulsi, ma spesso non è necessario in quanto al segnale viene aggiunto un valore minimo che garantisce la presenza di un piccolo impulso anche per il valore zero.

La figura 2.3 mostra cicli di lavoro differenti per un Arduino che ha una tensione di uscita di 5V.

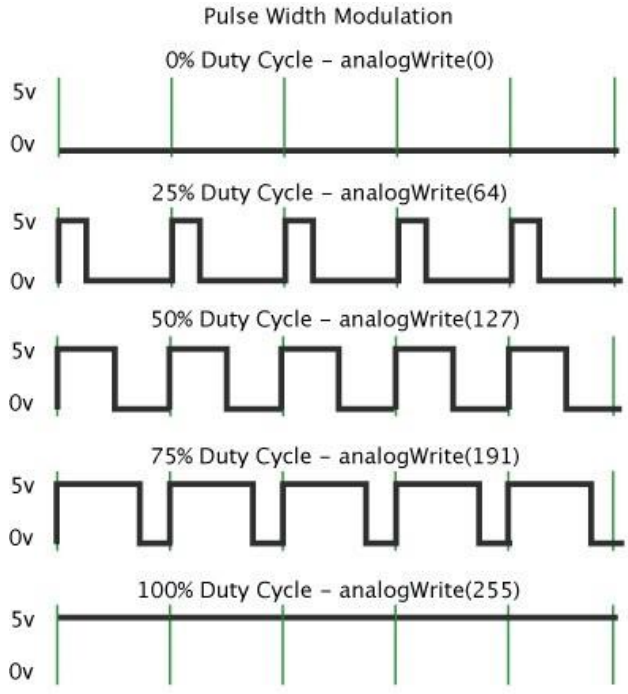


Figura 2.3: Segnali PWM differenti che variano da 0 a 255 [10].

# 2.4 Tecniche di localizzazione

La localizzazione sonora per gli esseri umani è principalmente un fenomeno binaurale e gli spunti sono perlopiù basati sulle differenze tra gli input che sopraggiungono alle due orecchie. Diciamo che le due tecniche che hanno giocato un ruolo dominante nell'ambito della localizzazione sonora sono state la Interaural Time Difference (ITD) e la Interaural Intensity Difference (IID). Una localizzazione sonora robusta si ottiene quando un sistema è in grado di combinare le informazioni raccolte da diversi segnali per localizzare suoni diversi.

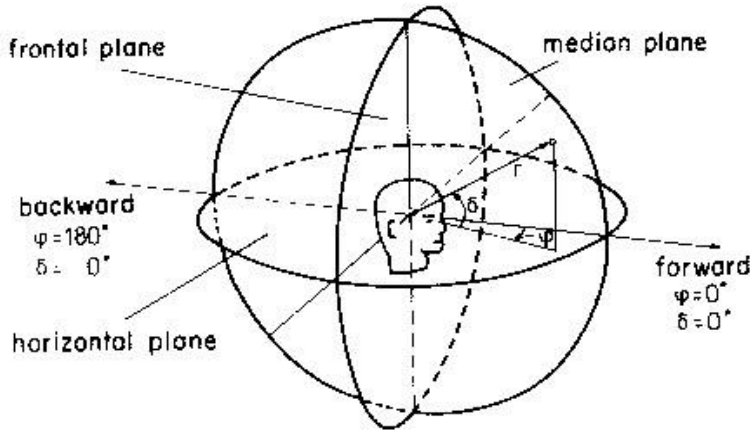


Figura 2.4: Schema usato per descrivere la localizzazione.

La ITD si basa sul fatto che le due orecchie sono situate ad una distanza l'una dall'altra finita. Il suono che giunge all'orecchio più vicino impiega del tempo prima di giungere all'orecchio più distante. I ritardi si aggirano intorno a 0 s per una sorgente situata direttamente davanti alla testa di un essere umano e arrivano fino a circa 700 μs per sorgenti che formano un angolo di ±90° rispetto alla testa dell'essere umano, proprio come si vede in figura 2.4. Per segnali a bassa frequenza, al di sotto di circa 1.5 KHz, l'ITD può essere misurato come ritardo di fase tra la forma d'onda del canale destro e quello della forma d'onda del canale sinistro. A frequenze più elevate, la risoluzione delle orecchie non è sufficiente per distinguere la differenza di fase. In tal caso l'ITD viene ricavata semplicemente calcolando la differenza tra gli istanti di tempo di arrivo del segnale alle due orecchie [11]. Occorre notare che esistono anche dei sistemi adattivi in cui non è nemmeno necessario specificare parametri caratteristici della testa come diametro, posizione del microfono, ecc.

Suoni ad alta frequenza, ossia quelli con lunghezza d'onda paragonabili alla larghezza della testa, sono ombreggiati dalla testa, dunque l'intensità del suono che entra nell'orecchio lontano è diminuita rispetto al suono che entra nell'orecchio vicino.

La Interaural Intensity Difference non può essere modellata facilmente così come la ITD. L'effetto ombra della testa può causare una IID fino ad un massimo di 20 dB e l'effetto dipende molto dalla frequenza.

Spesso i movimenti della testa rimuovono le ambiguità della localizzazione che possono verificarsi in seguito all'uso dell'ITD o dell'IID. È stato dimostrato che avendo a disposizione un sistema uditivo completamente sviluppato è possibile localizzare una sorgente sonora senza fare ricorso al movimento della testa [12]. In aggiunta, sono stati fatti anche altri esperimenti che hanno dimostrato che gli esseri umani adulti effettuano una localizzazione sonora con una maggiore risoluzione proprio nell'area che si trova di fronte la testa; di conseguenza ha molto senso orientare quest'ultima verso la sorgente sonora per una migliore localizzazione [13]. Altra tecnica molto importante che potrebbe essere utile per attuare la localizzazione di tipo non acustica è la cosiddetta "Vision" che però potrebbe entrare in conflitto con la localizzazione acustica infatti, in alcuni casi, i segnali uditivi vengono ignorati se sono in conflitto con i segnali visivi; ad esempio, quando si guarda la televisione o un film percepiamo che il discorso proviene dalla bocca delle persone invece che dagli altoparlanti, questo perché i due segnali sono entrati in conflitto e il segnale visivo ha prevalso su quello uditivo. È a causa di questo fenomeno che la Visione è stata scelta come riferimento per l'addestramento della rete neurale.

## 2.5 Localizzazione in ambienti realistici

È importante considerare il problema della localizzazione in un contesto realistico. Molti ricercatori, quando studiano la corretta localizzazione, lavorano in camere anecoiche per semplificare l'elaborazione o l'esperimento che viene eseguito. Quanto sottolineato, ovviamente, rende la localizzazione sonora molto più semplice, ma irrealistica in quanto noi normalmente viviamo e interagiamo in spazi chiusi che generano echi e riverberi causati dal fatto che le pareti e gli oggetti causano riflessione. Un suono di prova molto utilizzato nella ricerca psicoacustica è il tono continuo o sinusoidale, che è uno dei più difficili da localizzare in ambienti riverberanti. Sarebbe auspicabile avere un sistema in grado di attuare la localizzazione sonora sia in ambienti in cui non vi è riverbero sia in ambienti riverberanti, semplicemente utilizzando tecniche differenti che siano ottimali per ciascuna situazione. Un ambiente di ascolto realistico include stimoli naturalistici, presenza di suoni riflessi e presenza di riverbero (Figura 2.5). Si dice che l'ascoltatore si trova in un ambiente sonoro diretto se la sorgente sonora è sufficientemente vicina all'ascoltatore così che il primo suono che entra nelle orecchie è predominante rispetto agli echi successivi dovuti al fatto che il suono

originale è stato riflesso. In un ambiente riverberante, l'ascoltatore è abbastanza lontano dalla sorgente sonora, dunque il suono udito dall'ascoltatore è dovuto principalmente a riflessioni ripetute; di conseguenza la localizzazione diventa difficile in quanto il suono originale che dovrebbe arrivare all'ascoltatore viene presto corrotto da suoni riflessi che arrivano da tutte le direzioni. In ambienti riverberanti l'indicazione principale che deve essere utilizzata è la differenza tra gli istanti di insorgenza dell'inviluppo dei diversi segnali [14]. A causa del fatto che questa indicazione tende a scomparire subito dopo l'inizio del segnale, i toni continui non possono essere localizzati facilmente in ambienti riverberanti, mentre i clic e altri transitori, poiché sono caratterizzati da differenze di tempo di insorgenza dell'inviluppo nette, possono essere localizzati molto più facilmente. Questo è un aspetto positivo in quanto i suoni più comuni, quelli di tutti i giorni, raramente sono toni puri continui, quanto piuttosto sono segnali transitori e complessi.

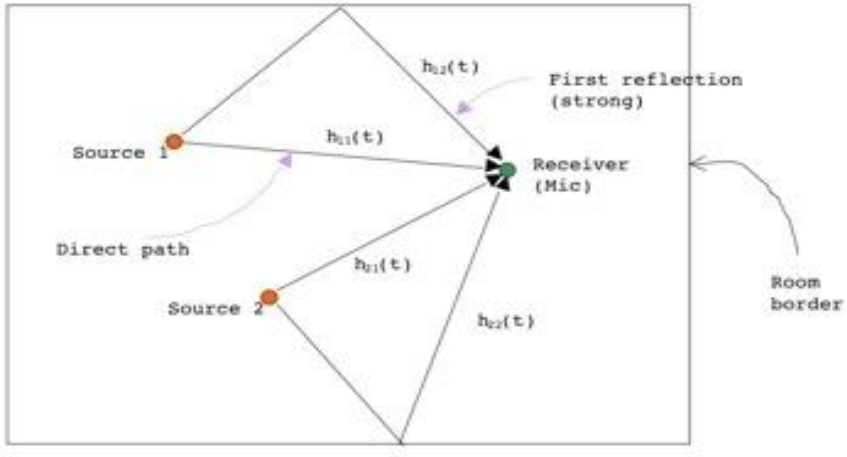


Figura 2.5: Modellazione delle riflessioni presenti in un ambiente chiuso.

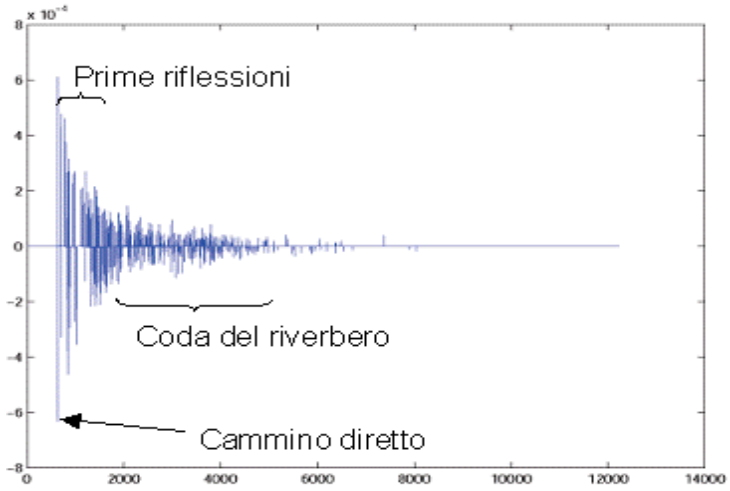


Figura 2.6: Tipica risposta impulsiva di un ambiente chiuso.

# Capitolo 3

# Stima dei tempi di ritardo

Questo capitolo descriverà qual è la tecnica di localizzazione sonora più utilizzata, sia facendo uso di due microfoni sia facendo uso di tre microfoni.

## 3.1 La correlazione

Dati due segnali x(t) e y(t) la loro correlazione è definita, formalmente, come il prodotto scalare tra x(t) e y(t+τ), al variare del ritardo τ, tra i due segnali. Quindi la correlazione esprime il prodotto scalare tra x(t) e tutte le possibili repliche traslate di y(t) e rappresenta il grado di "similitudine" tra queste grandezze [15].
In particolare, per segnali di energia, la cross correlazione è definita come:

$$R_{xy}(\tau) = \int_{-\infty}^{+\infty} x(t)y^*(t + \tau)$$

Per il Teorema di Parseval, invece, si ha che "il prodotto scalare dei segnali (x(t), y(t)) è uguale al prodotto scalare dei relativi spettri (X(f), Y(f))".

$$\text{Posto} \quad \begin{cases} y(t) = s(t) \\ x(t) = s(t+\tau) \end{cases} \qquad si\ ottiene$$

$$\int_{-\infty}^{+\infty} s(t)s^*(t + \tau)\mathrm{dt} = \int_{-\infty}^{+\infty} |S(f)|^2\, e^{j2\pi f\tau}\, df = R_{ss}(\tau)$$

In conclusione "la autocorrelazione di un segnale di energia è pari alla trasformata inversa di Fourier del quadrato della trasformata di Fourier del segnale stesso". Il valore assunto dalla autocorrelazione nell'origine equivale all'energia del segnale stesso:

$$R_{xx}(0) = \int_{-\infty}^{+\infty} x(t)x^*(t)\mathrm{dt} = E_x$$

# 3.2 La Cross Correlazione Generalizzata (GCC)

Per segnali tempo discreto x(n) e y(n), la cross correlazione è definita da:

$$R_{xy}(\tau) = \mathrm{E}[\mathrm{x(n)\ y(n-\tau)}]$$

in cui E[f(n)] è il valor medio di f(n) e per una finestra di osservazione di N campioni si può averne una stima tramite:

$$E[f(n)] = \frac{1}{N}\sum_{i=1}^{N} f(i)$$

Per il calcolo del tempo di ritardo è di grande importanza l'utilizzo di una variante della cross correlazione: la cross correlazione generalizzata (GCC) [1]:

$$R_{xy}^{(g)}(\tau) = E[(h_x(n) * x(n))\ (h_y(n-\tau) * y(n-\tau))]$$

La GCC si presta bene alle operazioni di pre-filtraggio che si rendono necessarie nella ottimizzazione della stima dei tempi di ritardo (TDOA). Per una facilità di calcolo può essere ricavata nel dominio della frequenza lo spettro di densità di potenza generalizzato $\Phi_{xy}(\omega)$ :

$$\Phi_{xy}(\omega) = [\,H_x(\omega)\,\mathrm{X}(\omega)]\,[H_y(\omega)\,\mathrm{Y}(\omega)]^*$$

Le funzioni pre-filtro $H_x(\omega)$ e $H_y(\omega)$, possono essere racchiuse in funzioni uniche:

$$\Psi_{xy}(\omega) = H_x(\omega)H_y^*(\omega).$$

Infine, nel caso in cui il dominio è discreto, per una finestra di N campioni, si definisce una frequenza discreta $\omega_k = \frac{2\pi k}{N}$ e la GCC è calcolata tramite l'antitrasformata di Fourier discreta (IDFT) di $\Phi_{xy}(\omega)$:

$$R_{xy}(\tau) = \frac{1}{N}\sum_{i=0}^{N-1} \Phi_{xy}(k)\, e^{j\frac{2\pi k i \tau}{N}}$$

Il tipo di funzione pre-filtro applicabile dipende dal contesto applicativo in cui si sta lavorando: prendiamo in considerazione il caso in cui si vogliono enfatizzare i picchi della funzione, operazione che risulta utile in presenza di riverbero.
Prendendo in considerazione il segnale arrivato a due microfoni distanziati tra loro, nel dominio del tempo, sarà:

$$\begin{cases} x(k) = s(k) + n_1(k) \\ y(k) = s(k + D) + n_2(k) \end{cases}$$

Dove s(k) rappresenta il segnale della sorgente, D è il numero di campioni di ritardo tra l'arrivo dei suoni ai due microfoni e n(k) è il rumore insito all'interno dei microfoni.

Nel dominio della frequenza la stessa relazione si scrive:

$$\begin{cases} X(\omega) = S(\omega) + N_1(\omega) \\ Y(\omega) = S(\omega)e^{j\omega D} + N_2(\omega) \end{cases}$$

Lo spettro di densità di cross potenza è composto da un termine che rappresenta la densità spettrale di potenza del segnale s(t) in cui è esplicitato il termine di sfasamento proporzionale a D e un altro termine in cui è incluso il rumore additivo bianco e incorrelato che può essere tralasciato:

$$\Phi_{xy}(\omega) = X(\omega)\, Y^*(\omega) = \Phi_{ss}(\omega)e^{-j\omega D} + \text{Noise}(\omega)$$

Antitrasformando questa grandezza si otterrà la cross correlazione centrata nel ritardo D:

$$R_{xy}(\tau) = F^{-1}\left[\Phi_{ss}(\omega)e^{-j\omega D}\right] = R_{ss}(\tau) * \delta(\tau - D) = R_{ss}(\tau - D)$$

Il nostro scopo è quello di ricavare il ritardo D avendo a disposizione la Rss ($\tau - D$). In realtà a causa del riverbero la cross correlazione potrebbe non avere un picco unico ma uno spettro allargato che rende difficile l'attribuzione del giusto ritardo al cammino diretto sorgente-microfono.
Il tempo di ritardo è dato semplicemente da:

$$\widehat{D} = \arg\max\{\hat{R}_{xy}(\tau)\}$$

In pratica, una volta calcolata la funzione di autocorrelazione tra i segnali che vengono captati dai due microfoni, si prende l'argomento, ossia il $\tau$ della funzione di autocorrelazione nel punto di massimo di quest'ultima e questo rappresenta proprio il tempo di ritardo cercato che verrà utilizzato per calcolare la posizione della sorgente sonora nel paragrafo successivo.

# 3.3 La stima della posizione della sorgente

Punto fondamentale della localizzazione di una sorgente sonora è la stima dei tempi di ritardo. Il fronte d'onda associato all'onda sonora giunge ai vari microfoni in tempi differenti; la valutazione dei tempi che intercorrono tra i vari microfoni (tempi di ritardo) permette, tramite tecniche geometriche di triangolazione, la stima della posizione della sorgente. In maniera semplificata, ipotizziamo un ambiente sonoro privo di rumore e dotato di ricevitori esenti da non idealità. Conoscendo a priori la distanza tra due microfoni è possibile, mediante semplici tecniche, stimare la direzione di provenienza dell'onda stessa (Figura 3.1).

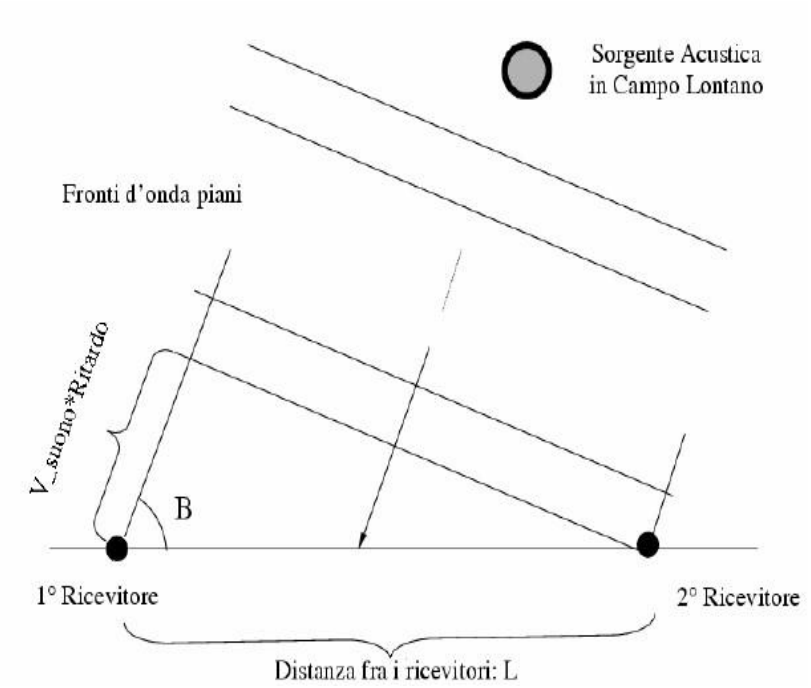


Figura 3.1: Geometria esemplificativa per il calcolo della direzione di provenienza del suono.

Infatti, essendo L la distanza tra i ricevitori(microfoni) e c la velocità del suono (345 m/s), è possibile calcolare l'angolo B di provenienza dell'onda mediante la formula:

$$cos(B) = \frac{c \cdot Ritardo}{L}$$

dunque:

$$B = \cos^{-1}\left(\frac{c \cdot Ritardo}{L}\right)$$

Mediante tre ricevitori è, quindi, possibile ottenere due tempi di ritardo e due direzioni di provenienza, che, intersecate tra loro, forniranno la posizione della sorgente (Figura 3.2).

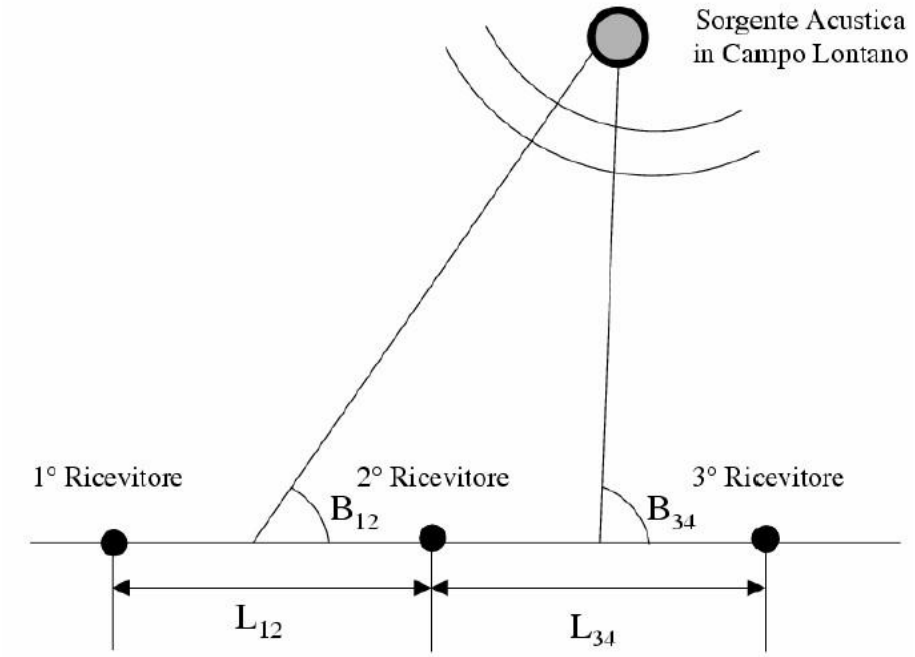


Figura 3.2: Stima della posizione mediante 3 microfoni.

Una situazione del genere manca tuttavia di realismo: è difatti poco probabile che le non idealità dovute ai microfoni o ad altri fenomeni reali (rumore, approssimazione di campo lontano non vera, ecc.) portino alla individuazione di un punto esatto. Nonostante ciò, mediante l'utilizzo di più microfoni (sicuramente più performanti), esistono diversi metodi che sopperiscono alle problematiche introdotte. Come detto sopra, per evitare di aumentare il numero di componenti, cosa che avrebbe fatto lievitare i costi, ho preferito non implementare questa tecnica di localizzazione sonora.

# Capitolo 4

# System Design e Implementazione

Questo capitolo descriverà quali componenti sono stati utilizzati per realizzare il progetto e gli aspetti software del progetto stesso.

## 4.1 Formulazione del problema

Il problema dinanzi al quale ci troviamo è quello della localizzazione sonora. Affinché si possa creare il prototipo per questa tesi occorre focalizzare l'attenzione su due aspetti principali. In primo luogo, per individuare la posizione della sorgente sonora i microfoni devono essere in grado di analizzare il suono in arrivo; dunque devono essere in grado di rilevare l'ampiezza del suono che poi, lato software, verrà utilizzata per effettuare il calcolo della FFT (Fast Fourier Transform). Fatto ciò, verrà prelevata l'ampiezza alla frequenza fondamentale rilevata da ogni singolo microfono per effettuare il calcolo dell'errore che è necessario per poter impartire il comando per effettuare la rotazione del servomotore. Ovviamente, il sistema non seguirà tutti i suoni presenti nell'ambiente circostante ma solo quelli aventi una frequenza di 1kHz. Tutti i suoni aventi una frequenza diversa da quest'ultima verranno filtrati attraverso un filtro passa banda. Una volta che l'entità dell'errore è stata determinata, in secondo luogo, occorre mettere in moto il servomotore che dovrà essere comandato in modo da ruotare verso la direzione dalla quale proviene il suono e arrestarsi proprio nell'istante in cui l'ampiezza rilevata dal microfono centrale diventa maggiore delle ampiezze rilevate dal microfono destro e dal microfono sinistro.

## 4.2 Struttura generale

Per quanto concerne la struttura generale si può affermare che la costruzione è costituita, fondamentalmente, da una piattaforma su cui ho provveduto ad installare i vari sensori. La piattaforma ha una struttura rettangolare in cui la distanza tra il lato destro e il lato sinistro è di circa 50 cm. Inoltre, sulle facce laterali del rettangolo, ossia su quella sinistra e su quella destra sono stati installati due dei tre microfoni a disposizione per il progetto, mentre sulla faccia centrale del rettangolo è stato installato il terzo e ultimo microfono.

Sempre sulla faccia centrale sono stati installati altri due sensori: sensore a ultrasuoni e il sensore IR (a infrarossi). Il primo mi è stato utile per poter implementare il Sonar mentre il secondo mi è stato utile per poter implementare la possibilità di acquisire un segnale emesso

da un telecomando che fa switchare il sistema dalla modalità Sonar alla modalità localizzazione sonora. Infine, la piattaforma è stata installata su un servomotore che, quindi, sarà responsabile della rotazione di tutta la struttura.

# 4.3 Hardware e elettronica

L'elenco dei componenti può essere visualizzato di seguito. Essi verranno descritti in dettaglio nei paragrafi seguenti. Lo schematico contenente tutti i componenti elettrici utilizzati è mostrato in figura 4.1

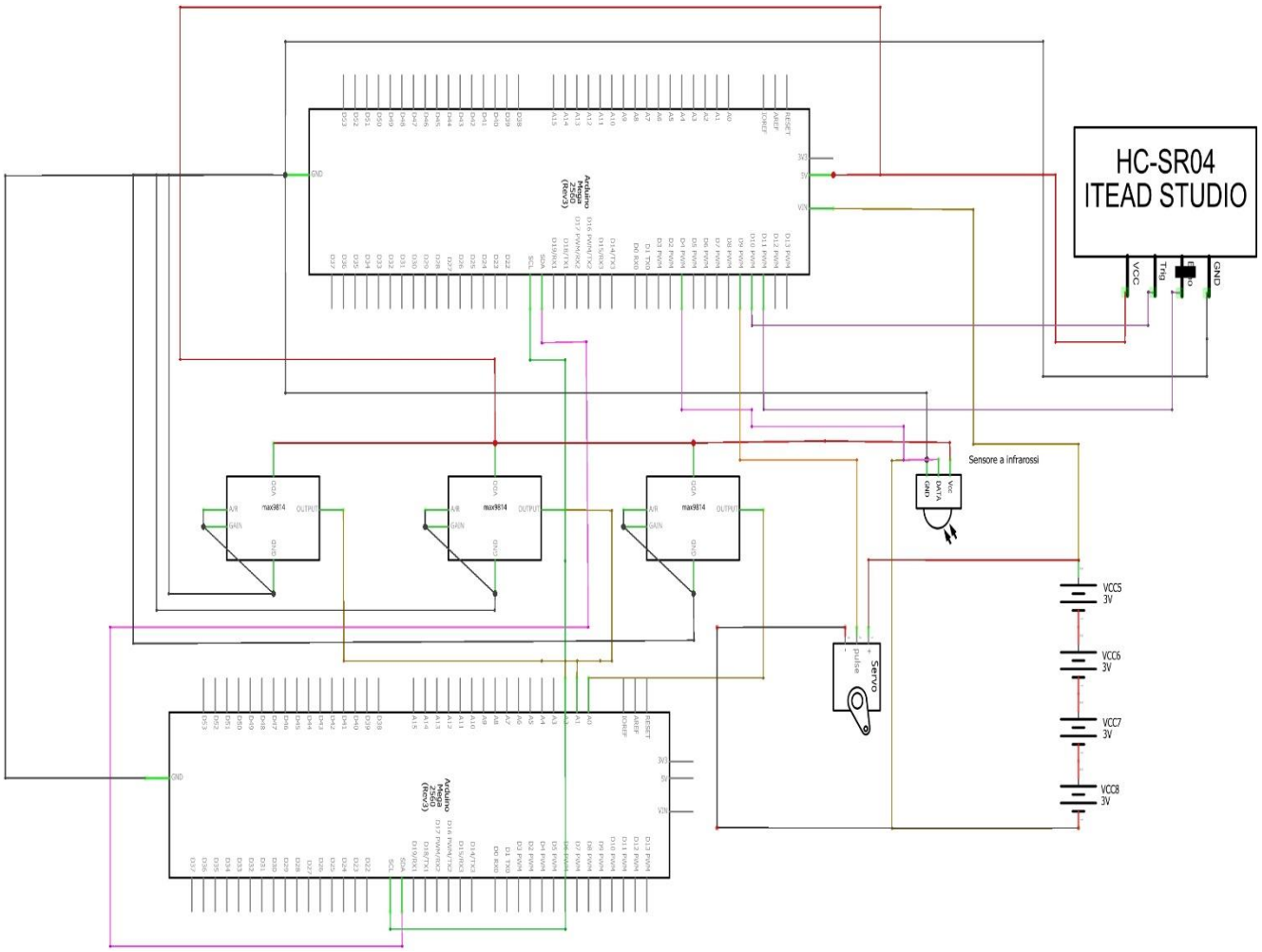


Figura 4.1: Schematico raffigurante i componenti elettronici utilizzati.

I componenti che sono stati utilizzati per realizzare il progetto sono: 2 Arduino Mega, servomotore, 3 microfoni, sensore a ultrasuoni, sensore IR (a infrarossi) e batteria da 12V. Ovviamente, per poter alimentare un servomotore di questa portata i 5V forniti da Arduino non sono sufficienti, ecco perché ho deciso di utilizzare una batteria da 12V. In pratica, per alimentare tutti i componenti, eccetto il servomotore, utilizzo i 5V forniti da Arduino; mentre per alimentare il servomotore utilizzo una batteria o un alimentatore da 12V.

### 4.3.1 Arduino Mega

Arduino Mega 2560 è composto da un microcontrollore ATmega2560, da 54 pin digitali e 16 analogici, 4 porte seriali UART, una porta USB e un jack di alimentazione, un header ICSP e un pulsante di reset. Ha tre tipi di memoria: Flash, SRAM ed EPROM. La scheda lavora ad una tensione nominale di 5V e sopporta una corrente massima di 40 mA [16]. Per quanto riguarda le dimensioni fisiche, Arduino Mega, rispetto al suo predecessore, è più esteso soprattutto in lunghezza, misurando 101.52 mm, mentre la larghezza è pressoché invariata (53.3 mm). Il peso è di 37g. Per poter utilizzare la scheda è sufficiente collegarla ad un PC tramite un cavo USB o alimentarla tramite una sorgente esterna. Il cervello dell'Arduino è rappresentato da un microcontrollore che, nella fattispecie, è denominato ATMEGA2560. Esso è un microcontrollore a 8 bit di tipologia AVR-RISC che integra una memoria flash da 256 KB per l'immagazzinamento del codice, un SRAM da 8 KB e una EPROM da 4KB (che possono essere letti tramite la libreria EPROM), un oscillatore, delle porte di I/O e componenti aggiuntivi come ADC, DAC, contatori e timer. Il dispositivo è in grado di raggiungere un throughput di 16 MIPS a 16 MHz operando ad una tensione nominale di 5V.

ATMega per poter funzionare ha bisogno di un clock con una frequenza di 16 MHz; per questo motivo è necessaria la presenza di un circuito oscillante ad alta frequenza. Per questo motivo troviamo integrato nella scheda un oscillatore, basato su oscillazioni di un materiale piezoelettrico (quarzo).

Per quanto concerne l'alimentazione, Arduino Mega può essere alimentato in 2 modi: tramite la porta USB o attraverso una sorgente esterna (adattatore AC-DC o batteria) utilizzando un power jack. In questo caso bisogna utilizzare un connettore da 2.1 mm. La tensione di funzionamento della scheda è compresa tra i 6 e i 22 volts, tuttavia nel caso si alimentasse la scheda con una tensione inferiore ai 7V o superiore ai 12V potrebbero verificarsi dei malfunzionamenti o instabilità. Per quanto riguarda la tensione di output, è possibile scegliere tra due differenti valori: 5V e 3.3V.

Sono disponibili quattro pin di alimentazione:

- **Pin VIN**, corrisponde alla tensione di ingresso della scheda quando si decide di alimentarla tramite sorgente esterna. Più nel dettaglio, è possibile usufruire di tale tensione nel caso l'input arrivi tramite il power jack.
- **Pin 5V**, è utilizzato per alimentare il microcontrollore o gli altri componenti della scheda con una tensione stabilizzata a 5V.
- **Pin 3.3V**, come il precedente, ma utilizzato nel caso di componenti che necessitano di questo tipo di voltaggio.
- **Pin GND**, corrisponde al pin negativo, ovvero la massa.


Figura 4.2: Arduino Mega 2560.

### 4.3.2 Microfoni utilizzati

Per realizzare il progetto è stato scelto un microfono a elettrete che offre un rapporto qualità-prezzo molto alto. Esso è denominato "Adafruit AGC Electret Microphone Amplifier – MAX9814" [17].

Si tratta di un microfono economico, di alta qualità, dotato di un amplificatore interno e con controllo automatico del guadagno (AGC). Il MAX9814 è costituito da diversi circuiti distinti: un preamplificatore a basso rumore, un amplificatore di guadagno variabile (VGA), un amplificatore di uscita, un microfono e circuiti di controllo AGC. Un generatore di tensione di polarizzazione del microfono interno fornisce una tensione di 2V adatta alla maggior parte dei microfoni a elettrete. Tale microfono amplifica l'ingresso in tre fasi distinte:

- Nel primo stadio, l'ingresso è bufferizzato e amplificato attraverso il preamplificatore a basso rumore avente guadagno pari a 12 dB.
- Il secondo stadio è costituito da un amplificatore di guadagno variabile (VGA) controllato dall'AGC (circuito di controllo). La combinazione VGA/AGC è in grado di variare il guadagno da 20 a 0 dB.
- Il terzo e ultimo stadio è rappresentato da un amplificatore di uscita in cui viene programmato un guadagno fisso di 8,18 o 20 dB mediante un singolo ingresso logico a tre livelli.

Senza compressione attuata da parte dell'AGC, il MAX9814 è in grado di fornire 40, 50 o 60 dB di guadagno, semplicemente, collegando il connettore GAIN in maniera differente a seconda del guadagno che si vuole ottenere.

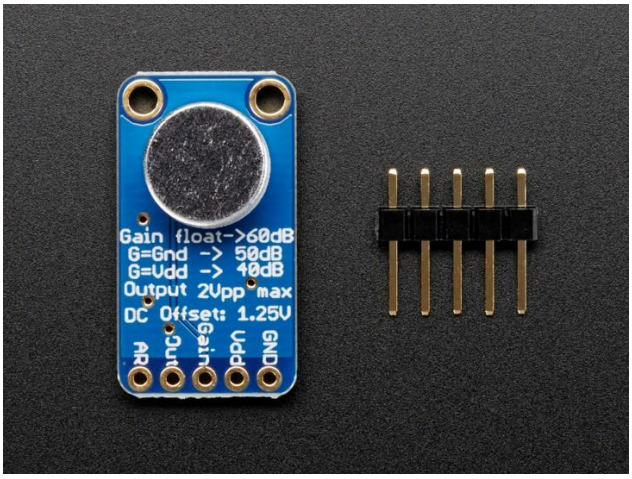


Figura 4.3: Adafruit AGC Electret Microphone Amplifier – MAX9814.

Come si nota dalla figura 4.4, il microfono è caratterizzato da 5 connettori: il primo (GND) rappresenta il terminale negativo del microfono e va collegato direttamente a massa, il secondo (VDD) rappresenta il terminale positivo del microfono e va collegato direttamente all'alimentazione, che in questo caso è di 5V, il terzo rappresenta il guadagno che può essere impostato in modo da assumere tre valori differenti: 40, 50 o 60 dB. Per ottenere un guadagno di 40 dB occorre collegare il connettore a VDD (alimentazione di 5V), per ottenerne uno di 50 dB occorre collegare il connettore a GND (massa), per ottenerne uno di 60 dB basta, semplicemente, non collegare il connettore. Nella fattispecie, facendo diversi test, ho scoperto che il valore del guadagno ottimale è di 50 dB, infatti il connettore è stato collegato a GND. Il quarto connettore (OUT) è stato collegato ad un PIN analogico dell'Arduino mentre il connettore A/R è stato collegato a massa. Questi collegamenti, ovviamente, sono stati fatti per tutti i microfoni utilizzati per realizzare il progetto.

# 4.3.3 Servomotore

Nella robotica per gli azionamenti, sono molto utilizzati i servomotori. Di solito questi si presentano come piccoli contenitori di materiale plastico da cui fuoriesce un perno in grado di ruotare di un angolo compreso tra 0 e 180° mantenendo stabilmente la posizione raggiunta. Per ottenere la rotazione del perno è utilizzato un motore a corrente continua e un meccanismo di demoltiplica che consente di aumentare la coppia in fase di rotazione. La rotazione del motore è effettuata tramite un circuito di controllo interno in grado di rilevare l'angolo di rotazione raggiunto dal perno tramite un potenziometro resistivo e bloccare il motore sul punto desiderato. Il motore e il potenziometro sono collegati al circuito di controllo e l'insieme di questi tre elementi definisce un sistema di feedback ad anello chiuso. Il circuito e il motore vengono alimentati da una tensione continua stabilizzata, in genere di valore compreso tra 4,8V e 6V, anche se molti motori sono in grado di accettare input di alimentazione fino a 7,2V. Per far girare il motore occorre inviare un segnale digitale al circuito di controllo. In questo modo si attiva il motore che, attraverso una serie di ingranaggi, è collegato al potenziometro. La posizione dell'albero del potenziometro indica una misura della posizione dell'albero motore del servomotore. Quando il potenziometro raggiunge la posizione desiderata, il circuito di controllo spegne il motore. È facile dedurre che i servomotori vengono progettati in genere per effettuare una rotazione parziale piuttosto che impostare un moto rotatorio continuo, come nel caso di un motore in continua o passo-passo. Anche se è possibile configurare un servomotore R/C perché ruoti in modo continuo, l'impiego fondamentale di un servomotore consiste nel raggiungere una posizione accurata dell'albero del motore, con movimenti compresi nell'intervallo tra 0° e 180°. Anche se questo movimento non sembra considerevole, può risultare più che sufficiente per manovrare un robot, per sollevare e abbassare le gambe, per ruotare un sensore che deve esaminare ciò che le circonda e molto altro ancora. La rotazione precisa di un angolo da parte di un servomotore in risposta a determinati segnali digitali rappresenta una delle funzionalità più sfruttate in tutti i campi della robotica. Normalmente un servomotore è costituito da un unico connettore a tre terminali, da saldare su un circuito stampato oppure da inserire in una breadboard. Due di questi terminali sono riservati all’alimentazione in corrente continua. Di solito, il positivo è di colore rosso, il negativo di colore nero e il terzo filo, normalmente di colore bianco, è riservato per il controllo del posizionamento. Tramite il terminale di controllo è necessario applicare un segnale impulsivo o PWM (Pulse Width Modulation). Generalmente con un impulso di durata pari a 1.5 ms il perno del servomotore si pone esattamente al centro del suo intervallo di rotazione.

Da questo punto, il perno può ruotare in senso antiorario se l'impulso fornito ha una durata inferiore a 1.5ms e in senso orario se l'impulso fornito ha durata superiore a 1.5ms.

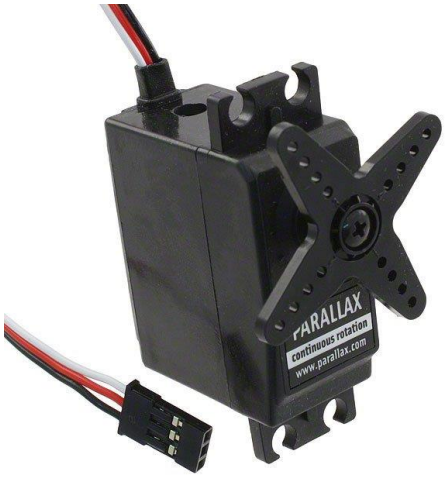


Figura 4.4: Esempio di servomotore.

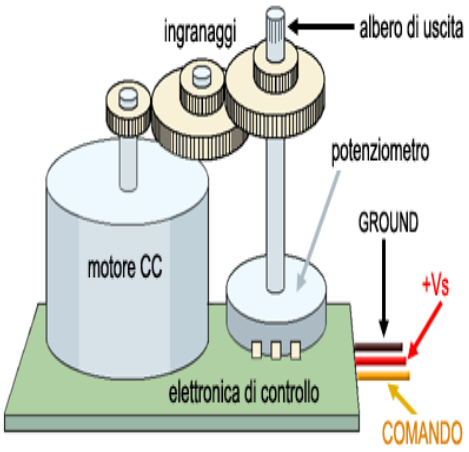


Figura 4.5: Struttura di un servomotore.

Il potenziometro del servomotore svolge un ruolo fondamentale che consente di stabilire l'istante in cui il motore ha impostato l'albero nella posizione desiderata. Questo potenziometro è fisicamente collegato all'albero di uscita del motore; in alcuni servomotori, l'albero del potenziometro coincide con l'albero stesso del motore. In questo modo, la posizione del potenziometro coincide precisamente con quella dell'albero del motore. Nel servomotore il potenziometro è configurato come un partitore di tensione e fornisce al circuito di controllo una tensione che varia in funzione della variazione dell'uscita del servomotore. Il circuito di controllo del servomotore mette in relazione questa tensione con la temporizzazione degli impulsi digitali di ingresso e genera un segnale di errore nel caso in cui debba correggere la tensione da inviare al motore. Questo segnale di errore è proporzionale alla differenza rilevata tra la posizione del potenziometro e la temporizzazione definita dal segnale in ingresso. Per compensare questa differenza, il circuito di controllo applica al motore un segnale che tiene conto di questo errore. Quando la tensione del potenziometro e la temporizzazione degli impulsi digitali coincidono, il segnale di errore viene annullato e il motore si ferma.


Figura 4.6: Servomotore utilizzato per lo sviluppo del progetto.

# 4.3.4 Sensore a ultrasuoni

Un sensore a ultrasuoni (figura 4.7) viene utilizzato per calcolare la distanza da un oggetto. Il funzionamento del sensore è il seguente: esso emette un impulso sonoro ad alta frequenza che è al di sopra del range di frequenze udibili dall'orecchio umano. Quando l'impulso viene riflesso da un oggetto che interferisce, quest'impulso sonoro riflesso viene ricevuto dal sensore, il quale registra il tempo che è intercorso tra l'emissione dell'impulso sonoro e la ricezione dello stesso, riflesso. La distanza può, quindi, essere rilevata utilizzando due dati: la velocità del suono che è nota e il tempo necessario affinché l'ultrasuono che è stato emesso all'istante di tempo zero possa essere riflesso e possa arrivare nuovamente al sensore [18].


Figura 4.7: Sensore a ultrasuoni utilizzato per realizzare il progetto.

Il sensore ad ultrasuoni è stato di fondamentale importanza per la realizzazione del progetto in questione, in quanto ha consentito di realizzare un sistema che non solo riesce ad orientarsi e a seguire la sorgente sonora ma riesce anche a rilevare gli oggetti presenti all'interno dell'ambiente circostante in un range che va da 0 a 180 gradi mostrando, di volta in volta, la distanza dagli oggetti rilevati.

# 4.3.5 Sensore IR (a infrarossi)

Il sensore utilizzato per realizzare questo progetto è l'AX-1838HS che è un ricevitore a infrarossi miniaturizzato per il controllo tramite telecomando e per quelle applicazioni che richiedono un elevato grado di reiezione alla luce ambientale. Il diodo PIN separato e il preamplificatore sono assemblati su un singolo leadframe. Il package epossidico contiene uno speciale filtro IR. Questo modulo ha prestazioni eccellenti anche nel caso di applicazioni che lavorano in ambienti in cui vi è un'azione di disturbo da parte della luce ambientale e fornisce protezione contro impulsi di uscita incontrollati [19].

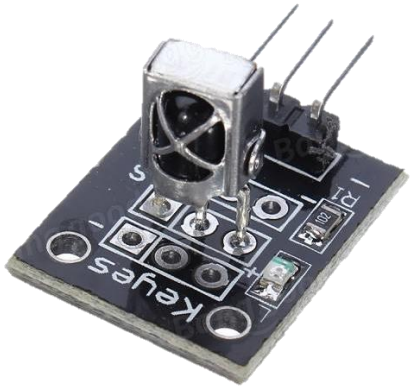

Figura 4.8: Sensore a infrarossi utilizzato per realizzare il progetto.

Dopo aver descritto il sensore a infrarossi utilizzato per realizzare il progetto, occorre citare il meccanismo di funzionamento di una trasmissione ad infrarossi che consiste nella generazione di un fascio luminoso, ad elevata frequenza, ad esempio 38 KHz (per i telecomandi tipici per la TV) che viene proiettato contro il ricevente (in questo caso il sensore ad infrarossi). Questo fascio luminoso non è altro che una sequenza di bit (0 e 1) che vengono codificati dal ricevente, attraverso la polarizzazione di photo-transistor. In ricezione, il ricevente capterà un segnale molto simile, se non uguale a quello inviato, e, in base alla corrispondenza che ha in memoria, sarà in grado di svolgere la richiesta effettiva, come ad esempio accendere la TV, o comandare un servomotore. Il fascio luminoso che viene inviato dal trasmettitore non si può vedere ad occhio nudo poiché ha una frequenza troppo elevata per l'essere umano. È stato deciso di utilizzare questa tipologia di sensore in quanto il sistema che è stato sviluppato prevede due diverse modalità di funzionamento che possono essere abilitate in maniera mutuamente esclusiva, semplicemente, premendo un pulsante del telecomando. Come detto nei capitoli precedenti, la prima modalità è stata denominata "localizzazione sonora" e permette al sistema di localizzare la sorgente sonora, mentre la seconda è stata denominata "Sonar"; quest'ultima consente di rilevare tutti gli oggetti presenti nell'ambiente circostante in un range compreso tra 0 e 180 gradi.

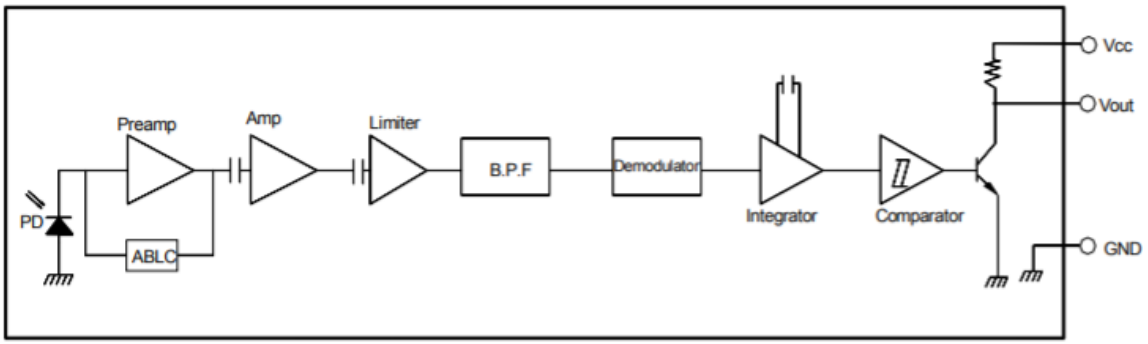


Figura 4.9: Schema a blocchi di un sensore IR.

## 4.4 Software

In questo paragrafo descriverò tutti gli aspetti software che caratterizzano il progetto. In particolare, descriverò come calcolare l'FFT, come calcolare l'angolo di rotazione del servomotore, come realizzare l'interfaccia grafica che sta alla base del Sonar e soprattutto come aumentare la frequenza di campionamento dell'Arduino, fattore che, purtroppo, rappresenta una grande limitazione dell'Arduino.

### 4.4.1 Frequenza di campionamento

Un fattore di notevole importanza per l'accuratezza del processo di localizzazione è la velocità con cui Arduino è in grado di acquisire un suono in entrata ma soprattutto quanto ritardo introduce il programma in fase di esecuzione. Visto che la determinazione dell'angolo di rotazione del servomotore si basa sul fatto che venga calcolata la differenza tra i valori di ampiezza rilevati dai microfoni, è importante che il programma sia in grado di eseguire le istruzioni che consentono di processare i valori acquisiti dai microfoni nel minor tempo possibile. Per aumentare la frequenza di campionamento, è possibile aumentare la frequenza di clock dell'Arduino. Questo può essere attuato modificando il cosiddetto "prescale factor" che dal valore 126 può essere portato fino a 16 o 8. In pratica si può incrementare la frequenza di campionamento fino a 16 volte il suo valore originario. Di seguito, il codice usato per aumentare la frequenza di campionamento di Arduino (figura 4.10), settando il prescale a 8.

```
#define FASTADC 1
// defines for setting and clearing register bits
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif


void setup() {
    Wire.begin();
    Serial.begin(115200);
    #if FASTADC
    // set prescale to 8
    cbi(ADCSRA, ADPS2);
    sbi(ADCSRA, ADPS1);
    sbi(ADCSRA, ADPS0);
    #endif
```

Figura 4.10: Codice utilizzato per aumentare la frequenza di campionamento di Arduino [20].

### 4.4.2 Controllo tramite telecomando

In questo paragrafo descriverò il codice utilizzato per fare in modo che il sistema possa essere dotato di due modalità di funzionamento differenti: la prima è denominata “Localizzazione sonora” mentre la seconda è denominata “Sonar”. Nella prima modalità il sistema è in grado di localizzare una sorgente sonora, per poi orientarsi verso di essa. Nella seconda modalità il sistema è in grado sia di localizzare tutti gli oggetti presenti nell’ambiente circostante in un range che va da 0 a 180 gradi sia di calcolare la distanza da essi. Per poter switchare da una modalità all’altra è possibile utilizzare un telecomando. In particolare, premendo il tasto 0 si passa alla modalità localizzazione sonora mentre premendo il tasto 1 si passa alla modalità sonar. Come si nota dalla figura 4.11, dopo aver incluso la libreria per la gestione del sensore IR, occorre dichiarare degli oggetti che saranno utili per la ricezione del segnale inviato dal telecomando e per la decodifica del segnale ricevuto.

```
//Inclusione delle librerie
#include "IRremote.h"

/*( Declare objects )*/
IRrecv irrecv(receiver);
decode_results results;
```

Figura 4.11: Codice per l’inclusione della libreria e la dichiarazione degli oggetti.

Dopo aver fatto ciò, occorre definire una funzione che si occupa di codificare il segnale ricevuto dal telecomando. In particolar modo, la codifica che è stata scelta consiste nel fatto che, se è stato premuto il tasto 0, la funzione dovrebbe restituire 0, altrimenti dovrebbe restituire 1 (nel caso in cui sia stato premuto il tasto 1 del telecomando) proprio come si nota in figura 4.12.

```
/*-----( Function )-----*/
char translateIR() // takes action based on IR code received
{
  switch(results.value)
  {
  case 0xFF6897: return '0'; break;
  case 0xFF30CF: return '1'; break;
  }// End Case
  delay(500);
}
```

Figura 4.12: Codice per la realizzazione della funzione che effettua la codifica del segnale ricevuto.

Un'altra porzione di codice, che vale la pena citare e che verrà utilizzata tantissimo all'interno del loop, è quella presente all'interno della figura 4.13

```
if (irrecv.decode(&results)) // have we received an IR signal?
  {
    scelta = translateIR();
    irrecv.resume(); // receive the next value
  }
```

Figura 4.13: Codice utilizzato per acquisire la scelta fatta, ossia il numero del tasto che è stato premuto.

Questo codice che, come detto, verrà molto utilizzato all'interno del loop, consente di acquisire il numero del tasto che è stato premuto e di ricevere il valore successivo.
Sulla base del tasto che è stato premuto, il cui valore viene memorizzato in una variabile denominata "scelta", verranno eseguite due porzioni di codice differenti che verranno descritte nelle pagine seguenti. Come detto, se l'utente preme il tasto 0 e dunque scelta è uguale a zero, verrà eseguito il codice che consente di attuare la localizzazione sonora, altrimenti, nel caso di scelta uguale a 1, verrà eseguito il codice che consente la realizzazione del sonar.

### 4.4.3 FFT (Fast Fourier Transform)

Il punto focale su cui ruota tutta la tesi è proprio la FFT. Essa è uno strumento che consente di effettuare un'analisi del segnale acquisito non più nel dominio del tempo ma nel dominio della frequenza. La FFT si è rivelata uno strumento molto potente e performante in quanto, come è noto, effettuare un filtraggio nel dominio del tempo è possibile ma è anche molto complesso soprattutto a causa della presenza del rumore mentre effettuare un filtraggio nel dominio della frequenza è molto più semplice in quanto i segnali che vengono usati per testare questo progetto sono dei monotonali, ossia dei suoni aventi una sola frequenza denominata frequenza fondamentale. Dunque, dopo essere passati nel dominio della frequenza basta mettere un filtro passa banda che fa passare soltanto le frequenze comprese in un certo range e il gioco è fatto. In questa tesi, l'unica frequenza che non viene filtrata è quella di 1kHz; tutte le altre vengono filtrate e, dunque, non vengono prese in considerazione. Ciò significa che se il suono acquisito ha una frequenza di 1kHz il sistema si mette in moto per localizzarlo e seguirlo mentre se ha una frequenza diversa da 1kHz allora il sistema resta immobile e non lo segue, anche perché potrebbe trattarsi, semplicemente, di rumore.

Altro fattore che bisogna citare è il fatto che se da una parte l'FFT è molto potente ed utile, d'altra parte è anche molto onerosa dal punto di vista computazionale. A causa di ciò, ho dovuto pensare ad una soluzione per poter migliorare e rendere più performante il codice. L'idea che mi è venuta in mente è stata quella di parallelizzare l'esecuzione del codice, semplicemente utilizzando due schede Arduino Mega. In una scheda è stato eseguito il codice relativo all'acquisizione e all'elaborazione dei segnali con conseguente calcolo della FFT, mentre nell'altra è stato eseguito il codice di pilotaggio del servomotore. Ovviamente, come ogni comunicazione che si rispetti necessita di un protocollo per lo scambio dei dati tra gli attori della comunicazione che in questo caso sono le due schede Arduino. Per la scelta del tipo di comunicazione mi sono trovato dinanzi ad un bivio; se da un lato potevo utilizzare la seriale, dall'altro potevo disporre del protocollo I2C. Alla fine ho optato per il protocollo I2C in quanto consente di instaurare una comunicazione e effettuare una trasmissione dei dati molto più velocemente rispetto alla stessa effettuata con l'ausilio della seriale. Ciò che è importante sottolineare è il fatto che il protocollo I2C si basa sulla presenza di due attori della comunicazione: un Master e uno Slave. Come suggeriscono i due termini, il tipo di comunicazione che si viene ad instaurare non è tra pari ma è una comunicazione in cui vi è un certo dislivello tra i dispositivi coinvolti, in quanto ad esempio è il Master che inizializza la comunicazione mediante una richiesta fatto allo Slave che, in seguito alla ricezione della richiesta esegue una funzione responsabile della trasmissione di un certo byte.

In questo paragrafo parlerò del codice utilizzato per il calcolo della FFT e del codice utilizzato per effettuare la trasmissione dei dati all'altra scheda Arduino; codici che sono stati eseguiti sulla scheda Arduino Slave.

Come si nota in figura 4.14, la parte iniziale del codice è relativa alla dichiarazione di oggetti su cui verranno invocate le funzioni necessarie per il calcolo della FFT e all'inclusione delle librerie necessarie per il calcolo della FFT e per l'instaurazione della comunicazione tramite protocollo I2C ("Wire.h").

```
//SLAVE
#include <Wire.h> //Protocollo I2C
#include <arduinoFFT.h>
#define SAMPLES 128
#define SAMPLING_FREQUENCY 2500

arduinoFFT FFT0 = arduinoFFT();
arduinoFFT FFT1 = arduinoFFT();
arduinoFFT FFT2 = arduinoFFT();
```

Figura 4.14: Dichiarazione delle variabili e inclusione delle librerie – Slave.

```
void loop(){
    for(int i=0; i<SAMPLES; i++)
    {
        microseconds = micros();
        vReal0[i] = analogRead(0);;
        vImag0[i] = 0;
        vReal1[i] = analogRead(1);
        vImag1[i] = 0;
        vReal2[i] = analogRead(2);
        vImag2[i] = 0;

        while(micros() < (microseconds + sampling_period_us)){
        }
    }
    /*FFT*/
    FFT0.Windowing(vReal0, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
    FFT0.Compute(vReal0, vImag0, SAMPLES, FFT_FORWARD);
    FFT0.ComplexToMagnitude(vReal0, vImag0, SAMPLES);
    FFT0.DCremoval(vReal0);
    FFT0.MajorPeak(vReal0, SAMPLES, SAMPLING_FREQUENCY, &x0, &v0); //frequenza fondamentale
    FFT0.BandPassFilter(vReal0,SAMPLES, SAMPLING_FREQUENCY,970, 1100,x0,&v0);
```

Figura 4.15: Codice per il calcolo della FFT.

Dalla figura 4.15 si evince il codice necessario per il calcolo della FFT. Nella prima parte del codice viene effettuata l'acquisizione dei campioni nel dominio del tempo rilevati dai tre microfoni; una volta effettuata l'acquisizione, si passa dal dominio del tempo a quello della frequenza mediante l'invocazione di una serie di funzioni sull'oggetto FFT0. Le tre funzioni importanti sono "Compute" che effettivamente effettua il calcolo della FFT, "MajorPeak" che restituisce la frequenza fondamentale del segnale acquisito ma soprattutto l'ampiezza alla frequenza fondamentale e "BandPassFilter" che effettua il filtraggio del segnale acquisito facendo passare soltanto i suoni aventi una frequenza di circa 1kHz. Ovviamente, sono stati istanziati tre oggetti in quanto verrà effettuato il calcolo della FFT per ogni singolo microfono.

In figura 4.16 si può notare il codice utilizzato dallo Slave per inizializzare la comunicazione e per porsi in attesa di una richiesta da parte dell'Arduino Master. In particolar modo, mediante la funzione "begin" ho provveduto ad assegnare un indirizzo allo Slave. Nella fattispecie, l'indirizzo assegnato è il 3. Ho dovuto assegnare un indirizzo a causa del fatto che il protocollo I2C consente di instaurare una comunicazione in cui vi è un Master e tanti Slave. Dunque, il Master deve sapere a quale dei tanti dispositivi inviare la sequenza di bytes. Ma come fa a saperlo? In pratica ad ogni attore della comunicazione viene assegnato un indirizzo univoco che lo identifica e che consente al Master di contattarlo.

```
Wire.begin(3); //L'indirizzo assegnato all'Arduino slave è 3
Wire.onRequest(InviaAmpiezze);
```

Figura 4.16: Codice per l'inizializzazione della comunicazione.

Sempre dalla figura 4.16 si nota che è stata utilizzata anche un'altra funzione denominata "onRequest". Quest'ultima consente di porsi in attesa di una richiesta proveniente dal Master che non appena sarà sopraggiunta comporterà l'esecuzione di una funzione denominata "InviaAmpiezze" mediante la quale lo Slave fornirà i dati al Master.

```
void InviaAmpiezze(){
if (v0 == 0 && v1 == 0 && v2 == 0){
t[2] = 'd';
}
else{
  t[2] = 'c';
}
if (v1 < v0){
  t[0] = 's';
}
else {
  t[0]= 'n';
}
if (v2 > v1 && v2 > v0){
  t[1]= 'a';
}
else{
  t[1] = 'b';
}
strncpy(temp,t,3);
//Serial.println(temp);
Wire.write(temp);
}
```

Figura 4.17: Funzione per la trasmissione dei dati al Master.

La funzione "InviaAmpiezze" verrà eseguita non appena sarà pervenuta una richiesta da parte del Master. Occorre fare una premessa, questa funzione contiene dei controlli che avrei potuto benissimo implementare all'interno del codice in esecuzione sul Master; ho deciso di implementarli direttamente sullo Slave in quanto trasferire le ampiezze dallo Slave al Master per poi effettuare i controlli direttamente sul Master è più complesso che effettuare i controlli direttamente sullo Slave, questo perché traferire dei caratteri tramite il protocollo I2C è molto più semplice rispetto al trasferimento di numeri con o senza virgola. Per evitare queste complicazioni, le ampiezze calcolate non vengono trasferite al Master ma permangono nello Slave che effettua i vari controlli per capire se il servomotore deve ruotare verso sinistra o verso destra e se il servomotore deve arrestare la sua rotazione in quanto ha localizzato la sorgente sonora. In relazione ai risultati che vengono prodotti in seguito alla verifica delle condizioni, all'interno di un vettore, in posizioni diverse, vado a memorizzare dei caratteri

che, a sua volta, verranno messi in serie in modo da formare una stringa. Una volta formata la stringa, byte per byte, viene effettuata la trasmissione al Master.
Di conseguenza, ciò che il Master riceve non è il valore delle ampiezze rilevate dai tre microfoni ma una stringa che a seconda dei caratteri che contiene permette al Master di capire se il servomotore deve andare verso destra, verso sinistra o deve arrestarsi in quanto ha localizzato correttamente la sorgente sonora. Ho deciso di attuare questa tecnica per semplificare la trasmissione dei bytes, in quanto, come detto, trasmettere stringhe o caratteri è molto semplice mentre trasmettere numeri è molto più complesso.

### 4.4.4 Localizzazione sonora

In questo paragrafo verrà descritto il codice utilizzato per attuare la localizzazione sonora.
La prima parte di codice è riservata alla dichiarazione delle variabili necessarie per poter raggiungere lo scopo proprio come si nota in figura 4.18

```
char v0[5] =  {};
int d;
int i;
int olddelta = 0;
int newdelta;
Servo myservo;
const int trigPin = 10;
const int echoPin = 11;
// Variables for the duration and the distance
long duration;
int distance;
int receiver = 5; // Signal Pin of IR receiver to Arduino Digital Pin 5
char scelta;
char ritardo[2] = {};
int ritardo_int;
```

Figura 4.18: Dichiarazione preliminare delle variabili.

Come si nota osservando la figura, la prima parte di codice è quella relativa alla dichiarazione delle variabili. In particolare, il vettore denominato v0[5] è molto importante in quanto contiene i caratteri inviati dallo Slave che, come detto, consentono al Master di capire se il robot debba ruotare verso destra, verso sinistra o debba arrestare la propria rotazione in seguito al raggiungimento della posizione desiderata.

```
void loop() {
Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
while (v0[2] == 'd'){
  myservo.detach();
  Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
}
myservo.attach(9);
```

Figura 4.19: Codice per l'acquisizione dei dati inviati dallo Slave.

Il codice evidenziato in figura 4.19 è molto importante in quanto mette in evidenza la comunicazione tra il Master e lo Slave. Come detto, la comunicazione avviene mediante il protocollo I2C. Come già affermato, la comunicazione non è tra pari ma è una comunicazione in cui abbiamo un Master e uno Slave; dunque, l'unico che può inizializzare una comunicazione è il Master, lo Slave attende soltanto di essere contattato dal Master. In tal caso, la funzione utilizzata per inizializzare la comunicazione e per fare una richiesta allo Slave prende il nome di "requestFrom" che accetta due parametri: il primo è l'indirizzo dello Slave da contattare mentre il secondo rappresenta il numero di bytes da acquisire una volta che lo Slave abbia risposto. Inoltre, l'acquisizione dei dati avviene per effetto di una funzione denominata "read" e una volta usciti dal ciclo while siamo sicuri che il vettore v0 contenga una stringa formata da diversi caratteri ciascuno dei quali assume un significato ben preciso. Abbiamo quindi capito che il codice contenuto all'interno del primo while consente di attuare l'acquisizione dei dati, vediamo adesso per cosa può essere utile il secondo while. Il codice contenuto all'interno di quest'ultimo consente di disabilitare il servomotore soltanto nell'ipotesi in cui v0[2] sia uguale a 'd'. Se questa condizione è verificata significa che dopo

aver effettuato il calcolo della FFT e aver effettuato il filtraggio, lo Slave ha capito che il segnale rilevato dai tre microfoni è stato totalmente filtrato in quanto caratterizzato da una frequenza diversa da 1kHz e dunque è inutile che il servomotore rimanga acceso perché tanto non si muoverebbe comunque. Sempre all'interno del secondo while è contenuto anche il codice necessario per poter effettuare l'acquisizione dei dati in quanto qualora in fase di esecuzione del codice contenuto all'interno del secondo while, ai tre microfoni si presentasse un suono avente frequenza di 1kHz allora si dovrebbe immediatamente uscire dal while in modo da abilitare istantaneamente il servomotore e far sì che il robot inizi a muoversi in modo da seguire la sorgente sonora. Ho deciso di spendere qualche parola in più per il codice utilizzato per effettuare l'acquisizione dei dati in quanto esso verrà utilizzato tantissimo all'interno del programma, ogni qualvolta vi sia la necessità di acquisire dei dati trasmessi al Master dallo Slave.

```
if (v0[0] == 's' && v0[2] == 'c'){ //if (v1 < v0)
Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
ritardo[0]= v0[3];
ritardo[1] = v0[4];
ritardo[2] = '\0';
ritardo_int = atoi(ritardo);
//Serial.println(ritardo_int);
for (i = olddelta; i < 180; i+=1){
  //Serial.println(ritardo_int);
delay(ritardo_int);
Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
```

Figura 4.20: Codice per attuare la rotazione da un certo angolo precedente a 180°.

Prima di commentare il codice contenuto in figura 4.20 occorre sottolineare che la tecnica utilizzata e che è stata implementata con l'ausilio del codice già parzialmente descritto è la seguente: nell'ipotesi in cui la frequenza del suono sia di un 1kHz, perché altrimenti il

servomotore non si muoverebbe, il robot inizia a ruotare verso la direzione dalla quale il suono percepito dai 3 microfoni è più elevato. Per capire verso quale direzione il servomotore debba ruotare ho utilizzato i microfoni montati nella parte laterale del robot; molto semplicemente, se l'ampiezza percepita dal microfono sinistro è maggiore di quella percepita dal microfono destro il servomotore inizia a ruotare verso sinistra, se, invece, l'ampiezza percepita dal microfono destro è maggiore di quella percepita dal microfono sinistro il servomotore inizia a ruotare verso destra. Le due situazioni sono state implementate mediante la verifica di due condizioni di cui una è quella mostrata in figura 4.20 che, se verificata, consente di effettuare la rotazione verso sinistra in quanto l'ampiezza rilevata dal microfono sinistro è maggiore di quella rilevata dal microfono destro. In questo paragrafo non verrà mostrato il codice utilizzato per implementare la seconda condizione in quanto è sostanzialmente identico a quello utilizzato per la verifica della prima condizione, l'unica differenza consiste nel fatto che stavolta l'ampiezza rilevata dal microfono destro è maggiore di quella rilevata dal microfono sinistro e dunque, sotto questa ipotesi, il servomotore deve ruotare verso destra. Abbiamo capito che nella fase iniziale il robot inizia a ruotare verso la direzione dalla quale il suono percepito è maggiore ma a questo punto la domanda sorge spontanea: come fa il robot ad arrestarsi non appena la sorgente sonora è situata proprio dinanzi ad esso? In pratica, non appena l'ampiezza rilevata dal microfono centrale diventa maggiore sia di quella rilevata dal microfono sinistro sia di quella rilevata dal microfono destro siamo sicuri che il robot si trova proprio dinanzi la sorgente sonora e dunque si deve arrestare. Per realizzare questo comportamento, lo stesso identico codice è stato implementato all'interno di quello utilizzato per la verifica delle due condizioni, dunque anche in questo caso lo mostrerò soltanto una volta.

Fatta questa premessa si può affermare che dalla figura 4.20 si evince che si esegue il codice contenuto all'interno della condizione soltanto se la stringa trasmessa dallo Slave, tra gli altri, contiene i due caratteri: 's' e 'c'. Se li contiene significa che l'ampiezza rilevata dal microfono sinistro è maggiore di quella rilevata dal microfono destro e, inoltre, la frequenza del suono rilevata dai tre microfoni è di 1kHz. Dopodiché viene nuovamente eseguito il codice per l'acquisizione dei dati in quanto come si vedrà sfogliando il paragrafo dei risultati ho deciso di implementare un regolatore che non fa altro che regolare la velocità di rotazione del servomotore agendo su un parametro denominato "delay" che aumentato o ridotto consente di far ruotare più lentamente o più velocemente il servomotore; ovviamente, il delay e dunque la velocità di rotazione è funzione dell'errore, ossia della differenza tra l'ampiezza rilevata dal microfono sinistro e quella rilevata dal microfono destro, calcolato direttamente dallo Slave. L'entità di questo ritardo viene passato al Master e viene memorizzato all'interno dell'array 'ritardo' per poi essere convertito in un numero intero.

Fatto ciò viene eseguito un ciclo for che effettivamente consente di attuare la rotazione del servomotore, in quanto il ciclo parte da un angolo precedente denominato “olddelta” che in pratica coincide con l’ultimo angolo che è stato raggiunto dal servomotore in fase di esecuzione del ciclo precedente e arriva a 180 gradi. All’interno del ciclo for, proprio come mostrato in figura 4.21, ad ogni iterazione viene effettuata la scrittura di un certo angolo che di volta in volta viene incrementato. Come detto, l’intervallo di tempo che intercorre fra un’iterazione e quella successiva e quindi la velocità di rotazione può essere modulata utilizzando il ‘delay’.

```
while (v0[2] == 'd'){
  myservo.detach();
  Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
}
    //delay(40);
    myservo.write(i);
    Serial.println(i);
```

Figura 4.21: Codice per la scrittura dell’angolo di rotazione.

Sempre da questa figura si nota come abbia inserito un ciclo while molto importante che fa sì che qualora in fase di rotazione, il robot si accorgesse di aver intercettato un suono a frequenza diversa da 1kHz si arresti istantaneamente in quanto, come già affermato, quest’ultimo deve seguire soltanto i suoni aventi come frequenza 1kHz.

```
if( v0[1] == 'a' && v0[2] == 'c'){
  while(v0[1] == 'a' && v0[2] == 'c'){
    //nuovo codice
    while(v0[1] == 'a' && v0[2] == 'c'){
    myservo.detach();
```

Figura 4.22: Codice per l’arresto del robot in seguito al raggiungimento della posizione desiderata.

In figura 4.22 ho voluto mettere in risalto la porzione di codice utilizzata per far sì che il robot si arresti non appena ha raggiunto la posizione desiderata.
Non appena la stringa inviata dallo Slave contiene il carattere ‘a’ che è indice del fatto che l’ampiezza rilevata dal microfono centrale è maggiore di quelle rilevate sia dal microfono destro sia dal microfono sinistro e sempre in presenza di un suono avente come frequenza 1kHz, si entra all’interno di un ciclo while all’interno del quale il servomotore viene disabilitato fino a quando lo Slave non invia una stringa contenente dei caratteri tutti diversi dal carattere ‘a’. In tal modo ho deciso di implementare l’arresto del robot in seguito al raggiungimento della posizione desiderata.

### 4.4.5 Sonar

In questo paragrafo verrà descritto il codice utilizzato per la realizzazione del Sonar. Per lo sviluppo di quest’ultimo è stato utilizzato Processing 3 e Arduino che è stato possibile far comunicare utilizzando la seriale di Arduino stesso. Dalla figura 4.23 si può notare come la realizzazione del Sonar, lato Arduino, sia molto semplice in quanto l’unica cosa che occorre fare è far ruotare il servomotore da 0 a 180 gradi e da 180 a 0 gradi, ininterrottamente. Come si nota dal codice, all’interno del ciclo for, che si occupa di definire l’angolo di rotazione del servomotore, di volta in volta, tramite seriale viene effettuato l’invio sia dell’angolo di rotazione del servomotore sia della distanza che vi è tra il sistema e l’oggetto rilevato; distanza che viene calcolata invocando una funzione denominata “calculateDistance”.

```
  // rotates the servo motor from 0 to 180 degrees
    for(int i=0;i<=180;i++){

    if (irrecv.decode(&results)) // have we received an IR signal?
    {
      scelta = translateIR();
      irrecv.resume(); // receive the next value
      if (scelta == '0') {
        scelta = '0';
        break;
      }
    }

    myservo.write(i);
    delay(30);
    distance = calculateDistance();
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance);
    Serial.print(".");
    }

  if (scelta == '0') break;

  // Repeats the previous lines from 180 to 0 degrees
  for(int i=180;i>0;i--){
     if (irrecv.decode(&results)) // have we received an IR signal?
  {
    scelta = translateIR();
    irrecv.resume(); // receive the next value
    if (scelta == '0') {
      scelta = '0';
      break;
    }
  }
  myservo.write(i);
  delay(30);
  distance = calculateDistance();
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
}//fine scelta 1 while
}//fine scelta 1
}
```

Figura 4.23: Codice per la realizzazione del Sonar, lato Arduino.

Dalla figura 4.24 si può notare il codice utilizzato per la realizzazione del Sonar, lato Processing.

```
import processing.serial.*; // Importo la libreria per la comunicazione seriale
import java.awt.event.KeyEvent; // Importo la libreria per l'acquisizione dei dati dalla seriale
import java.io.IOException;
Serial myPort; // Definizione di un oggetto seriale
// Dichiarazione delle variabili
String angle="";
String distance="";
String data="";
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;


void setup() {
 size (1920, 1050); // Impostazione della risoluzione del Sonar
 smooth();
 myPort = new Serial(this,"COM3", 9600); // Inizia la comunicazione seriale
 myPort.bufferUntil('.'); // Legge i dati dalla seriale fino a quando non trova il '.'.
 orcFont = loadFont("OCRAExtended-48.vlw");
}
```

Figura 4.24: Codice usato per l'inizializzazione del Sonar.

Da questa figura si evince che vi è una porzione di codice dedicata all'importazione delle librerie per la comunicazione e l'acquisizione dei dati tramite seriale, una porzione dedicata alla dichiarazione delle variabili e al setup, ossia all'inizializzazione del Sonar. Quest'ultima consiste nel settare la risoluzione dello schermo, inizializzare la comunicazione seriale mediante la definizione della porta utilizzata e acquisire i dati fino a quando non viene ricevuto il carattere '.'.

Inoltre, tra le funzioni importanti che sono state implementate troviamo:

- drawSonar(): Essa costruisce tutta l'interfaccia grafica del Sonar.
- drawLine(): Essa mostra le linee in verde che vengono mostrate all'interno dell'interfaccia grafica del Sonar.
- drawObject() che ha lo scopo di mostrare, all'interno del Sonar, gli oggetti di colore rosso qualora il sensore a ultrasuoni li rilevi.
- drawText(): Essa ha lo scopo di mostrare tutto il testo facente parte dell'interfaccia grafica del Sonar.

Ovviamente, è possibile consultare le funzioni che sono state implementate per la realizzazione del Sonar sfogliando le relative pagine contenute all'interno dell'Appendice A.

L’ultima porzione di codice (Figura 4.25) che vorrei mettere in risalto è quella relativa all’acquisizione dei dati tramite la seriale di Arduino; questo perché l’unico modo per far comunicare Arduino e Processing è l’utilizzo della porta seriale.

```
void serialEvent (Serial myPort) { // Inizia la lettura dei dati dalla porta seriale
  // Leggi i dati dalla seriale fino a quando non trovi il carattere '.' e inseriscili nella variabile data
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);

  index1 = data.indexOf(","); // Trova il carattere ',' e memorizzalo nella variabile "index1"
  angle= data.substring(0, index1); // Leggi i dati, dalla posizione "0" alla posizione contenuta nella variabile index1
  distance= data.substring(index1+1, data.length()); // Leggi i dati, dalla posizione "index1" fino alla fine dei dati,

  // Converti le stringhe in valori interi
  iAngle = int(angle);
  iDistance = int(distance);
}
```

Figura 4.25: Comunicazione seriale.

Dalla figura sovrastante si nota come venga effettuata l’acquisizione dei dati tramite seriale.
In particolare, avviene l’acquisizione dei due parametri principali: angolo e distanza, che all’atto della trasmissione degli stessi all’interno del codice Arduino sono separati dal carattere ‘,’ ma comunque costituiscono un’unica stringa che, all’atto della ricezione della stessa, deve essere presa e divisa in due sottostringhe contenenti, rispettivamente, angolo e distanza. Per far ciò, visto che il carattere separatore dell’angolo e della distanza è proprio la virgola, è stata utilizzata la funzione “substring” che a partire da una stringa ne genera un’altra contenente gli stessi caratteri della stringa di partenza ma fino ad una determinata posizione che in questo caso coincide proprio con quella della virgola.
In pratica, viene presa la stringa di partenza e, a partire da questa, vengono generate due sottostringhe; la prima contenente tutti i caratteri fino a prima della virgola mentre la seconda contenente tutti i caratteri a partire dalla virgola esclusa. Dopodiché viene effettuata la conversione dell’angolo e della distanza in valori interi. In conclusione, per dare l’idea al lettore di ciò che effettivamente è stato realizzato ho deciso di includere una figura contenente l’output che si otterrebbe se venisse eseguito il codice scritto all’interno di Processing 3 (Figura 4.26).

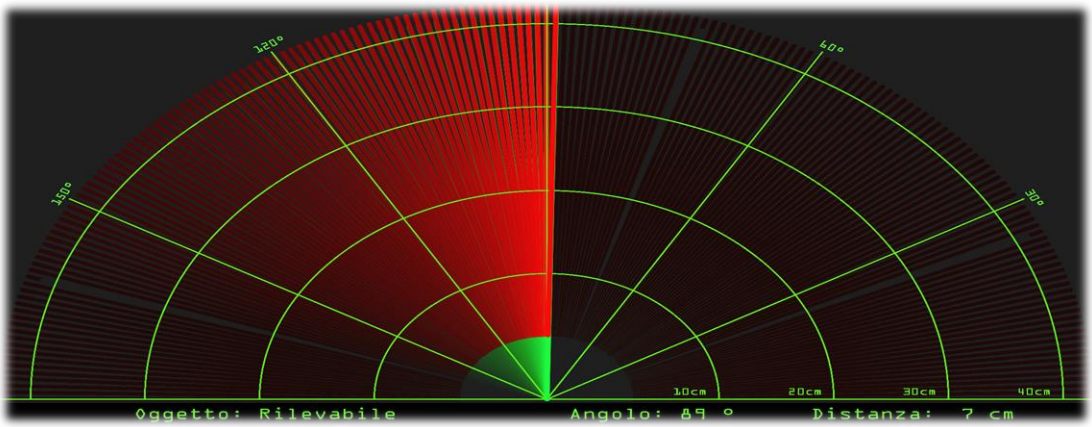


Figura 4.26: Sonar realizzato con Processing 3 e Arduino.

# Capitolo 5

# Risultati, discussione e conclusioni

In questo capitolo verranno descritti i risultati a cui sono giunto mediante l'ausilio del codice Matlab, verrà effettuata una discussione riguardante il progetto in generale, verranno tirate le somme del lavoro svolto e si parlerà di eventuali progetti futuri.

## 5.1 Risultati

Per quanto concerne i risultati a cui sono giunto, si può affermare che, in generale, i risultati raggiunti sono stati buoni in quanto la porzione di tesi relativa all'elaborazione del segnale si è rivelata efficace, efficiente e molto performante in quanto il calcolo della FFT e il filtraggio rappresentano un punto di forza di questa tesi. In seguito alle varie prove sperimentali che sono state svolte si è giunti a conclusione che il sistema è in grado di ruotare in modo da seguire il suono con un'accuratezza accettabile se consideriamo l'hardware a basso costo che è stato impiegato per realizzare il progetto. Dal punto di vista dell'accuratezza, svolgendo dei test al chiuso ho scoperto che il sistema ruota con una notevole precisione fino ad una distanza massima di circa 1.5/2 metri; precisione che tende a diminuire all'aumentare della distanza tra il sistema e la sorgente sonora. Ovviamente, facendo dei test in ambienti più silenziosi del normale è possibile ottenere una buona precisione anche a distanze superiori ai 2 metri. La maggior parte dei test sono stati condotti all'interno di ambienti al chiuso, ciò nonostante dai pochi test che sono stati condotti all'aperto è emesso il fatto che la precisione si riduce notevolmente rispetto a quella che si otterrebbe se gli stessi venissero condotti al chiuso. All'aperto, la differenza tra l'angolo desiderato e l'angolo di rotazione effettivo del servomotore si mantiene molto piccola in corrispondenza di distanze inferiori al metro. Se ci accingiamo ad allontanare la sorgente sonora portandola oltre il metro, il sistema inizia a mostrare segni di cedimento a causa della difficoltà di seguire la sorgente sonora.

Un altro fattore abbastanza importante che è stato messo in evidenza dai test condotti è il fatto che la rotazione del servomotore avveniva sempre alla stessa velocità e non teneva conto dell'errore, ossia della differenza tra l'ampiezza rilevata dal microfono destro e quella rilevata dal microfono sinistro. Ciò è stato un problema in quanto l'errore deve necessariamente influenzare la velocità con la quale il servomotore ruota, perché se l'errore è molto grande significa che la sorgente sonora si è spostata dal lato destro al lato sinistro del robot, o viceversa, e dunque la velocità di rotazione del servomotore deve essere alta altrimenti la sorgente si sposta da un lato all'altro del robot ma il robot impiega molto tempo per portarsi

alla posizione desiderata. Ovviamente, se l'errore è molto piccolo significa che la sorgente sonora si è spostata di poco e comunque si è mantenuta sempre nelle immediate vicinanze della faccia centrale del robot, dunque la velocità di rotazione del servomotore deve essere bassa, altrimenti si corre il rischio di ricadere in una situazione in cui la sorgente si sposta di poco mentre il robot ruota di molto andando, quindi, fuori posizione desiderata. È chiaro che per assicurare una certa dipendenza tra la velocità di rotazione del servomotore e l'errore (differenza tra le ampiezze rilevate dai microfoni: destro e sinistro) occorre utilizzare un regolatore che vada a stabilire la velocità di rotazione in funzione dell'errore. Il metodo che ho deciso di utilizzare è il seguente: in base all'entità dell'errore utilizzo un delay (ritardo) diverso. In tal modo, se l'errore è molto grande verrà impostato un delay molto piccolo in modo che la rotazione venga effettuata molto velocemente, se, invece, l'errore è piccolo verrà impostato un delay grande in modo che la velocità di rotazione del servomotore sia bassa. Allo scopo di dimostrare che l'aggiunta del regolatore ha portato a dei vantaggi ho deciso di utilizzare Matlab in modo da rappresentare su due grafici differenti l'andamento della curva dell'angolo in assenza del regolatore e l'andamento della curva dell'angolo in presenza del regolatore come mostrato in figura 5.1 e 5.2

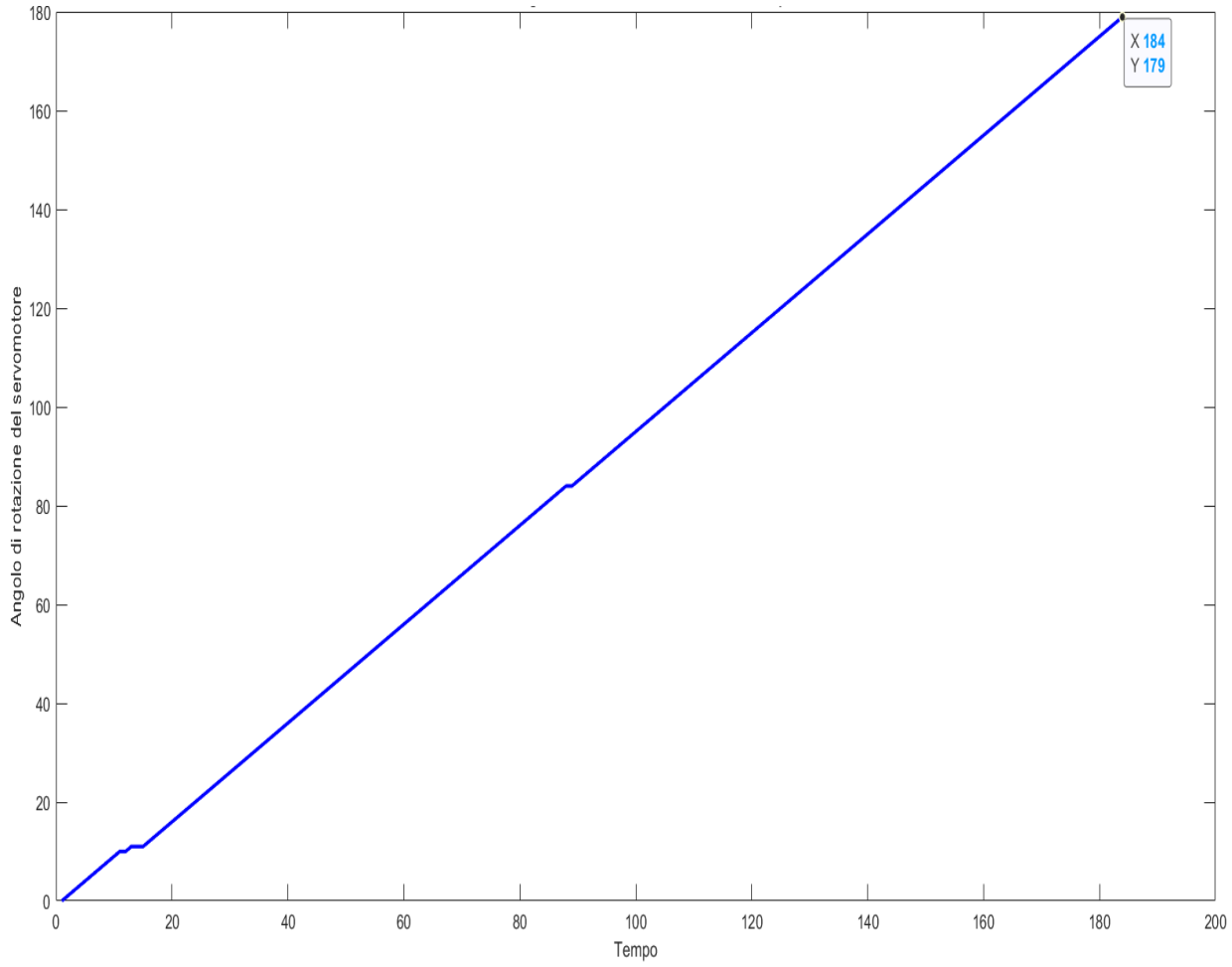


Figura 5.1: Angolo di rotazione del servomotore in assenza del regolatore.

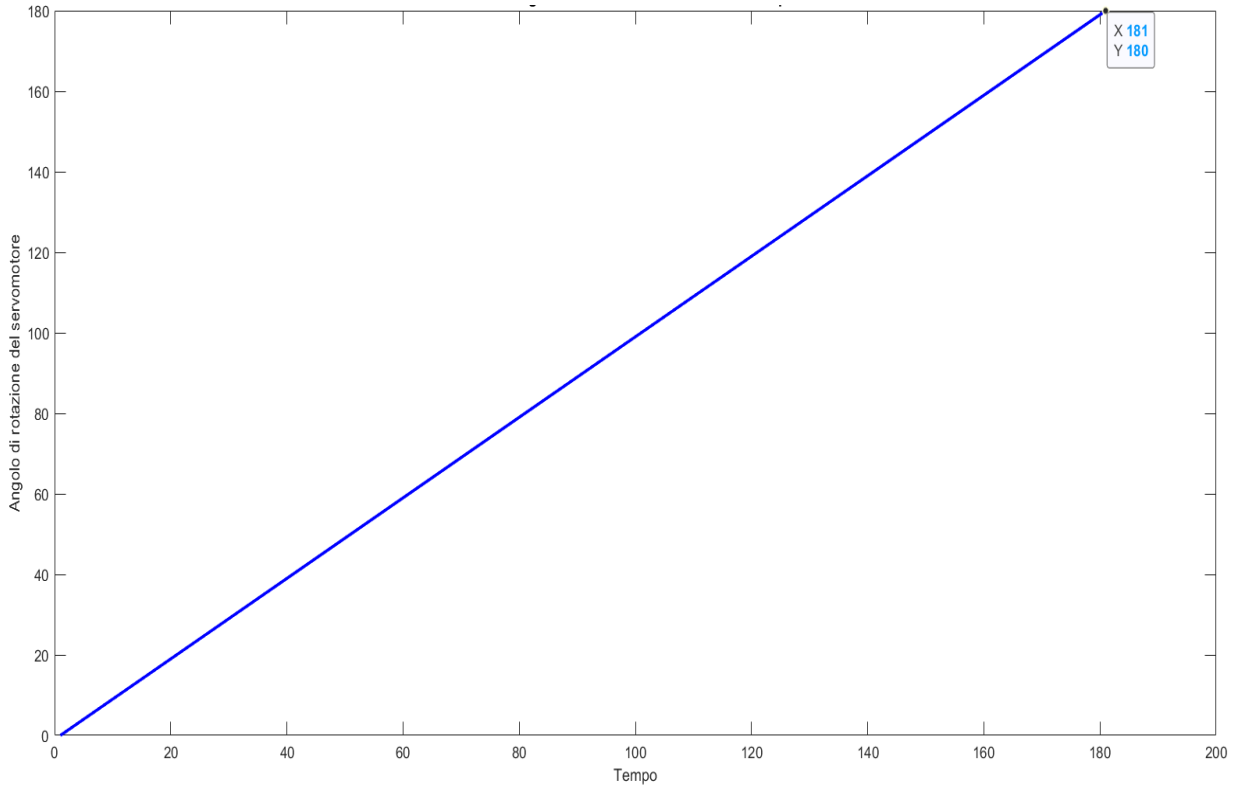


Figura 5.2: Angolo di rotazione del servomotore in presenza del regolatore.

Le due figure risultano essere abbastanza simili senonché ci sono delle differenze. Innanzitutto, l'aggiunta del regolatore ha consentito al robot di arrivare a regime prima (anche se non di molto) rispetto al caso senza regolatore. Inoltre, dal confronto dei due grafici si può evincere il fatto che la curva in assenza del regolatore è fondamentalmente crescente così come la curva in presenza del regolatore ma a differenza di quest'ultima, la curva in assenza del regolatore presenta dei tratti in cui la curva si appiattisce e diventa costante (anche se per poco). Questi sono dei tratti problematici in quanto diminuiscono la velocità con la quale il sistema raggiunge il valore di regime ma soprattutto provocano delle oscillazioni in quanto l'angolo per un certo lasso di tempo rimane costante per poi ricominciare a crescere bruscamente. Si nota come l'aggiunta del regolatore abbia migliorato la situazione. Ovviamente, entrambi i test sono stati condotti partendo da un angolo iniziale di 0 gradi per poi arrivare ad un angolo finale di 180 gradi allo scopo di verificarne l'andamento nel tempo.
Ultima cosa da mettere in evidenza è il grafico raffigurante l'ampiezza rilevata dai tre microfoni in funzione del tempo. Dalla figura 5.3 si nota una caratteristica importante di questi microfoni; come già accennato, questa tipologia di microfono è dotata di guadagno che viene regolato automaticamente in funzione dell'ampiezza percepita in modo da evitare che il microfono possa saturare. Infatti, sempre dalla figura 5.3 si nota come anche i picchi più elevati, ossia quelli in corrispondenza dei quali l'ampiezza risulta essere elevata, si mantengono distanti dal valore in corrispondenza del quale il microfono satura.

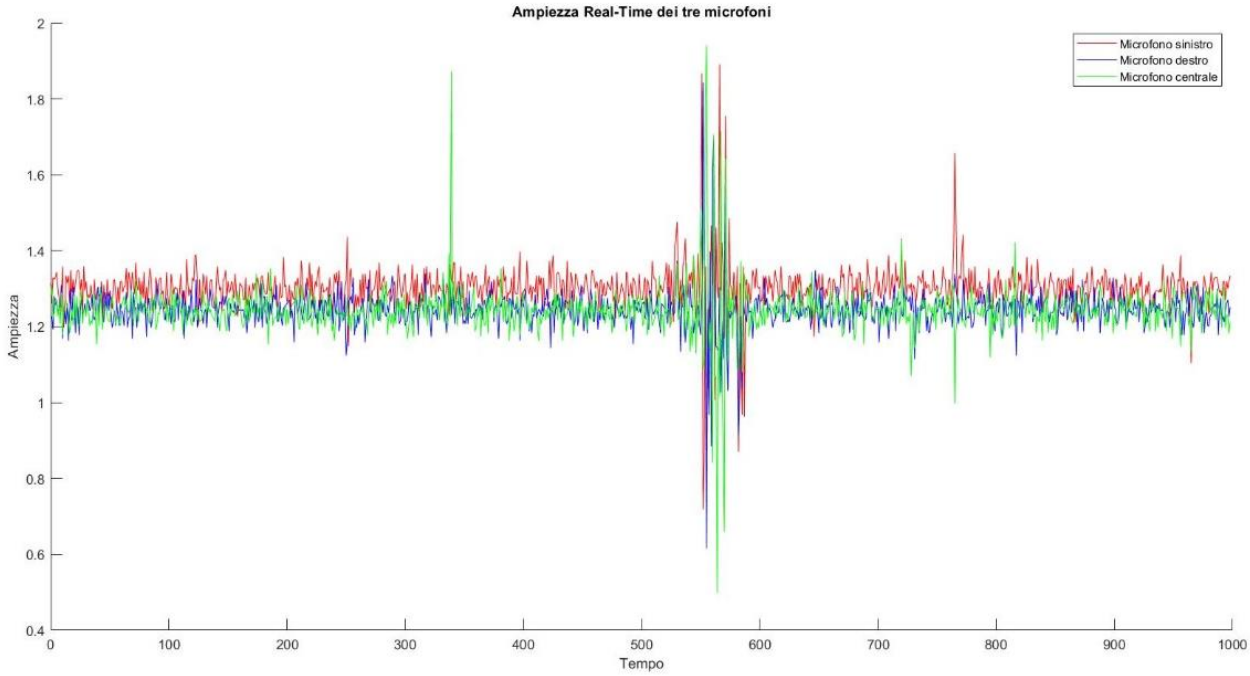


Figura 5.3: Ampiezza Real-Time dei tre microfoni.

Sempre osservando questa figura si può notare come tutti e tre gli andamenti siano correlati tra loro, questo perché, come detto, i tre microfoni sono dotati di un guadagno automatico ma soprattutto di un amplificatore interno che rende i microfoni molto sensibili anche se situati ad una distanza elevata gli uni dagli altri. Ecco il motivo per il quale ho dovuto installare i microfoni in modo che la distanza fosse di almeno 50 cm.

# 5.2 Discussione

La localizzazione di una sorgente sonora, elemento centrale in questa tesi, è stata studiata ed implementata in maniera considerevole nel corso degli ultimi anni, infatti sono state sviluppate moltissime tecniche di localizzazione tra cui possiamo nuovamente citare la TDOA (Time Difference of Arrival), ITD, IID, ecc. Diciamo che la maggior parte dei robot che implementano la possibilità di localizzare un suono si basano sulla TDOA anche se vi sono progetti che ne implementano altre. Documentandomi su tutto ciò che abbia a che fare con la localizzazione sonora ho scoperto che implementare la TDOA sarebbe stato molto oneroso per via della necessità di utilizzare componenti ad alto costo; quest'ultimo è stato un fattore che ha inciso molto sulla scelta di quale tecnica di localizzazione implementare in quanto il principale obiettivo di questa tesi, come detto, è di cercare di ottenere il massimo risultato con il minimo sforzo (economico) proprio come afferma il principio di Pareto. Di conseguenza, ho scelto di implementare una tecnica che si basa sul calcolo dell'errore che non è altro che una differenza tra due ampiezze, quella rilevata dal microfono sinistro e quella rilevata dal microfono destro, in funzione della quale il servomotore effettua una rotazione in verso orario o antiorario. Il ragionamento che ho seguito nell'implementare questa tecnica è stato il seguente: innanzitutto ho realizzato che se l'errore (differenza tra le ampiezze rilevate dal microfono destro e sinistro) è positivo o negativo, ossia, ad esempio, l'ampiezza rilevata dal microfono destro è maggiore di quella rilevata dal microfono sinistro è ovvio che il suono sta provenendo da destra e dunque il servomotore debba ruotare verso destra. Dunque, utilizzo l'errore per capire verso quale direzione il servomotore debba ruotare e l'ampiezza rilevata dal microfono centrale per determinare l'istante esatto in cui il servomotore deve arrestare la rotazione, in quanto è ovvio che non appena l'ampiezza rilevata dal microfono centrale diventa maggiore di quelle rilevate dai microfoni esterni la piattaforma è posizionata esattamente dinanzi alla sorgente sonora. Diciamo che il fattore mediante il quale è possibile confrontare le diverse tecniche di localizzazione è l'accuratezza con cui un robot riesce a localizzare una sorgente sonora. Facendo seguito ai diversi test sperimentali che sono stati portati a termine si può affermare che nonostante abbia utilizzato una tecnica diversa dalla TDOA i risultati sono stati più che soddisfacenti. I primi test sono stati effettuati senza l'utilizzo di alcun cono che, in qualche modo, potesse rendere i microfoni ancora più direzionali e già, in questa prima fase, i risultati sono stati positivi. Ho notato che il sistema funzionava molto bene nelle situazioni in cui la sorgente sonora si trovava molto a sinistra o molto a destra rispetto al centro del robot mentre iniziava a mostrare qualche difficoltà nelle situazioni in cui la sorgente sonora si trovava nelle vicinanze della faccia centrale del robot, ossia quella in cui è stato installato il sensore a ultrasuoni.

Le cose sono migliorate con l'aggiunta del cono che è stato installato nella faccia centrale del robot e che ha contribuito a rendere il microfono centrale più direzionale rispetto a quando non lo fosse in assenza del cono. In sintesi, nonostante abbia utilizzato una tecnica diversa da quella implementata nella maggior parte dei robot dotati di localizzazione sonora, i risultati sono stati soddisfacenti soprattutto in relazione al fatto che sono stati utilizzati componenti a basso costo.

Le domande che sono state poste nella parte iniziale della tesi erano due ed erano le seguenti:

- Quanto accuratamente può essere fatta la localizzazione di una sorgente sonora, calcolando l'angolo e la distanza da essa, utilizzando dei componenti a basso costo come ad esempio dei microfoni e Arduino Mega?
- Con quanta accuratezza possono essere effettuate delle misurazioni da parte di un sensore ad ultrasuoni, che consente la realizzazione di un Sonar?

Vediamo di rispondere ad entrambe le domande; per quanto riguarda la prima si può affermare che l'utilizzo di componenti a basso non ha inciso particolarmente sull'accuratezza con cui il sistema può localizzare una sorgente sonora, in quanto effettuando diversi test sono giunto a conclusione che l'accuratezza con cui vengono effettuate le misurazioni sono accettabili, infatti il sistema riesce a localizzare la sorgente sonora fino ad un massimo di 1.5/2 metri circa, distanza che può essere aumentata ulteriormente se si effettuano dei test all'interno di una stanza molto silenziosa e con l'assenza di ostacoli. Ovviamente, all'aumentare della distanza tra il robot e la sorgente sonora diminuisce l'accuratezza con la quale vengono effettuate le misurazioni. In particolare, se la sorgente sonora risulta essere ad una distanza maggiore di 2/3 metri l'errore ossia la differenza fra l'angolo di rotazione effettiva del servomotore e l'angolo desiderato inizia a crescere esponenzialmente. Diciamo che se ci si mantiene in un intorno di 1/2 metri l'accuratezza con la quale viene effettuata la localizzazione sonora è molto elevata e dunque l'angolo di rotazione effettivo del servomotore risulta essere vicinissimo se non addirittura uguale a quello desiderato.

Per quanto concerne il secondo quesito, si può affermare che le misurazioni effettuate dal sensore a ultrasuoni risultano essere molto precise soprattutto all'interno di ambienti poco rumorosi in cui si può sottolineare l'assenza di materiali che riflettono le onde sonore.

Ovviamente, facendo dei test all'interno di ambienti rumorosi o in cui si può registrare la presenza di materiali riflettenti che non fanno altro che deviare le onde sonore modificandone la velocità di propagazione si può notare come l'accuratezza diminuisca notevolmente.

# 5.3 Conclusioni e Progetti futuri

Il presente lavoro di tesi ha avuto per argomento lo sviluppo di un sistema per la localizzazione sonora. A partire dalla ispirazione biologica, ed in particolare al mondo degli insetti, ed a partire dai risultati raggiunti in precedenza nel nostro laboratorio, con il presente lavoro si è voluto rivisitare il problema utilizzando architetture semplici, più moderne ed a basso costo che potessero ospitare algoritmi in passato proibitivi dal punto di vista del carico computazionale.

In conclusione, si può affermare che gli obiettivi che sono stati prefissati nella fase iniziale della tesi sono stati raggiunti in quanto il sistema non solo è in grado di seguire una sorgente sonora ma è anche in grado di discriminare tra le varie sorgenti sonore presenti nell'ambiente circostante in quanto esso è in grado di elaborare i segnali in arrivo ai tre microfoni mediante il calcolo della FFT e di filtrarli in modo da seguire soltanto determinati suoni e trascurare altre tipologie di suoni come ad esempio il rumore.

I risultati raggiunti si prestano sia ad una immediata implementazione a bordo di robot mobili, opportunamente estesi a due gradi di libertà, che ad interessanti sviluppi futuri, infatti a valle del processamento effettuato, si possono utilizzare sistemi neurali e di apprendimento per estenderne le capacità.

Per quanto riguarda i progetti futuri, si può affermare che il sistema che è stato sviluppato può essere migliorato ulteriormente in quanto attualmente l'unica direzione verso la quale il robot può dirigersi è quella orizzontale. Come è noto, l'essere umano è in grado di localizzare una sorgente sonora e dopo averla localizzata, la testa di un essere umano può muoversi lungo diverse direzioni tra le quali troviamo quella orizzontale e quella verticale. Dunque, una feature aggiuntiva che sarebbe possibile includere in questo progetto consisterebbe nella capacità da parte del robot di ruotare non solo in orizzontale ma anche in verticale. Per far ciò occorrerebbe replicare due volte quanto già fatto per implementare la rotazione orizzontale, in quanto si dovrebbero adoperare molti più microfoni rispetto a quelli utilizzati in questo progetto, ma soprattutto si dovrebbe effettuare un upgrade dell'hardware in quanto il codice per il calcolo della FFT è molto oneroso dal punto di vista computazionale e, ovviamente, a causa del fatto che per ogni microfono utilizzato occorre eseguire una porzione di codice per il calcolo della FFT del segnale acquisito da quel particolare microfono, aumentare ulteriormente il numero di microfoni adoperati comporterebbe l'esecuzione di ulteriore codice per il calcolo della FFT, cosa che con l'hardware attuale sarebbe impossibile realizzare. Occorre sottolineare che già soltanto per eseguire il calcolo della FFT tre volte, in quanto i microfoni adoperati sono stati tre, è stato reso necessario l'impiego di due schede Arduino Mega. Si può facilmente immaginare cosa comporterebbe l'aggiunta di ulteriori microfoni per

implementare anche la rotazione verticale. Un'altra abilità molto interessante che sarebbe possibile includere all'interno del robot è l'abilità motoria in quanto, come detto anche nell'Abstract, sarebbe straordinario se il robot oltre a localizzare una sorgente sonora riuscisse anche a dirigersi fisicamente verso l'oggetto che sta producendo il suono. Queste sono esempi di features che potrebbero benissimo essere incluse all'interno del progetto, in futuro, in modo da migliorarlo e renderlo più completo e umano.

# Bibliografia

[0] P. Arena, L. Patanè, Editors, "*Spatial Temporal Patterns for Action-Oriented Perception in Roving Robots*", Cognitive Systems Monographs book series (COSMOS, volume 1), Springer, 2008.

[1] Gamble, E. & Rainton, D. (1994), Learning to Localize Sounds Using Vision, ATR Human Information Processing Laboratories, Kyoto Japan.

[2] Li, X., Shen, M., Wang, W., and Liu, H. (04 Jul 2012). Real–time sound source localization for a mobile robot based on the guided spectral-temporal position method. *International Journal of Advanced Robotic Systems*, pages 1–8.

[3] Hamada, N., Kasprzak, W., and Przybysz, P. (2012). Auditory scene analysis by time-delay analysis with three microphones. In *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on*, pages 1-8. IEEE.

[4] Huang, J., Supaongprapa, T., Terakura, I., Ohnishi, N., and Sugie, N. (1997). Mobile robot and sound localization. In *Intelligent Robots and Systems, 1997. IROS'97. Proceedings of the 1997 IEEE/RSJ International Conference on*, volume 2, pages 683–689. IEEE.

[5] Young, H. D. and Freedmen, R. A. (2014). *Sears and Zemansky's University Physics with modern physics technology update.* Pearson, 13th edition.

[6] Drezek, E. (2017). How do frequencies affect pitch?
Available from: http://eaninquiry1.weebly.com/how-do-frequencies-affect-pitch.html

[7] Audio-Technica (2017). What a microphone does.
Available from: http://www.audio-technica.com/cms/site/b0d226992d31e25d/

[8] Hampton, A. (2015). Sounds.
Available from: https: //comp1alexihampson.wordpress.com/2015/02/09/sounds/

[9] Johansson, H. (2013). *Elektroteknik.* Department of Machine Design KTH.

[10] Hirzel, T. (2017). Pwm.
Available from: https://www.arduino.cc/en/Tutorial/PWM

[11] Burgess, D. A. (1992), Techniques for Low Cost Spatial Audio, in 'Fifth Annual Symposium on User Interface Software and Technology', ACM, Monterey. (UIST '92).

[12] Blauert, J. (1983), Spatial Hearing, The MIT Press, Cambridge, MA.

[13] Mills, A. W. (1972), Auditory Localization, in J. V. Tobias, ed. 'Foundations of Modern Auditory Theory', Vol. II, Academic Press, New York, pp. 303-348.

[14] Bose, A. (1994), ‘Acoustics’, 6.312 lectures. MIT.

[15] C.H. Knapp, G.C. CARTER, “The Generalized Correlation Method for Estimation of Time Delay”, IEEE, August 1976.

[16] Arduino. [Arduino Mega specifications].
Available from: https://store.arduino.cc/mega-2560-r3

[17] Adafruit AGC Electret Microphone Amplifier – MAX9814 [Microphone specifications].
Available from: https://cdn-shop.adafruit.com/datasheets/MAX9814.pdf

[18] Electronics (2017). What is an ultrasonic sensor?
Available from: http://cmra.rec.ri.cmu.edu/content/electronics/boe/ultrasonic_sensor/1.html

[19] Infrared Receiver Module AX-1838HS (IR sensor specifications).
Available from: http://dalincom.ru/datasheet/AX-1838HS.pdf

[20] Arduino [Arduino forum – “Faster analog read”].
Available from: http://forum.arduino.cc/index.php?topic=6549.0

# Indice delle figure

# Appendice

## A Codice Processing

```
import processing.serial.*; // Importo la libreria per la comunicazione seriale
import java.awt.event.KeyEvent; // Importo la libreria per l'acquisizione dei dati dalla seriale
import java.io.IOException;
Serial myPort; // Definizione di un oggetto seriale
// Dichiarazione delle variabili
String angle="";
String distance="";
String data="";
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;


void setup() {
 size (1920, 1050); // Impostazione della risoluzione del Sonar
 smooth();
 myPort = new Serial(this,"COM4", 115200); // Inizia la comunicazione seriale
 myPort.bufferUntil('.'); // Legge i dati dalla seriale fino a quando non trova il '.'. In tal modo verrà effettuata la lettura di: angolo e distanza.
 orcFont = loadFont("OCRAExtended-48.vlw");
}


void draw() {

  fill(98,245,31);
  textFont(orcFont);
  noStroke();
  fill(0,4);
  rect(0, 0, width, height-height*0.065);

  fill(98,245,31); // colore verde
  // Invoco le funzioni per la creazione del radar
  drawSonar();
  drawLine();
  drawObject();
  drawText();
}

void serialEvent (Serial myPort) { // Inizia la lettura dei dati dalla porta seriale
  // Leggi i dati dalla seriale fino a quando non trovi il carattere '.' e inseriscili nella variabile data
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);



  index1 = data.indexOf(","); // Trova il carattere ',' e memorizzalo nella variabile "index1"
  angle= data.substring(0, index1); // Leggi i dati, dalla posizione "0" alla posizione contenuta nella variabile index1, che rappresentano i valori degli angoli inviati da Arduino tramite la seriale
  distance= data.substring(index1+1, data.length()); // Leggi i dati, dalla posizione "index1" fino alla fine dei dati, che rappresentano i valori delle distanze inviate da Arduino tramite la seriale

  // Converti le stringhe in valori interi
  iAngle = int(angle);
  iDistance = int(distance);
}

void drawSonar() {
  pushMatrix();
  translate(width/2,height-height*0.074);
  noFill();
  strokeWeight(2);
  stroke(98,245,31);
  arc(0,0,(width-width*0.0625),(width-width*0.0625),PI,TWO_PI);
  arc(0,0,(width-width*0.27),(width-width*0.27),PI,TWO_PI);
  arc(0,0,(width-width*0.479),(width-width*0.479),PI,TWO_PI);
  arc(0,0,(width-width*0.687),(width-width*0.687),PI,TWO_PI);
  line(-width/2,0,width/2,0);
  line(0,0,(-width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
  line(0,0,(-width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
  line(0,0,(-width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
  line(0,0,(-width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
  line(0,0,(-width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));
  line((-width/2)*cos(radians(30)),0,width/2,0);
  popMatrix();
}
```

```
void drawObject() {
  pushMatrix();
  translate(width/2,height-height*0.074);
  strokeWeight(9);
  stroke(255,10,10); //  colore rosso
  pixsDistance = iDistance*((height-height*0.1666)*0.025);
  if(iDistance<40){
  line(pixsDistance*cos(radians(iAngle)),-pixsDistance*sin(radians(iAngle)),(width-width*0.505)*cos(radians(iAngle)),-(width-width*0.505)*sin(radians(iAngle)));
  }
  popMatrix();
}
void drawLine() {
  pushMatrix();
  strokeWeight(9);
  stroke(30,250,60);
  translate(width/2,height-height*0.074);
  line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-height*0.12)*sin(radians(iAngle)));
  popMatrix();
}
void drawText() {

  pushMatrix();
  if(iDistance>40) {
  noObject = "Fuori Portata";
  }
  else {
  noObject = "Rilevabile";
  }
  fill(0,0,0);
  noStroke();
  rect(0, height-height*0.0648, width, height);
  fill(98,245,31);
  textSize(25);

  text("10cm",width-width*0.3854,height-height*0.0833);
  text("20cm",width-width*0.281,height-height*0.0833);
  text("30cm",width-width*0.177,height-height*0.0833);
  text("40cm",width-width*0.0729,height-height*0.0833);
  textSize(40);
  text("Oggetto: " + noObject, width-width*0.875, height-height*0.0277);
  text("Angolo: " + iAngle +" °", width-width*0.48, height-height*0.0277);
  text("Distanza: ", width-width*0.26, height-height*0.0277);
  if(iDistance<100) {
  text("        " + iDistance +" cm", width-width*0.225, height-height*0.0277);
  }

   textSize(25);
   fill(98,245,60);
   translate((width-width*0.4994)+width/2*cos(radians(30)),(height-height*0.0907)-width/2*sin(radians(30)));
   rotate(-radians(-60));
   text("30°",0,0);
   resetMatrix();
   translate((width-width*0.503)+width/2*cos(radians(60)),(height-height*0.0888)-width/2*sin(radians(60)));
   rotate(-radians(-30));
   text("60°",0,0);
   resetMatrix();
   translate((width-width*0.507)+width/2*cos(radians(90)),(height-height*0.0833)-width/2*sin(radians(90)));
   rotate(radians(0));
   text("90°",0,0);
   resetMatrix();
   translate(width-width*0.513+width/2*cos(radians(120)),(height-height*0.07129)-width/2*sin(radians(120)));
   rotate(radians(-30));
   text("120°",0,0);
   resetMatrix();
   translate((width-width*0.5104)+width/2*cos(radians(150)),(height-height*0.0574)-width/2*sin(radians(150)));
   rotate(radians(-60));
   text("150°",0,0);
   popMatrix();
}
```

# B Codice Arduino Master

```
//MASTER
#include <Servo.h>
#include "IRremote.h"
#include <Wire.h>


char v0[5] =  {};
int d;
int i;
int olddelta = 0;
int newdelta;
Servo myservo;
const int trigPin = 10;
const int echoPin = 11;
// Variables for the duration and the distance
long duration;
int distance;
int receiver = 5; // Signal Pin of IR receiver to Arduino Digital Pin 5
char scelta;
char ritardo[2] = {};
int ritardo_int;

/*-----( Declare objects )-----*/
IRrecv irrecv(receiver);     // create instance of 'irrecv'
decode_results results;      // create instance of 'decode_results'

#define FASTADC 1
// defines for setting and clearing register bits
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif


void setup() {
    Wire.begin();
    Serial.begin(115200);
    #if FASTADC
    // set prescale to 8
    cbi(ADCSRA, ADPS2);
    sbi(ADCSRA, ADPS1);
    sbi(ADCSRA, ADPS0);
    #endif
```

```
    pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
    pinMode(echoPin, INPUT); // Sets the echoPin as an Input
    irrecv.enableIRIn(); // Start the receiver
}

/*-----( Function )-----*/
char translateIR() // takes action based on IR code received
{
  switch(results.value)
  {
  case 0xFF6897: return '0'; break;
  case 0xFF30CF: return '1'; break;
  }// End Case
  delay(500);
}

void loop() {
  if (scelta != '0' && scelta != '1'){
if (irrecv.decode(&results)) // have we received an IR signal?
  {
    scelta = translateIR();
    irrecv.resume(); // receive the next value
  }
  }
 while(scelta == '0'){ //modalità localizzazione sonora
Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
while (v0[2] == 'd'){
  myservo.detach();
  if (irrecv.decode(&results)) // have we received an IR signal?
  {
    scelta = translateIR();
    irrecv.resume(); // receive the next value
    if (scelta == '1') {

      scelta = '1';
      break;
    }
    else {
      scelta = '0';
    }
  }
```

```
  Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
}
if (scelta == '1') break;
myservo.attach(9);
//primo if
if (v0[0] == 's' && v0[2] == 'c'){ //if (v1 < v0)
Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
ritardo[0]= v0[3];
ritardo[1] = v0[4];
ritardo[2] = '\0';
ritardo_int = atoi(ritardo);
for (i = olddelta; i < 180; i+=1){
  //Serial.println(ritardo_int);
delay(ritardo_int);
Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
ritardo[0]= v0[3];
ritardo[1] = v0[4];
ritardo[2] = '\0';
ritardo_int = atoi(ritardo);
while (v0[2] == 'd'){
  myservo.detach();
  Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
```

```
}
     //delay(40);
     myservo.write(i);
     Serial.println(i);
     if( v0[1] == 'a' && v0[2] == 'c'){
       while(v0[1] == 'a' && v0[2] == 'c'){
         //nuovo codice
         while(v0[1] == 'a' && v0[2] == 'c'){
         myservo.detach();
           Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
         }
Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}

}
   // Serial.println(i);
    //myservo.write(i);
    if( v0[1] == 'b' && v0[2] == 'c' ){
      break;
    }
        } //fine while
        break;
    }
    if (v0[0] == 'n' && v0[2] == 'c')  break;
   }//fine for
   olddelta = i;
   } //fine if
   else { //v1 > v0
    Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
ritardo[0]= v0[3];
ritardo[1] = v0[4];
```

```
ritardo[2] = '\0';
ritardo_int = atoi(ritardo);
//Serial.println(ritardo_int);
    for (i = olddelta; i > 0; i-=1){
      //Serial.println(ritardo_int);
    delay(ritardo_int);
Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
ritardo[0]= v0[3];
ritardo[1] = v0[4];
ritardo[2] = '\0';
ritardo_int = atoi(ritardo);
while (v0[2] == 'd'){
  myservo.detach();
  Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
}
    //delay(40);
    myservo.write(i);
    Serial.println(i);
    if( v0[1] == 'a' && v0[2] == 'c'){
      while(v0[1] == 'a' && v0[2] == 'c'){
        //nuovo codice
        while(v0[1] == 'a' && v0[2] == 'c'){
        myservo.detach();
          Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
  //Serial.println(v0[d]);
  d++;
  if (d>=5){d=0;}
}
        }
Wire.requestFrom(3,5);
while(Wire.available()){
  v0[d] = Wire.read();
```

```
  d++;
  if (d>=5){d=0;}
}
    //Serial.println(i);
    //myservo.write(i);
    if( v0[1] == 'b' && v0[2] == 'c' ){
      break;
    }
        } //fine while
      break;
    }
    if (v0[0] == 's' && v0[2] == 'c') break;
    }//fine for
   olddelta = i;
   }//fine else
    //delay(1);  //Repeat the process every second OR:
    //while(1);       //Run code once
 } //fine scelta 0
if (scelta == '1'){ //modalità sonar
  myservo.attach(9);
while (scelta == '1'){
   if (irrecv.decode(&results)) // have we received an IR signal?
  {

    scelta = translateIR();
    irrecv.resume(); // receive the next value
    if (scelta == '0') {
      scelta = '0';
      break;
    }
  }
// rotates the servo motor from 0 to 180 degrees
  for(int i=0;i<=180;i++){

  if (irrecv.decode(&results)) // have we received an IR signal?
  {
    scelta = translateIR();
    irrecv.resume(); // receive the next value
    if (scelta == '0') {
      scelta = '0';
      break;
    }
  }
  delay(5);
  myservo.write(i);
  distance = calculateDistance();
  Serial.print(i);
```

```
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
if (scelta == '0') {
  scelta = '0';
  break;
  }
  // Repeats the previous lines from 180 to 0 degrees
  for(int i=180;i>0;i--){
     if (irrecv.decode(&results)) // have we received an IR signal?
  {
    scelta = translateIR();
    irrecv.resume(); // receive the next value
    if (scelta == '0') {
      scelta = '0';
      break;
    }
  }
  delay(5);
  myservo.write(i);
  distance = calculateDistance();
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
}//fine scelta 1 while
}//fine scelta 1
}
// Function for calculating the distance measured by the Ultrasonic sensor
int calculateDistance(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance= duration*0.034/2;
  return distance;
}
```

# C Codice Arduino Slave

```
//SLAVE
#include <Wire.h> //Protocollo I2C
#include <arduinoFFT.h>
#define SAMPLES 128             //Must be a power of 2
#define SAMPLING_FREQUENCY 2500 //Hz, must be less than 10000 due to ADC

arduinoFFT FFT0 = arduinoFFT();
arduinoFFT FFT1 = arduinoFFT();
arduinoFFT FFT2 = arduinoFFT();

double x0; //frequenza fondamentale
double v0; //ampiezza alla frequenza fondamentale
double x1; //frequenza fondamentale
double v1; //ampiezza alla frequenza fondamentale
double x2; //frequenza fondamentale
double v2; //ampiezza alla frequenza fondamentale
unsigned int sampling_period_us;
unsigned long microseconds;
double errore;
int ritardo;
char ritardo_string[2];
char t[4] = {};
char temp[] = {};

double vReal0[SAMPLES];
double vImag0[SAMPLES];
double vReal1[SAMPLES];
double vImag1[SAMPLES];
double vReal2[SAMPLES];
double vImag2[SAMPLES];
void setup() {
    sampling_period_us = round(1000000*(1.0/SAMPLING_FREQUENCY));
    Serial.begin(115200);
    Wire.begin(3); //L'indirizzo assegnato all'Arduino slave è 3
    Wire.onRequest(InviaAmpiezze);
}

void loop(){
    for(int i=0; i<SAMPLES; i++)
    {
        microseconds = micros();
        vReal0[i] = analogRead(0);;
        vImag0[i] = 0;
        vReal1[i] = analogRead(1);
        vImag1[i] = 0;
        vReal2[i] = analogRead(2);
        vImag2[i] = 0;
```

```
        while(micros() < (microseconds + sampling_period_us)){
        }
    }

    /*FFT*/
    FFT0.Windowing(vReal0, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
    FFT0.Compute(vReal0, vImag0, SAMPLES, FFT_FORWARD);
    FFT0.ComplexToMagnitude(vReal0, vImag0, SAMPLES);
    FFT0.DCremoval(vReal0);
    FFT0.MajorPeak(vReal0, SAMPLES, SAMPLING_FREQUENCY, &x0, &v0); //frequenza fondamentale
    FFT0.BandPassFilter(vReal0,SAMPLES, SAMPLING_FREQUENCY,970, 1100,x0,&v0);

    FFT1.Windowing(vReal1, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
    FFT1.Compute(vReal1, vImag1, SAMPLES, FFT_FORWARD);
    FFT1.ComplexToMagnitude(vReal1, vImag1, SAMPLES);
    FFT1.DCremoval(vReal1);

    FFT1.MajorPeak(vReal1, SAMPLES, SAMPLING_FREQUENCY, &x1, &v1); //frequenza fondamentale
    FFT1.BandPassFilter(vReal1,SAMPLES, SAMPLING_FREQUENCY,970, 1100,x1,&v1);

    FFT2.Windowing(vReal2, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
    FFT2.Compute(vReal2, vImag2, SAMPLES, FFT_FORWARD);
    FFT2.ComplexToMagnitude(vReal2, vImag2, SAMPLES);
    FFT2.DCremoval(vReal2);

    FFT2.MajorPeak(vReal2, SAMPLES, SAMPLING_FREQUENCY, &x2, &v2); //frequenza fondamentale
    FFT2.BandPassFilter(vReal2,SAMPLES, SAMPLING_FREQUENCY,970, 1100,x2,&v2);

    errore = abs(v1-v0);
    errore = constrain(errore,1000,5000);
    //Serial.print(v0);
    //Serial.print(" ");
    //Serial.println(v1);
    //Serial.println(errore);
}

void InviaAmpiezze(){
if (v0 == 0 && v1 == 0 && v2 == 0){
t[2] = 'd';
}
else{
  t[2] = 'c';
}
if (v1 < v0){
  t[0] = 's';
}
```

```
else {
  t[0]= 'n';
}
if (v2 > v1 && v2 > v0){
  t[1]= 'a';
}
else{
  t[1] = 'b';
}

if (errore > 3000) {
t[3] = '5';
t[4] = ' ';
}
else{
t[3] = '2';
t[4] = '0';
}
//Serial.println(t[3]);
strncpy(temp,t,5);
Wire.write(temp);
}
```

# D Codice Matlab:
# Ampiezza Real-Time dei tre microfoni

```
clear all
a = arduino('COM3');
line1=  line(nan,nan, 'color', 'red');
line2 = line(nan,nan,'color','blue');
line3 = line(nan,nan,'color','green');
legend('Microfono sinistro','Microfono destro','Microfono centrale')
xlabel('Tempo');
ylabel('Ampiezza');
title('Ampiezza Real-Time dei tre microfoni');
i=0;
while 1
    mic1 = readVoltage(a,'A0');
    mic2 = readVoltage(a,'A1');
    mic3 = readVoltage(a,'A2');
    x1 = get(line1,'xData');
    y1 = get(line1, 'yData');
    x2 = get(line2,'xData');
    y2 = get(line2, 'yData');
    x3 = get(line3,'xData');
    y3 = get(line3, 'yData');
    x1= [x1 i];
    y1 =[y1 mic1];
    x2= [x2 i];
    y2 =[y2 mic2];
    x3= [x3 i];
    y3 =[y3 mic3];
    set(line1,'xData',x1,'yData',y1)
    set(line2,'xData',x2,'yData',y2)
    set(line3,'xData',x3,'yData',y3)
    i=i+1;
end
```

# E Codice Matlab:
# Angolo di rotazione del servomotore in assenza/presenza del regolatore

```
clear all
a = serial('COM4',"BaudRate",115200);
fopen(a);
index = 1;
while 1
     temp = fscanf(a);
     angolo(index) = str2num(temp);
     plot(angolo,'LineWidth',2,'Color',[0,0,1.0])
     ylim([0,180]);
     xlabel('Tempo');
     ylabel('Angolo di rotazione del servomotore');
     title('Grafico Real-Time angolo di rotazione del servomotore');
     drawnow
     if angolo(index) == 180
         break;
     end
     index= index+1;
end
fclose(a);
```

# F Schema dei componenti