UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea triennale in Ingegneria Biomedica

Caratterizzazione di un sensore low-cost per l'acquisizione di segnali audio finalizzati alla misura della frequenza respiratoria

Characterization of a low-cost sensor aimed at audio signals acquisition for respiration rate measurement

Tesi di laurea di:

Luca Sassaroli

Relatore: Chiar.ma

Prof.ssa **Susanna Spinsante**

Correlatore:

Prof.ssa **Stefania Cecchi**

Ing. **Alessandro Terenzi**

Anno Accademico 2019-2020

Indice

Introduzione

La tesi illustra la progettazione e la caratterizzazione di un sistema low-cost per l'acquisizione di segnali audio finalizzati alla misura della frequenza respiratoria (RR). La frequenza respiratoria corrisponde ad uno dei parametri vitali maggiormente monitorati. Il nostro obiettivo è la ricerca di una modalità di acquisizione del parametro che non si dimostri invasiva per il corpo umano. Lo strumento di misura proposto si basa sull'utilizzo di una board Arduino UNO collegata ad un microfono. Per mezzo degli ambienti di programmazione Arduino IDE e MATLAB, abbiamo implementato alcuni codici in grado di acquisire segnali audio di tipo *.WAV*, di elaborarli e di analizzarli. Uno dei due parametri relativi a tali registrazioni audio che abbiamo trattato è la SPL (Sound Pressure Level o Livello di Pressione Sonora), la quale corrisponde alla misura logaritmica del livello di pressione efficace di un'onda meccanica rispetto ad un valore di riferimento dipendente dal mezzo di propagazione. Per effettuare i test necessari abbiamo usato un pistonofono, uno strumento usato per la calibrazione di strumenti acustici, in grado di produrre onde sonore con livelli di pressione prestabiliti. Dopo aver acquisito questi dati ed aver testato il corretto funzionamento sia del microfono che del nostro programma di acquisizione, siamo passati all' analisi del parametro della frequenza respiratoria, obiettivo primario del progetto. Sono stati analizzati i segnali relativi a tre individui, ai quali è stato chiesto di respirare mentre compievano tre attività fisiche di intensità crescente per un intervallo temporale pari a cinque minuti per ogni prova. Le registrazioni acquisite sono state elaborate ed analizzate attraverso un programma, implementato in MATLAB, in grado di rilevare i massimi locali di un segnale fornito in input in base a delle modalità di ricerca da scegliere in fase di programmazione. I massimi locali individuati nelle registrazioni corrispondono agli atti respiratori compiuti dagli individui. Nel capitolo 1 si presenta lo stato dell'arte, dove vengono riportati dei modelli, presenti in questo momento in letteratura, per la rivelazione della RR. Nel capitolo 2 e nel capitolo 3 si espongono rispettivamente la teoria del nostro studio e la progettazione hardware. Nel capitolo 4 viene presentata la fase di acquisizione dei segnali. Nel capitolo 5 e nelle conclusioni del documento vengono analizzati i risultati ottenuti con grafici e tabelle relativi ai test eseguiti; inoltre viene riportato il confronto con ulteriori strumenti di misura in grado di raggiungere lo stesso obiettivo del nostro, vengono paragonati i margini di errore ottenuti in modo da dare una valutazione corretta dell'accuratezza dello strumento e vengono proposte alcune soluzioni per il miglioramento e l'aggiornamento del progetto. Viene aggiunta una breve analisi legata al fattore economico relativo agli strumenti usati.

Capitolo 1

Stato dell'arte

L'andamento nel tempo della frequenza respiratoria rappresenta un indicatore di considerevole interesse nell'ambito medico, sia nella prevenzione, sia nella diagnostica. Tuttavia, la respirazione è il segno vitale che viene monitorato meno frequentemente perché ritenuto meno critico e più difficilmente misurabile. In realtà esso è un importante fattore predittivo di eventi gravi come le malattie polmonari, l'arresto cardiaco e l'ammissione in reparti di terapia intensiva. I pattern respiratori anormali possono essere caratteristici di malattie del sistema respiratorio, ma anche di condizioni patologiche metaboliche o del sistema nervoso centrale o della muscolatura. Per questi motivi ci si sta indirizzando verso il monitoraggio sempre più massivo e continuo dell'attività respiratoria. Proprio per riuscire a prevenire e diagnosticare alcuni disordini che comprendono diversi apparati dell'organismo, sono state messe a disposizione numerose metodiche con differenti tipologie di strumentazione. La cura del soggetto è sempre più orientata a porre la persona al centro dei trattamenti, pertanto vengono realizzati dispositivi che coniughino una rapida misurazione con il comfort dell'individuo. In questo capitolo vengono presentati alcuni progetti pubblicati nella banca dati *IEEE Xplore* che concettualmente e strutturalmente si avvicinano all'argomento della tesi.

1.1 Misura della frequenza respiratoria con l'uso dello smartphone

In questo ambito si sono studiati due progetti pubblicati sulla banca dati *IEEE Xplore*. La differenza tra i due progetti sta nella modalità di acquisizione dei segnali audio: il primo utilizza il microfono a bordo dell'auricolare di norma fornito in dotazione con qualunque smartphone in commercio come presentato in [1], invece il secondo utilizza sia il microfono a bordo dell'auricolare sia quello dello smartphone stesso, come esposto nel documento [2].

1.1.1 Uso del microfono esterno

Come evidenziato in [1], lo studio propone uno strumento di basso costo, portabile, poco intrusivo e di facile utilizzo per stimare la frequenza respiratoria di un soggetto. Il suono

generato dal flusso di aria proveniente dal naso viene raccolto dal microfono esterno dato in dotazione insieme ad uno smartphone. È stata sviluppata un'applicazione Android che registra e memorizza i segnali audio dovuti alla respirazione con una frequenza di campionamento pari a 44100 Hz. In seguito, questi segnali vengono ricampionati ad una frequenza di 3.33 Hz e fatti passare attraverso un filtro di tipo passa-basso con frequenza di taglio pari a 0.9 Hz, in modo da eliminare il rumore di sottofondo che risulta essere una componente di disturbo nelle acquisizioni. Dopo questa prima fase di elaborazione si passa all'algoritmo che riesce a identificare i massimi locali all'interno delle acquisizioni audio: tale processo consiste nella stima del valore di una soglia del respiro che permette di distinguere l'atto di inalazione dall'atto di esalazione. La transizione del segnale dalla parte inferiore a quella superiore, delimitate dalla soglia, corrisponde ad un atto respiratorio. La stima della soglia dipende da parametri che fanno riferimento al numero di punti esaminati del segnale e ad un indicatore statistico a scelta, da applicare a tali punti. La stima della frequenza respiratoria ottenuta attraverso questo algoritmo viene confrontata con un valore che si assume essere il *valore vero* della frequenza respiratoria. Questo valore si ottiene attraverso l'applicazione citata sopra, usata nel modo seguente: agli individui esaminati viene chiesto di premere una parte dello schermo dopo aver compiuto l'atto di inalazione e un'altra parte dopo l'atto di esalazione. Lo smartphone memorizza la distanza temporale tra questi due eventi e calcola automaticamente la frequenza respiratoria.

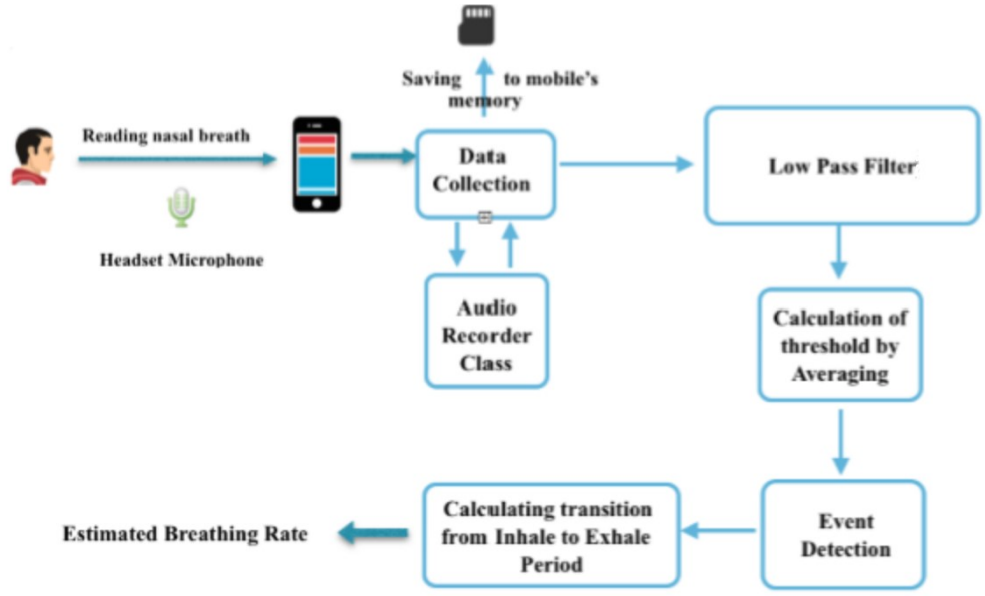

Figura 1.1. Diagramma a blocchi del procedimento descritto in [1].

I dati sono stati acquisiti da venticinque individui, di cui 11 maschi e 14 femmine, con un range di età compreso tra 10 e 50 anni. Le acquisizioni possono essere compiute mentre

l'individuo è seduto o disteso, e il microfono viene fissato sotto il naso. I tipi di esperimento compiuti sono basati sulla durata dell'acquisizione (1 minuto o 1 ora), sulla frequenza di campionamento scelta, sul consumo della batteria da parte dell'applicazione e sui diversi tipi di rumore che possono affliggere la procedura. In particolare, l'acquisizione di un minuto serve per valutare l'accuratezza del sistema: la frequenza respiratoria ottenuta attraverso l'algoritmo viene confrontata con il valore "vero", ottenuto dal numero di tocchi dello schermo da parte dell'individuo. I risultati evidenziano un'incertezza fra i due valori pari a circa il 6%-8%, che corrisponde ad una discrepanza di circa 1-3 atti respiratori per minuto.

1.1.2 Uso del microfono installato e di quello esterno

Lo studio presentato in [2] propone un metodo di rilevazione della frequenza respiratoria attraverso l'uso del microfono esterno e del microfono installato su uno smartphone. L'uso dei due microfoni corrisponde all'acquisizione di due tipologie di segnali audio: attraverso il microfono esterno, posizionato sotto il naso, vengono acquisiti i segnali audio prodotti dalla respirazione nasale; attraverso il microfono installato nello smartphone e posizionato sul collo, vengono acquisiti i segnali audio della trachea generati dalla respirazione. I dati vengono acquisiti attraverso un iPhone 4S e il relativo microfono aggiunto esternamente. Sono stati esaminati dieci individui non fumatori, di età compresa tra 20 e 40 anni, ai quali viene chiesto di respirare ad una determinata frequenza: un segnale acustico indica il momento in cui l'individuo deve iniziare l'atto di respirazione (inalazione seguita da esalazione). Il range delle frequenze esaminate varia da 0.1 Hz a 1.5 Hz e ad ogni test corrisponde un incremento pari a 0.1 Hz. Queste frequenze respiratorie corrispondono rispettivamente a 6 e 90 atti respiratori al minuto, con un incremento di 6 atti respiratori al minuto. Per ogni individuo si registrano tre minuti di acquisizione ad ogni frequenza. I segnali vengono memorizzati all'interno dello smartphone con una frequenza pari a 44100 Hz e alla risoluzione di 16 bits per campione. Il processo di elaborazione iniziale, illustrato in Figura 1.2, permette di ottenere dei segnali campionati alla frequenza variabile in un intorno di 100 Hz. Si applica un'ulteriore elaborazione attraverso MATLAB: viene applicato il processo di interpolazione *Cubic Spline* con il quale la frequenza viene fissata al valore costante di 100 Hz, successivamente i segnali vengono filtrati attraverso un filtro passa-banda che ammette frequenze comprese tra 0.19 e 4.6 Hz. La parte finale dell'elaborazione consiste nel nuovo campionamento dei segnali alla frequenza pari a 10 Hz e nell'eliminazione dei dieci secondi iniziali e finali di tutte le registrazioni.

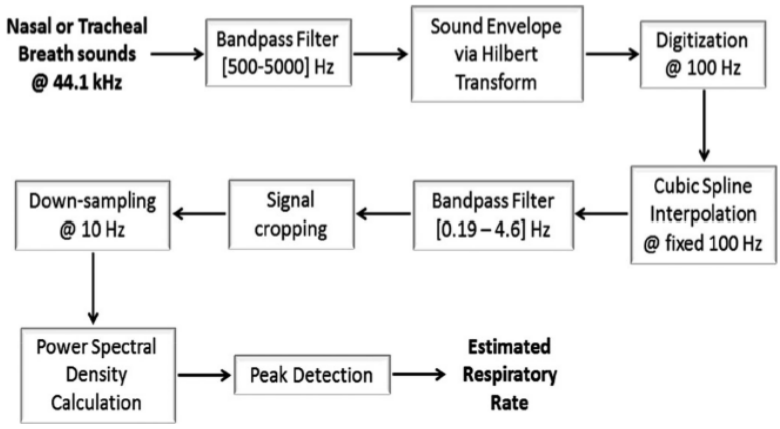

Figura 1.2. Diagramma a blocchi del procedimento secondo [2].

I segnali ottenuti vengono analizzati attraverso due algoritmi diversi: il metodo di Welch (spiegato in [3]) e la stima della densità spettrale attraverso il modello auto-regressivo di Burg (presentato in [4]). I valori della frequenza respiratoria ottenuti con questi metodi vengono confrontati con i valori ottenuti attraverso la tecnica di pletismografia ad induttanza, che viene usata in ambito clinico e risulta essere una delle tecniche più accurate a livello di misurazione della frequenza respiratoria. Da questo confronto si definisce l'errore nella stima della Respiration Rate (RR), attraverso la (1.1):

$$\varepsilon = \frac{mean(RR - RR_{est})^2}{mean(RR)^2} \cdot 100. \tag{1.1}$$

RR e RR_{est} corrispondono rispettivamente alla frequenza presa come riferimento e a quella stimata attraverso i due algoritmi precedenti. I risultati dei test effettuati evidenziano che l'errore relativo al procedimento è circa di 1%.

1.2 La frequenza respiratoria durante la fase del sonno

Nell'articolo [5] viene sviluppato un modulo a minimo contatto in grado di registrare i suoni prodotti dalla respirazione durante la fase di sonno di un individuo. Durante tale fase i muscoli dilatatori faringei diminuiscono la loro attività, così che il flusso di aria risulta essere turbolento e i suoni generati da esso aumentano di intensità. Vengono esaminati 204 individui usando un microfono RØDE NTG1 collegato ad un dispositivo di registrazione, che campiona i segnali alla frequenza di 16 kHz e risoluzione di 16 bits, memorizzandoli al suo interno. Come mostrato in Figura 1.3, la prima fase di elaborazione di questi segnali consiste nella rimozione del rumore di sottofondo con un algoritmo che incrementa il rapporto tra segnale e

rumore di 6 dB. Successivamente il segnale viene sottoposto ad un filtro temporale di 400 ms per rimuovere i rumori di transizione. Tale fase viene seguita da un processo di segmentazione temporale (12 s o 20 s) di ogni registrazione. Questi segmenti vengono sottoposti ad un ulteriore modulo: si calcola l'intervallo che divide un atto respiratorio dal successivo (intervallo respiratorio, BI), attraverso un algoritmo basato sulla formula dell'autocorrelazione di un segnale. Da tale stima il calcolo della frequenza respiratoria risulta immediato:

$$RR = \frac{60}{BI}. \tag{1.2}$$

La stima della RR attraverso questo metodo viene confrontata con la stima fatta con il metodo della pletismografia ad induttanza.

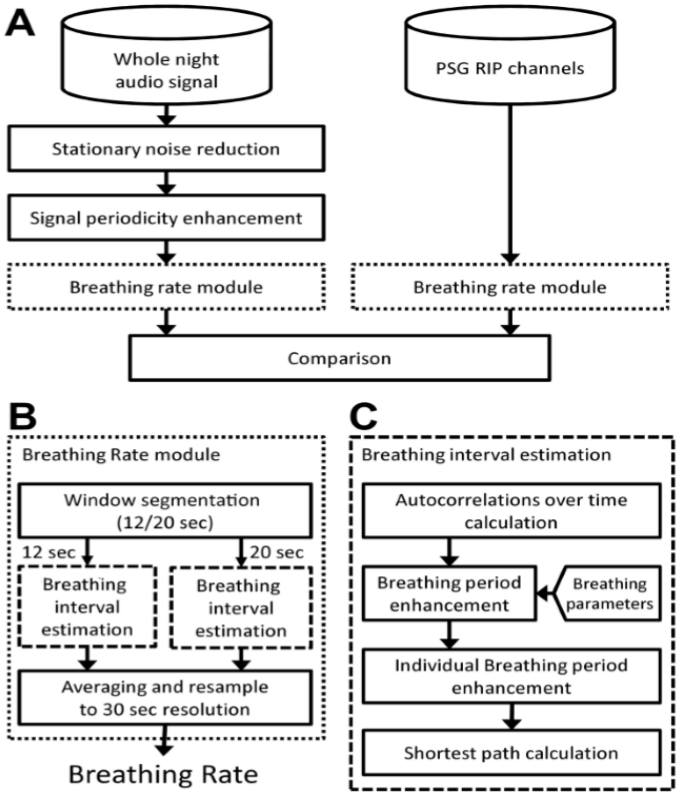

Figura 1.3. Diagramma a blocchi del procedimento proposto in [5].

Capitolo 2

Misura della frequenza respiratoria

2.1 La respirazione

2.1.1 Apparato respiratorio

L' apparato respiratorio corrisponde all'insieme di strutture rigide, semirigide ed elastiche, che assicurano il ricambio gassoso funzionando da regolatori di flusso d'aria. Schematicamente le vie aeree si ripartiscono in tre sezioni disposte in serie:

- il naso, organo esterno dell'apparato respiratorio, le cui cavità sono conformate in modo tale da trasformare il flusso d'aria da laminare a turbolento. Ciò favorisce il riscaldamento, l'umidificazione ed il filtraggio dell'aria inspirata ottenuto mediante il sistema pilifero interno.
- il dotto comprendente la faringe, la laringe e la trachea. La laringe unisce alla funzione di transito dell'aria quella più complessa della fonazione; la trachea, costituita da un condotto lungo circa 11 cm e con diametro variabile tra 1.5 cm e 2.5 cm, è invece la parte più propriamente sede del flusso respiratorio.
- l'albero bronchiale, che si sviluppa a livello della quinta vertebra toracica, dopo la biforcazione della trachea nei bronchi destro e sinistro con la loro successiva arborizzazione. I due bronchi, diramandosi circa 20 volte, danno luogo a dei condotti, i dotti alveolari, quindi ai bronchioli e, infine, agli alveoli.

I polmoni, che contengono circa 300 milioni di alveoli nell'adulto, sono costituiti dalle diramazioni finali dell'albero bronchiale (dotti alveolari, bronchioli e alveoli), hanno forma conica ed occupano la cavità toracica. Ciascun polmone presenta una superficie esterna convessa, una superficie inferiore o diaframmatica e una superficie mediale all'interno della quale vi sono il cuore e le altre strutture mediastiniche. L'apparato respiratorio ha il compito fondamentale di realizzare il processo chiamato respirazione esterna, processo che prevede lo

scambio di O_2 e CO_2 tra il sangue che irrora gli alveoli e l'aria contenuta in essi. Una rappresentazione grafica delle vie aeree è riportata in Figura 2.1.

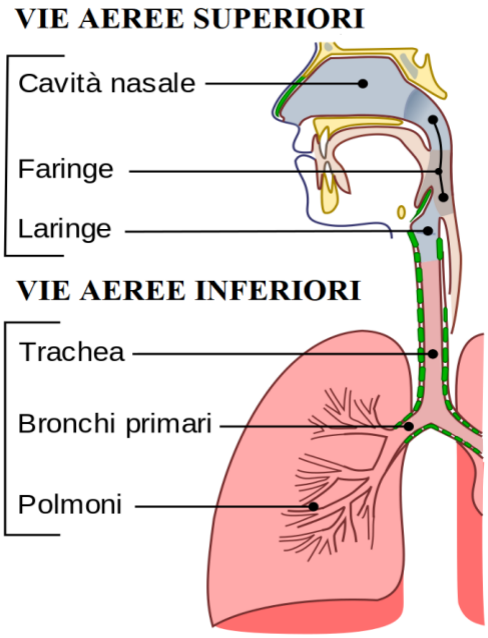

Figura 2.1. L'apparato respiratorio.

2.1.2 Inspirazione ed espirazione

Il flusso d'aria si sviluppa quando c'è un gradiente di pressione: va da aree ad alta pressione verso aree di bassa pressione. Il movimento della cassa toracica durante la ventilazione genera condizioni alternate di alta e bassa pressione all'interno dei polmoni così che possa avvenire lo scambio di aria tra ambiente esterno e polmoni. Secondo la legge di Boyle dei gas ($P_1 \cdot V_1 = P_2 \cdot V_2$) un aumento di volume determina una riduzione di pressione, cosa che avviene durante un ciclo respiratorio. Durante l'inspirazione infatti, il volume polmonare aumenta in seguito alla contrazione dei muscoli del torace e del diaframma. Quando il volume della gabbia toracica aumenta, la pressione all'interno diminuisce e l'aria fluisce nei polmoni. Mentre l'aria continua ad entrare negli alveoli, la pressione aumenta gradualmente fino a che la cassa toracica smette di espandersi. Al termine dell'inspirazione, il volume d'aria presente nei polmoni è al massimo valore raggiunto nel ciclo ventilatorio e la pressione alveolare è uguale a quella atmosferica. Il ritorno elastico dei polmoni riporta il diaframma e le coste alla posizione di partenza permettendo l'espirazione. In questa fase i volumi polmonari e del torace diminuiscono, mentre la pressione dell'aria nei polmoni aumenta. La pressione alveolare ora

è superiore a quella atmosferica, quindi il flusso d'aria si inverte e l'aria esce dai polmoni. Al termine dell'espirazione, il movimento dell'aria si blocca quando la pressione alveolare è di nuovo uguale a quella atmosferica. A questo punto il volume polmonare arriva al proprio valore minimo nel ciclo ventilatorio ed il ciclo respiratorio termina così che ne inizi un altro. Il numero di cicli respiratori, composti da fase di inspirazione e espirazione, fatti in un minuto rappresenta la frequenza respiratoria di un individuo (RR, *Respiration Rate*). Gli uomini hanno una frequenza di circa 12 atti respiratori al minuto, le donne di circa 20 atti respiratori al minuto, mentre i neonati hanno una frequenza variabile tra 35 e 45 atti respiratori al minuto. I valori di questo parametro variano mediamente tra 10 e 50 (atti respiratori per minuto), che corrispondo rispettivamente alle frequenze di 0.16 Hz e 0.83 Hz. L'ampio intervallo di questi valori è dovuto al fatto che la frequenza respiratoria è influenzata da molteplici fattori, quali l'età dell'individuo esaminato, il sesso, l'attività compiuta in fase di acquisizione, ed eventuali patologie.

2.2 Metodo di acquisizione: razionale

Lo strumento di acquisizione proposto in questo studio si presenta come un apparecchio facile da usare, a basso costo e non intrusivo per l'individuo in esame. Attraverso il microfono, che compone il modulo proposto, vengono acquisiti dei segnali audio (in formato *.WAV*) generati dal flusso d'aria relativo all'atto di inspirazione ed espirazione di individui che compiono alcuni test sotto sforzo fisico ad intensità variabile. Dopo la fase di acquisizione, i segnali vengono elaborati ed analizzati attraverso l'ambiente di calcolo matematico MATLAB. Il segnale, dopo una fase di elaborazione, presenta un andamento che oscilla fra i valori 1 e -1, essendo un segnale normalizzato. Questo andamento oscillatorio delinea una serie di massimi locali che si susseguono. I massimi locali che si seguono rappresentano atti respiratori composti da fase di inspirazione e espirazione. L'algoritmo sfrutta questo andamento e si basa sulla funzione, implementata in MATLAB, di nome *findpeaks( )* per individuare la posizione ed il numero di massimi locali. Questa funzione richiede diversi valori in ingresso:

```
findpeaks(y_filt,fs,'MinPeakDistance',0.2,'MinPeakProminence',0.002).
```

I parametri forniti alla funzione sono i seguenti: il segnale precedentemente filtrato "y_filt", la sua frequenza di campionamento "fs", il distanziamento minimo nell'asse del tempo fra un massimo e il successivo e il valore minimo di prominenza del picco. Il minimo del distanziamento temporale fra due picchi consecutivi è impostato a 200 ms. Con il termine

"prominenza" si fa riferimento alla misura della retta perpendicolare alla base del massimo locale che interseca il valore massimo del picco. Nel nostro caso viene impostato il valore minimo a 0.002 in ampiezza del segnale. Dopo questa fase di elaborazione, i massimi locali del segnale vengono segnalati e numerati nel grafico del segnale come rappresentato nella Figura 2.2.

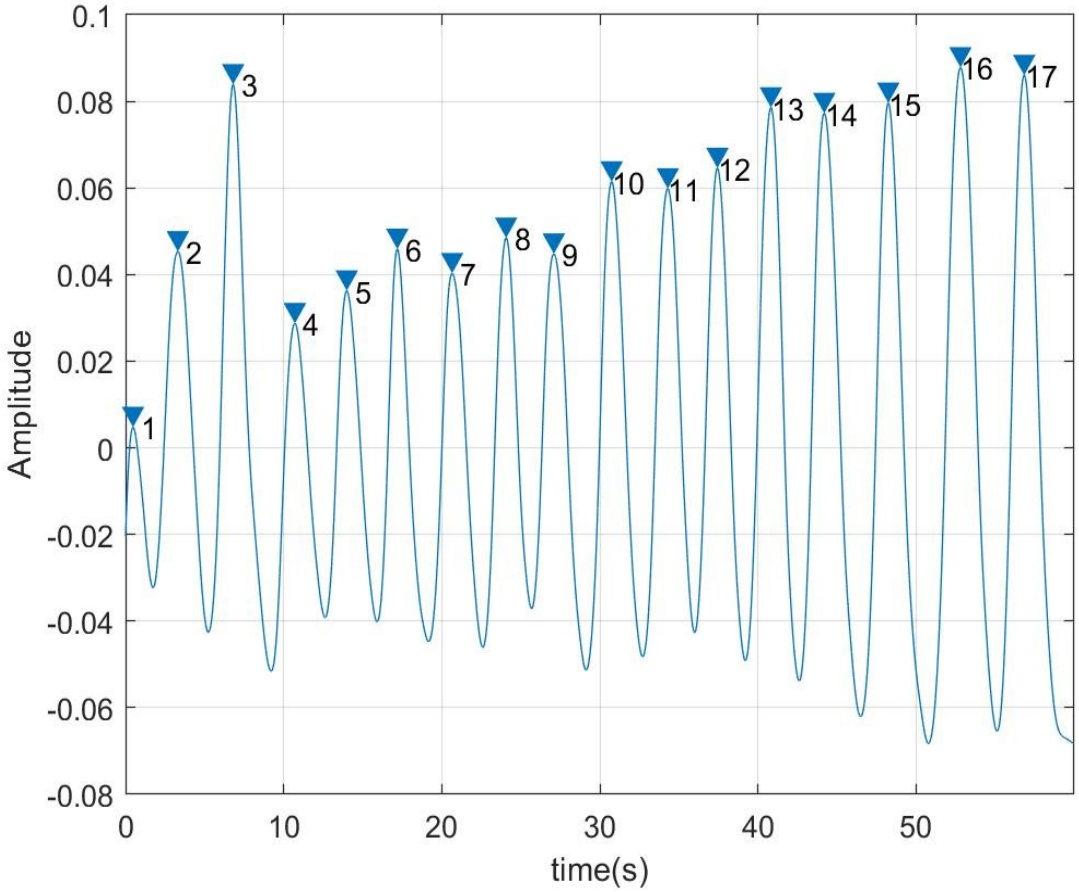

Figura 2.2. Massimi locali segnalati nel grafico dalla funzione findpeaks( ).

Capitolo 3

Progettazione del sistema

3.1 Arduino UNO

3.1.1 Caratteristiche e struttura hardware

La famiglia delle schede “Arduino” ha rappresentato negli anni uno strumento versatile, di semplice utilizzo e a basso costo per realizzare progetti anche senza possedere una specifica preparazione tecnica in elettronica o nella programmazione di microcontrollori. Una delle boards più utilizzate e più celebri corrisponde alla Arduino UNO.

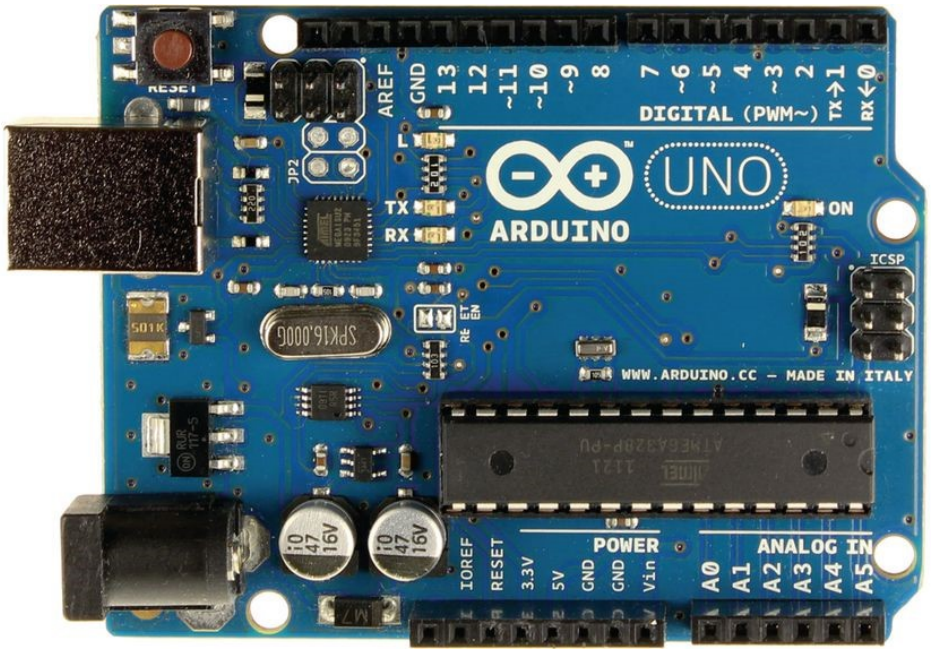

Figura 3.1. Arduino UNO.

La scheda Arduino UNO, mostrata in Figura 3.2, si basa sul microcontrollore ATmega328, dispone di 6 pin analogici e 14 digitali e l’alimentazione della board è garantita tramite porta USB o tramite l’apposito connettore, entrambi installati sulla scheda. Il microcontrollore è dotato di un modulo ADC (*Analog-to-Digital Converter*) interno in grado di convertire un segnale analogico in un segnale digitale, realizzando quindi i processi di campionamento e di quantizzazione sul segnale in ingresso. La frequenza di clock nominale del microcontrollore è

di 16 MHz, e da questo valore si può impostare un'arbitraria frequenza effettiva di conversione attraverso delle librerie specifiche. Trattando il processo di quantizzazione, la board Arduino UNO viene concepita come scheda con risoluzione di 10 bit. Il microcontrollore è dotato di tre tipi principali di memoria:

- La memoria flash usata per la memorizzazione degli sketch implementati.
- La memoria SRAM (*Static Random Access Memory*) che corrisponde allo spazio in cui Arduino crea e modifica le variabili chiamate.
- La memoria EEPROM (*Electrically Erasable Programmable Read-Only Memory*) che viene usata per immagazzinare informazioni a lungo termine.

Rispettivamente Arduino UNO presenta 32 Kbytes in memoria flash, 2 Kbytes in SRAM e 1 Kbyte in EEPROM. La memoria flash e la EEPROM sono memorie non volatili (mantengono le informazioni anche in assenza di alimentazione elettrica); al contrario la SRAM è una memoria volatile e viene persa ogni volta che la board viene disconnessa da una fonte di alimentazione elettrica. La scheda prevede anche un pulsante per il reset generale e un pin chiamato 'ICSP header' attraverso il quale il microcontrollore può essere programmato senza dover essere rimosso dalla board.

3.1.2 ADC

Il modulo ADC (*Analog-to-Digital Converter*) ha la funzione di convertire i segnali analogici in ingresso al microcontrollore in segnali digitali.
L'ADC del microprocessore ATmega328 è basato su un'architettura di tipo SAR (*Successive Approximation Register*) a 10-bit, che sebbene sia più lenta rispetto ad altri convertitori (come, ad esempio, quelli di tipo Flash) è piuttosto comune nei microcontrollori commerciali per l'interfaccia semplice, la linearità e l'efficienza dei consumi. L'architettura SAR utilizza, infatti, un comparatore che riceve in ingresso il segnale da convertire e un segnale generato internamente da un convertitore digitale-analogico (DAC, *Digital-to-Analog Converter*) e, seguendo una sorta di algoritmo di ricerca binaria, consente di approssimare la tensione in ingresso, dal bit più significativo a quello meno significativo; da cui il nome "ad approssimazioni successive". Ad ogni ciclo del clock interno all'ADC, l'uscita del DAC converge verso il valore del segnale in ingresso e il risultato è memorizzato nel registro di uscita. Questa operazione di conversione richiede un certo numero di cicli, che aumenta al crescere del numero di bit del convertitore. La durata di ciascun ciclo è comunque limita dai

tempi di commutazione della logica interna e del DAC. Le principali caratteristiche dell'ADC dell'ATmega328 sono comuni anche agli ADC degli altri microprocessori appartenenti alla famiglia Atmel AVR, tutti infatti basati sulla stessa architettura SAR a 10-bit. L'ADC dell'ATmega328 a 28 pin dispone di 6 ingressi che permettono la lettura di segnali di tipo unipolare e *single-ended*, nei quali la tensione in ingresso è compresa tra la tensione di funzionamento della scheda (5V o 3.3V) e la tensione imposta dal pin GND (0V). Un' altra caratteristica di tale convertitore è che non include ingressi relativi a segnali differenziali, ma è possibile ricorrere ad un amplificatore operazionale differenziale esterno che viene alimentato direttamente dalla scheda. Arduino UNO usa la funzione preimpostata *analogRead( )* per l' acquisizione di tensioni in ingresso al convertitore. Tale funzione seleziona il pin di ingresso e comanda la conversione. Tutti i pin analogici sono gestiti da un *multiplexer* che in conformità a quanto previsto dall'utente nello sketch, li connette al convertitore ADC interno il quale può convertire soltanto un ingresso alla volta.

3.1.3 Campionamento e quantizzazione

Il campionamento e la quantizzazione sono i due passi fondamentali per la conversione di un segnale da analogico a digitale. Lo strumento che ha la funzione di eseguire questi due processi è l'ADC illustrato nei paragrafi precedenti.

In generale, campionare un segnale significa registrarne il valore a certi istanti ed il numero di campioni registrati al secondo (ovvero la frequenza di campionamento) deve essere sufficientemente elevato in relazione alla velocità di variazione del segnale, che aumenta al crescere delle componenti ad alta frequenza appartenenti al segnale stesso. Com'è noto, questo problema è oggetto del teorema del campionamento per il quale, dato un segnale a tempo continuo e a banda limitata (f_{max}), la minima frequenza di campionamento (generalmente indicata in letteratura come *frequenza di Nyquist* ed in questo documento evidenziata come f_s) necessaria affinché il segnale campionato contenga la stessa informazione del segnale originario (cioè sia possibile ricostruire il segnale a partire dalla sequenza dei suoi campioni) è data da:

$$f_s \geq 2 \cdot f_{max}. \tag{3.1}$$

Dalla (3.1) possiamo dedurre che il minimo sampling rate deve essere almeno il doppio della banda del segnale da campionare. Se questa relazione non è rispettata, la ricostruzione basata sui singoli campioni produce un segnale assai diverso da quello originario: questo fenomeno

è chiamato *aliasing* e non è correggibile a posteriori, ma deve essere risolto a monte della conversione. Le schede dotate di microprocessore ATmega328 (come la scheda Arduino UNO) hanno una frequenza di clock nominale di 16 Mhz; da questo valore si può impostare la frequenza di campionamento tramite un divisore di frequenza (*prescaler*) il cui valore è determinato in base al contenuto dei primi tre bit (ADPS2, ADPS1, ADPS0) del registro ADCSRA (ADC *Control and Status Register A*). Per modificare i bit dei registri si utilizzano le funzioni *sbi( )* e *cbi( )* e i valori programmabili sono 2, 4, 8, 16, 32, 64 e 128, come riassunto nella tabella seguente:

ADPS2	ADPS1	ADPS0	DIVISORE DEL *PRESCALER*
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Tabella 3.2. I valori del prescaler relativi alle diverse combinazioni di bits.

Dopo il campionamento il segnale risulta essere tempo discreto ma i valori che può assumere variano nel continuo e per essere rappresentati richiederebbero una precisione infinita. Per superare questo problema si ricorre al processo della quantizzazione con il quale si divide la banda d'ampiezza del segnale in 2^n intervalli (n rappresenta il numero dei bit della risoluzione della scheda, 10 nel caso di Arduino UNO) aventi larghezza fissa δ (Quantizzazione Uniforme) e si sostituisce il valore (reale) di ciascun campione con il valore centrale (intero) dell'intervallo nel quale il campione cade. In sostanza durante il processo di quantizzazione viene assegnato al campione il valore discreto più vicino al valore continuo. È facilmente intuibile che facendo in tal modo vengono necessariamente introdotti degli errori dovuti alla differenza tra il valore quantizzato del segnale ed il suo valore reale. La tensione in ingresso al convertitore può essere stimata a partire dal valore risultante dalla conversione dell'ADC moltiplicandolo per la risoluzione di tensione:

$$V \approx ADC_{Value} \cdot {}^{V_{Ref}}/_{2^n}. \tag{3.2}$$

Nella (3.2) con V_{Ref} si indica la tensione di funzionamento della scheda e con n il numero di bit di risoluzione. Nel caso dell'ADC della scheda Arduino UNO a 10 bit di risoluzione e con tensione di funzionamento pari a 5.5V:

$$V \approx ADC_{Value} \cdot {}^{3.3}/_{2^{10}}. \quad (3.3)$$

3.2 Modulo “Grove Sound-Sensor”

3.2.1 Specifiche

Il sensore utilizzato per le acquisizioni audio da connettere alla scheda Arduino UNO corrisponde al modulo “Grove Sound-Sensor” (datasheet nel documento [8]). Tale sensore, mostrato in Figura 3.3, è sviluppato dal produttore Seeed Studio e si basa su un microfono elettrico e sull’amplificatore LM386.

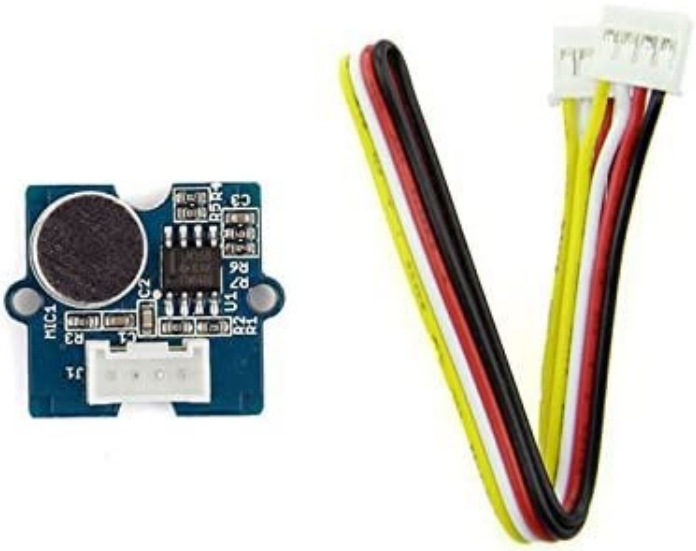

Figura 3.3. Il modulo “Grove Sound-Sensor”.

Il modulo è in grado di percepire il suono che lo circonda e generare un’uscita di tipo analogico che può essere campionata successivamente attraverso programmi di elaborazione logica. Risulta facile da utilizzare, molto economico e prevede delle specifiche indicate nella Tabella 3.4 seguente:

Item	Value
Operating Voltage Range	5 V
Operating Current(Vcc=5V)	4~5 mA
Voltage Gain(V=6V, f=1kHz)	26 dB
Microphone sensitivity(1kHz)	52-48 dB
Microphone Impedance	2.2k Ohm
Microphone Frequency	16-20 kHz
Microphone S/N Radio	54 dB

Tabella 3.4. Specifiche del microfono.

3.3 Comunicazione

3.3.1 Collegamento dei connettori

La comunicazione tra la scheda Arduino UNO, il microfono e l'elaboratore è garantita dal collegamento di quattro connettori:

- Il cavo rosso viene collegato al pin della tensione di funzionamento (3.3V nel nostro caso).
- Il cavo nero viene collegato al pin della tensione di massa (GND), usata come riferimento per la tensione di funzionamento.
- Il cavo giallo viene collegato al pin A0, in grado di acquisire il valore analogico dal microfono, che poi sarà successivamente convertito in digitale dall' ADC.
- Il cavo per la comunicazione seriale USB, installato in Arduino, viene collegato all'elaboratore.

3.3.2 Shield per la comunicazione con scheda microSD

I primi tentativi di acquisizioni si sono basati sul salvataggio dei dati prodotti dalla conversione dell'ADC e stampati a video nel monitor seriale dell'ambiente di programmazione 'Arduino IDE'. Attraverso un programma chiamato *Coolterm* (scaricabile da [9]) si generava un file in formato *.txt* con i valori relativi all'inizio della comunicazione seriale fra Arduino UNO (collegato al microfono) ed elaboratore. Questa tecnica si è rivelata errata ed è stata sostituita

da un altro metodo di acquisizione basato sull'uso della shield per la comunicazione SD (o *data logger*) mostrata in Figura 3.5. Questo apparecchio si inserisce sopra la scheda Arduino UNO e ne riproduce ogni pin; in più offre uno spazio nel quale può essere inserita una scheda di tipo microSD unita al suo adattatore. Nel nostro progetto è stata utilizzata una scheda SD con capacità pari a 16 Gigabytes. In questo modo le acquisizioni sono state salvate direttamente all'interno della scheda microSD in formato *.WAV*, tipico delle registrazioni audio. La libreria di Arduino IDE che inizializza la comunicazione fra microfono e scheda microSD è la *SD.h*.

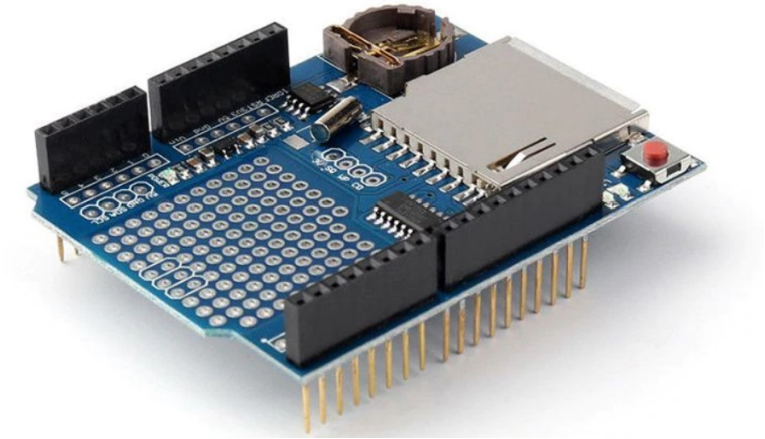

Figura 3.5. Data logger per la comunicazione con scheda microSD.

In Figura 3.6 viene mostrato il sistema di misura completo.

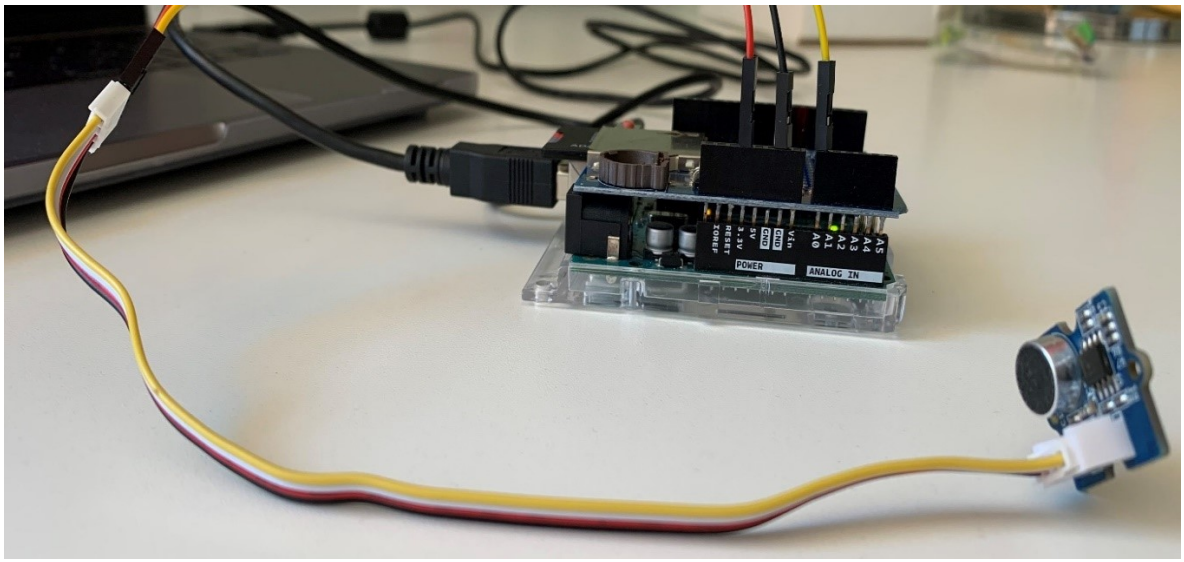

Figura 3.6. Sistema di misura completo.

Capitolo 4

Acquisizione dei segnali respiratori

4.1 Modalità di acquisizione

Dopo aver trattato nel capitolo 3 la struttura hardware dello strumento proposto in questo studio, il capitolo 4 si concentra sul codice implementato mediante l'Arduino Software (IDE) e sull'applicazione dello strumento alla acquisizione ed elaborazione dei segnali acquisiti da diversi individui. L'ambiente di programmazione Arduino Software (IDE) è di tipo *open-source* e fornisce la possibilità di strutturare e di scrivere un codice sorgente da caricare direttamente nella board Arduino.

4.1.1 Codice sorgente per acquisizioni

Il codice implementato con Arduino Software (IDE) è il seguente:

```
#include <SD.h>
#include <SPI.h>
#include <TMRpcm.h>

#define SD_ChipSelectPin 10

TMRpcm audio;

void setup() {

   Serial.begin(115200);

    bitSet(DIDR0, ADC0D);
    bitSet(DIDR0, ADC1D);

  if (!SD.begin(SD_ChipSelectPin)) {
    return;
  }else{
    Serial.println("SD OK");
  }

  audio.CSPin = SD_ChipSelectPin;
}

void loop() {

    if(Serial.available()){
```

```
28.       switch(Serial.read()){
29.         case 'r': audio.startRecording("test.wav", 16000, A0); Serial.println(
    "Recording"); break;
30.         case 's': audio.stopRecording("test.wav"); Serial.println("Stop"); bre
    ak;
31.
32.         case '1': audio.startRecording("test1.wav", 16000, A0); Serial.println
    ("Recording 1"); break;
33.         case 'q': audio.stopRecording("test1.wav"); Serial.println("Stop 1");
    break;
34.
35.         case '2': audio.startRecording("test2.wav", 16000, A0); Serial.println
    ("Recording 2"); break;
36.         case 'w': audio.stopRecording("test2.wav"); Serial.println("Stop 2");
    break;
37.
38.         case '3': audio.startRecording("test3.wav", 16000, A0); Serial.println
    ("Recording 3"); break;
39.         case 'e': audio.stopRecording("test3.wav"); Serial.println("Stop 3");
    break;
40.
41.         case '4': audio.startRecording("test4.wav", 16000, A0); Serial.println
    ("Recording 4"); break;
42.         case 'f': audio.stopRecording("test4.wav"); Serial.println("Stop 4");
    break;
43.
44.         case '5': audio.startRecording("test5.wav", 16000, A0); Serial.println
    ("Recording 5"); break;
45.         case 't': audio.stopRecording("test5.wav"); Serial.println("Stop 5");
    break;
46.
47.       }
48.     }
49.}   .
```

Analizzando il codice implementato osserviamo che le righe numerate da 1 a 17 servono essenzialmente ad includere le librerie usate, settare il parametro della *baudrate* a 115200 simboli per secondo e a disattivare i buffer digitali in modo da evitare il rumore dovuto ai circuiti digitali. Le librerie richiamate in questo codice sono:

- La libreria *SD.h*, che consente la comunicazione e il salvataggio di file direttamente sulla scheda SD.
- La libreria *SPI.h*, che consente la comunicazione seriale tra la board Arduino e gli altri componenti ad essa collegati.
- La libreria *TMRpcm.h*, che viene installata successivamente ed è la funzione che permette di creare le registrazioni audio in formato *.WAV*.

Dalla riga 19 alla 26 si nota una parte con funzione di verifica del corretto collegamento della shield e della corretta comunicazione con la scheda SD. La board stamperà sul monitor seriale la scritta “SD OK” nel caso di procedura corretta. Analizzando il codice fino alla fine si riconosce la parte essenziale ai fini dell’acquisizione del segnale audio: con l’ausilio di comandi da inserire direttamente nella riga di comando del monitor seriale si possono avviare e terminare le registrazioni audio. Sia l’inizio che la fine vengono segnalate rispettivamente dalle scritte a video “Recording” e “Stop”. I parametri delle acquisizioni da fornire in input alla funzione di acquisizione sono la frequenza di campionamento e il pin da “ascoltare”: in questo caso vengono scelti rispettivamente 16000 Hz e il pin A0.

4.1.2 Prove eseguite sugli individui

L’ implementazione del codice sorgente viene seguita da prove svolte sui segnali acquisiti da tre individui di cui due maschi di età 23 e 52 anni, ed una femmina di 16 anni, ai quali si fissa il microfono ad una distanza di 1 cm sotto il naso come mostrato in Figura 4.1.

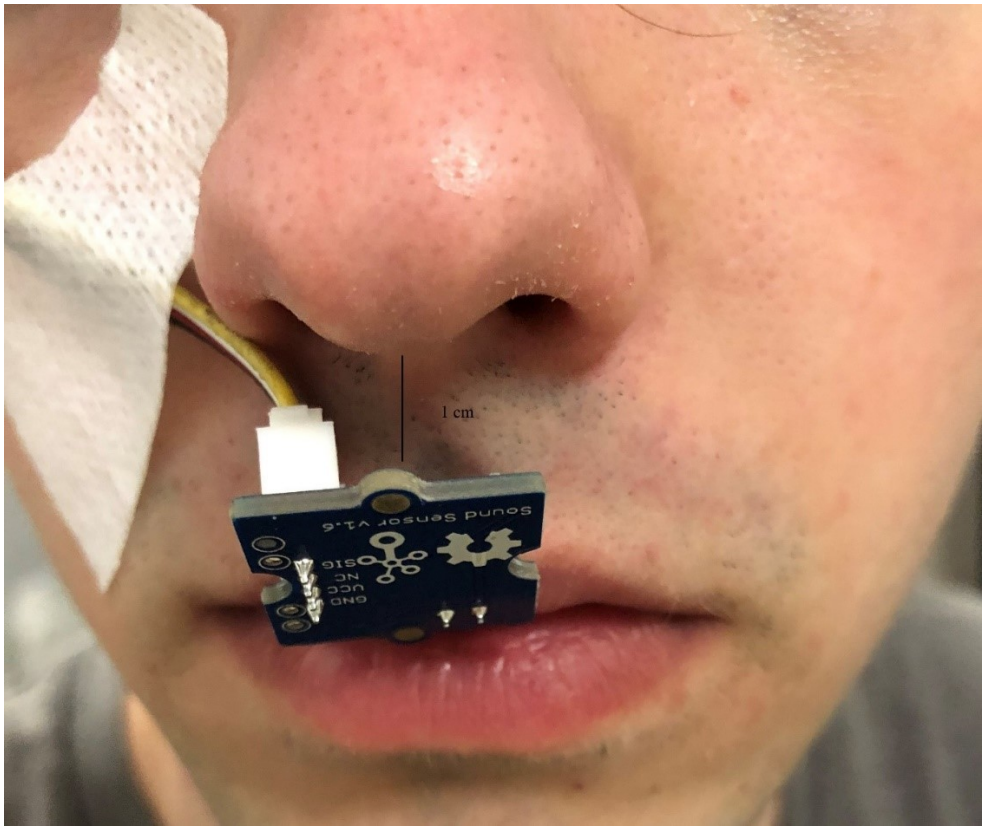

Figura 4.1. Installazione del microfono su un individuo.

Per ciascun individuo vengono eseguiti tre test nei quali la variabile è l’attività compiuta dal soggetto in fase di acquisizione:

- Nel primo test agli individui viene chiesto di respirare mentre si trovano in condizioni di riposo, seduti su una sedia, per un intervallo di cinque minuti.
- Nel secondo test agli individui viene chiesto di respirare mentre camminano a velocità moderata su un tapis roulant, per un intervallo di cinque minuti.
- Nel terzo test agli individui viene chiesto di respirare mentre eseguono un numero di 40 piegamenti distribuiti in un intervallo di cinque minuti.

Le tre attività vengono classificate rispettivamente come riposo, sforzo di lieve intensità e sforzo intenso.

4.2 SPL

Il livello di pressione sonora (SPL) è una misura logaritmica della pressione sonora efficace di un'onda sonora rispetto ad un livello di riferimento, che varia in base al mezzo di trasmissione, e nel caso dell'aria equivale al valore di 0,00002 Pa. Questo valore corrisponde alla soglia uditiva umana quando ci si trova alla frequenza di 1000 Hz. L'unità di misura della SPL è il dB (SPL) e la formula per calcolare questo valore corrisponde a:

$$SPL = 20 \cdot log_{10}(\frac{P_{rms}}{P_{ref}}). \tag{4.1}$$

Nella (4.1) il termine P_{rms} rappresenta il valore della pressione sonora efficace (*Root mean square*, rms), e il termine P_{ref} indica il valore della soglia uditiva. Il valore della pressione sonora efficace per un segnale discreto può essere ottenuto da:

$$P_{rms} = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} P_i}. \tag{4.2}$$

Ci siamo serviti di questo dato per validare il corretto funzionamento del microfono e per dare un valore alla sensibilità dell'apparecchio. Attraverso un pistonofono (secondo la procedura specificata nella norma "IEC 60942-2003"), ovvero uno strumento usato per la calibrazione di strumenti acustici, abbiamo generato un'onda sonora con livello preimpostato di 94 dB(SPL) e abbiamo acquisito alcune registrazioni di quest'ultima mediante il microfono a disposizione, collegato alla scheda Arduino UNO. Il codice usato per queste acquisizioni corrisponde a quello implementato per le acquisizioni respiratorie.

Capitolo 5

Elaborazione dei segnali acquisiti

5.1 Elaborazione in MATLAB dei segnali respiratori

5.1.1 Codice implementato in MATLAB

Dopo la fase di acquisizione trattata nel capitolo 4, le registrazioni vengono elaborate attraverso l'ambiente di calcolo matematico MATLAB (versione R2019b). Le acquisizioni fatte sui tre individui vengono elaborate con lo stesso codice, che ha come obiettivo la ricerca dei picchi di ampiezza del segnale acquisito, che corrispondono agli atti respiratori, e in seguito il calcolo del parametro RR. Il codice implementato per i segnali respiratori è il seguente:

```
1. [y,fs]=audioread('TEST.WAV');
2.
3. x=0:1/fs:(length(y)-1)/fs;
4. y=y-mean(y);
5.
6.
7. figure
8. plot(x,y);
9. xlabel('time(s)')
10.ylabel('Amplitude')
11.
12.Wn=0.9/fs;
13.[B,A]=butter(2,Wn,'low');
14.y_filt=filtfilt(B,A,y);
15.figure
16.plot(x,y_filt);
17.xlabel('time(s)')
18.ylabel('Amplitude')
19.
20.[P,L]=findpeaks(y_filt,fs,'MinPeakDistance',0.2,'MinPeakProminence',0.002);
21.
22.
23.figure
24.findpeaks(y_filt,fs,'MinPeakDistance',0.2,'MinPeakProminence',0.002);
25.title('Picchi con distanza minima di 200 ms e prominenza minima di 0.002');
26.text(L+.02,P,num2str((1:numel(P))'))
27.xlabel('time(s)')
28.ylabel('Amplitude')
29.
30.Picchi_e_posizione=[P,L]; .
```

Le 10 righe iniziali del codice provvedono a caricare la registrazione, a sottrarre il valore medio e a stamparla a video in un grafico che presenta il tempo lungo l'asse delle ascisse, e il valore dell'ampiezza del segnale lungo l'asse delle ordinate. Il segnale è rappresentato da valori generati direttamente dalla acquisizione tramite Arduino UNO collegato al microfono, ed oscilla tra gli estremi 1 e -1. Il segnale stampato a video si presenta come in Figura 5.1.

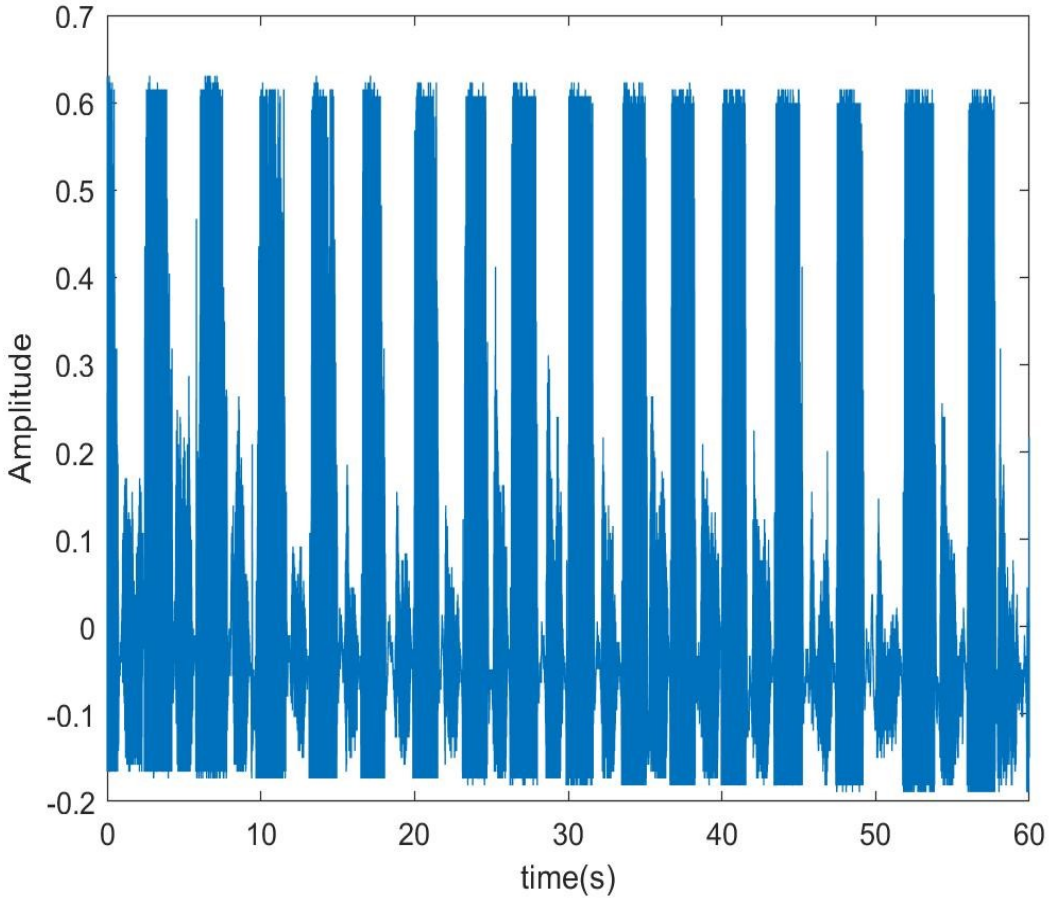

Figura 5.1. La figura rappresenta un estratto di sessanta secondi del segnale registrato in condizioni di riposo da uno dei tre individui.

Nella Figura 5.1 viene presentato un estratto di sessanta secondi del segnale registrato in condizioni di riposo da uno dei tre individui. Le acquisizioni relative ad ogni prova hanno ciascuna una durata di cinque minuti. Ogni intervallo di tempo nel quale il segnale si porta al valore massimo (picco) rappresenta un atto respiratorio. Come si può vedere dalla Figura 5.1 il segnale presenta rumore di fondo che può derivare da fonti diverse in fase di acquisizione. Per rimuovere questo tipo di distorsione, attraverso le righe di codice numerate dalla 12 alla 18, viene implementato un filtro 'passa-basso' con frequenza di taglio pari a 0.9 Hz; successivamente il valore della frequenza di taglio viene diviso per la frequenza di

campionamento del segnale da filtrare (in questo caso tutte le registrazioni sono state acquisite a 16000 Hz). In Figura 5.2 viene mostrato il risultato del filtraggio.

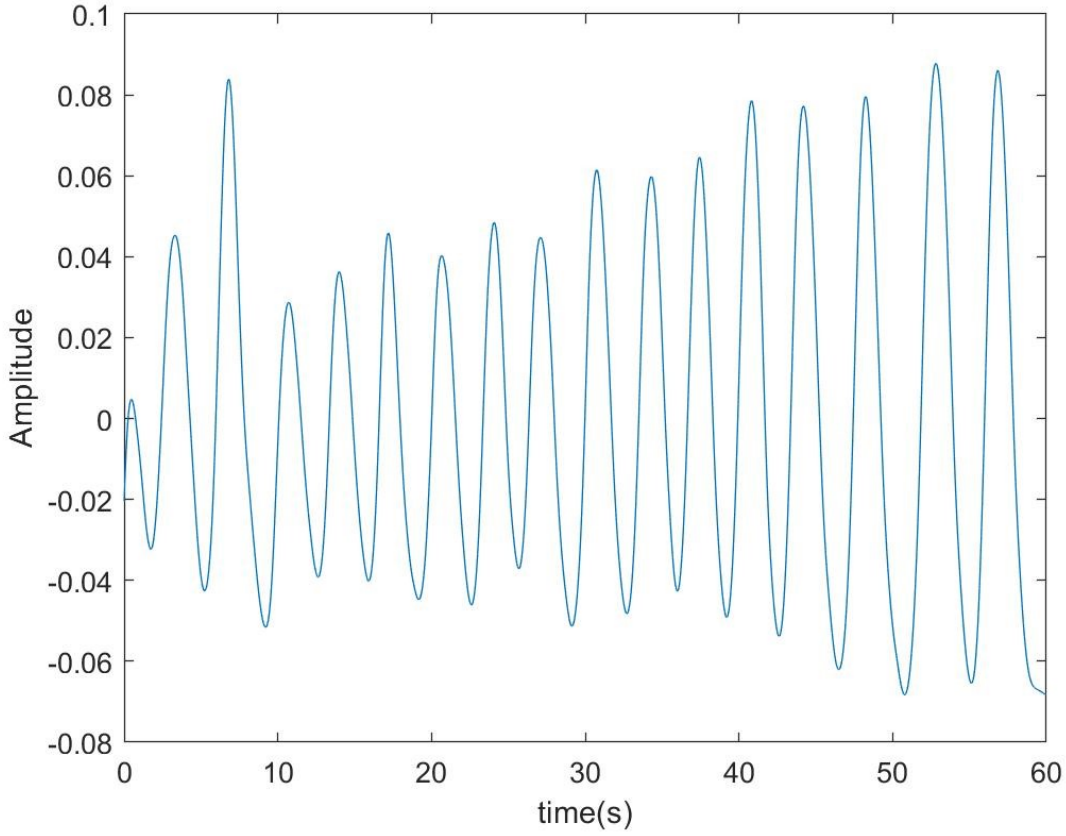

Figura 5.2. Il segnale in Figura 5.1 dopo il filtraggio.

L' ultima parte di codice (righe numerate da 20 a 30) consiste nell' utilizzo di una funzione preimpostata di MATLAB chiamata 'findpeaks' che permette di trovare i massimi locali di un segnale. Come input la funzione riceve il segnale da analizzare, la sua frequenza di campionamento e la modalità di ricerca dei picchi. In questo caso si è impostato che la funzione rilevi dei picchi a distanza minima l'uno dall'altro di 200 ms e con prominenza minima di 0.002. In Figura 5.2 vengono evidenziati i massimi locali trovati nell'acquisizione. Inoltre, attraverso l'ultima riga di comando del codice, MATLAB genera una matrice di dimensione $n\ x\ 2$, nella quale vengono inseriti i picchi rilevati attraverso l'algoritmo: ogni massimo locale viene numerato e le due colonne rappresentano rispettivamente la posizione lungo l'asse delle ascisse (asse del tempo) e l'ampiezza lungo quello delle ordinate (asse dell'ampiezza) del picco.

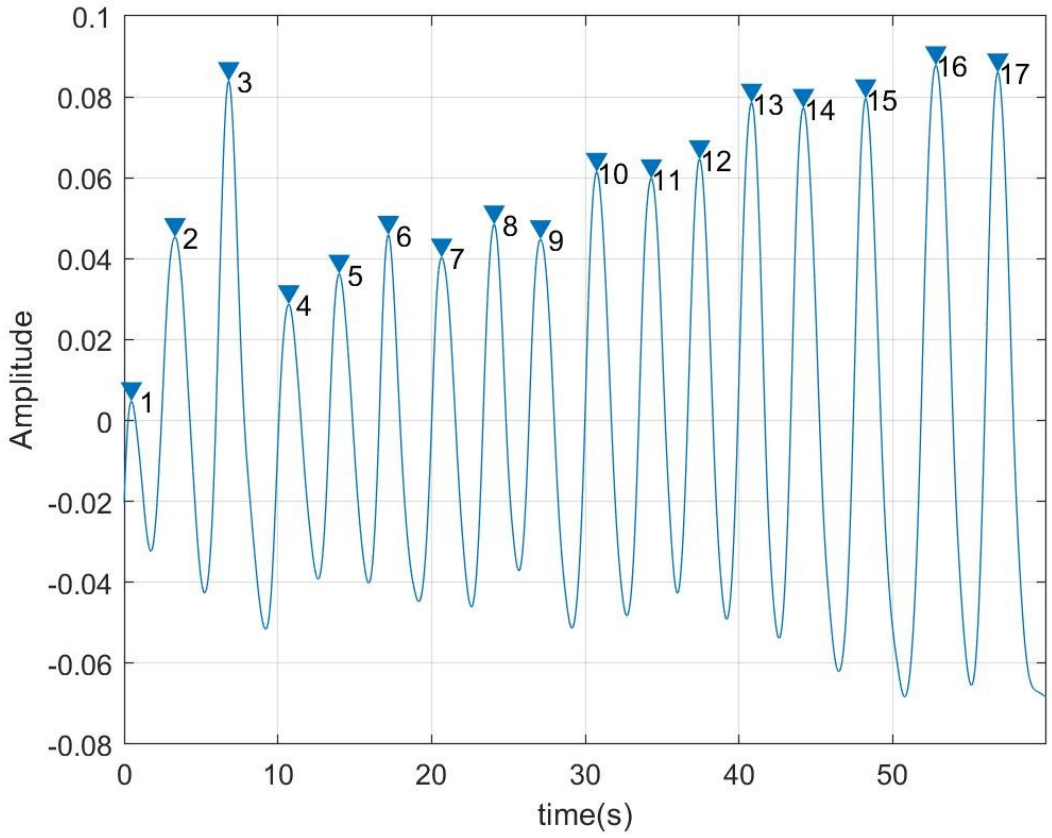

Figura 5.2. Massimi locali rappresentati sul segnale.

5.1.2 Risultati dei test eseguiti

Come spiegato nel capitolo 4, in fase di acquisizione dei segnali respiratori, i tre individui esaminati vengono sottoposti a tre prove nelle quali la variabile corrisponde all'intensità dell'attività in fase di acquisizione. Le tre attività vengono indicate come "Riposo", "Sforzo lieve" e "Sforzo intenso". I risultati acquisiti attraverso il codice implementato e descritto in 5.1.1 vengono raccolti nella Tabella 5.3. I valori della tabella rappresentano il numero totale di atti respiratori compiuti in cinque minuti. Viene riportato inoltre il paragone tra i conteggi ottenuti dall'algoritmo implementato in MATLAB ("Conteggio in MATLAB") e quelli approssimativi fatti dagli individui stessi ("Conteggio individuale"), durante la fase di acquisizione, in modo da confermare la veridicità e l'accuratezza dello strumento di misura progettato. I due dati sono mediati opportunamente fra le tre prove ed i tre individui.

Attività	*Conteggio in MATLAB*	*Conteggio individuale*
Riposo	103	100.33
Sforzo lieve	108.67	105.67
Sforzo intenso	145.67	141

Tabella 5.3. Numero di atti respiratori in cinque minuti mediato fra i tre individui.

5.2 Elaborazione in MATLAB di acquisizioni audio

Le acquisizioni relative alle prove fatte attraverso il pistonofono vengono elaborate attraverso l'ambiente di calcolo numerico MATLAB. Il codice implementato è il seguente:

```
[soundsig_n, Fs]= audioread('TEST.WAV');
N=length(soundsig_n);
soundsig_t=soundsig_n(5500:63800);
N_t=length(soundsig_t);
smp=1:1:N_t;

soundsig=soundsig_t*512;
soundsig_voltage= soundsig*3.3/1024;
figure;
subplot(3,1,1)
plot(smp,soundsig_t);
xlabel('Campioni');
ylabel('ADC value');


l_time=N_t/Fs;
time=1/Fs:1/Fs:l_time;
subplot(3,1,2);
plot(time,soundsig_voltage);
xlabel('time(s)');
ylabel('V');

sound(soundsig_t,Fs);

%CALCOLO SPL (SOUND PRESSURE LEVEL)

soundsig_P_Pa=soundsig_voltage/316.228;

P_eff=rms(soundsig_P_Pa);
subplot(3,1,3);
```

```
31. plot(time,soundsig_P_Pa);
32. xlabel('time(s)');
33. ylabel('Pa');
34. SPL=splMeter('SampleRate',Fs,'FrequencyWeighting','Z-weighting');
35.
36. C_Tone= audioread('Calibration_tone.WAV');
37. C_Tone=C_Tone(5500:63800);
38.
39. calibrate(SPL,C_Tone,94);
40.
41. [Spl_vect,Leq,Lpeak,Lmax] = SPL(soundsig_t);
42.
43. figure;
44. subplot(4,1,1);
45. plot(time,Spl_vect);
46. title('SPL segnale completo')
47. xlabel('time(s)')
48. ylabel('dB(SPL)')
49.
50. subplot(4,1,2);
51. plot(time(1:fix((N_t/3))),Spl_vect(1:fix((length(Spl_vect)/3))))
52. title('SPL 1:1/3 sig')
53. xlabel('time(s)')
54. ylabel('dB(SPL)')
55.
56. subplot(4,1,3);
57. plot(time(fix((N_t/3)):fix(N_t*2/3)),Spl_vect(fix(length(Spl_vect)/3):fix((lengt
    h(Spl_vect)*2/3))))
58. title('SPL 1/3 sig:2/3 sig')
59. xlabel('time(s)')
60. ylabel('dB(SPL)')
61.
62. subplot(4,1,4);
63. plot(time(fix(N_t*2/3):ceil(N_t)),Spl_vect(fix((length(Spl_vect)*2/3)):ceil(leng
    th(Spl_vect))))
64. title('SPL 2/3 sig:3/3 sig')
65. xlabel('time(s)')
66. ylabel('dB(SPL)')
```

Con le righe di codice numerate da 1 a 23 il segnale viene caricato in MATLAB attraverso la funzione *audioread( )*. Successivamente il segnale viene convertito dai valori in uscita dell'ADC a Volt attraverso la relazione (3.3) relativa alla scheda Arduino UNO. In questa parte di codice è presente anche la funzione *sound( )*, che riceve in input il segnale e la frequenza di quest'ultimo e permette di riprodurlo. La parte successiva del codice (le righe numerate da 24 a 33) converte il segnale da Volt a Pascal attraverso un fattore che si ottiene dalla sensibilità del microfono, specifica fornita direttamente dal produttore. Il segnale ottenuto corrisponde alla pressione che viene esercitata sul microfono dall'onda meccanica incidente in ogni istante discretizzato. Con il segnale in questa forma si rende possibile il

calcolo della pressione sonora efficace per un vettore discreto, secondo la formula (4.2), e successivamente il calcolo della SPL secondo la relazione (4.1). In questo caso ci siamo serviti di una funzione preimpostata in MATLAB chiamata *splMeter( )*. Questa funzione serve per creare un oggetto in MATLAB, rinominato 'SPL', e in questo caso riceve in ingresso la frequenza di campionamento del segnale e il tipo di pesatura delle frequenze. In più, alla funzione si dovrebbe fornire un fattore di calibrazione (influenzato dalle specifiche del nostro microfono, quale la sensibilità), che può essere ottenuto per mezzo della funzione *Calibrate( )*. Quest'ultima riceve come input l'oggetto creato in MATLAB da calibrare, un tono di calibrazione registrato dallo stesso microfono usato per le acquisizioni e il livello di SPL relativo a questo tono. Fornendo in ingresso all'oggetto creato e calibrato il segnale, questo genererà un vettore, rinominato "Spl_vect", che raffigura i valori SPL del segnale in ogni istante. Nel primo grafico di Figura 5.4 viene presentata la SPL del segnale completo con durata di circa quattro secondi. Attraverso l'ultima parte del codice il segnale viene diviso in tre parti e rappresentato nel grafico. Si può notare che la curva cresce istantaneamente ad una quota di 80 dB (SPL) e tende a stabilizzarsi in un intorno di 94 dB (SPL) negli istanti successivi.

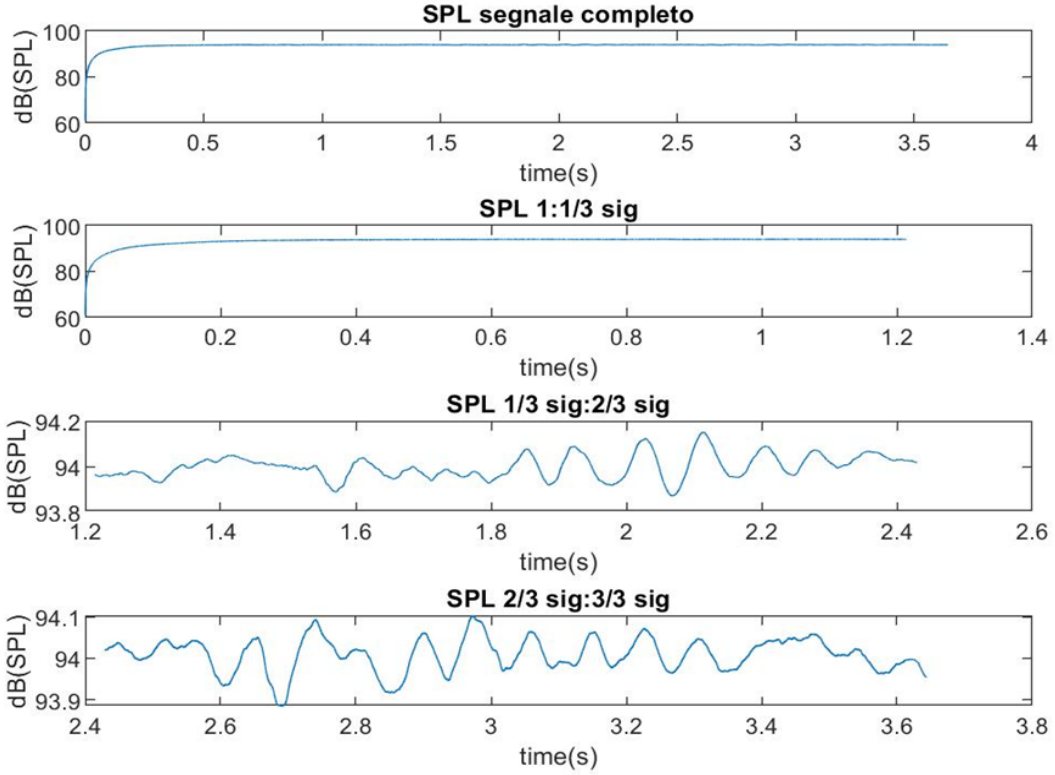

Figura 5.4. Rappresentazione a video dell'andamento del SPL in funzione del tempo.

Conclusioni

Dalle prove fatte e dai parametri osservati durante le attività svolte e riportate in questo documento, si possono trarre delle conclusioni. Le prove relative al parametro SPL hanno evidenziato il fatto che il codice riporta il valore vero della SPL del segnale fornito, essendo noto e impostato a 94 dB (SPL). La fonte di errore più comune che si è presentata ha riguardato la saturazione del segnale da parte del microfono, dovuta al fatto che inizialmente la tensione di funzionamento era impostata a 5 V. Successivamente questa ultima è stata impostata a 3.3 V, tensione che si è dimostrata corretta per il microfono. Le prove fatte riguardanti il parametro RR hanno evidenziato il fatto che gli atti respiratori in cinque minuti, e di conseguenza la frequenza respiratoria, aumentano in accordo con l'intensità dell'attività dell'individuo, come è ragionevole aspettarsi. Dal confronto fra il conteggio individuale (che si suppone essere "vero" e di riferimento) e il conteggio ottenuto da MATLAB emerge che nel caso del riposo e dell'attività a sforzo lieve questi due dati presentano una differenza di 3 atti respiratori. Nel caso dell'attività a sforzo intenso la variazione passa a circa 5 atti respiratori. Citando i documenti [1] e [2], già presentati e descritti in 1.1 come strumenti paragonabili a quello descritto in questo lavoro di tesi sotto l'aspetto concettuale e quello strutturale, i risultati ottenuti dai nostri test possono essere contestualizzati e paragonati: il parametro da analizzare per determinare una stima dell'accuratezza del nostro sistema risulta essere la discrepanza fra gli atti respiratori per minuto rilevati tramite algoritmo implementato e metodica di riferimento usata. In [1] e in [2] la metodica di riferimento usata è il metodo della pletismografia ad induttanza. Confrontando le discrepanze si riesce a dedurre che la discrepanza di 3-5 atti respiratori in 5 minuti di acquisizione, corrispondente ad un'incertezza di circa 0.6-1 atti respiratori al minuto, risulta accettabile in quanto:

- In [1] la discrepanza rilevata tra metodo implementato e metodo di confronto corrisponde a circa 1-3 atti respiratori al minuto.
- In [2] l'incertezza tra metodo implementato e metodo di confronto, quantificata attraverso la relazione (1.1), corrisponde a circa 1%.

Questi margini di errore risultano essere accettabili e permettono di validare lo strumento di misura realizzato. L'analisi economica supporta inoltre la soluzione implementata, in quanto i costi della board Arduino UNO, della shield per la comunicazione con la scheda microSD e

del modulo “Grove Sound-Sensor” sono contenuti. Uno dei punti da analizzare, per ampliare questa trattazione, può essere l’uso di un metodo di paragone dei risultati più accurato e preciso di quello adottato: ad esempio si potrebbero usare metodi quali la pletismografia ad induttanza o l’analisi attraverso pulsossimetro, strumenti usati abitualmente in ambito clinico per la rilevazione della frequenza respiratoria. A livello hardware, i miglioramenti che potrebbero supportare questo strumento possono essere ottenuti implementando un modulo compatto, sul modello di quelli installati su smartphone in [1] e [2], in modo da migliorare la qualità della registrazione e da facilitare le acquisizioni, facendo uso della tecnologia wireless. Gli obiettivi, che ci siamo preposti prima di questo percorso, sono raggiunti attraverso questa tesi, in quanto le caratteristiche principali dello strumento sono il costo contenuto, la semplicità nell’utilizzo e il fatto che risulta non invasivo per gli individui sui quali è applicato.

Bibliografia

[1] K. K. Phokela e V. Naik, "*Use of Smartphone's Headset Microphone to Estimate the Rate of Respiration*", *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, Bengaluru, India, (2020), p. 64-69, doi: 10.1109/COMSNETS48256.2020.9027297.

[2] Y. Nam, B. A. Reyes e K. H. Chon, *"Estimation of Respiratory Rates Using the Built-in Microphone of a Smartphone or Headset"* in IEEE Journal of Biomedical and Health Informatics, vol. 20, n. 6, p. 1493-1501, (Nov. 2016), doi: 10.1109/JBHI.2015.2480838.

[3] D.M. Martínez e A. Martínez, AG 2013, *'Performance evaluation of welch's periodogram-based energy detection for spectrum sensing'* in *IET communications*, no. 11, p. 1117.

[4] R. Bos, S. de Waele, Broersen PMT. '*Autoregressive spectral estimation by application of the Burg algorithm to irregularly sampled data*' in *IEEE Transactions on Instrumentation and Measurement, Instrumentation and Measurement,* 2002;51(6):1289-1294. doi:10.1109/TIM.2002.808031.

[5] E. Dafna, T. Rosenwein, A. Tarasiuk e Y. Zigel, *"Breathing rate estimation during sleep using audio signal analysis"*, *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Milano, (2015), p. 5981-5984, doi: 10.1109/EMBC.2015.7319754.

[6] S. Rolfe, *"The Importance of Respiratory Rate Monitoring"* in British Journal of Nursing, vol. 28, no. 8, (2019), p. 504–508. EBSCOhost, doi:10.12968/bjon.2019.28.8.504.

[7] https://www.arduino.cc/.

[8] https://files.seeedstudio.com/wiki/Grove_Sound_Sensor/res/LM386.pdf.

[9] http://freeware.the-meiers.org/.

[10] F. H. Martini, J. L. Nath, *"Fondamenti di Anatomia e Fisiologia"*, EdiSES (2010).

[11] P. Castano, R. F. Donato, *"Anatomia dell'uomo"*, Edi. Ermes (2006).

[12] F. P. Branca, *"Fondamenti di Ingegneria Clinica"*, vol. 1, Springer (2000).