ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA
Corso di Laurea in Ingegneria Elettronica

DEVELOPMENT OF AN EMBEDDED EMG-BASED WRISTBAND FOR HAND GESTURE RECOGNITION USING MACHINE LEARNING ALGORITHMS

Elaborato in
Hardware and Software Design of Embedded Systems

Tesi di Laurea di:
MATTIA MANCINI

Relatore:
Chiar.mo Prof. Ing.
LUCA BENINI

Correlatori:
Dott. Ing.
SIMONE BENATTI
Dott. Ing.
VICTOR JAVIER KARTSCH MORINIGO

SESSIONE III
ANNO ACCADEMICO 2017–2018

PAROLE CHIAVE

EMG

SVM

Embedded

Wearable

Gestures

Low Power

A mio nonno Marino e alla mia famiglia

Indice

Elenco delle figure

Elenco delle tabelle

Abstract

With the recent improvement of flexible electronics, wearable devices are becoming more and more non-invasive and comfortable, pervading fitness and health-care applications. Wearable devices allow unobtrusive monitoring of vital signs and physiological parameters, enabling advanced Human Machine Interaction (HMI) as well. On the other hand, battery lifetime remains a challenge especially when they are equipped with bio-medical sensors and not used as simple data logger. In this thesis, we present a flexible wristband, designed on a flexible PCB strip, for real-time EMG-based hand gesture recognition. Experimental results show the accuracy achieved by the algorithm and the system implementation. The proposed wristband executes a Support Vector Machine (SVM) algorithm reaching up to 96% accuracy in recognition of 5 hand gestures collecting data from 5 users. The system targets health-care and HMI applications, and can be employed to monitor patients during rehabilitation from neural traumas as well as to enable a simple gesture control interface (e.g. for smart-watches).

Introduzione

Nell'ultimo periodo, i dispositivi indossabili a basso costo hanno conquistato sempre più il mercato. Gli esempi più lampanti sono smartwatch, fitness tracker, braccialetti e altri oggetti indossabili che monitorano i nostri segnali vitali, la nostra attività fisica e le nostre abitudini influenzando sempre più la vita di tutti i giorni. Un aspetto in comune a questi dispositivi è la loro semplicità di utilizzo, che li rende adatti ad un'ampia gamma di utenti.

L'avvento di tali dispositivi ha portato ad una sempre maggiore necessità dell'utilizzo di segnali fisiologici, che in genere vengono acquisiti utilizzando strumenti professionali sopratutto in ambito medico. Da qui la nascita di sensori sempre più sofisticati in grado di essere implementati in dispositivi embedded, come ad esempio il cardiofrequenzimetro con cui monitorare il segnale cardiologico durante la corsa.

Un particolare tipo di segnale fisiologico è quello elettromiografico (EMG) che permette la rilevazione dell'attività muscolare. Per questo motivo, il segnale EMG trova particolare impiego per la rilevazione di pattern di attivazione muscolari, utili in fase di riabilitazione o per il controllo di una protesi.

Il segnale EMG può essere facilmente acquisito tramite l'utilizzo di elettrodi superficiali, i quali però risentono di molte interferenze, rendendo il sistema non standardizzabile.

Nonostante ciò, esistono algoritmi di machine learning e pattern recognition con i quali è possibile riconoscere la contrazione muscolare e associarla ad un determinato gesto in modo da controllare arti robotici o altri dispositivi in maniera semplice.

In questo contesto, l'obiettivo dell'elaborato è quello di realizzare un sistema in grado di acquisire il segnale EMG e processarlo direttamente su scheda in modo da riconoscere in tempo reale il gesto eseguito. A tale scopo è stato realizzato un PCB flessibile su cui sono posizionati gli elettrodi e che viene indossato al polso, rendendo quindi il sistema poco invasivo e facilmente integrabile in uno smartwatch. L'acquisizione del segnale e la sua classifica-

zione viene effettuata utilizzando il microcontrollore STM32 e visualizzata su cellulare grazie ad una applicazione Android. Si raggiunge un'accuratezza superiore al 95% ed una durata di oltre 10 ore con una batteria da $250mAh$.

Nel primo capitolo vengono presentati i vari sistemi già esistenti e le tecniche di classificazione utilizzate analizzando la letteratura presente sull'argomento.

Il secondo capitolo descrive i blocchi principali di un classico sistema di classificazione del segnale EMG andando a descrivere le caratteristiche principali del segnale, le tecniche di elaborazione ed infine il funzionamento dell'algoritmo di classificazione utilizzato.

Il terzo capitolo tratta le varie parti che costituiscono il sistema realizzato: la realizzazione del prototipo per l'acquisizione del segnale, la scheda di acquisizione, l'applicazione mobile e l'utilizzo di MatLab.

Nel quarto capitolo vengono riportati i risultati ottenuti eseguendo prove offline sui dati acquisiti dal dispositivo in modo da trovare la configurazione che fornisse un'accuratezza maggiore. In particolare è stato eseguito un confronto tra le varie tecniche di elaborazione del segnale EMG, su come eseguire il training per la creazione del modello, sul posizionamento degli elettrodi e sul numero di canali.

Una volta trovata la configurazione ottimale, si è passati alla sua implementazione online, descritta nel quinto capitolo, in cui il microcontrollore viene programmato per eseguire su scheda l'intera fase di processing e classificazione.

Il sistema realizzato trova la sua applicazione in dispositivi biomedicali e può essere implementato per monitorare in tempo reale il pattern di attivazione muscolare del paziente o per il controllo di un dispositivo elettronico. Il tutto tramite un sistema embedded poco invasivo e di basso costo.

Capitolo 1

Stato dell'arte

In letteratura esistono numerosi studi che trattano la classificazione di movimenti e gesti tramite l'analisi di segnali biometrici, con una moltitudine di tecniche diverse. Una porzione di questo insieme utilizza dati elettromiografici (EMG) in quanto permettono la rilevazione dell'attività muscolare. Per questo motivo il segnale EMG trova particolare impiego per la rilevazione di gesti, utilizzati in applicazioni Human Machine Interaction (HMI) come ad esempio le protesi. Un'interfaccia naturale basata sui movimenti della mano può migliorare infatti la qualità della vita per una persona con braccia amputate fornendogli un sistema con controllo real time intuitivo [1].

Nelle protesi, le quali richiedono vincoli su affidabilità e robustezza molto stringenti, il riconoscimento dell'attività muscolare è basato su soglie di rilevazione. I gesti che la protesi può eseguire vengono codificati in predefinite flessioni ed estensioni muscolari rendendoli poco intuitivi e richiedendo una necessità di allenamento. Con gli arti artificiali tradizionali, la persona, per compiere un movimento comune e semplice, deve pensare di muovere la sua parte di arto e quello applicato si muoverà per meccanica, rendendoli quindi difficile da utilizzare se non dopo molto allenamento [2].

Il segnale EMG può essere acquisito da elettrodi superficiali, che però risentono di molte interferenze: rumore di linea, alta variabilità del segnale dovuta all'impedenza di contatto degli elettrodi, crosstalk. Inoltre anche la ripetibilità della misura risulta essere un problema: la posizione degli elettrodi infatti cambia da prova a prova ed inoltre i muscoli sono diversi da paziente a paziente [3]. Tutto ciò rende il sistema non standardizzabile.

Esistono però algoritmi di machine learning e pattern recognition coi quali è possibile discriminare e identificare i gesti nonostante la variabilità della misura.

Tali algoritmi sono in grado di riconoscere la contrazione muscolare eseguita e associarla ad un determinato gesto. In questo modo è possibile realizzare dispositivi, come ad esempio arti robotici, che permetteranno alla persona che li porta di effettuare movimenti fluidi ed inconsapevoli, controllati da movimenti più naturali.

1.1 Tecniche utilizzate

La ricerca si è concentrata molto sul confronto tra le varie tecniche di machine learning da applicare al segnale EMG. Questi studi esplorano la classificazione dei gesti della mano su un numero di soggetti che varia da 3 a 6, con un numero di gesti che va da 4 a 9. Questi approcci confrontano diversi algoritmi di classificazione raggiungendo livelli di precisione oltre il 90. In particolare, in [4], la classificazione viene effettuata utilizzando la linear discriminant analysis (LDA). Essa risulta essere semplice e con un training veloce, ma le performance nell'identificazione dei movimenti sono limitate per via dei cambiamenti del segnale emg lungo il tempo. Ciò risulta essere un problema per l'utilizzo in protesi dove è richiesto un controllo prolungato di molte funzioni. In [5] viene confrontata la support vector machine (SVM) con LDA (linear discriminant analysis) e MLP (multiplayer perceptron) concludendo che la SVM ha più potenzialità per la classificazione in sistemi di controllo mioelettrici ed è in grado di riconoscere pattern più complessi. Ciò può portare ad una espansione di funzionalità.

Un'analisi di tali metodi viene effettuata anche in [6]. Vengono confrontati 4 differenti classificatori: Nonlinear Regression (NLR), Multi Layer Perceptron (MLP), Support Vector Machine (SVM) e Linear Discriminant Analysis (LDA). Gli esperimenti sono stati eseguiti sul segnale EMG raccolto da 30 persone, confrontando performance e costo computazionale. Il risultato è che per quanto riguarda le performance di classificazione la SVM risulta migliore, seguita da MLP e NLR.

Un altro punto chiave per il design sono la configurazione e il numero di elettrodi utilizzati per l'acquisizione. In [7] vengono confrontati elettrodi intramuscolari e di superficie acquisendo dati di 10 diverse classi di contrazione. Il risultato è che non ci sono particolari differenze tra l'uno e l'altro per la classificazione dei gesti.

Viene inoltre fatto un confronto tra i sensori attivi e quelli passivi [8] [9]. Sono stati analizzati semplici movimenti della mano classificando il segnale EMG ottenuto da 2 coppie di sensori attivi e passivi posizionati allo stesso modo. I dati sono stati raccolti su 5 soggetti e poi classificati con l'algorit-

mo di autoregressive (AR). I risultati non mostrano particolari differenze di classificazione tra i due (85% uno e 87% l'altro).

L'impatto del numero di elettrodi usati è stato esplorato con diversi risultati. In [7] i risultati mostrano che su 6 soggetti non c'è nessun miglioramento significativo con più di 4 sensori, mentre in [10] si raggiunge la migliore accuratezza, su 12 soggetti, utilizzando una configurazione a 7 elettrodi, confermando un approccio multicanale.

Un altro esempio di sistema è quello descritto in [11] che è in grado di rilevare sia il movimento delle dita (grazie al segnale EMG) sia le rotazioni del polso (grazie ad un accelerometro). Entrambi i tipi di sensore sono montati sul polso. Sono stati eseguiti 15 gesti, ognuno ripetuto 6 volte. Le feature estratte sono le seguenti: root mean square, standard deviation, peak amplitude. Sono poi state testate con SVM raggiungendo una precisione del 96%. I dati vengono acquisiti tramite Arduino e poi processati su PC.

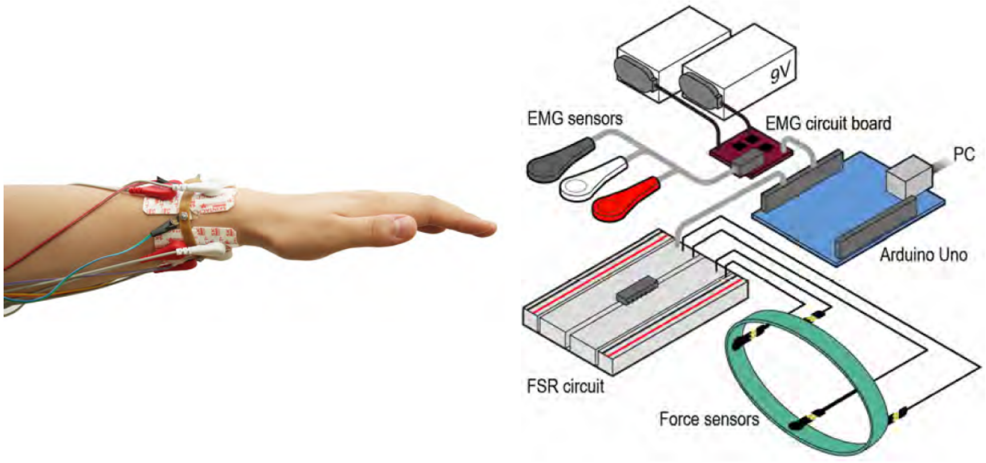

Figura 1.1: Prototipo e diagramma dell'hardware [11]

Tutti i risultati descritti fin'ora arrivano ad accuratezze oltre il 90%, eseguendo processing offline sul PC. Si è quindi dimostrato che gli algoritmi di pattern recognition possono ottenere accuratezza di classificazione elevata, ma gli studi non si concentrano sull'ottimizzazione per le piattaforme indossabili con risorse limitate. Questi approcci infatti, eseguendo processing su PC, non hanno particolari vincoli di dimensione, calcolo e consumi necessari invece per un dispositivo embedded indossabile come ad esempio una protesi.

1.2 Sistemi embedded

Tra i sistemi di controllo commerciali e accademici che utilizzano il segnale EMG per controllare dispositivi elettronici, esiste un divario molto accentuato. Questo è dovuto principalmente al fatto che difficilmente i sistemi proposti in letteratura soddisfino i criteri richiesti per venire accettati ed utilizzati facilmente dai pazienti. In particolare un sistema di questo tipo deve essere [2]:

- intuitivo: il controllo del dispositivo deve essere generato da movimenti naturali;
- adattabile all'utente: piccoli cambiamenti dovuti ad esempio dalla fatica vengono compensati automaticamente;
- real time: il ritardo di risposta non deve superare i 300ms;
- numero minimo di elettrodi, con bassa sensibilità al loro posizionamento;
- l'allenamento (training) deve essere semplice e non troppo lungo (training), senza la necessità di dover ricalibrare spesso il dispositivo;
- bassa complessità computazionale e basso consumo, in modo da poter essere implementato su un dispositivo indossabile.

Attualmente la soluzione commerciale più interessante per l'EMG da indossare per il riconoscimento di gesti è il bracciale MYO [12]. É un bracciale indossabile ed economico che utilizza sia sensori emg che inerziali. Si collega a pc o tablet tramite Bluetooth Low Energy (BLE) e permette lo stream dei dati raw e l'uso di librerie proprietarie per gesture recognition. L'elaborazione del segnale viene eseguita sulla piattaforma host e può riconoscere 5 semplici gesti usando 8 canali EMG e un accelerometro a 3 assi. É il primo sistema commerciale per il riconoscimento del segnale EMG del gesto della mano e può essere utilizzato per il controllo di protesi o altri dispositivi elettronici. Tuttavia è sprovvisto di capacità di calcolo embedded e non può essere quindi utilizzato come un sistema stand alone.

Il design di dispositivi real time efficienti è ancora in fase di studio. Per la scalabilità sono state proposte alcune soluzioni che permettono di integrare il segnale EMG in un sistema di gesture recognition, ma senza presentare un sistema completo: vengono utilizzati sensori low power, ma la maggior parte del processing è fatta su cellulare o pc.

In questo elaborato si vuole realizzare un dispositivo che sia in grado di rilevare il segnale EMG ed elaborarlo direttamente su scheda. Recentemente

Figura 1.2: Myo Motion Control Armband [12]

sono stati fatti studi in direzione di una piattaforma embedded per hand-gesture recognition con capacità di calcolo direttamente su scheda. Per lo sviluppo di un sistema indossabile e di bassa potenza, che ha come obiettivo l'alta precisione, l'approccio più promettente sembra essere la sinergia tra un front-end analogico (AFE) a bassa potenza e un micro, unendo quindi la flessibilità del sistema ad una buona qualità del segnale e mantenendo un buon compromesso tra il consumo di energia e la capacità di calcolo.

In [13] si parla di un AFE ASIC per EMG a 8 canali. Funziona efficientemente ma non fornisce la capacità di elaborazione del segnale. Gli studi fatti in [14] presentano AFE a IC programmabile multicanale, ma senza ADC.

Una soluzione efficiente è presentata in [15] che integra AFE con ADC e micro sullo stesso chip per costruire un ASIC biomedico multisensore. Viene usato un ARM cortex M0 con acceleratore hardware dedicato ottimizzato per l'esecuzione del segnale biomedico in maniera efficiente. Il limite del sistema è però dato dalla presenza di soli 3 sensori.

In [16] viene presentato un dispositivo indossabile per l'acquisizione del segnale EMG. Tale sistema utilizza un microcontrollore ARM Cortex M4 col quale è possibile eseguire direttamente su scheda l'algoritmo SVM per la classificazione in real-time. Tale sistema è in grado di preservare la qualità di acquisizione dei dati dei sensori attivi fornendo al contempo una soluzione configurabile e scalabile con capacità di elaborazione. Durante le prove sono stati acquisiti 7 gesti da 4 utenti diversi posizionando gli elettrodi sul-

l'avambraccio. I risultati del test dimostrano prestazioni simili tra il sistema completamente integrato e indossabile con elaborazione a bordo e il bracciale commerciale con classificazione su PC. Si ottiene una precisione del 92%, con un costo computazionale di soli $29.7mA$. Tale sistema è quindi perfetto per gli scopi descritti.

Figura 1.3: Posizionamento elettrodi [16]

Si vuole quindi sviluppare l'idea presentata in [16], realizzando però un dispositivo facilmente indossabile sul polso risultando così meno ingombrante e facilmente integrabile in un orologio.

Capitolo 2

Classificazione del segnale EMG

Figura 2.1: Descrizione dei macroblocchi del sistema per la classificazione del segnale EMG

Il sistema per la classificazione del segnale EMG può essere suddiviso tre macroblocchi (Figura 2.1):

- **Segnale EMG:** ogni contrazione muscolare genera un segnale elettrico che viene rilavato grazie ad appositi elettrodi e acquisito;
- **Preprocessing:** il segnale EMG grezzo non è di particolare interesse. Per questo motivo, una volta acquisito, subisce varie tecniche di elaborazione per renderlo più adatto al nostro utilizzo;
- **Classificazione SVM:** il segnale ottenuto viene infine dato in ingresso alla SVM grazie alla quale è possibile discriminare il tipo di gesto compiuto.

In seguito verranno analizzati in dettaglio tutti e tre i blocchi.

2.1 Segnale EMG

L'elettromiografia (EMG) è la disciplina legata alla generazione, la misura, l'analisi e l'utilizzo del segnale elettrico emanato dai muscoli [17]. Il suo obiettivo è quello di valutare il funzionamento dei muscoli attraverso i potenziali elettrici da essi generati. Trova particolare utilizzo in ambito medico, dove viene applicata per studiare l'attivazione muscolare di task posturali, per riabilitazione post traumatica e neurologica ed in ambito sportivo per l'analisi del movimento.

2.1.1 Generazione del segnale EMG

L'elemento principale del tessuto muscolare è la fibra muscolare. Ha forma allungata e fusiforme e si contrae a seguito di uno stimolo nervoso. La membrana della fibra muscolare è detta sarcolemma, mentre internamente la fibra è composta prevalentemente dalle miofibrille.

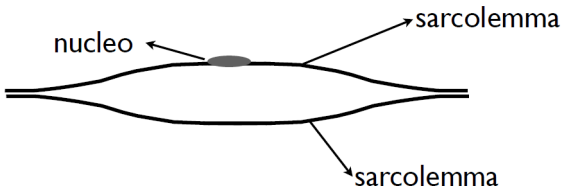

Figura 2.2: Sarcolemma[17]

L'unità motoria (MU) è la più piccola unità funzionale che caratterizza il controllo neurale della contrazione muscolare. É composta dal motoneurone α e dalle fibre muscolari innervate.

La pianificazione del movimento ha luogo nella corteccia pre-motoria, la quale eccita i neuroni della corteccia motoria primaria che a loro volta attivano i motoneuroni α.

L'impulso elettrico propagato dal motoneurone α arriva alla giunzione neuromuscolare e causa la depolarizzazione della giunzione, propagandosi fino ai tendini in entrambi i versi sotto forma di potenziale di azione.

La propagazione del potenziale d'azione si basa sulla generazione di nuovi potenziali nei punti successivi della fibra muscolare. La nascita di un potenziale d'azione in un punto crea una differenza di potenziale rispetto alle zone vicine che sono a riposo dando origine ad una corrente locale. La corrente locale avvia la depolarizzazione della zona inattiva fino al raggiungimento del valore di soglia. Se il valore di soglia viene oltrepassato si genera un nuovo potenziale d'azione e il processo viene ripetuto.

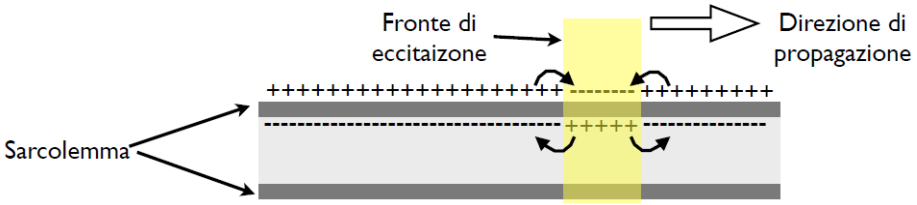

Figura 2.3: Generazione del segnale EMG[17]

2.1.2 Rilevazione del segnale EMG

Il segnale EMG rilevato è una combinazione dei singoli potenziali di azione generati dalla polarizzazione e depolarizzazione della membrana muscolare.

I potenziali d'azione di tutte le unità motorie rilevabili da un elettrodo sono elettricamente sovrapposti e sono rappresentati come segnali bipolari con valore medio pari a zero. Questo è chiamato Pattern di Interferenza (Figura 2.4).

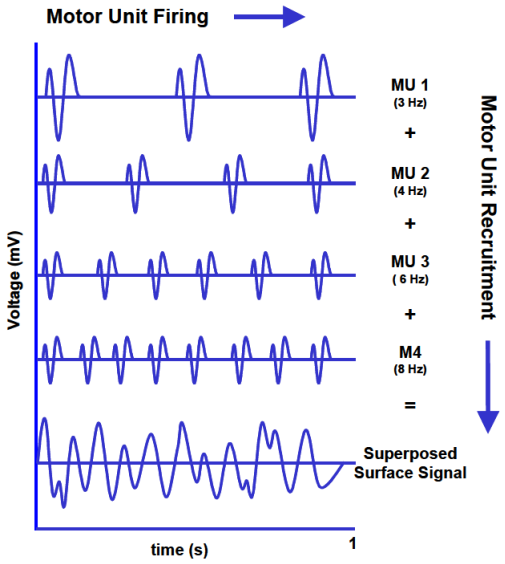

Figura 2.4: Reclutamento di 4 unità motorie a differenza frequenza di firing[17]

I sensori con cui è possibile rilevare il segnale EMG sono gli elettrodi, che possono essere intramuscolari o superficiali. Gli elettrodi utilizzati sono a gel bagnato che garantiscono una migliore conduzione ed un miglior

valore di impedenza. Esistono anche elettrodi asciutti che garantiscono un funzionamento più prolungato ma sono più costosi.

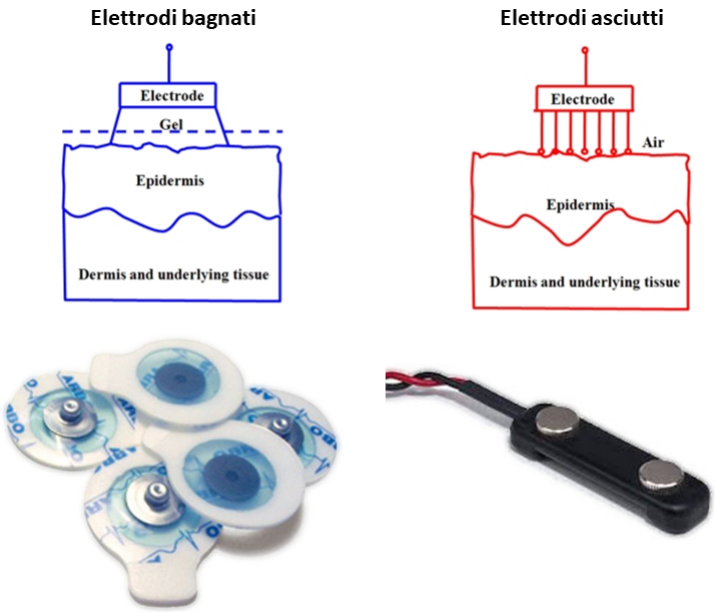

Figura 2.5: Elettrodi bagnati ed elettrodi asciutti

L'uso di elettrodi piccoli aumenta le selettività delle misure ma allo stesso tempo incrementa il valore dell'impedenza. Le coppie di elettrodi devono essere posizionate parallelamente ai fasci muscolari.

Oltre agli elettrodi utilizzati per rilevare il segnale muscolare, è necessario un elettrodo di riferimento che va posto in una zona poco interessata dall'attività muscolare, ad esempio un osso. Dato che nel nostro caso gli elettrodi sono posizionati lungo il polso, è stato preso come riferimento l'epifisi dell'ulna.

Bisogna inoltre prestare attenzione che gli elettrodi rimangano stabili sulla zona di fissaggio durante il movimento e che non si stacchino.

Un altro parametro importante per la corretta rilevazione del segnale EMG è la giusta scelta della frequenza di campionamento. Da un'analisi frequenziale (Figura 2.6) si osserva che la banda del segnale EMG è di 500Hz, mentre la maggior parte del contenuto informativo del segnale è compreso tra 10Hz a 250Hz. Per questo motivo, per rispettare il teorema di Nyquist ed evitare aliasing, è stata scelta una frequenza di campionamento di 1KHz.

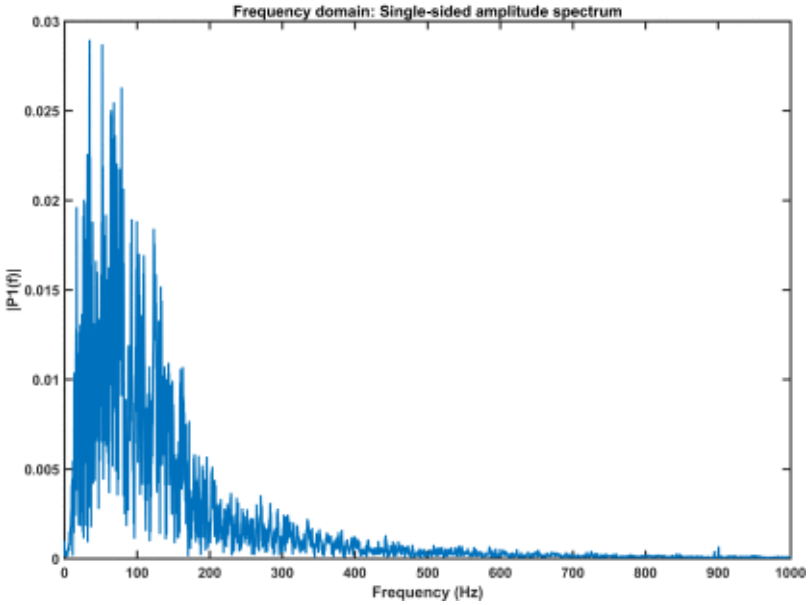

Figura 2.6: Analisi in frequenza del segnale EMG

2.1.3 Fattori che influenzano il segnale

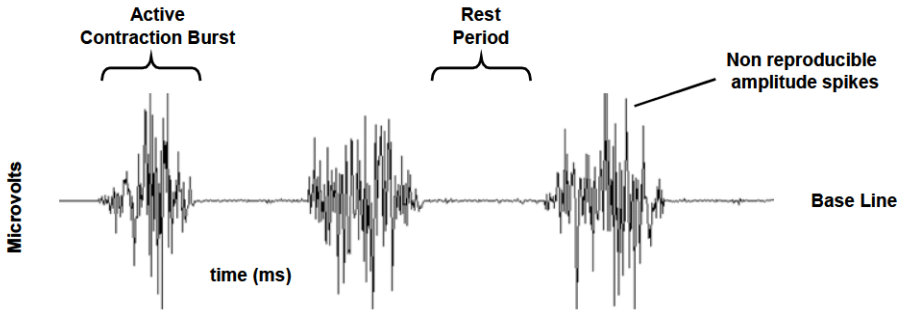

Figura 2.7: Segnale EMG grezzo[17]

In Figura 2.7 viene rappresentato il segnale elettromiografico grezzo rilevato da elettrodi di superficie. É un segnale a media nulla, con un range di ampiezza che va da meno di $50\mu V$ fino a $20 - 30mV$.

Quando il muscolo è a riposo, nessun potenziale di azione si propaga sulle fibre muscolari. In tale condizione si osserva una baseline che dipende da alcuni fattori, tra cui: qualità dell'elettrodo, rumore intrinseco, rumore ambientale e condizioni di misura. Il rumore o problemi del sistema di misura possono portare ad una interpretazione errata del segnale muscolare e venire

confusi con un'attività residua del muscolo. É pertanto opportuno cercare di limitarli il più possibile.

Il set delle unità motorie attivate cambia costantemente e gli effetti dei potenziali d'azione dell'unità motoria si sovrappongono arbitrariamente. Per questo motivo il pattern di interferenza del segnale EMG ha natura casuale. Ciò significa che, a parità di task motorio effettuato con la stessa forza, è del tutto improbabile osservare gli stessi pattern nel segnale, questo perchè il numero di unità motorie reclutate cambia continuamente, come anche la loro frequenza e quella degli impulsi neurali. Per questo motivo si parla quindi di non riproducibilità del segnale.

Applicando algoritmi di filtraggio (paragrafo 2.2) si cerca di limitare la parte non riproducibile del segnale. Idealmente vorremmo ottenere, tramite opportune tecniche di processing, un segnale che sia direttamente legato a una caratteristica del muscolo che si vuole rilevare.

Oltre ad essere irriproducibile, il segnale EMG viene influenzato anche da altri parametri che ne determinano la bontà:

- **Tipo di tessuto:** la conducibilità del corpo umano cambia a seconda del tipo di tessuto, spessore, condizioni fisiologiche e temperatura. Tali condizioni sono fortemente differenti da un soggetto all'altro e variano anche a seconda del posizionamento degli elettrodi.

- **Impedenza della pelle:** per ridurre il più possibile l'impedenza della pelle, è buona norma lavare la zona in cui vengono posizionati gli elettrodi.

- **Cross talk fisiologico:** durante la rilevazione del segnale muscolare, gli elettrodi fanno sensing, oltre che del muscolo di interesse, anche di quelli vicini. Questo tipo di interferenza viene chiamato cross talk fisiologico e in alcuni casi può non essere trascurabile.

- **Variazioni geometriche:** un problema intrinseco nelle misure dinamiche è che, durante la contrazione muscolare, la posizione degli elettrodi può cambiare. Ciò può essere causato anche da una variazione di pressione sull'elettrodo.

- **Rumore esterno:** un altro problema è l'accoppiamento con sorgenti elettromagnetiche esterne e con la frequenza di rete. Per ridurre quest'ultima è stato utilizzato un filtro notch.

- **Rumore di baseline:** viene influenzato da vari parametri, come la qualità degli amplificatori EMG, il rumore circostante, la qualità delle condizioni di rilevazione e la qualità degli elettrodi.

2.2 Preprocessing segnale EMG

Il segnale EMG contiene intrinsecamente importanti informazioni di carattere qualitativo. Ad esempio è possibile sapere se il soggetto utilizza un corretto pattern di attivazione dei muscoli per effettuare un determinato task, cosa molto importante in riabilitazione per verificare la corretta esecuzione di un movimento o semplicemente per sapere se il muscolo è attivo o a riposo.

Se l'obiettivo è quello di ottenere delle informazioni quantitative dall'analisi dell'ampiezza o della frequenza, il segnale EMG necessita di una fase preliminare di elaborazione. Si ricorda infatti che il pattern di interferenza del segnale EMG è di natura casuale e non riproducibile in quanto lo stesso task motorio genera scariche di impulsi simili ma mai uguali. Per risolvere questo problema si cerca quindi di eliminare la parte non riproducibile del segnale tramite tecniche di preprocessing che vadano ad evidenziare il parametro di interesse del segnale. Tali tecniche prendono in ingresso il segnale EMG grezzo e forniscono in uscita un segnale elaborato, che verrà utilizzato come feature per l'algoritmo di classificazione.

In letteratura vengono presentati diversi metodi di preprocessing. In particolare in [18] vengono proposte features nel dominio del tempo e features in quello della frequenza.

2.2.1 Features nel dominio del tempo

Integrated EMG

Integrated EMG (IEMG) viene calcolata sommando i valori assoluti delle ampiezze del segnale EMG. In genere vine utilizzata per rilevare l'attività muscolare ed è relazionata al firing point. Viene espressa come:

$$IEMG = \sum_{n=1}^{N} |x_n| \tag{2.1}$$

dove N è la lunghezza del segnale e x_n rappresenta il segnale EMG.

Mean Absolute Value

Mean Absolute Value (MAV) è calcolata prendendo la media del valore assoluto del segnale EMG:

$$MAV = \frac{1}{N} \sum_{n=1}^{N} |x_n| \tag{2.2}$$

Viene utilizzato come metodo per riconoscere i livelli di contrazione del muscolo.

Il valor medio può essere moltiplicato per dei coefficienti di peso dando così origine a MAV modificati:

$$MMAV = \frac{1}{N}\sum_{n=1}^{N} w_n|x_n| \tag{2.3}$$

Nel Modified Mean Absolute Value 1 (MMAV1) i pesi w_n valgono 1 se $0.25N \leq n \leq 0.75N$ e 0.5 altrimenti. Nel Modified Mean Absolute Value 2 (MMAV2) invece i pesi w_n valgono 1 se $0.25N \leq n \leq 0.75N$, $4n/N$ se $0.25N > n$ e $4(n-N)/N$ se $0.75N < n$.

Variance

La varianza del segnale EMG è una misura della potenza del segnale e può essere calcolata come:

$$VAR = \frac{1}{N-1}\sum_{n=1}^{N} x_n^2 \tag{2.4}$$

Root Mean Square

Il Root Mean Square (RMS) è una delle features più utilizzate per elaborazione del segnale EMG. Viene espresso come:

$$RMS = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x_n^2} \tag{2.5}$$

Waveform Length

La Waveform Length (WL) è la lunghezza cumulativa della forma d'onda lungo il tempo. É legata all'ampiezza dell'onda, alla frequenza e al tempo ed è data da:

$$WL = \sum_{n=1}^{N-1} |x_{n+1} - x_n| \tag{2.6}$$

Zero Crossing

Lo Zero Crossing (ZC) rappresenta il numero di volte che l'ampiezza del segnale EMG attraversa lo zero. Viene utilizzato un valore di soglia per via

del rumore di fondo ed è formulata:

$$ZC = \sum_{n=1}^{N-1} [sgn(x_n \times x_{n+1}) \cap |x_n - x_{n+1}| \geq threshold] \tag{2.7}$$

dove $sgn(x) = 1$ se $x \geq threshold$ e $sgn(x) = 0$ altrimenti.

Willison Amplitude

Willison Amplitude (WAMP) rappresenta il numero di volte che la differenza del segnale EMG tra due elementi adiacenti supera una determinata soglia:

$$WAMP = \sum_{n=1}^{N-1} f(|x_n - x_{n+1}) \tag{2.8}$$

dove $f(x) = 1$ se $x \geq threshold$ e $f(x) = 1$ altrimenti. É legata al potenziale di una unità motoria (MUAP) e al livello di contrazione del muscolo.

2.2.2 Features nel dominio della frequenza

Modified Median Frequency

Modified Median Frequency (MMDF) è la frequenza in cui lo spettro è diviso in 2 regioni con uguale ampiezza. É espresso da:

$$MMDF = \frac{1}{2} \sum_{j=1}^{M} A_j \tag{2.9}$$

dove A_j è l'ampiezza dello spettro del segnale EMG alla frequenza j-esima

Modified Mean Frequency

Modified Mean Frequency (MMNF) è la frequenza media. Viene calcolata come la somma dei prodotti tra l'ampiezza del segnale e la relativa frequenza, divisa per la somma totale delle ampiezze dello spettro:

$$MMNF = \frac{\sum_{j=1}^{M} f_j A_j}{\sum_{j=1}^{M} A_j} \tag{2.10}$$

2.2.3 DWT: sia in tempo che in frequenza

Le tecniche di processing analizzate fin'ora operano o nel dominio del tempo o in quello della frequenza. Esiste una particolare tecnica, che prende il nome di Discrete Wavelet Transform (DWT) in grado di fornire sia informazioni sul tempo che sulla frequenza [19]. Ha una vasta gamma di applicazioni nelle materie scientifiche, ingegneristiche, matematiche ed informatiche. Viene utilizzata per il processing di segnali biomedicali, in comunicazioni ultra-wideband (UWB), per l'elaborazione di immagini e per le comunicazioni digitali.

La DWT del segnale x in ingresso è calcolata attraverso l'uso di una serie di filtri. I campioni x_n sono fatti passare attraverso un filtro passa basso con risposta impulsiva g_n ottenendo come risultato la convoluzione dei due:

$$y_n = (x * g)_n = \sum_{k=-\infty}^{\infty} x_k g_{n-k} \tag{2.11}$$

Contemporaneamente, il segnale viene inoltre decomposto usando un filtro passa alto h_n:

$$d_n = (x * h)_n = \sum_{k=-\infty}^{\infty} x_k h_{n-k} \tag{2.12}$$

I coefficienti di uscita y_n vengono detti coefficienti di approssimazione, mentre i d_n coefficienti di dettaglio.

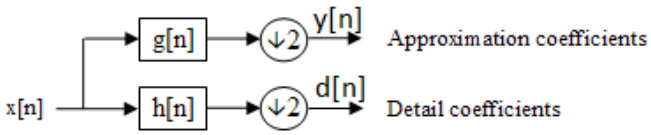

Figura 2.8: Schema a blocchi per il calcolo della DWT ad un livello, dove il blocco $\downarrow 2$ indica un sottocampionamento dimezzato

Ora, dato che la metà della frequenza del segnale è stata rimossa, per il teorema di Nyquist si possono scartare metà dei campioni. Per cui, l'uscita y_n del filtro passa basso viene sottocampionata di un fattore 2 e successivamente processata passando a sua volta in un nuovo filtro passa basso g ed in uno passa alto h con frequenza di taglio la metà del precedente:

$$y_{low,n} = \sum_{k=-\infty}^{\infty} x_k g_{2n-k} \tag{2.13}$$

$$y_{high,n} = \sum_{k=-\infty}^{\infty} x_k h_{2n-k} \tag{2.14}$$

Tale decomposizione viene ripetuta per migliorare la risoluzione di frequenza in uno schema ad albero. Nel caso di 3 livelli di filtri si ottiene lo schema di Figura 2.9, con l'ultimo livello che produce 2 uscite.

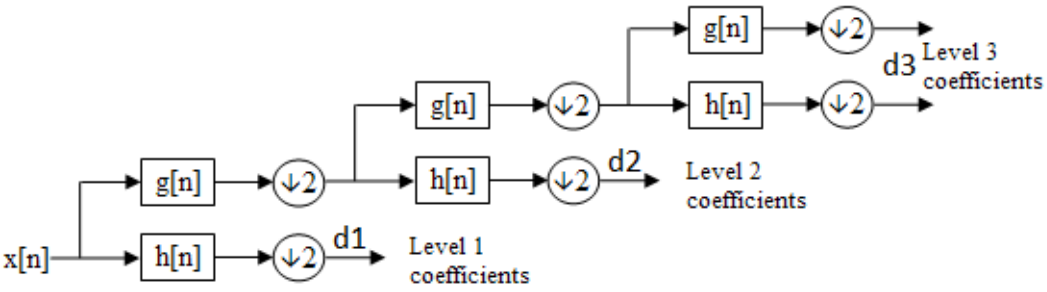

Figura 2.9: Schema a blocchi per il calcolo della DWT a 3 livelli, dove il blocco $\downarrow 2$ indica un sottocampionamento dimezzato

Se consideriamo ad esempio un vettore di 256 campioni con frequenza da 0 a f_n e 4 livelli di decomposizione, l'uscita è così formata (Figura 2.10):

- **Livello 1:** 128 campioni, frequenze da $f_n/2$ a f_n;
- **Livello 2:** 64 campioni, frequenze da $f_n/4$ a $f_n/2$;
- **Livello 3:** 32 campioni, frequenze da $f_n/8$ a $f_n/4$;
- **Livello 4:** 16 campioni con frequenze da $f_n/16$ a $f_n/8$ e altri 16 con frequenze da 0 a $f_n/16$

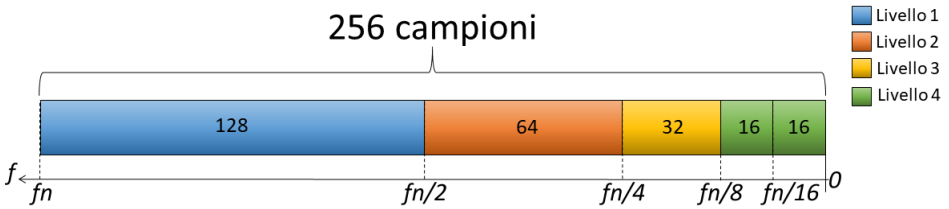

Figura 2.10: Esempio DWT a 4 livelli

Alla DWT è legato il livello di energia. L'energia ai diversi livelli di decomposizione è data dai coefficienti di dettaglio d_n in uscita dai vari livelli:

$$E_j = \sum_k |d_j|^2 \tag{2.15}$$

dove j indica il livello in esame.

2.3 Support Vector Machine: SVM

Tra i vari algoritmi esistenti di Machine Learning il Support Vector Machine (SVM) è sicuramente uno dei più conosciuti in quanto è considerato uno dei migliori algoritmi di apprendimento [4] [5] [6]. Questo algoritmo è stato introdotto da Vapnik e Chervonenkis nel 1965 come una tecnica per risolvere il problema dell'approssimazione di funzione. Tale metodo è basato su una statistica di apprendimento ed è molto conosciuto ed utilizzato come pattern recognition. La SVM ha poi allargato l'idea degli iperpiani di separazione a dati che non possono essere separati linearmente mappando i predittori in un nuovo spazio con più dimensioni dove possono essere separati linearmente.

L'algoritmo SVM lavora in due fasi: una di training e una di testing [20] [21].

Nella fase di **training** l'utente fornisce in ingresso un certo dataset x_n, ognuno etichettato con la propria classe di appartenenza y_n (label), che verranno a formare il training-set T:

$$T = \{x_n; y_n\}; n = 1, ..., N \tag{2.16}$$

Preso come ingresso il training-set, l'algoritmo di training della SVM costruisce una funzione di mappatura dei dati da uno spazio ad un altro, che prende il nome di modello.

Durante la fase di **testing**, l'utente fornisce in ingresso nuovi dati all'algoritmo. Grazie alla funzione modello creata precedentemente, l'algoritmo mappa i nuovi dati ricevuti nello stesso spazio di quelli di training e, confrontandoli con le relative label, fornisce in uscita una predizione sulla classe di appartenenza. L'obiettivo è quindi quello di capire la classe di appartenenza di nuovi dati in ingresso.

Inoltre, durante la creazione del modello, si possono incontrare due problemi frequenti:

- **Underfitting:** il modello creato è troppo semplice e non riesce a classificare bene i dati di testing (ad esempio sono stati considerati pochi

campioni per ogni classe o non sono state considerate alcune classi che poi si verificano nel testing)

- **Overfitting:** il modello creato è troppo complesso. In questo caso il modello generato è troppo specifico per il dataset di training e dati che si discostano da esso non vengono classificati in maniera ottimale.

2.3.1 SVM lineare

Si considera un dataset di training di n punti suddivisi in due classi:

$$(\overrightarrow{x_1}, y_1), ..., (\overrightarrow{x_n}, y_n) \tag{2.17}$$

dove le y_i valgono 1 o -1 e indicano la classe di appartenenza dei punti $\overrightarrow{x_i}$ e ogni $\overrightarrow{x_i}$ è un vettore reale p-dimensionale. Si vuole trovare l'iperpiano avente il massimo margine di separazione tra le due classi, ovvero quello che massimizza la distanza tra l'iperpiano ed il punto $\overrightarrow{x_i}$ più vicino.

Ogni iperpiano può essere scritto come l'insieme di punti di $\overrightarrow{x}$ che soddisfano la seguente relazione:

$$\overrightarrow{w} \cdot \overrightarrow{x} - b = 0 \tag{2.18}$$

dove $\overrightarrow{w}$ è il vettore normale all'iperpiano mentre il parametro $\frac{b}{\|\overrightarrow{w}\|}$ determina la distanza dell'iperpiano dall'origine lungo la direzione del vettore normale $\overrightarrow{w}$. Nel caso di vettori bidimensionali si ottiene il grafico in Figura 2.11.

Se il training-set è linearmente separabile possiamo selezionare due iperpiani paralleli che separano le due classi in modo tale che la distanza tra loro sia la più grande possibile. La regione compresa tra questi due iperpiani viene chiamata margine e l'iperpiano con distanza massima è quello che giace a metà del margine. I due iperpiani che definiscono gli estremi della zona di margine sono descritti da:

$$\begin{cases} \overrightarrow{w} \cdot \overrightarrow{x} - b = 1; \\ \overrightarrow{w} \cdot \overrightarrow{x} - b = -1 \end{cases} \tag{2.19}$$

Vengono completamente determinati dagli $\overrightarrow{x_i}$ che giacciono su di loro, che prendono il nome di support vector.

I dati che dobbiamo prevedere saranno quindi sopra o sotto ad uno dei due iperpiani e assenti nella regione di margine. Maggiore è il margine e maggiore sarà l'accuratezza nella fase di testing.

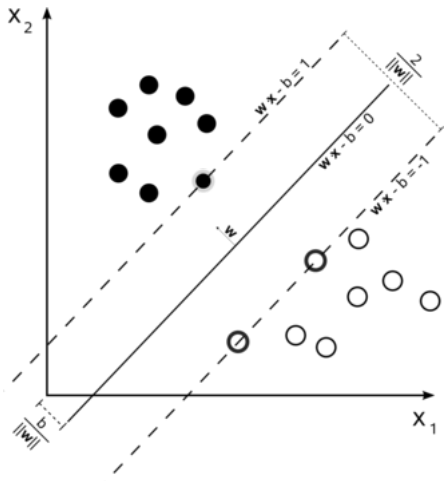

Figura 2.11: Esempio SVM lineare con dati bidimensionali [21]

Nella fase di training, l'iperpiano che viene calcolato deve rispettare la seguente relazione per ogni n:

$$y_n(\overrightarrow{w} \cdot \overrightarrow{x} - b) \geq 1 \tag{2.20}$$

ovvero equivale a dire che ogni dato deve giacere dalla parte giusta rispetto al margine.

Il problema viene quindi ricondotto all'individuazione dei support vector che determinano il massimo margine tra due iperpiani. I support vector sono quelli con distanza minima dall'iperpiano, per cui il problema di ottimizzazione deve risolvere la seguente funzione obiettivo:

$$min\|\overrightarrow{w}\| \; s.t. y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} - b) \geq 1, \forall i = 1, ..., n \tag{2.21}$$

Per estendere l'SVM ai casi non separabili linearmente si introduce la hinge loss function:

$$max(0, 1 - y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} - b)) \tag{2.22}$$

che vale 0 se 2.20 è soddisfatta, ovvero se $\overrightarrow{x_i}$ giace dalla parte giusta rispetto al margine. La funzione obiettivo da massimizzare diventa:

$$\left[\frac{1}{n}\sum_{i=1}^{n} max(0, 1 - y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} - b))\right] + \lambda\|\overrightarrow{w}\|^2 \tag{2.23}$$

dove λ rappresenta un parametro di trade-off che, se aumentato, accresce la regione di margine ma fa calare la certezza di avere la $\overrightarrow{x_i}$ dalla parte giusta, ovvero di avere la predizione corretta

2.3.2 SVM non lineare

In origine tale algoritmo è stato creato per risolvere problemi su dati linearmente separabili. Successivamente però tale concetto è stato esteso anche alla risoluzione di problemi di natura non lineare applicando quello che viene chiamato kernel trick. Formalmente l'approccio è il medesimo, ad eccezione che ogni prodotto scalare viene sostituito da una funzione non lineare (feature function):

$$\Phi : \mathbb{R}^N \to \mathbb{R}^M;\ M > N \tag{2.24}$$

In questo modo l'algoritmo modella l'iperpiano di margine massimo dallo spazio di partenza al feature space. Il feature space non è altro che uno spazio multidimensionale in cui le due classi possono essere separate linearmente.

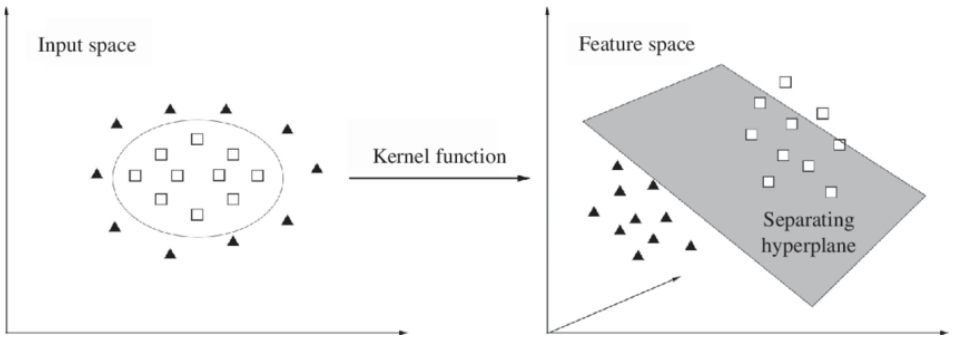

Figura 2.12: Mappatura dallo spazio di partenza al feature space tramite la feature function [21]

Il feature space è uno spazio fortemente multidimensionale e la sua dimensione potrebbe tendere all'infinito. Per questo motivo vengono definite delle funzioni, dette kernel function, che restituiscono il prodotto scalare delle feature functions:

$$K(\overrightarrow{x_i}, \overrightarrow{x_j}) = \Phi(\overrightarrow{x_i}) \cdot \Phi(\overrightarrow{x_j}) \tag{2.25}$$

I kernel più utilizzati sono:

- Lineare: $K(\overrightarrow{x_i}, \overrightarrow{x_j}) = \overrightarrow{x_i} \cdot \overrightarrow{x_j}$;
- Polinomiale: $K(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i} \cdot \overrightarrow{x_j} + 1)^d$;
- Gaussian radial basis function: $K(\overrightarrow{x_i}, \overrightarrow{x_j}) = exp(-\gamma \parallel \overrightarrow{x_i} - \overrightarrow{x_j} \parallel^2)$, con $\gamma > 0$;
- Sigmon a tangente iperbolica: $K(\overrightarrow{x_i}, \overrightarrow{x_j}) = tanh(k\overrightarrow{x_i} \cdot \overrightarrow{x_j} + c)$.

2.3.3 SVM multiclasse

Nelle applicazioni reali capita spesso di avere a che fare con dati che, oltre ad essere non linearmente separabili, si distinguono per più di due classi. L'approccio più usato per trattare questi tipi di problemi è quello di adottare la tecnica "One vs. All". Tale tecnica consiste nel suddividere il problema multiclasse in K problemi di classificazione binaria, in cui il k-esimo classificatore costruisce un iperpiano che separa la classe k dalle altre k-1 classi. Questo tipo di SVM multiclasse è quello su cui si basa la libreria LIBSVM [22] ed è quello che è stato utilizzato nella tesi.

Capitolo 3

Descrizione del sistema

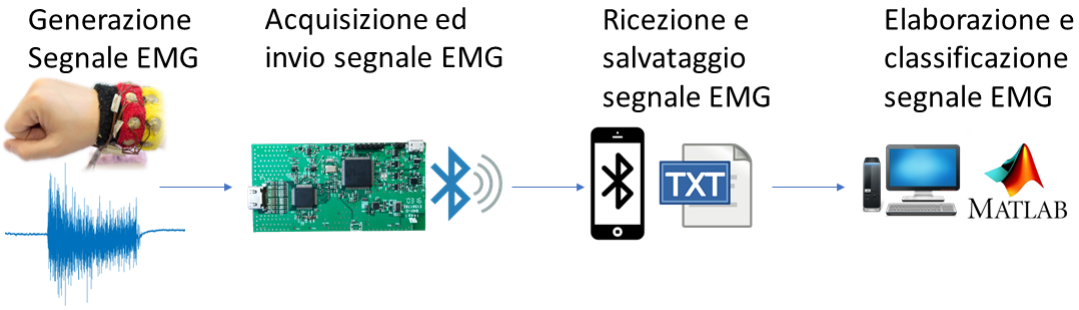

Figura 3.1: Schema del sistema durante il funzionamento offline

Il segnale EMG viene rilevato grazie ad alcuni elettrodi disposti uniformemente lungo la circonferenza di un polsino indossato dall'utente. Il segnale, corrispondente alle contrazioni muscolari relative ai movimenti della mano dell'utente, viene acquisito grazie ad un ADC con frequenza di campionamento pari a 1KHz che comunica col microcontrollore STM32. I dati acquisiti vengono poi inviati via bluetooth al telefono, su cui è preinstallata un'applicazione Android. Tramite questa applicazione è possibile visualizzare in tempo reale i dati che si stanno acquisendo e salvarli in un file di testo txt. I dati vengono poi importati su MatLab con cui vengono eseguite alcune prove di classificazione offline.

In questa prova il microcontrollore viene utilizzato solo per acquisire i dati e inviarli via bluetooth al telefono. Una volta capito, tramite MatLab, quali strategie è meglio adottare per una più corretta classificazione, si passa alla prova online in cui sarà il micro ad eseguire la parte di elaborazione del

segnale e classificazione dei movimenti, che verranno poi inviati al telefono per una loro visualizzazione.

3.1 Realizzazione prototipo

L'obiettivo è quello di classificare, tramite SVM, il segnale EMG acquisito tramite elettrodi disposti sul polso. Per poter realizzare ciò, come prima cosa è stato realizzato un prototipo con cui poter rilevare il segnale desiderato.

Per acquisire il segnale EMG sono stati utilizzati elettrodi adesivi con gel di diametro pari a $24mm$ e resistenza di 220Ω [23].

Gli elettrodi sono stati montati su bottoni ad incastro, posti uniformemente lungo un polsino, in modo da realizzare 7 canali differenziali, dove il canale positivo è quello più vicino alla mano mentre quello negativo è quello verso il braccio (Figura 3.2). Viene poi inserito, sempre sul polsino, un ulteriore elettrodo utilizzato come riferimento di tensione, per un totale di 15 elettrodi. Durante l'acquisizione dei dati, il riferimento va posizionato in una regione che non risente dell'attività muscolare. Per questo motivo è stato scelto come riferimento l'epifisi dell'ulna.

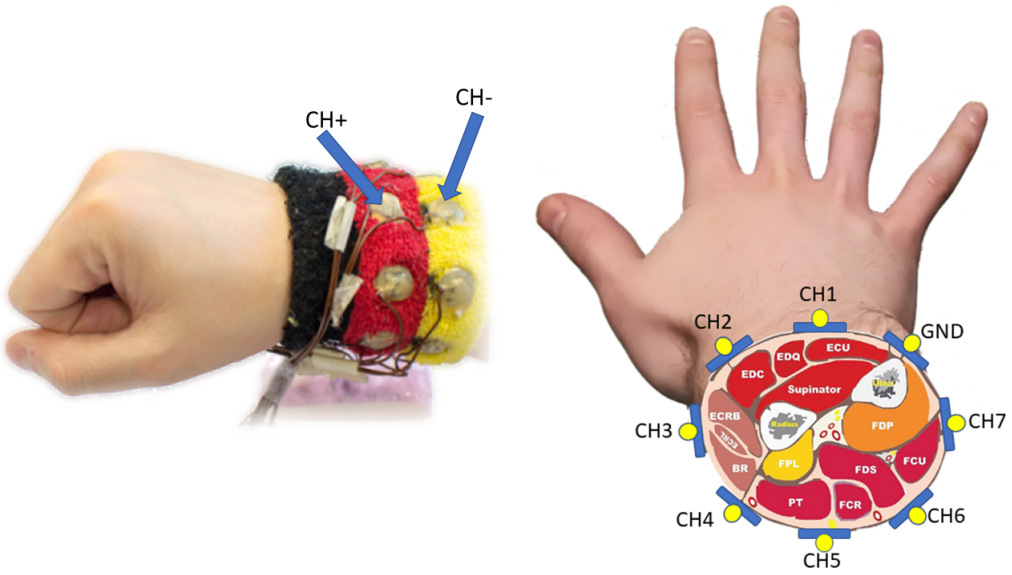

Figura 3.2: Prototipo polsino e posizionamento degli elettrodi

Per ridurre il rumore proveniente dall'ambiente circostante, le capocchie dei bottoni sono state rivestite con isolante elettrico in modo da limitarne la

conduttività. Il polsino è collegato tramite un cavo di tipo HDMI alla scheda di acquisizione.

3.2 Scheda di acquisizione

La scheda di acquisizione utilizzata è composta da una porta con connettore di tipo HDMI collegata agli elettrodi, un convertitore analogico digitale che acquisisce il segnale, un microcontrollore su cui è stato caricato il firmware per elaborare i dati ed un modulo bluetooth per comunicare con l'esterno. Vedremo ora nel dettaglio la parte hardware e quella software della scheda.

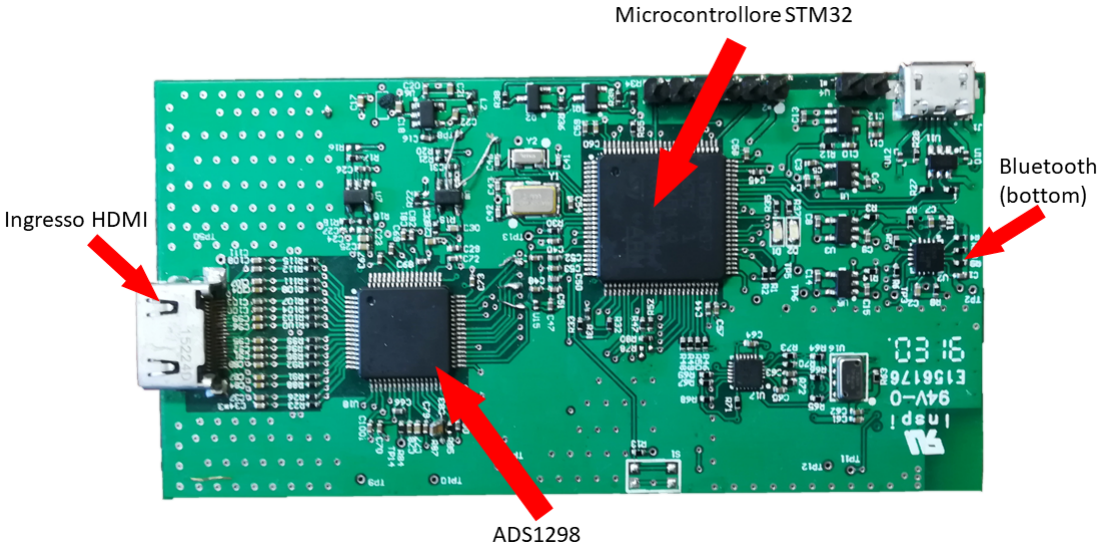

Figura 3.3: Scheda di acquisizione

3.2.1 Hardware

I componenti principali della scheda di acquisizione sono: il convertitore ADC con cui campioniamo i dati in ingresso, il microcontrollore per elaborare i dati e gestire FSM e Interrupt, ed il modulo bluetooth con cui comunicare all'esterno.

Figura 3.4: Schema a blocchi hardware

ADC: ADS1298 [24]

L'ADS1298 è un convertitore analogico digitale creato appositamente per i segnali biomedicali ECG, EMG ed EEG. É un convertitore multicanale costituito da 8 convertitori delta-sigma a 24-bit, ognuno dei quali ha un suo amplificatore di guadagno programmabile (PGA) a basso rumore. Ha un riferimento interno ed oscillatore. Il Data Rate va da 250sps a 32Ksps 32Ksps. Comunica col microcontrollore tramite interfaccia SPI.

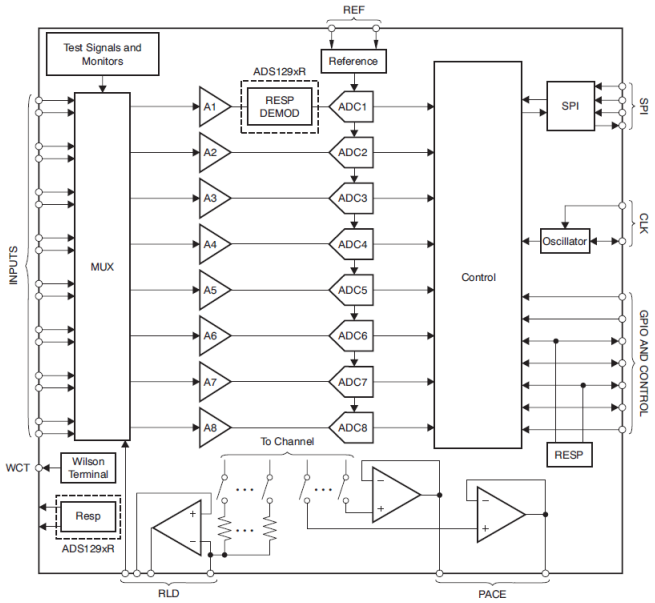

Figura 3.5: Schema semplificato ADS1298[24]

Microcontrollore: STM32F407VGT6 [25]

Il microcontrollore utilizzato è un ARM Cortex-M4 a 32 bit con FPU. Ha una memoria flash di 1MB e una RAM di 192+4 KB.

La frequenza di lavoro è pari a 168MHz, con operazioni in floating point e DSP. Fornisce una trasmissione veloce dei dati grazie agli Advanced High-performance Bus (AHB).

Gestisce anche la modalità di standby per applicazioni ultra low power, con una corrente in stanby minore di $1\mu A$.

Possiede anche numerose periferiche esterne: USB OTG, Ethernet 10/100, 6 porte UART, 3 porte SPI,3 porte I2C, 2 CAN, 2 DAC a 12 bit, 3 ADC a 12 bit, 17 timer e 5 porte di GPIO.

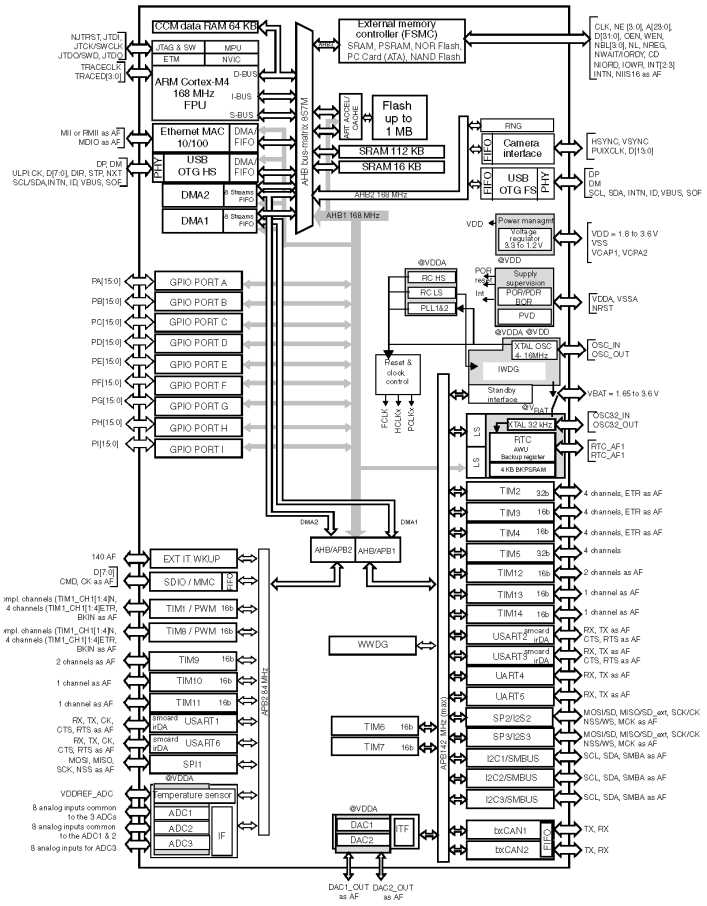

Figura 3.6: Schema a blocchi STM32F407VGT6[25]

Il dato è poi inviato all'utente tramite interfaccia seriale UART collegata ad un modulo bluetooth.

Bluetooth

Per inviare i dati al telefono, è stata utilizzata la trasmissione Bluetooth. Il dispositivo montato su scheda è il BlueGiga WT12, un bluetooth v2.1 + EDR (Enhanced Data Rate) con potenza di trasmissione pari a +3 dBm e sensitività di ricezione di -86 dBm. Grazie all'antenna integrata è in grado di coprire fino a 30m in line-of-sight.

Sensori

La scheda ha anche un sensore di pressione ed un accelerometro, non utilizzati in questa tesi. Il sensore di pressione montato è il MS5611-01BA, mentre l'accelerometro è il MPU-9150.

Figura 3.7: Modulo bluetooth e sensori scheda

3.2.2 Firmware

Il microcontrollore deve essere in grado di acquisire i dati provenienti dall'ADC, elaborarli e inviarli in uscita tramite il Bluetooth. Per gestire tutti i vari task è stata realizzata una macchina a stati (Figura 3.8).

Durante la fase di accensione vengono inizializzate le variabili e si entra poi nella macchina a stati che funziona nel seguente modo:

- **Init:** il sistema crea alcune variabili locali utili per il processo e abilita l'ADS.

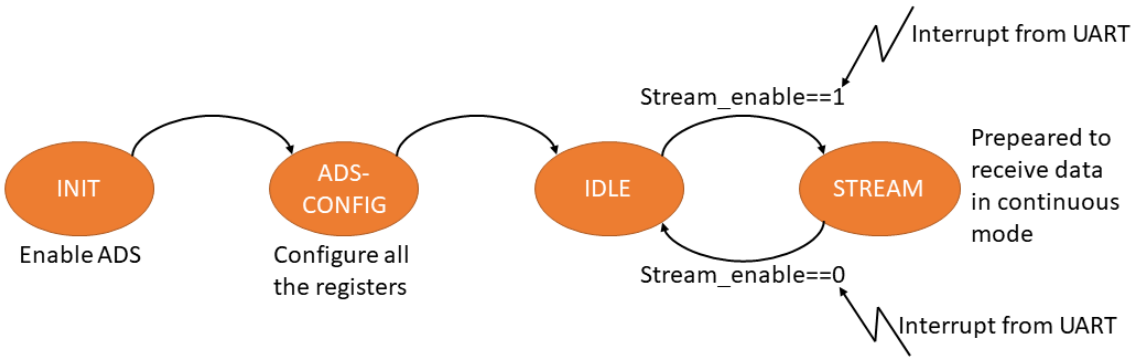

Figura 3.8: Macchina a stati

- **ADS configuration:** attravero SPI vengono mandati all'ADS i comandi per la sua configurazione. L'ADS viene fatto funzionare con una frequenza di campionamento pari a 1KHz ed in continuous mode. Ciò fa in modo che, appena l'ADS ha ricevuto i dati, li invia al microcontrollore.

- **IDLE:** Il microcontrollore entra poi nello stato di IDLE in attesa di qualche evento esterno che lo faccia cambiare di stato. Quando sull'applicazione Android viene dato il comando di play (vedi paragrafo 3.3), il bluetooth riceve tale segnale che viene inviato tramite UART all'STM32. Ciò provoca una interruzione nel microcontrollore che setta $stream\_enable = 1$ facendo passare la macchina a stati dallo stato IDLE allo stato Stream.

- **Stream:** in questo stato il micro riceve tramite SPI i dati dall'ADS in continuous mode. I dati ricevuti devono poi essere preparati per la trasmissione. Tramite SPI vengono infatti ricevuti 8 bit alla volta, che vengono inseriti in un vettore formato da: header+3B di payload. I dati, così assemblati, vengono inviati tramite la porta UART dell'STM32 al bluetooth che a sua volta li invia all'utente. Ciò continua fino all'insorgere di un ulteriore interrupt, generato dalla ricezione via bluetooth del comando di stop da parte dell'applicazione, che setta la variabile $stream\_enable = 0$. Durante il funzionamento in modalità online, questo stato verrà modificato in modo da permettere al micro di eseguire anche la parte di preprocessing e classificazione.

3.3 Applicazione mobile

La scheda acquisisce il segnale EMG degli elettrodi e lo invia tramite Bluetooth ad uno smartphone nel quale è stata preinstallata una applicazione mobile realizzata in laboratorio [26]. Grazie a tale applicazione è possibile vedere in tempo reale lo stream dei dati acquisiti (in modo da valutare direttamente la bontà del segnale) e salvarli in un file di testo, utile per una elaborazione offline.

3.3.1 Parti principali dell'applicazione

Una volta avviata l'applicazione, nel menù principale viene visualizzato l'elenco dei dispositivi bluetooth accoppiati. Scelto il dispositivo corretto, viene chiesta la lunghezza del buffer ed il numero di canali.

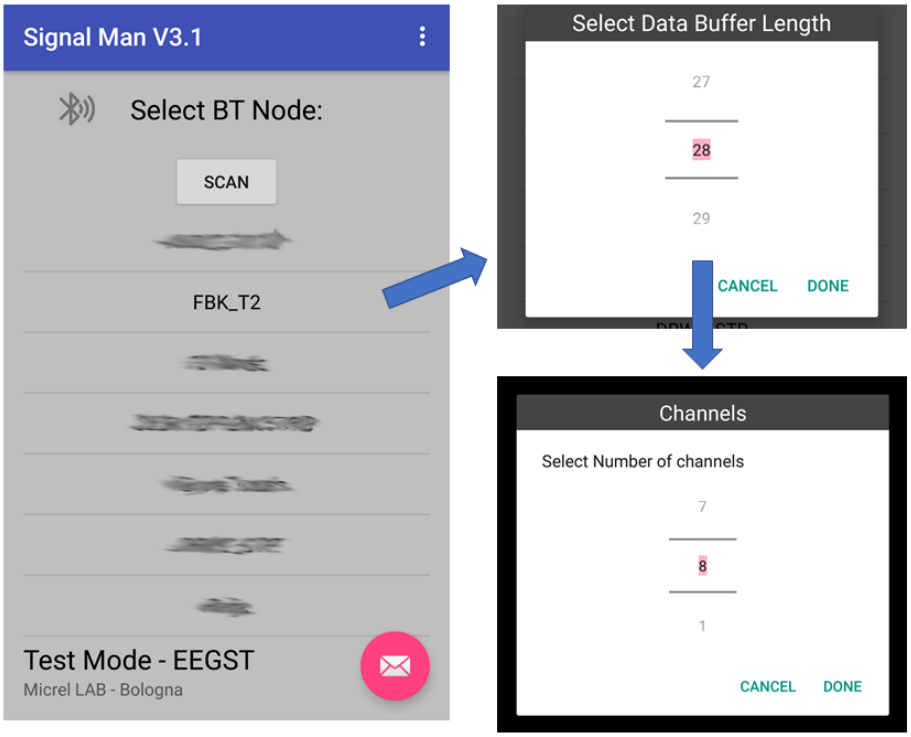

Figura 3.9: Menu iniziale applicazione

Una volta settati i parametri, ci si trova nella schermata di visualizzazione (Figure 3.10). Tramite il tasto setting è possibile impostare alcuni parametri

per una migliore visualizzazione dei segnali, come gli estremi di ascissa e ordinata, lo scaling ed il bias di ogni canale.

Premendo il tasto play viene inviato tramite bluetooth il comando al micro di acquisire ed inviare i dati, che vengono ricevuti e visualizzati dall'applicazione finchè non viene premuto il tasto di pausa. I dati acquisiti vengono salvati solo se viene premuto il tasto recording. Una volta premuto, tutti i dati successivamente ricevuti vengono salvati in un documento di testo, fino all premuta del tasto di stop. Tramite il tasto FFT è inoltre possibile visualizzare la Fast Fourier Transform del segnale.

É stato anche inserito il valore corrispondente alla classificazione SVM. Nell'implementazione online infatti sarà il micro ad eseguire la classificazione dei gesti ed inviarla all'applicazione (Capitolo 5). In questo modo è possibile visualizzare in tempo reale se la classe di appartenenza è quella attesa o meno.

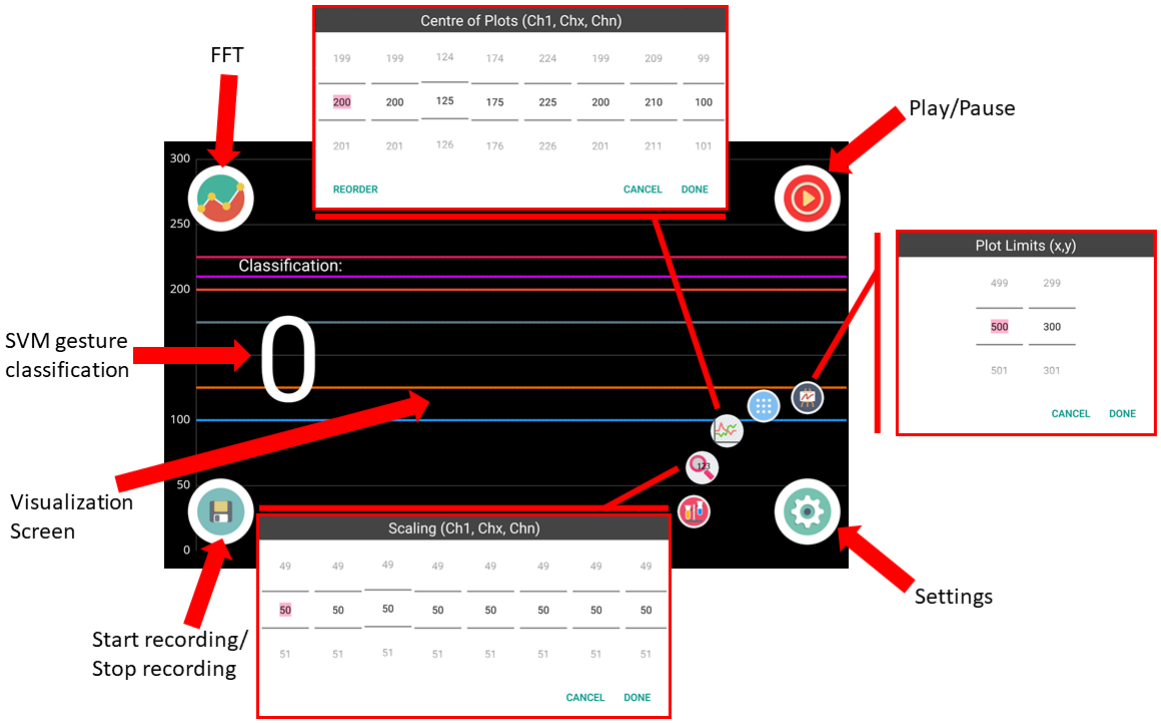

Figura 3.10: Funzionalità applicazione

Ci sono poi altre caratteristiche che sono state necessarie per la realizzazione dell'applicazione:

- Creazione client Bluetooth per la connessione con la scheda EMG;

- Ricezione dello stream di dati dalla scheda con un data rate di 500 letture al secondo;
- Visualizzazione dei canali ricevuti sotto forma di onda;
- Far scorrere il plot dando spazio ai nuovi valori ricevuti;
- Salvataggio dei dati ricevuti in un file per una analisi futura;
- L'applicazione può funzionare su ogni dispositivo su cui è installato Android 4.2 o maggiore.

3.3.2 Ricezione dal microcontrollore

Per gestire la porta seriale di un sistema Android, l'applicazione inizialmente controlla la disponibilità del modulo bluetooth. Una volta che il sistema ne garantisce la disponibilità, l'applicazione lo abilita e inizia a cercare i vari dispositivi Bluetooth attivi nelle vicinanze. Una volta selezionato il dispositivo a cui connettersi, il sistema inizializza una connessione client con la scheda di acquisizione usando la modalità SPP (Serial Port Profile). La SPP simula una tipica connessione seriale con cavo.

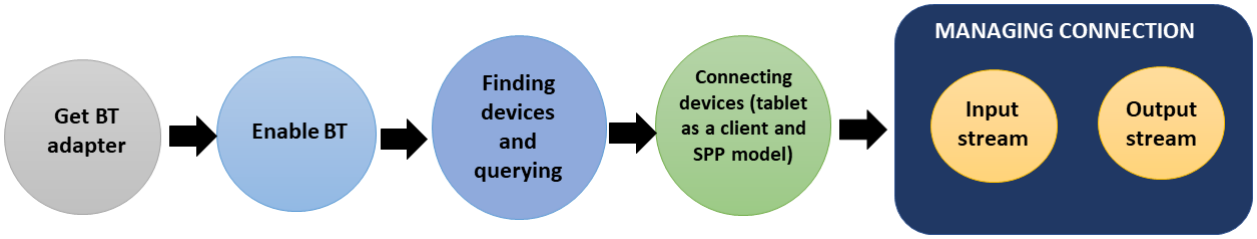

Figura 3.11: Schema a blocchi per la connessione Bluetooth

Una volta che la connessione è stata stabilita, l'applicazione è pronta a ricevere i dati dalla scheda. La ricezione dei dati è eseguita in un thread diverso dagli altri task in modo da garantire una risposta real time. Il thread principale si occupa del plotting dei dati ricevuti, del salvataggio e del cambiare i parametri per la visualizzazione. Il secondo thread invece controlla la ricezione e la lettura dei dati. I caratteri vengono ricevuti uno alla volta e vengono poi assemblati per ottenere un intero con segno.

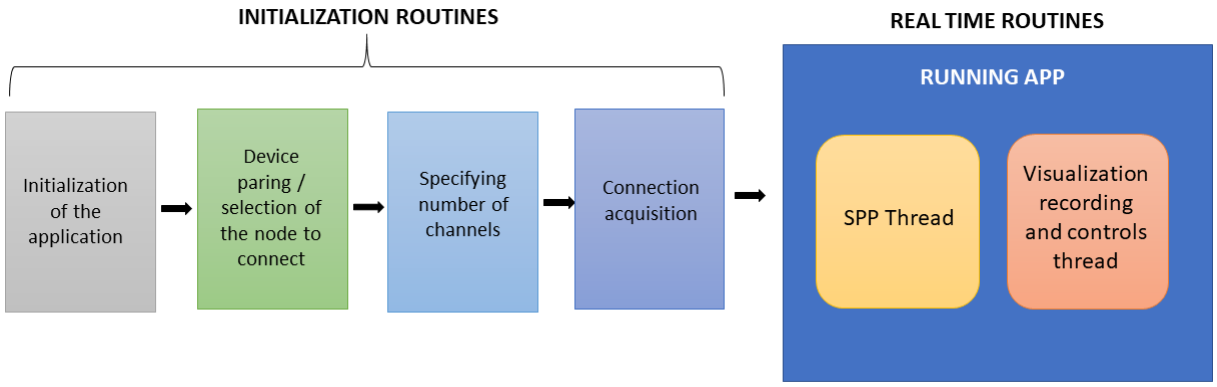

Figura 3.12: Diagramma di flusso applicazione

3.4 Elaborazione dei dati su MatLab

I dati salvati nel telefono vengono successivamente caricati su MatLab e poi elaborati. I dati in ingresso vengono rappresentati sotto forma di una matrice con numero di colonne pari al numero di canali e numero di righe pari al numero di campioni.

I parametri iniziali da settare sono:

- **Numero di canali:** indica il numero di canali differenziali che si vogliono tenere in considerazione. Nella prima parte di analisi viene tenuto fisso a 7, ovvero il numero massimo di canali presenti sul polsino. Successivamente si andrà ad agire su tale parametro per vedere fino a quanto possiamo ridurlo.

- **Numero di gesti:** è il numero totale di gesti da analizzare, ovvero il numero di classi su cui opererà la SVM. In questo valore è quindi da tenere conto anche il gesto di riposo.

- **Tipo di feature da utilizzare:** lo script analizza diverse tipologie di preprocessing, che sono RMS, DWT, WL, MMNF.

- **Altri parametri:** come ad esempio il downsamples o la lunghezza della finestra di elaborazione.

Una corretta configurazione di tali parametri è di fondamentale importanza in quanto determinano le caratteristiche del sistema, come il consumo, la complessità del modello e l'accuratezza in uscita. Lo script è suddiviso in due parti principali: training e testing.

3.4.1 Training

Una volta scelte le configurazioni opportune, i dati relativi al segnale EMG subiscono una fase di preprocessing e vengono poi utilizzati come training set per creare il modello della SVM.

Figura 3.13: Schema a blocchi training

Preprocessing

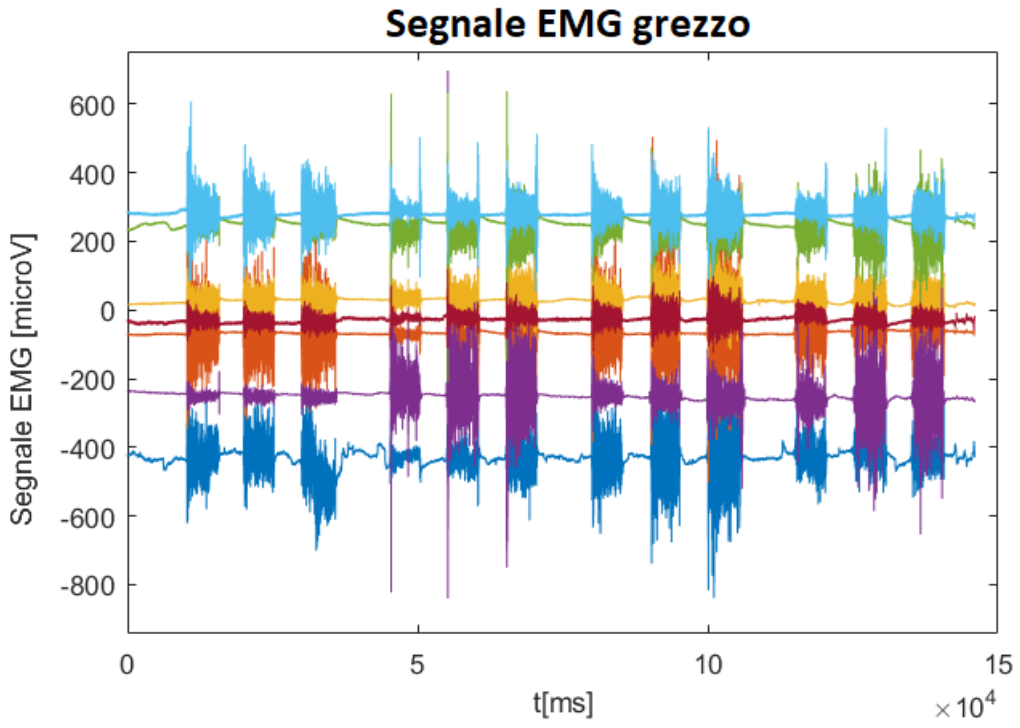

Figura 3.14: Dati EMG grezzi in ingresso

Come prima cosa, il segnale di ingresso viene filtrato tramite un filtro notch digitale a 50Hz. Tale passaggio è utile per eliminare la componente a

50Hz derivante dalla frequenza di rete che potrebbe portare energia e rumore e quindi alterare la misura.

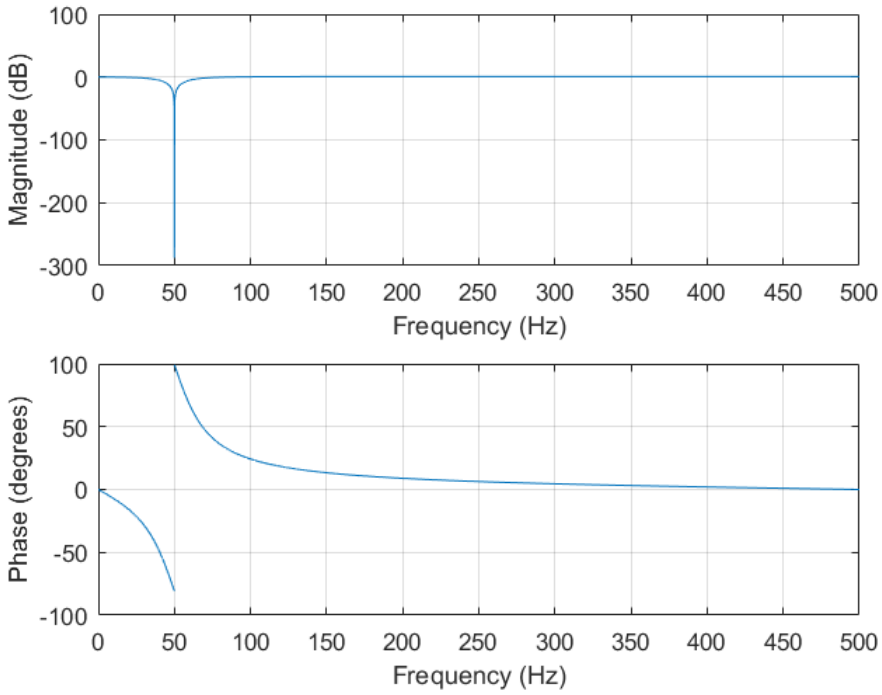

Figura 3.15: Funzione di trasferimento filtro notch 50Hz

Su questi dati viene eseguita l'operazione di estrazione della feature, utile per ricavare dal segnale le informazioni necessarie alla sua classificazione. Tale operazione viene eseguita su ogni canale in ingresso. Le feature analizzate sono:

- **RMS:** viene calcolato su una finestra mobile di 60 campioni che shifta di 1 ad ogni iterazione;
- **DWT:** La finestra di analisi è di 256 campioni a 4 livelli. Ogni canale in ingresso produrrà quindi 4 features in uscita;
- **WL:** viene calcolato su una finestra mobile di 60 campioni che shifta di 1 ad ogni iterazione;
- **MMNF:** viene calcolato su una finestra mobile di 60 campioni che shifta di 1 ad ogni iterazione.

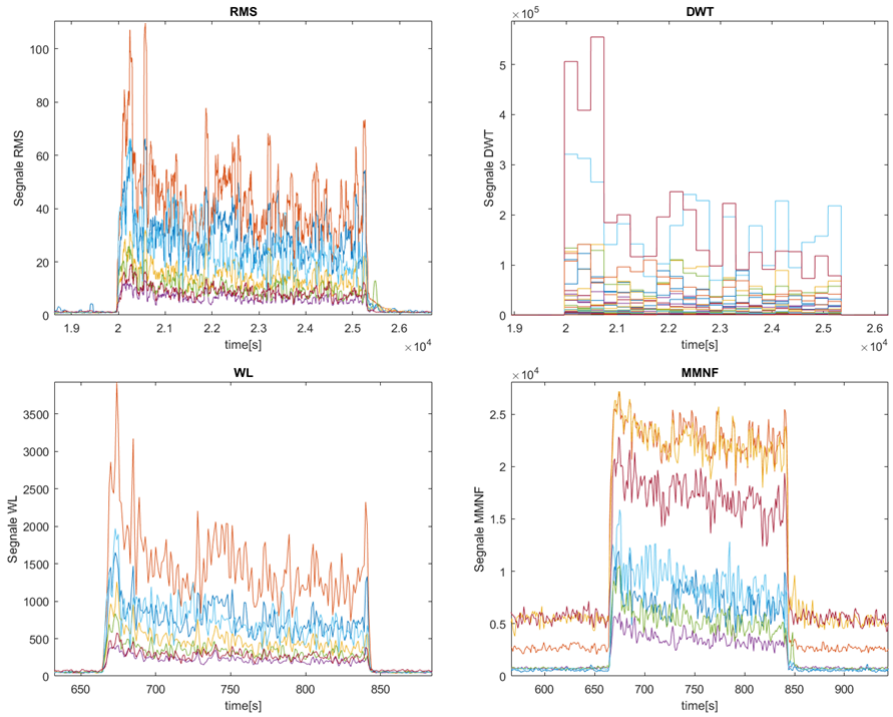

Figura 3.16: Features analizzate

Creazione modello

Si considera ora il segnale processato. Per ogni gesto sono state fatte n ripetizioni. É possibile decidere di usarne alcune come dati di training e testare le altre, oppure anche prenderle tutte come training e come testing caricare un nuovo dataset.

Per la creazione del training-set, lo script va a selezionare un intervallo del gesto in cui il segnale si è stabilizzato, escludendo il transitorio. Una volta che tutti gli intervalli relativi a quel determinato gesto sono stati selezionati, vi viene assegnata una label relativa alla classe di appartenenza. Tale operazione viene ripetuta per ogni gesto che si vuole utilizzare per il training, creando quindi 2 vettori: uno corrispondente alle gestures sotto analisi ed uno corrispondente alle relative labels (Figura 3.17).

I due vettori così creati vengono dati in ingresso alla funzione di training della SVM che restituisce in uscita il modello relativo.

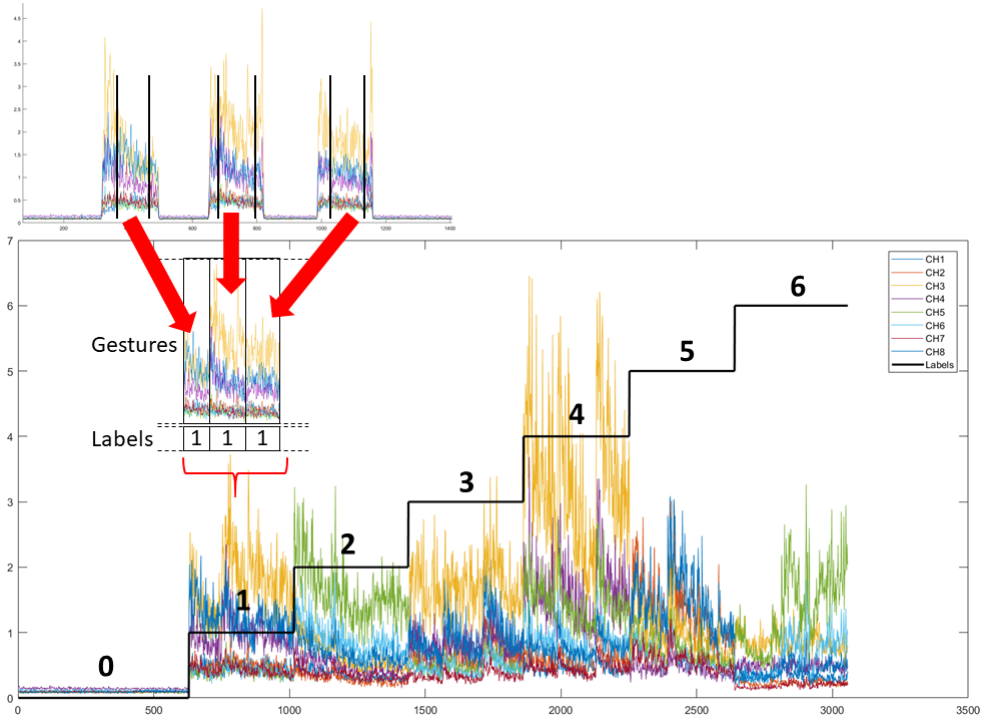

Figura 3.17: Preparazione del training set per la creazione modello per SVM

Scaling

Per un corretto funzionamento della SVM, i dati in ingresso devono essere compresi in un intervallo di valori che va da 0 a 10 [27]. Per tale motivo è necessaria una operazione di scaling che porta i valori in quel range.

Dato che non si conosceva a priori il valore di scaling ottimale per ogni feature, sono state realizzate varie prove utilizzando un valore di scaling variabile. Il valore di scaling è dato da:

$$SCALING = \frac{1}{traininset} \cdot \alpha \tag{3.1}$$

con α variabile. Questo valore andrà poi a moltiplicare il training-set precedentemente creato in modo da portarlo nell'intervallo compreso tra 1 e α.

Una volta impostando tale valore come scaling, viene calcolato il modello, che sarà diverso da scaling a scaling, così come il numero di support vector. Il modello ottenuto viene poi utilizzato nella parte di testing per classificare il dataset di ingresso. Tale operazione viene ripetuta per diversi valori di α ottenendo risultati diversi di accuratezza.

3.4.2 Testing

In questa fase, operando offline, viene caricato un nuovo dataset che si vuole valutare. Vengono eseguite anche su di lui le stesse operazione di preprocessing (filtro e feature) eseguite al training set. I dati vengono scalati con lo stesso fattore di scaling utilizzato per il training.

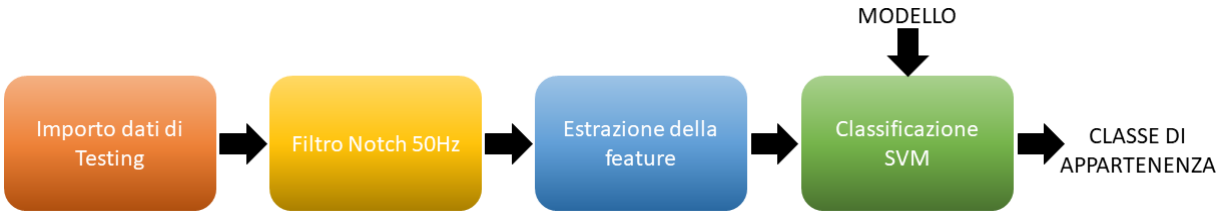

Figura 3.18: Schema a blocchi testing

Calcolo accuratezza

I dati, dopo essere stati processati, vengono dati in ingresso alla SVM che, in funzione del modello precedentemente creato, restituisce in uscita la sua predizione.

Per poter sapere quanto vale l'accuratezza della classe assegnata è necessario confrontare i dati in uscita dalla SVM con quelli attesi. Ciò è possibile proprio perchè, lavorando in modalità offline, si è già a conoscenza dei gesti che sono stati eseguiti. Per tale misura è quindi sufficiente creare una maschera contenente i valori attesi e confrontarla con i valori predetti in uscita dalla SVM.

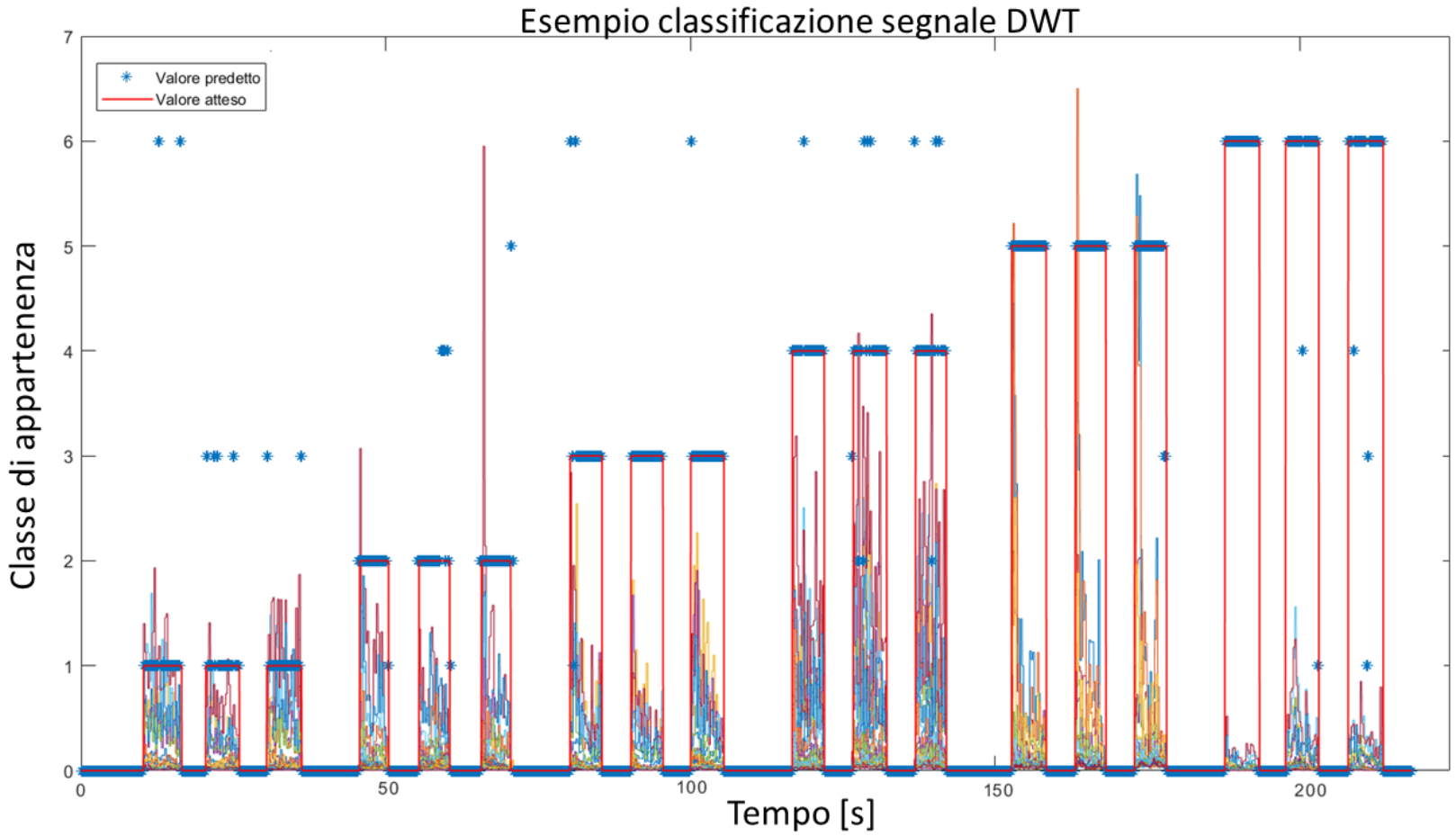

Figura 3.19: Esempio di classificazione di un segnale EMG comprendente 7 classi e processato tramite DWT

Capitolo 4

Analisi Off-line

Una volta definita e realizzata l'intera architettura, si è passati alla parte sperimentale vera e propria in cui sono state eseguite diverse prove utili per comprendere quali strategie adottare in termini di: tecniche di preprocessing, tecniche di training, rimozione gesti e rimozione canali. Tutte le prove seguenti sono state realizzate con dati reali acquisiti tramite l'architettura descritta nei capitoli precedenti che sono poi stati elaborati in modalità offline grazie all'utilizzo di MatLab. In seguito verranno presentate le varie prove ed i risultati ottenuti.

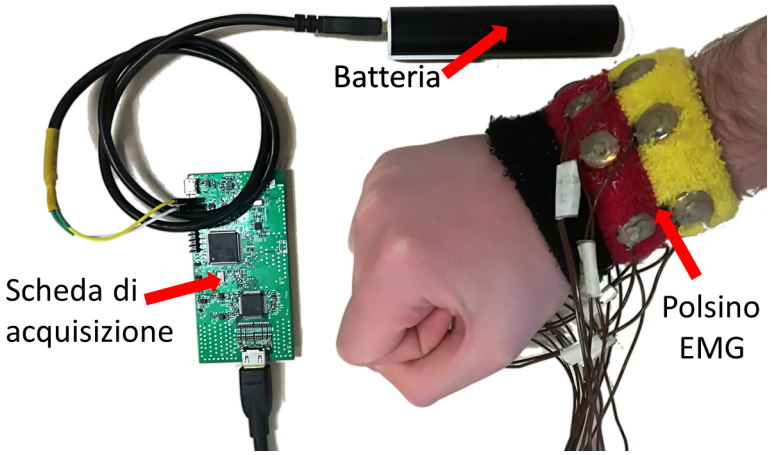

Figura 4.1: Foto del sistema relativo alle prove offline, comprendente scheda, batteria e polsino

4.1 Definizione della prova

4.1.1 Scelta gesti e features

Il primo step del progetto è stato quello di definire le modalità di esecuzione della prova, in particolare quali gesti realizzare e quali tecniche di preprocessing utilizzare.

A tale scopo sono stati scelti 10 gesti: riposo, mano aperta, pugno, indice, pinch, ok, flessione polso indietro, flessione polso chiuso, flessione verso mignolo, flessione verso pollice. Ad ognuno di essi è stata poi associata una classe di appartenenza, come mostrato in Figura 4.2.

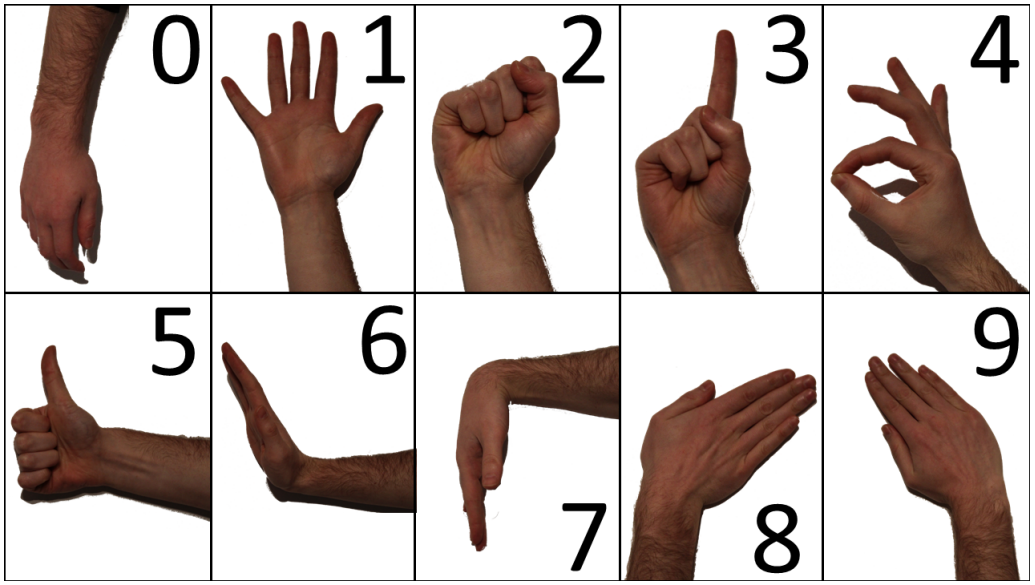

Figura 4.2: Gesti eseguiti e relative classi di appartenenza: riposo (0), mano aperta (1), pugno (2), indice (3), pinch (4), ok (5), flessione polso indietro (6), flessione polso chiuso (7), flessione verso mignolo (8), flessione verso pollice (9)

Il polsino è stato indossato nella mano destra ed è stata poi eseguita un'unica sessione di acquisizione su 5 utenti diversi a cui è stato chiesto di ripetere per 5 volte ogni gesto e di mantenerlo per 5 secondi. Ogni ripetizione è intervallata da altri 5 secondi di riposo, mentre tra un gesto ed il successivo viene mantenuto lo stato di riposo per 10 secondi.

I segnali sono stati caricati su Matlab in cui hanno subito un'azione di preprocessing con diverse metodologie. Quelle che sono state scelte sono: RMS e DWT (ampiamente utilizzate in letteratura), WL (per il basso costo computazionale), MMNF (per avere un'analisi in frequenza). Tali tecniche vengono attuate su ciascuno dei 7 canali in esame.

Per ogni gesto vengono scelte 2 ripetizioni da usare come training dell'algoritmo SVM, tramite le quali viene creato il modello. Le restanti 3 ripetizioni vengono utilizzate come testing. L'algoritmo, tramite il modello creato nella fase di training, fornisce in uscita la classe di appartenenza stimata che viene poi confrontata con quella attesa per ottenere l'accuratezza. Questa operazione è stata ripetuta per 5 soggetti diversi e per ognuna delle tecniche di preprocessing scelte. In Tabella 4.1 vengono mostrati i dati relativi alle percentuali di precisione di ogni gesto (mediate tra i vari soggetti) per ogni tipo di feature e la precisione media per ogni tipo di processing. Viene inoltre riportato il numero di Support Vector che massimizza la precisione media.

Si osserva come l'utilizzo di RMS o DWT o WL porti ad una precisione

	Gesto	**RMS**	**DWT**	**WL**	**MMNF**
0	Riposo	91	96	97	84
1	Mano aperta	95	97	98	77
2	Pugno	95	93	99	96
3	Indice	94	94	95	73
4	Pinch	88	91	93	94
5	Ok	97	94	99	95
6	Flex polso indietro	81	86	96	70
7	Flex polso chiuso	82	88	94	58
8	Flex verso mignolo	90	89	98	89
9	Flex verso pollice	91	87	97	36
	Precisione media	90	91	96	77
	nSV per PrecMax	297	320	140	29

Tabella 4.1: Precisione di ogni gesto, mediata su 5 soggetti differenti, per diverse tipologie di preprocessing

media nell'intorno del 90%. La DWT risulta essere la feature che utilizza più support vector, mentre quella che ne utilizza meno è la MMNF che però è la peggiore da un punto di vista dell'accuratezza, infatti alcuni gesti vengono completamente sbagliati (ad esempio la classe 9 ha una percentuale di accuratezza solo del 36%). Tra tutte spicca la WL, che riesce ad arrivare a valori di accuratezza molto alti con un numero di support vector non eccessivo.

4.1.2 Rotazione polsino

Se si pensa ad una sua implementazione ad esempio in un orologio, durante il suo utilizzo il polsino può ruotare e difficilmente resta fisso nella stessa posizione per molto a lungo. Per simulare questo comportamento e per vedere quali sono gli effetti che ne derivano sono state eseguite nuove sessioni di lavoro in cui il polsino viene ruotato rispetto alla posizione iniziale. In particolare (Figura 4.3):

- **centrale (c):** il polsino, una volta tolto, viene riposizionato nella posizione centrale;
- **rotazione verso destra (r):** guardando la mano col dorso rivolto verso l'alto, il polsino viene ruotato verso destra;
- **rotazione verso sinistra (l):** guardando la mano col dorso rivolto verso l'alto, il polsino viene ruotato verso sinistra.

L'ampiezza della rotazione è di mezzo passo tra un elettrodo e l'altro. Ogni posizione diversa è stata acquisita in una sessione differente da quella di training.

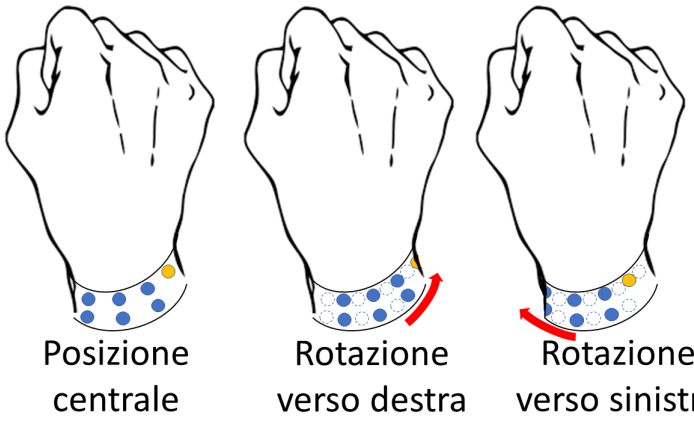

Figura 4.3: Rotazioni eseguite durante la fase di testing

4.1.3 Metodologie di training

L'elettrodo di riferimento è fissato sul polsino. Questo significa che, nonostante ogni volta si cerchi di posizionarlo sempre nella posizione corretta, la sua posizione sarà diversa da una sessione all'altra a seconda di come si indossa il polsino. Inoltre una rotazione del polsino in senso orario o antiorario provoca uno spostamento in tale direzione da parte di tutti gli elettrodi, e quindi anche di quello utilizzato come riferimento. Per ottenere una migliore accuratezza nella stima della classe di appartenenza di uscita, è necessario creare un buon modello. Grazie ad un buon modello, la stima della SVM sarà buona nonostante le variazioni di misura del segnale o eventuali spostamenti del polsino. Vengono quindi confrontate due diverse metodologie di training:

- **Training singolo:** il polsino viene indossato nella posizione centrale, con l'elettrodo di riferimento posto sopra l'epifisi dell'ulna. Il modello viene creato eseguendo il training solamente in tale posizione e viene poi utilizzato in fase di testing per classificare, oltre alla posizione centrale, anche la rotazione verso destra e quella verso sinistra (Figura 4.4).

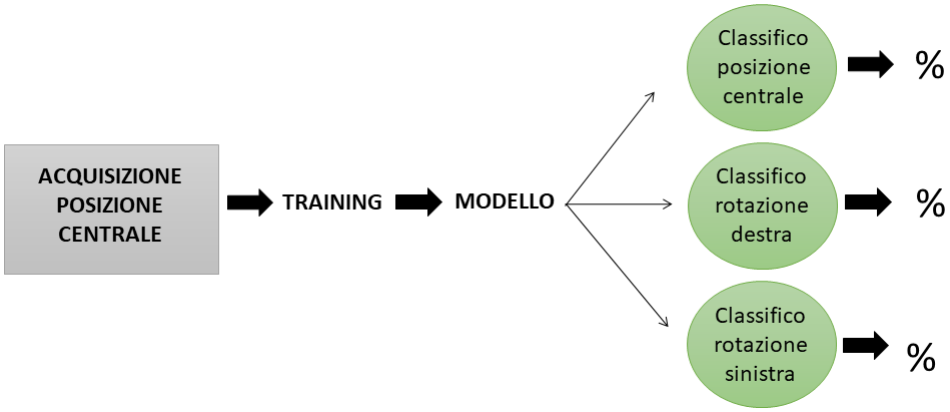

Figura 4.4: Schema training singolo

- **Training multiplo:** il modello viene creato utilizzando, oltre ai dati acquisiti nella posizione centrale, anche quelli relativi alla rotazione in senso orario e antiorario del polsino. Tali dati vengono poi concatenati insieme creando un unico modello (Training multiplo concatenato, Figura 4.5) che viene usato dalla SVM per classificare le varie posizioni, oppure vengono tenuti separati creando tre modelli distinti e per la classificazione sarà utilizzato quello relativo alla posizione di interesse. (Training multiplo Ad-Hoc, Figura 4.6).

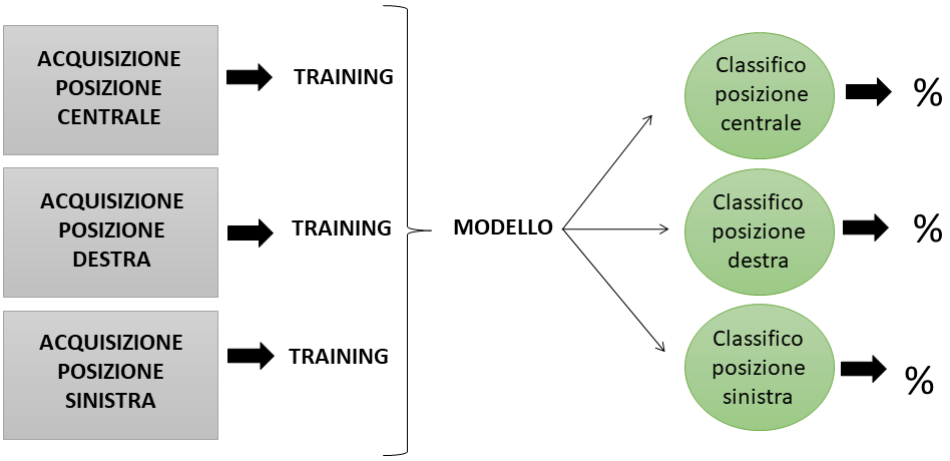

Figura 4.5: Schema training multiplo concatenato

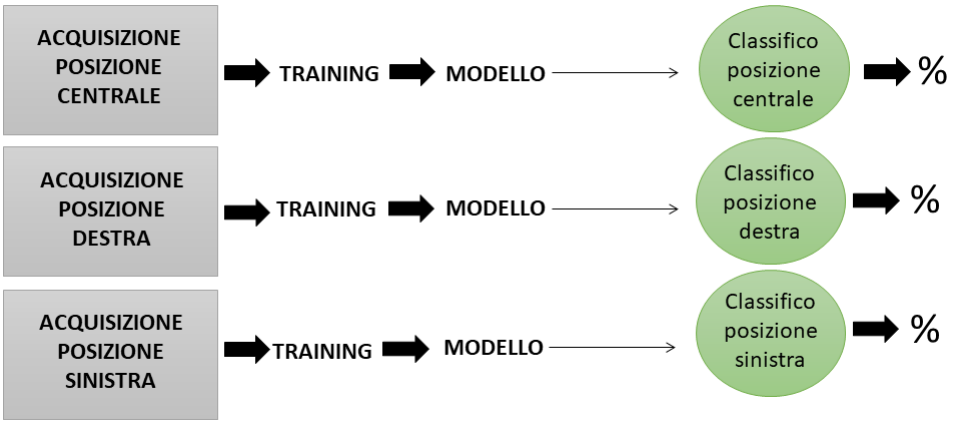

Figura 4.6: Schema training multiplo Ad-Hoc

4.2 Training singolo

Vengono acquisiti i dati da 5 soggetti differenti, indossando il polsino nella posizione centrale. I gesti ripetuti sono gli stessi descritti in Figura 4.2. Per ogni gesto sono state eseguite 3 ripetizioni, che andranno così a formare il modello per la SVM. Per la fase di testing la procedura di acquisizione è stata poi ripetuta togliendo il polsino e rimettendolo nella posizione centrale (c), ruotato verso destra (r), ruotato verso sinistra (l). Questi 3 testing-set vengono poi processati utilizzando 4 diverse features: RMS, DWT, WL, MMNF. Successivamente vengono classificati tramite SVM utilizzando il modello precedentemente calcolato. I valori relativi alle percentuali di accuratezza delle predizioni vengono riportati in tabella 4.2. Osservando la colonna relativa al testing nella posizione centrale e confrontandola con la Tabella 4.1, si nota che togliendo e rimettendo il polsino la precisione media cala. Inoltre, avendo fatto il training solo nella posizione centrale, appena il polsino ruota un po' a sinistra o a destra, la precisione media cala (r ed l sempre minori di c) rendendo quindi il dispositivo session dependent. Questo è un problema perchè nella vita di tutti i giorni il polsino si comporta proprio in questa maniera, ed è quasi impossibile che resti sempre fisso o che venga indossato sempre nella stessa posizione.

Classe di	**RMS**			**DWT**			**WL**			**MMNF**		
appartenenza	c	r	l	c	r	l	c	r	l	c	r	l
0	96	97	96	98	99	98	98	99	98	92	97	92
1	96	87	91	95	77	62	98	85	79	30	48	22
2	60	1	58	37	0	29	60	1	64	20	35	12
3	93	98	91	80	94	98	86	95	92	91	88	93
4	67	91	54	96	86	82	42	94	41	58	40	638
5	97	97	100	95	100	95	93	88	99	99	98	99
6	92	71	6	100	79	12	97	88	7	82	47	0
7	59	4	85	78	20	94	87	12	87	73	72	84
8	97	73	84	76	98	92	93	78	91	93	62	86
9	89	42	89	98	73	100	97	85	97	65	64	28
Precisione media	84	66	75	85	73	76	85	73	75	70	65	55
nSV	244	581	451	450	746	291	364	608	410	36	35	36

Tabella 4.2: Precisione percentuale per ogni feature di ogni gesto, mediate su 5 soggetti, nel caso di training singolo

4.3 Training multiplo

Per cercare di rendere il dispositivo session independent si è pensato di provare a fare il training in più posizioni. Mentre per il training singolo si avevano 4 dataset (1 training + 3 testing), qui se ne hanno 6. L'acquisizione dei dati utilizzati per il training è infatti ripetuta per 3 volte, ognuna delle quali mettendo il polsino in una posizione diversa rispetto al caso precedente. L'acquisizione dei dati per la fase di testing è invece la stessa descritta nel caso del training singolo. In seguito vengono riportati i valori ottenuti per il caso di training concatenato ed ad-hoc.

4.3.1 Training multiplo concatenato

Classe di	**RMS**			**DWT**			**WL**			**MMNF**		
appartenenza	c	r	l	c	r	l	c	r	l	c	r	l
0	95	94	90	98	98	97	98	97	97	94	97	96
1	98	96	82	95	97	93	99	99	87	91	63	72
2	90	79	84	82	88	88	88	85	87	72	75	84
3	90	91	81	93	95	94	88	90	88	91	93	92
4	39	39	35	84	87	46	38	79	33	44	43	45
5	95	80	98	97	98	84	91	93	98	94	94	98
6	95	82	21	97	83	14	98	98	13	58	67	4
7	88	7	78	86	11	90	87	11	90	96	72	96
8	91	86	96	95	98	97	91	81	97	94	73	86
9	93	81	80	97	94	91	97	94	91	70	48	31
Precisione media	87	73	74	92	85	79	88	83	78	80	72	70
nSV	823	772	899	1069	1129	1129	761	805	805	104	99	98

Tabella 4.3: Precisione percentuale per ogni feature di ogni gesto, mediate su 5 soggetti, nel caso di training multiplo concatenato

Confrontando la Tabella 4.2 con la Tabella 4.3 si nota che col training multiplo concatenato la precisione media aumenta rispetto al training singolo. Questo accade perchè il modello della SVM ora è più completo essendo a conoscenza anche delle altre posizioni. Si nota inoltre un significativo aumento del numero dei support vector e quindi un modello molto più complicato (soprattutto per quanto riguarda la DWT).

4.3.2 Training multiplo Ad-Hoc

Classe di	**RMS**			**DWT**			**WL**			**MMNF**		
appartenenza	c	r	l	c	r	l	c	r	l	c	r	l
0	97	94	96	98	98	97	97	97	98	79	90	91
1	97	99	90	98	100	83	94	99	82	46	1	75
2	91	81	59	59	87	49	70	66	51	45	47	56
3	97	77	93	86	96	100	81	83	93	65	99	75
4	72	59	37	91	70	26	44	62	71	58	44	32
5	95	79	100	90	100	87	89	95	100	96	93	95
6	94	86	21	100	83	37	98	94	16	48	71	0
7	78	14	96	77	23	83	82	22	92	98	96	98
8	86	35	79	80	89	99	95	74	92	95	39	92
9	91	77	91	100	100	71	96	97	91	64	20	56
Precisione media	90	70	76	88	85	73	85	79	79	70	60	67
nSV	240	276	270	437	374	510	222	239	204	40	38	41

Tabella 4.4: Precisione percentuale per ogni feature di ogni gesto, mediate su 5 soggetti, nel caso di training multiplo Ad-Hoc

Questo tipo di training è sostanzialmente un training singolo fatto però su diverse posizioni del polsino. Confrontando infatti la Tabella 4.4 con la Tabella 4.2 si nota che la prima colonna è sostanzialmente la stessa (salvo pochi punti percentuali dovuti al fatto che i dati di partenza non sono gli stessi). Confrontando le colonne relative alle rotazioni della Tabella 4.4 con la Tabella 4.3 si nota che i valori di accuratezza sono confrontabili. Il numero di Support Vector è migliore rispetto al caso precedente dato che modello utilizzato per classificare una certa posizione utilizza i dati di training relativi solo a quella posizione e non a tutte come nel caso del training concatenato.

4.4 Confronto tra le prove

Per confrontare tutte le varie prove e capire quale tipologia di training è quella migliore (e quindi quella da adottare), vengono riportate in tabella le precisioni medie di ogni feature in ogni posizione per tutte le configurazioni.

	Tr. Singolo			**Tr. Conc.**			**Tr. Ad-Hoc**		
	c	r	l	c	r	l	c	r	l
RMS	84	66	75	87	73	70	90	70	76
DWT	85	73	76	92	85	79	88	85	73
WL	85	73	75	88	83	74	85	79	79
MMNF	70	65	55	80	72	70	70	60	67

Tabella 4.5: Confronto prove

Considerando i dati rappresentati in Tabella 4.5 si evince che per la fase di training è preferibile acquisire i dati relativi a diverse rotazioni del polsino in modo da fornire al classificatore SVM una maggiore conoscenza della variabilità della misura e consentire quindi una migliore predizione. Le prove successive di riduzione di gesti e canali vengono quindi realizzate effettuando il training multiplo. In particolare nell'implementazione real-time è stato utilizzato il training multiplo concatenato. Questo perchè il training multiplo Ad-Hoc necessita di una fase iniziale in cui il micro deve andare a scegliere il modello giusto da utilizzare per la classificazione. Ciò è difficile da implementare in real time in quanto una scelta errata del modello porterebbe ad un notevole peggioramento dell'accuratezza. Porta inoltre a costi computazionali aggiuntivi che potrebbero non compensare il minor numero di support vector utilizzati.

4.5 Riduzione dei gesti

Inizialmente, non sapendo quali gesti venissero classificati meglio, si era partiti eseguendo le prove su 10 classi. Dalle analisi precedenti si può osservare che, anche effettuando il training concatenato, non tutti i gesti vengono classificati in maniera ottimale e si è quindi deciso di rimuoverli. In particolare vengono rimossi: pinch, flessione polso indietro e flessione poso chiuso. Se si pensa ad una implementazione di questa architettura per il controllo di un dispositivo come uno smartwatch ciò non è limitativo in quanto è facilmente controllabile anche con meno gesti. Inoltre è in linea col numero di gesti analizzati in letteratura [16]. I gesti considerati in questa prova sono quelli rappresentati in Figura 4.7.

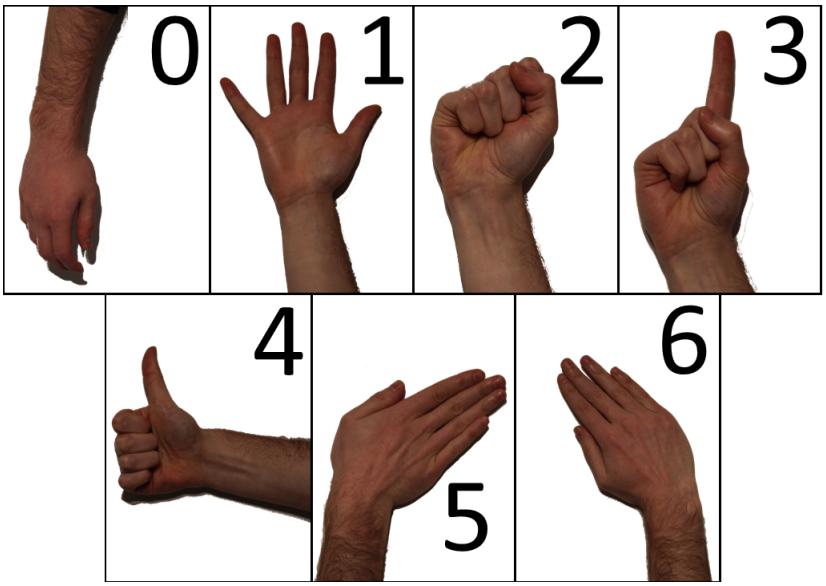

Figura 4.7: Gesti eseguiti e relative classi di appartenenza: riposo (0), mano aperta (1), pugno (2), indice (3), ok (4), flessione verso mignolo (5), flessione verso pollice (6)

Durante la fase di training vengono acquisite 3 ripetizioni per ogni gesto (come nei casi precedenti) per tutte e tre le rotazioni di interesse adottando quindi la topologia di training multiplo. Le percentuali relative all'accuratezza della classe di appartenenza del gesto sono riportate in Tabella 4.6 e Tabella 4.7. Confrontandoli con i dati in Tabella 4.3 e Tabella 4.4 si ottiene un miglioramento nella precisione media. Anche il modello risulta essere

meno complesso avendo meno gesti da discriminare e ciò comporta quindi l'utilizzo di un numero di Support Vector minore.

Classe di	**RMS**			**DWT**			**WL**			**MMNF**		
appartenenza	c	r	l	c	r	l	c	r	l	c	r	l
0	97	95	93	99	99	98	99	99	97	97	97	97
1	99	99	81	98	100	95	99	96	78	93	95	84
2	90	93	85	92	100	93	86	89	85	88	97	95
3	94	96	86	100	100	96	89	95	91	92	65	88
4	96	88	99	99	100	100	90	97	99	94	97	100
5	100	100	99	100	100	99	99	100	100	100	97	97
6	93	47	58	98	95	88	99	94	85	84	72	24
Precisione media	96	88	86	98	99	96	94	96	91	93	89	84
nSV	347	315	389	687	718	687	329	635	362	58	60	45

Tabella 4.6: Precisione percentuale per ogni feature di ogni gesto, mediate su 5 soggetti, nel caso di 7 gesti e training multiplo concatenato

Classe di	**RMS**			**DWT**			**WL**			**MMNF**		
appartenenza	c	r	l	c	r	l	c	r	l	c	r	l
0	99	97	98	99	99	99	99	98	99	87	88	99
1	98	100	92	100	94	81	98	94	92	26	13	77
2	91	99	96	73	96	98	68	98	97	68	95	84
3	95	97	93	88	100	100	78	91	90	75	49	77
4	90	87	100	90	100	100	90	95	97	93	98	96
5	100	90	92	100	98	100	96	81	92	98	96	94
6	96	95	91	100	100	80	95	99	93	69	22	49
Precisione media	96	95	95	93	98	94	89	94	94	74	66	82
nSV	109	137	120	205	197	265	123	145	91	28	23	26

Tabella 4.7: Precisione percentuale per ogni feature di ogni gesto, mediate su 5 soggetti, nel caso di 7 gesti e training multiplo Ad-Hoc

4.6 Riduzione dei canali

Il polsino realizzato presenta lungo la sua circonferenza 7 canali più l'elettrodo di riferimento. Questo significa che l'ADC dovrà acquisire 7 canali e su ognuno di essi il micro dovrà eseguire l'operazione di preprocessing. Ciò genera una feature per ognuno di essi nel caso di RMS, WL e MMNF e 4 nel caso della DWT (essendo a 4 livelli). Una riduzione del numero di canali si traduce quindi in meno dati da acquisire, minore elaborazione dei dati, meno features, ed in generale un minore consumo di potenza. L'obiettivo è quindi provare a vedere se con la SVM si ottiene un valore di accuratezza buono anche con meno canali.

Inizialmente viene fatta una classificazione dei gesti descritti nella section precedente utilizzando tutti i canali, in modo da avere un valore di partenza con cui confrontare i valori successivi, ottenendo i seguenti valori di precisione media:

- **RMS:** 95%
- **DWT:** 91%
- **WL:** 88%

Poi si procede togliendo un canale alla volta fino a rimanere con 4 canali. Se togliendo un canale l'accuratezza cambia di poco significa che quel canale non conta molto e può essere rimosso. Vengono provate tutte le permutazioni possibili, per ogni gesto e per le features ottenute con RMS, DWT e WL. Vengono inoltre testate tutte le varie rotazioni. Riportare tutti i risultati sarebbe stato ridondante e tedioso, per questo in Tabella 4.8 vengono mostrati solo i valori di accuratezza massimi ed il relativo numero di support vector.

numero canali rimossi		**RMS**	**DWT**	**WL**
1	Pmax	91	89	88
	nSV	137	316	133
2	Pmax	87	89	91
	nSV	155	341	133
3	Pmax	91	89	88
	nSV	219	346	129

Tabella 4.8: Precisione massima ottenibile rimuovendo 1,2,3 canali e relativo numero di support vector

Dalla Tabella 4.8 si osserva che anche se il numero di canali viene ridotto da 7 a 4, la classificazione è ancora buona. Oltre al valore di accuratezza, è importante anche il numero di support vector in quanto vanno ad incidere sulle performance durante l'implementazione online e anche sul memory footprint del micro. Per questo motivo il parametro di interesse risulta essere il rapporto tra l'accuratezza raggiunta ed il numero di support vector utilizzati dal modello (Figura 4.8). Il trade-off migliore si raggiunge utilizzando come feature la WL, la quale riesce ad avere un valore di accuratezza simile a quello ottenuto con la DWT ma utilizzando molti meno support vector. Per questo motivo è quella che è stata scelta per l'implementazione online.

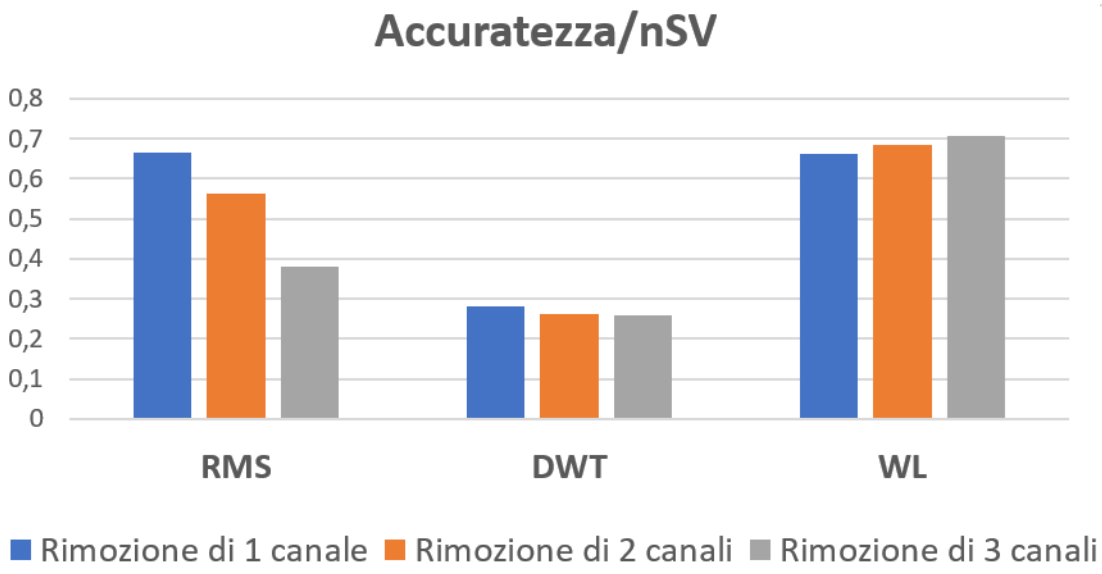

Figura 4.8: Trade off tra accuratezza e numero di support vector

Capitolo 5

Analisi Real Time

Figura 5.1: Schema del sistema durante il funzionamento online

Il segnale EMG viene rilevato grazie ad alcuni elettrodi disposti uniformemente lungo la circonferenza di un PCB flessibile indossato dall'utente.

Il segnale, corrispondente alle contrazioni muscolari relative ai movimenti della mano dell'utente, viene acquisito grazie ad un ADC con frequenza di campionamento pari a 1KHz che comunica con il microcontrollore STM32.

I dati acquisiti vengono poi elaborati direttamente sulla scheda dove subiscono una azione di preprocessing. Successivamente, tramite un modello precedentemente caricato nel microcontrollore, viene determinata la classe di appartenenza del gesto eseguito.

Grazie al modulo bluetooth la scheda invia al telefono i dati e la classe di appartenenza del gesto eseguito, così da poterlo visualizzare in tempo reale. É possibile anche salvare i dati in modo da essere caricati su MatLab per avere un valore numerico di accuratezza.

5.1 Prototipo nuovo

É stato realizzato un PCB di polimeri flessibile che funge da base di supporto per gli strati di rame. La parte interna del bracciale presenta piazzole circolari di rame su cui vengono posizionati elettrodi auto adesivi a base di gel gommoso. Sulla parte superiore vengono connessi i cavi.

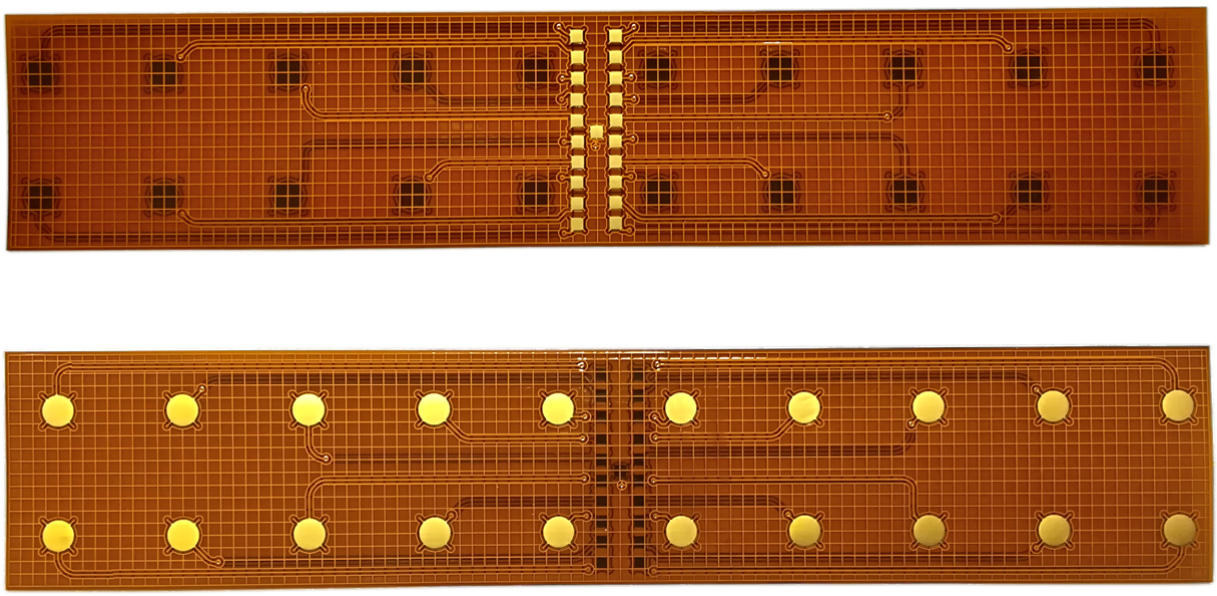

Figura 5.2: Fronte e retro del PCB flessibile del nuovo bracciale

Vengono scelti 4 gesti più riposo, che vengono ripetuti 3 volte per ognuna delle 3 posizioni descritte nei capitoli precedenti (Figura 5.3). Il segnale viene acquisito da 4 canali differenziali disposti lungo il bracciale.

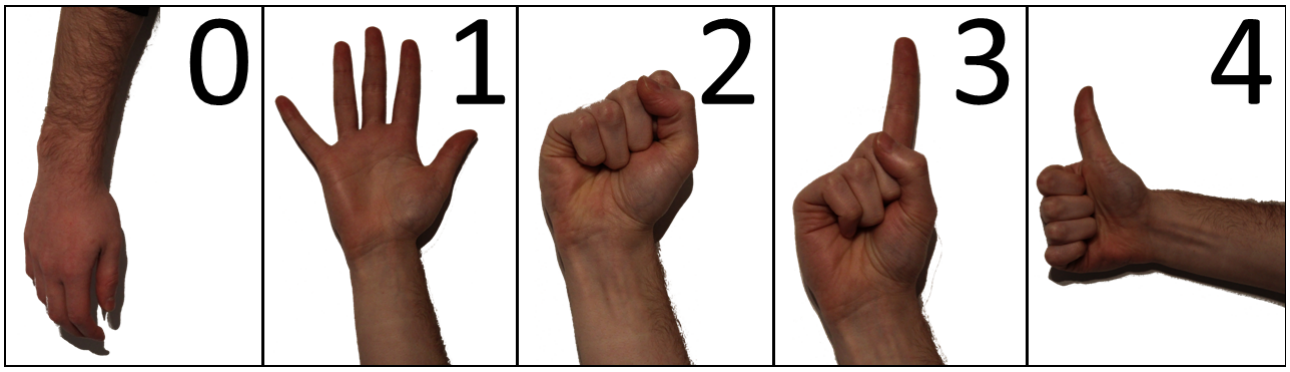

Figura 5.3: Gesti eseguiti nella prova online

5.2 Creazione modello MatLab

Il modello della SVM viene calcolato offline. Si acquisiscono i campioni relativi al segnale EMG dei gesti eseguiti e vengono poi salvati e importati in Matlab in cui subiscono l'azione di preprocessing. Confrontando le varie tabelle ottenute nell'analisi offline, si può osservare che le tecniche di preprocessing che portano ad un valore migliore di accuratezza sono la DWT e la WL. La WL ha come vantaggio un costo computazionale minore rispetto alle DWT, essendo solo somme di differenze. Inoltre il modello utilizza anche un numero minore di support vector. Per tale motivo, per l'implementazione online è stato scelto di utilizzare la WL come preprocessing. Il nuovo bracciale utilizzato è in grado di rilevare il segnale da 10 canali differenziali. Abbiamo però visto che ne sono sufficienti 4. La finestra della WL viene lasciata di 60 campioni, mentre invece l'overlap, che prima era di 1 campione, viene portato a 30. Questo perchè, durante il funzionamento in online mode, il micro riceve dall'ADC un nuovo campione ogni $1ms$. Mettere un overlap di un solo campione significa che ogni ms il micro deve avere finito tutto il preprocessing e la classificazione relativa al dato precedente altrimenti si ha una perdita di informazione. L'overlap di 30 campioni permette quindi al micro di eseguire tale operazione una volta ogni $30ms$, dandogli il tempo per finire le operazioni sui campioni precedenti. Dato quindi che sul micro la feature necessita di un overlap maggiore per essere calcolata, l'overlap è stato modificato anche in fase di training. Una volta eseguite le azioni di preprocessing sui dati importati, viene creato il modello come nel caso offline. La metodologia di training adottata per la creazione del modello è quella del training multiplo concatenato.

5.3 Modifiche firmware

Una volta creato il modello, i vari parametri vengono caricati nel firmware del microcontrollore. Dato che ora tutta la parte di preprocessing e classificazione viene eseguita dal micro, sono state apportate alcune modifiche al codice.

La prima parte della FSM rimane invariata (Figura 3.8). Ora però, una volta ricevuti i dati, occorre eseguire l'operazione di preprocessing e classificazione. Per tale motivo lo stato di Stream è stato modificato come in Figura 5.4.

Le matrici su cui vengono salvati i dati ricevuti sono A e B, entrambe formate da 8 righe (una per ogni canale) e 30 colonne. Ogni dato ricevuto è un vettore colonna di 8 elementi, dove ogni elemento rappresenta il valore

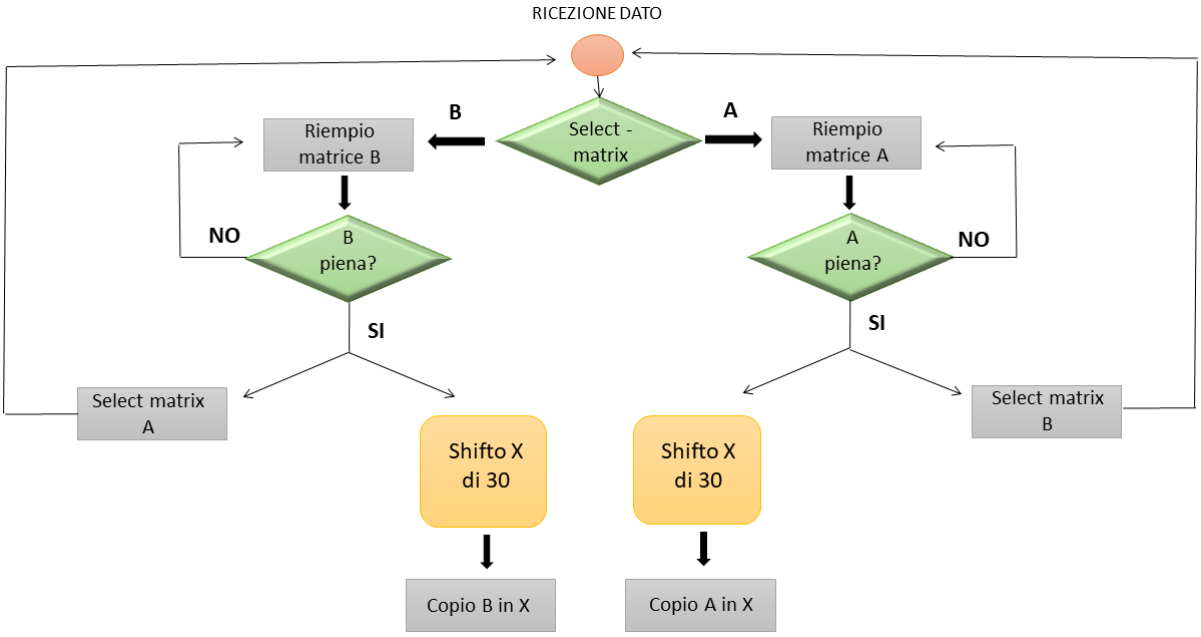

Figura 5.4: Flowchart relativo alla preparazione del dato da elaborare

relativo ad un canale. In base al valore del parametro *select_matrix*, i dati ricevuti andranno a riempire la matrice A oppure la matrice B. La presenza di due matrici per la ricezione dei dati è fondamentale per non perdere nessun campione in ingresso. Infatti, una volta che una delle due matrici viene riempita, quella piena viene utilizzata per creare la matrice da processare X mentre nel frattempo viene riempita l'altra. La matrice X che verrà utilizzata per il preprocessing è formata da 8 righe e 60 colonne e viene inizializzata a zero. Una volta che la prima matrice è stata riempita, i suoi valori vengono copiati nelle ultime 30 posizioni di X. Al ciclo successivo i campioni precedenti vengono shiftati a sinistra di 30 elementi facendo posto ai nuovi dati e così via.

Ogni volta che la matrice X è aggiornata viene eseguita su di essa l'azione di preprocessing. Inizialmente passa da un filtro notch a 50Hz, poi viene calcolata la WL su 60 campioni ed un overlap delle successive finestre del 50%. L'overlap del 50% è necessario per permettere al micro di eseguire tutte le funzioni successive prima che sia pronto un nuovo dato da elaborare. Una volta calcolata la feature viene eseguita la SVM che, grazie al modello caricato, fornisce in uscita la classe di appartenenza. Per limitare valori transitori di predizione errati, invece di inviare subito il dato in uscita dalla SVM, è stata inserita una fase di postprocessing (majority voting). I valori

in uscita dalla SVM vengono cioè caricati su un vettore di 10 elementi posti inizialmente a 0. Ogni volta che un nuovo valore di predizione è pronto, viene inserito nella posizione iniziale del vettore shiftando gli altri in avanti. Si controlla poi quale classe di appartenenza ha più ricorrenze e la si manda in uscita. Questo sarà il valore che tramite UART viene inviato al modulo Bluetooth per essere visualizzato sull'applicazione.

Per la scelta della lunghezza della finestra utilizzata per il majority voting viene calcolata l'accuratezza su 5 soggetti per vari valori della finestra. Anche se non tutti i picchi di accuratezza sono nello stesso punto, si può osservare che sono tutti concentrati nell'intorno del valore 10.

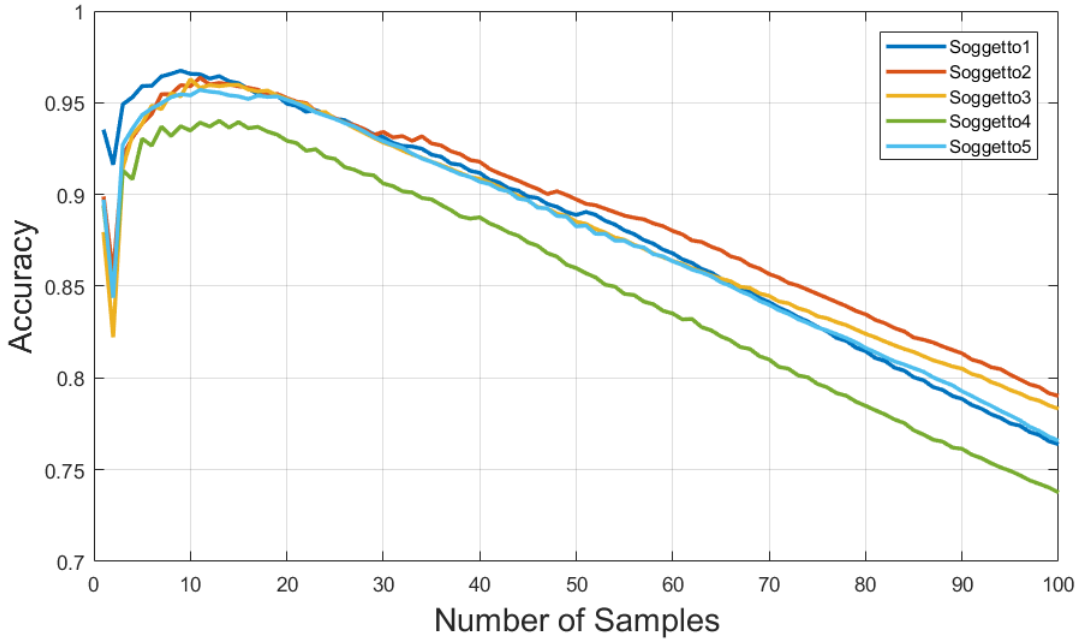

Figura 5.5: Sistema online

5.4 Tempistiche e consumi

Per effettuare un'analisi sulle tempistiche è sufficiente andare a misurare il numero di cicli di clock che impiega il micro per eseguire ogni azione e dividere poi il risultato ottenuto per la frequenza di lavoro $f_s = 168MHz$. Ogni ms, il segnale campionati dall'ADC arriva in ingresso al microcontrollore che impiega $0,03ms$ per processarlo e salvarlo nella matrice come spiegato precedentemente. Una volta che il micro ha ricevuto 30 campioni, la matrice è

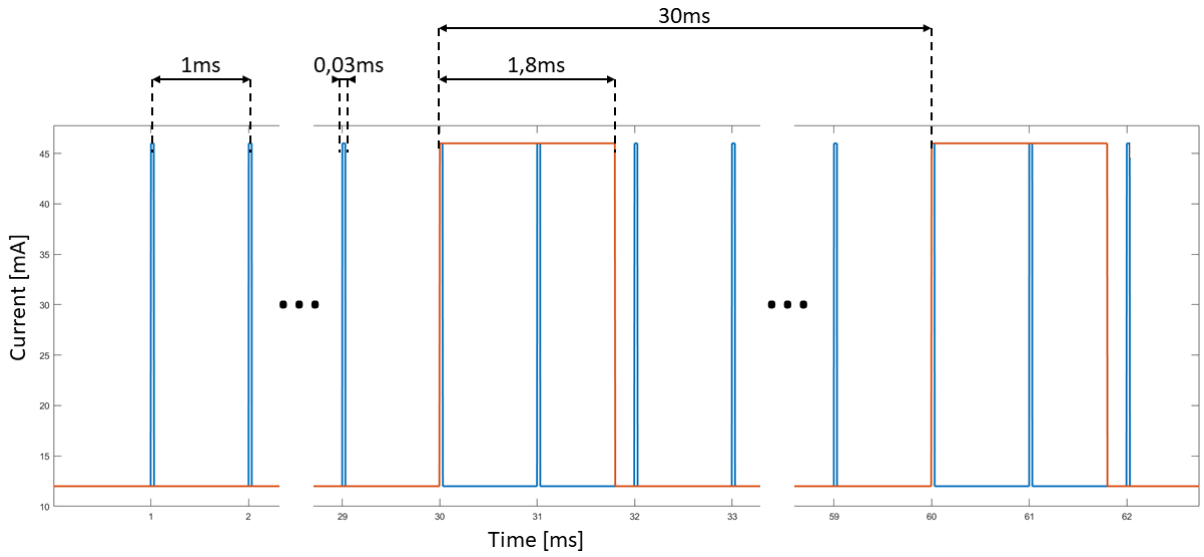

Figura 5.6: Tempistiche

pronta per essere elaborata. L'operazione di filtraggio, estrazione della feature, la classificazione SVM ed il majority voting impiegano in tutto $1,8ms$. Nel frattempo continua a ricevere i dati dall'ADC. Il tutto viene ripetuto ogni $30ms$, che risulta quindi essere il duty cycle del dispositivo. Il sistema lavora per il $9,5\%$ in run mode e per il $90,5\%$ in sleep mode. Dato che il microcontrollore è alimentato con una tensione pari a $3,3V$ e richiede una corrente di $46mA$ in run mode e $12mA$ in sleep mode , il consumo di potenza è dato da:

$$P_{STM32} = 3,3V \cdot (0,095 \cdot 46mA + 0,905 \cdot 12mA) = 50,25mW \tag{5.1}$$

A questo consumo va inoltre aggiunto quello dell'ADS1298 che è pari a $17,5mW$. In totale quindi si ottiene:

$$P_{TOT} = P_{STM32} + P_{ADC} = 67,75mW \tag{5.2}$$

Questo valore calcolato corrisponde al valore misurato su scheda nell'implementazione online. Se alimentato con una batteria da 250mAh, utilizzata in genere per applicazioni embedded in quanto piccola e leggera, il sistema garantisce un funzionamento vicino alle 13 ore.

5.5 Risultati

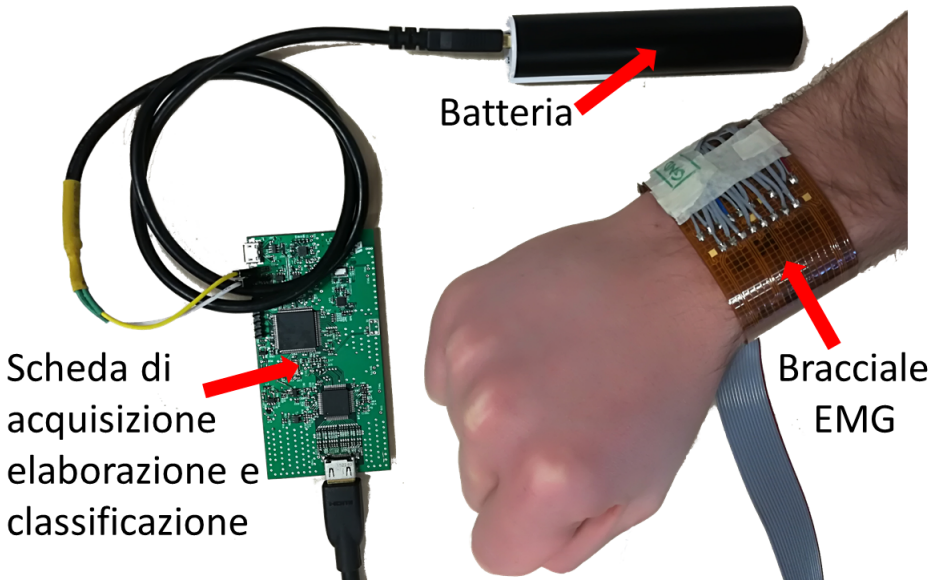

Figura 5.7: Foto del sistema relativo alle prove online, comprendente scheda, batteria e bracciale

Una volta verificato il funzionamento dell'intero sistema, è stato effettuato il training acquisendo i dati da 5 persone diverse nelle varie posizioni di rotazione del bracciale. Tramite MatLab è stato creato un modello per ogni soggetto e successivamente importato sul microcontrollore. I soggetti in esame hanno poi testato il dispositivo in giorni differenti rispetto alla sessione di training, in modo da verificare la capacità del sistema ad essere session independent. Grazie all'applicazione è possibile visualizzare in tempo reale la classe di appartenenza del gesto stimata dalla SVM calcolata sul micro e salvarla in un file di testo. In questo modo è possibile importare tale valore su MatLab e calcolare l'accuratezza dell'implementazione real-time. Il testing viene eseguito per le varie rotazioni del polsino, ottenendo una precisione media del $92,8\%$ con un numero di support vector molto limitato. I dati relativi alle precisioni di ogni soggetto e al numero di support vector utilizzati nei modelli sono riportati in Tabella 5.1

Per migliorare l'accuratezza è stata successivamente eseguita un'ulteriore fase di testing in cui il valore in uscita dalla SVM subisce una fase di postprocessing tramite il majority voting. In Tabella 5.2 viene riportato il confronto tra i valori di accuratezza ottenuti precedentemente e quelli ottenuti aggiungendo la fase di postprocessing. In tutti e 5 i soggetti si nota un netto

	centrale	destra	sinistra	media sogg.	nSV
Soggetto1	95,5	96,6	94,7	95,6	20
Soggetto2	94,7	89,7	91,0	91,8	21
Soggetto3	93,4	85,8	94,8	91,3	20
Soggetto4	89,6	93,0	92,8	91,8	18
Soggetto5	96,7	95,6	88,6	93,6	24

Tabella 5.1: Valori relativi alle percentuali di accuratezza, relativi a 5 soggetti, ottenute tramite la prova in real-time e relativi support vector

miglioramento. In particolare si è passati dal $92,8\%$ al $95,8\%$ ottenendo quindi un incremento di 3 punti percentuali.

	media senza postprocess	media con postprocess
Soggetto1	95,6	96,8
Soggetto2	91,8	96,4
Soggetto3	91,3	96,3
Soggetto4	91,8	94,0
Soggetto5	93,6	95,7
Media totale	92,8	95,8

Tabella 5.2: Confronto tra i valori di accuratezza ottenuti con e senza postprocessing

Si ricorda che la finestra su cui viene calcolata la WL è di 60 campioni, con un overlap del 50%, e che ogni campione viene ricevuto ogni ms. Ciò significa che ogni 30ms il micro esegue la classificazione del segnale corrispondente al gesto eseguito. L'aggiunta della fase di postprocessing si traduce in un ritardo nella risposta del sistema. La variazione tra un gesto ed un altro viene infatti segnalata solo quando la maggioranza delle stime all'interno della finestra corrisponde al nuovo gesto. Dato che per il postprocessing è stato utilizzato un buffer di lunghezza 10 elementi, il ritardo è di $300ms$, comunque conforme ai vincoli imposti per il real-time.

Conclusioni

In questa tesi è stata dimostrata la possibilità di classificare i gesti eseguiti dalla mano tramite un sistema embedded indossabile poco costoso e a basso consumo energetico che preleva il segnale EMG dal polso. Questa sua peculiarità lo rende meno ingombrante rispetto agli altri sistemi EMG che vengono indossati sull'avambraccio, in quanto può essere facilmente integrabile in un dispositivo da polso come un orologio o uno smartwatch.

Sono state eseguite varie prove offline con un prototipo realizzato in laboratorio, acquisendo dati da 5 persone differenti, in modo da valutare la configurazione da adottare per il dispositivo finale. In particolare si è capito che è preferibile eseguire il training del dispositivo in più posizioni rispetto che in una sola, per far si che la classificazione del gesto sia accurata anche ad eventuali rotazioni dello stesso. Si è poi ridotto il numero di gesti ottenendo un significativo aumento nell'accuratezza. Se si pensa ad una implementazione di questa architettura per il controllo di un dispositivo come uno smartwatch ciò non è limitativo in quanto è facilmente controllabile anche con meno gesti. É stato inoltre notato che l'accuratezza continua a rimanere alta anche con soli 4 canali differenziali. In seguito sono state confrontate le varie metodologie di estrazione delle feature da cui si evince che quella che garantisce un migliore rapporto tra accuratezza e numero di support vector per la classificazione EMG è la WL.

Tramite le informazioni apprese dalla prova offline è stato realizzato un PCB flessibile con cui viene eseguita l'implementazione realtime. Il risultato è un sistema stand-alone e session independent in grado di raggiungere un'accuratezza in real time mode del 96% ed in grado di durare fino a 13 ore se alimentato da una batteria di $250mAh$.

Gli studi ed i risultati presentati in questo elaborato sono stati la base di partenza per la stesura dell'articolo [28] che verrà presentato ad ISCAS 2018 ed in fase di pubblicazione.

Bibliografia

[1] M. Ahsan, M. I. Ibrahimy, and O. O. Khalifa. Advances in electromyogram signal classification to improve the quality of life for the disabled and aged people. *Journal of Computer Science*, 6(7):706, 2010.

[2] D. Farina, N. Jiang, H. Rehbaum, A. Holobar, B. Graimann, H. Dietl, and O. Aszmann. The extraction of neural information from the surface emg for the control of upper-limb prostheses: Emerging avenues and challenges. *Neural Systems and Rehabilitation Engineering, IEEE Transactions*, 22(4):797–809, 2014.

[3] S. Benatti, E. Farella, L. Benini, and E. Gruppioni. Analysis of robust implementation of an emg pattern recognition based control. *Intl. Conf. on Bio-inspired Systems and Signal Processing, BIOSIGNALS*, pages 45–54, 2014.

[4] H. Zhang, Y. Zhao, F. Yao, L. Xu, P. Shang, and G. Li. An adaptation strategy of using lda classifier for emg pattern recognition. *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, page 4267–4270, July 2013.

[5] M. A. Oskoei, S. Member, and H. Hu. Support vector machine-based classification scheme for myoelectric control applied to upper limb. *Engineering in Medicine and Biology Society (EMBC)*, 35th Annual International Conference of the IEEE, 2013.

[6] A. D. Bellingegni, E. Gruppioni, G. Colazzo, A. Davalli, R. Sacchetti, E. Guglielmelli, and L. Zollo. Nlr, mlp, svm and lda: a comparative analysis on emg data from people with trans-radial amputation. *Journal of neuroengineering and rehabilitation*, 14(1):82, 2017.

[7] L. Hargrove, K. Englehart, and B. Hudgins. A comparison of surface and intramuscular myoelectric signal classification. *Biomedical Engineering, IEEE Transactions*, 54(5):847–853, 2007.

[8] B. Giroux and M. Lamontagne. Comparisons between surface electrodes and intramuscular wire electrodes in isometric and dynamic conditions. *Electromyography and clinical neurophysiology*, 30(7):397–405, 1990.

[9] Y.-H. Chiou, J.-J. Luh, S.-C. Chen, J.-S. Lai, and T.-S. Kuo. The comparison of electromyographic pattern classifications with active and passive electrodes. *Medical engineering physics*, 26(7):605–610, 2004.

[10] A. Andrews, E. Morin, and L. McLean. Optimal electrode configurations for finger movement classification using emg. *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 2987–2990, 2009.

[11] J. McIntosh, C. McNeill, M. Fraser, F. Kerber, M. L¨ochtefeld, and A. Kruger. Empress: Practical hand gesture classification with wrist-mounted emg and pressure sensing. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. ACM*, pages 2332–2342, 2016.

[12] MYO ARM BAND. http://www.thalmic.com/.

[13] R. Yazicioglu, P. Merken, R. Puers, and C. Van Hoof. A 200 uw eight-channel eeg acquisition asic for ambulatory eeg systems. *IEEE Journal of Solid-State Circuits (JSSC)*, 43(12):3025–3038, 2008.

[14] C. Lopez, D. Prodanov, D. Braeken, I. Gligorijevic, W. Eberle, C. Bartic, R. Puers, and G. Gielen. A multichannel integrated circuit for electrical recording of neural activity, with independent channel programmability. *IEEE Trans. on Biomedical Circuits and Systems*, 6(2):101–110, 2012.

[15] N. Van Helleputte, M. Konijnenburg, J. Pettine, D.-W. Jee, H. Kim, A. Morgado, R. Van Wegberg, T. Torfs, R. Mohan, A. Breeschoten, H. de Groot, C. Van Hoof, and R. Yazicioglu. A 345—,uw multi-sensor biomedical soc with bio-impedance, 3-channel ecg, motion artifact reduction, and integrated dsp. *IEEE Journal of Solid-State Circuits (JSSC)*, 50(1):230–244, 2015.

[16] S. Benatti, F. Casamassima, B. Milosevic, E. Farella, P. Schonle, S. Fateh, T. Burger, Q. Huang, and L. Benini. A versatile embedded platform for emg acquisition and gesture recognition. *Proceedings of the IEEE*, 2015.

[17] http://www.centropiaggio.unipi.it/sites/default/files/course/material/15.

[18] A. Phinyomark, C. Limsakul, and P. Phukpattaranont. A novel feature extraction for robust emg pattern recognition. *JOURNAL OF COMPUTING*, 1(1), 2009.

[19] https://en.wikipedia.org/wiki/Discrete$_w avelet_t ransform$.

[20] Fabio Montagna. Scalability of neuromodulation algorithms for ultra low power brain machine interfaces. Master's thesis, Alma Mater Studiorum University of Bologna, Bologna, Italy, 2015.

[21] https://en.wikipedia.org/wiki/Support$_v ector_m achine$.

[22] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(27):1–27, 2011.

[23] Covidien. *Kendall™ ECG Electrodes Product Data Sheet Arbo™ H124SG.*

[24] Texas Instruments. *ADS129x Low-Power, 8-Channel, 24-Bit Analog Front-End for Biopotential Measurements.*

[25] STMicroelectronics. *STM32F407VG, High performance foundation line, ARM Cortex-M4 core with DSP and FPU, 1 Mbyte Flash, 168Mhz CPU, ART accelerator, Ethernet, FSMC Datasheet.*

[26] Victor Javier Kartsch Morinigo. Online alpha wave detector: An embedded hardware-software implementation. Master's thesis, Alma Mater Studiorum University of Bologna, Bologna, Italy, 2016.

[27] Andrew Ng. Machine Learning, by Stanford University, www.coursera.com.

[28] V. Kartsch, S. Benatti, M.Mancini, M.Magno, and L.Benini. Smart wearable wristband for emg based gesture recognition powered by solar energy harvester. *ISCAS*, 2018.

Ringraziamenti

Ed eccomi qua, dopo 5 anni di studi intensi mi trovo alla chiusura di un ciclo. Non è stato sempre tutto rosa e fiori, sopratutto in questi ultimi 2 anni di laurea specialistica, ma grazie al sostegno di tutte quelle persone che mi sono sempre state vicine e che mi hanno sostenuto sono riuscito a centrare l'obiettivo.

Ringrazio innanzitutto il Professore Luca Benini, per avermi proposto questo progetto e per la grande disponibilità e cortesia.

Ringrazio ovviamente anche il Correlatore Simone Benatti, per il suo continuo appoggio, l'infinita pazienza, e per l'entusiasmo trasmessomi nella ricerca.

Ringrazio inoltre Victor, una persona che stimo molto per il suo impegno e la sua dedizione per tutto quello che fa. Ho imparato molto da lui, sia in termini ingegneristici che umani (come ad esempio la vita "nel suo paese") instaurando un rapporto di amicizia oltre che lavorativo.

Un grazie a Fabio per il suo sostegno ed il suo perenne buon umore e ovviamente grazie anche a Mattia, Alberto, Tommaso e Hanie per avermi allietato le giornate. Perchè anche in un momento stressante come può essere quello di una tesi, avere un gruppo così aiuta molto.

Ringrazio inoltre tutti i miei compagni di corso, in particolare Alan, Enrico, Matteo e Fede, che sono stati in grado di alleggerire questo percorso universitario, rafforzando ancora di più il nostro rapporto di amicizia.

Un ringraziamento speciale va inoltre ai miei genitori, che non mi hanno mai fatto mancare nulla e hanno dovuto far fronte a tanti sacrifici per rendere possibile tutto ciò aiutandomi sia moralmente che economicamente. Le parole che mi vengono in mente non saranno mai sufficienti per compensare tutto quello che mi avete dato. Mi siete sempre stati vicini e non vi ringrazierò mai abbastanza.

Colgo inoltre l'occasione per ringraziare tutta la mia famiglia e i miei zii, che mi hanno sempre incoraggiato e sostenuto, e mio fratello Mirko per avermi distratto dagli studi regalandomi momenti di svago con la Play. Ringrazio

poi il nonno Bruno e la nonna Pina per essere sempre stati interessati nelle mie attività e la nonna Santa per le tante parole di affetto.

Un grazie speciale va inoltre al nonno Marino, a cui ho dedicato questa tesi. Fin da bambino il suo sogno era diventare un ingegnere ma non ha potuto realizzarlo e ha riposto in me le sue speranze, trasmettendomi la forza necessaria per andare avanti nonostante le sue condizioni di salute. Alla fine qualcuno della famiglia è riuscito a realizzare il tuo sogno, e ne sono molto orgoglioso.

Grazie inoltre a tutti quegli amici che non mi hanno mai abbandonato, anche quando il tempo libero scarseggiava. Grazie a Vlad, Tarlo, Caste, Elena, Lenisa e Nicole per avermi saputo aiutare a svagarmi nei momenti più pesanti.

Per concludere non posso fare a meno di ringraziare Jessica, una persona per me davvero molto importante, che mi sostiene quotidianamente e su cui so di poter sempre contare. Grazie, perchè con una persona così al proprio fianco è più facile affrontare le difficoltà della vita. Grazie anche a tutta la sua famiglia per avermi accolto come un figlio.

Grazie a tutti.