POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

IDENTIFICAZIONE DI ANOMALIE IN TRACCIATI ECG MEDIANTE DISPOSITIVI INDOSSABILI

Algoritmi per l'analisi in movimento ed implementazione ottimizzata per sistemi ultra low-power

Relatore: Porf. Giacomo Boracchi
Correlatore: Dott.ssa Pasqualina Fragneto

Tesi di Laurea di:
Marco Longoni
Matr. n. 838315

Anno Accademico 2016-2017

A mio nonno

Ringraziamenti

Vorrei ringraziare il prof. Giacomo Boracchi, relatore di questa tesi, per l'aiuto fornito, la costante presenza e la capacità di trasmettere conoscenza con il sorriso. Ringrazio Lilli, Bea e Diego, siete tre persone fantastiche, è stato bello poter lavorare con voi, vi ringrazio per l'aiuto la conoscenza e la serenità che sapete trasmettere. Ringrazio Angela per esserci sempre e soprattutto con il sorriso. Ringrazio la mia famiglia, Argo e gli amici di sempre. In ultimo ringrazio tutti coloro che hanno letteralmente corso.

Abstract

In this thesis we will present a system capable of detecting anomalous heartbeats during the acquisition of electrocardiographic traces (ECG) through wearable devices. In particular, the system is composed by a wearable device and a development board, which communicate in real-time via a Bluetooth connection. The wearable device, called Bio2Bit, acquires and transmits the ECG trace to the board, called NUCLEO STM32, which analyses each single beat and detects possible anomalies.

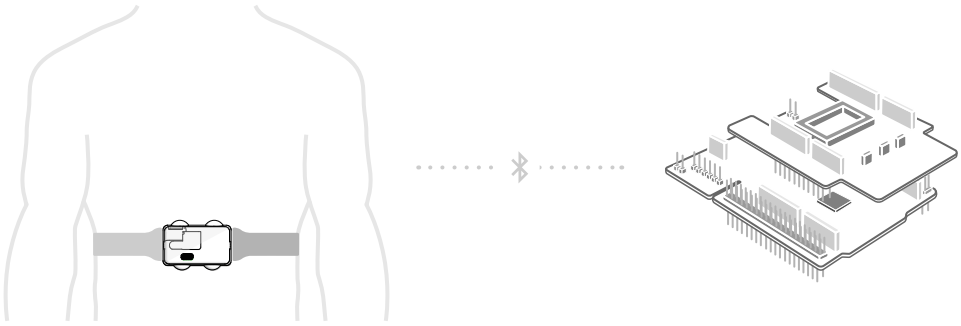

Figura 1: The Bio2Bit device acquires and transmits in real-time the ECG trace to the NUCLEO STM32 board.

We assume that ECG heartbeats are well described by a *sparse model*, namely a model based on the sparse representation theory. We suppose that normal ECG heartbeats, $\mathbf{s} \in \mathbb{R}^p$ can be modeled with a collection of few columns of a matrix $D \in \mathbb{R}^{p \times n}$ called *dictionary*. In particular, we propose to learn D in a *data-driven* fashion, i.e. directly from the data acquired by the wearable device. Once the dictionary D is learned, we can solve the anomaly detection problem, by analyzing each beat and verifying if it can not be reconstructed by a sparse representation based on D. Anomalies might be due, for instance, to arrhythmias, movements or acquisition errors.

This work consists of two main parts. In the first, we focused on how to make the anomaly detection system robust to the noise generated by the user's movements. To make the system operational in real-world scenarios is necessary to distinguish cardiac anomalies from noisy beats generated by user's movements. Our proposed solution is able to predict the occurrence of noise beats due to user's movements,avoiding generating false anomalies with the anomaly detector. The noisy heartbeat prediction has been addressed as a machine learning problem, in particular as a binary classification problem, where given the MEMS signal and the current beat (classifier's input signal) the classifier predicts the correct label, noisy or not noisy.

The second part is dedicated to integrate the anomaly detection system on the NUCLEO board STM32. We present the software modules needed to achieve the anomalous real time detection, analyzing each single beat, and how they interact with each other. These modules allow to:

- filter the ECG signal in order to remove the baseline and high frequency noise,
- detect and discard noisy beats due to the user's movements,
- detect and label anomalous beats.

For the last module we have implemented an optimized library for ultra low-power embedded systems, which efficiently resolves the sparse coding problem.

The contribution of this thesis is twofold: firstly, the neural network reduces the false alarm rate due to user's movements. On the other hand, the implemented software modules allow the real-time execution of the anomaly detection system.

The thesis document is structured as follows. In Chapter 2 we will first introduce how the ECG signal is physically generated, we will present a solution to the anomaly detection problem based on the sparse representation theory and lastly we will describe the implemented pipeline used during the real-time electrocardiographic monitoring. In Chapter 3 we will deal with the problem of false anomalies due to the user's movements: a solution based on a predictive model, able to predict the occurrence of this phenomenon, will be described. The chosen model is a neural network that takes as input a set of features extracted from the MEMS signal (accelerometer and gyroscope) and the current heartbeat, and outputs whether the beat is noisy or not. In Chapter 4 we will present the framework used for the training

phase and lastly and we will show the classifier results. In Chapter 5 we will illustrate the components and the hardware and software architecture. In the Chapter 6 we will present the software modules used to integrate the anomaly detector on the system described in Chapter 5. In Chapter 7 we will report the conclusions and future works.

Indice

Capitolo 1

Introduzione

In questa tesi presenteremo un sistema in grado di rilevare anomalie cardiache durante l'acquisizione di tracciati elettrocardiografici (ECG) attraverso dispositivi indossabili. In particolare, l'acquisizione dell'ECG avviene attraverso il sistema rappresentato in Figura 1.1: il dispositivo indossabile denominato Bio2Bit acquisisce e trasmette in tempo reale via Bluetooth il tracciato alla scheda NUCLEO STM32 che analizza ogni singolo battito rilevando eventuali anomalie.

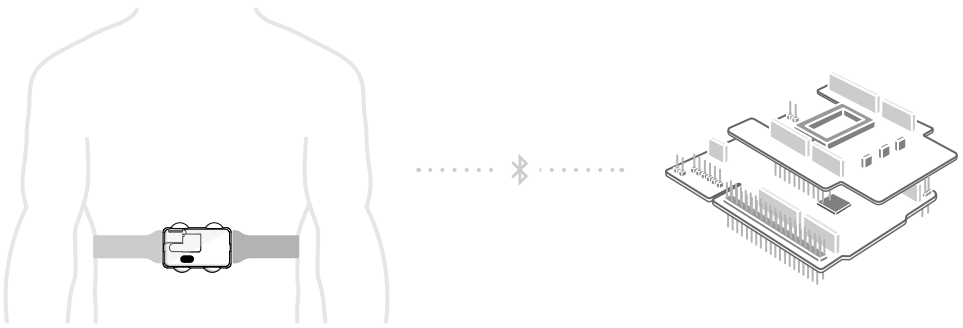

Figura 1.1: Il dispositivo Bio2Bit acquisisce e trasmette in real-time il tracciato ECG alla scheda NUCLEO STM32.

Il rilevamento delle anomalie si basa sulla teoria della rappresentazione sparsa. Tale teoria si pone l'obiettivo di descrivere vettori $\mathbf{s} \in \mathbb{R}^p$ (nel nostro caso battiti cardiaci) mediante una collezione di poche colonne di una matrice $D \in \mathbb{R}^{p \times n}$ chiamata *dizionario*. Nella nostra soluzione la matrice D è ottenuta mediante un approccio *data-driven*, ossia apprendendo le caratteristiche che descrivono l'attività cardiaca degli utenti a partire dai loro

stessi dati elettrocardiografici. Una volta appreso il dizionario, è possibile risolvere il problema del *rilevamento delle anomalie* (indicato anche come *anomaly detection*), analizzando ogni singolo battito e verificando se si presentano situazioni (*anomale*) in cui il battito non è ben ricostruito tramite una rappresentazione sparsa di D. I *battiti anomali* potrebbero esser causati da vere e proprie *aritmie* cardiache, *movimenti* (i quali si verificano frequentemente durante acquisizioni lunghe) o da errori di acquisizione.

Il presente lavoro è composto da due parti principali. Nella prima, ci siamo focalizzati su come rendere il sistema di rilevamento delle anomalie robusto al rumore generato da movimenti compiuti dall'utente. Infatti, al fine di rendere il sistema utilizzabile in scenari reali è opportuno saper distinguere le anomalie cardiache dai disturbi che alternano la struttura morfologica del battito. In particolare, proponiamo una soluzione in grado di predire il verificarsi di disturbi dovuti al movimento un cui i battiti predetti come rumorosi vengono scartati, evitando di generare false anomalie attraverso l'anomaly detector. La predizione dei battiti rumorosi è affrontata come un problema di classificazione binaria mediante tecniche di *machine learning*, dove i dati d'ingresso sono il battito corrente e i segnali MEMS corrispondenti, rilevati sempre dal dispositivo Bio2Bit.

La seconda parte è dedicata all'integrazione del sistema sulla scheda NUCLEO STM32. In particolare, presentiamo sia i moduli software per l'analisi dei singoli battiti e il rilevamento delle anomalie in tempo reale sia le modalità in cui interagiscono tra loro. Tali moduli permettono di:

- filtrare il segnale elettrocardiografico in modo da eliminare la baseline ed eventuali rumori ad alte frequenze,
- rilevare e scartare i battiti rumorosi a causa del movimento,
- rilevare ed etichettare i battiti anomali.

Per il modulo dedicato a quest'ultima funzionalità abbiamo sviluppato una libreria ottimizzata per sistemi embedded ultra low-power, che risolve efficientemente la ricerca della soluzione sparsa di un sistema.

Il contributo di questo lavoro è duplice: innanzitutto la rete neurale sviluppata nella prima parte ci permette di ottenere una riduzione dei falsi allarmi causati dai movimenti dell'utente; secondariamente i moduli software sviluppati nella seconda parte permettono l'esecuzione in tempo reale del rilevamento di anomalie in tracciati elettrocardiografici.

Il documento di tesi è strutturato nel modo seguente, nel Capitolo 2 verrà descritto il fenomeno fisico dal quale ha origine il segnale elettrocardiografico, verrà presentata una soluzione al problema del rilevamento delle anomalie basata sulla teoria della rappresentazione sparsa, e verrà presentata la pipeline sviluppata per poter risolvere il problema dell'*anomaly detection* durante il monitoraggio in tempo reale del tracciato elettrocardiografico. Nel Capitolo 3 affronteremo il problema delle false anomalie generate dai movimenti dell'utente e verrà proposto un modello in grado di predire il verificarsi di questo fenomeno basato su una rete neurale avente come ingresso *features* estratte dai segnali MEMS e dal battito corrente. Nel Capitolo 4 verrà presentato il framework utilizzato per l'addestramento del modello e verranno presentati i risultati ottenuti. Nel Capitolo 5 verrà descritto il sistema di monitoraggio del tracciato elettrocardiografico basato sulla scheda di sviluppo NUCLEO STM32L476RG e sul dispositivo indossabile Bio2Bit. Nel Capitolo 6 presenteremo i moduli software necessari all'integrazione del rilevamento delle anomalie sul sistema presentato nel Capitolo 5. Infine nel Capitolo 7 riportiamo le conclusioni e gli sviluppi futuri di questo lavoro.

Capitolo 2

Anomaly Detector per ECG

In questo Capitolo descriveremo il fenomeno fisico dal quale ha origine il segnale elettrocardiografico focalizzandoci sul singolo *ciclo cardiaco*. Inoltre forniremo una soluzione al problema del rilevamento delle anomalie basata sulla teoria della rappresentazione sparsa. Tale teoria si pone l'obiettivo di descrivere vettori $\mathbf{s} \in \mathbb{R}^p$ (nel nostro caso battiti cardiaci) mediante una collezione di poche colonne di una matrice $D \in \mathbb{R}^{p \times n}$ chiamata *dizionario*. La ricerca della rappresentazione di $\mathbf{s}$ mediante le colonne di D si realizza risolvendo il sistema lineare $\mathbf{s} = D\mathbf{x}$ richiedendo che il vettore dei coefficienti $\mathbf{x} \in \mathbb{R}^n$ sia *sparso*, ovvero abbia al più un numero κ di componenti non nulle, con $\kappa < n$. Nella nostra soluzione la matrice D è appresa con un approccio *data-driven*, ossia apprendendo le caratteristiche che descrivono l'attività cardiaca degli utenti a partire dai loro stessi dati elettrocardiografici. Una volta appresa, quest'ultima ci permetterà di risolvere il problema del *rilevamento delle anomalie* (indicato anche come *anomaly detection*), analizzando ogni singolo battito e verificando se si presentano situazioni *anomale* in cui il battito non è ben ricostruito tramite una rappresentazione sparsa di D. Infine descriveremo la Pipeline sviluppata per poter risolvere il problema dell'*anomaly detection* durante il monitoraggio in real-time del tracciato elettrocardiografico tramite il dispositivo indossabile Bio2Bit.

2.1 Telemedicina e Telecardiologia

La telemedicina permette l'erogazione di servizi sanitari a distanza attraverso l'utilizzo di tecnologie informatiche e delle telecomunicazioni al fine di scambiare informazioni utili alla diagnosi, al trattamento e alla prevenzione delle malattie.

La telemedicina nasce con lo scopo di migliorare la qualità di vita del paziente, agevolare il lavoro di medici e infermieri e incrementare l'efficienza e la produttività del servizio sanitario. In particolare essa rappresenta una modalità di erogazione dell'assistenza sanitaria attraverso cui si rendono disponibili risorse mediche, sia di base sia specialistiche, a pazienti che non possono accedere direttamente ai servizi sanitari per difficoltà di varia natura. L'impiego di sistemi di telecomunicazione avanzati rende possibile la trasmissione a distanza di informazioni mediche dal paziente alla struttura sanitaria e viceversa, evitando spostamenti fisici. Consente inoltre il controllo extraospedaliero di pazienti affetti da patologie gravi, garantendo interventi tempestivi nei casi di emergenza.

Oggigiorno, grazie ai dispositivi indossabili (o *wearable*), la telemedicina dispone di tutte le potenzialità necessarie per migliorare la degenza e il monitoraggio dello stato di salute degli utenti da remoto. In particolare i dispositivi wearable permettono di memorizzare e trasmettere segnali o parametri biologici (come ad es. il battito cardiaco, la saturazione dell' ossigeno o la frequenza respiratoria) e di movimento (come ad es. l'accelerazione) attraverso acquisizioni in tempo reale. Inoltre essi possono segnalare il superamento di determinate soglie critiche per i parametri, abilitando diversi tipi di interventi. Attualmente le applicazioni piú diffuse fanno uso bracciali hi-tech, sensori su fasce, orologi o magliette intelligenti.

Il processo di telemonitoraggio è costituito da tre funzioni fondamentali, rappresentate graficamente in Figura 2.1:

- Rilevamento ed invio di segnali clinicamente significativi dagli utenti verso un sistema di controllo (di assistenza o di servizio).
- Acquisizioni, analisi e valutazione di tali segnali da parte del sistemi.
- Attuazione di interventi presso gli utenti realizzabili con modalità differenti a seconda dei casi.

La telecardiologia è una delle branche più diffuse della telemedicina e consiste nella trasmissione e refertazione in tempo reale del tracciato elettrocardiografico di un utente. Essa è particolarmente vantaggiosa soprattutto quando le distanze tra l'utente e il sistema controllo, così come le tempistiche di intervento, sono un fattore critico. Infatti un'immediata consulenza cardiologica on-line rappresenta uno strumento sempre piú importante in diversi ambiti clinici che spaziano da situazioni di urgenza a progetti di monitoraggio elettrocardiografico su larga scala.

Gli elementi principali che caratterizzano un sistema di monitoraggio telecardiologico sono i seguenti: apparecchiature elettromedicali o dispositivi indossabili dotati di sensori e trasduttori in grado di acquisire il tracciato ECG, un sistema di trasmissione necessario ad inviare i dati dell'utente verso il sistema di controllo (che può essere una clinica medica o un server nel quale vengono salvati e analizzati i dati del paziente attraverso appositi software).

Una delle principale applicazioni di telecardiologia è il monitoraggio cardiologico che consiste nella registrazione del segnale elettrocardiografico. In questo caso, innanzitutto l'utente sceglie quando e per quanto tempo effettuare l'esame (ad es. durante passeggiate o esercizi di riabilitazione), successivamente i dati rilevati dal dispositivo indossabile vengono trasmessi al sistema di controllo ed esaminati, e infine, terminato l'esame, l'utente riceve il referto e le eventuali prescrizioni.

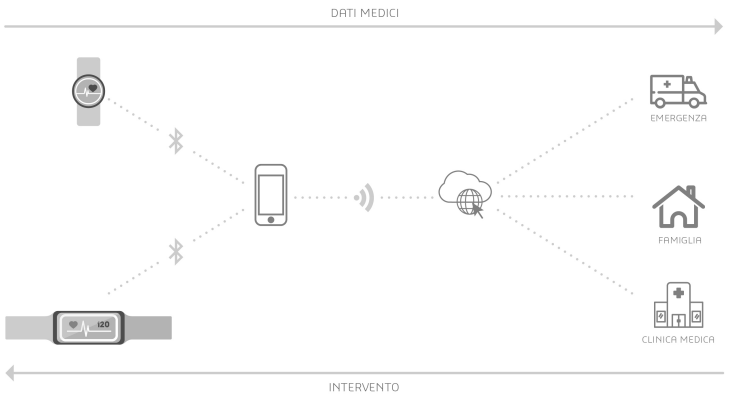

Figura 2.1: Schema a blocchi di un sistema di telemedicina. I segnali biologici vengono rilevati e trasmessi dall'utente al sistema di controllo; qui i segnali vengono analizzati e qualora necessario vengono attivati interventi presso gli assistiti.

2.2 L' Elettrocardiogramma

In questo paragrafo verrà data una descrizione sintetica del fenomeno fisico dal quale ha origine il segnale elettrocardiografico. Le informazioni riportate sono state ricavate da [8], dove è possibile trovare una descrizione più dettagliata del fenomeno.

L'attività cardiaca è comunemente controllata attraverso l'analisi dell'elettrocardiogramma (o ECG), il quale rappresenta il grafico dell'andamento del potenziale cardiaco rispetto al tempo.

I muscoli del cuore si contraggano attraverso la propagazione di impulsi elettrici, generando fenomeni di polarizzazione e depolarizzazione nelle diverse parti dell'organo. Tali fenomeni danno origine a campi elettrici che si estendono fino sulla superficie corporea. Conseguentemente l'attività cardiaca può essere monitorata attraverso l'utilizzo di coppie di elettrodi posti in diverse posizioni del corpo (ad es. caviglie, polsi o torace), ad ognuna delle quali corrisponde un *canale* del tracciato ECG. In particolare, il potenziale elettrico varia in direzione, verso e ampiezza a seconda del canale osservato. Ogni singolo *battito* descrive un *ciclo cardiaco*, il quale comprende fasi di *sistole*, ovvero di contrazione, e di *diastole*, ovvero di rilassamento. Un battito cardiaco *normale* ha origine con un impulso prodotto nel nodo seno-atriale (SA), il quale si diffonde velocemente attraverso le cellule degli atri grazie alle giunzioni serrate, causandone la contrazione simultanea.

La contrazione degli atri stimola il nodo atrio-ventricolare, generando impulsi che vengono condotti ai ventricoli attraverso il fascio di His e le fibre di Purkinje. Lo stimolo si diffonde poi attraverso la massa muscolare dei ventricoli e ne causa la contrazione.

In Figura 2.2 è rappresentata la morfologia di un singolo battito cardiaco. La contrazione atriale causa una deflessione nel tracciato ECG chiamata curva P. Le altre curve, Q, R, S e T sono dovute all'azione dei ventricoli e sono quindi note come *complesso ventricolare*. In particolare, il complesso QRS, rappresenta la diffusione dello stimolo elettrico attraverso la muscolatura ventricolare. L'intervallo PR, indica il periodo di tempo che intercorre tra l'inizio dell'onda P e il complesso QRS e corrisponde al tempo di propagazione dell'impulso di depolarizzazione prodotto dal nodo SA fino a raggiungere il nodo atrio-ventricolare.

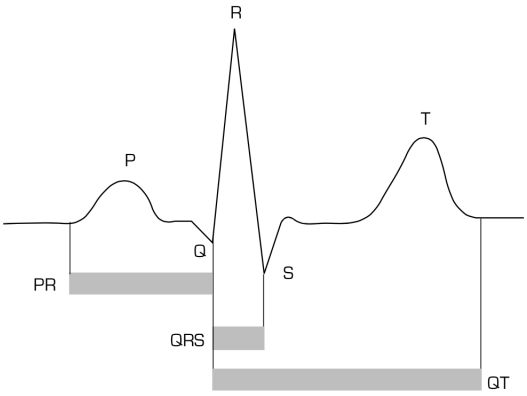

Figura 2.2: Rappresentazione grafica di un battito sintetico.

Normalmente l'intervallo PR ha una durata media di 0.12-0.2 secondi, mentre il complesso QRS richiede un massimo di 0.12 secondi. L'intervallo QT esprime il tempo necessario al miocardio ventricolare per depolarizzarsi e ripolarizzarsi e viene misurato dall'inizio del complesso QRS fino al termine dell'onda T. Esso è correlato alla frequenza cardiaca, in particolare maggiore è la frequenza cardiaca minore sarà la dimensione dell'intervallo QT.

La morfologia del battito varia a seconda della frequenza cardiaca, della posizione degli elettrodi sul corpo, e dall'utente. In Figura 5.1 è riportato un battito cardiaco ad una frequenza di 70 bpm (battiti per minuto) estratto da una registrazione effettuata con il dispositivo indossabile Bio2Bit posizionato al centro del torace.

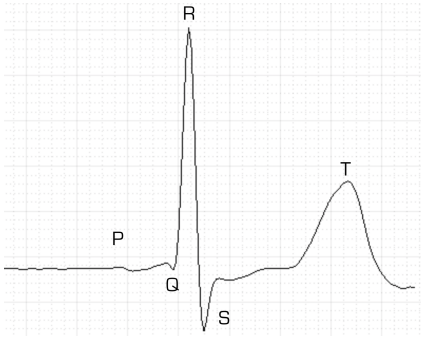

Figura 2.3: Battito normale ad una frequenza di 70 bmp (battiti per minuto) estratto da una registrazione effettuata con il dispositivo indossabile Bio2Bit posizionato al centro del torace.

Le grandezze descritte in precedenza (dimensione degli intervalli QRS, QT, ecc.) vengono solitamente utilizzate per discriminare se il battito presenta una morfologia *normale* o *anomala* (per esempio, valutando se i loro valori sono compresi in un determinato intervallo [16]). Questo metodo d'analisi, indicato come metodo *expert driven* si basa sull'analisi di caratteristiche (o *features*) stabilite a priori da medici e dottori i cui valori di riferimento sono appresi valutando l'andamento medio di una certa popolazione.

Un'alternativa a tale approccio consiste nell'apprendere direttamente dai dati (in questo caso il tracciato ECG) le caratteristiche necessarie a classificare un battito come normale o anomalo. Questo metodo d'analisi, denominato *data-driven*, ha il vantaggio, rispetto al caso precedente, di essere strettamente correlato con l'utente.

Nel prossimo paragrafo verrà introdotto il problema del rilevamento di battiti anomali nel tracciato ECG. Tale problema verrà affrontato attraverso un approccio *data-driven*, ossia apprendendo un modello in grado di sintetizzare le caratteristiche rilevanti per descrivere l'attività cardiaca dell'utente partendo direttamente dal tracciato ECG dell'utente stesso.

2.3 Formulazione Matematica del Problema

Denotiamo con $s\colon \mathbb{N} \to \mathbb{R}$ il tracciato ECG campionato uniformemente nel tempo, e assumiamo che ogni singolo battito si possa estrarre automaticamente dal tracciato (ad es. attraverso il procedimento descritto in [17]). Indichiamo l'*i-esimo* battito cardiaco $\mathbf{s}_i \in \mathbb{R}^p$ come il seguente vettore:

$$\mathbf{s}_i = \{s(t_i + u) \ : \ u \in \mathcal{U}\}, \tag{2.1}$$

dove $\mathcal{U}$ è un intorno dell'origine contenente p campioni del segnale elettrocardiografico, e t_i è il campione corrispondente all'*i-esimo* picco R del tracciato. Assumiamo che i battiti normali di ciascun utente siano generati da un processo stocastico $\mathcal{P}_N$ che ne caratterizza la morfologia. Il nostro obiettivo è innanzitutto quello di apprendere un modello matematico in grado di descrivere tali battiti. Questo modello verrà poi utilizzato per risolvere il problema del *rilevamento delle anomalie*, al quale ci riferiremo in seguito anche come *anomaly detection* per l'ECG.

In particolare, definiamo *anomali* quei battiti che hanno una morfologia differente da quelli generati dal processo P_N. Assumiamo che i battiti anomali siano generati da un processo stocastico $\mathcal{P}_A \neq \mathcal{P}_N$. I *battiti anomali* potrebbero esser causati da vere e proprie *aritmie* cardiache, da *movimenti* (i quali si verificano frequentemente durante lunghe acquisizioni) oppure da errori di acquisizione. In Figura 2.4 è riportato un battito *anomalo* registrato con il dispositivo indossabile Bio2Bit. Il rilevamento delle anomalie avviene analizzando ogni singolo battito $\mathbf{s}_i$ e verificando se esso rispetta o meno le caratteristiche del modello appreso dai battiti generati da $\mathcal{P}_N$.

Figura 2.4: Battito anomalo causato da un movimento dell'utente durante l'acquisizione con il dispositivo indossabile Bio2Bit. Il rumore generato dal movimento si somma al segnale causando la scomparsa delle onde P e T e alterando così la corretta morfologia del battito.

Assumiamo che i battiti $\mathbf{s}_i \in \mathbb{R}^p$ generati dal processo stocastico $\mathcal{P}_N$ possano venir approssimati correttamente dal seguente modello lineare:

$$\mathbf{s}_i \approx D\mathbf{x}_i \,, \tag{2.2}$$

dove $D \in \mathbb{R}^{p \times n}$ è una matrice denominata *dizionario* e $\mathbf{x}_i \in \mathbb{R}^n$ è un vettore di coefficienti. In particolare assumiamo che il vettore $\mathbf{x}_i$ sia *sparso*, ovvero che il numero di elementi non nulli κ sia significativamente inferiore rispetto alla cardinalità del vettore, i.e. $\|\mathbf{x}_i\|_0 \leq \kappa$, dove $\kappa < n$.

Il *dizionario* D è appreso da un *training set* contenente un insieme di m battiti generati dal processo stocastico $\mathcal{P}_N$ di un singolo utente. Gli m battiti vengono acquisiti e incolonnati nella matrice $S \in \mathbb{R}^{p \times m}$. Il processo d'apprendimento è formalizzato come:

$$[D, X] = \underset{\widetilde{D} \in \mathbb{R}^{p \times n},\, \widetilde{X} \in \mathbb{R}^{n \times m}}{\arg\min} \|\widetilde{D}\widetilde{X} - S\|_2 \text{ t.c. } \|\widetilde{\mathbf{x}}_i\|_0 \leq \kappa, \;\; i = 1, \ldots, n \tag{2.3}$$

dove $X \in \mathbb{R}^{n \times m}$ è una matrice le cui colonne contengono i vettori dei coefficienti relativi ai battiti in S. Nella pratica il problema riportato al punto (2.3) può essere risolto attraverso l'algoritmo K-SVD [1], il quale verrà descritto in seguito. Il dizionario è *user-specific*, ovvero le colonne di D (denominate *atomi*), sintetizzano la morfologia di n battiti rilevanti che descrivono l'attività cardiaca di un singolo utente.

Apprendere D in modo tale che l'equazione (2.2) sia soddisfatta corrisponde ad apprendere un'unione di sottospazi a bassa dimensionalità di $\mathbb{R}^n$ ai quali appartengono i battiti aventi una corretta morfologia.

Nella risoluzione del problema (2.2) imponiamo che solo κ tra le n colonne di D possono venir selezionate, di conseguenza i sottospazi avranno una dimensione massima pari a κ.

Il vettore $\mathbf{x}_i$ è la *rappresentazione sparsa* del battito corrente $\mathbf{s}_i$, ed essa viene ottenuta proiettando $\mathbf{s}_i$ nello spazio generato dagli atomi del dizionario D in modo tale da minimizzare l'*errore di ricostruzione* (2.5). In Figura 2.5 è riportata la rappresentazione grafica della proiezione appena descritta. Il problema del calcolo di $\mathbf{x}_i$ è noto in letteratura come *sparse coding* [12] e la sua rappresentazione matematica è la seguente:

$$\mathbf{x}_i = \underset{\widetilde{\mathbf{x}} \in \mathbb{R}^n}{\arg\min} \|D\widetilde{\mathbf{x}} - \mathbf{s}_i\|_2 \;\text{ such that }\; \|\widetilde{\mathbf{x}}\|_0 \leq \kappa. \tag{2.4}$$

Tale problema appartiene alla classe NP-Hard, e tipicamente problemi di questo tipo vengono risolti mediante *algoritmi greedy*. Uno dei possibili metodi di risoluzione è l'*Orthogonal Matching Pursuit* (OMP) descritto in [18].

Le anomalie vengono rilevate verificando se un battito può o meno venire ricostruito correttamente attraverso l'uso del dizionario D e una rappresentazione sparsa. In particolare, viene prima risolto il problema (2.4) ottenendo il vettore $\mathbf{x}_i$, e successivamente viene calcolato l'*errore di ricostruzione*:

$$r_i = \|D\mathbf{x}_i - \mathbf{s}_i\|_2. \tag{2.5}$$

Quest'ultimo viene poi utilizzato per discriminare se il battito s_i è stato generato dal processo $\mathcal{P}_A$ oppure dal processo $\mathcal{P}_N$. Infatti, valori elevati di $\mathbf{r}_i$ stanno ad indicare che il battito corrente non può essere ricostruito correttamente come combinazione sparsa delle colonne di D (le quali, come già specificato precedentemente, sintetizzano l'insieme di battiti generati dal processo $\mathcal{P}_N$). I *battiti anomali* vengono individuati verificando se r_i supera una determinata soglia $\gamma > 0$, la quale viene definita in modo empirico e determina il valore del *false positive rate* (FPR).

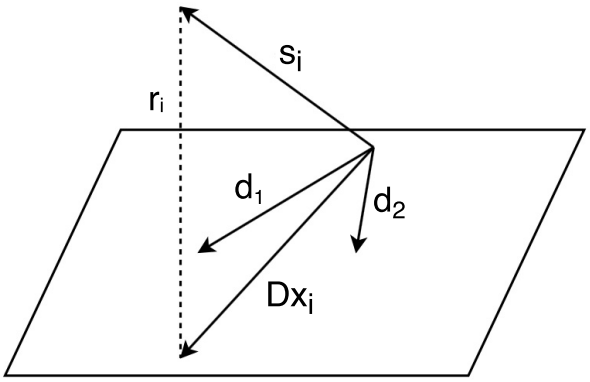

Figura 2.5: Semplificazione del processo di proiezione di $\mathbf{s}_i$ nel sottospazio contenente gli atomi di D. Il battito corrente viene ricostruito come combinazione lineare di $\mathbf{d}_1$ e $\mathbf{d}_2$ con il vettore $\mathbf{x}$, è inoltre riportato l'errore di ricostruzione r_i.

2.3.1 Orthogonal Matching Pursuit

L'Orthogonal Matching Pursuit (OMP) [18] è un algoritmo iterativo *greedy* utilizzato per trovare una soluzione al problema (2.4). Il suo pseudocodice è riportato nell'Algoritmo 1. Durante la fase di inizializzazione $\mathbf{r}_i$ viene posto uguale al vettore da ricostruire mediante D (nel caso dell'*anomaly detector* per ECG, l'*errore di ricostruzione* viene posto uguale al *battito corrente*, $\mathbf{r}_i = \mathbf{s}_i$). Ad ogni iterazione viene calcolata la similarità tra gli atomi del dizionario D non ancora selezionati nelle iterazioni precedenti e il vettore $\mathbf{r}_i$ (linea 3 dell'algoritmo). L'atomo maggiormente correlato viene selezionato (linea 4). Successivamente vengono calcolati i valori dei coefficienti $\mathbf{x}_I$ (linea 5) e aggiornato l'*errore di ricostruzione* (linea 6).

Algorithm 1 OMP

Require: Dictionary D, signal $\mathbf{s}$, target sparsity κ
Ensure: Sparse representation $\mathbf{x}$ such that $\mathbf{s} \approx \mathbf{D}\mathbf{x}$

1: Set $I := (\ )$, $\mathbf{r} := \mathbf{s}$, $\mathbf{x} := \underline{0}$
2: **while** (*Stopping criterion not met*) **do**
3: $\quad \hat{k} := \arg\max_k |\underline{d}_k^T \mathbf{r}|$
4: $\quad I := (\ I, \hat{k})$
5: $\quad \mathbf{x}_I := (\mathbf{D}_I)^+ \mathbf{s}$
6: $\quad \mathbf{r} := \mathbf{s} - \mathbf{D}_I \mathbf{x}_I$
7: **end while**

2.3.2 K-SVD

Il K-SVD [1] è un algoritmo iterativo utilizzato per risolvere il problema dell'apprendimento di D (2.3). Il suo pseudocodice è riportato nell'Algoritmo 2. Durante la fase di inizializzazione viene impostato il numero di iterazioni J e inizializzati il dizionario $D^{(0)}$ (o utilizzando vettori con componenti casuali, oppure scegliendo un sottoinsieme dal training set S).

Ad ogni iterazione vengono calcolati i coefficienti della matrice X attraverso l'algoritmo OMP (linea 3). Successivamente viene calcolato l'errore residuale rispetto all'atomo selezionato (linea 8) e vengono ottimizzati rispettivamente l'atomo corrente e il vettore dei coefficienti (linea 9). L'ultimo passo consiste nell'aggiornamento dell'atomo e del vettore dei coefficienti (linea 10), calcolati al passo precedente.

Algorithm 2 K-SVD

Require: Initial Dictionary $D^{(0)}$, training set $S = [\mathbf{s}_1, ..., \mathbf{s}_m]$, target sparsity T, number of iterations J

Ensure: Dictionary $D = D^{(j)}$, coefficient matrix X s.t. $\|X - S\|_F^2$ and $\|\mathbf{x}_i\|_0 \leq T$ for $i = 1, ..., n$

1: **while** $j < J$ **do**
2: **for** $I = 1, ..., m$ **do**
3: Compute $\mathbf{x}_i^{j+1} \arg\max_x \|D^{(j)}\mathbf{x} - \mathbf{s}_i\|_2$ s.t. $\|\mathbf{x}\|_0 \leq T$ (OMP stage)
4: **end for**
5: **for** $I = 1, ..., N$ **do**
6: Set $I =$ indices of signal in S whose reprentations use $\mathbf{d}_I^{(j)}$
7: Set $\mathbf{d}_I^{(j+1)} = \mathbf{0}$
8: Set $E = S_I - D^{(j+1)} X_I^{(j+1)}$
9: Compute $\{\hat{\mathbf{d}}_I, \hat{\mathbf{y}}\} = \arg\min_{d,y} \|E - \mathbf{d}\mathbf{y}^T\|_F$ s.t. $\|\mathbf{d}\|_2 = 1$
10: Update $\mathbf{d}_I^{(j+1)} = \hat{\mathbf{d}}$ and $X_{I,I} = \hat{\mathbf{y}}^T$
11: **end for**
12: Set $j = j + 1$
13: **end while**

2.4 Pipeline per l'Anomaly Detection

In questa Sezione verranno descritti gli step necessari a risolvere il problema dell'*anomaly detection* durante il monitoraggio cardiaco con il sistema descritto in precendenza. La pipeline si compone di due fasi, quella di *Training* e quella di *Test*, rappresentate graficamente rispettivamente nelle Figure 2.6 e 2.7. Nella Sezione 2.4.1 verranno descritti i passi della fase di Training, che è finalizzata all'apprendimento di D attraverso la risoluzione dell'equazione (2.3), mentre nella Sezione 2.4.2 verranno descritti i passi della fase di Testing finalizzata al rilevamento dei battiti anomali durante il monitoraggio in tempo reale.

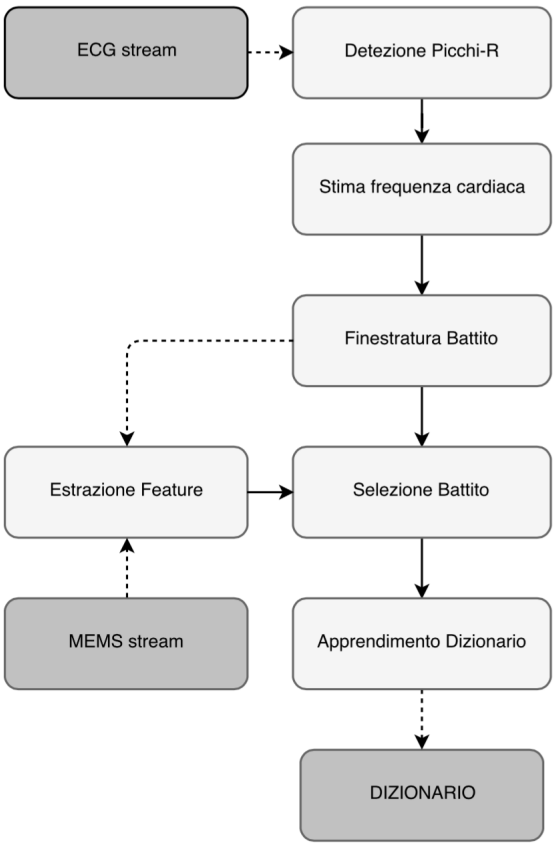

Figura 2.6: Schema a blocchi fase di training.

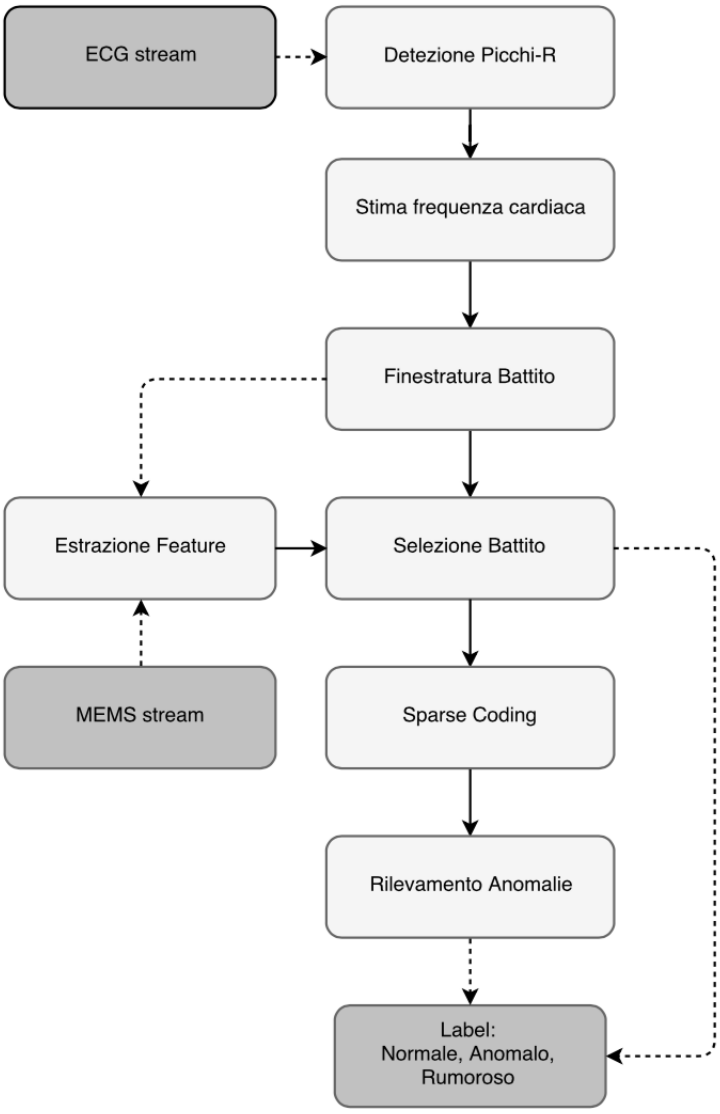

Figura 2.7: Schema a blocchi fase di test.

2.4.1 Training

L'obiettivo di questa fase è collezionare un numero sufficiente di battiti (derivanti dal processo $\mathcal{P}_N$) che verranno usati come input per risolvere il problema (2.3), generando così in uscita il dizionario D. La fase di training ha una durata di circa 20 minuti, durante i quali viene registrata l'attività cardiaca dell'utente a riposo. Oltre al tracciato ECG, vengono registrati i dati dei segnali MEMS (accelerometri e giroscopi) i quali vengono utilizzati per scartare i battiti rumorosi (generati dal processo $\mathcal{P}_A$) a causa di movimenti sporadici dell'utente durante l'acquisizione. Verranno ora descritti, in ordine, i passi che caratterizzano questa fase.

Detezione dei Picchi-R

Il tracciato ECG viene innanzitutto filtrato attraverso l'uso di due filtri mediani al fine di rimuovere la *baseline* (rumore di fondo che altera la struttura del segnale) e successivamente mediante un filtro passa-basso in modo da rimuovere i disturbi ad alta frequenza. Una volta ripulito il segnale ECG si procede ad individuare i picchi-R, utilizzando l'algoritmo *Pan-Tompkins* [17]. L'i-esimo picco-R verrà successivamente utilizzato come riferimento per il battito $\mathbf{s}_i$.

Stima della Frequenza Cardiaca

Per ogni picco-R viene misurata la frequenza cardiaca (FC), calcolando la distanza temporale tra due picchi consecutivi, denominata distanza RR. Ad ogni battito viene associata una frequenza cardiaca pari alla media delle ultime n (ad es. n=10) distanze RR.

Finestratura del Battito

Il battito $\mathbf{s}_i$ viene estratto dal tracciato ECG in corrispondenza dell'i-esimo picco-R, applicando una finestratura di dimensione variabile a seconda di FC. Ad ogni frequenza corrisponde una finestra specifica definita a priori. Questo è necessario perchè battiti a frequenze basse si estendono su intervalli temporali maggiori, e pertanto richiedono finestre più ampie, mentre battiti a frequenze alte risultano più rapidi, richiedendo finestre più piccole.

Estrazione delle Features di Moto

Questa fase, oltre a prendere in ingresso il tracciato ECG, sfrutta anche i segnali MEMS. In particolare, per ogni picco-R vengono estratte delle *features* rilevanti da entrambi i canali d'ingresso. Tali features sono volte a determinare se il battito corrente è corrotto dal movimento dell'utente. Le features verranno utilizzate nello step successivo per selezionare o scartare il battito corrente.

Selezione del Battito

In questa fase viene selezionato un certo numero di battiti $s_i \in \mathbb{R}^p$ ad una determinata frequenza cardiaca f_{TR}. I battiti selezionati vengono incolonnati in una matrice $S \in \mathbb{R}^{p \times m}$, la quale verrà poi utilizzata per la generazione del dizionario D, le cui colonne sintetizzeranno la morfologia dei battiti generati dal processo $\mathcal{P}_N$ alla frequenza $FC = f_{TR}$.

L'apprendimento del dizionario D richiede che i battiti abbiano tutti la stessa dimensione, ovvero siano acquisiti alla stesso frequenza cardiaca. Solitamente in condizioni di riposo FC varia tra 60 e 90 bpm (la variazione dipende dal sesso e dall'età dell'utente). È quindi necessario innanzitutto selezionare solo quei battiti aventi $FC = f_{TR}$.

Inoltre è importante che i battiti non siano soggetti ad effetti causati da movimenti durante l'acquisizione. In questa fase, le features di moto estratte dai dati MEMS e dal battito corrente vengono valutate per decidere se salvare o meno $\mathbf{s}_i$ in $R \in \mathbb{R}^{p \times m}$.

Il battito $\mathbf{s}_i$ viene selezionato se entrambe le seguenti condizioni sono verificate:

- Al battito $\mathbf{s}_i$ è associata la frequenza cardiaca f_{TR}.
- Il battito $\mathbf{s}_i$ non è soggetto a disturbi causati dal movimento.

Apprendimento del Dizionario

Questo è l'ultimo passo del processo di learning. Denotato con $S \in \mathbb{R}^{p \times m}$ il training set ottenuto mediante il processo di selezione, il dizionario D è ottenuto applicando l'algoritmo K-SVD.

2.4.2 Testing

L'obiettivo della fase di test è quello di rilevare i battiti anomali. Gli input di questa fase sono il tracciato ECG, il dizionario D appreso nella fase di learning e i dati campionati dai segnali MEMS. Questi ultimi vengono utilizzati per determinare quando la morfologia del battito corrente è soggetta ad effetti causati dal movimento. Quando ciò accade il battito viene scartato evitando così di generare false anomalie cardiologiche. Da questo momento in poi indicheremo come *anomali* quei battiti ben acquisiti (ovvero non soggetti ad effetti causati dal movimento) che mostrano una morfologia diversa rispetto a quelli generati dal processo $\mathcal{P}_N$. Al contrario, indicheremo come *rumorosi* quei battiti generati dal processo $\mathcal{P}_A$ a causa del movimento.

Innanzitutto il battito corrente $\mathbf{s}_i$ viene estratto dal tracciato ECG e viene controllato che esso non sia rumoroso. Successivamente viene risolto il problema dell'*anomaly detection* calcolando prima la rappresentazione sparsa di $\mathbf{s}_i$ rispetto a D attraverso (2.4), e successivamente il valore dell'*errore di ricostruzione* attraverso (2.5). Sulla base dell'errore di ricostruzione viene associata una *label* che identifica se il battito è anomalo o meno. Allo stesso modo della fase di training, anche la fase di testing è basata sull'analisi dei singoli battito. Ora verranno descritti i passi che caratterizzano questa fase.

Detezione dei Picchi-R

Come nella fase di training, anche in questa fase il tracciato ECG viene opportunamente filtrato con due filtri mediani (per rimuovere la *baseline*) e successivamente con un filtro a media mobile (per rimuovere i disturbi ad alta frequenza). Vengono poi individuati i picchi-R attraverso l'algoritmo *Pan-Tompkins* [17]. L'i-esimo picco-R verrà successivamente utilizzato come riferimento per il battito $\mathbf{s}_i$.

Stima della Frequenza Cardiaca

Anche in questo caso, per ogni battito viene calcolata la corrispondente FC. In particolare, al battito i-esimo viene associata la media degli ultimo n (per esempio, $n = 10$) intervalli RR.

Finestratura del Battito

Il battito $\mathbf{s}_i$ viene estratto prelevando un numero di *sample* corrispondenti ad una certa finestratura. Analogamente alla fase di training, la dimensione della finestra dipende dalla frequenza cardiaca.

Estrazione delle Features di Moto

Questo passo riceve in ingresso i segnali MEMS e il battito corrente, ed estrae estratte le features di moto. L'estrazione di tali features verrà descritta dettagliatamente nel Capitolo 5.

Selezione del Battito

Questo passo è volto a scartare i battiti soggetti ad effetti causati da movimenti durante l'acquisizione. Le features estratte al passo precedente vengono analizzate permettendo di classificare ogni singolo battito come rumoroso o meno. Al termine di questa analisi i battiti rumorosi verrano esclusi dal processo.

Calcolo della Rappresentazione Sparsa

In questa fase, dopo aver adattato D alla frequenza del battito corrente, si passa alla risoluzione del problema descritto in (2.4) utilizzando l'algoritmo OMP [18]. L'OMP riceve in ingresso il battito corrente $\mathbf{s}_i$, il dizionario D e genera in uscita la rappresentazione sparsa $\mathbf{x}_i$. Successivamente viene calcolato il valore dell'errore di ricostruzione attraverso (2.5).

Rilevamento delle Anomalie

Questo è l'ultimo passo della fase di test. L'errore di ricostruzione viene comparato con un valore soglia $\gamma > 0$ fissato in maniera empirica. Ad ogni battito viene associata una *label* che identifica se il battito corrente è *normale* o *anomalo*.

Capitolo 3

Motion Artifacts

Al fine di rendere il dispositivo indossabile Bio2Bit robusto ai movimenti dell'utente e quindi utilizzabile in scenari reali è opportuno saper distinguere le anomalie cardiache dai disturbi che alternano la struttura morfologica del battito. In particolare, movimenti compiuti dall'utente durante lunghe acquisizioni possono alterare la struttura morfologica del tracciato ECG generando così false anomalie. In questo Capitolo proporremo una soluzione in grado di predire il verificarsi di questo fenomeno. I battiti predetti come rumorosi verranno scartati, evitando di generare false anomalie in fase di testing attraverso l'anomaly detector. La predizione dei battiti anomali verrà affrontata come un problema di classificazione binaria dove, dato in ingresso il segnale dei MEMS e il battito corrente, vogliamo predire la classe di appartenenza del battito: C_1 (battito rumoroso) oppure C_2 (battito non rumoroso). Successivamente descriveremo le features estratte dai segnali MEMS e dal battito che verranno utilizzate per risolvere il problema della classificazione. Verrà poi affrontata la scelta del modello $\mathcal{K}$ da usare come classificatore, tenendo in considerazione che esso dovrà venire integrato su un sistema embedded low-power. In ultimo descriveremo un processo denominato *sequential future selection* che verrà utilizzato per selezionare il sottoinsieme di features maggiormente correlate con il fenomeno della generazione dei battiti rumorosi. Questo ci permetterà di estrarre meno *features*, minimizzando così il numero di operazioni richieste durante la fase di predizione.

3.1 Rumore causato dal movimento

Per poter utilizzare il dispositivo indossabile in scenari reali è necessario affrontare il problema della generazione delle false anomalie causate dai movimenti dell'utente. Movimenti bruschi (come ad es. la rotazione del tronco o repentini cambi di posizione) alterano la struttura morfologica del segnale ECG acquisito dal dispositivo, generando falsi allarmi durante il monitoraggio. In Figura 3.1 è riportato un esempio del fenomeno appena descritto. In particolare è possibile notare la correlazione tra movimenti effettuati dall'utente (catturati dalla variazione di intensità nel segnale accelerometrico, ovvero il segnale nella seconda riga), e la generazione di false anomalie (rappresentate dall'alterazione della struttura morfologica del segnale ECG, nella prima riga). L'attività cardiaca (rappresentata dal tracciato ECG) e fisica (rappresentata dal segnale accelerometrico) è stata registrata attraverso il dispositivo Bio2bit.

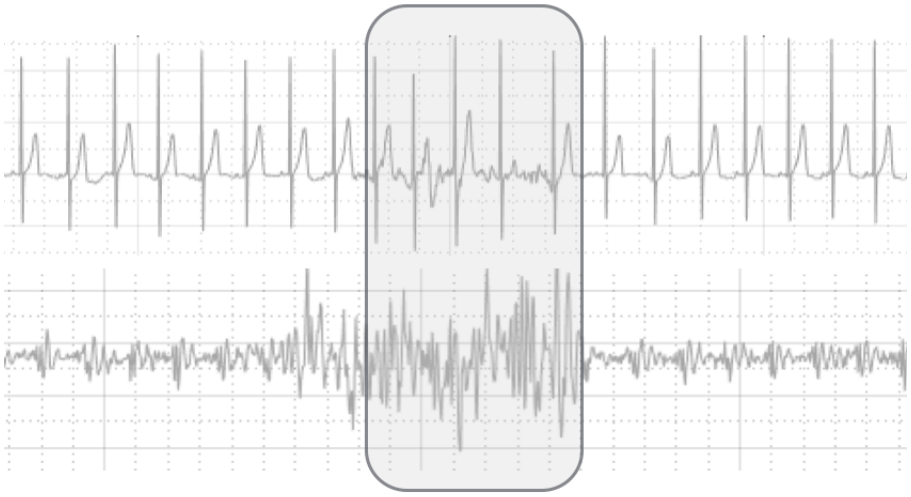

Figura 3.1: Porzione di tracciato ECG soggetto a disturbi a causa del movimento. Il segnale nella prima riga è l'ECG, mentre quello nella seconda riga rappresenta l'attività fisica registrata dagli accelerometri.

L'obiettivo di questa fase del lavoro era quello di integrare nel sistema d'acquisizione (STM32L476RG) un modulo software in grado di prevenire il verificarsi di queste situazioni, rilevando e scartando tutti quei battiti che potrebbero generarle.

In particolare, il rilevamento dei battiti rumorosi è stato affrontato come un problema di classificazione binaria, nel quale vogliamo associare ad ogni battito un'etichetta: $\langle rumoroso, non\ rumoroso \rangle$, osservando l'andamento dei segnali MEMS (accelerometri e giroscopi) e del battito corrente.

Anche in questo caso il problema è stato impostato secondo un approccio *data-driven*, dove, dopo aver collezionato un insieme di dati generati dal fenomeno da monitorare (falsi allarmi durante le registrazioni di tracciato ECG), abbiamo estratto un modello in grado di etichettare ogni singolo battito come rumoroso o meno. Il tutto è stato reso possibile grazie ad un software d'analisi di tracciati elettrocardiografici in grado di rilevare le porzioni di ECG corrotte da rumore.

Tale programma funziona in modalità off-line (ossia legge ed analizza un file sul quale è salvato l'ECG) e viene utilizzato come strumento di supporto da medici e cardiologi per facilitare l'analisi del tracciato. Esso è in grado di rilevare le regioni di tracciato ECG corrotte da rumore osservando solamente l'andamento del segnale elettrocardiografico. Il nostro obiettivo è quello di replicarne il funzionamento implementando un classificatore $\mathcal{K}$, il quale verrà utilizzato per predire la classe di appartenenza del battito corrente sulla basse dei segnali in ingresso (come detto, MEMS e battito corrente).

3.2 Formulazione Matematica del Problema

Denotiamo con $\mathbf{s}_i \in \mathbb{R}^p$ il battito corrente estratto dal tracciato ECG e con $\mathbf{M}_i \in \mathbb{R}^{q \times 6}$ la matrice contenente i dati dei segnali MEMS corrispondenti a tale battito. In particolare, $\mathbf{s}_i$ è un vettore centrato rispetto al picco-R del battito corrente (estratto dal tracciato ECG come descritto nel Capitolo 2), mentre $\mathbf{M}_i$ è una matrice le cui colonne contengono i *samples* estratti dagli accelerometri e dai giroscopi, anch'esse centrate temporalmente rispetto al picco-R. Le prime tre colonne di M_i contengono i *samples* relativi alle tre componenti dell'accelerazione lineare, mentre le ultime tre contengono i *samples* corrispondenti alle componenti dell'accelerazione angolare. Il problema della classificazione binaria può venire così descritto:

$$\{\mathbf{s}_i, \mathbf{M}_i\} \longrightarrow \langle 0, 1 \rangle \tag{3.1}$$

dove, $\{\mathbf{s}_i, \mathbf{M}_i\}$ rappresenta l'ingresso del classificatore $\mathcal{K}$, mentre $\langle 0, 1 \rangle$ sono i possibili valori d'uscita, 1 nel caso in cui il battito è rumoroso, 0 altrimenti. La lunghezza dei vettori colonna $\mathbf{m}^p$ appartenenti alla matrice $\mathbf{M}_i$ è fissata, e corrisponde al numero di campioni temporali raccolti. Eventualmente è possibile estrarre più matrici $\mathbf{M}$ (dai segnali MEMS) di dimensioni diverse, in modo da catturare l'effetto di movimenti che si protraggono per intervalli temporali differenti.

Al fine di poter risolvere il problema della classificazione dei battiti rumorosi sul dispositivo embedded ultra low-power (STM32L476RG) è di fondamentale importanza ridurre la dimensionalità dell'input del classificatore. Le limitate risorse del dispositivo infatti non permettono di operare direttamente sui segnali d'ingresso descritti in precedenza (il battito è costituito da 156 samples e i dati MEMS hanno una dimensione di circa 100 samples per ogni asse degli accelerometri e giroscopi). Nella prossima Sezione verranno presentate tutte le fasi necessarie alla risoluzione pratica del problema 3.1.

3.3 Soluzione Proposta

In questa Sezione presentiamo la soluzione che abbiamo sviluppato per il problema del rilevamento dei battiti rumorosi. Descriveremo innanzitutto le *features* estratte dai segnali d'ingresso (Sezione 3.3.1), successivamente presenteremo il modello utilizzato per classificare i battiti (Sezione 3.3.3) e in ultimo descriveremo la fase di *feature selection* utilizzata per ridurre la dimensionalità dell'input (Sezione 3.3.4).

3.3.1 Estrazione delle Features

Con il termine *feature extraction* si fa riferimento al processo tramite il quale si riduce la dimensionalità di un dato in input estraendone delle informazioni che lo descrivono in maniera sintetica. In letteratura esistono diversi studi sull'estrazione di *features* da segnali MEMS al fine di risolvere problemi di *context recognition* (ad es. per determinare il tipo d'attività svolta dell'utente [13]). In [6] viene presentata un'analisi dettagliata delle principali *features* estraibili dai segnali degli accelerometri e giroscopi, valutandone le caratteristiche e descrivendo i principali campi d'utilizzo.

Il nostro obiettivo è quello di selezionare le *features* migliori per risolvere il problema del rilevamento dei battiti rumorosi sul dispositivo STM32L476RG. In particolare abbiamo dovuto tenere in considerazione che tale processo di estrazione non deve richiedere un'elevata potenza computazionale e che allo stesso tempo le features devono essere tali da catturare le condizioni di moto che generano battiti rumorosi.

Ora presentiamo le *features* che abbiamo deciso di estrarre rispettivamente dai segnali MEMS e dal battito corrente.

Features Estratte dai Segnali MEMS

- *Media*: La media ($\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$) calcolata su una finestra di samples di ampiezza fissata è una metrica rappresentativa dell'andamento del segnale in quella finestra. Viene utilizza in maniera diretta o indiretta (ad es. in ambito *context recognition* viene sfruttata per determinare la postura dell'utente [10]).

- *Varianza*: La varianza ($\sigma^2 = \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}$) viene utilizzata come indicatore di stabilità del segnale in ingresso (sempre in [13] è utilizzata insieme ad altre *features* per determinare la posizione e l'attività compiuta dall'utente).

- *Root Mean Square*: La *Root Mean Square* (RMS) di un segnale x_i rappresentato da una sequenza di n valori discreti $\{x_1, x_2, ..., x_n\}$ viene calcolata come: $x_{RMS} = \sqrt{\frac{x(t)_1^2 + x(t)_2^2 + ... + x(t)_n^2}{n}}$. Anch'essa è una *feature* utilizzata in ambito *activity recognition* (ad es. in [15] costituisce il dato in ingresso di una rete neurale).

- *Signal Magnitude Area*: La Signal Magnitude Area (SMA è una metrica utilizzata per distinguere lo stato di riposo da quello di attività motoria. Viene calcolata come: $SMA = \frac{1}{t}(\int_0^t \|x(t)\|dt + \int_0^t \|y(t)\|dt + \int_0^t \|z(t)\|dt)$, dove x, y e z corrispondono rispettivamente alle accelerazioni sui tre assi (in [11] la SMA viene utilizzata per determinare i momenti in cui l'utente sta compiendo un'attività fisica).

- *Signal Vector Magnitude*: La Signal Vector Magnitude (SVM) viene calcolata come: $SVM = \frac{1}{n}\sum_{i=1}^{n}\sqrt{x(t)_i^2 + y(t)_i^2 + z(t)_i^2}$. Anch'essa viene utilizzata in ambito *activity recognition* per classificare l'attività svolta dell'utente. Solitamente rappresenta, insieme ad altre *features*, l'ingresso di un classificatore (ad es. una rete neurale come in [11]).

- *Mean-Crossing*: La Mean-Crossing è definità come il numero di volte in cui il segnale oltrepassa la media in una finestra di samples di dimensione fissata (In ambito *activity recognition* viene utilizzata ad es. per discriminare la camminata dalla corsa [5]).

Features Estratte dal Battito

Le *features* estratte dal battito devono essere tali da catturare i disturbi causati dal movimento senza influenzare in alcun modo la fase successiva di rilevamento delle anomalie. In particolare, dovranno sintetizzare l'effetto del rumore (causato dal movimento) evitando di catturare gli effetti che alterano la struttura del battito a causa di anomalie morfologiche. A tale scopo risulta efficace estrarre da $\mathbf{s}_i$ (il battito corrente) le seguenti due *features*:

- *Media del battito*: La media ($\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$) del battito viene calcolata come media dei *samples* appartenenti al vettore $\mathbf{c}_i$ estratto dal tracciato ECG e centrato rispetto al picco-R del battito corrente ($\mathbf{s}_i$). Il vettore $\mathbf{c}_i$ potrà avere dimensione differente rispetto a $\mathbf{s}_i$ in modo da catturare l'effeto di movimenti che si protraggono su intervalli temporali maggiori rispetto alla durata di $\mathbf{s}_i$.

- *Media isoelettrica*: La linea isoelettrica è rappresentata dal livello orizzontale di registrazione nel momento in cui non vi è attività cardiaca

(durante l'intervallo tra l'onda T e l'onda P). La media isoelettrica ($\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$) viene calcolata come media dei *samples* appartenenti a questo tratto del tacciato ECG.

3.3.2 Formulazione finale del Problema

A questo punto il problema della classificazione può essere descritto come:

$$\{\mathbf{f}_i\} \longrightarrow \langle 0, 1 \rangle, \tag{3.2}$$

dove $\mathbf{f}_i \in \mathbb{R}^p$ è il nuovo ingresso ottenuto estraendo le *features* descritte nella Sezione precedente dai segnali $\mathbf{s}_i$ e $\mathbf{M}_i$. In particolare, per ogni picco-R (rappresentativo del battito corrente) estraiamo dal tracciato ECG il vettore $\mathbf{s}_i$ e dai segnali MEMS la matrice $\mathbf{M}_i$. Successivamente a partire da $\mathbf{M}_i$ vengono calcolati due vettori $\mathbf{acc\_3d}_i$ e $\mathbf{gyr\_3d}_i$. Nello specifico il vettore $\mathbf{acc\_3d}_i$ viene ricavato dai primi tre vettori colonna di $\mathbf{M}_i$ (contenenti, ricordiamo, i *samples* delle accelerazioni lineari x,y e z) utilizzando la formula:

$$\mathbf{acc\_3d}(j)_i = \sqrt{\mathbf{M_i}(j,1)^2 + \mathbf{M_i}(j,2)^2 + \mathbf{M_i}(j,3)^2}. \tag{3.3}$$

Il vettore $\mathbf{gyr\_3d}_i$ viene ricavato in modo analogo dalle ultime tre colonne di $\mathbf{M}_i$.

In ultimo, il vettore $\mathbf{f}_i$ è ottenuto estraendo le *features* descritte nella Sezione 3.3.1 dai vettori $\mathbf{acc\_3d}_i$ e $\mathbf{gyr\_3d}_i$, ed aggiungendo le *features* estratte dal battito corrente $\mathbf{s}_i$.

3.3.3 Classificatore

In generale, l'obiettivo di un problema di classificazione [4] è quello di associare un vettore di ingresso $\mathbf{x}$ a una tra K classi possibili C_k, dove $k = 1, ..., K$. Nel nostro caso specifico, vogliamo associare il vettore d'ingresso $\mathbf{f}_i$ ad una delle due classi d'uscita: C_1 (battiti rumorosi), C_2 (battiti non rumorosi). Si parla in questo caso di un problema di classificazione binaria, dove il classificatore divide lo spazio di input in due regioni, chiamate *decision regions*, i cui confini vengono indicati come *decision boundaries*.

Linear Discriminant Function

Uno dei metodi più semplici per risolvere il problema della classificazione binaria è basato sulla creazione di un modello discriminativo chiamato *discriminant function*. Questo modello è descritto da una combinazione lineare dell'input $\mathbf{x}$ attraverso un vettore $\mathbf{w}$ detto *vettore dei pesi*. La fase di predizione della classe di appartenenza avviene valutando il segno della *discriminant function*:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0, \tag{3.4}$$

dove w_0 è comunemente indicato come *bias*. Il dato in ingresso viene associato alla classe C_1 se $y(x) \geq 0$ altrimenti alla classe C_2 se $y(x) < 0$. Il *decision boundary* è definito dalla relazione $y(x) = 0$, la quale corrisponde all'iperpiano di dimensione $D-1$ (dove D rappresenta la dimensionalità dello spazio di input). L'impiego di questo modello risulta adatto per risolvere problemi di classificazione su sistemi embedded low-power (come STM32L476RG), dove le risorse limitate del dispositivo non permetterebbero l'utilizzo di algoritmi con elevata complessità computazionale. Lo svantaggio però è legato alla linearità del modello, che se da una parte risulta d'essere di facile interpretazione e implementazione, dall'altra non è in grado di descrivere superfici di separazione complesse.

Perceptrone

Un altro modello adatto a risolvere problemi di classificazione binaria su sistemi low-power è il *percptrone* di Rosemblatt [20]. In questo caso, la classe d'uscita viene predetta secondo la relazione:

$$y(\mathbf{x}) = f(\mathbf{w}^T \phi(\mathbf{x})), \tag{3.5}$$

dove $\phi(\cdot)$ è una trasformazione non lineare applicata al dato d'ingresso $\mathbf{x}$, mentre $f(\cdot)$ è una *step function* definita come:

$$f(a) = \begin{cases} +1 & a \geq 0 \\ -1 & a < 0. \end{cases}$$

Questo modello risolve il problema della classificazione binaria valutando il segno della combinazione lineare $\mathbf{w}^T \phi(\mathbf{x})$. Lo svantaggio anche in questo caso è dovuto dalla linearità del modello.

Rete Neurale Feed-Forward

Al fine di risolvere il problema della classificazione dei battiti rumorosi è opportuno utilizzare un modello parametrico capace di descrivere relazioni non lineari. La nostra scelta è ricaduta su una rete neurale (NN) feed-forward con un singolo hidden-layer. Verrà ora data una descrizione sintetica degli elementi che caratterizzano una rete neurale. Per una descrizione più approfondita si veda [3].

Una rete neurale è definita da un insieme di neuroni connessi secondo una specifica topologia. Gli elementi che caratterizzano la rete sono:

- *Input layer*: Insieme di neuroni che riceve in ingresso i dati da processare.
- *Output layer*: Insieme di neuroni che generano in uscita il risultato della rete.
- *Hidden layer*: Insieme di neuroni che riceve in ingresso i dati generati dal layer precedente e genera in uscita l'ingresso per il layer successivo.

In una NN feed-forward il segnale si propaga partendo dall'input layer fino all'output layer. Ogni singolo neurone è caratterizzato da una funzione di attivazione non lineare $h(\cdot)$, solitamente definita come una *Sigmoide*:

$$h(a) = \sigma(a) = \frac{1}{1 + exp(-a)}. \tag{3.6}$$

La nostra Architettura di Rete

In Figura 3.2 è riportata la rete neurale che abbiamo utilizzato per risolvere il problema della classificazione dei battiti rumorosi. Essa è costituita da un input layer che riceve in ingresso il vettore di features $\mathbf{f} \in \mathbb{R}^r$ relativo al battito corrente, un singolo hidden layer caratterizzato da un numero di neuroni pari a $5r$ e avente come funzione di attivazione h la sigmoide. L'hidden layer processa l'input generando in uscita l'ingresso dell'output layer. L'output layer è caratterizzato da un singolo neurone avente come funzione d'attivazione g la sigmoide. Complessivamente, l'uscita della rete è descritta dalla relazione:

$$y(\mathbf{f}) = g(\sum_{j=1}^{5r} W_j \cdot h(\sum_{i=1}^{r} w_{ji}\mathbf{f})). \tag{3.7}$$

Si osservi che abbiamo utilizzato una notazione specifica per i pesi, in particolare indichiamo con $\{w_i\}$ i pesi relativi all'hidden layer, e con $\{W_j\}$ i pesi relativi all'output layer. La Formula 3.7 indica la probabilità che il battito corrente appartenga alla classe C_1 (dei battiti rumorosi), oppure alla classe C_2 (dei battiti non rumorosi). Per quanto riguarda il setting dei parametri, il numero di neuroni dell'hidden layer è stato fissato tramite un tuning manuale, mentre la fase d'apprendimento dei pesi verrà descritta nel Capitolo 4 Sezione 4.3.

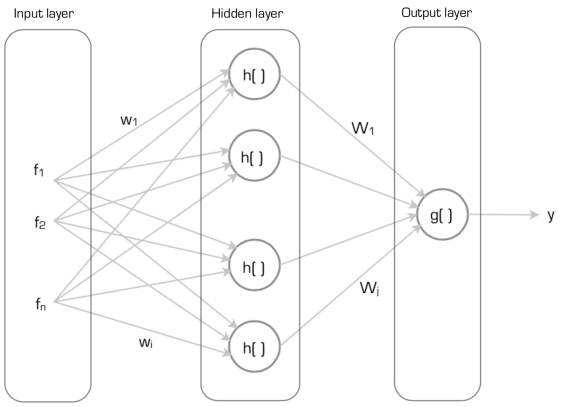

Figura 3.2: Rappresentazione grafica della rete neurale utilizzata per risolvere il problema della classificazione dei battiti rumorosi. Essa è costituita da un input layer che riceve in ingresso il vettore $\mathbf{f}$ *contenente le features del battito corrente e genera in uscita un valore rappresentante la classe di appartenenza: rumoroso o non rumoroso.*

3.3.4 Selezione delle Features

Una volta aver definito il modello è possibile procedere alla selezione delle *features* maggiormente correlate con l'output. Lo scopo di questo processo è quello di ridurre la dimensionalità dell'input (nel nostro caso la dimensione del vettore $\mathbf{f}$), in modo da minimizzare il numero di operazioni eseguite dal classificatore $\mathcal{K}$ in fase di predizione.

La selezione delle *features* (nota come *feature selection* [7]) consiste nel trovare il miglior sottoinsieme nello spazio di input tale da minimizzare l'errore di predizione. Sostanzialmente, quello che ci poniamo di fare è selezionare le miglior componenti del vettore $\mathbf{f}$ (ingresso del classificatore) in grado di risolvere il problema della predizione dei battiti rumorosi. Uno dei metodi comunemente utilizzati per la costruzione del miglior sottoinsieme è la *sequential feature selection* [9]. Questo è un algoritmo *greedy*, che ad ogni iterazione seleziona la miglior *feature* (ovvero la migliore componente del vettore $\mathbf{f}$) valutando una funzione di costo. L'algoritmo termina quando la selezione di nuove *features* non porta ad alcun miglioramento nella fase di predizione. Solitamente nel caso dei problemi di classificazione viene scelto come criterio di arresto la valutazione dell'errore di misclassificazione (ossia il numero di dati classificati in maniera scorretta dal classificatore). L'algoritmo termina quando l'aggiunta di nuove *features* non permette di ridurre tale errore.

Capitolo 4

Esperimenti sui Motion Artifacts

In questo Capitolo verranno presentati i risultati ottenuti dal classificatore $\mathcal{K}$ per risolvere il problema della determinazione dei battiti rumorosi. Innanzitutto verrà descritta la creazione del dataset dal quale è stato possibile apprendere il modello. Successivamente verranno presentate le *features* estratte dal processo di *sequential features selection*, descritto nella Sezione 3.3.4 Capitolo 3. In ultimo presenteremo il *framework* utilizzato per addestrare e misurare le performance del modello.

4.1 Generazione del Dataset

In questa sezione presenteremo il processo di creazione del dataset ($\mathcal{D}$) utilizzato per l'addestramento della rete neurale descritta nel Capitolo 3. Al fine di poter apprendere il modello che permette di classificare correttamente i battiti come rumorosi o meno, è necessario disporre di un dataset etichettato, contenente esempi del fenomeno che si vuole descrivere, ovvero le anomalie generate dai movimenti. In particolare nel nostro caso $\mathcal{D}$ avrà la seguente struttura:

$$\mathcal{D} = \langle \mathbf{F}, \mathbf{t} \rangle, \tag{4.1}$$

dove $\mathbf{F} \in \mathbb{R}^{r \times m}$ è una matrice le cui colonne ($\mathbf{f} \in \mathbb{R}^r$) contengono le *features* relative al battito i-esimo (mentre m indica il numero totale di battiti), in particolare le r features considerate sono quelle descritte nel Capitolo 3 Sezione 3.3.1. Mentre $\mathbf{t} \in \mathbb{N}^m$ è un vettore avente cardinalità pari al numero di colonne di $\mathbf{F}$, dove ogni elemento contiene la classe di appartenenza del battito i-esimo (classe C_1: battito rumoroso, classe C_2: battito non rumoroso).

La creazione di $\mathcal{D}$ è stata effettuata registrando l'attività cardiaca (segnale ECG) e fisica (segnali degli accelerometri e giroscopi) di 15 utenti attraverso il dispositivo indossabile Bio2Bit.

Durante la registrazione ad ogni utente è stato chiesto di seguire un protocollo (riportato in Tabella 4.1) costituito da una sequenza di movimenti. Questi sono stati definiti selezionando dei movimenti molto comuni che possono essere compiuti nella quotidianità (come ad es. alzarsi dalla sedia o prendere degli oggetti con la mano). Il protocollo è stato suddiviso in due parti, nella prima l'utente esegue una serie di movimenti da seduto, mentre nella seconda parte, gli stessi movimenti vengono ripetuti camminando.

Al termine della registrazione il tracciato è stato analizzato attraverso un apposito software di controllo in grado di associare ad ogni singolo battito una tra le seguenti etichette: normale, anomalo o rumoroso. Le etichette assegnate dal software sono state validate da un medico specialista. Successivamente, per ogni battito etichettato come normale o rumoroso, sono state estratte le *features* elencate nella Sezione 3.3.1. Per ogni utente u è stato ottenuto uno specifico dataset $\mathcal{D}_u = \langle \mathbf{F}_u, \mathbf{t}_u \rangle$. Infine i dataset sono stati raggruppati in un unico insieme contenente i dati da tutte le 15 registrazioni.

Tabella 4.1: Protocollo movimenti per la creazione del dataset

Attività	Durata
Parte 1: Movimenti da seduto	
Seduto	15 sec
Alzarsi e sedersi	una volta (3 sec)
Rotazione del tronco	sei volte (15 sec)
Seduto	15 sec
Allacciare le stringhe	2 volte (15 sec)
Prendere 6 oggetti a destra e sinistra	15 sec
Seduto	15 sec
Sbadigliare	5 sec
Seduto	15 sec
Parte 2: Movimenti durante la camminata	
Camminare	15 sec
Rotazione del tronco	sei volte (15 sec)
Allacciare le stringhe	2 volte (15 sec)
Camminare	15 sec
Prendere 6 oggetti a destra e sinistra	15 sec
Camminare	15 sec
Sbadigliare	5 sec
Camminare	15 sec

4.1.1 Etichettatura del Dataset

Al fine di facilitare la supervisione da parte di un medico specialista per validare le etichette assegnate dal software di analisi, abbiamo sviluppato un' interfaccia grafica in Matlab. Tale interfaccia permette rispettivamente di caricare la registrazione precedentemente acquisita con il dispositivo Bio2Bit (salvata in un file binario contenente i dati del tracciato ECG e dei segnali MEMS), filtrare il segnale elettrocardiografico (con gli appositi filtri mediani e passa basso, come descritto nel Capitolo 2 Sezione 2.4.1), rilevare i picchi-R dei battiti presenti nel tracciato (attraverso l'algoritmo *Pan Tompkins* [17]), e modificare le etichette assegnate dal software d'analisi. La modifica avviene semplicemente cliccando sul battito che si desidera modificare. In ultimo, l'interfaccia permette di visualizzare i risultati ottenuti con il classificatore $\mathcal{K}$, mettendo a confronto le etichette assegnate dal classificatore rispetto al target $\mathbf{t}$ (rumoroso, non rumoroso) e mostrando

l'andamento dei segnali MEMS in modo sincronizzato rispetto al tracciato ECG. Grazie a questa interfaccia è stato possibile ottenere agevolmente il dataset $\mathcal{D}$ utilizzato poi come *ground truth* per l'apprendimento del modello, ed in particolare correggere e validare le etichette contenute nel vettore $\mathbf{t}$.

4.2 Feature Selection

Questa fase consiste nell'applicare l'algoritmo *sequential feature selection* (descritto nel capitolo 2 sezione 3.3.4) in modo da selezionare le features migliori a partire dal vettore $\mathbf{f}$ (vettore contenente tutte le features). L'algoritmo riceve in ingresso il dataset $\mathcal{D}$, composto dalla matrice $\mathbf{F}$ e dal *target vector* $\mathbf{t}$, e seleziona iterativamente le righe di $\mathbf{F}$ (ovvero le *features*) maggiormente correlate con l'etichettatura del fenomeno (battito rumoroso, non rumoroso). La selezione viene effettuata minimizzando una funzione di costo. Nel nostro caso è stato scelto l'errore di misclassificazione, calcolato come la somma dei battiti classificati in maniera scorretta dal modello.

In particolare, ogni feature (definita come nel capitolo 3 sezione 3.3.4) è stata estratta per 4 volte considerando 4 intervalli temporali differenti, rispettivamente di 0.5, 1, 1.5 e 2 secondi, in modo tale da catturare le variazioni dei segnali MEMS dovute a movimenti dell'utente che si protraggono per tempi diversi. L'algoritmo *sequential feature selection* selezionerà poi le features appartenenti all'intervallo migliore per descrivere il fenomeno.

Verrà ora descritto nel dettaglio il processo di estrazione delle features dai segnali ECG e MEMS. Il segnale elettrocardiografico viene acquisito dal dispositivo Bio2Bit ad una frequenza di campionamento di 256 Hz, mentre le accelerazioni lineari e angolari vengono acquisite ad una frequenza di campionamento di 64 Hz. Ne deriva che in corrispondenza di ogni picco-R (relativo al battito corrente) vengono estratti 4 vettori (relativi ai 4 intervalli temporali) di dimensione rispettivamente 128, 256, 384 e 512 contenenti i samples del tracciato ECG. Successivamente, da ognuno di questi vettori viene estratta la *media del battito* (secondo la definizione del Capitolo 3 Sezione 3.3.4). Un procedimento analogo è applicato ai segnali degli accelerometri e giroscopi. In questo caso, per ognuna delle 3 componenti dei segnali, vengono estratti 4 vettori (centrati rispetto al picco-R del battito corrente) di dimensione rispettivamente 32, 64, 96 e 128. Successivamente, per ognuno di questi intervalli temporali, sono calcolati i vettori **acc_3d** e **gyr_3d** come descritto nel Capitolo 3 Sezione 3.3.1. Infine vengono estratte le *features dei segnali MEMS* (secondo le definizioni del Capitolo 3 Sezione

3.3.4).

4.2.1 Risultati Feature Selection

Presenteremo ora i risultati ottenuti mediante il processo di *sequential feature selection*. In tabella 4.2 sono riportate le features estratte applicando l'algoritmo al dataset: $\mathcal{D}_{rest} = \langle \mathbf{F}_{rest}, \mathbf{t}_{rest} \rangle$ dove, $\mathbf{F}_{rest}$ è una matrice le cui colonne contengono le features estratte dai segnali MEMS e dal segnale ECG registrati durante la prima parte del protocollo descritto in tabella 4.1 (movimenti eseguiti dall'utente da seduto).

Tabella 4.2: Features relative ai movimenti da seduto

Feature	Dimensione finestra
Root Mean Square Gyroscope	0.5 sec
Signal Magnitude Area	2 sec
Varianza Accelerometri	2 sec
Media Battito	0.5 sec
Media Battito	2 sec

In tabella 4.3 sono riportare le features estratte applicando l'algoritmo *sequential feature selection* al dataset: $\mathcal{D}_{walk} = \langle \mathbf{F}_{walk}, \mathbf{t}_{walk} \rangle$, dove in questo caso $\mathbf{F}_{walk}$ è la matrice delle features estratte dai segnali MEMS e dai battiti relativi alla seconda parte del protocollo descritto in tabella 4.1 (movimenti eseguiti durante la camminata).

Tabella 4.3: Features relative ai movimenti durante la camminata

Feature	Dimensione finestra
Root Mean Square Gyroscope	0.5 sec
Media Accelerometri	0.5 sec
Signal Magnitude Area	2 sec
Varianza Accelerometri	2 sec
Media Battito	0.5 sec
Media Battito	2 sec

4.3 Addestramento Classificatore

La fase successiva alla creazione del dataset consiste nell'apprendimento del modello in grado di descrivere la relazione tra l'ingresso (**F**) e l'uscita (**t**). Il modello scelto per risolvere il problema della classificazione dei battiti rumorosi è una rete neurale feed-forward. Quest'ultima appartiene alla classe dei modelli parametrici, nei quali la fase di apprendimento consiste nel determinare il valore dei parametri liberi ($\{w_i\}$ e $\{W_j\}$ della rete) che meglio permettono di descrivere il fenomeno (relazione tra i movimenti dell'utente e battiti rumorosi).

4.3.1 Apprendimento della Rete

Daremo ora una sintetica descrizione del processo d'apprendimento dei pesi di rete neurale estratto da [3].

Definiamo la *cross-entropy* nel modo seguente:

$$L(\mathbf{w}) = -\sum_{i=1}^{m} t_i \ln y_i(\mathbf{w}) + (1 - t_i) \ln(1 - y_i((\mathbf{w}))). \tag{4.2}$$

Essa rappresenta la funzione di costo utilizzata per apprendere i pesi della rete nel caso in cui essa venga adottata per risolvere problemi di classificazione. In particolare t_i rappresenta la classe di appartenenza dell' i-esimo battito ($t_i = 1$ identifica un battito rumoroso e $t_i = 0$ un battito non rumoroso), mentre y_i rappresenta la classe predetta in uscita della rete. Nel caso specifico della nostra architettura di rete (descritta nella Sezione 3.3.3), l'output della rete è definita come (3.7). La *cross-entropy* si ottiene applicando il logaritmo alla probabilità congiunta, espressa come:

$$\prod_{i=1}^{m} y_i^{t_i} (1 - y_i)^{1-t_i}, \tag{4.3}$$

dove l'output della rete y_i viene interpretata come la probabilità condizionata $p(\mathcal{C}_1|\mathbf{x})$ nel caso in cui il battito i-esimo appartenga alla classe C_1, altrimenti come $1 - y_i$ nel caso in cui appartenga alla classe C_2. La fase d'apprendimento consiste nel determinare il valore dei parametri liberi (pesi della rete) che massimizzano 4.2:

$$\arg\max_{\mathbf{w}} L(\mathbf{w}) = \arg\min_{\mathbf{w}} -\sum_{i=1}^{m} t_i \ln y_i(\mathbf{w}) + (1 - t_i) \ln(1 - y_i(\mathbf{w})). \tag{4.4}$$

Un modo di procedere è quello di applicare la tecnica del *gradient descent* [14], un algoritmo che iterativamente calcola il valore dei pesi al fine di risolvere (4.4).

4.3.2 Framework per addestrare il modello

In questa sezione descriveremo il framework utilizzato per addestrare e misurare le performance del modello appreso. Innanzitutto il dataset $\mathcal{D}$ è stato permutato in modo da mescolare i dati dei singoli utenti, successivamente esso è stato suddiviso casualmente in k sottoinsiemi, $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2, ..., \mathcal{D}_k$. Il modello è stato appreso iterativamente da $k-1$ insiemi, e le performance sono poi state valutate sul k-esimo insieme escluso. Specificamente, il dataset $\mathcal{D}$ è stato suddiviso in 10 parti, il modello è stato appreso iterativamente dall'unione di nove sottoinsiemi (ad ogni iterazione l'ultimo sottoinsieme viene sostituito con quello escluso), e le performance sono state poi valutate sul decimo escluso. Per ogni modello appreso è stata calcolata la corrispettiva *confusion matrix*, una tabella che permette di descrive graficamente le performance di un classificatore. In Figura 4.3.2 è riportata la struttura di tale tabella. Infine, le performance sono state valutate mediando le 10 confusion matrix.

Tabella 4.4: Confusion Matrix

		Classe predetta	
		Positivo	Negativo
Classe reale	Positivo	Veri Positivi (VP)	Falsi Negativi (FN)
	Negativo	Falsi Positivi (FP)	Veri Negativi (VN)

Nel nostro caso, i Veri Positivi (VP) sono i battiti non soggetti a rumore classificati come tali dal modello, i Veri Negativi (VN) sono i battiti affetti da rumore (causato dal movimento) e riconosciuti dal classificatore. I Falsi Positivi (FP) sono tutti quei battiti soggetti a disturbi che il modello classifica come normali, in ultimo i Falsi Negativi (FN) sono i battiti normali (non soggetti a disturbi) che il classificatore etichetta come rumorosi. L'obiettivo principale è quello di ridurre i falsi allarmi generati dal movimento evitando che eventuali anomalie vengano scartate in fase di classificazione.

Verranno ora riportati i risultati ottenuti con il classificatore $\mathcal{K}$ in due diversi casi. Nel primo caso verrà mostrata la *confusion matrix* ottenuta addestrando e testando il classificare con il dataset $\mathcal{D}_{rest}$ contenente le *features* definite nella tabella 4.2. Nel secondo caso, verrà presentata la *confusion matrix* ottenuta utilizzando il dataset $\mathcal{D}_{walk}$ contenente le *features* riportate in Tabella 4.3.

Tabella 4.5: Confusion Matrix
Movimenti da seduto

		Classe predetta	
		Positivo	Negativo
Classe reale	Positivo	92.47% (2.63%) (VP)	7.53% (2.63%) (FN)
	Negativo	23.14% (17.98%) (FP)	76.86% (17.98%) (VN)

Dai risultati mostrati in Tabella 4.5 si evince che il classificatore è in grado di ridurre i falsi allarmi generati dal movimento durante il monitoraggio (Il 76.86% dei battiti rumorosi viene rilevato come tale). La riduzione dei falsi allarmi avviene a discapito del 7.53% dei battiti non rumorosi che il classificatore sbaglia a etichettare, tra questi potrebbero esserci le anomalie cardiache che non verrebbero così analizzate dall'*anomaly detector*.

Tabella 4.6: Confusion Matrix
Movimenti camminando

		Classe predetta	
		Positivo	Negativo
Classe reale	Positivo	95.78% (8.67%) (VP)	4.22% (8.67%) (FN)
	Negativo	41.50% (44.73%) (FP)	58.50% (44.73%) (VN)

In questo caso (Tabella 4.6) il classificatore non è in grado di classificare correttamente i battiti rumorosi (solo il 58.50% dei battiti rumorosi viene riconosciuto come tale), mentre i battiti aventi una corretta morfologia vengono classificati in maniera corretta. Il peggioramento delle prestazioni in questo caso è dovuto alla "semplicità" delle *features* estratte dai segnali MEMS e ECG che, se da un lato non richiedono un elevata potenza di calcolo, dall'altro non permettono di discriminare puntualmente i battiti rumorosi in condizioni di moto. Potenzialmente si potrebbe utilizzare un maggior numero di *features* più "pesanti" per migliorare le prestazioni, tuttavia questo non sarebbe compatibile con le caratteristiche del nostro device (scheda NUCLEO STM32L476).

In conclusione, i risultati ottenuti mostrano che il classificatore riduce in modo considerevole il numero di falsi allarmi generati in condizioni di movimenti da seduto.

Capitolo 5

Sistema Hardware e Software

In questo Capitolo verrà analizzato il sistema di monitoraggio del tracciato elettrocardiografico basato sulla scheda di sviluppo NUCLEO STM32L476RG e sul dispositivo indossabile Bio2Bit. Inizialmente verranno elencate e descritte le componenti hardware che caratterizzano il sistema. Successivamente verrà descritta l'architettura software attualmente presente sulla scheda NUCLEO STM32L476RG. In particolare verranno analizzati i moduli che la costituiscono e le modalità in cui interagiscono tra loro.

5.1 Architettura del sistema di monitoraggio

Attualmente il sistema di acquisizione è costituito dal Bio2Bit, prototipo progettato e sviluppato da STMicroelectronics, il quale è in grado di acquisire il tracciato ECG, oltre ad altri parametri fisici, e di trasmetterli via Bluetooth Low Energy alla scheda NUCLEO STM32L476RG, equipaggiata con l'espansione board X-NUCLEO-IDB05A1 (Modulo Bluetooth Low Energy).

Il Bio2Bit è in grado di operare in due modalità differenti:

- *Streaming*: in questa modalità il tracciato elettrocardiografico e altri parametri fisici (ad es. accelerazioni lineari e angolari) vengono acquisiti e trasmessi direttamente, via Bluetooth Low Energy, alla NUCLEO STM32L476RG.
- *Monitoring*: in questa modalità i dati vengono acquisiti e memorizzati completamente sul dispositivo. Al termine dell'acquisizione è possibile trasmetterli alla NUCLEO STM32L476RG.

La logica applicativa presente sulla NUCLEO STM32L476RG permette di comunicare via Bluetooth con il dispositivo Bio2Bit, di ricevere lo streaming dati ECG e di calcolare la frequenza cardiaca. Il nostro obiettivo è quello di integrare nel sistema un modulo software in grado di analizzare il tracciato elettrocardiografico e rilevare eventuali *anomalie* (ad es. le *aritmie*). Quest'ultime, successivamente, potranno essere trasmettesse ad un server, sul quale sarà possibile utilizzare i dati ricevuti per estrapolare informazioni utili (ad es. il momento della giornata in cui si sono presentate le anomalie) e creare uno storico dello stato medico-cardiaco del paziente.

In Figura 5.1, è riportato lo schema a blocchi del sistema di acquisizione e della trasmissione dati. Il Bio2Bit acquisisce lo streaming dati ECG e lo invia alla scheda NUCLEO STM32L476RG (via Bluetooth Low Energy). Il tracciato ECG ricevuto viene analizzato rilevando eventuali anomalie. Queste informazioni vengono poi trasmesse al sever.

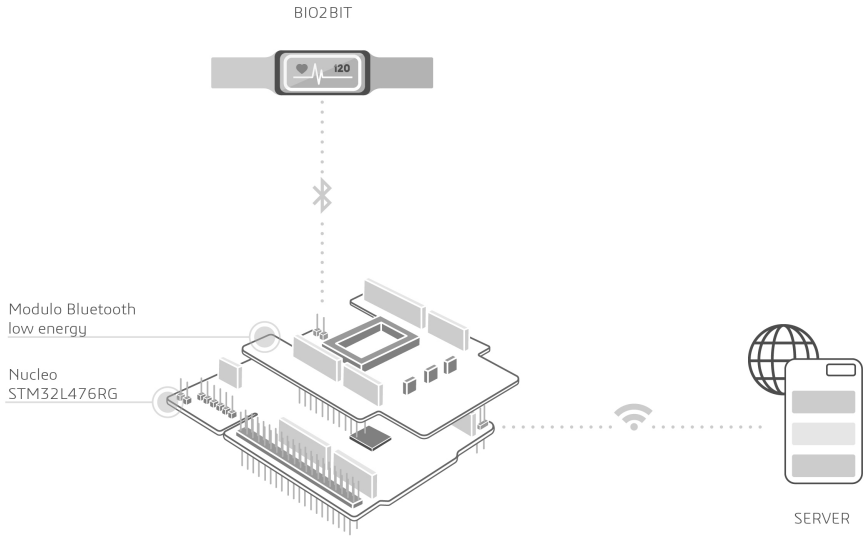

Figura 5.1: Sistema di acquisizione e monitoraggio del segnale ECG basato sul dispositivo Bio2Bit, la scheda NUCLEO STM32L476RG e il modulo Bluetooth Low Energy.

5.2 Architettura Hardware

5.2.1 Bio2Bit

Il dispositivo Bio2Bit (rappresentato in Figura 5.2) offre all'utente la possibilità di monitorare diversi parametri fisici e fisiologici quali il segnale elettrocardiografico, la bioimpedenza del tessuto cutaneo e la risposta galvanica della pelle (GSR). Esso dispone inoltre di trasduttori MEMS (Micro Electro Mechanical System) integrati nel componente LSM6DS3H di STMicroelectronics [24], utilizzati per acquisire le accelerazioni angolari e lineari sui tre assi (x, y, z). Il Bio2Bit è equipaggiato con un processore ultra-low power ARM Cortex-M4 a 32 bit [2], una batteria da 3.7 Vdc agli ioni di litio da 155 mAh e un modulo dedicato alla carica tramite collegamento micro USB. Dispone inoltre di un modulo di comunicazione Bluetooth Low Energy e di un sistema di acquisizione ultra-low power per segnali biologici integrato nel componente HM121 [22].

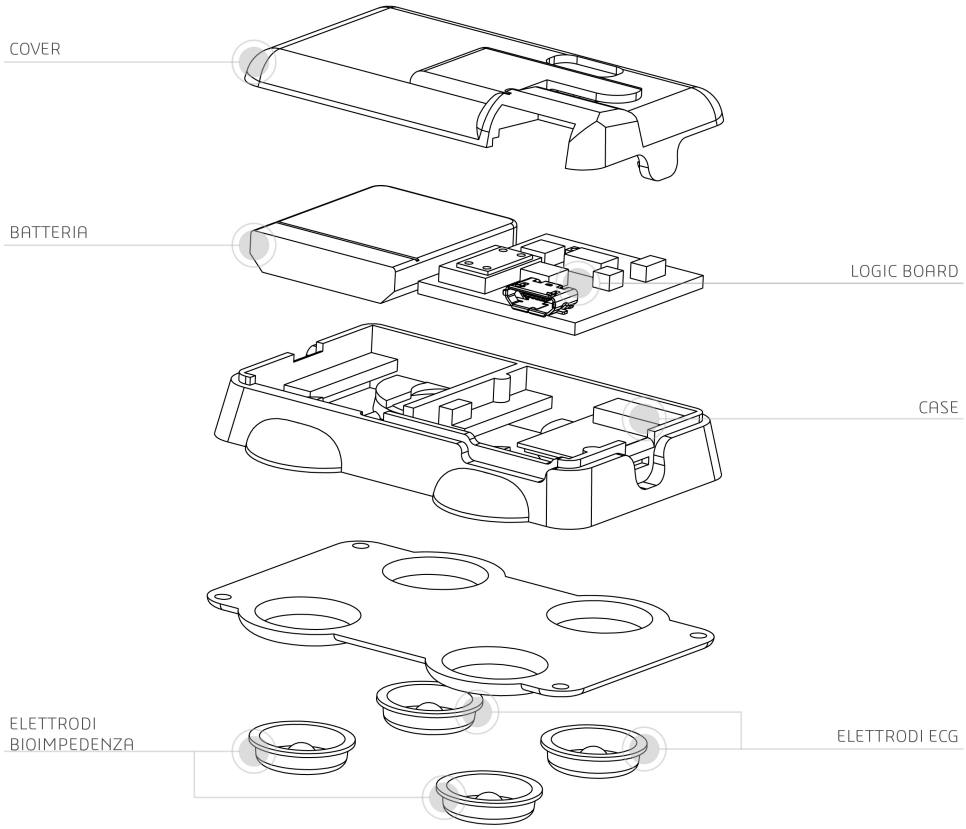

Figura 5.2: Estruso del dispositivo indossibile Bio2Bit. In Figura sono riportati i componenti principali. La LOGIC BOARD ospita il processore ARM Cortex-M4, il componente HM121 e i MEMS (LSM6DS3H). Sono inoltre indicate le coppie di elettrodi usati per acquisire il segnale ECG e la bioimpedenza corporea.

Componente: HM121

Il componente HM121 [22] è dotato di tre canali di acquisizione, come mostrato in Figura 5.3. Il primo canale è dedicato all'acquisizione del segnale ECG. Esso misura il segnale elettrocardiografico in ingresso e fornisce in uscita il segnale digitalizzato attraverso la porta SPI. Inoltre questo canale dispone di un *contact check* che permette di misurare la qualità della connessione tra l'elettrodo e la superficie cutanea. Il secondo canale viene utilizzato per acquisire la bioimpedenza corporea. Tale grandezza viene misurata iniettando una corrente AC (compresa tra 20nA a 250nA ad una frequenza massima di 1KHz) attraverso la pelle e misurando la differenza di potenziale tra gli elettrodi. Il terzo canale è impiegato per l'acquisizione del segnale GSR, anch'esso ottenuto misurando la differenza di potenziale generata dall'iniezione di una corrente AC avente le stesse specifiche riportate sopra.

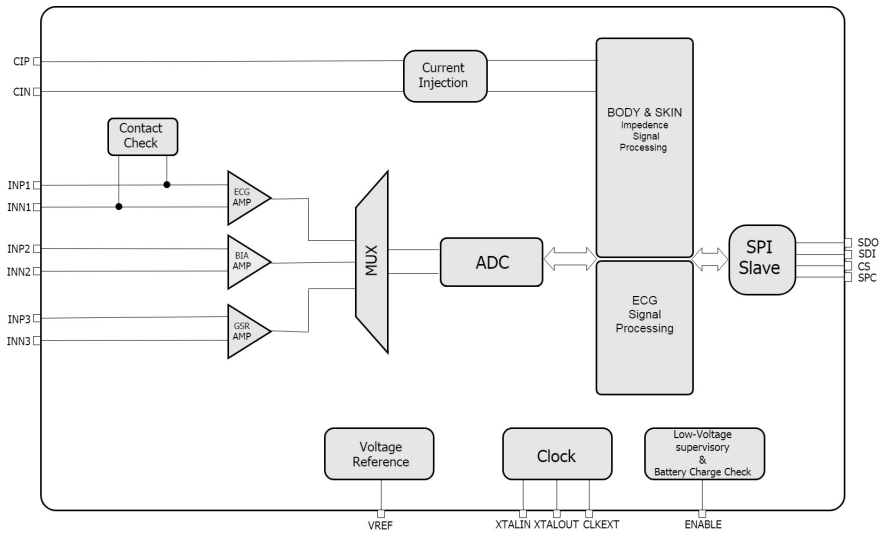

Figura 5.3: Schema a blocchi del componente HM121.

Processore: ARM Cortex-M4

I processori ARM sono caratterizzati da un basso consumo di potenza, proprietà che li rende molto diffusi nei dispositivi portatili come in smartphone, tablet e nei sistemi embedded. Viene definito embedded un sistema di elaborazione specializzato, integrato in un dispositivo fisico in modo da controllarne le funzioni tramite un apposito programma dedicato. I processori ARM appartengono alla famiglia dei processori RISC (Reduced Instruction Set Computer) e fanno uso di un set limitato di istruzioni che permettono la realizzazione di un architettura semplice e lineare. Alcune istruzioni provvedono a caricare i dati dalla memoria nei registri (istruzioni di *load/store*) ed eseguire operazioni aritmetiche e logiche (ad es. somma, sottrazione e AND), denominate *ALU instruction*, altre istruzioni controllano specificatamente il flusso del programma (*branch insctruction*). Il processore ARM Cortex-M4 dispone di una Floting Point Unit (FPU) a singola precisione alla quale sono dedicati trentadue registri.

5.2.2 Scheda di Sviluppo: STM32L476RG

La nucleo STM32L476RG [25] è una scheda di sviluppo progettata e realizzata da STMicroelectronics. Essa dispone di un processore ARM Cortex-M4 a 32 bit [2] in grado di operare ad una frequenza di clock massima di 80MHz. La scheda è dotata di una memoria flash di dimensioni di 1Mbyte e di una memoria SRAM di 128 Kbyte. Inoltre dispone di un FSMC (Flexible Static Memory Controller) configurabile che permette di connettere al dispositivo una memoria esterna. La nucleo è equipaggiata con periferiche hardware analogiche alimentate indipendentemente. Sono presenti tre convertitori Analogici-Digitali a 12 bit, due convertitori Digitali-Analogici a 12 bit, due amplificatori operazionali e due comparatori di tensione. La scheda è predisposta con ben venti interfacce di comunicazione, tra le quali troviamo un'interfaccia di comunicazione USB OTG 2.0, due interfacce seriali audio e cinque USARTs (Universal Synchronous-Asynchrous Receiver/Transmitter), periferiche hardware utilizzate per la comunicazione seriale. L'STM32L476 è pensata per lo sviluppo e la realizzazione di prototipi software e hardware e offre il vantaggio di poter connettere moduli dedicati quali ad esempio il modulo per camera digitale, i driver di controllo per motori DC, e i moduli di comunicazioni (ad es. Bluetooth Low Energy).

Modulo Bluetooth Low Energy

Il modulo Bluetooth Low Energy X-NUCLEO-IDB05A1 [23] consente di instaurare una comunicazione via Bluetooth tra il dispositivo indossabile Bio2Bit e la NUCLEO STM32L476RG. In particolare permette di ricevere lo streaming dati e inviare comandi di controllo al dispositivo, come ad es. iniziare o interrompere una nuova registrazione.

Il Bluetooth è uno standard di comunicazione di uso comune presente in dispositivi quali smartphone, tablet e auricolari. Esso viene utilizzato per la trasmissione wireless a corto raggio, nella banda ISM (Industrial, Scientific and Medical) a 2.4GHz, tra dispositivi che possiedono chip compatibili. La comunicazione si basa sulla trasmissione di pacchetti, ovvero gruppi di *bit* che contengono non solo l'informazione vera e propria ma anche dati aggiuntivi per il riconoscimento e la sincronizzazione tra i dispositivi. La configurazione base di trasmissione vede coinvolti due apparati, di cui uno assume il ruolo di *master* e l'altro di *slave*. Il *master* ha il compito di gestire la comunicazione (inizio, sincronizzazione, termine), mentre lo *slave* si limita a seguire le istruzioni del *master*. Nel sistema mostrato in Figura 5.1 il *master* è la scheda STM32L476RG equipaggiata del modulo Bluewtooth Low Energy, mentre lo *slave* è il dispositivo indossabile Bio2Bit.

5.3 Architettura Software

In questa Sezione verrà data la descrizione dell'architettura software del sistema di monitoraggio. In particolare ci concentreremo sull'architettura presente sulla scheda NUCLEO STM32L476RG.

5.3.1 Tool STM32CubeL4

Il tool STM32Cube Embedded Software è stato proposto da STMicrolectronics per facilitare lo sviluppo software sulle schede STM32 (tra le quali rientra la STM32L476RG) al fine di ridurre i tempi e i costi dello sviluppo. Il tool è costituito da una serie di pacchetti software che permettono di avere una buona astrazione rispetto all'hardware sottostante. Tra questi troviamo il STM32CubeL4 Hardware Abstraction Layer (HAL) e il Low Layer (LL). L'HAL è costituito da un insieme di driver ad alto livello pensati per mascherare la complessità della MCU (MicroController Unit) e delle periferiche presenti sulla scheda di sviluppo. I driver HAL offrono delle API in grado di semplificare lo sviluppo delle applicazioni. Ad esempio, essi permettono di inizializzare e configurare periferiche di comunicazione quali la UART. D'altro canto i driver LL offrono API che permettono di operare a livello di registri, richiedendo tuttavia un'ottima conoscenza dalla MCU e di tutte le periferiche.

FreeRTOS

FreeRTOS [19] è un sistema operativo per sistemi embedded sviluppato e mantenuto da Real Time Engineers Ltd. Esso è pensato per applicazioni in real-time su sistemi embedded a singolo microprocessore. FreeRTOS è un kernel in tempo reale (o scheduler in tempo reale) e permette alle applicazioni di essere organizzate come una raccolta di *thread* indipendenti. Al momento della creazione, ad ogni *thread* viene associato un nome. Il kernel decide quale *thread* dovrebbe essere eseguito esaminando la priorità assegnata a ciascuno di essi e verificando lo stato in cui si trovano (pronto, esecuzione, stop o attesa). I diversi *thread* (chiamati anche *task* nel seguito) comunicano tra loro scambiandosi messaggi attraverso delle code.

5.3.2 Sistema software presente sulla scheda STM32L476RG

In questa Sezione descriveremo il sistema software presente sulla NUCLEO STM32L476RG. Esso è basato sul sistema operativo FreeRTOS, nel quale troviamo i tre seguenti *task*: VcoTask, BleTask e AlgTask.

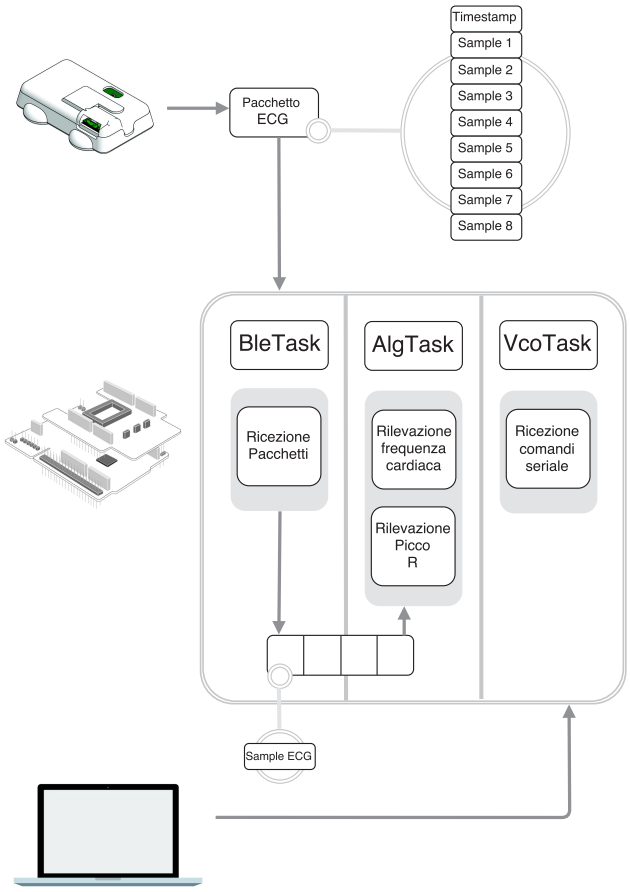

Figura 5.4: Rappresentazione grafica dell'architettura software presente sulla scheda NUCLEO STM32L476RG basata sul sistema operativo FreeRTOS. In Figura sono mostrati i 3 task: BleTask, AlgTask e VcoTask. BleTask riceve i pacchetti trasmessi via Bluetooth dal dispositivo Bio2Bit, li formatta, estrapolando ogni singolo sample, e li passa all'AlgTask. AlgTask è in grado di riconoscere il sample relativo al picco-R del battito corrente e stimare la frequenza cardiaca. VcoTask è responsabile della comunicazione seriale tra il laptop e la NUCLEO STM32L476RG. Esso riceve messaggi, come comandi per avviare una nuova registrazione, li interpreta ed esegue le dovute azioni (ad es. invia comandi al Bio2Bit per avviare una nuova registrazione).

Ogni *task* svolge una funzione specifica:

- il primo, VcoTask, è responsabile della gestione della comunicazione seriale tra la NUCLEO STM32L476RG e il computer. Esso riceve i messaggi inviati via seriale (ad es. attraverso uno script in Matlab), li interpreta e decide quali azioni eseguire. Tra i possibili comandi troviamo: "inizia una nuova registrazione", "termina registrazione", oppure "inizia una sessione in streaming".

- Il secondo, BleTask, riceve i pacchetti dal modulo Bluetooth e li trasmette all'AlgTask. In particolare il dispositivo Bio2Bit invia pacchetti di dimensione pari a 20 Byte, ognuno dei quali può contenere informazioni relative ai dati ECG oppure ai dati MEMS (accelerometri o giroscopi). Nel caso in cui il pacchetto contenga dati dell'elettrocardiogramma, esso risulta così composto: i primi 4 Byte indicano il timestamp, mentre gli altri 16 Byte corrispondono a 8 sample del tracciato ECG. Ogni sample ECG ha dimensione pari a 2 Byte e il timestamp fa riferimento all'istante temporale rispetto al quale è stato acquisito il primo sample. Nel caso in cui il pacchetto contenga dati MEMS, esso risulta così composto: i primi 4 Byte contengono il timestamp, mentre gli altri 12 Byte fanno riferimento ai sample (gli ultimi 4 byte non vengono utilizzati). Ogni sample ha dimensione pari a 2 Byte; i primi 3 *sample* del pacchetto corrispondono ai valori acquisiti rispetto ai 3 assi x, y e z all'istante di tempo $t = timestamp$, mentre gli ultimi 3 corrispondono ai sample campionati all'istante di tempo successivo. In Figura 5.4 è riportato il sistema software appena descritto. Ogni pacchetto Bluetooth viene ricevuto dalla scheda NUCLEO STM32L476RG, qui il BleTask estrae ogni sample dal pacchetto e lo invia all'AlgTask.

- Il terzo, AlgTask, riceve i sample ECG ed esegue gli algoritmi. È in grado di riconoscere il sample corrispondente al picco-R del battito corrente e stimare la frequenza cardiaca.

Capitolo 6

Monitoraggio ECG in Tempo Reale su STM32L476

In questo capitolo verranno presentati i moduli software necessari all'integrazione del rilevamento delle anomalie nel sistema presentato nel Capitolo 5. Quest'ultimi permetteranno di filtrare il segnale ECG in ingresso, selezionare il battito corrispondente al picco R, rilevare e scartare il battito soggetto a rumore causato dal movimento e in ultimo rilevare i battiti anomali.

6.1 Configurazione del Dispositivo

Al fine di rilevare le anomalie con il sistema descritto nella Sezione 5.3.2, è necessario integrare nel dispositivo STM32L476RG i seguenti moduli software: *Filtraggio*, *Finestratura battito*, *Motion Artifact*, *Rilevazione Anomalie* e *Ricezione Trasmissione dati seriale*. Il nuovo sistema è rappresentato in Figura 6.1. In particolare, il modulo denominato *Filtraggio* permette di filtrare il tracciato elettrocardiografico attraverso rispettivamente due filtri mediani e un filtro passa basso (come descritto nella Sezione 2.4.1). Il modulo *Finestratura battito* permette di estrarre dal tracciato ECG (costituito dai sample trasmessi dal dispositivo Bio2Bit alla scheda NUCLEO STM32L476RG) il battito corrente, ovvero $\mathbf{s}_i$. Prima di risolvere il problema del rilevamento delle anomalie è necessario riconoscere se il battito corrente è affetto da disturbi causati dal movimento. Il modulo denominato *Motion Artifact* rileva e scarta i battiti rumorosi come descritto nel Capitolo 4. Successivamente è possibile procedere al rilevamento delle anomalie utilizzando il modulo *Rilevazione anomalie*. Quest'ultimo calcola il vettore $\mathbf{x}$ (2.4.2), rappresentazione sparsa basata sul dizionario D e l'errore di ricostruzione (r_i) (2.5). Il dizionario D, descrive le caratteristiche principali dell'attività cardiaca dell'utente, viene appreso mediante l'Algoritmo KSVD (Algoritmo 2). Il modulo *Ricezione Trasmissione dati seriale* permette infine di trasferire i dati via seriale del tracciato ECG dalla scheda NUCLEO STM32L476RG al computer, sul quale verrà appresa la matrice D. L'Algoritmo KSVD richiede un elevata memoria e potenza di calcolo , per questo motivo il dizionario viene appreso su un dispositivo esterno, potrà quest'ultimo essere un computer o uno smartphone.

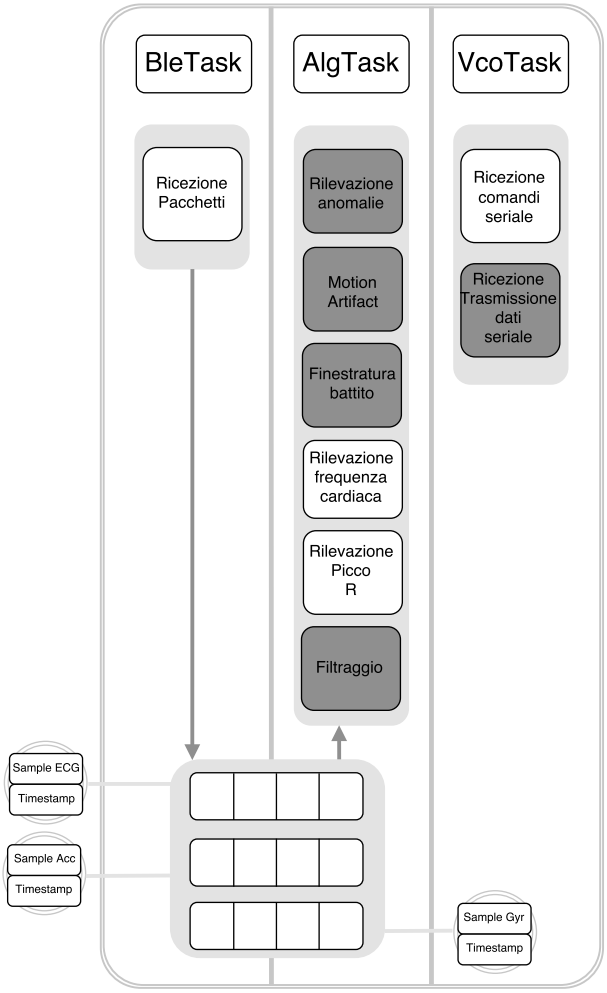

Figura 6.1: Sistema software aggiornato con i moduli software necessari all'integrazione del rilevamento delle anomalie sul sistema esistente (descritto nel Capitolo 5). In particolare sono state aggiunge due code per la trasmissione dei dati MEMS dal BleTask all'AlgTask. Inoltre sono stati integrati i seguenti moduli software: il modulo dedicato al filtraggio del tracciato ECG (Filtraggio), il modulo per estrarre il battito corrente $\mathbf{s}_i$ dal tracciato ECG (Finestratura battito), il modulo dedicato alla rilevazione dei battiti rumorosi (Motion Artifact), il modulo per etichettare il battito corrente come anomalo o meno (Rilevazione Anomalie). In ultimo nel task VcoTask è stata integrata una funzione per la trasmissione e ricezione dei dati via seriale dalla scheda NUCLEO STM32L476RG al computer.

6.1.1 Dictionary Learning

Come anticipato in precedenza, al fine di rilevare le anomalie è necessario disporre della matrice D (dizionario). Tale matrice viene appresa attraverso l'algoritmo KSVD (Sezione 2.3.2). Quest'ultimo riceve in ingresso il *training set* $S \in \mathbb{R}^{p \times m}$ (collezione di battiti di training) e genera in uscita il dizionario $D \in \mathbb{R}^{p \times n}$. Le colonne della matrice S contengono i battiti dell'utente estratti alla frequenza cardiaca f_{TR}. In particolare, il *training set* S viene appreso registrando l'attività cardiaca dell'utente selezionando tutti i battiti aventi frequenza cardiaca uguale a f_{TR} (nel nostro caso f_{TR} è stata impostata a 80 bpm).

La selezione può essere effettuata in due differenti modalità. La prima consiste nell'inviare i *sample* relativi all'attività cardiaca (tracciato ECG) e fisica (dati MEMS) direttamente al computer, salvarli, e successivamente risolvere il problema della generazione del dizionario D. Il tutto viene fatto attraverso un apposito programma (uno script in Matlab) che implementa le tre funzionalità rappresentate in Figura 6.1 dai blocchi: *Filtraggio*, *Finestratura battito* e *Motion Artifact*. In particolare, il tracciato ECG viene filtrato attraverso il blocco *Filtraggio*, e successivamente vengono estratti tutti i battiti aventi frequenza cardiaca pari a f_{TR}. I battiti selezionati vengono raggruppati nella matrice S in modo da formare il *training set* dal quale verrà appresa la matrice D (attarverso l'algortimo KSVD). Una volta appreso, il dizionario verrà trasmesso via seriale e salvato sulla scheda NUCLEO STM32L476RG. Il vantaggio di questa soluzione è quello di non richiedere un elevato carico computazionale sulla scheda NUCLEO STM32L476RG, la scheda si limiterà a ricevere i sample trasmessi dal dispositivo Bio2Bit e li inviarà direttamente al computer attraverso il modulo *Ricezione Trasmissione dati seriale*.

L'alternativa consiste nell'eseguire le tre operazioni (*Filtraggio*, *Finestratura battito* e *Motion Artifact*) direttamente sulla scheda, trasmettendo al computer solamente quei battiti aventi frequenza pari a f_{TR}. La differenza sostanziale rispetto al caso precedente sta nel fatto che, in questo caso, il carico computazionale richiesto per la selezione dei battiti è collocato sulla NUCLEO STM32L476RG, il computer riceve semplicemente i battiti da salvare nella matrice S.

6.2 Preprocessing

Il modulo denominato *Filtraggio* riceve in ingresso i sample del tracciato ECG trasmessi dal Bio2Bit e genera in uscita i sample filtrati. Quest'ultimo è composto da due filtri mediani di dimensione rispettivamente 51 e 154, e un filtro a media mobile di dimensione 13. Il filtro mediano serve a correggere la baseline del tracciato rendendola orizzontale. Il filtro a media mobile agisce invece come filtro passa-basso e serve a rimuovere eventuale rumore ad alte frequenze. Il principio di funzionamento di tale modulo è rappresentato in Figura 6.2.

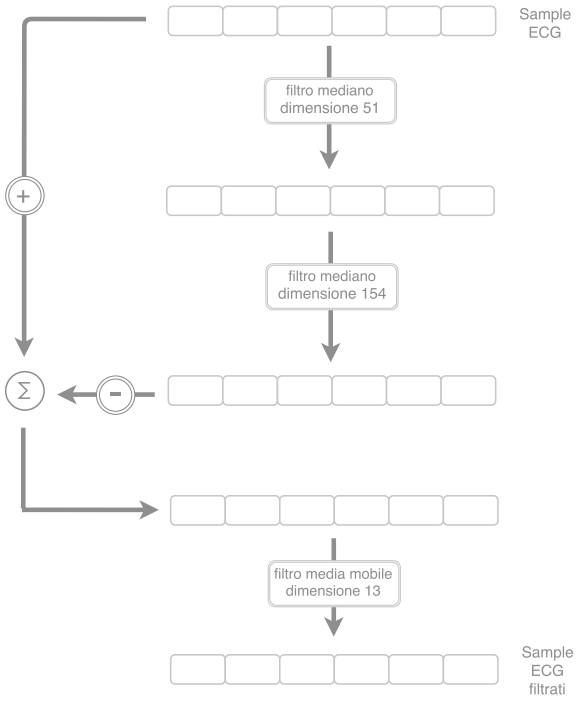

Figura 6.2: Il tracciato ECG viene filtrato attraverso i due filtri mediani in cascata, il primo di dimensione 51 e il secondo di dimensione di 154. Il risultato ottenuto viene sottratto al segnale originale e successivamente filtrato attraverso il filtro a media mobile di dimensione 13.

Il tracciato ECG viene prima filtrato attraverso i due filtri mediani. Il risultato viene sottratto al segnale originale e successivamente filtrato attraverso il filtro a media mobile.

Il processo appena descritto deve venir eseguito in modalità *online*. Ogni sample ECG viene estratto dal pacchetto Bluetooth corrente, trasmesso all'AlgTask e filtrato attraverso il processo descritto sopra. I sample ECG filtrati vengono poi salvati in un buffer circolare come rappresentato in Figura 6.3. Tutto il preprocessing è eseguito da una macchina a stati finiti, la quale verrà descritta nella prossima Sezione.

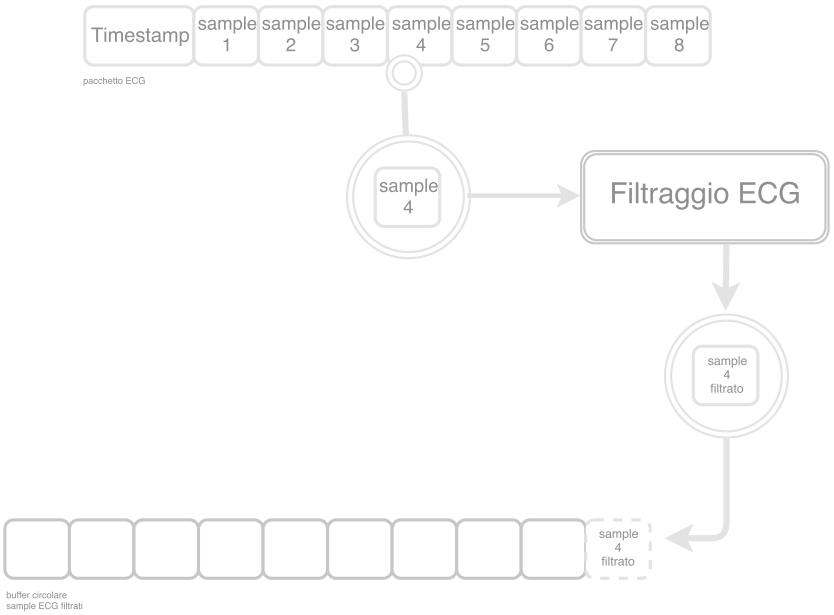

Figura 6.3: Rappresentazione grafica del processo di filtraggio. Ogni singolo sample ECG viene estratto dal pacchetto Bluetooth, trasmesso all'AlgTask (mediante l'apposita coda) e filtrato attraverso il modulo Filtraggio.

6.2.1 Macchina a Stati

Il modulo *Filtraggio* è stato implementato mediante una macchina a stati finiti, il cui principio di funzionamento è rappresentato in Figura 6.4.

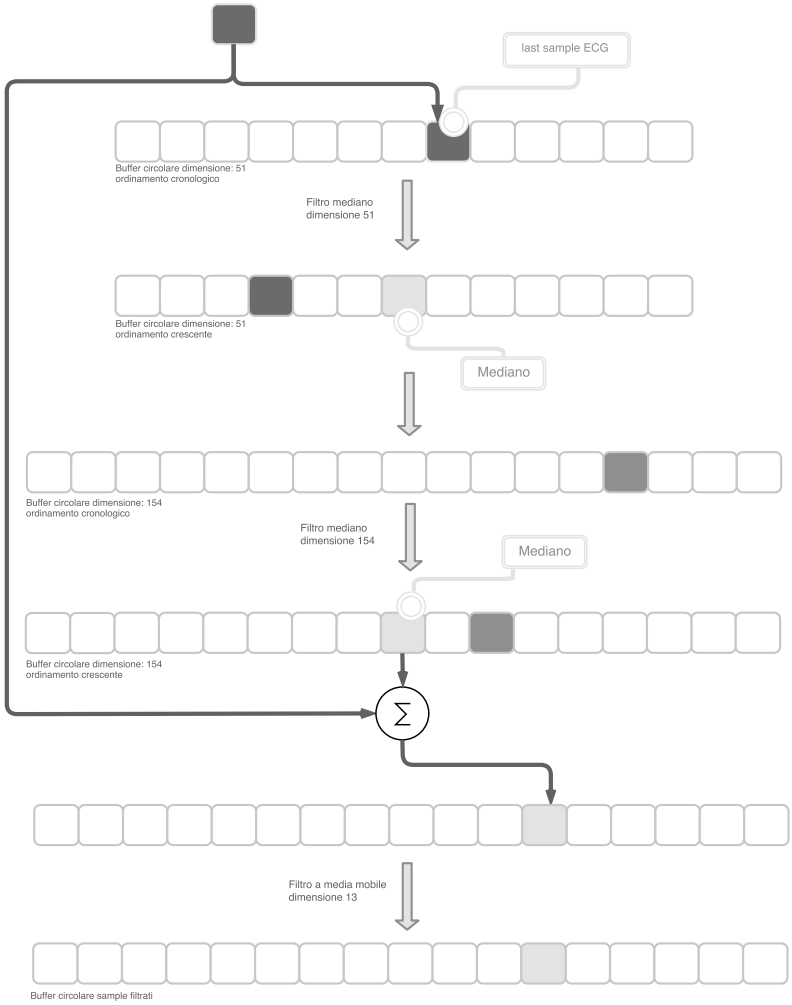

Figura 6.4: Rappresentazione grafica del processo di filtraggio mediante la macchina a stati finiti.

Il sample ECG corrente, indicato in grigio scuro in Figura 6.4, viene sostituito con il sample più vecchio del primo e secondo buffer circolare (array di sample). Il primo buffer è ordinato in ordine cronologico, e permette quindi di determinare la posizione del sample più vecchio nel secondo buffer. Successivamente si procede al calcolo del primo mediano attraverso il secondo buffer. L'uscita di quest'ultimo viene sostituita con il sample più vecchio del terzo e quarto buffer (anche in questo caso il terzo buffer viene utilizzato per mantenere l'ordine cronologico e permettere la ricerca del sample più vecchio nel quarto buffer). Successivamente viene estratto il mediano dal quarto buffer e sottratto al sample originale. Il risultato viene poi salvato nel quinto buffer, al quale viene applicato il filtro a media mobile in modo da ottenere il sample filtrato. Quest'ultimo viene poi salvato nell'ultimo buffer circolare dal quale sarà possibile estrarre il battito corrente.

Analisi Tempi Macchina a Stati

L'analisi dei tempi è stata condotta misurando il tempo d'esecuzione necessario a filtrare 1500 sample ECG. I tempi necessario al riordinamento dei due buffer (usati per il calcolo del mediano) e per l'operazione di ricerca variano in funzione della posizione dei sample. L'algoritmo utilizzato per l'ordinamento è l'*Insertion Sort* avente complessità temporale pari: $\mathcal{O}(n^2)$ nel caso pessimo e $\mathcal{O}(n)$ nel caso ottimo (array già ordinato). Il tempo medio impiegato per filtrare un singolo sample è 307.8053 μs (calcolato come media dei 1500 tempi, deviazione standard pari a 21.2166 μs). In Figura 6.5 è riportato l'istogramma della distribuzione dei tempi impiegati per ordinare 1500 sample ECG.

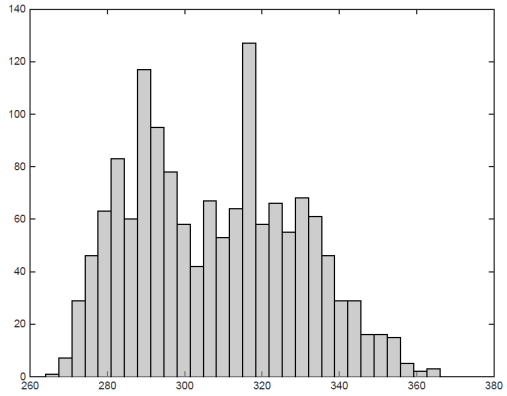

Figura 6.5: Distribuzione dei tempi, espressi in μs, per il riordinamento di 1500 sample. Sulle ascisse sono indicati i tempi mentre sulle ordinate il numero di elementi.

6.2.2 Finestratura Battito

La fase successiva al filtraggio consiste nella determinazione del picco-R del battito corrente, funzionalità che è già implementata nel modulo software *rilevazione picco R* (all'interno dell'AlgTask). Una volta rilevato il picco, si procede al calcolo della frequenza cardiaca corrente che viene ottenuta come mediana delle ultime 10 distanze R-R (distanza temporale tra un picco R corrente e il precedente). Sulla base del risultato ottenuto viene scelta la finestra di dimensione opportuna per l'estrazione del battito corrente (come descritto nella Sezione 2.4.2). La dimensione del battito, espressa in numero di sample, varia al variare della frequenza cardiaca, ad es. maggiore è la frequenza cardiaca corrente minore saranno il numero di *sample* che costituiscono il battito.

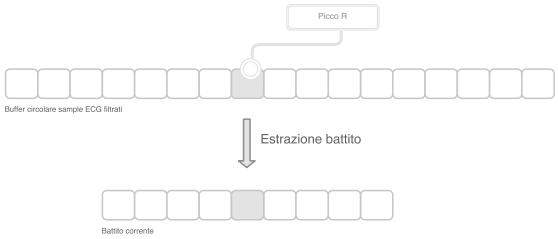

Figura 6.6: Estrazione battito corrente. Il picco R viene rilevato e, sulla basa della frequenza cardiaca corrente, viene scelta la dimensione della finestra per l'estrazione.

6.2.3 Motion Artifact

In questa Sezione verrà presentato il modulo software denominato *Motion Artifact*. Esso è stato implementato partendo dal modello descritto nella Sezione 3.3.3 (NN feed-forward). La rete neurale riceve in ingresso le *feature* estratte dai segnali MEMS (accelerometri e giroscopi) e dal battito corrente (descritte nel Capitolo 4) e predice la classe d'appartenenza del battito, ovvero *rumoroso* o *non rumoroso*. I pacchetti Bluetooth relativi ai dati degli accelerometri e giroscopi vengono ricevuti dal modulo Ricezione Pacchetti (nel task BleTask), e utilizzati per estrarre i moduli dei segnali (come descritto nel Capitolo 3) e il relativo timestamp. In Figura 6.7 è riportato il processo appena descritto sui pacchetti relativi ai dati degli accelerometri. Una volta estratti, i valori vengono trasmessi all'AlgTask mediante l'apposita coda (come descritto in Figura 6.1). Qui i dati vengono salvati in un buffer circolare dal quale verranno estratte le *feature*. In particolare, ogni volta che viene rilevato un nuovo picco R (mediante il modulo Rilevazione Picco R) si estraggono dai buffer circolari (rispettivamente degli accelerometri, giroscopi e sample ECG) le relative *feature*. Quest'ultime vengono estratte da finestre temporali di dimensione 0.5 e 2 secondi centrate rispetto al picco R corrente (come indicato in Figura 6.8). Successivamente le *feature* vengono utilizzate come ingresso della rete neurale, implementata salvando i pesi appresi durante la fase di *learning* (descritta nella Sezione 4.3).

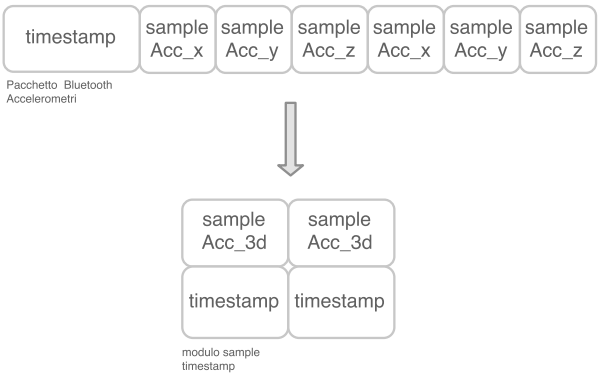

Figura 6.7: Rappresentazione grafica del processo di estrazione del modulo del segnale degli accelerometri.

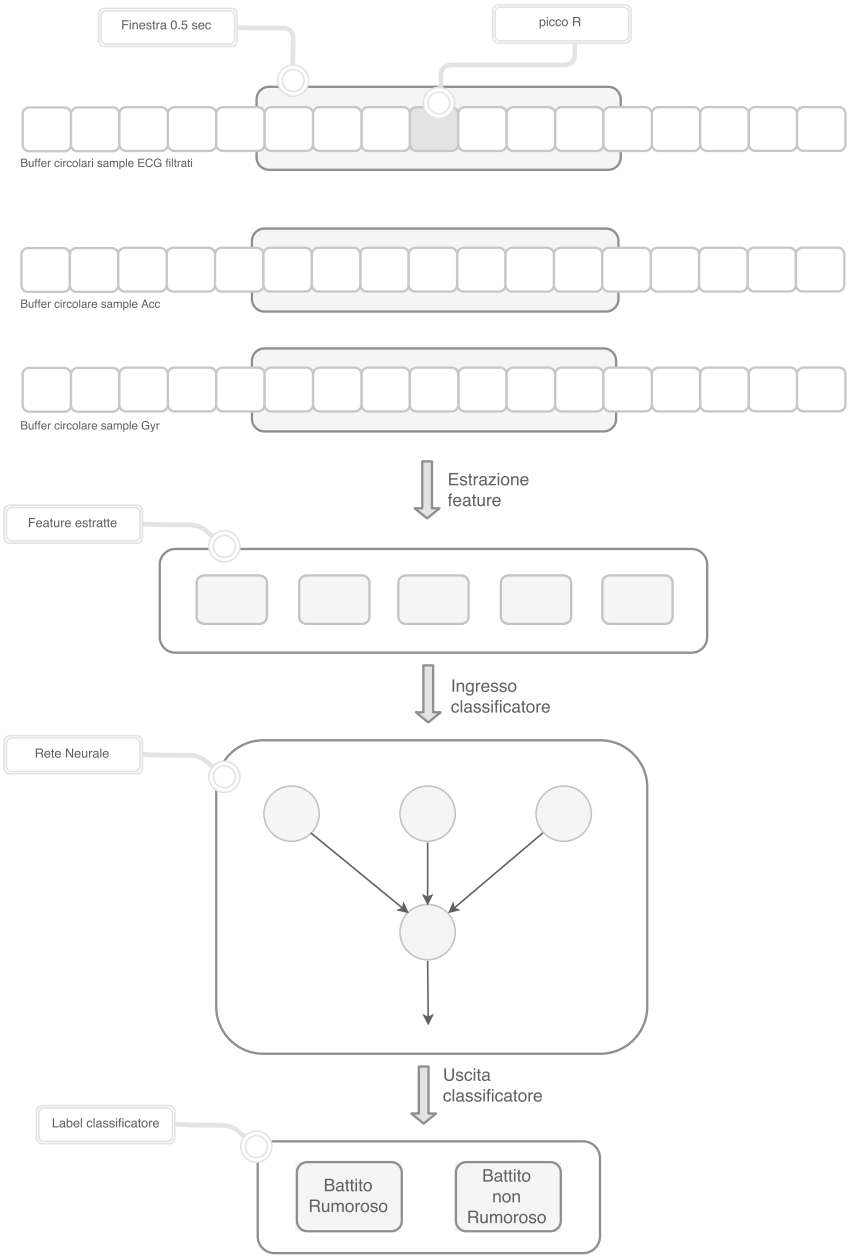

Figura 6.8: Principio di funzionamento del modulo denominato Motion Artifact. Dai tre buffer (sample ECG , accelerometri e giroscopi) vengono estratte le feature dalle finestre temporali (in Figura, per chiarezza, viene mostrata solamente la finestra da 0.5 secondi). Successivamente le features vengono date in ingresso alla rete neurale, la quale genererà in uscita la classe d'appartenenza del battito, rumoroso o non rumoroso.

Analisi Tempi Motion Artifact

L'analisi dei tempi è stata condotta misurando il tempo d'esecuzione necessario a classificare ogni singolo battito come rumoroso o meno, ottenendo un tempo pari a 936 μs (il numero di operazioni eseguite dalla rete neurale è costante). La complessità temporale è $\mathcal{O}(n)$, il numero di operazioni dipendono dal numero di neuroni nel singolo *hidden layer*.

6.2.4 Rilevazione anomalie

La rilevazione delle anomalie avviene risolvendo il problema di *sparse coding* (2.4), e successivamente valutando l'errore di ricostruzione (2.5). Il modulo software denominato *Rilevazione anomalie* riceve in ingresso il battito corrente (vettore di sample estratto dal tracciato ECG), il dizionario D, e restituisce il valore dell'errore di ricostruzione, il quale verrà successivamente comparato al valore soglia γ al fine di etichettare il battito come anomalo o normale. Per risolvere il problema dello *sparse coding* abbiamo implementato una versione efficiente dell'Algoritmo 1 che sfrutta la fattorizzazione inversa di Cholesky. Tale algoritmo è noto come *Orthogonal Matching Pursuit Cholesky* (OMP Cholesky), ed è riportato nell'Algoritmo 3.

Algorithm 3 OMP-Cholesky

Require: Dictionary $\mathbf{D}$, signal $\mathbf{s}$, target sparsity κ
Ensure: Sparse representation x such that $\mathbf{s} \approx \mathbf{Dx}$

Set $I := (\ )$, $\mathbf{L} := [1]$, $\mathbf{r} := \mathbf{s}$, $\mathbf{x} := \underline{0}$, $\alpha := \mathbf{D}^T\mathbf{s}$, $n := 1$
while (*Stopping criterion not met*) **do**
  $\hat{k} := \arg\max_k |\underline{d}_k^T \mathbf{r}|$
  if $n > 1$ **then**
    $\underline{w} :=$ Solve for $\underline{w}$ $\quad \{\mathbf{L}\underline{w} = \mathbf{D}_I^T \underline{d}_k\}$
    $\mathbf{L} := \begin{bmatrix} \mathbf{L} & 0 \\ \underline{w}^T & \sqrt{1 - \underline{w}^T \underline{w}} \end{bmatrix}$
  end if
  $I := (\ I, \hat{k})$
  $\mathbf{x}_I :=$ Solve for $\underline{c}$ $\{\mathbf{L}\mathbf{L}^T \underline{c} = \underline{\alpha}_I\}$
  $\mathbf{r} := \mathbf{s} - \mathbf{D}_I \mathbf{x}_I$
  $n := n + 1$
end while

L'OMP Cholesky si differenzia rispetto all'algortimo OMP classico (descritto nel Capitolo 2) per il calcolo del vettore $\mathbf{x}$. In particolare, questa implementazione permette di ridurre la complessità computazionale dovuta dalla risoluzione del problema $\mathbf{x}_I = (D_I^T D_I)^{-1} D_I^T \mathbf{s}$ (linea 5 dell'Algoritmo OMP). La matrice $D_I^T D_I$ è simmetrica e semidefinita positiva, quindi è possibile ottenere la matrice all'iterazione corrente, $A_n = D_I^T D_I$, partendo dalla fattorizzazione di Cholesky della matrice all'iterazione precedente (A_{n-1}), evitando così il calcolo dell'inversa (per una descrizione più dettagliata è possibile consultare [21]).

L'algoritmo OMP Cholesky è stato implementato attraverso una libreria che permette di inizializzare le strutture dati necessarie (void initOMP(void)) e di calcolare l'errore di ricostruzione attraverso la funzione: fixedpt ompCholesky (HeartBeat *heartBeat, Dictionary *dictionary, int SPAR). Quest'ultima riceve in ingresso il battito corrente (HeartBeat *heartBeat), il dizionario (Dictionary *dictionary), la sparsità (SPAR, definita come il numero massimo di colonne selezionabili dal dizionario) e restituisce l'errore di ricostruzione.

La libreria è stata implementata utilizzando la rappresentazione in virgola fissa, soluzione che ha permesso di ridurre la complessità computazionale derivante dall'utilizzo della *Floating Point Unit* (FPU). In virgola fissa ogni valore ha un numero fisso di bit utilizzati per rappresentare la parte intera (i) e un numero di bit utilizzato per la parte decimale (f):

$$[i.f].$$

La risoluzione del numero in virgola fissa è dettata dal numero di bit dedicati alla parte decimale secondo la relazione: 2^{-f}. L'effetto collaterale, dovuto all'utilizzo di una rappresentazione di questo tipo, è quello di accumulare errori durante le operazioni, rischiando così di generare errori non tollerabili. Il nostro obiettivo è stato quello di trovare una buona configurazione in termini di numero di bit che ci permettesse di ottenere un miglioramento delle prestazioni, limitando le perdite in termini d'accuratezza nel risultato finale.

6.2.5 Accuratezza dell'errore di ricostruzione

Per mantenere una buona accuratezza nel calcolo dell'errore di ricostruzione è stata scelta la seguente configurazione: *numero bit parte intera* (i): 6 bit, e *numero bit parte decimale* (f): 28 bit. I risultati sperimentali mostrano che la degradazione dell'errore di ricostruzione calcolato con la codifica in virgola fissa è molto vicina all'errore calcolato con il tipo a virgola mobile (ottenuto implementando l'algoritmo OMP Cholesky in matlab con i tipi double). In Figura 6.9 è riportata la comparazione tra l'errore di ricostruzione calcolato utilizzando la rappresentazione in virgola fissa, r_{fixed}, e la rappresentazione in virgola mobile (utilizzando i double), $r_{floating}$.

In particolare, l'istogramma mostra la distribuzione delle differenza tra i due errori calcolata come:

$$G = \frac{\mathbf{r}^2_{fixed} - \mathbf{r}^2_{floating}}{\mathbf{r}^2_{floating}}. \tag{6.1}$$

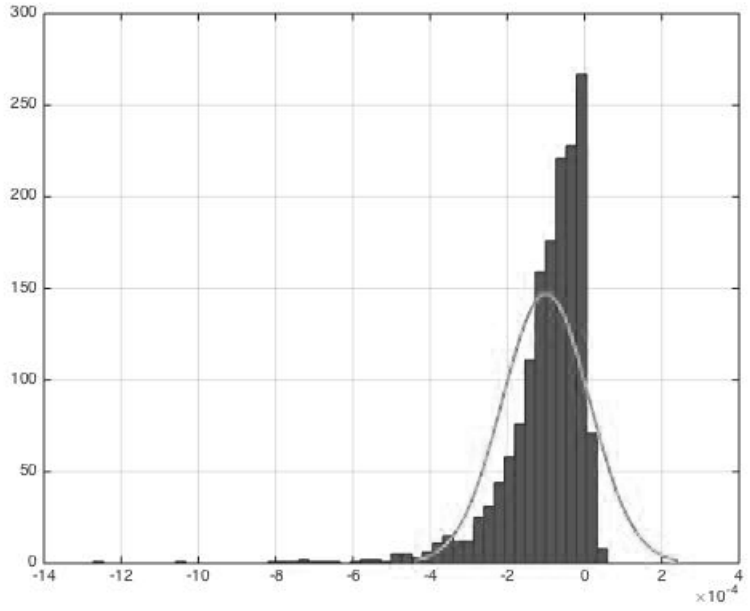

Figura 6.9: Distribuzione della differenza tra l'errore di ricostruzione ottenuto con la rappresentazione a virgola fissa e l'errore di ricostruzione ottenuto con la rappresentazione a virgola mobile. L'istogramma mostra sulle ascisse il valore della differenza mentre sulle ordinate il numero di elementi. Tali valori sono stati ottenuti confrontando l'errore di ricostruzione di 1560 battiti aventi frequenza cardiaca pari a 80 bpm.

L'analisi è stata condotta calcolando l'errore di ricostruzione di 1560 battiti aventi frequenza cardiaca pari a 80 bpm (dimesione: 156 sample) utilizzando lo stesso dizionario D (precedentemente appreso), avente dimensione: 156×8 (dove 8 indica il numero di atomi) e impostando il valore della sparsità pari a 3 (l'algoritmo OMP Cholesky seleziona tre colonne su 8 della matrice D).

Analisi Tempi OMP Cholesky

L'analisi dei tempi è stata condotta misurando il tempo d'esecuzione, espresso in s, necessario a calcolare l'errore di ricostruzione r_i attraverso l'algoritmo OMP Cholesky (implementato attraverso la libreria in virgola fissa). In Tabella 6.1 sono riportati i tempi calcolati utilizzando i seguenti valori di sparsità: 2,3,4,5 e 6 (questi numeri corrispondono al valore del parametro che indica il numero massimo di colonne del *dizionario* selezionate dall'algoritmo OMP Cholesky). Nel nostro caso specifico il valore di sparsità è stato impostato a 3, il battito corrente verrà ricostruito scegliendo i migliori 3 atomi del dizionario.

Tabella 6.1: Tabella tempi OMP Cholesky in virgola fissa

Sparsità	Tempo
2	800 μs
3	1308 μs
4	1885 μs
5	2528 μs
6	3265 μs

Capitolo 7

Direzioni future di ricerca e Conclusioni

7.1 Conclusioni

In questa tesi abbiamo sviluppato un sistema in grado di rilevare anomalie cardiache durante l'acquisizione del tracciato elettrocardiografico (ECG) attraverso il dispositivo indossabile Bio2Bit e la scheda NUCLEO STM32L476RG. Il contributo del lavoro è stato duplice: nella prima parte abbiamo reso il sistema di rilevamento delle anomalie robusto al rumore generato dai movimenti dell'utente. In particolare, per risolvere il problema della classificazione dei battiti rumorosi abbiamo implementato un modello predittivo. La nostra scelta è ricaduta su una rete neurale (NN) feed-forward con un singolo hidden-layer. In questo modo abbiamo ottenuto una riduzione dei falsi allarmi, corrispondenti alle anomalie causate dal movimento dell'utente, pari al 76.86%.

La seconda parte è stata dedicata all'integrazione del sistema di rilevamento delle anomalie sulla scheda NUCLEO STM32L476RG. In particolare, abbiamo implementato sia i moduli software per l'analisi dei singoli battiti e il rilevamento delle anomalie sia le modalità in cui interagiscono tra loro. Al fine di permettere l'esecuzione in tempo reale abbiamo sviluppato una libreria ottimizzata per sistemi embedded ultra low-power, che calcola in modo efficiente una rappresentazione sparsa del singolo battito cardiaco. Le nostre analisi mostrano che il tempo richiesto a processare un singolo battito è pari a circa 2,5 ms. Dal momento che non stiamo considerando le operazioni richieste per la trasmissione Bluetooth e dal sistema operativo, si tratta di una stima ottimistica.

Tuttavia ci aspettiamo che questi risultati garantiscano un buon margine per il rilevamento di anomalie in un tracciato elettrocardiografico in tempo reale.

7.2 Sviluppi futuri

Ulteriori investigazioni in termini di validità dell' intera Pipiline sono necessarie. A tal fine abbiamo intenzione di implementare una demo che ci permetterà rispettivamente di collezionare nuove registrazione, testare il sistema e analizzare nuovi dati. La demo verrà presentata in STMicroelettronics all'interno di una conferenza nazionale (ASTDay: Innovation Domains in IoT), inoltre abbiamo intenzione di sottomettere la nostra candidatura alla conferenza internazionale ICML (International Conference on Machine Learning).

Bibliografia

[1] Michal Aharon, Michael Elad, and Alfred Bruckstein. *rmk*-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[2] ARM. *ARM Cortex-M4: Technical Reference Manual*, 2010.

[3] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[4] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[5] Jonny Farringdon, Andrew J Moore, Nancy Tilbury, James Church, and Pieter D Biemond. Wearable sensor badge and sensor jacket for context awareness. In *Wearable Computers, 1999. Digest of Papers. The Third International Symposium on*, pages 107–113. IEEE, 1999.

[6] Davide Figo, Pedro C Diniz, Diogo R Ferreira, and João M Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14(7):645–662, 2010.

[7] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[8] John R Hampton. *The ECG made easy*. Elsevier/Churchill Livingstone, 2008.

[9] Anil Jain and Douglas Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE transactions on pattern analysis and machine intelligence*, 19(2):153–158, 1997.

[10] Do-Un Jeong, Se-Jin Kim, and Wan-Young Chung. Classification of posture and movement using a 3-axis accelerometer. In *Convergence*

Information Technology, 2007. International Conference on, pages 837–844. IEEE, 2007.

[11] Dean M Karantonis, Michael R Narayanan, Merryn Mathie, Nigel H Lovell, and Branko G Celler. Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. *IEEE transactions on information technology in biomedicine*, 10(1):156–167, 2006.

[12] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2007.

[13] Seon-Woo Lee and Kenji Mase. Activity and location recognition using wearable sensors. *IEEE pervasive computing*, 1(3):24–32, 2002.

[14] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525–533, 1993.

[15] Taketoshi Mori, Yushi Segawa, Masamichi Shimosaka, and Tomomasa Sato. Hierarchical recognition of daily human actions based on continuous hidden markov models. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 779–784. IEEE, 2004.

[16] Arthur J Moss, Peter J Schwartz, Richard S Crampton, Emanuela Locati, and Eric Carleen. The long qt syndrome: a prospective international study. *Circulation*, 71(1):17–21, 1985.

[17] Jiapu Pan and Willis J Tompkins. A real-time qrs detection algorithm. *IEEE transactions on biomedical engineering*, (3):230–236, 1985.

[18] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE, 1993.

[19] Real Time Engineers Ltd. *FreeRTOS: Multitasking in Small Embedded Systems.*

[20] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[21] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. *Cs Technion*, 40(8):1–15, 2008.

[22] STMicrolectronics. *HM121: Ultra low power AFE for ECG, bio-impedance and skin resistance measurements*, 4 2015. Rev. 0.41.

[23] STMicrolectronics. *X-NUCLEO-IDB05A1: Bluetooth Low Energy expansion board based on the SPBTLERF module for STM32 Nucleo*, 7 2015. Rev. 1.

[24] STMicrolectronics. *LSM6DS3H: iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope*, 9 2017. Rev. 5.

[25] STMicrolectronics. *STM32L476: Ultra-low-power ARM Cortex-M4 32-bit MCU+FPU, 100DMIPS, up to 1MB Flash, 128 KB SRAM, USB OTG FS, LCD, ext. SMPS*, 7 2017. Rev. 5.