POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Sviluppo di un'interfaccia Android per device medicali per la misurazione non-invasiva e senza cuffia della pressione arteriosa

Relatori

Prof. Eros PASERO

Ing. Vincenzo RANDAZZO

Candidato

Simone VALVO

Aprile 2021

Sommario

Lo sviluppo tecnologico in ambiente biomedicale ha ottenuto un notevole incremento in grado di sopperire alle numerose difficoltà che possono incorrere in ambito diagnostico, chirurgico e preventivo. Nuove applicazioni soppiantano tecniche obsolete o rendono fattibile ciò che qualche anno fa ci sembrava impossibile. Questo lavoro di tesi si propone di agevolare la misurazione e l'acquisizione dei parametri vitali tramite un dispositivo facilmente accessibile e funzionalmente semplice, utilizzabile anche da persone senza specifica competenza medica. Tale dispositivo, chiamato PulseEcg, ha le sembianze di uno smartwatch ed è in grado di effettuare l'acquisizione del tracciato elettrocardiografico (ECG) e pletismografico (PPG) permettendo, perciò, di monitorare l'attività cardiaca, il livello di saturazione di ossigeno nel sangue e la frequenza cardiaca. Inoltre, mediante gli algoritmi di intelligenza artificiale, consente di derivare i valori di pressione sistolica e diastolica. Costituisce quindi il primo dispositivo wearable in grado di risalire al valore di pressione con un approccio cuff-less, ovvero senza usufruire dello sfigmomanometro. Il PulseEcg, tramite una connessione Bluetooth 4.0, è in grado di trasmettere le misurazioni effettuate ad una applicazione Android; permette la visualizzazione di queste attraverso grafici e strutture dati opportune; consente l'immagazzinamento delle stesse all'interno del proprio smartphone e la loro condivisione, sia come file di testo che come pdf, servendosi delle applicazioni di comunicazione installate. L'applicazione provvede anche alla segnalazione di eventuali anomalie presenti nelle misurazioni, divenendo quindi un mezzo di prevenzione per il soggetto ed un ausilio per gli esperti che potranno richiedere, ove necessario, ulteriori accertamenti. Il progetto nasce da una vecchia implementazione del dispositivo in grado di ottenere unicamente l'ECG dell'utente. La fase iniziale ha dunque previsto un aggiornamento dell'applicazione Android, integrando nuove funzioni e migliorando l'interfaccia grafica, l'usabilità e la stabilità della stessa. Successivamente, è stata sviluppata una seconda interfaccia per usufruire delle nuove funzionalità del PulseEcg. Questa definisce, in aggiunta al segnale ECG, nuove funzioni per l'elaborazione e la visualizzazione del segnale PPG. Infine, per realizzare l'algoritmo di calcolo della pressione, è stata allenata e testata una rete neurale mediante l'utilizzo delle librerie keras e tensorflow. Tale algoritmo è stato, in un secondo tempo, integrato all'interno dell'applicazione, affinché riportasse valori con un'alta precisione senza compromettere la velocità e la fluidità dello smartphone.

Ringraziamenti

Al termine di questo percorso di tesi, desidero ringraziare le persone che mi sono state vicine in tutti questi anni.
In primis, desidero ringraziare il Prof. Eros Pasero per la disponibilità, l'attenzione e la gentilezza mostrata durante il progetto.
Un sentito grazie all'Ing. Vincenzo Randazzo, correlatore di tesi, per gli aiuti costanti, per le dritte indispensabili e precise e per la complicità dimostrata.
Alla mia famiglia, i miei genitori e mio fratello, per aver sempre creduto in me, anche quando io stesso non lo facevo, e avermi supportato anche nei momenti difficili, non solo universitari. Non sarei arrivato fin qui se non fosse stato per voi e ve ne sarò eternamente grato.
Un ringraziamento speciale va ad Alessia che mi ha invece sopportato durante la stesura della tesi (e tante altre volte) e mi ha sempre spinto ad andare avanti.
Ai miei cugini e ai miei zii, ai compagni di mille avventure competitive, a tutti i miei amici. Vi ringrazierei uno per uno, elencando i vostri nomi, ma questi ringraziamenti non possono occupare 100 pagine. Vi mando quindi un grazie di cuore per non avermi mai fatto sentire solo, per l'affetto, per le risate, per avermi consolato e aiutato nei momenti difficili.
Perchè, grazie a tutti voi, sono felice e sono orgoglioso di essere me stesso.
Vi auguro il meglio.
Con affetto.

Indice

Elenco delle tabelle

Elenco delle figure

Acronimi

AC

Corrente Alternata

ADC

Convertitore Analogico-Digitale

AOT

Ahead-Of-Time

API

Application Programming Interface

AV

Atrioventricolare

BLE

Bluetooth Low Energy

BPM

Battiti per minuto

CCCD

Client Characteristic Configuration Descriptor

CE

Conformità Europea

CPU

Unità di Processamento Centrale

DC

Corrente Continua

ECG

Elettrocardiogramma

EDR

Enhanced Data Rate

FA

Fibrillazione Atriale

FIR

Risposta ad Impulso Finito

GATT

Profilo di attributo generico

GFSK

Gaussian frequency-shift keying

GPIO

General Purpose Input Output

GPS

Global Positioning System

HR

Frequenza Cardiaca

IDE

Integrated Development Environment

IO

Input - Output

IR

Infrarosso

ISM

Industrial Scientific Medical

PAN

Personal Area Network

PDF

Portable Document Format

PPG

Fotopletismografia

RFCOMM

Radio Frequency Communication

UART

Universal Asynchronous Receiver Transmitter

UHF

Ultra high frequency

USB

Universal Serial Bus

XML

eXtensible Markup Language

Capitolo 1

Introduzione

1.1 Premessa

La tesi nasce in un contesto sanitario, culturale ed economico complesso causato dall'avvento della malattia respiratoria acuta da SARS-CoV-2 (COVID-19). In questo stato di pandemia, la tecnologia, nelle sue varie forme, assume un ruolo fondamentale permettendo alle varie istituzioni di evitare uno stato di immobilità permanente, di avere un controllo sui contagi e di affrontare la malattia con strumentazioni mediche avanzate. Basti pensare come l'Internet of Things (IoT) abbia fornito sin dall'inizio una piattaforma che consente alle agenzie di salute pubblica il monitoraggio della pandemia COVID-19 o come i Big Data diano l'opportunità di studiare il modello dell'attività virale [1].
Lo sviluppo di tecnologie mobili, indossabili e semplici può giocare un ruolo vitale, individuando la possibile presenza di sintomi in modo veloce, così da poter dare all'utente un'indicazione da far poi valutare ad un esperto [2].
L'azione di rilevamento della malattia è fondamentale per poter evitare l'espansione incontrollata dei contagi ma anche per evitare il sovraffollamento degli ospedali.
Indipendentemente dalla pandemia, il monitoraggio continuo delle condizioni di salute può aiutare a rilevare disfunzioni causate da altre patologie e tenere sotto controllo i parametri all'interno di valori accettabili.
Lo sviluppo di tecnologie medicali indossabili, con un'interfaccia utente semplice e completa, risulta quindi essere un investimento utile non solo per il periodo corrente, ma anche per il futuro.

1.2 Obiettivo della tesi

La tesi si propone di sviluppare un meccanismo di prevenzione che sia semplice da utilizzare ma, allo stesso tempo, affidabile dal punto di vista delle misurazioni e con un costo accessibile. L'applicazione Android vuole così agevolare la misurazione di alcuni valori, indice dello stato di salute di un utente, senza incorrere all'utilizzo di molteplici strumentazioni che possono essere ingombranti o scomode.
In particolare, l'applicazione, connessa tramite Bluetooth ad un dispositivo chiamato PulseEcg, permette di acquisire un elettrocardiogramma (ECG) e un fotopletismogramma (PPG) in modo sincrono. Mediante i valori di questi dati, l'applicazione sarà in grado di calcolare la saturazione dell'ossigeno nel sangue (SpO2) e la frequenza cardiaca (HB). Una delle maggiori innovazioni risiede nella capacità di calcolare le pressioni sistolica e diastolica attraverso una rete neurale precedentemente addestrata ed integrata nell'applicazione.

1.3 Stato dell'arte

Negli anni, i dispositivi indossabili sono diventati sempre più accessibili e utilizzati dai cittadini. Questi, infatti, permettono di usufruire di funzionalità diverse in maniera facile e veloce, senza dover necessariamente prendere uno smartphone o altre tecnologie più ingombranti. Si pensi alla possibilità di rispondere alle chiamate, di visualizzare il proprio battito cardiaco o le email attraverso un orologio. Con la loro espansione territoriale, lo sviluppo tecnologico di questi dispositivi è incrementato esponenzialmente, aumentandone anche la versatilità. I dispositivi indossabili possono quindi essere impiegati per gestire situazioni o problemi più complessi, aiutati dalla possibilità di connettersi a device computazionalmente più potenti, quali smartphone, tablet o computer.
Le applicazioni per smartphone, sia con un sistema operativo basato su Android che su iOS, diventano un mezzo indispensabile per facilitarne l'utilizzo all'utente ma, soprattutto, per aumentare le capacità di calcolo senza dover pesare sul dispositivo wearable, diminuendone quindi l'utilizzo energetico, la dimensione ed i costi. I linguaggi di programmazione utilizzati dalle applicazioni sono infatti enormemente supportati da varie comunità di esperti (e non), le quali creano librerie per affrontare calcoli anche molto complessi. Basti pensare all'integrazione di algoritmi per il machine learning e il deep learning all'interno di dispositivi mobili che aprono le strade a moltissime opportunità.

1.4 Organizzazione della tesi

La tesi si sviluppa partendo da una introduzione ai concetti medici legati all'applicazione così da poter comprendere correttamente i risultati ottenuti. Questa verterà quindi sulla descrizione dell'elettrocardiogramma, della fotopletismografia e, infine, della pressione sanguigna, definendo le possibili anomalie e gli strumenti attualmente utilizzati per la loro misurazione.
In merito alla strumentazione di misurazione, il capitolo successivo verterà sulle caratteristiche tecniche dei dispositivi indossabili che utilizzano come interfaccia le applicazioni sviluppate: EcgWatch e PulseEcg.
L'argomento seguente descrive i linguaggi di programmazione utilizzati sia per la programmazione dell'applicazione Android che per l'integrazione della rete neurale. Susseguentemente, verranno presentate nel dettaglio:

1. l'applicazione legata al dispositivo EcgWatch, specificando gli aggiornamenti effettuati;

2. la nuova applicazione connessa al dispositivo PulseEcg.

Verranno illustrati con precisione i filtri utilizzati nei segnali acquisiti e gli algoritmi eseguiti per i calcoli all'interno del programma.
In seguito verrà delineato il preprocessamento eseguito sulla rete neurale per renderla compatibile con l'applicazione Android e con il linguaggio di programmazione utilizzato (Java).
Al termine, saranno esposti alcuni dei risultati ottenuti durante la fase di testing e verranno trattate le conclusioni di questa tesi.

Capitolo 2

Concetti di medicina

L'applicazione utilizza delle notazioni mediche specifiche per lo sviluppo degli algoritmi presenti internamente. Nello specifico, le nozioni presenti nell'applicazione riguarderanno:

- elettrocardiogramma;
- fotopletismografia;
- frequenza cardiaca;
- saturazione di ossigeno nel sangue;
- pressioni sistolica e diastolica.

In questo capitolo verranno fornite le descrizioni di questi elementi così da permettere al lettore una maggiore comprensione dei temi trattati successivamente.

2.1 Elettrocardiogramma

Un elettrocardiogramma registra i segnali elettrici nel cuore. È un test comune e indolore usato per individuare rapidamente i problemi cardiaci e monitorare la salute. Aiuta a diagnosticare molte patologie cardiache comuni in persone di tutte le età, come, ad esempio, aritmie che possono portare a patologie più complesse. Gli elettrocardiogrammi (ECG o EKG) sono spesso eseguiti nello studio di un medico, in una clinica o in una stanza d'ospedale. Le strumentazioni per gli ECG sono attrezzature standard nelle sale operatorie e nelle ambulanze. Se i sintomi tendono però ad andare e venire, potrebbero non essere catturati durante una registrazione ECG standard. In questo caso può essere raccomandato il monitoraggio dell'elettrocardiogramma a distanza o continuo. Risultano quindi molto utili le tecnologie indossabili per la sua misurazione.

2.1.1 Descrizione

La funzione centrale del cuore è quella di contrarsi ritmicamente per pompare il sangue ai polmoni ossigenandolo e pompare nuovamente una volta arricchito di ossigeno nella circolazione generale (sistemica). Il segnale per la contrazione cardiaca è la diffusione di correnti elettriche attraverso il muscolo cardiaco. Queste correnti sono prodotte sia dalle cellule pacemaker e dal tessuto di conduzione specializzato all'interno del cuore che dal muscolo cardiaco stesso. La stimolazione elettrica degli atri destro e sinistro segnala a questi di contrarsi e di pompare il sangue contemporaneamente attraverso le valvole tricuspide e mitrale nei ventricoli destro e sinistro. Lo stimolo elettrico raggiunge quindi i tessuti di conduzione specializzati nella giunzione atrioventricolare (AV). Il segnale elettrico si diffonde poi simultaneamente lungo i rami del fascio di sinistra e di destra nel miocardio ventricolare (muscolo ventricolare) per mezzo di cellule conduttrici specializzate. Da qui, il segnale elettrico si diffonde attraverso il muscolo miocardico verso l'epicardio (bordo esterno). Proprio come la diffusione degli stimoli elettrici attraverso gli atri porta alla contrazione atriale, così la diffusione degli stimoli attraverso i ventricoli porta alla contrazione ventricolare, con pompaggio del sangue ai polmoni e nella circolazione generale [3].
Il termine più tecnico per il processo di attivazione cardiaca è depolarizzazione, derivato dal fatto che le normali cellule miocardiche "a riposo" sono polarizzate cioè portano cariche elettriche sulla loro superficie. La depolarizzazione produce una differenza di tensione elettrica sulla superficie esterna della cellula tra l'area depolarizzata stimolata e l'area polarizzata non stimolata. Il ritorno delle cellule del muscolo cardiaco al loro stato di riposo dopo la stimolazione (depolarizzazione) è chiamato ripolarizzazione.

La Fig. 2.1 mostra tutte le fasi di questo processo legandole alle onde visualizzate all'interno dell'elettrocardiogramma.

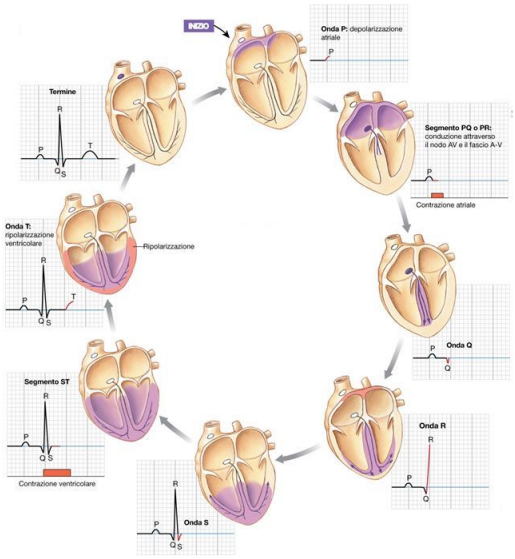


Figura 2.1: Ciclo di attività cardiaca

L'elettrocardiogramma (v. Fig. 2.2) è un grafico che rappresenta l'attività elettrica del cuore da un istante all'altro. Così, esso fornisce un grafico tempo-tensione del battito cardiaco. Per molti pazienti, questo test è una componente chiave della diagnosi e della gestione clinica sia in ricovero che in ambiente ambulatoriale. Il grafico è in grado di visualizzare solo le correnti relativamente grandi prodotte dalla massa del muscolo cardiaco funzionante. I segnali di ampiezza molto più piccola generati dal nodo del seno e dal nodo AV sono invisibili nelle registrazioni cliniche.

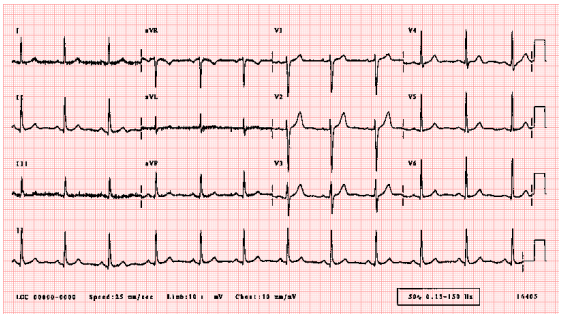


Figura 2.2: Segnale elettrocardiografico rappresentato su carta millimetrata

La corrente elettrica depolarizzante viene registrata dall'ECG come onda P (quando gli atri sono stimolati e si depolarizzano) e come complesso QRS (quando i ventricoli sono stimolati e si depolarizzano). La ripolarizzazione ventricolare è registrata dall'ECG come segmento ST, onda T e onda U (la ripolarizzazione atriale è solitamente oscurata dai potenziali ventricolari). Poiché la depolarizzazione e la ripolarizzazione cardiaca avvengono normalmente in modo sincronizzato, l'ECG è in grado di registrare queste correnti elettriche come forme d'onda specifiche (onda P, complesso QRS, segmento ST, onda T e onda U) rappresentate nella Fig. 2.3. In sintesi, che l'ECG sia normale o anormale, registra solo due eventi fondamentali: la depolarizzazione, cioè la diffusione di uno stimolo attraverso il muscolo cardiaco, e la ripolarizzazione, cioè il ritorno del muscolo cardiaco stimolato allo stato di riposo [4].

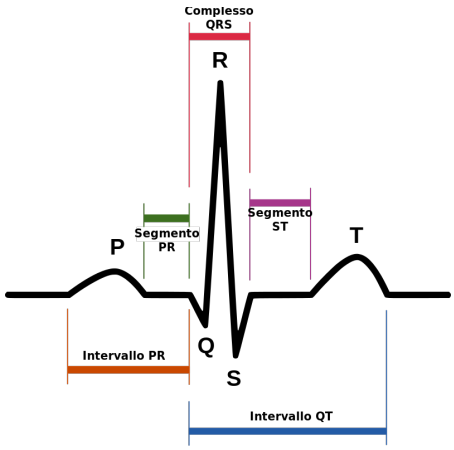


Figura 2.3: Dettaglio delle forme d'onda del segnale elettrocardiografico

2.1.2 Derivazioni

Gli elettrodi di registrazione posti ad una certa distanza dal cuore, come sulle braccia, sulle gambe o sulla parete del torace, sono in grado di rilevare le tensioni delle correnti cardiache condotte in questi luoghi. Il modo usuale di registrare queste tensioni dal cuore è con le 12 derivazioni standard dell'ECG. Le derivazioni mostrano effettivamente le differenze di tensione (potenziale) tra gli elettrodi posti sulla superficie del corpo. Si noti che ogni derivazione presenta un modello diverso, come se si osservasse il cuore da diversi angoli di visuale.

Le derivazioni possono essere suddivise in due gruppi: le sei derivazioni degli arti (estremità) e le sei derivazioni del torace (precordiali). Le derivazioni I, II e III sono le derivazioni standard (bipolari) degli arti (v. Fig. 2.4) e registrano le differenze di tensione elettrica tra le estremità. Queste possono essere rappresentate schematicamente in termini di un triangolo, chiamato Triangolo di Einthoven [5]. Il triangolo mostra l'orientamento spaziale delle tre derivazioni standard degli arti ottenendo che i conduttori bipolari sono collegati dalla seguente Eq. (2.1):

$$derivazioneI + derivazioneIII = derivazioneII \tag{2.1}$$

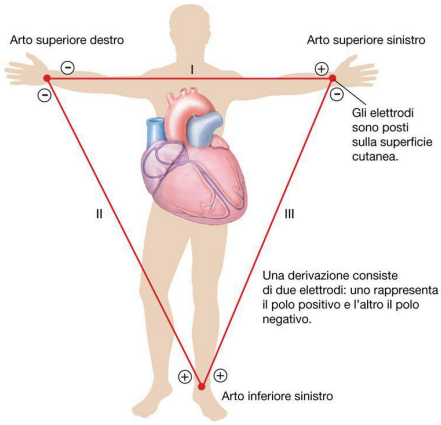


Figura 2.4: Disposizione degli elettrodi nelle derivazioni primarie degli arti I, II e III

2.1.3 Frequenza cardiaca

La frequenza cardiaca è il numero di volte in cui il cuore batte in un certo periodo di tempo, di solito un minuto. Essa in un adulto sano a riposo è normalmente tra i 60 e i 100 battiti al minuto. La sua misurazione fornisce importanti informazioni sulla salute di una persona. Si può facilmente determinare una frequenza media, che sia regolare o meno, semplicemente contando il numero di complessi QRS in un intervallo di tempo conveniente e moltiplicando questo numero per il fattore appropriato, ad esempio, 6 per ottenere la frequenza in battiti per 60 secondi. La frequenza cardiaca è inversamente correlata a un altro intervallo, il cosiddetto intervallo R-R (o intervallo QRS-QRS), cioè la distanza tra punti consecutivi ed equivalenti sul QRS precedente o successivo. Queste misurazioni, quando vengono effettuate utilizzando programmi informatici digitali su un gran numero di intervalli, costituiscono la base degli studi sulla variabilità della frequenza cardiaca (HRV).

Gli intervalli R-R possono essere convertiti in frequenza cardiaca (HR) con le seguenti due semplici formule equivalenti, a seconda che si misuri l'intervallo R-R in secondi (sec) (Eq. 2.2) o millisecondi (msec) (Eq. 2.3):

$$HR(bpm) = 1.0/RR(sec) * 60 \tag{2.2}$$

$$HR(bpm) = 1000/RR(ms) * 60 \tag{2.3}$$

Una frequenza cardiaca superiore a 100 battiti/min viene definita tachicardia, mentre una frequenza cardiaca inferiore a 60 battiti/min viene chiamata bradicardia [4].

2.1.4 Fibrillazione atriale

La fibrillazione atriale (FA) (v. Fig. 2.5) è un'aritmia cardiaca caratterizzata da una completa irregolarità dell'attivazione elettrica degli atri. In presenza di tale anomalia, le normali contrazioni atriali vengono sostituite da movimenti caotici, completamente inefficaci ai fini della propulsione del sangue, che aumentano il rischio di ictus cerebrali e influenzano negativamente l'emodinamica cardiovascolare. Questo tipo di sintomatologia cardiaca è presente in un grande numero di persone, soprattutto con l'avanzare dell'età [6].

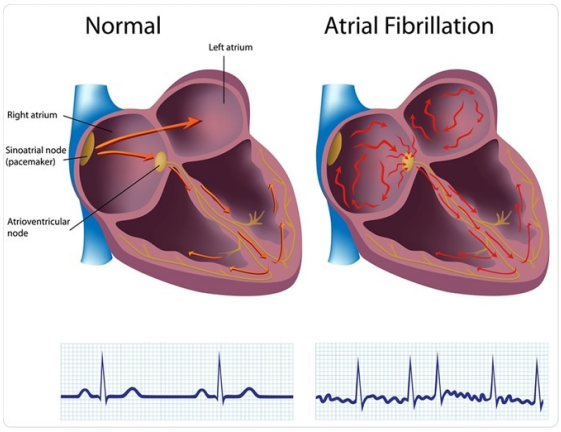


Figura 2.5: Esempio di fibrillazione atriale

La fibrillazione atriale può essere riconosciuta in un elettrocardiogramma (v. Fig. 2.5) attraverso due caratteristiche [7]:

1. scomparsa delle onde di attivazione atriale (onde P), che vengono sostituite da rapide oscillazioni della linea isoelettrica, dette onde di fibrillazione (onde f). Le onde f sono del tutto irregolari, con continue variazioni di forma, di voltaggio e degli intervalli f-f, hanno frequenza molto elevata (400- 600/minuto) e durano per tutto il ciclo cardiaco (sono continue), determinando un aspetto frastagliato della linea isoelettrica;

2. irregolarità degli intervalli R-R in corso di FA: un grande numero di impulsi di origine atriale raggiunge la giunzione atrioventricolare (AV), ma solo una parte di essi viene effettivamente trasmessa ai ventricoli. La quantità di impulsi che raggiunge i ventricoli dipende, infatti, dalle caratteristiche elettrofisiologiche del nodo AV e dalle altre porzioni del sistema di conduzione, dalla presenza di eventuali vie accessorie, dal tono del sistema nervoso autonomo e dall'azione di farmaci concomitanti. Tutte queste variabili contribuiscono alla costante variazione di durata degli intervalli R-R .

Sintomi	Totale(%)
Palpitazioni	54,1
Angina	10,1
Dispnea	44,4
Vertigini	10,4
Astenia	14,3
Altri	0,9
Nessuno	11,4

Tabella 2.1: Sintomi provocati dalla fibrillazione atriale con la percentuale di frequenza

All'interno della Tab. 2.1, possiamo vedere alcuni sintomi che possono essere associati alla presenza di fibrillazione atriale. In particolare, possiamo osservare come le percentuali siano molto più alte per quanto riguarda le palpitazioni e la dispnea. Con la presenza di tali sintomi è dunque utile effettuare un controllo dell'andamento del segnale elettrocardiografico.

2.2 Fotopletismografia

La fotopletismografia (v. Fig. 2.6) è una semplice tecnica ottica utilizzata per rilevare le variazioni volumetriche del sangue nella circolazione periferica. È un metodo a basso costo e non invasivo che effettua misurazioni sulla superficie della pelle. Il PPG utilizza la luce infrarossa (IR) a bassa intensità. Quando la luce viaggia attraverso i tessuti biologici viene assorbita dalle ossa, dai pigmenti della pelle e dal sangue sia venoso che arterioso. Poiché la luce è assorbita più fortemente

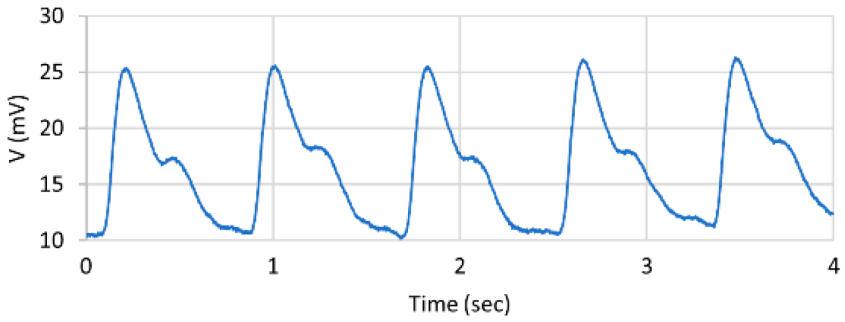


Figura 2.6: Esempio di segnale fotopletismografico

dal sangue che dai tessuti circostanti, i cambiamenti nel flusso sanguigno possono essere rilevati dai sensori PPG come cambiamenti nell'intensità della luce. Il segnale di tensione del PPG è proporzionale alla quantità di sangue che scorre nei vasi sanguigni. Possono essere rilevati persino piccoli cambiamenti nel volume del sangue con questo metodo, anche se non può essere utilizzato per quantificare la quantità di sangue [8].
Alcuni dei principali fattori che influenzano le registrazioni del PPG sono il sito di misurazione e la forza di contatto tra il sito e il sensore. La tecnologia della fotopletismografia è stata utilizzata in una vasta gamma di dispositivi medici disponibili in commercio per misurare la saturazione di ossigeno, la pressione sanguigna e la portata cardiaca, valutare la funzione autonoma e anche rilevare le malattie vascolari periferiche.

2.2.1 Saturazione di ossigeno nel sangue

La saturazione di ossigeno è la frazione di emoglobina satura di ossigeno rispetto all'emoglobina totale (insatura + satura) nel sangue. Il corpo umano richiede e regola un equilibrio molto preciso e specifico di ossigeno nel sangue. I normali livelli di saturazione dell'ossigeno nel sangue arterioso negli esseri umani sono del 95-100 per cento. Se il livello è inferiore al 90%, è considerato basso ed è chiamato ipossiemia. L'ossigenazione del sangue (SpO2) può essere determinata facendo brillare la luce rossa e poi quella del vicino infrarosso attraverso il tessuto vascolare, con una rapida commutazione tra le lunghezze d'onda. Le ampiezze dei segnali di corrente alternata (AC) rossi e nel vicino infrarosso sono sensibili ai cambiamenti della SpO2 a causa delle differenze nell'assorbimento della luce di HbO2 e Hb a queste due lunghezze d'onda. Dal loro rapporto di ampiezza e dalle corrispondenti componenti di corrente continua (DC) del PPG, è possibile stimare la SpO2. Questo processo di solito comporta un fattore di calibrazione derivato empiricamente. Si presuppone che la componente pulsatile del segnale PPG derivi esclusivamente dai cambiamenti di volume del sangue arterioso ad ogni battito cardiaco. I limiti della pulsossimetria sono che la tecnica si basa sulla presenza di un polso periferico, le letture della saturazione di ossigeno possono essere influenzate dalle disemoglobinemie e la sua precisione può diminuire a bassi livelli di saturazione [9]. Lo strumento che utilizza questa tecnica per la misurazione è chiamato pulsossimetro (v. Fig. 2.7). Altre tecnologie in via di sviluppo utilizzano una riflessione con luce verde al posto dell'infrarosso e della luce rossa.

Figura 2.7: Pulsossimetro Entweg, usato per calcolare il valore di saturazione di ossigeno

La Tab. 2.2 mostra la classificazione dei valori di saturazione del sangue. Un valore superiore a 95% indica una ossigenazione normale e non è sintomo di alcun problema; un valore tra 91% e 94% è da tenere sotto controllo poichè potrebbe indicare una possibile ipossia. Il valore deve essere riesaminato in maniera costante e sarebbe il caso di chiamare un medico se non dovesse risalire e si ha una sensazione di affanno. Sotto il 90% si è in una situazione di carenza di ossigeno e bisognerebbe contattare un medico.

Categoria	Ossigenazione(%)
Normale	> 95
Possibile ipossia	91 - 94
Ipossia	<91

Tabella 2.2: Classifica dei valori di saturazione di ossigeno nel sangue

2.3 Pressione sanguigna

La pressione sanguigna è la forza che muove il sangue attraverso il nostro sistema circolatorio e si crea quando il cuore si contrae ad ogni battito cardiaco per far uscire il sangue. È una forza importante perché senza di essa l'ossigeno e i nutrienti non verrebbero spinti nel nostro sistema circolatorio e non potrebbero quindi nutrire i tessuti e gli organi.
La pressione sanguigna è anche vitale perché fornisce i globuli bianchi e gli anticorpi per l'immunità e ormoni come l'insulina. Inoltre, il sangue puro che viene trasportato è in grado di raccogliere i prodotti di scarto tossici del metabolismo, compresa l'anidride carbonica che espiriamo ad ogni respiro, e le tossine che eliminiamo attraverso il fegato e i reni.
Il sangue stesso porta con sé una serie di altre proprietà tra cui la sua temperatura e le piastrine coagulanti che prevengono la perdita di sangue dopo una ferita, una delle nostre difese contro i danni ai tessuti.

La misurazione avviene attraverso un dispositivo chiamato sfigmomanometro (v. Fig. 2.8). Esistono diverse versioni di questo dispositivo, ad esempio sfigmomanometro al mercurio o elettronico, e utilizzano componenti diversi per la misurazione della pressione.

Figura 2.8: Sfigmomanometro elettrico per la misurazione delle pressioni sistolica e diastolica

La Tab. 2.3 mostra, in maniera semplificata, la classificazione dei valori per la pressione arteriosa, con la loro denominazione e rischio. Questi valori sono stati introdotti dal "International Society of Hypertension" (ISH) [10]. Anche il "National Institutes of Health" fornisce come indicazione per la pressione arteriosa normale dei valori al di sotto di 120 mmHg per la pressione sistolica e di 80 mmHg per la diastolica [11].

Categoria	Sistolica(mmHg)	Diastolica(mmHg)
Ottimale	<120	<80
Normale	< 139	< 89
Ipertensione di Grado 1 lieve	140 – 159	90 – 99
Ipertensione di Grado 2 moderata	160 – 179	100 – 109
Ipertensione di Grado 3 grave	> 179	> 109

Tabella 2.3: La classificazione dei valori di pressione arteriosa e della loro portata di rischio suggerita dall'OMS/ISH

Capitolo 3

Dispositivi indossabili

Gli strumenti di misurazione per gli elementi sopra citati, soprattutto per la pressione, possono essere ingombranti, complessi da utilizzare o poco precisi.
Nonostante esistano molti dispositivi indossabili che provano a calcolare queste informazioni, attualmente nessuno (o quasi) sembra avere una certificazione standard di Conformità Europea (CE). Questo vuol dire che non possiamo essere certi della validità dei risultati ottenuti dalle loro misurazioni. Per questo motivo, la continua ricerca di un dispositivo che fornisca risultati precisi, con un margine di errore accettabile, rimane un'azione motivata. I dispositivi utilizzati in questo progetto riescono ad avere un'alta precisione, confrontabile con la strumentazione utilizzata in ambito medico e certificata. Inoltre, sono stati progettati per essere espandibili e quindi aggiornare le proprie funzionalità nel corso degli anni.

3.1 EcgWatch

Il dispositivo EcgWatch (v. Fig. 3.1) è stato sviluppato con lo scopo di monitorare la condizione cardiaca dell'utente e individuare alcune anomalie che potrebbero presentarsi (ad esempio, la fibrillazione atriale descritta nella Sez. 2.5).

Figura 3.1: Dispositivo EcgWatch

La misurazione viene fatta utilizzando due elettrodi presenti nella copertura del bracciale: uno di essi è posto nella parte inferiore poggiando, quindi, sul polso dell'utente dal quale riuscirà a rilevare i segnali elettrici; l'altro, invece, si trova nella parte superiore e acquisirà il segnale elettrico solo quando verrà posto il dito sulla sua superficie.
Il dispositivo non presenta una interfaccia grafica diretta ma invia le misurazioni ad una applicazione sviluppata per Android. L'applicazione provvederà poi a filtrare in modo corretto il segnale e a visualizzarlo nel proprio smartphone.

3.1.1 Caratteristiche tecniche

Il dispositivo, il cui circuito stampato è rappresentato nella Fig. 3.2, è composto da:

1. un microcontrollore Texas Instrument l'MSP430G2955 che tiene conto dei vincoli quali una quantità sufficiente di memoria per permettere la memorizzazione dell'acquisizione, un dispositivo di I/O appropriato per interfacciare il modulo Bluetooth e un ADC abbastanza preciso;

2. due elettrodi di Ag/AgC, perché riutilizzabili e per il loro deterioramento quasi assente dovuto all'uso;

3. modulo Bluetooth 2.0 F2M03GLA, che supporta RFCOMM e permette la comunicazione via UART, GPIO o USB;

4. una batteria ai polimeri di litio, ricaricabile e in grado di fornire l'alto picco di corrente richiesto dal modulo Bluetooth trasmittente.

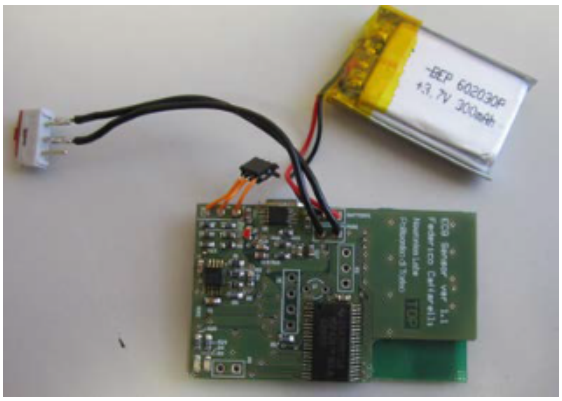


Figura 3.2: La scheda dell'EcgWatch senza la copertura superiore

Dalla Fig. 3.3 possiamo vedere il ciclo analogico svolto dal dispositivo.
Il primo stadio del progetto analogico è un filtro passa-alto utilizzato per annullare l'effetto batteria degli elettrodi e il potenziale DC del corpo umano sui due elettrodi. Il terzo filo è il riferimento pilotato dalla DRL. Questa parte comprende anche un sistema di protezione per gli utenti.
Il secondo stadio rappresenta la prima operazione di amplificazione. Al fine di ridurre il rumore causato dall'interferenza principale (50Hz), il segnale viene amplificato differentemente.
Il terzo stadio fornisce il filtro passa-banda e un secondo stadio di amplificazione. Il passa-banda del filtro è di circa [10-170Hz] e l'amplificazione è 10Hz. Presenta anche un polo del secondo ordine a 10Hz. Il componente utilizzato è l'OPA4241. Il sistema di feedback è un Driven Right Leg Circuit o "DRL". Si tratta di un circuito elettrico che viene spesso aggiunto agli amplificatori di segnali biologici per ridurre le interferenze di modo comune [12].

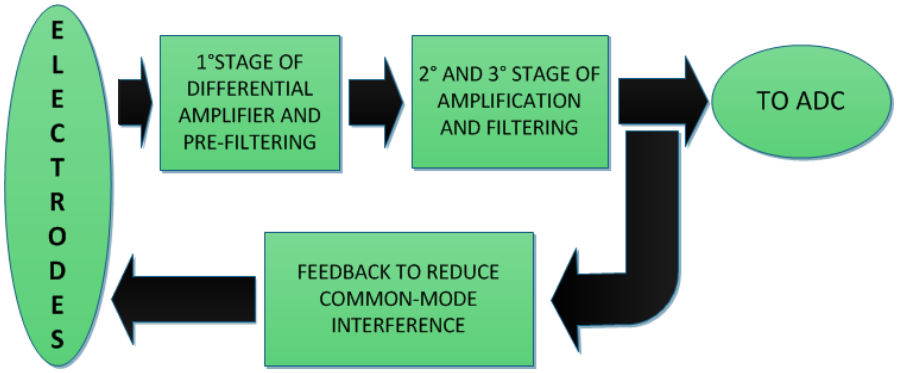


Figura 3.3: Operazioni analogiche svolte dall'EcgWatch

3.2 PulseEcg

Il dispositivo PulseEcg (v. Fig. 3.4) nasce come evoluzione dell'EcgWatch. Essendo un modello molto più recente (l'EcgWatch è stato sviluppato nel 2014, il PulseEcg nel 2019), le componenti utilizzate permettono una misurazione più precisa dell'elettrocardiogramma, riducendo il rumore dovuto dal segnale a 50Hz. Inoltre, il PulseEcg presenta un sensore per la fotopletismografia, calcolata attraverso la luce rossa e infrarossa.

Figura 3.4: Dispositivo PulseEcg

L'avanzamento tecnologico avviene anche nella connessione con lo smartphone, passando ad un Bluetooth Low Energy (BLE), ovvero Bluetooth 4.0, che permette un consumo di corrente inferiore per il dispositivo indossabile e una connessione stabile e continua con l'applicazione.

3.2.1 Caratteristiche tecniche

Il dispositivo può essere visto come due grandi blocchi costituiti da circuiti stampati (printed circuit board, PCB), uno specifico per la misurazione dell'elettrocardiogramma, l'altro, più piccolo, per la fotopletismografia, come vediamo in Fig. 3.5.
In particolare, il primo circuito è composto da:

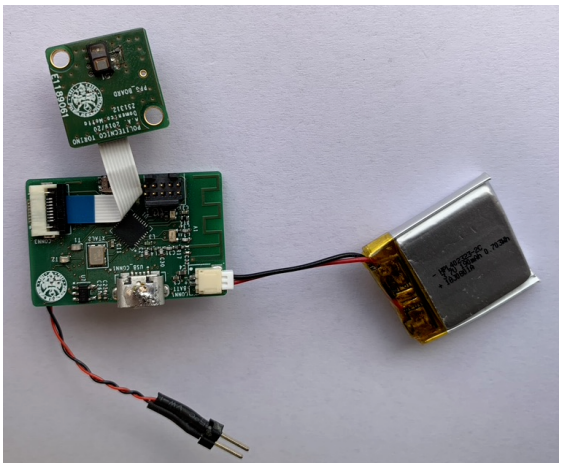


Figura 3.5: Dettaglio del chip del PulseEcg

1. connettori: il connettore degli elettrodi, il connettore USB, il connettore piatto e il connettore della batteria formano l'interazione tra il PCB e il mondo esterno;

2. modulo Bluetooth 4.0: usato per la comunicazione dei dati ad altri dispositivi, in particolare per inviare i dati all'applicazione sviluppata da questa tesi;

3. blocchi di alimentazione: Battery Charger, Battery Gauge sono usati per controllare lo stato della batteria e la sua modalità di carica; Voltage Regulator e Analog Power Domain sono due componenti che generano una tensione di alimentazione stabile e una tensione di riferimento usata da tutti gli altri componenti che devono avere un'alimentazione stabile;

4. filtri: questo blocco è utilizzato per il condizionamento del segnale elettrocardiografico;

5. microcontrollore Texas Instruments CC2640R2F: controlla tutti gli altri blocchi ed è usato per raccogliere i dati dell'elettrocardiogramma attraverso il suo convertitore analogico-digitale (ADC) interno mentre il valore di carica della batteria e il valore della fotopletismografia attraverso la comunicazione I2C.

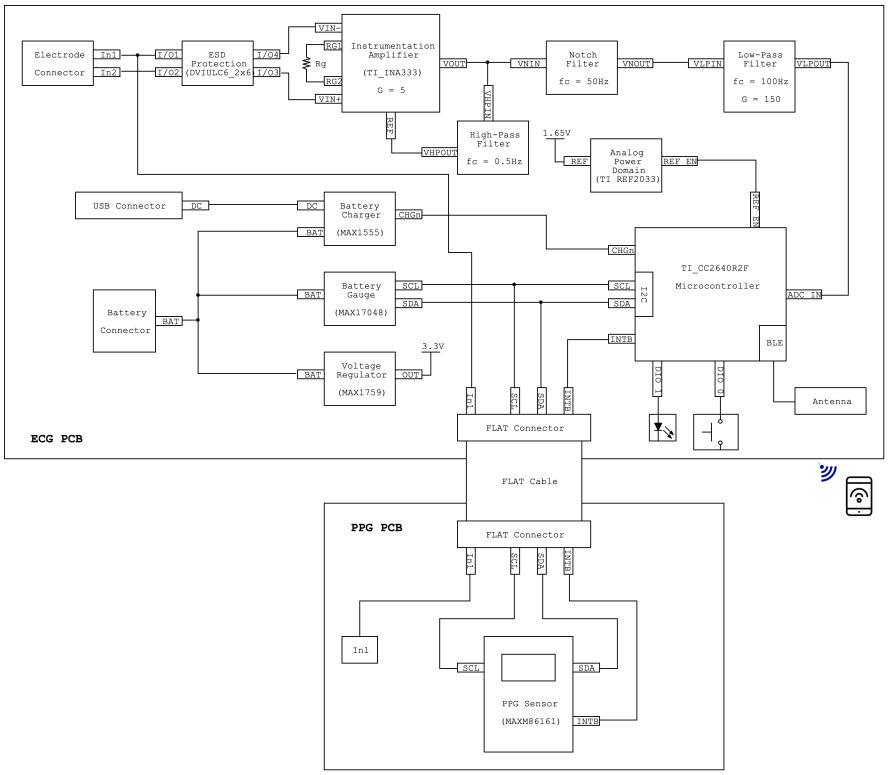


Figura 3.6: Schema tecnico del PulseEcg

Il circuito che si occupa della misurazione della fotopletismografia possiede due componenti principali:

1. connettore piatto: usato per collegare il circuito stampato al circuito principale attraverso il cavo piatto che porta le linee di alimentazione e le linee I2C;

2. sensore Maxim Integrated MAXM86161: impiegato per ricavare i dati utili per la fotopletismografia, utilizza la luce rossa e infrarossa.

Possiamo vedere i dettagli del circuito appena descritto nello schema tecnico in Fig. 3.6. Il dispositivo riesce dunque a trasmettere in tempo reale e in modo sincrono i segnali elettrocardiografici e fotopletismografici permettendo così un'analisi dettagliata e una elaborazione specifica attraverso l'applicazione implementata in Android [13].

3.3 Bluetooth

Bluetooth è un protocollo standard IEEE [14] di comunicazione per lo scambio di dati tra dispositivi fissi e mobili in sostituzione dei cavi dati [15], progettato principalmente per un basso consumo energetico, con un breve raggio d'azione basato su microchip ricetrasmittenti a basso costo in ogni dispositivo. Il protocollo utilizza onde radio con frequenze "ultra" alte (UHF) nelle bande ISM, da 2,402 GHz a 2,480 GHz, e costruendo reti personali (PAN). Poiché i dispositivi utilizzano un sistema di comunicazione radio (broadcast), non devono essere in linea visiva l'uno con l'altro; tuttavia, un percorso wireless quasi ottico deve essere praticabile. La portata dipende dalla classe di potenza [16]. I dispositivi dotati di Bluetooth si dividono in 4 classi di potenza di trasmissione che possiamo vedere nella Tab. 3.1. Questa si basa sulla potenza ERP, cioè la massima potenza trasmissiva in radiofrequenza comprendente l'incremento dovuto al guadagno in trasmissione dell'antenna del dispositivo, e sulla distanza, ovvero il raggio massimo di copertura a portata ottica (senza ostacoli) entro cui può avvenire il collegamento fra dispositivi BT.

Classe	Potenza ERP(mW)	Distanza(m)
1	100	100
2	2,5	10
3	1	1
4	0,5	0,5

Tabella 3.1: Classifica delle classi di Bluetooth in base alla potenza ERP del segnale

3.3.1 Bluetooth 2.0

Questa versione della specifica di base del Bluetooth è stata rilasciata prima del 2005. La differenza principale sta nell'introduzione di un Enhanced Data Rate (EDR) per un trasferimento dati più veloce. Il bit rate di questa modalità è di 3 Mbit/s, anche se la massima velocità di trasferimento dati è di 2,1 Mbit/s. L'Enhanced Data Rate utilizza una combinazione della Variazione di Frequenza Gaussiana (GFSK) e della modulazione a spostamento di fase (PSK) con due varianti. Inoltre, la modalità può fornire un consumo energetico inferiore attraverso un "duty cycle" ridotto, risultando compatibile con gli oggetti elettronici portatili.

3.3.2 Bluetooth Low Energy

La tecnologia Bluetooth Low Energy opera nella stessa gamma di frequenze (ISM) della tecnologia Bluetooth classica ma utilizza un diverso insieme di canali: invece dei classici 79 canali Bluetooth di 1-MHz, Bluetooth Low Energy ha 40 canali di 2-MHz. All'interno di un canale, i dati vengono trasmessi utilizzando la Modulazione di Variazione di Frequenza Gaussiana (GFSK), simile al classico rateo di base del Bluetooth. Il bit rate è di 1 Mbit/s e la potenza di trasmissione massima è di 10 mW.
Lo stack di protocollo BLE è composto da due parti principali: il Controller e l'Host. Il Controller comprende il Physical Layer e il Link Layer (v. Fig. 3.7), ed è tipicamente implementato come un piccolo System-on-Chip (SOC). L'Host gira su un processore applicativo e include funzionalità di livello superiore, cioè il Logical Link Control and Adaptation Protocol (L2CAP), l'Attribute Protocol (ATT), il Generic Attribute Profile (GATT), il Security Manager Protocol (SMP) e il Generic Access Profile (GAP). La comunicazione tra l'host e il controller è standardizzata come Host Controller Interface (HCI) [17].
A differenza del Bluetooth classico, il Bluetooth Low Energy è progettato per

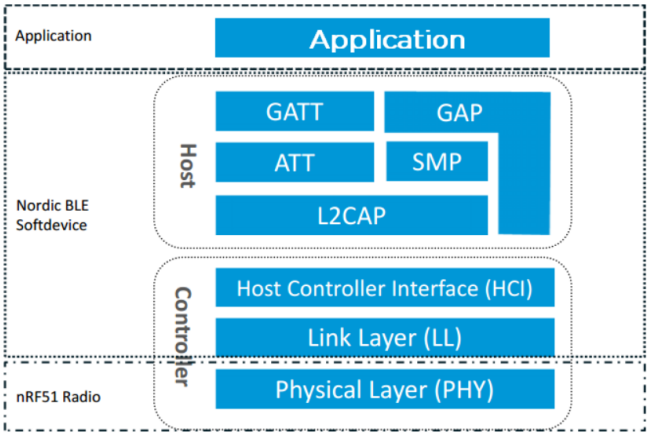


Figura 3.7: Schema di un Bluetooth Low Energy

fornire un consumo energetico significativamente inferiore. Ciò è possibile a causa dell'efficienza energetica del protocollo Bluetooth Low Energy che trasmette solo piccoli pacchetti rispetto al Bluetooth classico. Questo permette alle applicazioni Android di comunicare con i dispositivi BLE che hanno requisiti di alimentazione più severi, come i sensori di prossimità, i cardiofrequenzimetri e i dispositivi di fitness. Il Bluetooth Low Energy rappresenta quindi un compromesso tra consumo energetico, latenza, dimensioni della piconet e throughput.

Questo protocollo possiede degli elementi chiave:

- Generic Attribute Profile (GATT): definisce il modo in cui due dispositivi Bluetooth Low Energy trasferiscono dati usando concetti chiamati servizi e caratteristiche. Fa uso di un protocollo dati generico chiamato Attribute Protocol (ATT), che viene utilizzato per memorizzare servizi, caratteristiche e dati correlati in una semplice tabella di ricerca utilizzando ID a 16 bit per ogni voce della tabella;

- servizi: sono usati per suddividere i dati in entità logiche e contengono pezzi specifici di dati chiamati caratteristiche. Un servizio può avere una o più caratteristiche, e ogni servizio si distingue dagli altri servizi per mezzo di un identificatore numerico unico chiamato UUID, che può essere a 16 bit (per i servizi BLE adottati ufficialmente) o a 128 bit (per i servizi personalizzati);

- caratteristica: anche questa riconosciuta tramite un UUID, è il concetto di livello più basso nelle transazioni GATT. Incapsula un singolo punto di dati ed è il punto principale in cui si interagisce con una periferica BLE;

- descrittore: valore opzionale che fornisce ulteriori informazioni su una caratteristica.

Le connessioni effettuate tramite protocollo GATT sono esclusive. Ciò significa che una periferica BLE può essere collegata solo a un dispositivo centrale alla volta. All'interno del protocollo dovremo, infatti, definire dei dispositivi come server e client che comunicheranno tra loro.
Le caratteristiche possono essere di lettura o di scrittura. A queste caratteristiche è però possibile fornire una configurazione ad una specifica caratteristica attraverso il "Client Characteristic Configuration Descriptor" (CCCD), un descrittore dedicato a tale funzionalità. In particolare, siamo interessati, per questo progetto, a una impostazione per l'invio dei dati: "Notify". La sottoprocedura di notifica è usata quando un server è configurato per inviare un valore caratteristico a un client senza aspettare una conferma di recezione del livello di protocollo degli attributi. Il client può quindi ricevere dati asincroni dal dispositivo senza dover inviare alcun messaggio al server con conseguenti vantaggi nella velocità e nell'efficienza energetica seppur a discapito della sicurezza.

Capitolo 4

Ambiente di sviluppo

In questo capitolo vedremo alcuni dettagli sui linguaggi di programmazione e sulle librerie utilizzate durante il progetto di tesi. In particolare, verranno descritti tutti i dettagli per la programmazione dell'applicazione in Android e una introduzione a Python, dove è stata implementata la rete neurale. Oltre a questi, durante il progetto è stato utilizzato anche il software di Matlab per poter verificare i filtri, che sono poi stati implementati, e per poter elaborare alcuni dei dati forniti alla rete.

4.1 Android

Android è un sistema operativo creato da Google basato su kernel linux [18] e, inizialmente, progettato per sistemi embedded quali smartphone, tablet, smartwatch e altri. La programmazione per tali dispositivi è fornita dalla possibilità di creare interfacce utente personalizzate in base alle caratteristiche da esso possedute.
Questo sistema ha subito uno sviluppo molto rapido negli anni, probabilmente dovuto alla licenza libera Apache 2.0 che permette a chiunque sia interessato di potersi cimentare nella sua programmazione, seppur con dei limiti [19].
Attualmente, l'ultima versione di Android attualmente disponibile è la 11 (API 30), ma sono presenti delle anteprime della versione 12. Nonostante ciò, le versioni di Android precedenti sono ancora utilizzate dagli utenti poichè non tutti i dispositivi permettono l'aggiornamento della versione. Per questo risulta importante mantenere la retrocompatibilità delle applicazioni con le versioni precedenti, considerando anche il numero enorme di utenti che utilizzano questo sistema operativo [20].

4.1.1 Kernel Linux

Alla base dello stack Android c'è il kernel Linux. Esso non interagisce mai veramente con gli utenti e gli sviluppatori, ma è il cuore dell'intero sistema. Il kernel agisce come un livello di astrazione hardware (HAL) tra il livello fisico del dispositivo e lo stack software Android. Questo gestisce:

1. I permessi e la sicurezza;
2. La memoria di basso livello;
3. Processi e thread;
4. Strato di rete;
5. Display, tastiera, fotocamera, memoria flash, file audio;

4.1.2 Ambiente di esecuzione

Inizialmente, il sistema operativo Android utilizzava la macchina virtuale Dalvik per l'esecuzione delle applicazioni. Questa era ottimizzata per sfruttare la poca memoria presente nei dispositivi mobili, consentiva di far girare diverse istanze della macchina virtuale contemporaneamente e nascondeva al sistema operativo sottostante la gestione della memoria e dei thread. Dalla versione di Android 5.0 (Lollipop), la macchina virtuale Dalvik è ufficialmente sostituita dalla runtime ART (Android RunTime) [21]. ART è basata su tecnologia Ahead-Of-Time (AOT), che esegue l'intera compilazione del codice durante l'installazione dell'app e non durante l'esecuzione stessa del software, portando vantaggi in termini di prestazioni e gestione delle risorse. Questo guadagno di prestazioni e gestione delle risorse in fase di esecuzione si traduce in un maggior costo nei tempi di installazione di un'app, costo che comunque deve essere pagato solo una volta, e che è generalmente quasi impercettibile, specie in vista di hardware sempre più potenti [22].

4.2 Android Studio

Android Studio è l'ambiente di sviluppo integrato (IDE) più utilizzato per applicazioni Android e vanta di un'ampia gamma di funzionalità utili per facilitare la programmazione. Questo software consente di sviluppare applicazioni attraverso linguaggi di programmazione a oggetti, quali Java e Kotlin, e metalinguaggi per la definizione di linguaggi di markup, quale XML. Inoltre, permette una facile integrazione dei modelli di machine learning, creati attraverso Tensorflow, sfruttando le librerie di Tensorflow Lite (fare riferimento al Cap. 8).
La Fig. 4.1 viene presentata una finestra di lavoro di Android Studio in cui, sulla destra, si può osservare la scheda per la modifica dei layout, una funzionalità molto utile per sviluppare la grafica visualizzando direttamente il risultato (la grafica presentata corrisponde ad un determinato file XML, nel caso specifico activity_main.xml), mentre sulla sinistra si possono osservare alcuni dei file contenuti nel progetto dell'applicazione PulseEcg.

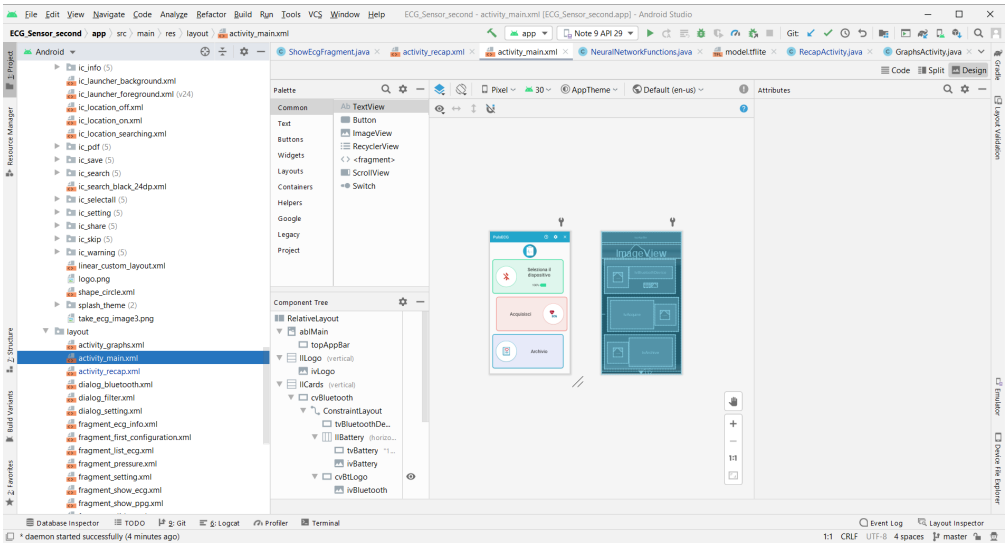


Figura 4.1: Finestra di lavoro di Android Studio

4.3 Java

Java è un linguaggio di programmazione ad alto livello, orientato agli oggetti e alla tipizzazione statica, specificamente progettato per essere il più possibile indipendente dalla piattaforma hardware di esecuzione. La versione utilizzata all'interno dell'applicazione è la 1.8 che permette di semplificare il codice in maniera efficiente grazie soprattutto alle funzioni lambda.

La scelta del linguaggio è ricaduta su Java (e non su Kotlin) in quanto l'applicazione precedentemente utilizzata per l'iterfacciamento con il dispositivo era descritta con questo linguaggio, quindi, per semplicità, è stato deciso di utilizzarlo per ripristinare alcune delle funzioni dichiarate precedentemente.

4.3.1 Librerie

All'interno del linguaggio Java sono state utilizzate alcune librerie esterne per poter gestire in modo efficiente alcune delle funzionalità presenti nelle applicazioni. L'utilizzo di librerie è infatti consigliato in quanto sono il frutto di un lavoro lungo che coinvolge un vasto gruppo di lavoro. Verranno quindi risolti eventuali problemi e aggiunte nuove funzionalità. Non si potrebbero ottenere gli stessi risultati attraverso una programmazione veloce e non specifica.

MPAndroid Chart La libreria MPAndroid Chart [23] è stata selezionata per la rappresentazione dei grafici nelle applicazioni EcgWatch e PulseEcg, sostituendo la libreria importata precedentemente, ovvero la GraphView. La vecchia libreria era infatti diventata obsoleta [24] soprattutto dopo che il creatore ha segnalato che non avrebbe più supportato il progetto. Inoltre, MPAndroid Chart ha una versione compatibile per iOS, utile per la creazione dell'applicazione PulseEcg svolta in parallelo per i dispositivi Apple.

Figura 4.2: Grafico di esempio di MPAndroidChart

La libreria ha permesso una gestione semplice dei grafici sia dal punto di vista grafico che di elaborazione. La possibilità di visualizzare molteplici segnali indipendenti su un grafico (v. Fig. 4.2), ci ha permesso di visualizzare in maniera specifica i picchi R. Inoltre, con delle opportune modifiche alla libreria, è stato possibile mostrare un grafico molto preciso, con una griglia che corrisponde alla carta millimetrata utilizzata in ambito medico.

Google Play Services - Location Le due applicazioni possono, in base alla richiesta dell'utente, inserire la posizione o all'interno del corpo di un messaggio durante la condivisione o direttamente all'interno del PDF. Inizialmente era stata aggiunta una localizzazione GPS nativa che rilevava la posizione direttamente dal dispositivo. Purtroppo, la soluzione risultava lenta, poco precisa e dipendente sia dalle capacità hardware del dispositivo che dalla posizione dello smartphone. In una situazione di emergenza questi problemi non sono certamente ammissibili.
Google fornisce diverse librerie per migliorare l'esperienza sia degli utenti Android che degli sviluppatori di software attraverso una suite di risorse chiamate Google Play Services (GPSS); questi servizi vanno dalle API per interagire con le funzioni di Google Sign In, Mobile Vision, rilevare volti o codici QR, alle API per interagire meglio con l'hardware di localizzazione integrato. Con l'integrazione del servizio di localizzazione della Google, i problemi illustrati sono scomparsi ottenenendo stabilità e affidabilità nel sistema. I servizi di Google Play possono essere eseguite in background su ogni dispositivo Android certificato da Google e utilizzano delle funzioni fornite dal produttore per accedere ad essi quando necessario [25]. Con questa scelta però, la localizzazione diventa dipendente da una libreria non open source e disponibile solo per smartphone che hanno accesso ai servizi Google.

Gson Per la memorizzazione dei dati raccolti dal dispositivo, sono stati utilizzati, nell'applicazione PulseEcg, dei file di tipo Json (JavaScript Object Notation). La scelta è ricaduta su questo tipo di file per la loro flessibilità e compatibile con diversi linguaggi di programmazione.
La libreria utilizzata per la serializzazione e la deserializzazione dei dati è Gson [26], implementata da Google. Essa fornisce delle API molto semplici in grado di ricostruire un oggetto di una classe custom direttamente da un file Json (se il formato del file risulta corretto) e di creare un file Json da un oggetto. Per fare questo, è necessario dichiarare la funzione toString() all'interno della classe da serializzare.

4.3.2 eXtensible Markup Language (XML)

Il linguaggio di markup eXtensible Markup Language (XML) viene utilizzato in Android Studio per la descrizione dell'interfaccia grafica dell'applicazione in modo facile e immediato. L'ambiente di sviluppo possiede, infatti, la possibilità di vedere istantaneamente il risultato delle modifiche attuate sui file XML (v. Fig. 4.3). Questo metalinguaggio possiede molti vantaggi:

1. meno sforzi di manutenzione;
2. migliore riutilizzo;
3. maggiore capacità di mirare a dispositivi multipli;
4. possibilità di ridurre le risorse dedicate alla implementazione di progetti.

L'integrazione di questo linguaggio in Android è molto vasto e prevede molte librerie per la gestione efficiente delle risorse grafiche. Nell'applicazione EcgWatch, il layout risulta più semplice e compatibile con dispositivi di vecchia generazione. L'applicazione PulseEcg possiede dei costrutti più innovativi, come, per esempio, il vasto utilizzo della libreria MaterialDesign fornita da Google che rende l'esperienza utente maggiormente piacevole. La Fig. 4.3 mostra del codice XML all'interno di Android per lo sviluppo di una semplice interfaccia grafica (visibile alla destra del codice).

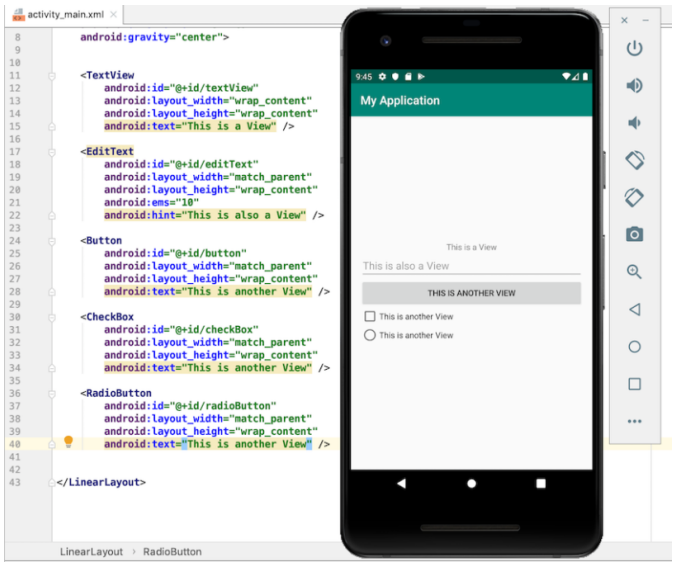


Figura 4.3: Esempio di script XML su Android Studio

Material Design Components Il Material Design è una guida completa per la progettazione visiva, del movimento e dell'interazione, attraverso le piattaforme e i dispositivi. Per utilizzare il Material Design è possibile importare una libreria di supporto all'interno dell'applicazione. La libreria comprende nuovi componenti e nuovi stili da utilizzare nell'interfaccia grafica.
Android fornisce le seguenti caratteristiche per costruire applicazioni in material design, alcune delle quali sono visibili nella Fig. 4.4:

1. un tema per app in material design per dare stile a tutti i tuoi widget UI;
2. widget per viste complesse come elenchi e schede;
3. nuove API per ombre e animazioni personalizzate.

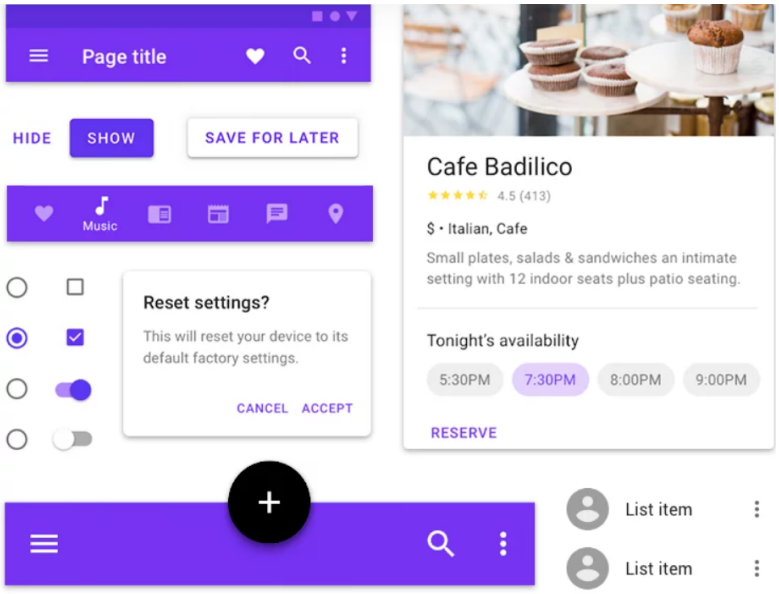


Figura 4.4: Esempi di componenti in material design

4.4 Python

Python è un linguaggio di programmazione di più alto livello rispetto alla maggior parte degli altri linguaggi, è orientato a oggetti ed è adatto, tra gli altri usi, a sviluppare applicazioni distribuite, scripting, computazione numerica e system testing. Questo linguaggio è stato utilizzato per progettare la rete neurale inserita poi all'interno dell'applicazione PulseEcg. In particolare, sono stati utilizzati Google Colaboratory come piattaforma di sviluppo (v. Fig. 4.5) e alcuni file in formato ipynb. Questo tipo di formato è molto comodo in quanto consente l'esecuzione di alcune parti di codice rispetto ad altre e altre come, per esempio, saltare il codice per l'esecuzione del training. Per descrivere la rete neurale è stata utilizzata la libreria Tensorflow 1.15. Durante la tesi, è stato generalizzato il codice Python in modo da poter funzionare con qualsiasi tipo di dato in input (prima specifico per un solo file) ed è stata creata una versione con Tensorflow 2 che possiede numerosi vantaggi rispetto a quella precedente [27].

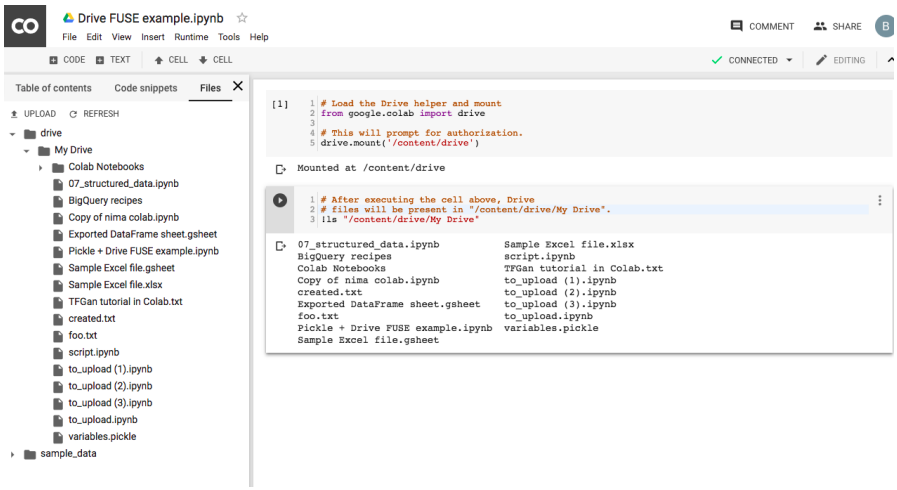


Figura 4.5: Esempio di un ambiente di lavoro Google Colaboratory

Capitolo 5

Aggiornamento applicazione EcgWatch

La prima fase del progetto corrisponde all'aggiornamento di una applicazione chiamata Ecg Sensor e presente all'interno del Google Play Store. Questa applicazione è stato il primo interfacciamento con il bracciale progettato nel 2014 e descritto nel Cap. 3.1. L'aggiornamento riguarda sia l'esperienza utente, cambiando la grafica dell'applicazione e inserendo funzionalità utili per la gestione da parte dell'utente, sia alcune utilità all'interno dei segnali, come, per esempio, l'integrazione di una griglia specifica. In questo capitolo farò una introduzione alla vecchia applicazione e successivamente verrà descritta nel dettaglio l'applicazione aggiornata EcgWatch, evidenziando le varie le funzionalità aggiunte e i cambiamenti effettuati rispetto alla versione precedente.

5.1 EcgWatch prima versione: Ecg Sensor

La prima versione dell'applicazione per la connessione con il bracciale EcgWatch (v. Cap. 3.1), fu concepita per fare dei test riguardo la comunicazione con il dispositivo indossabile e osservare la qualità del filtraggio dell'elettrocardiogramma attraverso un dispositivo che possa essere alla portata di tutti, ovvero lo smartphone o il tablet [12]. Questa versione è risultata però poco adatta all'esperienza dell'utente in quanto poco intuitiva e non molto gradevole graficamente. Inoltre, le innovazioni che sono state presentate negli anni, dal punto di vista sia del sistema operativo Android che dell'ambiente di sviluppo, sono state molteplici. Per avere informazioni sulle funzioni riguardo l'elaborazione del segnale elettrocardiografico, rimando la lettura al Cap. 7.

5.1.1 Schermata di caricamento

La schermata di caricamento iniziale, che vediamo presentata in Fig. 5.1, prevede un'animazione con sonoro per un determinato numero di secondi (5 secondi). Questa schermata di caricamento però risulta fittizia e non attende la creazione di nessun componente. L'animazione viene quindi mostrata all'utente con il solo scopo di avere una introduzione all'applicazione. L'utente però si ritroverà ad aspettare per troppo tempo su una schermata che non porta a nulla, minando l'esperienza e perdendo dei secondi che possono essere importanti in una situazione di emergenza.

Figura 5.1: Schermata di caricamento iniziale

5.1.2 Pagina Principale

La schermata principale di Ecg Sensor, visualizzata in Fig. 5.2, presenta tre bottoni:

1. Fig. 5.2A: il bottone inizia il processo di acquisizione inviando un messaggio (con scritto il numero di secondi dell'acquisizione) al dispositivo tramite Bluetooth. Questo processo viene identificato da notifiche di tipo Toast che non risultano molto chiare e che spariscono durante l'acquisizione. L'utente non avrà nessuna conoscenza riguardo la misurazione, nè se sta procedendo in maniera corretta o si è presentato qualche errore, nè riguardo il suo avanzamento;

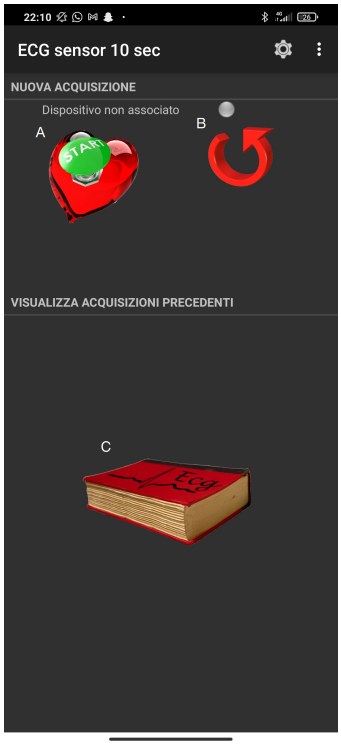


Figura 5.2: Schermata principale dell'applicazione. A: pulsante per inizio acquisione; B: pulsante per ricerca dispositivo Bluetooth; C: pulsante per accedere alla schermata "Archivio"

2. Fig. 5.2B: la freccia identifica la ricerca del dispositivo Bluetooth. Purtroppo, premendo il bottone non sarà possibile selezionare il dispositivo associato attraverso una lista ma verrà inserito manualmente all'interno delle impostazioni (v. Sez. 5.1.3). L'immagine segnalerà la connessione avvenuta con successo tramite un cerchio posto in alto a sinistra: se rosso, il dispositivo non è connesso, se verde, la connessione è stata effettuata (v. Fig. 5.3);

Figura 5.3: Segnale di Bluetooth attivo/disattivo

3. Fig. 5.2C: l'immagine con il libro aprirà l'attività da cui selezionare delle acquisizioni precedenti (Rif. 5.1.4). Questa non presenta nessuna animazione alla pressione.

In alto a destra sono presenti l'icona per aprire le impostazioni e l'icona, inserita all'interno di un menù a scorrimento, per la chiusura dell'applicazione.

5.1.3 Impostazioni

Le impostazioni dell'applicazione Ecg Sensor (v. Fig. 5.4) permettono di cambiare alcune informazioni dell'utente quali nome, cognome e email del medico a cui mandare i file. Il nome e cognome vengono utilizzati per il nome dei file. L'impostazione di autosalvataggio permette di scegliere se salvare il file in automatico all'interno del dispositivo.

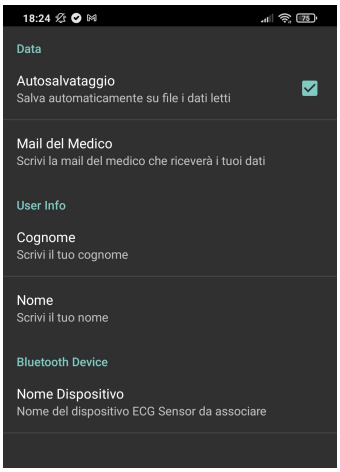


Figura 5.4: Impostazioni di Ecg Sensor

L'ultima voce della Fig. 5.4, nome dispositivo, assume un ruolo fondamentale per poter selezionare il dispositivo da cui acquisire. Il nome deve essere scritto manualmente all'interno di un dialog (v. Fig. 5.5), senza la possibilità di selezionarlo da qualche lista. Bisogna quindi essere a conoscenza del nome esatto del dispositivo. Inoltre, il bracciale deve essere stato precedentemente associato allo smartphone, ma la necessità di questa azione non viene indicata. Infine, le informazioni che vengono modificate dall'utente non vengono mostrate direttamente nella grafica ma è necessario premere il testo per conoscerne il loro valore.

Figura 5.5: Dialog per inserire il nome del dispositivo da collegare

5.1.4 Archivio

La sezione "Archivio" permette di navigare all'interno di qualsiasi cartella dello smartphone, indipendentemente che questa cartella sia o meno legata in qualche modo all'applicazione. Questo può creare alcuni problemi di sicurezza e incompatibilità con le nuove versioni di Android che hanno un maggiore controllo sui permessi di accesso alla memoria esterna.
In questa schermata manca inoltre un'opzione per tornare indietro in quanto la freccia verde serve per navigare all'interno della memoria. Nella Fig. 5.6, vediamo la cartella dedicata all'applicazione all'interno della memoria esterna con i due file di esempio che vengono caricati automaticamente durante l'installazione. Di questi file possiamo vedere le dimensioni, la data, l'ora di creazione e il nome dello stesso.

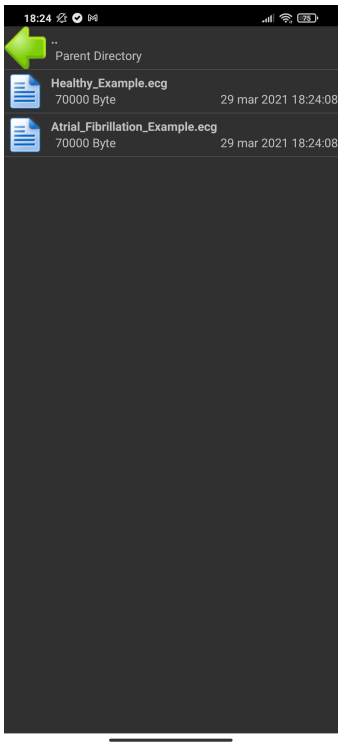


Figura 5.6: Gestione dei file da parte dell'applicazione

5.1.5 Grafico dell'elettrocardiogramma

Come illustrato precedentemente nella Sez. 4.3.1, la GraphView contiene parecchi limiti. Il grafico infatti non possiede una griglia significativa e non vi è il permesso di aggiungere il numero di linee richieste. Il segnale viene invece rappresentato in maniera corretta grazie anche ai filtri applicati (v. Cap. 7).
Le funzioni presenti nella barra di applicazione non sono tutte funzionanti. In particolare, la funzione di invio del file crea un errore nei nuovi dispositivi perchè mancano dei permessi per l'accesso alla memoria esterna (come su già descritto). Vediamo un esempio nella Fig. 5.7.

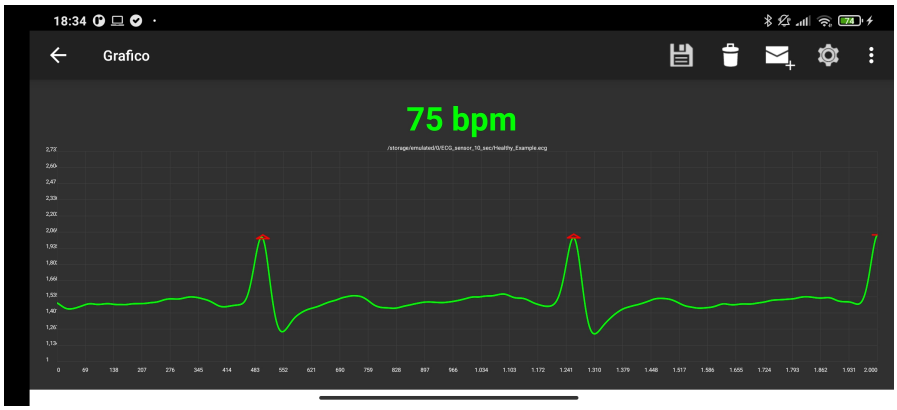


Figura 5.7: Elettrocardiogramma rappresentato tramite GraphView

Un elettrocardiogramma che presenta una fibrillazione atriale, illustrato nella Fig. 5.8, verrà segnalato solo attraverso una notifica sonora ad altissimo volume, poco significativa ed efficace.

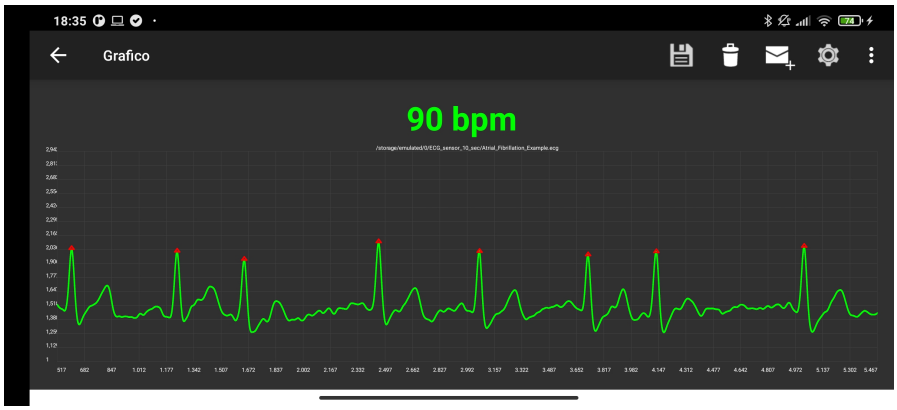


Figura 5.8: Grafico elettrocardiografico con una fibrillazione atriale

5.2 EcgWatch seconda versione: Front-end

Figura 5.9: Logo dell'applicazione EcgWatch

L'applicazione EcgWatch, che possiede il logo rappresentato nella Fig. 5.9, vuole essere un aggiornamento dell'applicazione Ecg Sensor. La struttura dell'applicazione si basa su una serie di attività che gestiscono le varie "sezioni" dell'applicazione. Le parti dell'applicazione che verranno visualizzate consistono in:

1. schermata di caricamento: gestita direttamente come layout del main activity;
2. presentazione dell'applicazione: visualizzata automaticamente al primo accesso e poi disponibile tramite la barra di applicazione;
3. prima configurazione: mostrata solo al primo accesso, consente di inserire i valori di configurazione;
4. pagina principale: contiene le componenti che permettono di accedere alle varie funzionalità e sezioni dell'applicazione;
5. schermata di acquisizione: mostrata quando si ricevono dati dal dispositivo indossabile fornendo informazioni sull'avanzamento del processo;
6. archivio: permette di visualizzare la lista di file contenenti gli elettrocardiogrammi precedentemente misurati;
7. elettrocardiogramma: permette la visualizzazione della misurazione effettuata o di un vecchio file aperto dall'archivio;
8. informazioni generali sull'elettrocardiogramma: visualizza una schermata di spiegazione riguardo al segnale elettrocardiografico.

EcgWatch si presenta con un tema scuro, così come la vecchia applicazione. I colori secondari richiamano invece i colori del nuovo logo (v. Fig. 5.9). Tendenzialmente, i colori scuri sono stati individuati dai vari suggerimenti introdotti da Material Design, quindi sono utilizzati in maniera standard dalla maggior parte delle applicazioni [28].

colors.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorBackground">#171a21</color>
    <color name="colorSurface">#2a475e</color>
    <color name="colorPrimary">#1b2838</color>
    <color name="colorPrimaryDark">#001B3A</color>
    <color name="colorAccent">#218db1</color>
    <color name="colorText">#FCFCFC</color>
    <color name="colorError">#7E0315</color>

    <color name="grey_50">#FAFAFA</color>
    <color name="grey_100">#F5F5F5</color>
    <color name="grey_200">#EEEEEE</color>
    <color name="grey_300">#E0E0E0</color>
    <color name="grey_400">#BDBDBD</color>
    <color name="grey_500">#9E9E9E</color>
    <color name="grey_600">#757575</color>
    <color name="grey_700">#616161</color>
    <color name="grey_800">#424242</color>
    <color name="grey_900">#212121</color>
    <color name="grey_1000">#1a1a1a</color>

    <color name="grey_3">#f7f7f7</color>
    <color name="grey_5">#f2f2f2</color>
    <color name="grey_10">#e6e6e6</color>
    <color name="grey_20">#cccccc</color>
    <color name="grey_40">#999999</color>
    <color name="grey_60">#666666</color>
    <color name="grey_80">#37474F</color>
    <color name="grey_90">#263238</color>
</resources>
```

Figura 5.10: File color.xml utilizzato dall'applicazione EcgWatch

La Fig. 5.10 rappresenta la palette di colori utilizzata dall'applicazione. Come possiamo notare, è composta per lo più da varie saturazioni di grigio utili per creare degli effetti di sovrapposizione e di rilievo tra i vari elementi presenti.
L'applicazione è stata sviluppata e testata anche su telefoni di vecchia generazione, in particolare sulla versione 5 di Android (Fig. 5.11). La retrocompatibilità con le versioni di Android, permette a qualsiasi utente di utilizzare EcgWatch senza dover avere smartphone recenti. EcgWatch è infatti pensata per essere utilizzata anche da utenti più anziani che possono trarre particolare beneficio dalle funzionalità del bracciale e dell'applicazione.

Questo ha concentrato lo sviluppo sulla facilità di utilizzo dell'applicazione e su una grafica intuitiva e senza architetture complesse [29].
L'applicazione utilizza il servizio Bluetooth 2.0, anche questo disponibile nella maggior parte degli smartphone. Le funzionalità della connessione, i dati ricevuti e inviati non sono cambiati dalla versione precedente dell'applicazione in quanto il firmware del dispositivo indossabile non è variato [12]. La gestione del Bluetooth all'interno dell'applicazione e il metodo di selezione dispositivo è invece cambiato.

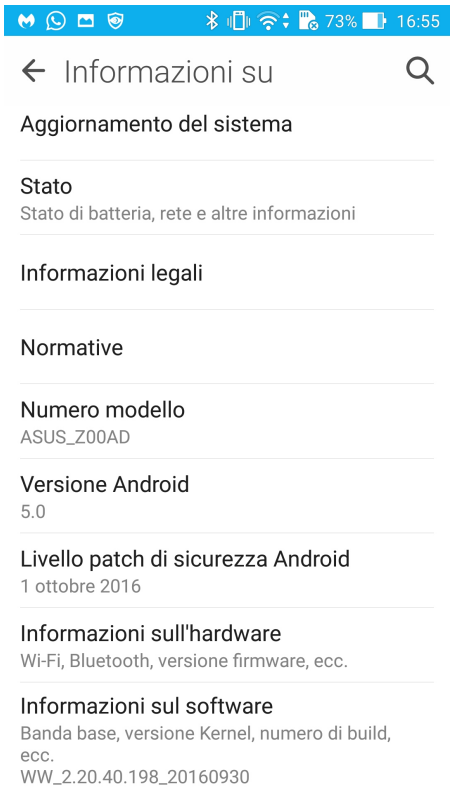


Figura 5.11: Informazioni sul sistema operativo di uno degli smartphone su cui è stata installata l'applicazione

5.2.1 Schermata di caricamento

Figura 5.12: Splash screen per il caricamento dell'applicazione

La schermata di caricamento viene visualizzata all'apertura dell'applicazione e non possiede alcuna corrispondenza ad attività a se stanti come in Ecg Sensor (v. Sez. 5.1.1) ma è semplicemente un tema iniziale per l'attività principale dichiarata nel manifest file (v. Fig. 5.13).

```
59      <activity
60          android:name=".activity.MainActivity"
61          android:theme="@style/SplashTheme"
62          android:screenOrientation="portrait">
63          <intent-filter>
64              <action android:name="android.intent.action.MAIN" />
65              <category android:name="android.intent.category.LAUNCHER" />
66          </intent-filter>
```

Figura 5.13: Splash screen dichiarato nel manifest per l'attività principale

Questo tema è stato dichiarato all'interno del file style.xml nel modo seguente:

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item
    name="android:windowBackground">@color/colorBackground</item>
        <item name="android:textColor">@color/colorText</item>
    </style>

    <style name="SplashTheme" parent="AppTheme">
        <item
    name="android:windowBackground">@drawable/splash_theme</item>
    </style>

</resources>
```

Questa soluzione permette di alleggerire di molto l'applicazione in quanto lo splash screen sarà un semplice layout iniziale. Inoltre, e più importante, lo splash screen verrà visualizzato solamente il tempo necessario per caricare tutte le informazioni all'interno della pagina principale. Non abbiamo, dunque, un timer prefissato ma il tempo sarà dinamico e dipendente dalla velocità di processamento del processore (CPU) dello smartphone. Questo permette di velocizzare l'apertura dell'applicazione, migliorando l'esperienza dell'utente, e di visualizzare la schermata di caricamento per una effettiva motivazione.
Il tema viene poi cambiato programmaticamente al termine del caricamento.

5.2.2 Pagine di introduzione

La prima apertura dell'applicazione è caratterizzata da schermate che verranno visualizzate solo la prima volta. Queste consistono in alcune schermate di introduzione (v. Fig. 5.14) e una schermata nel quale verranno inserite le informazioni dell'utente (v. Sez. 5.2.3).
L'introduzione include quattro fragment. La navigazione è permessa sia attraverso i bottoni posti in fondo alla pagina, sia attraverso lo slide della schermata. In alto a destra è possibile saltare questa introduzione per ritrovarsi direttamente nella schermata delle impostazioni iniziali o nella pagina principale. Infatti, è possibile accedere alla schermata di introduzione anche dall'attività principale attraverso un'icona presente nella barra delle applicazioni (v. Sez. 5.2.4).

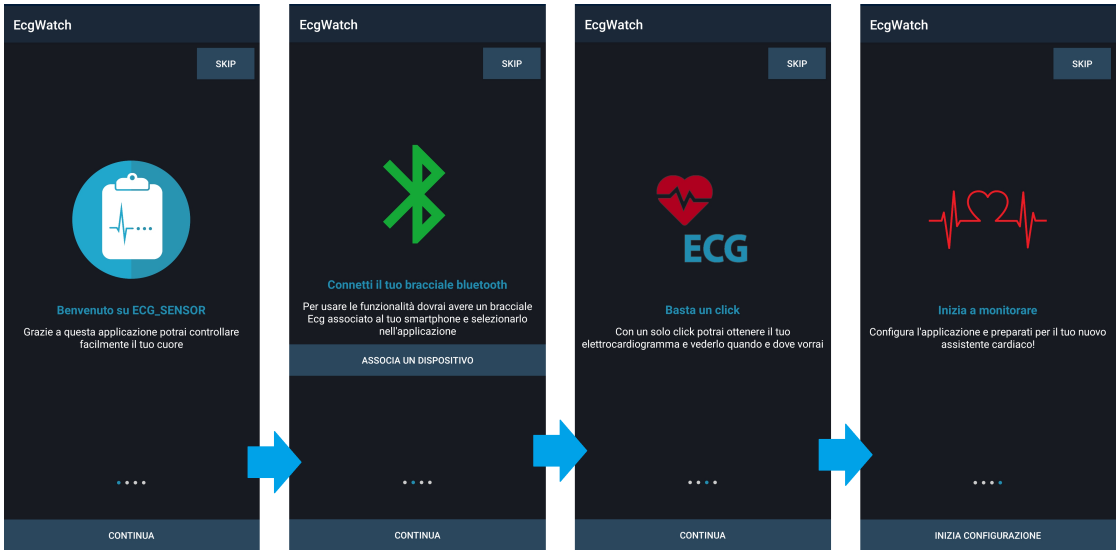


Figura 5.14: Schermate di presentazione dell'applicazione

La seconda schermata, come vediamo in dettaglio nella Fig. 5.15, possiede un pulsante centrale che permette di accedere alle impostazioni del Bluetooth per poter associare il dispositivo indossabile allo smartphone. Questa operazione potrà comunque essere svolta in seguito facendo riferimento ad un dialog, accessibile dal menù principale, per aprire le impostazioni Bluetooth dello smartphone.

Figura 5.15: Dettaglio del secondo fragment

All'interno delle impostazioni, basterà selezionare il nostro bracciale tra i dispositivi Bluetooth rilevati, come in Fig. 5.16. I bracciali avranno un nome come, per esempio, ECG_1 dove il numero risulterà unico per ognuno. Una volta associato, il dispositivo sarà facilmente selezionabile direttamente dall'applicazione attraverso l'apposito dialog (v. Fig. 5.22).

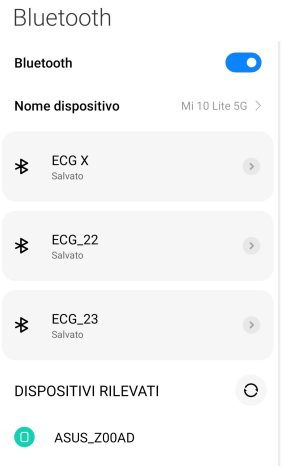


Figura 5.16: Impostazioni Bluetooth dello smartphone Xiaomi Mi 10 lite

5.2.3 Impostazioni iniziali

La schermata per inserire le informazioni dell'utente viene mostrata solo alla prima esecuzione dell'applicazione. All'interno della schermata, visibile in Fig. 5.17, sono presenti tre valori da inserire obbligatoriamente per procedere con l'utilizzo dell'applicazione: Nome, Cognome, Email. Anche la scelta della posizione del bracciale non può essere nulla. Questi valori si rendono necessari in quanto i file verranno salvati utilizzando il nome e cognome dell'utente. L'indirizzo email del dottore viene utilizzata nel caso si volesse condividere il proprio elettrocardiogramma. In tal caso, se l'indirizzo del medico è stata inserito, l'applicazione provvederà ad aggiungere il destinatario della email in automatico.

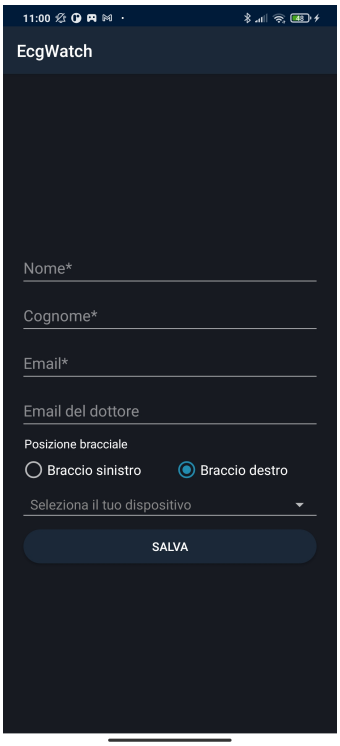


Figura 5.17: Schermata per inserire le prime informazioni dell'utente all'interno dell'applicazione

La posizione del bracciale, invece, influenza la misurazione dell'elettrocardiogramma: invertendo i poli, il segnale verrà invertito rispetto l'asse y (voltaggio). Questo rende necessario una modifica al segnale prima del salvataggio, come illustrato nel Cap. 7. Se i campi risultano vuoti, verrà segnalato un errore e non verrà aperta nessuna schermata, altrimenti si continuerà mostrando la pagina principale.
Tutte le informazioni inserite saranno modificabili in seguito tramite le impostazioni (v. Sez. 5.2.5).

5.2.4 Pagina principale

La pagina principale di EcgWatch (v. Fig. 5.18) è composta da tre CardView interagibili che permettono di accedere alle tre funzionalità principali dell'applicazione. La grafica della schermata, come vedremo tra poco, risulta essere dinamica in quanto verrà modificata in base alle variazioni del Bluetooth e della connessione con il dispositivo. Attraverso la toolbar possiamo accedere alla schermata di introduzione presentata in precedenza (v. Sez. 5.2.2) e alle impostazioni (v. Sez. 5.2.5).

Figura 5.18: Pagina principale dell'applicazione EcgWatch

Card Bluetooth Il primo bottone della pagina principale varia in base allo stato del Bluetooth (v. Fig. 5.19):

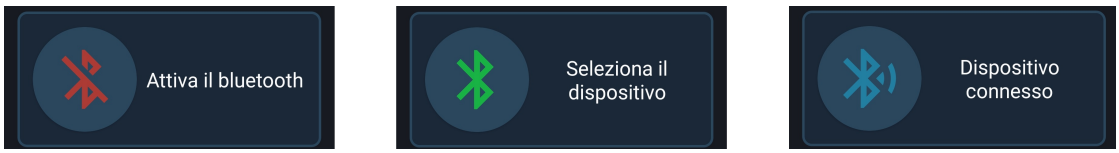


Figura 5.19: Stato del Bluetooth mostrato dalla prima CardView

1. icona rossa: il Bluetooth dello smartphone è disattivato. Premendo sulla CardView verrà richiesto all'utente se si vuole consentire all'applicazione di attivare il Bluetooth (v. Fig. 5.20);

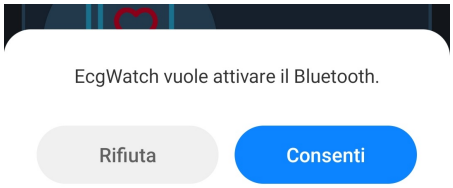


Figura 5.20: Richiesta di attivazione Bluetooth

2. icona verde: il Bluetooth dello smartphone è attivo ma l'applicazione non ha ancora effettuato la connessione con il dispositivo. In questo caso, il click sul bottone aprirà un custom dialog sopra la main activity.
   Il dialog mostra un avviso nel caso non sia stato rilevato nessun dispositivo ECG associato, come mostrato in Fig. 5.21. Sarà possibile aprire facilmente le impostazioni Bluetooth dello smartphone attraverso il bottone "Associa un dispositivo".

Figura 5.21: Dialog per la selezione del dispositivo quando nessun bracciale è associato

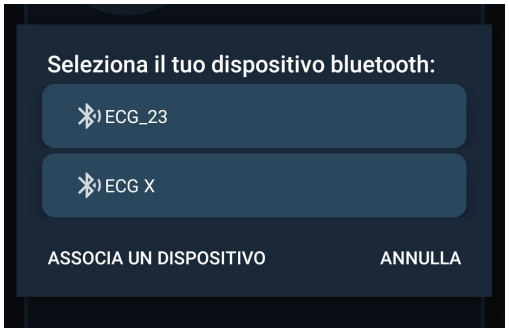


Figura 5.22: Dialog per la selezione del dispositivo con una lista di dispositivi associati. Si noti che vengono visualizzati solo i dispositivi "ECG"

Nel caso in cui siano invece presenti dei dispositivi associati, questi verranno mostrati all'interno di una recycler view all'interno del dialog, come in Fig. 5.22. Selezionando uno dei device disponibili, verrà visualizzata una barra di caricamento circolare che indicherà la ricerca del dispositivo e la creazione

del socket di comunicazione. Con l'avvenuta connessione, il dispositivo verrà salvato tra i dati locali dell'applicazione.
Nel caso non si riuscisse a creare la connessione, perchè, ad esempio, il bracciale è spento o non è nelle vicinanze, verrà mostrato un toast con un avviso di connessione mancata.

3. icona blu: l'applicazione ha aperto un canale di comunicazione (socket) con il bracciale ed è pronto per chiedere e ricevere i dati.
   Premendo sulla card, apparirà un dialog, in Fig. 5.23 che chiederà se si vuole chiudere la comunicazione con il dispositivo attualmente associato.

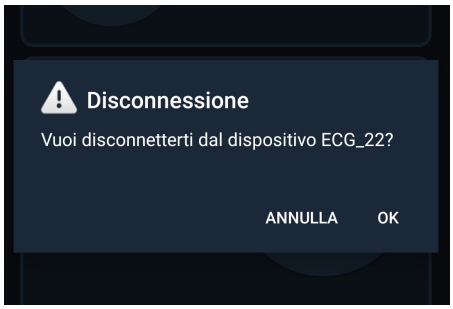


Figura 5.23: Richiesta di disconnessione dal bracciale

Tutti i cambiamenti del Bluetooth sono rilevati automaticamente dall'applicazione attraverso un BroadcastReceiver di cui parleremo approfonditamente nella parte di back-end (v. Sez. 5.3). Oltre alla CardView dedicata al Bluetooth, la scritta posizionata in basso rileverà tutti i cambiamenti di stato dello stesso, indicando anche i casi di disconnessione e connessione.
Infine, se un dispositivo è stato collegato in un avvio precedente, l'applicazione, alla sua apertura, cercherà di connettersi automaticamente al bracciale salvato in memoria.

Card acquisizione dell'elettrocardiogramma Facendo riferimento alla Fig. 5.18, il pulsante centrale permette di iniziare il processo di acquisizione dei dati dal bracciale, aprendo l'activity dedicata a questo processo (v. Sez. 5.2.6).
Se il dispositivo non è connesso, il pulsante cercherà di aprire una comunicazione con il dispositivo salvato in memoria (se presente) altrimenti mostrerà un avviso tramite toast. In caso di socket aperto e funzionante, il processo continuerà nell'attività successiva.

Card archivio L'ultima card della pagina principale (v. Fig. 5.18) apre banalmente la sezione archivio dell'applicazione, ove è visualizzabile la lista di misurazioni precedentemente salvate (v. Sez. 5.2.7).

5.2.5 Impostazioni

Le impostazioni dell'applicazione EcgWatch comprendono (v. Fig. 5.24):

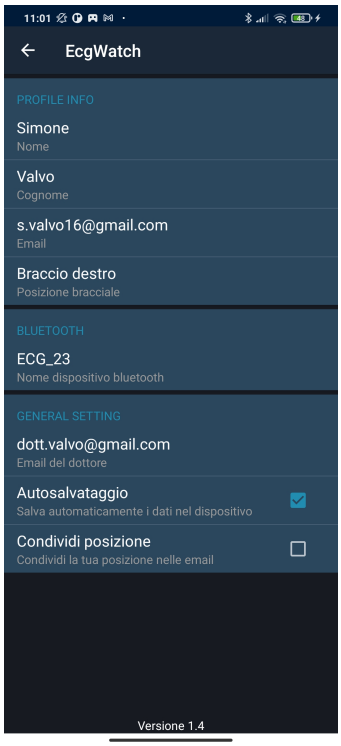


Figura 5.24: Schermata delle impostazioni di EcgWatch

1. nome dell'utente: utilizzato per salvare il file della misurazione e per identificare la persona che ha effettuato la misurazione;
2. cognome dell'utente: anche questo utilizzato per il salvataggio del file;
3. email dell'utente: usato nelle funzioni di condivisione;
4. posizione del bracciale: utilizzato per l'elaborazione del segnale prima del salvataggio;
5. nome del dispositivo associato: identifica l'ultimo dispositivo con cui l'applicazione ha effettuato la connessione e viene aggiornato automaticamente. Utilizzato per la ricerca del dispositivo e creazione del socket di comunicazione;
6. email del dottore: utilizzata dalle funzioni di condivisione tramite email;
7. autosalvataggio: permette di scegliere se salvare un file in memoria o meno;

8. condivisione della posizione: consente l'invio della posizione geografica dell'utente attraverso le funzioni di condivisione presenti nell'applicazione. Questo vuole essere un metodo di prevenzione nel caso in cui la persona presenti valori instabili e possa quindi sentirsi male. Risulterà particolarmente utile se, in futuro, verrà aggiunto un accelerometro al bracciale.

In fondo alla schermata è possibile leggere la versione di EcgWatch installata nel proprio smartphone o tablet.
Tutti questi valori vengono salvati in memoria utilizzando la classe LocalData che sfrutta il tipo di oggetto SharedPreferences. Questa contiene delle costanti che vengono utilizzate come chiavi per l'accesso a dei dati. Utilizzandole è dunque possibile accedere, in scrittura o in lettura, a dei determinati valori. Questi verranno salvati in maniera permanente in memoria fino a quando l'applicazione non viene disinstallata. Le chiavi presenti nella classe e due esempi di accesso, una in lettura e una in scrittura, sono descritte nel codice sottostante:

```
public class LocalData {
    public static final String NAME = "name_tv";
    public static final String SURNAME = "surname_tv";
    public static final String EMAIL = "email_tv";
    public static final String DEVICE = "device_tv";
    public static final String DOCTOR = "doctor_tv";
    public static final String AUTOSAVE = "autosave_tv";
    public static final String POSITION = "position_tv";
    public static final String FIRST_ACCESS = "first_time_tv";
    public static final String LOCATION = "location_tv";
    public static final String ALGO = "algorithm";
    private final static String PREF_KEY = "key";
    public static void setPrefString(Context context, String key,
   String value) {
        SharedPreferences settings =
   context.getSharedPreferences(PREF_KEY, 0);
        SharedPreferences.Editor editor = settings.edit();
        editor.putString(key, value);
        editor.apply();
    }
    public static String getSharedPreferenceString(Context context,
   String key, String defValue) {
        SharedPreferences settings =
   context.getSharedPreferences(PREF_KEY, 0);
        return settings.getString(key, defValue);
    }
```

5.2.6 Schermata di acquisizione

La schermata di acquisizione (v. Fig. 5.25) ha, in realtà, un layout molto semplice. Essa mostra, inizialmente, delle indicazioni per permettere all'utente di poggiare il dito sul bracciale e iniziare la misurazione. Quando il dispositivo avvierà la misurazione, l'applicazione mostrerà una barra di avanzamento che permetterà all'utente di tenere sotto controllo la progressione del processo. Infine, verrà mostrata una schermata di caricamento mentre i dati vengono elaborati dall'applicazione. Durante tutto il processo, sarà disponibile una animazione al centro della schermata. Questa animazione è stata creata come una sequenza di immagini (frame) che si susseguono uno dopo l'altro. I dettagli riguardo la comunicazione e l'elaborazione dei dati vengono rimandati alla Sez. 5.3.

Figura 5.25: Schermata di ricezione dei dati da parte del bracciale

5.2.7 Archivio

La sezione "Archivio" dell'applicazione mostra tutti i file salvati all'interno della memoria dello smartphone/tablet. I file vengono mostrati all'interno di una RecyclerView (v. Fig. 5.26) suddiviso in categorie che corrispondono al mese e all'anno di acquisizione. Se non fosse possibile ricavare questi dati dal file, è presente la categoria "Altro" in cui verranno mostrati tutti i file non categorizzati.
La lista mostrerà, in ordine, i file più recenti per un determinato utente proseguendo fino ai file più "vecchi". I file per un determinato utente vengono visualizzati consecutivamente, indipendentemente dalla data. All'interno dell'archivio sono disponibili varie opzioni, come possiamo vedere nella Fig. 5.26. Queste cercano di

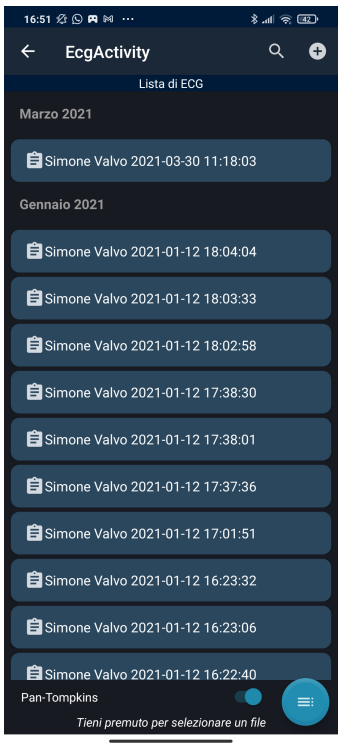


Figura 5.26: Schermata della sezione "Archivio" dell'applicazione EcgWatch

migliorare l'esperienza utente inserendo delle funzionalità di ricerca, filtraggio e gestione multipla dei file. Nel dettaglio, le operazioni sono:

1. importa un file esterno: l'icona in alto a destra con il simbolo "+" nella toolbar aggiunge alla lista un file preso da una qualsiasi cartella della memoria esterna o interna. Nel dettaglio, copia il file all'interno della cartella dedicata all'applicazione per l'immagazzinamento di questi file;

2. ricerca: alla sinistra del pulsante precedente troviamo la lente di ingrandimento. Questa è un'icona "espandibile" (v. Fig. 5.27), ovvero cliccando su di essa si espande lungo la toolbar permettendo l'inserimento di un testo.

Figura 5.27: Apertura della barra di ricerca

La ricerca dei caratteri inseriti avviene automaticamente anche durante la scrittura e permette di mostrare solo gli elementi della lista che contengono, anche parzialmente, la stringa inserita, come mostrato in Fig. 5.28;

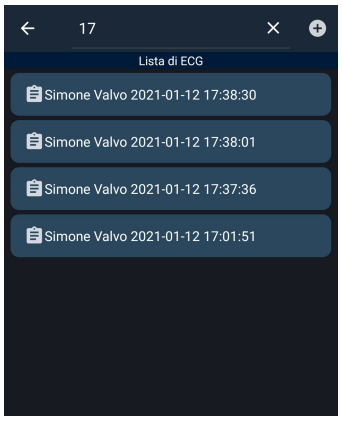


Figura 5.28: Esempio di una ricerca

3. filtri di ricerca: nell'angolo in basso a destra, è presente un floating button che permette di filtrare i file della lista in maniera più specifica rispetto alla ricerca normale. Premendo il bottone verrà mostrato un dialog come nella Fig. 5.29.

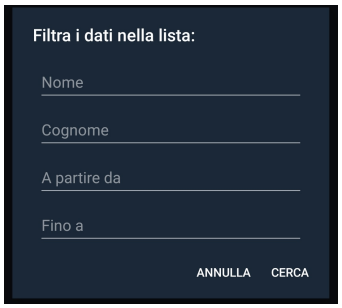


Figura 5.29: Dialog con i filtri di ricerca disponibili

Questo dialog permette di impostare il nome e il cognome dell'utente di cui si vogliono visualizzare i file e l'intervallo di tempo ammesso attraverso un calendario, come raffigurato nella Fig. 5.30.

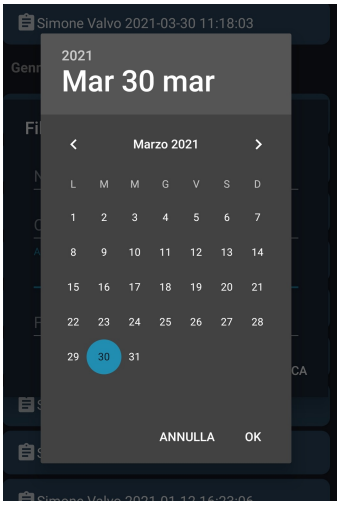


Figura 5.30: Calendario per la scelta delle date da filtrare

4. cambio algoritmo di rilevazione picchi R: lo switch posto in basso permette di cambiare l'algoritmo di rilevazione dei picchi R nell'elettrocardiogramma. Questo è stato inserito per fare dei test su diversi segnali misurati. Parleremo approfonditamente di questi algoritmi nel Cap. 7;

5. selezione multipla: tenendo premuto un elemento della lista, si aprirà un menù contestuale in cui sarà possibile selezionare più file e compiere su questi delle azioni definite nella barra dell'applicazione (v. Fig. 5.31).

Action Mode Selezionando uno o più elementi della lista, verrà attivata la cosiddetta "Action mode". Questa mostra un menù contestuale (v. Fig. 5.31) che consente tre nuove funzionalità:

1. selezionare tutti gli elementi della lista;

2. cancellare i file selezionati sia dalla lista che dalla memoria;

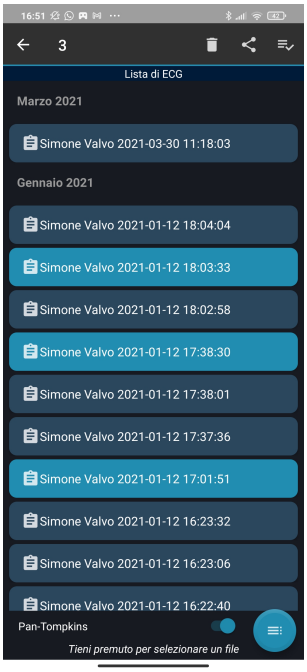


Figura 5.31: Archivio con degli elementi selezionati e il menù contestuale

3. condividere i file selezionati attraverso una applicazione di comunicazione installata e scelta attraverso un dialog di condivisione (v. Fig. 5.32). All'interno del corpo del messaggio verrà inserito l'indirizzo corrente se è stata impostata la condivisione della posizione. Se viene scelto di condividere tramite email, verrà impostata l'email del dottore in automatico come destinatario (se è stato precedentemente inserito) (v. Sez. 5.2.5).

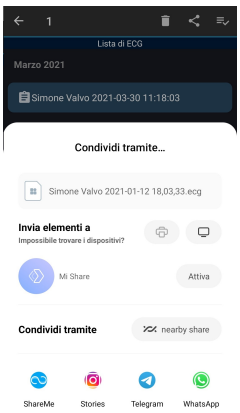


Figura 5.32: Dialog per la scelta dell'applicazione per la condivisione

5.2.8 Visualizzazione elettrocardiogramma

Il segnale viene visualizzato all'interno di un MPAndroid Chart (v. Fig. 5.33), descritto nel Cap. 4.3.1. Il grafico rappresenta una griglia che tende ad essere il più possibile simile alla carta millimetrata usata in ambito medico.
Ogni quadrato piccolo equivale a 0.1mV sull'asse delle ordinate, che rappresentano il voltaggio (in mV), e a 40ms sull'asse delle ascisse, che rappresentano il tempo (in secondi), perfettamente in linea con i valori ufficialmente utilizzati [30]. Il grafico mostra di default solo cinque secondi di acquisizione all'interno della schermata con la possibilità di scorrere a destra e sinistra per visualizzare le altre parti del segnale. Questo viene fatto per una migliore visibilità. Inoltre, è possibile fare uno zoom del grafico nel caso si voglia osservare un dettaglio specifico.
I picchi R sono identificati da una barra verticale posta alla base del grafico, in modo tale da non modificare la visibilità del segnale e garantirne una corretta lettura. Attraverso la distanza R-R riusciamo a calcolare la frequenza cardiaca, che viene mostrata nella parte alta del layout.

Figura 5.33: Schermata di visualizzazione dell'elettrocardiogramma

Nel caso venga rilevata una possibile anomalia di tipo "fibrillazione atriale" (v. Sez. 2.1.4) durante la misurazione, l'applicazione mostrerà un dialog di avvertimento (v. Fig. 5.34) insieme ad un segnale sonoro. Questo dialog permetterà all'utente, qualora lo desideri, di condividere la misurazione direttamente con il suo medico, oppure di visualizzare semplicemente il suo elettrocardiogramma. Il dialog viene mostrato solo la prima volta che il segnale viene acquisito e visualizzato, non verrà più visualizzato aprendolo dall'archivio.

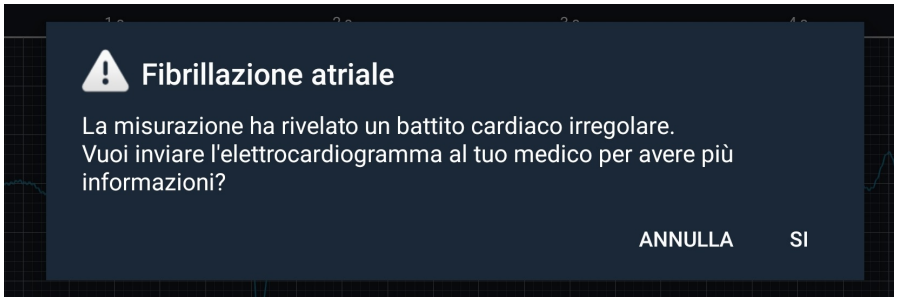


Figura 5.34: Dialog di avvertimento per la rilevazione di una possibile fibrillazione atriale

La fibrillazione atriale viene indicata nella schermata del grafico, come mostrato nella Fig. 5.35. Una scritta e delle icone sulla parte inferiore dello schermo indicheranno se nel grafico è stata rilevata la possibilità di fibrillazione atriale. In questo modo, un utente può sempre vedere se in una determinata misurazione, anche passata, era presente una possibile anomalia.

Figura 5.35: Visualizzazione di un elettrocardiogramma che presenta fibrillazione atriale

Azioni possibili Il menù presente nella toolbar (v. Fig. 5.36) permette di eseguire le seguenti azioni (partendo da destra):

Figura 5.36: Menù della toolbar presente nella activity di visualizzazione dell'elettrocardiogramma. A: apre la schermata di informazione; B: elimina il file; C: condivide il file; D: salva il file; E: crea PDF e condivide

1. Fig. 5.36A: apre la finestra di informazione, una pagina in cui sono presenti alcune informazioni generali riguardo l'elettrocardiogramma (v. Sez. 5.2.9);
2. Fig. 5.36B: cancella il file dalla memoria dello smartphone;
3. Fig. 5.36C: condivide il file dell'elettrocardiogramma tramite un'applicazione presente sul dispositivo;
4. Fig. 5.36D: salva il file in memoria. Viene visualizzato solo se l'autosalvataggio non è stato abilitato nelle impostazioni (v. Sez. 5.2.5) e quindi solo dopo aver eseguito una acquisizione (non per i file dell'archivio);
5. Fig. 5.36E: crea e condivide un file PDF con le informazioni dell'utente e l'elettrocardiogramma, come visualizzato in Fig. 5.37. Il PDF viene creato con pagine di dimensioni pari ad un foglio A4, in modo da semplificarne la stampa. La griglia viene rappresentata cercando di mantenere le proporzioni richieste dalla carta millimetrata specifica per l'elettrocardiogramma. Verranno inoltre riportate alcune informazioni quali: nome, cognome, data, durata, posizione (facoltativa), frequenza cardiaca e se è stata rilevata qualche anomalia.

Figura 5.37: Dettaglio del file PDF creato mediante l'applicazione EcgWatch

5.2.9 Informazioni generali elettrocardiogramma

La schermata presenta alcuni dettagli che riguardano la lettura dell'elettrocardiogramma, descrivendo le principali onde presenti all'interno del segnale e alcuni valori indicativi per un elettrocardiogramma standard [31]. Nella Fig. 5.38 vediamo una parte delle informazioni generali inserite, bisognerà utilizzare lo scroll verticale nell'applicazione per visualizzare il resto del testo.

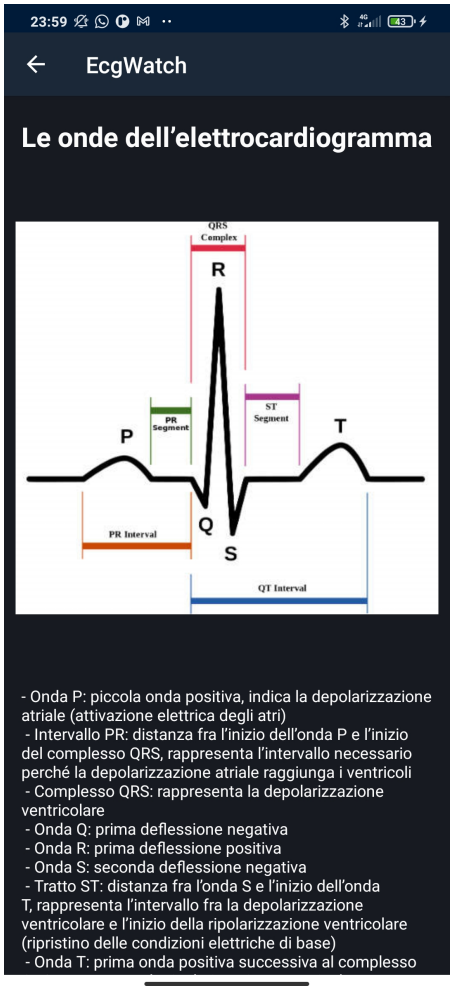


Figura 5.38: Schermata con delle informazioni generali sull'elettrocardiogramma

5.3 EcgWatch seconda versione: Back-end

5.3.1 Bluetooth

La connessione Bluetooth 2.0 viene gestita attraverso una classe singleton con un oggetto di tipo BluetoothSocket e con funzioni sincrone. Una classe si definisce singleton se esiste un'unica istanza di questo oggetto e risulta accessibile in modo globale. Il codice sottostante presenta l'oggetto BluetoothSocket e una funzione di esempio della classe descritta. Il BluetoothSocket è dichiarato in modo statico, ovvero non sarà legato ad una classe specifica. Come funzione di esempio viene presentata la chiusura del socket, definita "synchronized" in modo da assicurare accessi unitari alla funzione. Questa chiude semplicemente la connessione con il bracciale e imposta l'oggetto statico come vuoto.

```
public class SingletonSocket {
    private static BluetoothSocket socket;

    public static synchronized void closeSocket(){
        try {
            if(SingletonSocket.socket!= null) {
                SingletonSocket.socket.close();
                SingletonSocket.socket = null;
            }
        } catch (IOException e) {
            Log.e("Connection", "Could not close the client socket");
        }
    }
}
```

La scelta è ricaduta su questa soluzione in modo da poter accedere al canale di comunicazione con il dispositivo da diverse attività e poter gestire in maniera più ampia la connessione con questo.
Il socket viene creato nella pagina principale quando:

1. si preme sul nome di un dispositivo all'interno del dialog Bluetooth (v. Fig. 5.22);
2. si crea la pagina principale o si accende il Bluetooth (in automatico);
3. si prova ad acquisire una misurazione ma il dispositivo non è ancora stato connesso.

Ovviamente il dispositivo dovrà essere acceso e nell'area di ricezione altrimenti la connessione non andrà a buon fine e il socket non verrà creato.

Le prove di connessione vengono effettuate in un thread secondario. In tal modo si evita di bloccare l'applicazione durante tutto il processo di rilevamento e di connessione con il bracciale. Sarà quindi possibile mostrare delle schermate di caricamento all'utente in attesa [32].
I messaggi inviati dallo smartphone al dispositivo sono:

- di segnalazione;
- di due tipologie;
- memorizzati in due costanti, mostrate nel codice in basso.

```
byte[] READ_START={'R'};
byte[] WRITE_START={'1'};
```

Inviando tramite Bluetooth WRITE_START, il dispositivo avvia il rilevamento dei segnali elettrici sugli elettrodi e salverà questi dati internamente. Nell'applicazione verrà intanto mostrata la schermata di caricamento (v. Fig. 5.25). Una volta terminata l'acquisizione, di durata 10 secondi, l'applicazione provvederà ad inviare il valore READ_START al dispositivo e a mettersi in attesa della lettura del socket. Il dispositivo invierà una serie di pacchetti con all'interno i valori salvati durante la misurazione. Questi verranno ricevuti ed elaborati dall'applicazione stessa, così come accadeva in Ecg Sensor [12]. Durante la lettura, ho inserito dei periodi di controllo del socket durante il quale viene verificato se la lettura è disponibile. L'applicazione eviterà quindi di rimanere ferma in un ciclo di attesa infinito.

Broadcast Receiver Come è stato già detto nella sezione 5.2.4, l'applicazione è in grado di rilevare lo stato del Bluetooth: disabilitato, abilitato, connesso.
Questo è dovuto ad una classe base che riceve e gestisce gli intenti di trasmissione inviati da Context.sendBroadcast(Intent), ovvero la classe BroadcastReceiver. Quest'ultima necessita di essere "registrata" alla creazione della pagina principale e di essere dismessa alla chiusura tramite le apposite funzioni Context.registerReceiver() e Context.unregisterReceiver(). Per registrare il BroadcastReceiver è inoltre necessario creare un IntentFilter con le varie azioni che si intende gestire. Nel codice vediamo la registrazione di mReceiver con i filtri:

- ACTION_STATE_CHANGED: notifica i cambiamenti di stato del Bluetooth;
- ACTION_ACL_CONNECTED: notifica l'avvenuta connessione con il bracciale;
- ACTION_ACL_DISCONNECTED: notifica la disconnessione del dispositivo.

```
    public void setUpIntentFilter() {
        IntentFilter filter = new IntentFilter();
        filter.addAction(BluetoothAdapter.ACTION_STATE_CHANGED);
        filter.addAction(BluetoothDevice.ACTION_ACL_CONNECTED);
        filter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECTED);
        this.registerReceiver(mReceiver, filter);
    }
```

Quando viene istanziata la classe BroadcastReceiver, nel nostro caso la variabile mReceiver, è necessario specificare la funzione che riceve gli intent e studiare i vari casi che sono stati registrati per tale classe. Nel codice sottostante possiamo vedere come, per esempio, viene individuato e gestito il caso di Bluetooth disattivato. In questo caso verrà modificata la grafica della pagina principale, come mostrato in 5.2.4, e verrà chiuso il canale di comunicazione (se attivo).

```
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
 public void onReceive(Context context, Intent intent) {
  String action = intent.getAction();
  switch (Objects.requireNonNull(action)) {
   case BluetoothAdapter.ACTION_STATE_CHANGED:
    final int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
     BluetoothAdapter.ERROR);
     switch (state) {
      case BluetoothAdapter.STATE_OFF:

       ((TextView) findViewById(R.id.bluetoothStateView))
        .setText(getString(R.string.bluetooth_off));

       ((ImageView) findViewById(R.id.bluetooth_image))
        .setImageResource(R.drawable.ic_bluetooth_inactive);

       ((TextView) findViewById(R.id.bluetooth_text))
        .setText(getString(R.string.device_connection_off_text));

       SingletonSocket.closeSocket();
       break;
    }
  }
};
```

5.3.2 Localizzazione

La localizzazione è stata inserita come nuova funzionalità dell'applicazione. Questa vuole essere un sistema di sicurezza nel caso in cui la persona che effettua la misurazione si stia sentendo male. In questo caso, ha senso che la condivisione avvenga con la posizione esatta dell'utente. In particolare, è stata pensata per uno sviluppo futuro in cui si vorrebbe integrare un termometro e un accelerometro per segnalare movimenti bruschi o anomali.
Per l'integrazione della localizzazione, è stato inizialmente implementato un sistema GPS basato sulla sola classe LocationManager che utilizzasse le funzioni già presenti per il calcolo della posizione dell'utente (date come coordinate geografiche). Il sistema però risultava lento e impreciso, fortemente dipendente dalle prestazioni hardware dello smartphone.
Per questo motivo, è stato introdotto il servizio di localizzazione di Google Play, come descritto nella sezione 4.3.1. Se non saranno disponibili i servizi Google, verrà utilizzata la funzione descritta in precedenza per calcolare la posizione.
La funzione scritta di seguito è un esempio semplificato di come ricavare la posizione in un tempo dipendente dal servizio. Questa richiede che sia stato fornito il consenso per l'accesso alla posizione (ACCESS_COARSE_LOCATION e ACCESS_FINE_LOCATION). Questi permessi devono essere entrambi dichiarati all'interno del manifest così da essere già noto il loro utilizzo durante l'installazione dell'applicazione. Inoltre, necessita la presenza dei servizi Google Play. Una volta confermata la presenza di questi fattori, si cercherà di ottenere la posizione mediante il servizio di localizzazione e la sua funzione "getCurrentLocation". Essa verrà legata ad un listener per rilevare il completamento dell'operazione o il suo fallimento.

```
public void getLocation(final Context ctx, LocationProcessing
    locationProcessing) {
  this.locationProcessing = locationProcessing;

  if (checkLocationPermissions(ctx) && checkPlayServices(ctx)) {
    FusedLocationProviderClient locationProvider =
   LocationServices.getFusedLocationProviderClient(ctx);

    Task<Location> locationTask =
   locationProvider.getCurrentLocation(LocationRequest
    .PRIORITY_HIGH_ACCURACY, null);
    locationTask.addOnCompleteListener(new
   OnCompleteListener<Location>() {
    @Override
    public void onComplete(@NonNull Task<Location> task) {
```

```
      setLocationRetreived(task.getResult());
      }
    })
    .addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
      setLocationRetreived(null);
      }
    })
    });
  }
}
```

All'interno dell'applicazione PulseEcg, vengono effettuati ulteriori controlli riguardo le impostazioni dei servizi Google Play e viene visualizzata una schermata di caricamento fino al termine dell'operazione di localizzazione

5.3.3 Gestione file

I file utilizzati dall'applicazione EcgWatch possiedono un'estensione di tipo ".ecg", gestiti già in Ecg Sensor. Questi file sono, in realtà, dei semplici file di testo al cui interno viene scritto l'array di valori ricevuti dal bracciale. Questi valori vengono scritti come stringhe, una soluzione in realtà poco efficiente e pratica, ma che è stata mantenuta per compatibilità con altri programmi che utilizzano il medesimo formato. Personalmente, avrei optato per la scrittura dei valori direttamente come double attraverso un OutputStream, risparmiando spazio in memoria e tempo nella scrittura e nella lettura.
La creazione del file avviene nel momento in cui l'acquisizione dei dati termina. Questo può essere un file permanente salvato in memoria o un file temporaneo salvato nella cache, in dipendenza del valore impostato per l'autosalvataggio. Il file viene scritto attraverso un oggetto di tipo FileWriter.
Durante la lettura risulta necessario convertire le stringhe in double in maniera tale da poter rappresentare in maniera corretta i valori sul grafico MPAndroidChart.

5.3.4 Grafico MPAndroidChart

La gestione del grafico ha impegnato diverso tempo in quanto, in principio, veniva utilizzata una libreria ormai obsoleta, ovvero GraphView. La selezione del nuovo grafico ha portato al cambiamento degli oggetti rappresentati. Infatti, GraphView rappresentava i dati attraverso la classe DataPoint, mentre MPAndroidChart utilizza degli oggetti chiamati Entry. Purtroppo, anche diverse funzioni legate all'elaborazione del segnale sfruttavano i DataPoint per il loro calcolo. Il reworking ha quindi previsto una nuova descrizione per le funzioni che filtrano il segnale, che rilevano i picchi R e il valore di frequenza cardiaca (HR).
Inoltre, è stata aggiunta la possibilità di creare un PDF direttamente dall'applicazione. Questa operazione sfrutta la funzione nativa della libreria per ottenere il "disegno" (bitmap) del grafico (getChartBitmap()) ma questo necessita di un pre-processamento prima di essere introdotto all'interno del file in maniera corretta. Il codice presenta un esempio di creazione di un file PDF con un nome e un grafico di tipo LineChart, presente nella libreria MPAndroidChart. Per prima cosa viene creato un oggetto PDFDocument e si inserisce una pagina vuota di dimensioni pari a 500x500 pixel. Di questa pagina si ricava il Canvas, oggetto utilizzato per modificare la pagina, e sceglieremo la posizione e lo stile della scrittura. Scriveremo quindi il nome e, successivamente, verrà disegnato il grafico, trasformato in un oggetto Bitmap mediante la funzione getChartBitmap.

```
private void createPdf(LineChart lineChart, String name, LineChart
    lineChart){
     PdfDocument document = new PdfDocument();
     PdfDocument.PageInfo pageInfo = new PdfDocument.PageInfo
     .Builder(X, Y, 1).create();
     PdfDocument.Page page = document.startPage(pageInfo);
     Canvas canvas = page.getCanvas();
     Paint paint = new Paint();
     paint.setTextAlign(Paint.Align.LEFT);
     paint.setTextSize(80);
     paint.setTypeface(Typeface.create(Typeface.DEFAULT, Typeface.BOLD));
     int xPos = PDF_PAGE_X / 6;
     int yPos = 120;
     canvas.drawText(ns, xPos, yPos, paint); //line with name
     yPos += paint.descent() - paint.ascent();
     Bitmap graph = lineChart.getChartBitmap();
     canvas.drawBitmap(graph, 100, yPos, new
    Paint(Paint.FILTER_BITMAP_FLAG));
     document.finishPage(page);
}
```

Capitolo 6

Applicazione PulseEcg

L'applicazione PulseEcg è stata sviluppata per poter interfacciarsi con il bracciale di seconda generazione (v. Sez. 3.2). Questa riprende molti aspetti caratteristici dell'applicazione precedente ma ne rinnova la grafica, rendendola più "leggera" e piacevole. Vengono infatti sostituite molte delle attività con dei semplici fragment che permettono una gestione migliore dell'applicazione e di risparmio delle risorse. Il tema è stato rivisto e cambiato con uno stile chiaro. Gli oggetti grafici presenti sono stati sostituiti con la libreria di Material Design. L'elettrocardiogramma non sarà più elaborato nella modalità precedente in quanto l'hardware misurerà i dati in maniera completamente diversa. Le funzionalità aggiuntive riguardano, invece, la visualizzazione della batteria del bracciale, della fotopletismografia, della saturazione di ossigeno nel sangue e delle pressioni sistolica e diastolica. La connessione attraverso il bracciale avverrà attraverso Bluetooth Low Energy che non sarà più gestito da una classe singleton ma avrà una struttura più innovativa e efficiente, ovvero un servizio collegato all'attività principale.
L'inserimento di una rete neurale porta una grande fonte di innovazione. La possibilità di integrare dei modelli già addestrati all'interno di un'applicazione Android, consente di avere una vasta gamma di algoritmi complessi da poter utilizzare in ogni momento mediante lo smartphone. La possibilità di poter ricavare i valori di pressione arteriosa in modo così semplice e veloce, posando il dito su un bracciale e guardando una applicazione, sembra già un grande passo avanti. Questa operazione viene fatta attraverso la libreria mobile di Tensorflow, che è in continuo aggiornamento. Anche la saturazione dell'ossigeno nel sangue risulta una caratteristica molto importante, soprattutto con l'avvento di malattie che attaccanno il sistema respiratorio. Date le alte richieste computazionali, PulseEcg è pensata per essere utilizzata in smartphone più moderni rispetto all'applicazione EcgWatch. In questo capitolo parleremo inizialmente delle innovazioni grafiche inserite e, successivamente, delle nuove funzionalità presenti all'interno del codice.

6.1 Front-End

Come precedentemente annunciato, il tema dell'applicazione è diverso rispetto a quello utilizzato in EcgWatch (v. Cap. 5.2). L'applicazione presenta infatti un tema chiaro, costituito principalmente dal bianco e dai colori del logo ma presenta anche altri colori come possiamo vedere nella palette in Fig. 6.1.

```
<color name="colorAccent">#218db1</color>
<color name="colorPeak">#F44336</color>
<color name="colorPeakVariant">#D32F2F</color>

<color name="colorCardRed">#F8E9E7</color>
<color name="colorCardBorderRed">#F4938D</color>
<color name="colorCardBlue">#E7EAF9</color>
<color name="colorCardBorderBlue">#22A6CB</color>
<color name="colorCardGreen">#DFF6ED</color>
<color name="colorCardBorderGreen">#30DD9A</color>
<color name="colorCardYellow">#FFFDE7</color>
<color name="colorCardBorderYellow">#F57F17</color>
<color name="colorCardGray">#66848C</color>
<color name="colorCardBorderGray">#66848C</color>
```

```
<color name="colorPrimary">#22A6CB</color>
<color name="colorPrimaryVariant">#2994B2</color>
<color name="colorOnPrimary">#FFFFFF</color>
<color name="colorSecondary">#84DAF5</color>
<color name="colorSecondaryVariant">#35C1E6</color>
<color name="colorOnSecondary">#000000</color>
<color name="colorError">#B00020</color>
<color name="colorOnError">#FFFFFF</color>
<color name="colorSurface">#FFFFFF</color>
<color name="colorOnSurface">#212121</color>
<color name="colorBackground">#FAFAFA</color>
<color name="colorOnBackground">#000000</color>
```

Figura 6.1: Palette di colori dell'applicazione PulseEcg

Molte schermate rimangono simili all'applicazione di cui abbiamo parlato in precedenza, come possiamo già vedere dalla schermata di caricamento iniziale in Fig. 6.2. Anche il logo rimane lo stesso di EcgWatch. La grafica cerca di rendere intuitive le varie funzionalità dell'applicazione cercando di non lasciare nulla senza alcuna spiegazione. Le icone utilizzate sono quelle standard così come il loro posizionamento, facilitando la navigazione da parte dell'utente. Le schermate hanno un orientamento fisso che è verticale per tutte le schermate tranne quelle che mostrano i dettagli (v. Sez. 6.1.5) poichè queste presentano l'elettrocardiogramma e la fotopletismografia.

Figura 6.2: Schermata di caricamento iniziale dell'applicazione PulseEcg

6.1.1 Pagina principale

La pagina principale ha il compito di gestire la connessione con il bracciale e consente la navigazione all'interno delle varie schermate del PulseEcg. La sua schermata, visualizzata in Fig. 6.3, richiama l'organizzazione di EcgWatch ma utilizza delle componenti presenti nella libreria di MaterialComponents, ad esempio le MaterialCardView. Oltre questo dettaglio, possiamo notare che è stato inserito il valore di carica della batteria, presente solo quando il dispositivo è connesso all'applicazione insieme al nome del bracciale. Uno dei valori inviati dal Bluetooth è, infatti, il valore di carica della batteria, sia come voltaggio che come percentuale.

Figura 6.3: Pagina principale dell'applicazione PulseEcg

L'icona all'interno della card, cambierà in base alla carica della batteria del dispositivo (v. Fig. 6.4):

Figura 6.4: Icone che rappresentano lo stato della batteria

1. icona verde se la carica della batteria è superiore al 50%;
2. icona gialla se la carica della batteria è compresa tra il 50% e il 20%;
3. icona rossa se la carica della batteria è inferiore al 20%;

Le azioni effettuate dalle card rimarranno pressocchè invariate rispetto all'applicazione precedente ad esclusione delle animazioni di transizione tra le varie schermate.

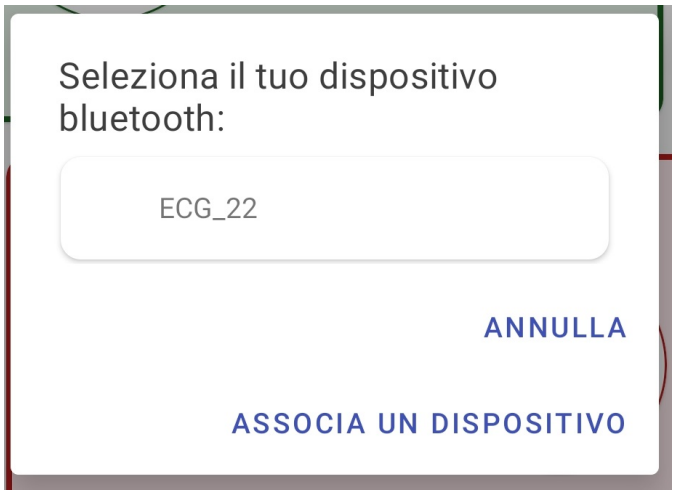


Figura 6.5: Dialog per la selezione del dispositivo Bluetooth

Anche in questo caso provvederemo alla connessione con il dispositivo attraverso un dialog (v. Fig. 6.5). La differenza avverrà nelle funzioni interne all'applicazione in quanto non si tratterà più di una connessione Bluetooth 2.0 ma Bluetooth Low Energy (Bluetooth 4.0).

6.1.2 Impostazioni

Anche le impostazioni, mostrate nella Fig. 6.6, non mostrano particolari cambiamenti: l'editing viene fatto premendo sull'icona di modifica in alto a destra che permette di cambiare i valori all'interno della stessa schermata, mentre prima bastava cliccare all'interno dell'elemento per poterlo modificare attraverso un dialog. Questo cambiamento rende più immediata e leggera la modifica delle informazioni senza la necessità di aprire ulteriori schermate. Inoltre, è presente una configurazione aggiuntiva per indicare il tempo di misurazione. Attualmente, l'applicazione e il dispositivo gestiscono una misurazione pari a 10 secondi ma si pensa di sviluppare, in futuro, la compatibilità con 30 secondi e un 1 minuto di acquisizione. Anche qui sono cambiati gli elementi con la libreria di MaterialComponent.

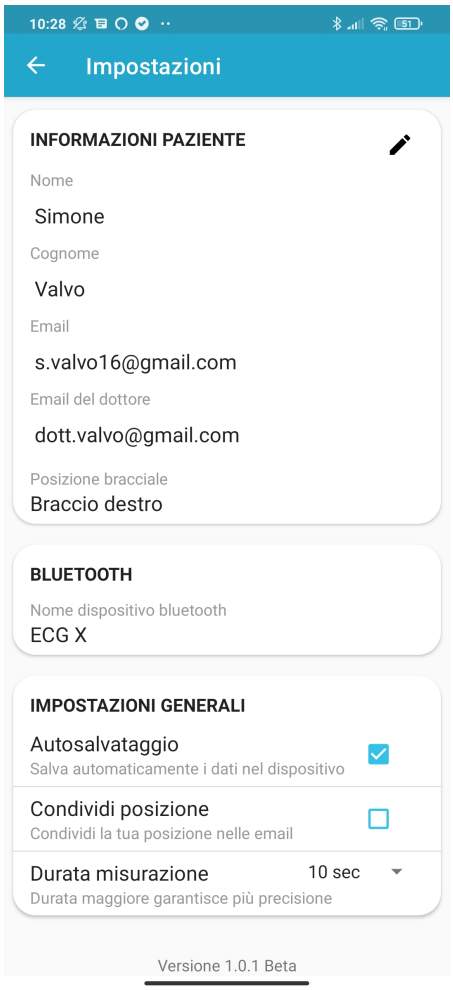


Figura 6.6: Pagina delle impostazioni dell'applicazione PulseEcg

6.1.3 Schermata di acquisizione

L'animazione utilizzata durante l'acquisizione è stata cambiata in modo da renderla più leggera e fluida. Questa consiste in una semplice immagine con la transizione di una View vuota sopra di essa. Ciò è stato fatto perchè l'animazione precedente occupava parte della memoria RAM. Anche se questo non mostrava particolari problemi, è stato deciso di optare per una soluzione diversa e meno costosa in termini di costo computazionale.

Figura 6.7: Pagina visualizzata durante l'acquisizione dal bracciale

6.1.4 Attività di riepilogo della misurazione

Dato il maggior numero di dati fornito dal dispositivo, si è visto necessario inserire una schermata di visualizzazione che presenti ogni elemento misurato. Questa schermata viene aperta subito dopo l'acquisizione oppure selezionando un file dall'archivio. Nel primo caso, il segnale viene elaborato sia per il salvataggio del file che per la visualizzazione e dovranno essere calcolati i valori di saturazione, di frequenza cardiaca e di pressione; nel secondo caso, il file presente in memoria conterrà le informazioni sopra citate e si occuperà solo del filtraggio per la visualizzazione. Se l'autosalvataggio non risulta attivo, non viene creato un file temporaneo (come accadeva in EcgWatch) ma viene gestito tutto programmaticamente all'interno dell'applicazione.

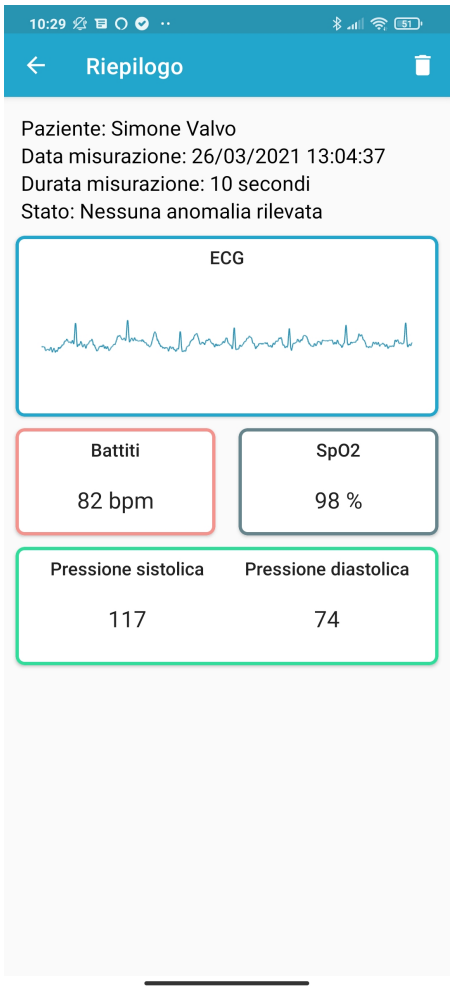


Figura 6.8: Schermata con il riepilogo delle informazioni riguardo la misurazione

La schermata di riepilogo (v. Fig. 6.8) include:

1. i dettagli informativi per quanto riguarda il paziente, la data e la misurazione;
2. un grafico parziale dell'elettrocardiogramma che mostra 6 secondi di acquisizione senza griglia;
3. il valore dei battiti cardiaci;
4. il valore della saturazione di ossigeno nel sangue;
5. i valori di pressione sistolica e diastolica.

L'utente potrà così vedere in maniera semplice, chiara e veloce, tutte le informazioni ricavate dalla misurazione. Inoltre, l'applicazione segnalerà dei valori che non rientrano nei rispettivi intervalli di sicurezza, come mostrato in Fig. 6.9. In particolare, gli avvisi riguarderanno:

- fibrillazione atriale;
- battiti per minuto (bpm) sotto i 40 o sopra i 150;
- saturazione al di sotto del 90%;
- pressione sistolica con valore superiore a 160mmHg o inferiore a 100mmHg;
- pressione diastolica con valore superiore a 90mmHg o inferiore a 50mmHg.

Premendo sulle varie schede, sarà possibile visualizzare la schermata dei dettagli.
Se si tenta di aprire un file con un'estensione o con una formattazione non supportata, verrà mostrato un dialog con una segnalazione, come indicato in Fig. 6.10.

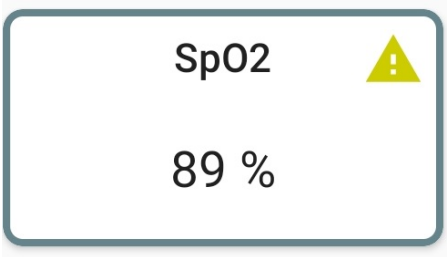


Figura 6.9: Esempio di avviso nella schermata di riepilogo per un valore di SpO2 inferiore al 90%

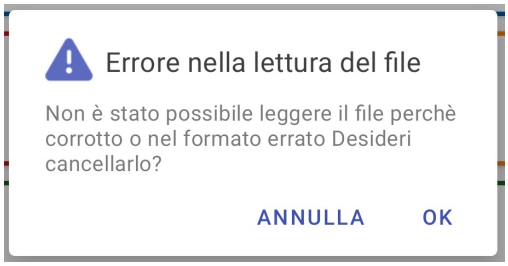


Figura 6.10: Dialog di avvertimento per un tipo di file sconosciuto

6.1.5 Finestre dei dettagli

Premendo su un qualsiasi elemento nell'attività di riepilogo, si aprirà una schermata collegata alla sezione selezionata. In queste nuove schede, sarà possibile visualizzare in modo dettagliato l'elettrocardiogramma, prima visualizzato solo come anteprima, e la fotopletismografia, che non viene invece presentata all'interno del riepilogo. I valori di pressione saranno visualizzati con un ingrandimento in una scheda dedicata e con un avviso nel caso di valori troppo alti o bassi.

Elettrocardiogramma Il dettaglio dell'elettrocardiogramma (v. Fig. 6.11) è molto simile all'attività di visualizzazione dell'elettrocardiogramma in EcgWatch (v. Sez. 5.2.8). Le uniche differenze rilevabili sono la gestione della schermata mediante un fragment e la possibilità di cambiare schermata attraverso un semplice scroll o utilizzando le schede presenti nella parte alta della schermata. La navigazione tra i dettagli dei vari dati è molto semplice e intuitiva. Anche qui vengono mostrati solo intervalli di cinque secondi per una migliore visibilità del segnale ed è sempre possibile fare uno zoom.

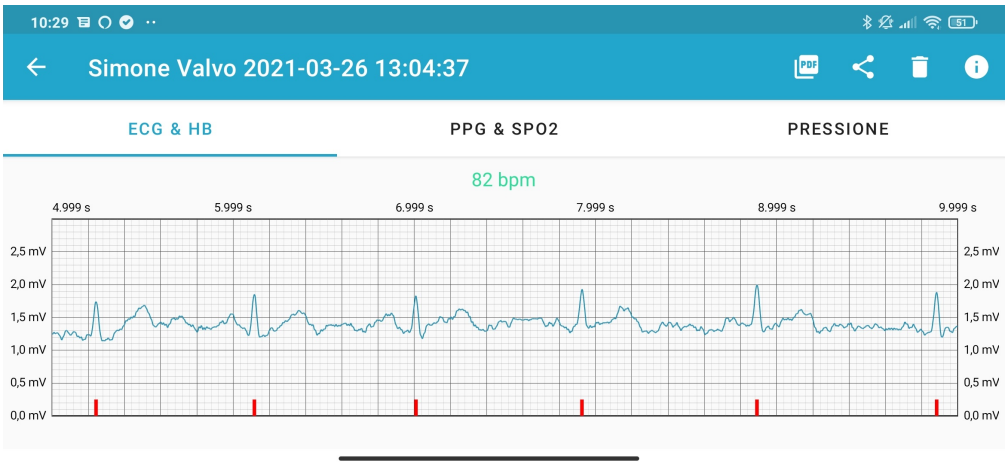


Figura 6.11: Dettaglio dell'elettrocardiogramma con il numero di battiti per minuto

Fotopletismografia La fotopletismografia è il nuovo segnale ricavato attraverso il sensore rosso e infrarosso presente all'interno del dispositivo. Questo grafico, diversamente dall'elettrocardiogramma, non necessita di una particolare carta millimetrata e non presenta valori significativi lungo l'asse delle ordinate. Dai segnali ottenuti, è possibile ricavare la saturazione di ossigeno nel sangue con una formula che verrà dettagliatamente descritta nel capitolo successivo (v. Cap. 7).

Come per l'elettrocardiogramma, vengono mostrati intervalli di cinque secondi ed è possibile fare lo zoom del grafico. Il segnale ha solitamente una caratteristica forma con una sequenza di massimi e minimi, come vediamo nella Fig. 6.12.

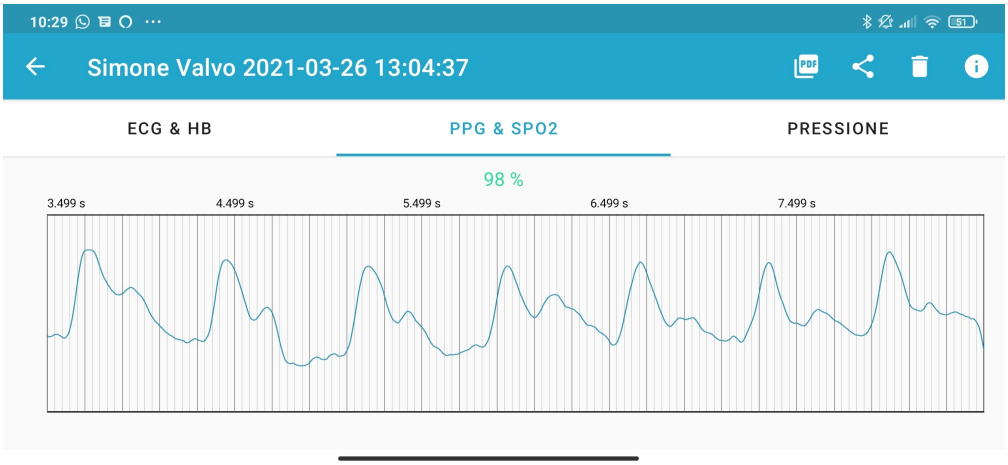


Figura 6.12: Dettaglio della fotopletismografia con il valore di ossigenazione calcolato

Pressione Infine, è presente una scheda per la visualizzazione delle pressioni sistolica e diastolica (v. Fig. 6.13). La finestra può includere dei dettagli riguardo dei valori fuori da un determinato intervallo, specificato nella Sez. 6.1.4, che dovranno comunque essere riesaminati e valutato da un esperto.

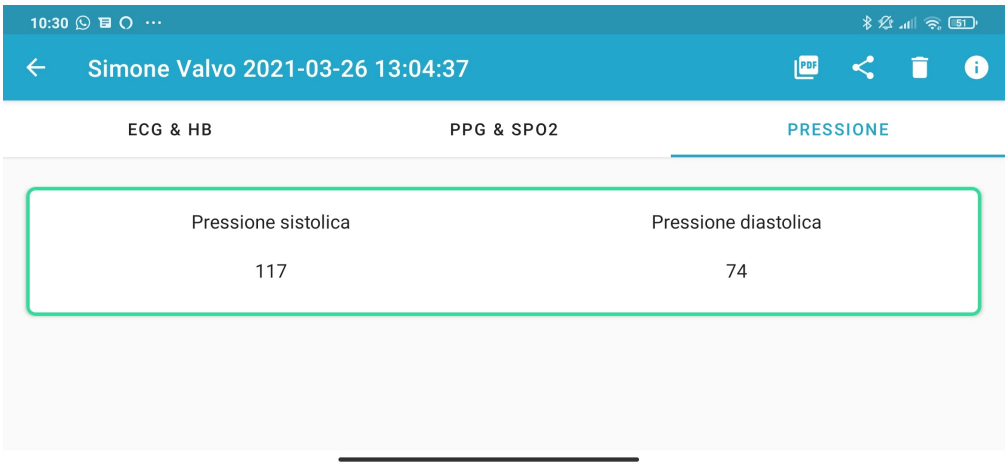


Figura 6.13: Dettaglio delle pressioni sistolica e diastolica

PDF Anche il file PDF, date le nuove informazioni, è stato modificato in modo tale da poter visualizzare tutte le informazioni acquisite e poterle condividere senza che il destinatario abbia necessariamente l'applicazione installata. Il PDF, mostrato in Fig. 6.14, includerà:

- nome e cognome dell'utente che ha effettuato la misurazione;
- data e orario di misurazione;
- durata della misurazione;
- posizione dell'utente (se attivata precedentemente dalle impostazioni);
- anomalie rilevate;
- frequenza cardiaca;
- saturazione dell'ossigeno nel sangue;
- elettrocardiogramma con una carta millimetrata rossa. Il lato dei quadrati piccoli corrisponderà a 1mm ed equivarranno a 40 millisecondi lungo l'asse delle ascisse e a 0.1mV lungo l'asse delle ordinate;
- fotopletismografia che non contiene una griglia specifica ma solo delle indicazioni temporali.

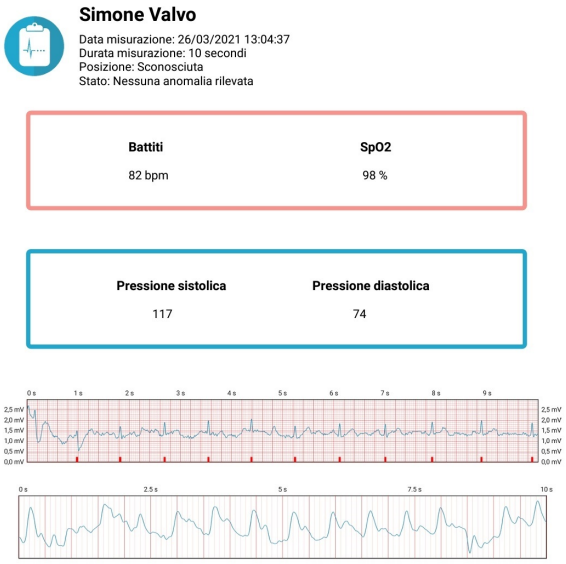


Figura 6.14: PDF creato dall'applicazione PulseEcg

6.2 Back-End

6.2.1 Gestione attività

L'interfaccia di PulseEcg è gestita in maniera molto diversa rispetto a quella di EcgWatch. La nuova applicazione presenta un numero molto basso di activity ma un numero elevato di fragment (v. Fig. 6.15). Le activity fungono da contenitori per i fragment e hanno il compito di gestire ed elaborare i dati che verranno utilizzati e visualizzati.

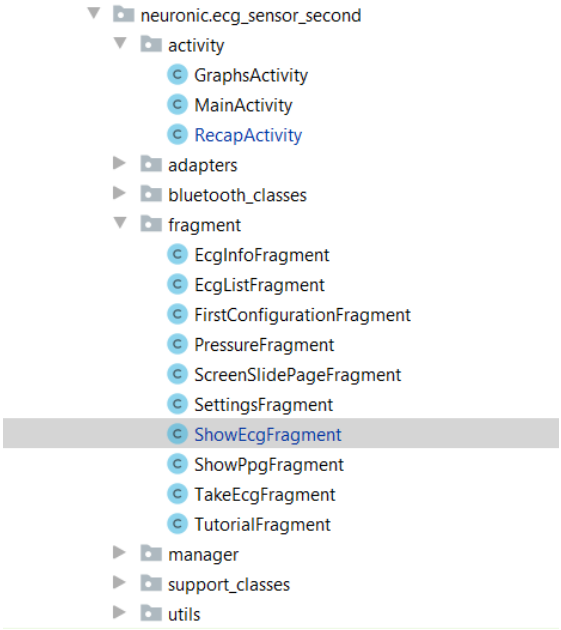


Figura 6.15: Organizzazione del progetto di PulseEcg

Il codice sottostante mostra la classe SectionPageAdapter presente all'interno del file GraphsActivity. Questa gestisce i fragment che mostrano i dettagli dei segnali e della pressione arteriosa. Mediante la funzione newInstance, si effettua il passaggio dei dati al fragment corrispondente. L'operazione viene svolta alla creazione del fragment, passando alla funzione un identificatore per selezionare la finestra da aprire.

```
public class SectionsPagerAdapter extends FragmentStateAdapter {
    public SectionsPagerAdapter(FragmentActivity activity) {
        super(activity);
    }
    @NonNull
```

```
    @Override
    public Fragment createFragment(int position) {
        switch (position) {
            case 0:
                showEcgFragment =
    ShowEcgFragment.newInstance(healthObject.getECGSamples(),
    healthObject.getHB(), healthObject.isAtrialFibrillation());
                return showEcgFragment;
            case 1:
                showPpgFragment =
    ShowPpgFragment.newInstance(healthObject.getPPGSamplesRED(),
    healthObject.getSpO2());
                return showPpgFragment;
            case 2:
                pressureFragment =
    PressureFragment.newInstance(healthObject.getPressSis(),
    healthObject.getPressDia());
                return pressureFragment;
            default:
                return null;
        }
    }
}
```

All'interno del fragment, avremo la funzione corrispondente che permette di salvare internamente i dati passati all'interno di un oggetto di tipo Bundle, contenitore di vari tipi di dato. Il codice mostra l'estrazione tre tipi di dato diverso inviati attraverso il SectionPagerAdapter.

```
public static ShowEcgFragment newInstance(double[] ecgSamples, int hb,
    boolean af) {
    Bundle args = new Bundle();
    ShowEcgFragment fragment = new ShowEcgFragment();
    args.putDoubleArray("ECG", ecgSamples);
    args.putInt("HB", hb);
    args.putBoolean("AF", af);
    fragment.setArguments(args);
    return fragment;
}
```

6.2.2 Bluetooth Low Energy

Come visto nel capitolo 3.3.2, il funzionamento del Bluetooth Low Energy è profondamente diverso rispetto alla versione 2.0. Affinchè vi sia un corretto funzionamento è necessario richiedere un maggior numero di permessi all'interno del manifest file, mostrati nel codice seguente:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
    />
<uses-permission
    android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
    />
```

Richiede dei permessi non soltanto legati prettamente al servizio Bluetooth ma anche dei permessi di localizzazione e di accesso alla rete. Questi permessi dovranno dunque essere richiesti all'utente al primo avvio dell'applicazione. La comunicazione BLE è basata su un sistema client-server gestita attraverso il profilo di attributo generico (GATT). Nel nostro caso l'applicazione PulseEcg verrà identificata come client mentre il bracciale come server. Il servizio che andremo ad utilizzare per la comunicazione sarà unico.
Per quanto riguarda le caratteristiche, possiamo trovarne tre di tipo "Notify" e una di tipo "Write". Le caratteristiche di notifica corrispondono alla lettura della batteria, dei valori misurati dagli elettrodi e dal sensore rosso e infrarosso. Esse devono essere dapprima abilitate attraverso l'applicazione e successivamente, inviando il tempo di acquisizione tramite la caratteristica di scrittura, il GATT server sarà pronto per la misurazione e l'invio dei dati. Quando verrà rilevato un segnale elettrico sul sensore rosso/infrarosso, il bracciale inizierà a trasmettere i dati.
La funzione askNotify permette di attivare la caratteristica di notifica nel bracciale e poter inviare le varie misurazioni. Viene quindi inviato il descrittore con le informazioni della caratteristica e il segnale di abilitazione.

```
public boolean askNotify(BluetoothGattCharacteristic characteristic) {
    BluetoothGattDescriptor descriptor = characteristic.getDescriptor(
        UUID.fromString(BleManager.CLIENT_CHARACTERISTIC_CONFIG));

    descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
    return mBluetoothGatt.writeDescriptor(descriptor);
}
```

Per richiedere successivamente l'acquisizione, si utilizza la funzione requireAcquisition che invia semplicemente la caratteristica di scrittura, con il numero di secondi di acquisizione, al bracciale. La gestione del Bluetooth all'interno dell'applicazione è demandata a due classi: l'attività principale, che si occuperà di mostrare in maniera corretta le varie fasi e di inviare correttamente i dati ricevuti, e la classe BleService. Quest'ultima è la classe principale che verrà utilizzata "legata" alla mainActivity attraverso la funzione bindService(). Questa funzione permette di "vincolare" un servizio che diventa il server in un'interfaccia client-server. Permette ai componenti (come le attività) di legarsi al servizio, di inviare richieste, di ricevere risposte e di eseguire comunicazioni interprocesso (IPC). Un servizio vincolato vive tipicamente solo mentre serve un altro componente dell'applicazione e non viene eseguito in background indefinitamente. Il passaggio dei dati avviene registrando un BroadcastReceiver all'interno dell'activity che permetterà di gestire gli intent inviati tramite la funzione sendBroadcast(intent) da parte del servizio.

```
private final BroadcastReceiver mGattUpdateReceiver = new
   BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        gattReceiveAnswer.set(true);
        final String action = intent.getAction();
        switch (action) {
        case BleService.ACTION_GATT_CONNECTED:
            ivBluetooth.setImageResource(R.drawable.ic_ble_connected);
            tvBluetoothDevice.setText(bleDevice.getName());
            mBtConnected.set(true);
            LocalData.setPrefString(MainActivity.this,
   LocalData.DEVICE, bleDevice.getName());
            break;
        }
    }
};
```

Il BroadcastReceiver gestirà l'avvenuta connessione/disconnesione con il GATT server, la notifica di lettura del valore della batteria, la notifica di arrivo dei segnali e dei due messaggi di errore. In particolare, alla conferma di arrivo dei segnali, questi verranno rimodulati prima di essere passati all'attività successiva, in cui avverrà l'elaborazione vera e propria degli stessi. Il codice mostra un esempio di gestione del BroadcastReceiver quando si rileva l'avvenuta connessione con il dispositivo BLE. Questa modificherà l'icona del Bluetooth nella schermata principale e salverà il nome del dispositivo in memoria in modo tale da poter poi riconnettersi automaticamente.

6.2.3 Gestione dati

Data la maggior mole di dati processati e memorizzati dall'applicazione, essa contiene una classe serializzabile, chiamata HealthObject, che comprende tutte le informazioni acquisibili da una misurazione, oltre al nome e cognome dell'utente e alla data di effettuazione della stessa. Le nozioni presenti nella classe sono:

- i valori del segnale acquisito tramite gli elettrodi, ovvero l'elettrocardiogramma (non filtrato);
- i due segnali ricavati dal sensore rosso e infrarosso, ovvero la fotopletismografia (entrambi senza filtri);
- il numero dei battiti per minuto calcolato;
- la saturazione di ossigeno nel sangue;
- le pressioni sistolica e diastolica;
- il valore booleano per la verifica della fibrillazione atriale.

L'alto numero di variabili inserite all'interno del file Json (v. Fig. 6.16) permette di effettuare i calcoli solo una volta, prima del salvataggio del file. Questo viene fatto perchè le funzioni di misurazione potrebbero occupare la memoria RAM, il processore e rallentare l'applicazione. Inoltre, il file di tipo Json consente una facile amministrazione di questi dati, oltre che un'ampia compatibilità con altri sistemi nel caso i dati si volessero esportare in qualche altro software.

Figura 6.16: Esempio di un file Json salvato da PulseEcg e visualizzato con Json Editor Online

La lettura e la scrittura del file vengono effettuate con l'ausilio della libreria Gson (v. Sez. 4.3.1). Le funzioni interne della libreria, visualizzate nel codice successivo, permettono la serializzazione e la deserializzazione di un oggetto "serializable" come HealthObject in maniera automatica, dichiarando la funzione toString() nella classe. Le funzioni risultano molto veloci nonostante i dati inseriti nel file siano molteplici e anche di grandi dimensioni. Prima della deserializzazione della stringa Json, la lettura viene fatta attraverso un FileInputStream mentre la scrittura dopo la serializzazione mediante un BufferedWriter.

```
public static String serializationJson(HealthObject patient, Context
    ctx) {
     Gson gson = new GsonBuilder().setPrettyPrinting().create();
     String save= gson.toJson(patient);

     return file;
}
public static HealthObject deserializationJson(File file) {
     Gson gson = new Gson();
     String json= getJsonFromFile(file);
     Type healthObjectType = new TypeToken<HealthObject>() { }.getType();
     return gson.fromJson(json, healthObjectType);
}
```

Capitolo 7

Elaborazione dei segnali

Le due applicazioni richiedono diversi tipi di filtri per ridurre il rumore dei segnali e per visualizzarli correttamente. Sono inoltre richiesti degli algoritmi per estrapolare le informazioni dai valori dell'elettrocardiogramma e della fotopletismografia. Questi algoritmi serviranno per ricavare i picchi dei segnali, per misurare la frequenza cardiaca e per calcolare il valore di saturazione dell'ossigeno nel sangue. Gli algoritmi implementati o testati sono presi da fonti ufficiali per cui sono implementati anche in strumentazioni ufficiali. Prima dell'inserimento nell'applicazione Android sono stati comunque provati e ottimizzati per il nostro specifico dispositivo attraverso Matlab. Questo software è infatti progettato appositamente per la manipolazione di segnali e le funzioni interne permettono di provare in modo rapido alcuni filtraggi e determinati calcoli. Il capitolo descriverà i tipi di filtri e gli algoritmi utilizzati e la loro implementazione.

7.1 Funzioni di EcgWatch

Le funzioni di filtraggio del segnale elettrocardiografico, di identificazione dei picchi R e del calcolo della frequenza cardiaca, erano già state implementate nell'applicazione Ecg Sensor. In questo progetto sono state riscritte in modo più efficiente, sono state valutate ulteriori alternative e sono state generalizzate in quanto dipendenti dalla libreria GraphView, non più utilizzata. In particolare, si è stato modificato il tipo di filtraggio della misurazione rispetto alla versione del 2014 [12] passando ad un filtro di tipo Butterworth.

7.1.1 Filtro mediano

Il filtro mediano è una tecnica di filtraggio digitale non lineare, spesso usata per rimuovere il rumore dai segnali. Tale riduzione del rumore è un tipico passo di pre-elaborazione per migliorare i risultati dell'operazione successiva. In questo caso particolare è stato utilizzato per rimuovere il rumore dovuto al movimento del paziente. Per permettere ciò, è stata creata una linea che segue l'andamento dell'ECG e poi sottratta alla forma d'onda originale [12].

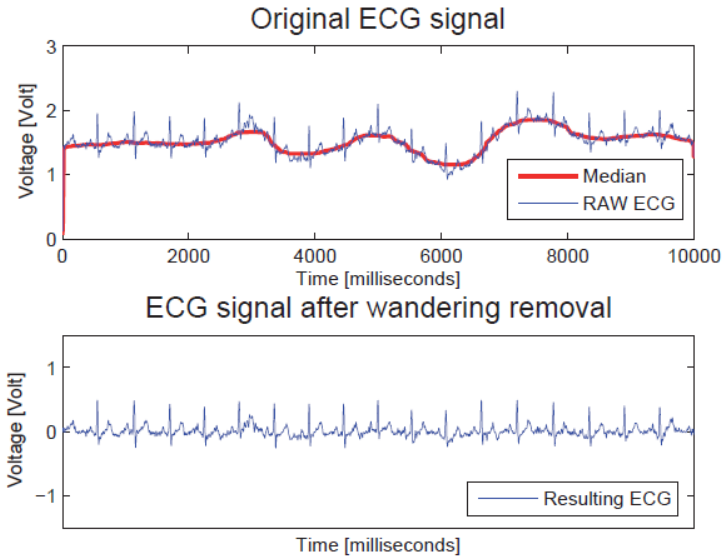


Figura 7.1: Applicazione di un filtro mediano ad un elettrocardiogramma misurato

Il filtro all'interno dell'applicazione è utilizzato due volte: il primo utilizza una finestra di mille elementi, la seconda ne usa una di quattordici elementi. In corrispondenza dei limiti del vettore, si è deciso di sommare l'ultimo valore limite dell'array tante volte quante sono i valori della finestra mancanti.
Questo filtro permetterà di avere uno "smoothing" del segnale (v. Fig. 7.1), ovvero verrà evidenziato il pattern significativo [33]. Dopo l'applicazione di questo filtro, ne verrà introdotto un altro chiamato Butterworth.

```
public static void filterEcgBias(double[] input, double[] output, int
    window) {
     int i,j;
     for (i=0; i<input.length; i++) {
         output[i]=0;
         for ( j=(-window); j <= window; j++)
```

```
        {
            if((i+j) < 0)
                output[i] += input[0];
            else if((i+j) >= input.length)
                output[i] += input[input.length-1];
            else
                output[i] += input[i+j];
        }
        output[i]=output[i]/((2*window)+1);
    }
}
```

7.1.2 Filtro Savitzky-Golay

Nella prima versione dell'applicazione e durante la tesi che ha portato alla realizzazione della prima applicazione [12], era stato implementato un filtro Savitzky-Golay. Quest'ultimo è un filtro digitale che può essere applicato a un insieme di punti di un segnale digitale allo scopo di smussare i dati, cioè di aumentarne la precisione senza distorcere la tendenza dell'impulso (v. Fig. 7.2). Questo si ottiene, in un processo noto come convoluzione, adattando successivi sottoinsiemi di punti dati adiacenti con un polinomio di basso grado con il metodo dei minimi quadrati lineari [34]. Venne implementato in questo tipo di applicazione poichè i filtri a risposta

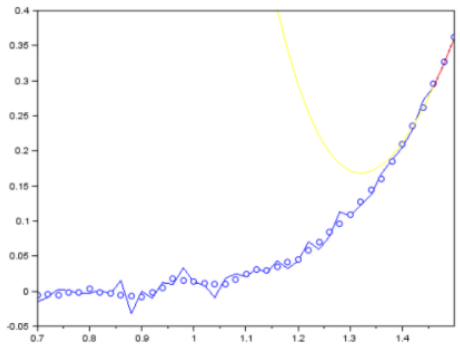


Figura 7.2: Esempio di applicazione del filtro Savitzky-Golay

finita all'impulso (FIR) tendevano a filtrare una parte significativa del contenuto ad alta frequenza del segnale insieme al rumore. Il filtro Savitzky-Golay però, seppur preservando gli impulsi ad alta frequenza, risultava poco efficace quando i livelli di rumore erano particolarmente alti. Per questo è stato successivamente sostituito con un filtro di tipo Butterworth, spiegato nella sezione successiva.

7.1.3 Filtro Butterworth

Il filtro Butterworth è un tipo di filtro di elaborazione del segnale progettato per avere una risposta in frequenza il più piatta possibile nella banda mentre fuori banda ha una funzione di trasferimento monotona, tendente a zero passante. È anche indicato come filtro a magnitudine massimamente piatta. Il più semplice filtro Butterworth è il filtro passa-basso standard di primo ordine, che può essere modificato per ottenere un filtro passa-alto, e combinato in serie con altri per ottenere filtri passa-banda, filtri elimina-banda, e versioni di ordine superiori di questi [35].
Nel nostro caso, il filtro butterworth è utilizzato come filtro passa-banda.

```
public static void filterButterworth(float[] input)
{
    float[] temp= new float[NO_OF_SAMPLES];
    float calc;
    float[] inputHistory = new float[2];
    float[] outputHistory = new float[3];
    int i;
    float[] a = new float[]{
            -1.86689227971171f,
            0.875214548253684f
    };
    float[] b = new float[]{
            0.00208056713549229f,
            0.00416113427098458f,
            0.00208056713549229f
    };

    for (i=0; i<1000; i++) {
        calc=b[0] * input[0] + b[1] * inputHistory[0] + b[2] *
   inputHistory[1] - a[0] * outputHistory[0] - a[1] * outputHistory[1];
        inputHistory[1] = inputHistory[0];
        inputHistory[0] = input[0];

        outputHistory[2] = outputHistory[1];
        outputHistory[1] = outputHistory[0];
        outputHistory[0] = calc;
    }
    for (i = 0; i < input.length; i++)
    {
        calc=b[0] * input[i] + b[1] * inputHistory[0] + b[2] *
   inputHistory[1]  - a[0] * outputHistory[0] - a[1] * outputHistory[1];
```

```
        inputHistory[1] = inputHistory[0];
        inputHistory[0] = input[i];

        outputHistory[2] = outputHistory[1];
        outputHistory[1] = outputHistory[0];
        outputHistory[0] = calc;
        temp[i] = outputHistory[0];
    }

    System.arraycopy( temp, 0, input, 0, input.length );
}
```

Il codice è stato precedentemente ricavato e testato su Matlab. Da qui, sono stati ricavati i valori presenti nei vettori a e b, funzionali per imporre un filtro butterworth a 15Hz. Le costanti verranno utilizzate per il calcolo degli output, in dipendenza dei valori in input e dei valori in output precedentemente calcolati. Questa funzione viene richiamata due volte, la prima con il segnale reale e la seconda invertendo il segnale filtrato rispetto l'asse delle ascisse.
Il filtro butterworth riesce ad ottenere un segnale abbastanza chiaro con l'EcgWatch ma presenta delle limitazioni in quanto questo filtro potrebbe decrementare anche il valore dei picchi R. Per questo motivo non verrà implementata la stessa funzione nel PulseEcg. Nelle immagini successive possiamo vedere un elettrocardiogramma prima del filtro (v. Fig. 7.3) e dopo il filtro (v. Fig. 7.4)

Figura 7.3: Esempio di un segnale prima dell'applicazione del filtro butterworth

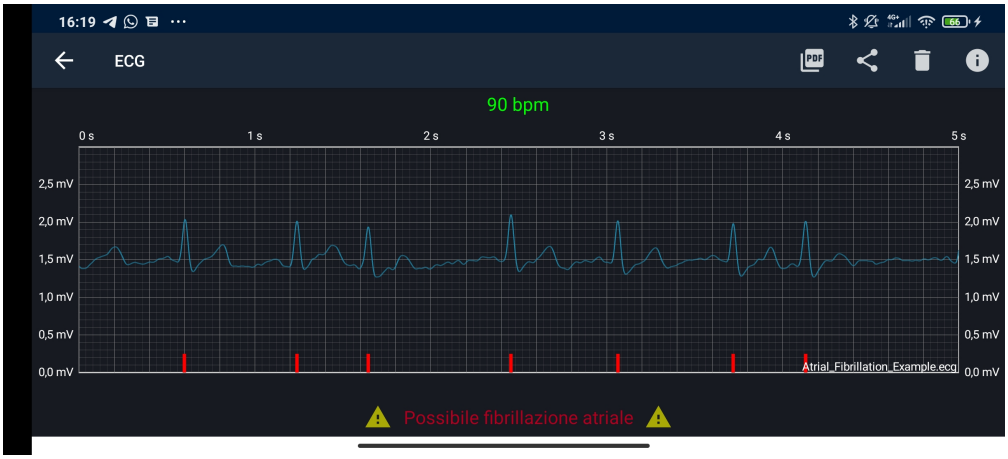


Figura 7.4: Esempio di un segnale dopo l'applicazione del filtro butterworth

7.1.4 Calcolo della frequenza cardiaca

La frequenza cardiaca viene calcolata facendo riferimento alla distanza tra i picchi R rilevati (distanza R-R). In particolare, la somma di tutte queste distanze viene utilizzata per dividere il valore di una costante (con valore 60000) moltiplicata per 0,88.

$$HeartBeat = \frac{60000 * 0{,}88}{RR} \tag{7.1}$$

Grazie alla Eq. 7.1, è possibile calcolare, entro un errore accettabile, il valore della frequenza cardiaca. In base ai valori intermedi dei battiti cardiaci, viene incrementato un contatore che rileva se l'andamento presenta delle anomalie, ovvero se il valore dei battiti attuale si discosta di 5 o più dal valore dei battiti precedenti. Se questo contatore supera per più della metà il numero totale di picchi rilevati, allora si segnalerà una fibrillazione atriale.
Grazie alla fotopletismografia sarà in seguito possibile valutare ulteriori alternative più precise per il calcolo della frequenza cardiaca [36].

7.1.5 Algoritmo di rilevamento picchi

Sergey Chernenko L'algoritmo utilizzato nell'applicazione Ecg Sensor richiama un algoritmo creato da Sergey Chernenko [37]. L'algoritmo era stato da lui implementato in Matlab ed è stato successivamente riadattato per Android [12]. L'algoritmo prevedeva di rimuovere la componente a bassa frequenza, di rilevare i massimi locali e di applicare un filtro di threshold. Questo tipo di elaborazione cercava di marcare in maniera evidente i valori più "alti" del segnale. Per la rimozione della componente a bassa frequenza, è prevista l'applicazione di una trasformata veloce di Fourier (FFT) prima della rimozione e di una trasformata inversa dopo averli eliminati. Per i massimi locali, viene semplicemente definita una finestra di ricerca.
L'applicazione di questo algoritmo risultava abbastanza semplice in Matlab poichè esso possiede un linguaggio specificatamente definito per operazioni matematiche complesse. Per una applicazione Android, definire questi elementi risulta maggiormente complesso ed è stata valutata una soluzione più semplice che fornisce un buon rilevamento dei picchi.

Pan-Tompkins Un'altra soluzione che è stata valutata e implementata è l'algoritmo di Jiapu Pan e William Tompkins [38]. L'algoritmo Pan-Tompkins applica una serie di filtri per evidenziare il contenuto di frequenza di questa rapida depolarizzazione del cuore e rimuove il rumore di fondo. Successivamente, quadra il segnale per amplificare il contributo del QRS. Infine, applica soglie adattive per rilevare i picchi del segnale filtrato. Queste azioni corrispondono nello specifico al:

1. filtro passa-banda per aumentare il rapporto tra il segnale e il rumore;
2. filtro passa-alto se si vuole ridurre il costo computazionale e permettere il rilevamento durante l'acquisizione;
3. filtro derivativo per avere informazioni riguardo la pendenza del complesso QRS;
4. segnale elevato alla potenza di due in modo da sviluppare ulteriormente i picchi dominanti;
5. filtro a media mobile per ottenere informazioni sulla durata del complesso QRS.

Nella Fig. 7.5 possiamo vedere l'applicazione di questa serie di operazioni, semplificate per non decrementare la velocità dell'applicazione. I picchi R risultano molto accentuati rispetto al resto del segnale. A questo punto basterà rilevare i picchi massimi locali, ovvero dove il segnale cambia direzione.

Dopo ogni picco, non può esserne rilevato un altro nei 200 millisecondi successivi, vincolo fisiologico dovuto al periodo refrattario durante il quale la depolarizzazione ventricolare non può avvenire anche in presenza di uno stimolo [38]. L'algoritmo però era soggetto a problemi se il segnale risultava rumoroso da una misurazione effettuata [39]. Successivamente, è stato valutato un ulteriore algoritmo.

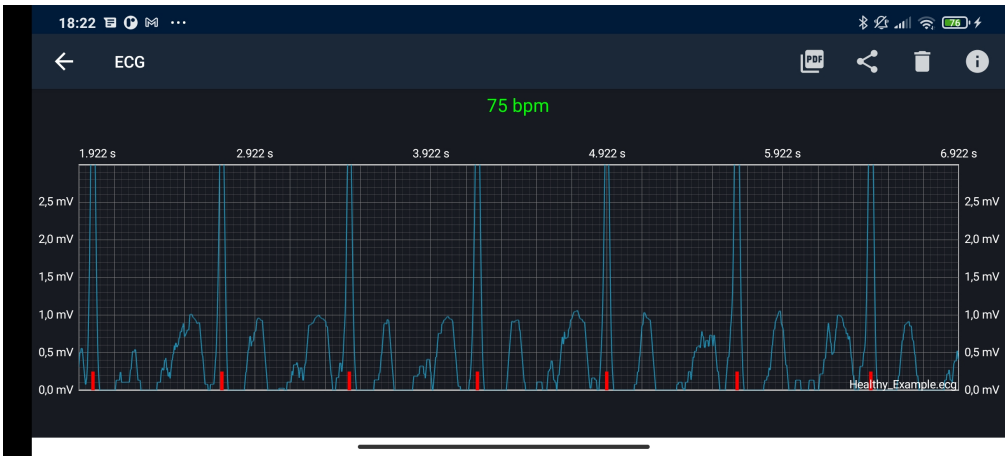


Figura 7.5: Esempio di applicazione del filtraggio del segnale necessario per l'algoritmo Pan-Tompkins

Algoritmo EcgWatch L'algoritmo attualmente utilizzato controlla, per ogni valore, la pendenza della curva nell'intervallo di tempo precedente e successivo al punto attualmente studiato. In particolare, la pendenza sarà confrontata con una percentuale della media misurata tra i cinque valori più alti dell'intero segnale, valore calcolato all'inizio della funzione. L'algoritmo ha come premessa il fatto che due picchi R consecutivi non possano essere ad una distanza di tempo inferiore a 300 millisecondi. Se all'interno di questo intervallo di tempo viene rilevato un altro possibile punto di picco R, verrà considerato solo quello che avrà valore maggiore.

```
public static ArrayList<Integer> findPeakNew(double[] input) {
    int length=input.length;
    int interval = 150;
    int k=0, i, j;
    int[] maxima = new int[length/200];
    int temp_max;
    ArrayList<Integer> peak =new ArrayList<>();
    double absolute_max = averaged_max(input);
    // FIND PEAKS
```

```
    for (i=0; (i*interval) < length; i++)
    {
        temp_max = i*interval;
        for (j = (i*interval); j < length && j < (i+1)*interval; j++)
        {
            if (input[j] > input[temp_max])
                temp_max = j;
        }
        if (temp_max < 35 || // A peaks cannot be at the beginning
                temp_max > length - 35 ||
                (input[temp_max]-input[temp_max+30]) <
    (absolute_max-1.5)*0.40 ||
                (input[temp_max]-input[temp_max-20]) <
    (absolute_max-1.5)*0.35)
            continue;

        if(k>0)
        {
            if (temp_max-maxima[k-1] < 300) // Two peaks cannot be too
    close
            {
                double slope_temp =
    (input[temp_max]-input[temp_max+30]);
                double slope_old =
    (input[maxima[k-1]]-input[maxima[k-1]+30]);
                if (slope_temp>slope_old)
                {
                    maxima[k-1]=temp_max;
                }
                continue;
            }

        }
        peak.add(temp_max);
        maxima[k] = temp_max;
        k++;
    }
    return peak;
}
```

Qui di seguito possiamo vedere l'algoritmo applicato a quattro segnali diversi: la Fig. 7.6 mostra un segnale con un andamento normale e molto chiaro; la Fig. 7.7 possiede un picco R molto basso; la Fig. 7.8 è invece un po' rumorosa con dei picchi R molto alti; infine la Fig. 7.9 mostra un segnale molto rumoroso. Nonostante i vari segnali, l'algoritmo riesce ad individuare correttamente i picchi R.

Figura 7.6: Esempio di rilevamento picchi con un segnale normale

Figura 7.7: Esempio di rilevamento picchi con un segnale che presenta dei picchi R molto bassi

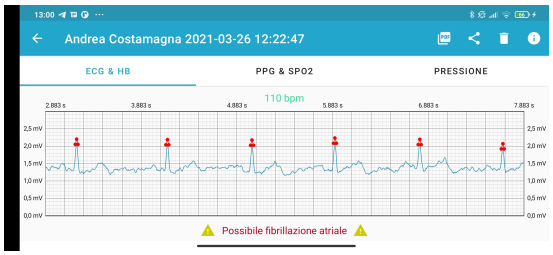


Figura 7.8: Esempio di rilevamento picchi con un segnale che presenta dei picchi R molto alti

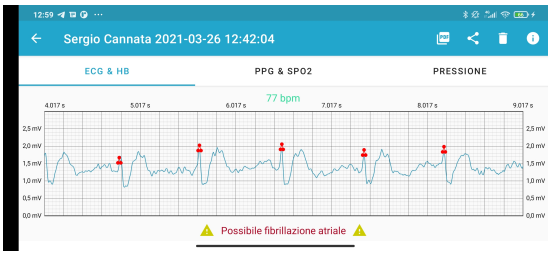


Figura 7.9: Esempio di rilevamento picchi con un segnale rumoroso

7.2 Funzioni di PulseEcg

All'interno del PulseEcg troviamo due ulteriori difficoltà da gestire per l'elaborazione dei segnali: la fotopletismografia risulta un segnale completamente nuovo che non veniva gestito nella precedente applicazione; la rete neurale necessita di un particolare pre-processamento dei dati prima di essere inviati in input alla rete. Oltretutto, il campionamento dei segnali è diverso da quello precedente. Data questa caratteristica e data una configurazione hardware differente, anche il segnale elettrocardiografico dovrà essere elaborato in maniera diversa.

7.2.1 Elaborazione segnale elettrocardiografico

Il segnale dell'elettrocardiogramma, come precedentemente annunciato, ha un campionamento diverso, con una frequenza pari a 500Hz. Il bracciale acquisirà quindi 5000 valori in un tempo di 10 secondi (nell'EcgWatch erano 10000). Il segnale sarà inoltre molto rumoroso e con valori molto alti. In questo capitolo vedremo quindi come ottenere un segnale filtrato leggibile e disponibile per l'utente.

Pre-processamento Prima di ogni filtraggio, il segnale acquisito verrà traslato verso il basso con un valore pari a 1650mV e verrà poi diviso il segnale ottenuto per 750, utilizzato come divisore di frequenza. Otterremo un segnale come quello in Fig. 7.10.

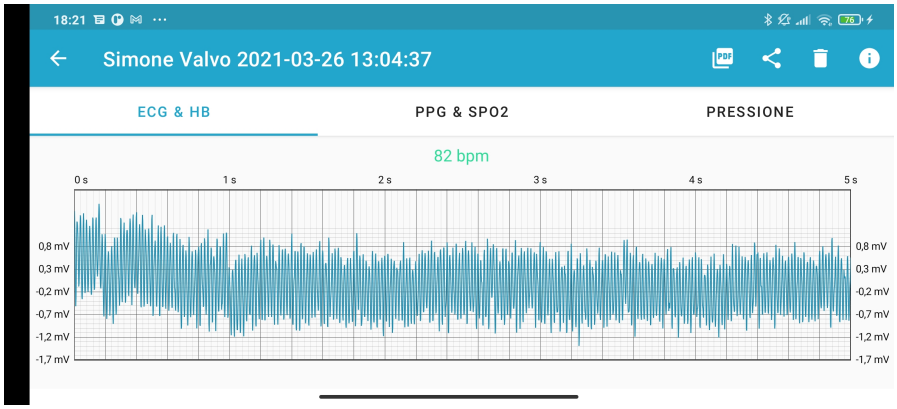


Figura 7.10: Esempio di un segnale dopo il pre-processamento

Filtro notch Il filtro notch, o filtro elimina-banda, è un filtro che fa passare la maggior parte delle frequenze inalterate, ma attenua quelle in una gamma specifica a livelli molto bassi. È l'opposto di un filtro passa-banda. Esistono diversi modi di implementare un filtro notch [40], soprattutto in hardware. L'implementazione software all'interno dell'applicazione PulseEcg prevede l'utilizzo di una funzione biquadratica ripetuta quattro volte con valori diversi. La funzione biquadratica, nello specifico, sarà:

```
public static void bquad(double[] input, double b0, double b1, double
    b2, double a1, double a2) {
     double x1=0, x2=0, y1=0, y2=0;
     double part1, part2;
     for(int i =0; i< input.length; i++) {
         part1 = b0 * input[i] + b1 * x1 + b2 * x2;
```

```
        part2 = a1 * y1 + a2 * y2;
        x2=x1;
        x1=input[i];
        input[i]= part1-part2;
        y2=y1;
        y1=input[i];
    }
}
```

Per ottenere questo tipo di filtro, è stato impiegato l'utilizzo di Matlab e sono state fatte varie prove con il segnale. L'applicazione del filtro notch darà un segnale meno rumoroso ma ancora non distinguibile, come mostrato nell'esempio in Fig. 7.11.

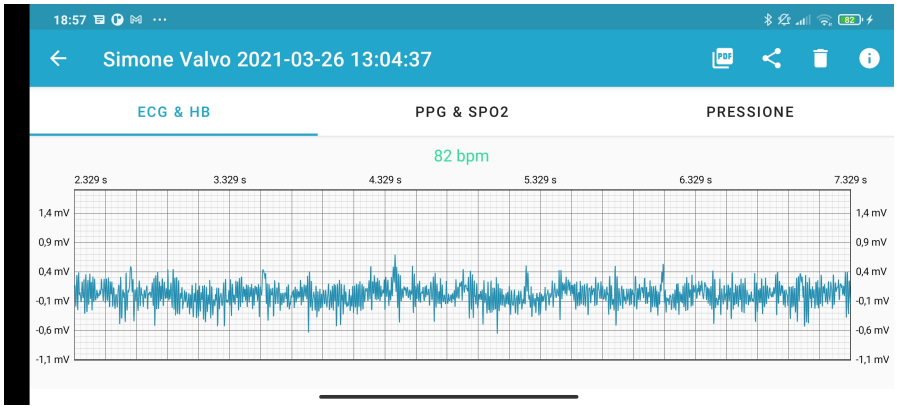


Figura 7.11: Esempio di un segnale dopo il filtro notch

Media mobile La media mobile viene usato per fare smoothing dei dati, cioè una correzione della distribuzione del segnale per ridurre le misure che sono eccessivamente diverse dalle altre. In questo modo si ha una riduzione del rumore dei dati. Con questa operazione si riesce ad appiattire il segnale in modo da visualizzare in maniera corretta l'elettrocardiogramma.

```
public static double[] movingAverage(double[] vector, int window) {
    int j=0;
    double temp =0;
    int medium = window/2, count=0;
    double[] output= new double[vector.length];

    for(int i=0; i<vector.length; i++)
```

```
    {
        for(j=i-medium; j<= i+medium; j++) {
            if(j>= 0 && j<vector.length) {
                temp+=vector[j];
                count++;
            }
        }
        output[i]=temp/count;
        temp=0;
        count=0;
    }
    return output;
}
```

La funzione riceve come parametro il segnale da filtrare e la grandezza della finestra su cui calcolare la media. È stato scelto abbastanza piccolo lasciando permanere un po' del rumore del segnale, come vediamo nella Fig. 7.12. Si è scelto volutamente di lasciare un filtraggio più leggero del segnale così da non perdere l'informazione sottostante, al contrario di altri programmatori che tagliano fortemente le frequenze per avere un segnale bello da visualizzare ma meno significativo a livello medico.

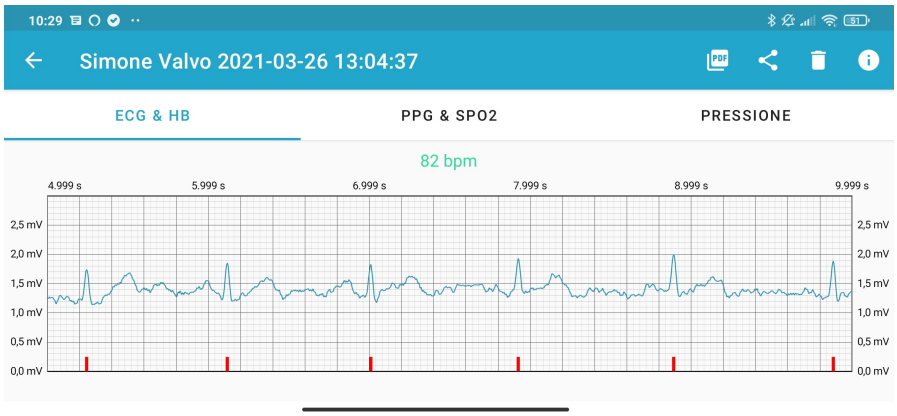


Figura 7.12: Esempio di un segnale dopo aver completato tutte le fasi del filtraggio

7.2.2 Elaborazione segnale fotopletismografico

Il filtraggio della fotopletismografia è risultato più semplice rispetto all'elettrocardiogramma. Anche questo segnale ha una frequenza di 500Hz e possiede 5000 campioni per il rosso e 5000 campioni per l'infrarosso. Una volta arrivato il segnale allo smartphone, questo inizierà con il suo pre-processamento.

Pre-processamento I segnali rosso e infrarosso arrivano allo smartphone invertiti rispetto all'asse delle ordinate, sarà quindi necessario provvedere a capovolgere i segnali rispetto ai loro valori mediani. L'Eq. utilizzata per l'inversione è indicata in 7.2.

$$invertedValue = 2 * average - Value \tag{7.2}$$

Nella Fig. 7.13 vediamo il segnale dopo l'applicazione di tale formula.

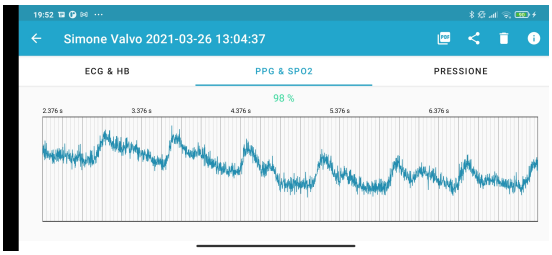


Figura 7.13: Fotopletismografia dopo il pre-processamento

Media mobile e detrending Il filtraggio consiste semplicemente nell'applicazione della media mobile per due volte consecutive con una finestra pari a 20. Questo filtro mostra già un segnale fotopletismografico definito come vediamo nella Fig. 7.14. Possiamo però notare un aspetto discendente del segnale dovuto all'acquisizione dal dispositivo. Per raddrizzare questo segnale, applichiamo una funzione di detrending, ovvero rimuoviamo la tendenza discendente dalla serie temporale. Questo viene fatto sfruttando una media mobile con una finestra molto grande e la media totale. Nello specifico, la funzione traslerà il valore in modo che abbia significato rispetto l'asse di origine delle ordinate e lo sposterà successivamente nella mediana calcolata in precedenza. Il risultato è mostrato nella Fig. 7.15.

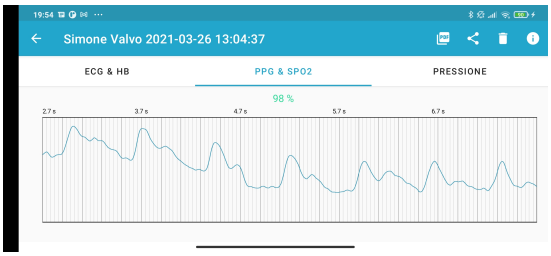


Figura 7.14: Fotopletismografia dopo aver applicato la media mobile

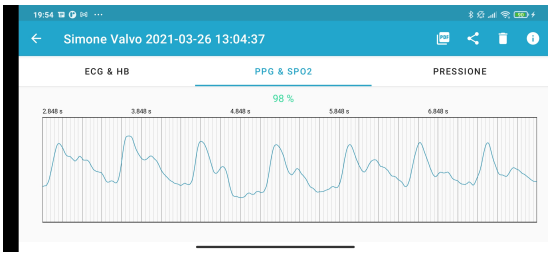


Figura 7.15: Fotopletismografia dopo aver completato il filtraggio

7.2.3 Calcolo della saturazione di ossigeno nel sangue

La saturazione di ossigeno nel sangue fornisce informazioni sulle prestazioni del sistema respiratorio. La misurazione non invasiva della saturazione utilizza impulsi fotopletismografici nelle regioni del rosso e dell'infrarosso e sfrutta i diversi spettri di assorbimento della luce da parte dell'emoglobina ossigenata e de-ossigenata [41]. Il principio è già utilizzato dai classici pulsossimetri (o saturimetri) commerciali. La funzione presente nell'applicazione utilizza i segnali rosso e infrarosso calcolandone le coppie di massimi e minimi locali all'interno di una finestra, la cui dimensione è stata scelta dopo diversi tentativi. Per ogni coppia di massimi e minimi viene calcolato un valore (chiamato Ratio) pari alla Eq. 7.3. Il valore nel pedice indica la coppia i-esima.

$$Ratio_i = (massimo_i - minimo_i)/minimo_i \tag{7.3}$$

Viene poi calcolato il rapporto tra i valori determinati del segnale rosso e il valore corrispondente alla coppia nella medesima posizione del segnale infrarosso. Si ottiene un vettore con all'interno il risultato di questi rapporti. Questi valori verranno utilizzati per calcolare un vettore di valori di saturazione attraverso un'ulteriore formula. Il valore poi mostrato sarà la media di questi valori. Il codice sottostante mostra la funzione di calcolo spiegata in questo paragrafo.

```
public static int spo2Evaluation(double[] ppgR, double[] ppgIR) {
    boolean checkR= true, checkIr=true;
    ArrayList<Double> maxR= new ArrayList<>(), minR= new ArrayList<>(),
            maxIr= new ArrayList<>(), minIr= new ArrayList<>();
    double temp;
    ArrayList<Double> ratioR= new ArrayList<>(), ratioIr= new
   ArrayList<>(),
            ratioTot= new ArrayList<>();
    int spo=0;

    for(int i=0; i< ppgR.length; i++) {
        if(checkR) {
            if(controlMaxorMin(ppgR, i, true)) {
                maxR.add(ppgR[i]);
                checkR=false;
            }
        } else {
            if(controlMaxorMin(ppgR, i, false)) {
                minR.add(ppgR[i]);
                checkR=true;
            }
```

```
        }
        if(checkIr) {
            if(controlMaxorMin(ppgIR, i, true)) {
                maxIr.add(ppgIR[i]);
                checkIr=false;
            }
        } else {
            if(controlMaxorMin(ppgIR, i, false)) {
                minIr.add(ppgIR[i]);
                checkIr=true;
            }
        }
    }
    for(int i=0; i<minR.size() && i< maxR.size(); i++) {
        temp=(maxR.get(i)-minR.get(i));
        temp=temp/minR.get(i);
        ratioR.add(temp);
    }
    for(int i=0; i<minIr.size() && i< maxIr.size(); i++) {
        temp=(maxIr.get(i)-minIr.get(i));
        temp=temp/minIr.get(i);
        ratioIr.add(temp);
    }
    for(int i=0; i<ratioIr.size() && i<ratioR.size(); i++) {
        temp=ratioR.get(i)/ratioIr.get(i);
        temp = 104- (17*temp);
        ratioTot.add(temp);
    }
    int spo2= (int) Utility.meanEvaluation(ratioTot);
    return spo2;
}
```

Capitolo 8

Rete Neurale

Molte delle tecnologie innovative utilizzate in questi anni sono basate sui principi del machine learning e del deep learning. Queste branche di ricerca sono ormai utilizzate in molteplici ambiti, dalla medicina alla statistica, e permettono numerose soluzioni innovative. Il principio su cui sono fondate è l'apprendimento automatico da parte di un calcolatore, quale computer, smartphone o altri. Gli algoritmi di apprendimento consentono la costruzione di un modello matematico basato su dati campione, noti come "dati di formazione", al fine di fare previsioni o prendere decisioni senza essere esplicitamente programmati per eseguire il compito. Nel nostro caso parleremo di una rete neurale di tipo "deep" che, dato un elettrocardiogramma e un fotopletismogramma opportunamente rielaborati, riesce a calcolare i valori di pressione sistolica e diastolica. La rete neurale è stata precedentemente addestrata durante un'altra tesi [42]. In questo progetto abbiamo generalizzato il codice utilizzato per il testing, valutato la rete e trovato valori di correzione. Una volta ottenuti dei valori con un errore accettabile dal punto di vista medico, presente anche negli sfigmomanometri elettronici e certificati, sono state inserite delle funzioni per l'esportazione di tale rete ed è stata integrata all'interno dell'applicazione Android PulseEcg. Dato l'argomento così complesso e ampio, in questo capitolo ci soffermeremo solo sui concetti utilizzati per la rete neurale addestrata.

8.1 Concetti generali di apprendimento automatico

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" [43]

La definizione iniziale è stata data da Tom Mitchell e descrive in poche parole il concetto di apprendimento da parte di un software attraverso l'esperienza e i risultati raggiunti per un determinato obiettivo. Questo è il fondamento del machine learning. Gli algoritmi di questa branca sono implementati in modo tale che, ricevendo dei dati di varia natura in input, l'algoritmo cercherà di ricavare le informazioni significative necessarie per risolvere un determinato problema o per compiere una determinata azione. I compiti dell'apprendimento automatico vengono tipicamente classificati in tre ampie categorie, a seconda della natura dell'input [44]:

- apprendimento supervisionato: al modello vengono forniti degli "esempi" nella forma di possibili input e i rispettivi output attesi. L'obiettivo è quello di estrarre una regola generale che associ l'input all'output corretto;
- apprendimento non supervisionato: gli input non hanno una struttura ben definita ma deve essere l'algoritmo a creare un modello basandosi solo sulle loro caratteristiche;
- apprendimento per rinforzo: il modello interagisce con un ambiente dinamico nel quale cerca di raggiungere un obiettivo sapendo solo quando ha successo.

Un'altra categorizzazione dei compiti dell'apprendimento automatico si rileva quando si considera l'output desiderato:

- classificazione: solitamente supervisionato, gli output sono divisi in due o più classi e il sistema di apprendimento deve produrre un modello che assegni gli input non ancora visti a una o più di queste;
- regressione: anch'essa è un problema supervisionato ma l'output e il modello utilizzati sono continui;
- clustering: un insieme di input viene diviso in gruppi.

Il nostro problema può essere identificato come apprendimento automatico supervisionato e di regressione.

Rete neurale La rete neurale è un modello di apprendimento automatico ispirato dal funzionamento dei neuroni biologici all'interno del nostro cervello. Il modello ha una serie di componenti principali:

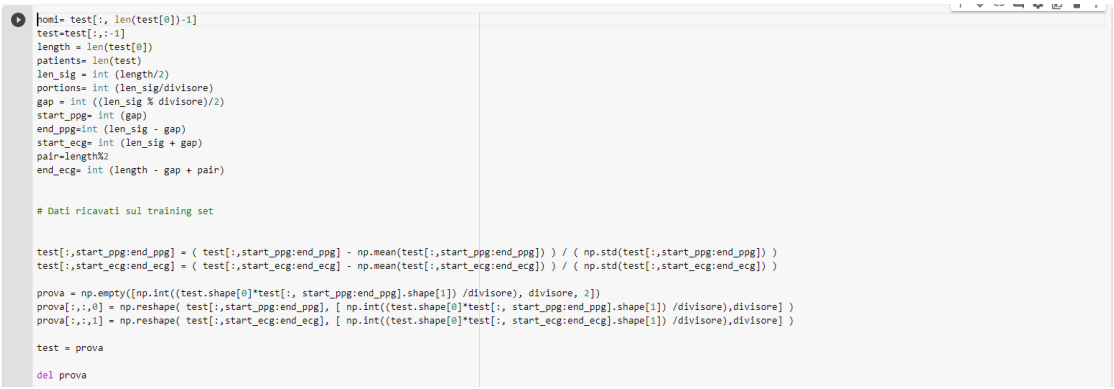
- neuroni: unità di base di elaborazione delle informazioni di una rete neurale. Essi ricevono un input, lo passano attraverso una funzione di attivazione, e producono un output utilizzando una funzione di uscita. La funzione di attivazione limita l'ampiezza dell'uscita del neurone e aggiunge una certa non linearità;
- architettura: definisce la struttura della rete, cioè il numero di neuroni artificiali nella rete e le loro interconnessioni;
- sinapsi: interconnessioni tra i vari neuroni;
- pesi: associati alle sinapsi e ai neuroni, i pesi memorizzano la conoscenza acquisita attraverso la fase di allenamento ed è grazie ad essi che riusciamo ad ottenere degli output specifici per il problema da risolvere;
- algoritmo di apprendimento: procedura utilizzata per eseguire il processo di apprendimento, che consiste nel modificare i pesi per compensare ogni errore trovato durante l'apprendimento. Uno dei metodi più popolari è basato sulla discesa del gradiente e si chiama algoritmo di backpropagation [45].

All'interno di questo progetto utilizziamo una rete neurale "deep", profonda. Nello specifico è stata implementata una rete neurale convoluzionale. La rete presenta infatti degli strati convoluzionali, ovvero con dei neuroni non interamente connessi a tutti i neuroni del livello successivo ma solo con connessioni specializzate. Questa architettura permette alla rete di concentrarsi su piccole caratteristiche di basso livello nel primo strato nascosto, per poi assemblarle in caratteristiche più grandi di livello superiore nel livello nascosto successivo. Il loro meccanismo portante sono i kernel convoluzionali (o filtri) che producono delle "feature map", che identificano i neuroni con gli stessi pesi [46].

8.2 Modifica e testing del modello in Python

La rete neurale implementata in Python prende come input un vettore di 1250 valori: 625 per il fotopletismogramma e 625 per l'elettrocardiogramma, entrambi i segnali con frequenza di 125Hz. L'output corrisponde invece ai valori di pressione sistolica e diastolica. La fase di testing implementata nel codice era programmata per essere provata con un dataset specifico e non prevedeva il funzionamento di input con un numero diverso di valori o che non avessero dei dati fin troppo specifici.

Per il testing della rete, è stato necessario ampliare il codice inserendo una parte di gestione dell'input di un dataset qualunque, purchè formato da una matrice di pazienti nelle righe e i segnali nelle colonne. Il codice mostrato in Fig. 8.1 mostra il pre-processamento dell'input prima della fase di testing.



```
homi= test[:, len(test[0])-1]
test=test[:,:-1]
length = len(test[0])
patients= len(test)
len_sig = int (length/2)
portions= int (len_sig/divisore)
gap = int ((len_sig % divisore)/2)
start_ppg= int (gap)
end_ppg=int (len_sig - gap)
start_ecg= int (len_sig + gap)
pair=length%2
end_ecg= int (length - gap + pair)


# Dati ricavati sul training set


test[:,start_ppg:end_ppg] = ( test[:,start_ppg:end_ppg] - np.mean(test[:,start_ppg:end_ppg]) ) / ( np.std(test[:,start_ppg:end_ppg]) )
test[:,start_ecg:end_ecg] = ( test[:,start_ecg:end_ecg] - np.mean(test[:,start_ecg:end_ecg]) ) / ( np.std(test[:,start_ecg:end_ecg]) )

prova = np.empty([np.int((test.shape[0]*test[:, start_ppg:end_ppg].shape[1]) /divisore), divisore, 2])
prova[:,:,0] = np.reshape( test[:,start_ppg:end_ppg], [ np.int((test.shape[0]*test[:, start_ppg:end_ppg].shape[1]) /divisore),divisore] )
prova[:,:,1] = np.reshape( test[:,start_ecg:end_ecg], [ np.int((test.shape[0]*test[:, start_ecg:end_ecg].shape[1]) /divisore),divisore] )

test = prova

del prova
```

Figura 8.1: Codice modificato di Python per elaborazione dati

Durante le prove con i vari dataset, sono emersi degli errori costanti. Per questo sono stati valutati e integrati dei fattori di correzione all'interno del codice.
Il discostamento che è stato riscontrato tra le misurazioni effettuate con diversi dispositivi, che potevano anch'essi contenere un errore, e i valori misurati dalla rete era accettabile.
Oltre questa modifica, il codice utilizzato si basava su Tensorflow versione 1.15. Questa versione di Tensorflow forniva come output del modello solo dei file nel formato .ckpt che non erano sufficienti per l'integrazione su Android. In una nuova porzione di codice, è stata perciò aggiunta la possibilità di ottenere i file ".pb" contenenti la descrizione della rete, e di conversione del modello nei seguenti formati: frozen_graph, saved_model e tflite_model. Per l'integrazione in Android, è stato utilizzato il modello Tensorflow Lite.

8.3 Pre-processamento dei dati in Android

Dato che la rete si aspetta uno specifico input, ovvero un vettore con 625 valori di fotopletismogramma e 625 di elettrocardiogramma entrambi a 125Hz, si è visto necessario elaborare il segnale in modo compatibile con la rete.

Ricampionamento Il primo passaggio riguarda il ricampionamento del segnale da 500Hz a 125Hz. Essendo comunque una frequenza multipla, l'operazione è risultata molto semplice da implementare in quanto corrisponde alla selezione di un valore ogni quattro elementi del vettore, come possiamo vedere semplicemente dal codice sottostante.

```
public static double[] resampling500to125(double[] signal) {
    double[] output = new double[signal.length/4];
    for(int i=0, j=0; i<signal.length; i+=4, j++) {
        output[j]= signal[i];
    }
    return output;
}
```

Nello specifico, con dieci secondi di acquisizione otterremo 1250 elementi di PPG e 1250 di ECG.

Divisione in batch Ottenuti i vettori dei segnali 125Hz, sarà necessario dividere questi valori in gruppi da 625 elementi e creare i vettori di input attesi dalla rete, mettendo insieme le porzioni di PPG ottenute con quelle dell'ECG. Il procedimento è abbastanza semplice e viene mostrato nel seguente codice:

```
public static ArrayList<float[]> checkBatches(double[] ecg, double[]
    ppg) {
    ArrayList<float[]> batches = new ArrayList<>();
    float[] temp = new float[1250];
    int numbat=0;
    for(int i=0, j=0; i<ppg.length;) {
        temp[j] = (float) ppg[i];
        i++;
        j++;
        if(i%625 == 0) {
            batches.add(temp);
            temp = new float[1300];
            j=0;
```

```
            numbat++;
        }
    }
    for(int i=0, j=625, k=0; i<ecg.length && k<numbat;) {
        batches.get(k)[j] = (float) ecg[i];
        i++;
        j++;
        if(i%625 == 0) {
            k++;
            j=625;
        }
    }
    return batches;
}
```

Nel nostro caso otterremo solamente due vettori dato che la dimensione iniziale dei vettori era 1250. Possiamo passare infine all'ultima fase, ovvero la normalizzazione.

Normalizzazione La normalizzazione effettuata deve basarsi, o essere uguale se possibile, su quella inizialmente utilizzata per addestrare la rete neurale. La normalizzazione è basata sulla media del segnale e sulla deviazione standard del segnale attraverso la formula 8.1.

$$Normalization = \frac{(Value - Mean)}{\sigma} \tag{8.1}$$

8.4 Implementazione rete neurale in Android

Per l'implementazione su Android sono state utilizzate le funzionalità dell'ultima versione di Tensorflow Lite. Inserendo il modello ottenuto dalla rete neurale in un formato del tipo ".tflite", Android sarà capace di gestire la rete e creare delle funzioni per il suo utilizzo.

Tensorflow Lite La libreria Tensorflow Lite [60] nasce come successore della libreria Tensorflow Mobile. Secondo Google, fornisce migliori prestazioni e una dimensione binaria più piccola grazie a kernel ottimizzati, attivazioni pre-fuse e meno dipendenze. Un modello generale Tensorflow pre-addestrato può essere in teoria convertito in formato .tflite e successivamente utilizzato per l'inferenza su Android o iOS. Il cambiamento del formato del file (.tflite invece di .pb) è causato dall'uso di una nuova libreria di serializzazione FlatBuffers che permette di accedere ai modelli salvati senza un passo di parsing/unpacking, spesso accoppiato

con l'allocazione di memoria per oggetto. Infine, la nuova libreria è compatibile con Android NNAPI e può funzionare di default con l'accelerazione hardware su dispositivi con chipset e driver appropriati [47].

Utilizzo della rete neurale in Android Inserendo all'interno del progetto il file contenente il modello Tensorflow Lite, Android Studio riesce ad identificare la rete e fornire una funzione di esempio, mostrata nella Fig. 8.2, per poter utilizzare la rete. Questo equivale all'istanziamento e all'utilizzo di un oggetto di tipo Interpreter.

Figura 8.2: Funzione di esempio per l'utilizzo del modello inserito e formulata automaticamente da Android Studio

Una volta elaborati i dati e inviati alla funzione apposita, la rete fornirà in output un numero di coppie di valori, che corrisponderanno alla pressione sistolica e alla diastolica, pari al numero di batch (gruppi da 1250 valori, precedentemente ricavati dai segnali). Nel nostro caso specifico, avremo due misure di pressione sistolica e due di diastolica. Verrà dunque fatta la media aritmetica di queste e verrà inserito il fattore di correzione prima di essere salvato e visualizzato. Nella Fig. 8.3 vediamo un esempio di pressioni sistolica e diastolica calcolate attraverso l'applicazione.

Figura 8.3: Esempio di pressioni sistolica e diastolica calcolate dall'applicazione PulseEcg

Capitolo 9

Testing

La fase di testing è stata, purtroppo, un'operazione difficile e non è stato possibile sottoporre ai test di prova un campione ampio di soggetti a causa delle restrizioni presenti all'interno delle regioni per lo stato di pandemia. La fase di testing si è dunque ridotta alle persone presenti nel laboratorio. All'interno della tesi, non verranno presentate le misurazione acquisite nel rispetto della privacy poichè contengono dati sensibili.

9.1 Test

I test sono stati effettuati confrontando i risultati ottenuti attraverso il PulseEcg con altre strumentazioni quali sfignomanometro elettronico, pulsossimetro ed elettrocardiografo. Si è potuto notare come l'applicazione riusciva ad ottenere valori simili, con un grado di errore compatibile, alle strumentazioni certificate. Bisogna tener conto che anche le strumentazioni hanno un loro grado di errore e il discostamento non molto significativo di una misurazione col bracciale rispetto alla strumentazione, non è una imprecisione dovuta necessariamente al PulseEcg.
Analizzando i dati, abbiamo notato in particolare:

Elettrocardiogramma Nonostante non si discosti molto dalla strumentazione certificata, il segnale risulta un po' più rumoroso e l'onda S meno pronunciata rispetto alla strumentazione. Essendo presente però solo un prototipo del bracciale, questi problemi potrebbero essere dovuti a minime variazioni presenti nell'hardware o nell'ambiente di lavoro. Per quanto riguarda i valori di frequenza cardiaca, essi risultano abbastanza precisi, con un discostamento di ± 5 battiti per minuto.

Fotopletismografia Il segnale fotopletismografico ottenuto con i sensori rosso e infrarosso è purtroppo sensibile alle condizioni ambientali che possono esserci con il sensore. Esso, infatti, può essere influenzato dalla presenza della luce esterna ed è un problema globalmente noto, anche per i classici pulsossimetri. Nonostante questo, grazie al filtraggio software integrato, l'andamento del segnale risulta abbastanza chiaro e significativo, soprattutto dopo i primi 400 millisecondi. Il valore di saturazione dell'ossigeno nel sangue calcolato con questo metodo risente anch'esso lievemente del rumore e può variare tra $\pm 2\%$ rispetto al valore calcolato con il pulsossimetro.

Pressione I valori di pressione differiscono dipendentemente dalla strumentazione certificata di riferimento. La pressione sistolica presenta un discostamento dalla strumentazione in un intervallo che varia tra ± 10mmHg. La pressione diastolica differiva invece di ± 7mmHg. Si noti come le misurazioni sono state effettuate anche con diverse strumentazioni certificate che presentavano valori diversi anche tra loro (il massimo discostamento rilevato tra questi è stato di 12mmHg). Questo perchè, come già detto, la strumentazione presenta un errore interno e, inoltre, le misure non possono essere prese con una totale sincronia e i valori possono cambiare in base a diversi fattori, quali respirazione o posizione.

9.2 Commenti

Nonostante la fase di testing sia stata molto ridotta, i risultati ottenuti sono compatibili con le strumentazioni mediche ufficiali. Purtroppo le misurazioni sono sensibili a fattori esterni quali, ad esempio:

- la presenza di luce nell'ambiente;
- la pressione e la posizione del dito sull'elettrodo e sul sensore;
- la posizione dell'utente;
- la presenza di apparecchiature di disturbo all'interno dell'ambiente;
- il valore di carica della batteria.

Vi è quindi certamente un margine di miglioramento sotto questi punti di vista. Per quanto riguarda l'applicazione Android, si potrebbero valutare dei filtraggi che permettano di ridurre maggiormente il rumore senza avere una perdita di informazione significativa sul segnale. I valori di pressione arteriosa risultano accettabili ma il test è stato fatto solo su pochi soggetti, nessuno dei quali riscontrava patologie o anomalie. Ampliando la fase di testing, potrebbe essere valutato un riaddestramento della rete con dataset più ampi e comprendenti situazioni più complesse.

Capitolo 10

Conclusioni

Grazie all'applicazione è stato possibile sviluppare un sistema di controllo e di prevenzione per tre caratteristiche mediche fondamentali: elettrocardiogramma, fotopletismogramma e pressione arteriosa. Ottenere dei valori così rilevanti in modo così semplice è un passo estremamente importante per lo sviluppo tecnologico in ambito medico. Con delle tecnologie ancor più avanzate e precise, si potrebbero sostituire alcune delle tante misurazioni invasive che possono essere effettuate sul nostro corpo. Inoltre, avere sempre a portata di mano un oggetto che controlla la propria salute, aiuterebbe nella prevenzione delle malattie, riuscendo a incorrere ad un peggioramento graduale o istantaneo dello stato fisico.
Non bisogna però sottovalutare l'importanza del medico: se l'applicazione riesce ad individuare una possibile anomalia, questa deve essere valutata e confermata da uno specialista e sarà sempre lui a fornire un trattamento o ulteriori esami per confermare o confutare la misurazione del dispositivo. La tecnologia non deve quindi sostituire il ruolo del medico ma essere un aiuto, un supporto.
In EcgWatch e PulseEcg questo è stato messo in luce ed evidenziato con la possibilità di condividere i risultati con il medico curante e con la ricezione delle notifiche. Inoltre, le applicazioni non propongono nessuna informazione per quanto riguarda un possibile trattamento, che deve essere fornito solo dal medico.

10.1 Sviluppi futuri

Gli sviluppi di questo dispositivo con l'applicazione possono essere molteplici.
Per quanto riguarda il lato software attuale, potrebbe essere valutato l'utilizzo di Tensorflow Mobile [48], versione più vecchia e con funzioni non innovative ma con un maggiore supporto che magari apporterebbe dei miglioramenti sulle prestazioni e sulla velocità di calcolo. Tensorflow Lite subirà invece degli aggiornamenti negli anni che lo porteranno ad essere probabilmente la libreria più innovativa e con più supporto nell'ambito di machine learning in Android. Sarà dunque necessario tenere sotto controllo tali aggiornamenti e utilizzarli all'interno dell'applicazione.
Potrebbero essere rivisti i filtri dei segnali e provare ad ottenere, se possibile, un segnale senza rumore mantenendo la precisione dello stesso, magari lavorando anche sul lato hardware.
Si potrebbe inoltre valutare, sul lato hardware, l'utilizzo di una soluzione alternativa ai segnali rosso e infrarosso, in quanto, data l'importanza attuale dei pulsossimetri, sono in fase di progettazione diverse soluzioni come il doppio infrarosso [49] o la luce verde [50].
Sarebbe possibile integrare un sistema di monitoraggio senza la necessità di premere un pulsante sull'applicazione ma semplicemente poggiando il dito sul sensore, acquisendo così i valori e mostrandoli direttamente sul dispositivo.
Per quanto riguarda le innovazioni che sarà possibile inserire si può pensare di aumentare il numero di sensori all'interno del dispositivo per incrementare il numero di caratteristiche utili per la prevenzione monitorate dall'applicazione. Un sensore di temperatura [51] permetterebbe di monitorare il grado di calore corporeo dell'utente. Un accelerometro permetterebbe di rilevare dei movimenti bruschi in corrispondenza di misurazioni anomale da parte degli altri sensori, notificando a un medico o ad una persona di fiducia una situazione di attenzione, magari inviando la posizione dell'utente, già gestita all'interno dell'applicazione.
Le possibilità di sviluppo sono dunque molte e continueranno ad aumentare negli anni con lo sviluppo di ulteriori tecnologie sempre più piccole ed efficienti.

Bibliografia

[1] Victor Dzau Daniel Shu Wei Ting Lawrence Carin e Tien Y. Wong. «Digital technology and COVID-19». In: 26 (apr. 2020), pp. 458–464 (cit. a p. 1).

[2] X. Ding et al. «Wearable Sensing and Telehealth Technology with Potential Applications in the Coronavirus Pandemic». In: *IEEE Reviews in Biomedical Engineering* 14 (2021), pp. 48–70. DOI: 10.1109/RBME.2020.2992838 (cit. a p. 1).

[3] Ary L. Goldberger, Zachary D. Goldberger e Alexei Shvilkin. «Chapter 1 - Key Concepts». In: *Goldberger's Clinical Electrocardiography (Eighth Edition)*. A cura di Ary L. Goldberger, Zachary D. Goldberger e Alexei Shvilkin. Eighth Edition. Philadelphia: W.B. Saunders, 2013, pp. 2–4. ISBN: 978-0-323-08786-5. DOI: https://doi.org/10.1016/B978-0-323-08786-5.00001-4. URL: https://www.sciencedirect.com/science/article/pii/B9780323087865000014 (cit. a p. 5).

[4] Ary L. Goldberger, Zachary D. Goldberger e Alexei Shvilkin. «Chapter 2 - ECG Basics: Waves, Intervals, and Segments». In: *Goldberger's Clinical Electrocardiography (Eighth Edition)*. A cura di Ary L. Goldberger, Zachary D. Goldberger e Alexei Shvilkin. Eighth Edition. Philadelphia: W.B. Saunders, 2013, pp. 5–15. ISBN: 978-0-323-08786-5. DOI: https://doi.org/10.1016/B978-0-323-08786-5.00002-6. URL: https://www.sciencedirect.com/science/article/pii/B9780323087865000026 (cit. alle pp. 7, 9).

[5] Ary L. Goldberger, Zachary D. Goldberger e Alexei Shvilkin. «Chapter 3 - ECG Leads». In: *Goldberger's Clinical Electrocardiography (Eighth Edition)*. A cura di Ary L. Goldberger, Zachary D. Goldberger e Alexei Shvilkin. Eighth Edition. Philadelphia: W.B. Saunders, 2013, pp. 16–25. ISBN: 978-0-323-08786-5. DOI: https://doi.org/10.1016/B978-0-323-08786-5.00003-8. URL: https://www.sciencedirect.com/science/article/pii/B9780323087865000038 (cit. a p. 8).

[6] E. Vallaperta G. Mazzone M. Girlanda. «Fibrillazione atriale». In: *Italian Journal of Emergency Medicine* (mag. 2013) (cit. a p. 9).

[7] D. Battigelli O. Brignoli G. Ermini A. Filippi B. Guillaro S. E. Giustini. «Fibrillazione Atriale in Medicina Generale». In: *Disease Management Società Italiana di Medicina Generale* (2013) (cit. a p. 10).

[8] S. Cheriyedath. «Photoplethysmography (PPG)». In: *News-Medical* (mar. 2021). URL: https://www.news-medical.net/health/Photoplethysmography-(PPG).aspx (cit. a p. 11).

[9] John Allen. «Photoplethysmography and its application in clinical physiological measurement». In: *Physiological Measurement* 28 (apr. 2007), R1–39. DOI: 10.1088/0967-3334/28/3/R01 (cit. a p. 12).

[10] Dorairaji Ramirez Agustinl Schlaich Markusm Stergiou George Tomaszewski Maciejp Wainford Richard Williams Bryanu Schutte Aletta Unger Thomasa Borghi Claudiob Charchar Fadic Khan Nadia Poulter Neil Prabhakaran. «2020 International Society of Hypertension global hypertension practice guidelines». In: *Journal of Hypertension* 38 (giu. 2020). DOI: 10.1097/HJH.0000000000002453 (cit. a p. 14).

[11] «High Blood Pressure». In: (2020). URL: https://www.nhlbi.nih.gov/health-topics/high-blood-pressure (cit. a p. 14).

[12] F. Caffarelli. «Wearable Bluetooth ECG Sensor For Biometric Identification». Tesi di laurea mag. Torino: Politecnico di Torino, apr. 2014 (cit. alle pp. 17, 32, 40, 61, 83–85, 89).

[13] D. Motta. «Design and Development of a Multi-Parameter Wearable Medical Device». Tesi di laurea mag. Torino: Politecnico di Torino, 2020 (cit. a p. 20).

[14] «IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.1a: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPAN)». In: *IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002)* (2005), pp. 1–700. DOI: 10.1109/IEEESTD.2005.96290 (cit. a p. 21).

[15] C. Bisdikian. «An overview of the Bluetooth wireless technology». In: *IEEE Communications Magazine* 39.12 (2001), pp. 86–94. DOI: 10.1109/35.968817 (cit. a p. 21).

[16] P. McDermott-Wells. «What is Bluetooth?» In: *IEEE Potentials* 23.5 (2005), pp. 33–35. DOI: 10.1109/MP.2005.1368913 (cit. a p. 21).

[17] Carles Gomez, Joaquim Oller e Josep Paradells. «Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology». In: *Sensors* 12.9 (2012), pp. 11734–11753. ISSN: 1424-8220. DOI: 10.3390/s120911734. URL: https://www.mdpi.com/1424-8220/12/9/11734 (cit. a p. 22).

[18] C. Hoffman. «Android is Based on Linux, But What Does That Mean?» In: *How-To Geek* (mag. 2014) (cit. a p. 24).

[19] R. Stallman. «Is Android really free software?» In: *The Guardian* (set. 2011) (cit. a p. 24).

[20] R. Brandom. «There are now 2.5 billion active Android devices». In: *The Verge* (mag. 2019) (cit. a p. 24).

[21] C. Romano. «Android 5.0: addio Dalvik, ART sarà la runtime di default nel prossimo major update». In: *International Business Times* (giu. 2014) (cit. a p. 25).

[22] *Android Developers: ART and Dalvik*. Ott. 2020 (cit. a p. 25).

[23] P. Jahoda. *MPAndroidChart*. URL: `https://github.com/PhilJay/MPAndroidChart` (cit. a p. 27).

[24] J. Gehring. *GraphView*. URL: `https://github.com/jjoe64/GraphView` (cit. a p. 27).

[25] Google. *Overview of Google Play services*. URL: `https://developers.google.com/android/guides/overview` (cit. a p. 28).

[26] L.Santaniello. «Parsing JSON semplice con google GSON». In: *HTML.it* (mar. 2012) (cit. a p. 28).

[27] a. jaokar. «TensorFlow 1.x vs 2.x. – summary of changes». In: *Data Science Central* (set. 2019) (cit. a p. 31).

[28] Material Design. *Dark Theme*. URL: `https://material.io/design/color/dark-theme.html` (cit. a p. 39).

[29] K. Błaszkiewicz M. Eibes B. Trendafilov C. Montag A. Markowetz I. Andone. «How age and gender affect smartphone usage». In: *UbiComp '16: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct* (set. 2016), pp. 9–12. DOI: `https://doi.org/10.1145/2968219.2971451` (cit. a p. 40).

[30] I. Kaur Rajni. «Electrocardiogram Signal Analysis - An Overview». In: *International Journal of Computer Applications* 84 (dic. 2013) (cit. a p. 56).

[31] S. Stabellini. «Interpretazione dell'elettrocardiogramma nell'adulto». In: *Nurse24* (mag. 2017) (cit. a p. 59).

[32] Bruce Hopkins e Ranjith Antony. *Bluetooth for Java*. APress L. P., 2003. ISBN: 1590590783 (cit. a p. 61).

[33] Ery Arias-Castro e David L. Donoho. «Does median filtering truly preserve edges better than linear filtering?» In: *The Annals of Statistics* 37.3 (2009), pp. 1172–1206. DOI: `10.1214/08-AOS604`. URL: `https://doi.org/10.1214/08-AOS604` (cit. a p. 84).

[34] R. W. Schafer. «What Is a Savitzky-Golay Filter? [Lecture Notes]». In: *IEEE Signal Processing Magazine* 28.4 (2011), pp. 111–117. DOI: 10.1109/MSP.2011.941097 (cit. a p. 85).

[35] Siti Farah Hussin, Zunainah Hamid e Gauri. «Design of Butterworth Band-Pass Filter». In: (gen. 2016), pp. 128–2883 (cit. a p. 86).

[36] Yu C Liu Z McKenna T Reisner AT Reifman J. «A method for automatic identification of reliable heart rates calculated from ECG and PPG waveforms.» In: *J Am Med Inform Assoc.* (2006). DOI: 10.1197/jamia.M1925 (cit. a p. 88).

[37] S. Chernenko. *ECG processing — R-peaks detection.* URL: http://www.librow.com/articles/article-13 (cit. a p. 89).

[38] Willis J. Pan Jiapu; Tompkins. «A Real-Time QRS Detection Algorithm». In: *IEEE Transactions on Biomedical Engineering* (mar. 1985). DOI: 10.1109/TBME.1985.325532.PMID (cit. alle pp. 89, 90).

[39] M. A. Z. Fariha, R. Ikeura, S. Hayakawa e S. Tsutsumi. «Analysis of Pan-Tompkins Algorithm Performance with Noisy ECG Signals». In: *Journal of Physics: Conference Series* 1532 (giu. 2020), p. 012022. DOI: 10.1088/1742-6596/1532/1/012022. URL: https://doi.org/10.1088/1742-6596/1532/1/012022 (cit. a p. 90).

[40] B. M. Schiffman e G. L. Matthaei. «Exact Design of Band-Stop Microwave Filters». In: *IEEE Transactions on Microwave Theory and Techniques* 12.1 (1964), pp. 6–15. DOI: 10.1109/TMTT.1964.1125744 (cit. a p. 93).

[41] «Principles of pulse oximetry». In: *Anaesthesia UK* (set. 2004) (cit. a p. 97).

[42] s. Villata. «Cuffless Blood Pressure Measurement». Tesi di laurea mag. Torino: Politecnico di Torino, set. 2014 (cit. a p. 99).

[43] T. Mitchell. «Machine Learning». In: *McGraw Hill* (1997) (cit. a p. 100).

[44] Xian-Da Zhang. «Machine Learning». In: *A Matrix Algebra Approach to Artificial Intelligence*. Singapore: Springer Singapore, 2020, pp. 223–440. DOI: 10.1007/978-981-15-2770-8_6. URL: https://doi.org/10.1007/978-981-15-2770-8_6 (cit. a p. 100).

[45] Raúl Rojas. «The Backpropagation Algorithm». In: *Neural Networks: A Systematic Introduction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 149–182. ISBN: 978-3-642-61068-4. DOI: 10.1007/978-3-642-61068-4_7. URL: https://doi.org/10.1007/978-3-642-61068-4_7 (cit. a p. 101).

[46] A. Courville Y. Bengio I. Goodfellow. «Convolutional Network». In: *Deep Learning*. Ott. 2015 (cit. a p. 101).

[47] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley e Luc Van Gool. «AI Benchmark: Running Deep Neural Networks on Android Smartphones». In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. Set. 2018 (cit. a p. 105).

[48] R. Gour. *The Essential Guide To Learn TensorFlow Mobile and Tensorflow Lite*. Apr. 2019. URL: https://towardsdatascience.com/the-essential-guide-to-learn-tensorflow-mobile-and-tensorflow-lite-a70591687800 (cit. a p. 109).

[49] Nitzan I Yossef Hay O Cohen M. «Pulse Oximetry with Two Infrared Wavelengths without Calibration in Extracted Arterial Blood». In: *Sensors (Basel)* (ott. 2018). DOI: 10.3390/s18103457 (cit. a p. 109).

[50] Verkruysse W. Moço A. «Pulse oximetry based on photoplethysmography imaging with red and green light : Calibratability and challenges». In: *J Clin Monit Comput* (2020). DOI: 10.1007/s10877-019-00449-y (cit. a p. 109).

[51] Toshiyo Tamura, Ming Huang e Tatsuo Togawa. «Current Developments in Wearable Thermometers». In: *Advanced Biomedical Engineering* 7 (apr. 2018), pp. 88–99. DOI: 10.14326/abe.7.88 (cit. a p. 109).