ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

IL SISTEMA TRAUMA TRACKER - INDIVIDUAZIONE E ANALISI DI PARAMETRI VITALI ACQUISITI DA MONITOR MULTIPARAMETRICO.

Elaborato in
PROGRAMMAZIONE DI SISTEMI EMBEDDED

Relatore
Prof. ALESSANDRO RICCI

Co-relatore
Prof. ALESSANDRO BEVILACQUA

Presentata da
FEDERICO BELLINI

Seconda Sessione di Laurea
Anno Accademico 2015 – 2016

PAROLE CHIAVE

Trauma Tracker

Healthcare

Smart Glass

IoT Hands-Free System

Embedded Technology

Dedicata a tutti coloro
che mi hanno sempre sostenuto nelle scelte
e aiutato a portare a compimento
questo mio grande cammino.

Indice

Elenco delle figure

Introduzione

Lo scopo della presente tesi è lo studio e la progettazione di un sistema *Hands-Free* applicato in una realtà *Healthcare*, volto ad aiutare il personale sanitario nello svolgimento delle mansioni lavorative. Questo progetto, denominato Trauma Tracker, ha avuto origine grazie alla collaborazione con medici ed infermieri dell'ospedale Maurizio Bufalini di Cesena. La necessità dell'informatizzazione di alcune loro procedure ha portato alla cooperazione dell'ospedale con la nostra università di Ingegneria e Scienze Informatiche di Cesena. In particolare la loro esigenza è sorta a causa di prolungati disagi sviluppatisi nell'ambito del Pronto Soccorso. La normale procedura che finora è stata adottata per ogni paziente che si reca in ospedale per operazioni d'urgenza, è la seguente: vengono applicati tutti i provvedimenti necessari per far sì che il degente non sia più in pericolo di vita e, solo alla fine quando risulta stabilizzato, viene stilato un rapporto contenente le informazioni su ogni singola manovra eseguita. Oggi tutto questo viene fatto a memoria dai medici, cercando di ricordare tutte le operazioni compiute. Siccome le procedure di stabilizzazione possono prolungarsi per molto tempo, la ricostruzione del report finale può risultare difficile, ed è chiaro come una dimenticanza in fase di scrittura di tale documento può risultare disastrosa. Per questo motivo l'obiettivo degli operatori sanitari del Pronto Soccorso è di informatizzare queste procedure nel più breve tempo possibile, riducendo notevolmente le possibilità di errori dovuti a fattori umani. Il sistema Trauma Tracker è nato da queste esigenze, ed è stato sviluppato tenendo in considerazione la necessità degli operatori di avere sempre le mani libere. Unendo questi requisiti, ed aggiungendo ad esso qualche funzionalità in più, si può capire come questo progetto possa perfettamente essere ideato come un sistema *wearable* applicato all'*healthcare*.

Trauma Tracker, almeno in queste prime fasi, non si propone come uno strumento immediatamente utilizzabile sul campo e pronto ad affiancare i medici, poiché necessiterebbe subito di qualità come robustezza, affidabilità e molte altre a livelli estremamente elevati. Per questo motivo il progetto è stato trattato come un *Proof of Concept*, ossia un prototipo che ha lo scopo di dimostrare la fattibilità di tale sistema nella realtà, e di verificarne l'utilità una volta applicato in uno scenario vero.
L'argomento trattato ha quindi una grande importanza nel mondo reale, poiché getta le basi di una tecnologia che un giorno potrà aiutare medici ed infermieri a svolgere al meglio l'impegnativo compito di salvare vite.

La tesi si compone di cinque capitoli più uno finale, volto a presentare test e dimostrazioni del sistema Trauma Tracker concepito. Nel primo capitolo verrà esposta una breve panoramica sullo stato dell'arte dei sistemi *hands-free* in generale e *hands-free* applicati in ambito *healthcare*, introducendo anche elementi relativi alle tecnologie *wearable*. Nel secondo capitolo si procede esaminando le specifiche del progetto Trauma Tracker, per poi analizzarne i requisiti software ed hardware nel capitolo successivo. Nel quarto capitolo l'analisi si concentra sulla progettazione software del sistema e nel quinto capitolo ci si occupa invece di presentare le principali caratteristiche implementative del sottosistema impiegato nel riconoscimento dei parametri.

Capitolo 1

Background

In questo capitolo verranno presentati e approfonditi i più alti livelli di sviluppo finora raggiunti nei campi scientifici e nelle tecnologie di cui il progetto Trauma-Tracker si compone.

1.1 Wearable Technologies e IoT

Le *Wearable Technologies* o tecnologie indossabili e i Sistemi Integrati hanno dimostrato, nel corso degli ultimi decenni, significativi progressi in termini di miniaturizzazione, integrazione, funzionalità, comfort, elaborazione dei dati e comunicazioni. Tutto ciò è dovuto principalmente all'enorme progresso delle scienze e delle micro-tecnologie, ma soprattutto da una crescente domanda di smartphone e in generale di smart-objects sempre più piccoli e preformanti. Gli imponenti investimenti in ricerca e sviluppo compiuti dai grandi produttori di cellulari hanno infatti contribuito in gran parte alla miniaturizzazione delle tecnologie, che con la sua enorme richiesta di dispositivi di ridotte dimensioni e con l'esigenza di offrire sempre più nuove e complesse funzionalità, necessitano dell'investimento di molto denaro per rimanere competitivi sul mercato. Basti pensare agli ultimi hardware montati sui cellulari oggi in commercio per capire quanto le cose siano cambiate negli ultimi anni: touch screen sensibili alla pressione esercitata, riconoscimento delle impronte digitali o ricarica contactless. Anche i costi di produzione si sono abbassati notevolmente, sempre a causa della fabbricazione in vasta scala di smart-objects. Proprio grazie a queste condizioni di sviluppo favorevoli, oggi il settore delle tecnologie indossabili e

degli oggetti così detti "intelligenti" è davvero ricco e fiorente. Così l'*Internet of Things* sta diventando un argomento di conversazione e dibattito sempre più crescente e acceso. E' un concetto che non solo ha il potenziale cambiare il modo in cui lavoriamo, ma anche quello in cui viviamo. Ma che cosa è esattamente l'Internet delle cose, e che impatto sta avendo su di noi? Internet è diventato sempre più disponibile, ed il costo del collegamento è in diminuzione, rendendo così l'accesso disponibile pressoché a chiunque. Inoltre tutti i nuovi dispositivi tecnologici, dalle fotocamere alle televisioni, vengono creati con la possibilità di collegarsi alla rete. Tutti questi fattori stanno influenzando lo sviluppo di questi sistemi, dando origine a una "tempesta perfetta" per lo sviluppo dell'*IoT*. Già da molti anni sono chiare le enormi possibilità che può offrire una connessione globale di oggetti e persone. Nikola Tesla nel 1926 disse:

> *"When wireless is perfectly applied the whole earth will be converted into a huge brain, which in fact it is, all things being particles of a real and rhythmic whole. We shall be able to communicate with one another instantly, irrespective of distance. Not only this, but through television and telephony we shall see and hear one another as perfectly as though we were face to face, despite intervening distances of thousands of miles; and the instruments through which we shall be able to do this will be amazingly simple compared with our present telephone. A man will be able to carry one in his vest pocket."* [1]
>
> (Nikola Tesla, 1926)

L'*Internet delle cose* è comunque da considerare qualcosa che va oltre il semplice ambito mobile, esso aggiunge un livello quasi magico agli oggetti di uso quotidiano, che grazie alle emergenti tecnologie di comunicazione, localizzazione, sensori integrati e alle possibilità di interazioni real-time, vengono trasformati in oggetti ritenuti davvero intelligenti, in grado di comprendere, reagire e comunicare con l'ambiente a loro vicino. Tali oggetti vengono considerati come i mattoni fondamentali costituenti le basi per l'*IoT* [2]. La definizione che meglio descrive questi oggetti è stata riferita da Nicholas Negroponte, fondatore dell'MIT Media-Lab:

"When we talk about an Internet of things, it's not just putting RFID tags on some dumb things so we smart people know where that dumb thing is. It's about embedding intelligence so things become smarter and do more than they were proposed to do."

(Nicholas Negroponte)

1.1.1 Breve panoramica su sistemi in Realtà Aumentata, Virtuale ed Eyewear Computing

Le tecnologie in realtà aumentata e virtuale si stanno rapidamente sviluppando in questi ultimi anni, e stanno poco alla volta entrando a far parte sempre più delle nostre vite. Questo trend è favorito da grossi investimenti di giganti come Google, Microsoft o Sony che stanno scommettendo nel futuro di queste tecnologie. Un ulteriore spinta a questo settore è stata data anche dalla notevole diminuzione del costo di questi accessori, espandendo così il loro possibile mercato. Le aziende impegnate in questa rivoluzione della realtà aumentata, stanno gettando le basi per il raggiungimento di un obiettivo ambizioso: evolvere le modalità di fruizione degli elementi del mondo reale, arricchendo il campo visivo dell'utente con nuovi input e stimoli. Sviluppata dapprima in campo militare, come quasi tutte le tecnologie che si sono poi rivelate di fondamentale importanza per lo sviluppo tecnologico, la realtà aumentata viene applicata oggi nel marketing, nella pubblicità, nei videogiochi e ora anche nel mondo del business. Per realtà aumentata o *augmented reality*, si intende l'arricchimento della percezione sensoriale umana mediante informazioni, in genere manipolate e convogliate elettronicamente, che non sarebbero percepibili con i cinque sensi [3]. Molti software di *augmented reality* sono oggi alla portata di chiunque, grazie alle numerose applicazioni per smartphone che si possono trovare gratuitamente negli store. La maggior parte di esse sono applicate in ambito ludico, e tuttora non sono troppo diffuse quelle nate per affiancare l'uomo aiutandolo nelle sue mansioni. Sono invece differenti gli scopi per cui è sorta la realtà virtuale. Essa è nata come realtà totalmente simulata, dove le informazioni presenti sullo schermo sono volte ad ingannare i sensi, senza presentare alcun elemento della vera realtà, con lo scopo di distogliere l'utente dalla percezione del mondo reale e facendogli provare la sensazione di

trovarsi immersi in nuove situazioni, totalmente ricostruite al computer.
La differenza fondamentale tra realtà aumentata e virtuale sta nel concetto e nel livello di simulazione utilizzato. La realtà virtuale risulta essere molto più immersiva rispetto a quella aumentata, poiché tale realtà è completamente generata dal computer. La realtà aumentata invece non è volta ad ingannare l'utente, ma anzi a fornirgli informazioni aggiuntive sul mondo che lo circonda, talvolta per aiutarlo a comprenderlo meglio.
Nel progetto Trauma Tracker che si andrà a sviluppare, non sarà applicato però il vero e proprio concetto di realtà aumentata, poiché gli elementi che verranno proiettati sugli smart-glass non saranno solidali con la realtà tangibile, ma ne saranno disaccoppiati. E' infatti più corretto definirlo come un *"Eyewear Computing"*, ossia un sistema che, a differenza di quelli tradizionali, è sempre presente con l'utente, e ha il potenziale di percepire il mondo dal suo punto di vista. *"Eyewear computing has the potential to fundamentally transform the way machines perceive and understand the world around us and to assist humans in measurably and significantly improved ways." [4]* Questa è la definizione che è stata data originariamente a questa nuova classificazione dei sistemi. In poche parole si tratta di un potenziamento volto a migliorare le percezioni dell'individuo, ed è proprio questo il fattore chiave che ci ha condotti all'ideazione, progettazione e sviluppo del sistema TraumaTracker.

1.2 Sistemi Hands-Free

Una branca dell'*IoT* e delle tecnologie indossabili si occupa dei sistemi *Hands-Free*, ovvero di tutte quelle apparecchiature intelligenti che possono essere usufruite senza l'avvalersi delle mani, ad esempio tramite comandi vocali. Le tecnologie a mani libere sono probabilmente una delle invenzioni più utili al giorno d'oggi. Basti pensare al largo utilizzo che se ne fa alla guida di un veicolo: le case automobilistiche predispongono tutti i modelli moderni con sistemi per la comunicazione diretta con lo smartphone dell'utente, che così può comodamente esser lasciato in tasca o nella borsa, senza mai la necessità di maneggiarlo alla guida, permettendo comunque di scrivere messaggi, inoltrare mail, cambiare la musica alla radio o rispondere alle telefonate.
Le tecnologie hands-free sono oggi applicate nei più svariati ambiti, dagli as-

sistenti vocali in computer e smartphone, al mondo del gioco, come hanno già sperimentato alcune importanti aziende che da qualche anno hanno integrato sistemi di riconoscimento vocale e dei movimenti nelle console di ultima generazione, portando così il semplice gioco su di un altro livello, sempre più realistico e coinvolgente, facendo mangiare la polvere ai vecchi controller manuali [5].
L'importanza di questi sistemi è quindi indiscutibile, sopratutto se si pensa agli enormi benefici che si possono avere applicando queste nuove tecnologie in campi come la salute delle persone. La prossima sezione introdurrà le ultime tecnologie *hands-free* applicate nel settore ospedaliero, e ne esporrà i più avanzati sviluppi tecnologici.

1.3 Hands-Free applicato in realtà Healthcare

In ambito *Healthcare*, le tecnologie *Hands-Free* stanno facendo la loro comparsa solamente in questi ultimi anni. Quelle poche che per il momento sono già attive e in funzione, sono essenzialmente dei prototipi in fase di testing. Le difficoltà nel creare sistemi perfettamente funzionanti in ogni condizione, che non devono mai generare errori o rallentamenti, sommati ai possibili problemi di sicurezza dei dati dei pazienti, sono probabilmente la causa della nascita tardiva di questi sistemi rispetto agli altri campi di applicazione. Tutto ciò non vuol dire però che aziende di grosso calibro non stiano pensando di tuffarsi in questo settore o che non l'abbiano già fatto. Molti chirurghi e altri professionisti infatti hanno già intravisto l'enorme potenziale che si nasconde nell'avere le mani sempre libere ed essere comunque connessi alla rete, usufruendo dei suoi servizi. Così aziende e ospedali si sono attivati, cercando di stabilirsi come leader di questo nuovo mercato ancora inviolato [6][7][8].
Diversi prototipi e sperimentazioni di applicazioni hands-free in ambito medico sono già state compiute, sia dal lato dei pazienti per aiutarli nei loro compiti, come per esempio permettendo a disabili di guidare la propria sedia a rotelle riconoscendo gesti fatti con la testa [9], sia da quello di medici ed infermieri, fornendogli per esempio la possibilità di eseguire rapidi test ed analisi quantitative e qualitative [10], o di essere aiutati durante operazioni chirurgiche [11].
Il progetto da noi sviluppato e analizzato più nel dettaglio nel seguente capito-

lo, si compone delle qualità hands-free applicate in ambito ospedaliero appena introdotte, ed è volto ad aiutare medici, chirurghi ed infermieri a svolgere i loro compiti lavorativi quotidiani.

Capitolo 2

Caso di studio

Come detto in precedenza, l'utilizzo di sistemi *Hands-Free* in ambito *Healthcare* può sicuramente apportare grandi benefici per chi si avvale di tali tecnologie. Grazie alla recente diffusione di smart-glass sempre più economici, piccoli e con prestazioni in costante aumento, è oggi possibile avvalersi di essi per l'invenzione di sistemi *Hands-Free* utili in ambito medico. Alcune possibili applicazioni di questi occhiali possono essere per esempio la telemedicina, in cui medici possono aiutare a distanza pazienti in situazioni di emergenza; la dettatura virtuale, che può aiutare a raccogliere e collezionare informazioni in situazioni in cui è richiesto l'utilizzo sempre libero delle mani; forme di realtà aumentata, utili per avere accesso più rapido alle informazioni, e tante altre ancora[12]. Sulla base di queste innovazioni è nato il progetto **Trauma-Tracker**, che di seguito verrà presentato.

2.1 Trauma-Tracker Proof of Concept

Il progetto Trauma-Tracker è un'idea nata nel Giugno 2016, con la collaborazione dell'ospedale Maurizio Bufalini di Cesena. Tutto ha avuto inizio quando i responsabili dell'ospedale si sono rivolti all'Università di Ingegneria e Scienze Informatiche di Cesena, per capire se fosse possibile la realizzazione di un dispositivo capace di registrare autonomamente le operazioni eseguite dal Trauma-Leader nell'ambito del Pronto Soccorso. In pratica finora, quando un paziente si reca in Pronto Soccorso, vengono applicati tutti i provvedimenti necessari per renderlo stabile e, solamente alla fine di tutto il procedimento,

il coordinatore del team, il così detto "Trauma-Leader", ha la responsabilità di compilare un rapporto con tutte le singole operazioni svolte sul degente. Siccome lo svolgimento di tutte le manovre necessarie alla stabilizzazione può richiedere anche lunghi periodi di tempo, ricordarsi a memoria precisamente ogni procedura compiuta può essere difficile, e gli errori dovuti al fattore umano possono risultare disastrosi. Siccome il Trauma-Leader necessita sempre di avere le mani libere, è evidente come l'applicazione di una tecnologia Hands-Free si adatti perfettamente in questo frangente.
La realizzazione del progetto Trauma-Tracker, è per ora stata concepita come un *Proof of Concept*, ossia come un sistema volto a dimostrare in modo pratico la fattibilità ed i funzionamenti dei principi costituenti l'applicativo commissionato.
L'idea di eliminare la parte di compilazione cartacea dei documenti da parte dei medici, passando tutto il carico di lavoro ad un qualche assistente intelligente ed automatizzato, non è nuova nello stato dell'arte [13]. Questo però non è il solo compito che ci è stato commissionato: in aggiunta alle precedenti direttive, è stata richiesta anche la lettura automatizzata dei parametri vitali dal monitor multi-parametrico presente nella sala del Pronto Soccorso, così da rimuovere un ulteriore compito a carico del Trauma-Leader, automatizzando ulteriormente la procedura di compilazione del report finale.

2.1.1 Concezione iniziale del sistema

L'ideazione di questo sistema è stata guidata dalle richieste dei medici dell'ospedale Maurizio Bufalini di Cesena. Tra le varie caratteristiche che ci sono state richieste per il progetto, ci è stata inoltre imposta la stringente limitazione di non poter interfacciarci in maniera diretta con i macchinari del Pronto Soccorso, poiché si potrebbero creare condizioni particolari in cui potrebbero generarsi malfunzionamenti, mettendo in pericolo la vita dei pazienti. Analizzando così le richieste che ci sono state fatte, il sistema è stato inizialmente ideato come un insieme di tre dispositivi comunicanti tra loro, che comprendeva:

- Un paio di **smart-glass**, aventi la funzione di mostrare al Trauma-Leader i dati che si stanno raccogliendo, ed attraverso la telecamera inserita negli

stessi rilevare la posizione del monitor multiparametrico e prelevarne i valori visualizzati;

- Un **microfono**, necessario per registrare i comandi impartiti;

- Uno **smartphone**, sempre presente nella tasca dell'utilizzatore del sistema, che funge da coordinatore degli altri due dispositivi, e che ha il compito di memorizzare i dati registrati durante le operazioni di Pronto Soccorso.

2.2 Cambiamenti in corso d'opera

Il progetto finora elaborato, ha subito dei significativi cambiamenti in corso d'opera. Nel momento in cui è stato elaborato un primo modello del sistema, è subito parso sconveniente implementare il riconoscimento dei parametri del monitor direttamente sullo smart-glass. Questo infatti voleva dire che il Trauma-Leader doveva per qualche istante voltarsi verso lo schermo raffigurante i parametri, distogliendo lo sguardo dalle sue mansioni principali, permettendo così al sistema di rilevarne i valori. Il gesto così compiuto dall'utilizzatore, non solo vanifica buona parte dell'essenza di tutto il progetto, ma potrebbe risultare anche svantaggioso dal punto di vista dell'utilizzo pratico. Dal momento in cui il Trauma-Leader è costretto a volgere il proprio sguardo verso il monitor, avrà sicuramente la capacità di rilevare in meno di un battito di ciglia il valore desiderato, senza poi aver la necessità di richiedere al sistema Trauma-Tracker il parametro di cui necessita, poiché l'avrebbe già letto molto prima di quanto esso possa impiegarci per individuarlo e mostrarglielo. Dopo queste considerazioni, è stato quindi necessario aggiungere al progetto un qualche sistema di controllo, sempre presente e vigile sul monitor, in grado di interpretare continuativamente i valori presenti su di esso, ed inviarli all'occorrenza agli occhiali. Il dispositivo ideale per svolgere questo compito è sicuramente un microprocessore, collocato sempre in una posizione adiacente al monitor da rilevare, così che il Trauma-Leader possa sempre essere a conoscenza dei parametri del paziente su cui sta operando, senza mai distogliere lo sguardo da esso. In questo modo è stato aggiunto al progetto un vero e pro-

prio livello "magico", incorporando ad esso ciò che mancava per poter essere definito un vero e proprio *Enchanted-Object*.

Incorporando queste nuove caratteristiche, ora il progetto è diventato somigliante ad un altro *Proof of Concept* già ideato dalla *Philips Healthcare*. Nel sito online dedicato[1] vengono denotate le sue caratteristiche principali e messi in luce gli scopi finali e le potenzialità del loro sistema, molto affine a Trauma-Tracker:

> *"Imagine a device that allows doctors performing surgery to simultaneously monitor a patient's vital signs and react to changes – without ever having to take their eyes off the procedure or patient. We've imagined it and more. [..] We have simulated the first proof of concept for the seamless transfer of patient vital signs into Google Glass. The potential? A whole new way for doctors to quickly get the information they need when they need it most. [..]" [14]*
> (Philips Healthcare)

E' anche possibile vedere sempre nel medesimo sito, un video dimostrativo [2] molto significativo, volto a mostrare le potenzialità di questo sistema. Nonostante il *Proof of Concept* da loro ideato possa sembrare simile, se non addirittura identico al nostro progetto, in realtà le differenze tra i due sistemi sono più di una, partendo innanzitutto dal fatto che la Philips, essendo un grosso produttore di macchinari ospedalieri, ha avuto fin da subito la possibilità di poter comunicare direttamente con le proprie apparecchiature mediche, senza avere l'incombenza di dover acquisire i dati visivamente dai monitor.

Ulteriori cambiamenti in corso d'opera sono stati successivamente apportati al progetto originale: vista la dislocazione dei differenti locali del Pronto Soccorso in cui possono essere compiute le manovre di salvataggio, si è pensato di aggiungere dei dispositivi nei singoli ambienti, per rilevare la posizione in cui il paziente si trova, ed aggiungere così ulteriori informazioni al report finale. Per implementare queste funzionalità possono essere utilizzati dei *Beacon*, ossia dei dispositivi Bluetooth da applicare in ogni stanza, che fungono

[1] Philips Healthcare Proof of Concept: `http://www.usa.philips.com/healthcare/innovation/research-and-exploration/google-glass-and-intellivue`

[2] Philips Healthcare Proof of Concept video: `https://www.youtube.com/watch?v=ssldTFWBv3E`

da faro, emettendo un segnale periodico, attraverso la cui analisi può essere identificato il locale in cui ci si trova.

2.3 Porzione d'interesse del progetto

Vista la grande mole dell'intero progetto, esso è stato suddiviso in diverse sezioni, ognuna maggiormente approfondita da un diverso elemento del team di sviluppo, ripartendo così il carico di lavoro totale. In particolare, la porzione di progetto di mio interesse è quella riguardante il microprocessore per il riconoscimento dei parametri vitali e il loro invio/ricezione verso lo smartphone. Ciò non significa che ogni membro ha lavorato solo ed esclusivamente alla realizzazione della porzione assegnatagli, ma solamente che si è prestata maggiore attenzione e dedizione verso le parti conferite, contribuendo comunque attivamente alla creazione e allo sviluppo anche delle sezioni restanti.

Capitolo 3

Analisi dei requisiti

Questo capitolo è volto innanzitutto ad analizzare i requisiti che il sistema deve possedere e, in secondo luogo, ad esaminare l'hardware che si è deciso di impiegare per la sua realizzazione con i motivi che ci hanno spinto alla sua scelta.

3.1 Requisiti del sistema

Come già precisato in precedenza, i requisiti di questo sistema sono stati ben definiti da coloro che ne hanno commissionato la realizzazione, ossia medici ed infermieri dell'ospedale Maurizio Bufalini di Cesena. Le principali qualità del sistema che sono state richieste possono essere schematizzate come segue:

1. **Consultazione parametri vitali** senza mai distogliere lo sguardo dal paziente su cui si sta operando. Questo è uno dei punti fondamentali dell'applicazione. Per la realizzazione di questa caratteristica è stato scelto di utilizzare degli smart-glass, che lasciano libere in ogni momento le mani del medico, e attraverso la trasparenza delle lenti possono essere proiettate immagini contenenti le informazioni richieste;

2. **Compilazione automatica report.** Anche questo assieme al primo punto costituisce la base del progetto. Per raggiungere questo risultato si è pensato di utilizzare uno smartphone, collocato al centro di tutto il sistema, che funge da controllore per tutti i dispositivi ad esso collega-

ti, ne raccoglie le informazioni principali per la compilazione del report finale, e lo stila nel momento in cui viene richiesto;

3. **Hands-Free system.** Il sistema non soltanto deve garantire al Trauma-Leader la visualizzazione dei parametri vitali senza doversi girare verso il monitor, ma tutto questo deve essere realizzato anche in un ottica hands-free. Gli smart-glass già compiono bene questa funzione, mostrando i dati all'utente senza l'utilizzo delle mani, avendo la possibilità di comandare il sistema attraverso comandi vocali;

4. **Funzionamento garantito in ogni situazione**, anche in quelle di emergenza dove si può creare rumore di fondo che può disturbare la ricezione dei comandi. A questo scopo può essere usato un laringofono, che grazie alle sue caratteristiche, diverse da quelle di un normale microfono, assicura la perfetta acquisizione dei comandi vocali in ogni condizione acustica;

5. **Gestione eventi di pericolo.** Se per esempio il paziente ha un battito cardiaco insufficiente, il sistema deve avvisare il medico, allarmandolo opportunamente, così che possa intervenire prontamente;

6. **Reattività.** Ovviamente se il sistema non presenta questa caratteristica è pressoché inutile. La prontezza nella gestione degli eventi di pericolo è indispensabile per la salute del paziente, ed anche la rapidità con cui esso reagisce ai comandi impartiti dall'utilizzatore è di fondamentale importanza;

7. **Facilità di utilizzo e confidenza.** Il sistema deve essere facilmente utilizzabile da ogni medico o infermiere dell'ospedale. I comandi impartiti devono essere universalmente riconosciuti come validi, per garantire la fruibilità del sistema a chiunque;

8. **Uso intelligente delle risorse.** Tra le varie risorse utilizzate dal sistema, il consumo energetico in particolare non deve essere eccessivo, e deve consentire un utilizzo prolungato, di qualche ora almeno, prima di necessitare di essere ricaricato.

3.1.1 Utilizzo del sistema

Secondo i requisiti che ci sono stati forniti dai medici, l'utilizzatore del sistema Trauma-Tracker deve essere sempre e solo uno, ovvero il Trauma-Leader. Egli può richiedere di visualizzare sugli smart-glass i parametri del paziente corrente, di avviare e terminare la sessione di registrazione e di stilare il rispettivo report, contenente i dettagli delle operazioni svolte. Contestualmente l'operatore può anche impartire comandi per la registrazione dei parametri letti e delle manovre eseguite, e far partire o terminare la registrazione di un video da parte degli smart-glass. Ogni volta che un ordine di registrazione viene pronunciato, il sistema oltre a memorizzare i dati comandati dal Trauma-Leader, aggiungerà tutta una serie di informazioni riguardanti per esempio la data, l'ora, il locale in cui ci si trova ed ulteriori dettagli volti a migliorare la qualità del report finale prodotto. Il seguente diagramma dei casi d'uso rappresenta le varie possibilità di utilizzo del sistema appena descritte.

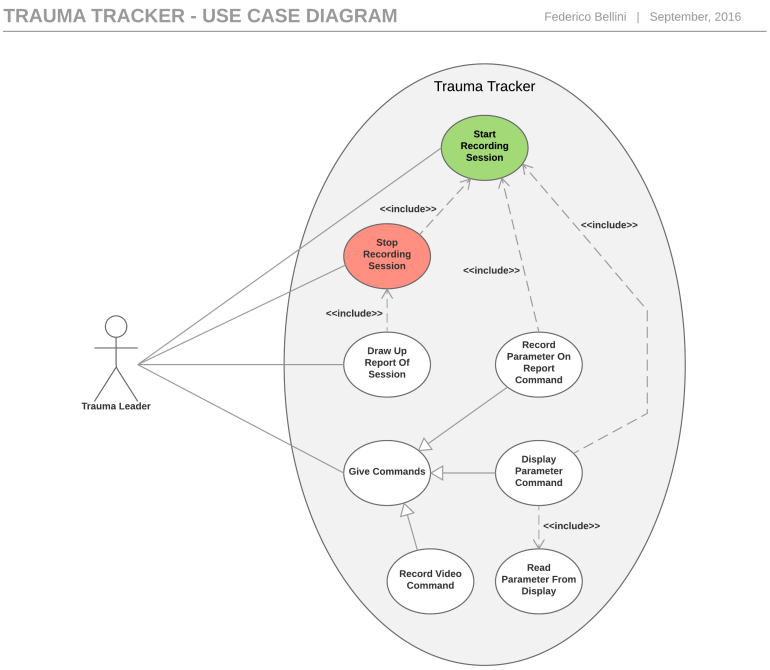

Figura 3.1: Use-Case Diagram Trauma-Tracker.

3.2 Hardware impiegato

Di seguito verranno descritti i diversi apparati hardware impiegati per la realizzazione del progetto, e le principali caratteristiche per cui sono stati scelti.

3.2.1 Smartglass - Vuzix M300

La scelta degli smart-glass da utilizzare è stata molto complessa e a lungo dibattuta. Alla fine è stato scelto di utilizzare i *Vuzix M300*[1], per la loro estrema versatilità ad essere usati in ogni campo e per le sue proprietà di connettività, alte prestazioni, efficienza energetica e ottima qualità della camera. La scelta di utilizzare il modello più recente rispetto ai precedenti M100 è stata fatta valutando qualità e prestazioni del prodotto, nettamente migliori nel dispositivo più moderno.

Figura 3.2: Smartglass - Vuzix M300. [2]

3.2.2 Microprocessore - Raspberry Pi 3 Model B

Come microprocessore è stato scelto di utilizzare il Raspberry Pi 3 Model B. La scelta è stata determinata innanzitutto da un fattore economico, infatti questo piccolo computer si trova a buon mercato, ed è stato preferito rispetto al modello precedente, il Pi 2, poiché leggermente più performante ed anche

[1]Vuzix M300: `https://www.vuzix.com/Products/m300-smart-glasses`
[2]Smartglass- Vuzix M300: `https://www.vuzix.com/Products/m300-smart-glasses`

perché presenta il modulo Wi-Fi già integrato. Sul microprocessore è stato installato il sistema operativo *Raspbian*[3].

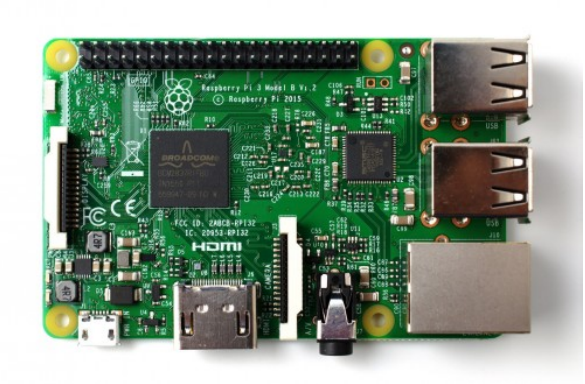

Figura 3.3: Microprocessore - Raspberry Pi 3 Model B. [4]

E' stato necessario inoltre aggiungere a questo sistema una telecamera USB per la cattura delle immagini del monitor multiparametrico.

3.2.3 Microfono - Braudel KBL 002

Al posto di un normale microfono, è stato deciso di utilizzare un laringofono, a causa dell'eccezionale resistenza al rumore di fondo che offre. Il modello utilizzato è il *Braudel KBL-002*. Esso è economico, e presenta nativamente un modulo Bluetooth 4.0 di ultima generazione con cui comunicare con lo smartphone. Per garantire massima libertà di movimento e un minor intralcio nel compimento delle azioni del Trauma-Leader, è stato scelto di utilizzare un dispositivo wireless piuttosto che uno provvisto di cavo per la comunicazione.

[3]Raspbian: `https://www.raspberrypi.org/downloads/`

[4]Microprocessore - Raspberry Pi 3 Model B: `https://www.raspberrypi.org/weekly/connected/`

Figura 3.4: Laringofono - Braudel KBL-002. [5]

3.2.4 Beacons - Estimote Beacons

I *Beacons* sono dei piccoli dispositivi hardware che attraverso una connessione Bluetooth-Low-Energy(BLE) trasmettono il loro codice identificativo univoco a tutti i device nelle loro vicinanze. Essi possono essere considerati come una sorta di *fari marittimi* che periodicamente suggeriscono a chi li può percepire la posizione in cui si trovano. Il principale utilizzo che se ne fa è quello di eseguire differenti operazioni sui dispositivi che ne captano il segnale, a seguito della comprensione della posizione in cui ci si trova. Nel nostro caso vengono utilizzati per informare il sistema Trauma-Tracker del locale del Pronto Soccorso in cui si stanno eseguendo le operazioni, aggiungendo così ulteriori informazioni al report finale.

Figura 3.5: Beacons - Estimote Beacons. [6]

[5]Laringofono - Braudel KBL-002: `https://www.facebook.com/kafka1984/photos/a.1445294705777577.1073741828.1438726846434363/1449443008696080/?type=3&theater`

3.2.5 Smartphone - Android

Per questo hardware non è stato scelto un dispositivo in particolare, tablet o smartphone che sia, ma è stato solamente deciso di utilizzare un qualsiasi device che monti *Android* come sistema operativo. Questo perché conta poco l'utilizzo un cellulare piuttosto che un altro, ma la scelta del sistema operativo su cui lavorare è decisiva, poiché applicando lo stesso programma sviluppato per *Android* su di un altro sistema operativo, quasi certamente non funzionerebbe.

3.2.6 Monitor Multiparametrico - Drager M540

Il monitor multiparametrico utilizzato dall'ospedale Maurizio Bufalini di Cesena è il *Drager M540*. In questo schermo possono essere visualizzati fino a cinque parametri in contemporanea, ognuno contraddistinto da una posizione ed un colore differente.

Figura 3.6: Monitor Multiparametrico - Drager M540. [7]

[6]Beacons - Estimote Beacons: `http://estimote.com/`

[7]Monitor multiparametrico - Drager M540: `http://www.draeger.com/Sites/enus_us/Pages/Hospital/Infinity-M540-monitor.aspx`

Siccome non è stato possibile avere questo dispositivo in prestito per eseguire dei test, per tutta la fase di collaudo del progetto è stata utilizzata un'interfaccia grafica che replica fedelmente *Simmon*[8], ossia l'applicazione che ci è stata consigliata direttamente dai medici, poiché riproduce fedelmente lo schermo del *Drager M540*, e che viene utilizzata dagli stessi per formare nuovi infermieri e medici.

3.2.6.1 Riepilogo hardware

In conclusione schematizzando tutto l'hardware utilizzato, si può comporre il seguente diagramma di deployment, contenete tutte le parti fisiche utilizzate nel progetto finale.

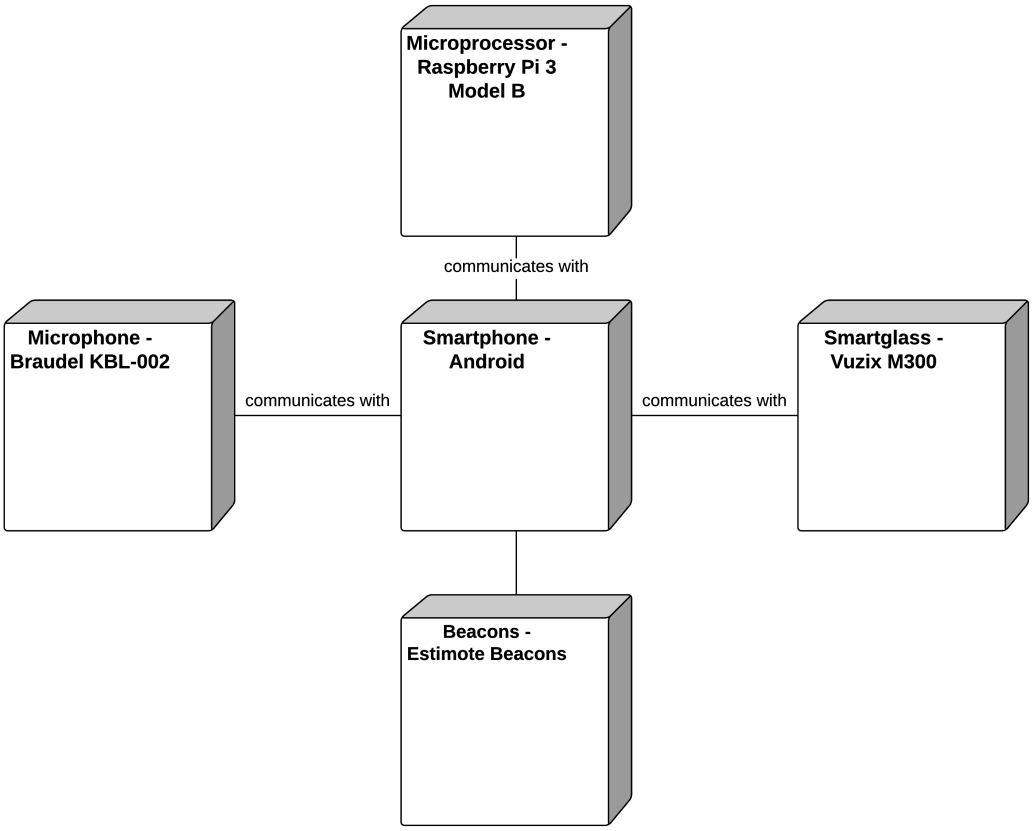

Figura 3.7: Deployment Diagram Trauma-Tracker.

[8]Simmon application website: `http://castleandersen.dk/apps/simmon/`

Capitolo 4

Progettazione del sistema

L'architettura generale del sistema è stata progettata da tutti i membri del team di sviluppo. Il progetto, nella sua integrità, è stato la parte centrale e più importante dei nostri studi e attività. Data la sua grande mole, è stato però necessario suddividerlo in macro sezioni, di cui ogni membro del team si è maggiormente occupato, prestando più attenzioni e riguardi sia nella sua implementazione che nel suo sviluppo. Le tre aree principali identificate sono state:

- Una parte core più corposa, presente nel dispositivo Android, che ha il compito di gestire le comunicazione con gli altri hardware e di produrre il report finale;

- Il sottosistema che comprende il microfono, che ha il compito di elaborare gli input provenienti da esso e di inviare il risultato dell'analisi alla parte core;

- Il sottosistema che concerne l'individuazione e l'analisi dei parametri vitali presenti sul monitor multi-parametrico.

In questo capitolo verranno presentate inizialmente le possibili problematiche che possono insorgere nel sistema, in seguito la progettazione dell'architettura generale del sistema, e infine la parte riguardante la sezione del progetto a me assegnata.

4.1 Possibili problematiche

Dopo aver analizzato i requisiti di cui il sistema necessita, le principali problematiche che possono insorgere sono le seguenti:

- Per consentire agli utilizzatori un uso prolungato del sistema, è necessario limitare il più possibile il consumo energetico di ogni singolo elemento di cui esso si compone. Per fare ciò, è indispensabile progettare il sistema in modo da alleggerire il carico di lavoro alle parti più stressate. In particolare, è stato utile l'inserimento di un server web, frapposto nella comunicazione tra il Raspberry e lo smartphone Android, con il compito di raccogliere i dati provenienti dal Raspberry, mantenendo sempre lo stato dei parametri aggiornato, ed inviandoli all'occorrenza al sottosistema dello smartphone. Questa scelta progettuale verrà approfondita maggiormente nella Sezione 4.3.
  Con questo ampliamento del sistema, è necessario garantire che sia sempre disponibile una connessione ad internet per assicurarne il funzionamento, altrimenti il sistema di analisi dei dati dal monitor e quello centrale dello smartphone risulterebbero scollegati tra loro. Questa problematica è in parte già superata, poiché nelle direttive forniteci dai medici dell'ospedale, viene assicurata la presenza di una connessione in ogni locale della struttura ospedaliera;

- E' possibile inoltre che il sistema di riconoscimento vocale dei comandi generi degli errori dovuti alla cattiva ricezione o interpretazione del comando impartito. Per ovviare a questo problema è stato pensato di fornire un sistema di feedback all'utente, grazie al quale può sempre avere sotto controllo i comandi pronunciati, verificandone la correttezza, ed eventualmente annullandoli se errati o pronunciandoli una seconda volta;

- Infine possibili problemi possono insorgere nell'apparato di rilevazione dei parametri dal monitor. Infatti l'analisi viene fatta su immagini catturate da telecamere poste nell'ambiente che inquadrano lo schermo contenente i dati. Lavorando su immagini digitalizzate si hanno così tutti i problemi di rumore, distorsione e soggezione alle variazioni delle condizioni

di luce e riflessi ambientali legati intrinsecamente a tale tipologia di immagini. Per rimediare a questi inconvenienti si è pensato di allegare al report finale anche lo screenshot del monitor catturato dalla telecamera sul quale sono stati applicati gli algoritmi di riconoscimento dei parametri. In questo modo anche se i valori non vengono letti correttamente, è sempre possibile a posteriori correggere eventuali errori o imperfezioni nell'interpretazione dei parametri.

4.2 Architettura generale

Rispettando i requisiti progettuali presentati nel capitolo precedente, il sistema Trauma Tracker è stato concepito per trovarsi sempre in due possibili stati: attivo o inattivo. Fin tanto che il sistema è inattivo, il componente incaricato del riconoscimento dei comandi resta in ascolto, reagendo solamente quando l'ordine di avvio del sistema viene pronunciato, oppure quando viene proferito quello per stilare il report, ignorando quindi tutti gli altri possibili input. Dal momento in cui il sistema passa dallo stato di inattività a quello di attività, inizia a reagire ad ogni ordine imposto. I possibili comandi che possono essere impartiti riguardano la terminazione della sessione di registrazione, la visualizzazione di un parametro sui glass, oppure la registrazione di un nuovo record nel report finale.

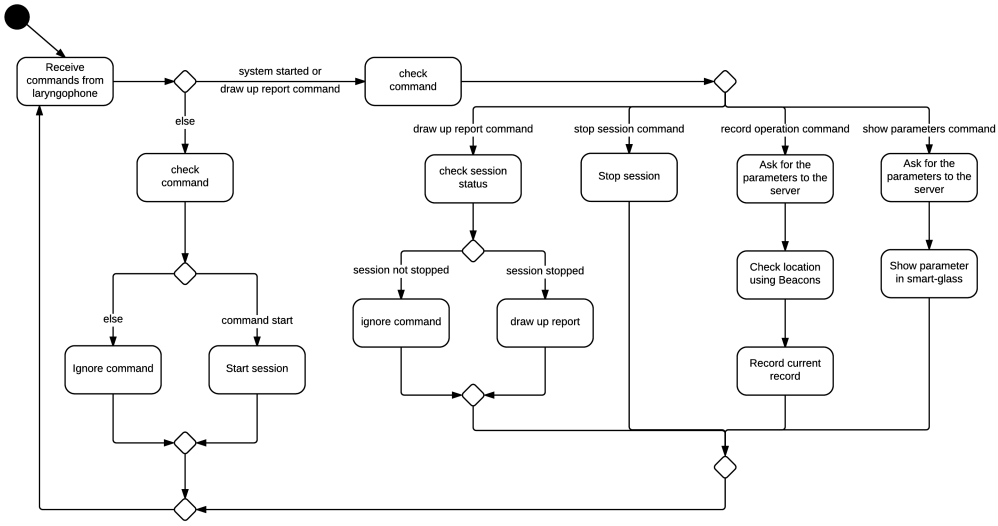

Figura 4.1: Activity Diagram Trauma-Tracker.

Come si può notare dal diagramma di Figura 4.1, nel caso in cui volesse essere aggiunto un nuovo record di dati, il sistema elabora quale parametro o manovra vengono richiesti per essere salvati, interroga il server per ricevere gli ultimi valori aggiornati del parametro richiesto, cattura i segnali emessi dai Beacons per individuare il locale in cui ci si trova, ed infine compila il report con i dati ottenuti.

Entrando più nel dettaglio dell'architettura software del sistema, come si può notare dalla Figura 4.2, nella sua struttura interna è possibile riconoscere il componente principale denominato *TraumaTrackerAgent*, che comunica con tutti gli altri sottosistemi, elabora le informazioni raccolte per la creazione del report finale e si occupa di reagire intraprendendo la giusta azione ogni volta che viene ricevuto un comando. Il seguente diagramma dei componenti descrive la struttura interna del sistema software in termini dei suoi componenti principali e delle relazioni tra essi.

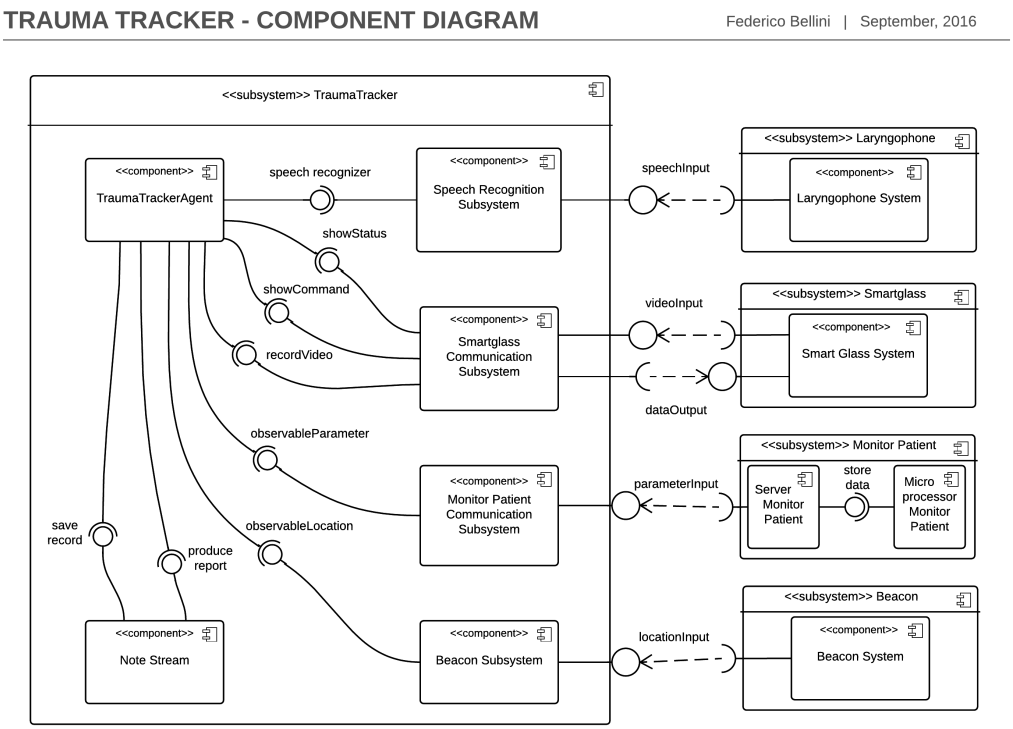

Figura 4.2: Component Diagram Trauma-Tracker.

Come si può notare dal precedente diagramma, il progetto è stato suddiviso in più sottosistemi. Le interfacce esposte da ogni subsystem sono quelle utilizzate per la comunicazione e lo scambio di dati tra di essi. All'interno del sistema principale *TraumaTracker*, si può notare che il componente fondamentale *TraumaTrackerAgent* non ha un'interazione diretta con i componenti esterni, ma che per ognuno di essi è presente un componente dedicato che si occupa della comunicazione, dello scambio di dati e dell'elaborazione degli input provenienti dai dispositivi periferici. Il sottosistema riguardante la parte di rilevamento dei parametri vitali dal monitor, il funzionamento del server che si frappone nella comunicazione tra il microprocessore e la parte centrale del sistema, verranno approfonditi nella sezione successiva.
Per la realizzazione della parte di Trauma Tracker appena descritta sono stati utilizzati due differenti approcci metodologici: uno *Agent-Oriented* ed uno *Object-Oriented*. La sottosezione al centro di tutto il progetto, ovvero quella denominata *TraumaTracker*, e che ha il compito di gestire le relazioni con tutti gli altri dispositivi e di reagire agli stimoli esterni, è stato progettato di implementarla sfruttando tutte le caratteristiche di reattività, adattabilità, distribuzione e concorrenza che la programmazione ad agenti offre. La scelta è stata fatta anche pensando a futuri sviluppi del progetto, che potrebbero includere anche le interazioni con altri sistemi come questo, scambiandosi dati ed informazioni, e diventando così un sistema distribuito multi agente. Tutti i sistemi che invece si trovano più in periferia rispetto a questa parte centrale, ossia tutti i subsystems rimanenti, sono stati implementati con linguaggi orientati agli oggetti, garantendo così in ogni modo affidabilità e robustezza al sistema.

4.2.1 Object-Oriented Programming, Agent-Oriented Programming e Multi-Agents System

In questa sottosezione verranno esposte le principali qualità della programmazione orientata agli agenti, dei sistemi multi agente e le differenze con i classici linguaggi orientati agli oggetti.

Grazie alla crescente diffusione degli *IoT* e di ambienti distribuiti, la filosofia di programmazione ad agenti si sta rapidamente consolidando come un vero e proprio metodo di programmazione largamente riconosciuto. Avendo oggi a disposizione un numero sempre più crescente di dispositivi intelligenti connessi alla rete, si possono concepire applicazioni distribuite, formate da una comunità di componenti, detti "agenti", che comunicano tra di loro. Gli **agenti** sono definiti come sistemi computazionali situati in un **ambiente**, capaci di azioni autonome per raggiungere i loro obiettivi. Essi non hanno completa conoscenza dell'ambiente in cui sono immersi, e non hanno nemmeno le capacità per poterlo controllare, poiché le azioni intraprese su di esso possono anche fallire. In poche parole, l'ambiente può cambiare dinamicamente indipendentemente dal comportamento dell'agente. Per questo motivo l'ambiente viene considerato non-deterministico, ovvero che non può essere previsto a priori il suo comportamento futuro. Un **agente intelligente** deve quindi essere **reattivo**, reagendo tempestivamente ai cambiamenti che si verificano nell'ambiente; **pro-attivo**, compiendo azioni anticipate basate sulla percezione delle possibili tendenze di cambiamento dell'ambiente; **sociale**, interagendo con altri agenti per portare a compimento il proprio obiettivo. Grazie alle forti similitudini degli agenti con gli esseri viventi, essi spesso vengono modellati per mezzo di concetti cognitivi basati su stati mentali, come credenze, desideri o intenzioni. Le credenze o **beliefs** sono le informazioni sul mondo esterno di cui l'agente dispone. Esse possono essere incorrette ed incomplete. I desideri o **desires** rappresentano le situazioni che l'agente vorrebbe realizzare e le **intentions** sono i desideri che l'agente si è impegnato a realizzare. Gli agenti svolgono così un lavoro ciclico e continuativo: acquisiscono nuove percezioni dell'ambiente, aggiornano le proprie credenze, generano nuovi desideri, selezionandone uno da realizzare e cercando un piano per tentare di soddisfarlo. E poi ancora ed ancora.

Fornendo inoltre le capacità agli agenti di comunicare tra di loro, è facile capire le enormi potenzialità che possono scaturire dalla formazione di sistemi multi agente che, come un esercito di formiche, collaborano per soddisfare le loro intenzioni, in modo molto più agile e veloce di quanto il singolo individuo o agente possa mai fare.

Potrebbe sembrare che la somiglianza tra agenti e oggetti sia elevata, e addirittura che queste due filosofie possano essere sovrapposte. In entrambi i casi infatti lo stato dell'oggetto viene incapsulato e non è reso visibile all'esterno, ambedue eseguono azioni su altri oggetti e monitorano ciò che gli accade intorno, quasi come se venisse implementato un semplice *pattern Observer* sull'ambiente che li circonda. Le differenze in realtà sono notevoli, fondamentalmente perché gli oggetti non hanno una propria autonomia decisionale, ma sono costretti ad eseguire i comandi che gli vengono impartiti dall'esterno. Differentemente gli agenti non garantiscono l'esecuzione dei comandi impartiti, aggiungendo così la caratteristica di autonomia decisionale di cui i semplici oggetti sono sprovvisti. In breve: ***"Objects do it for free. Agents do it because they want to."***

4.3 Architettura dettagliata del Patient Monitor Subsystem

Come già detto, l'intera struttura del progetto è stata divisa in più sottosistemi, ognuno dei quali maggiormente approfondito da uno specifico elemento del team di sviluppo. In questa sezione verrà approfondita la parte a me assegnata, ossia quella riguardante il Patient Monitor Subsystem. Del sottosistema attribuitomi fa parte anche il server web utilizzato per l'archiviazione e l'invio dei dati rilevati.

Il Raspberry presente nel sistema svolge una funzione ciclica e continuativa, volta a fornire in tempo reale informazioni sui valori dei parametri visualizzati nel monitor del paziente. Le attività svolte da un singolo ciclo di analisi in sequenza sono: acquisizione di una nuova immagine del monitor, rilevamento della posizione dei parametri, eventuale riconoscimento dei valori ed invio dei dati al server.

Il diagramma di attività mostrato di seguito presenta l'algoritmo appena illustrato.

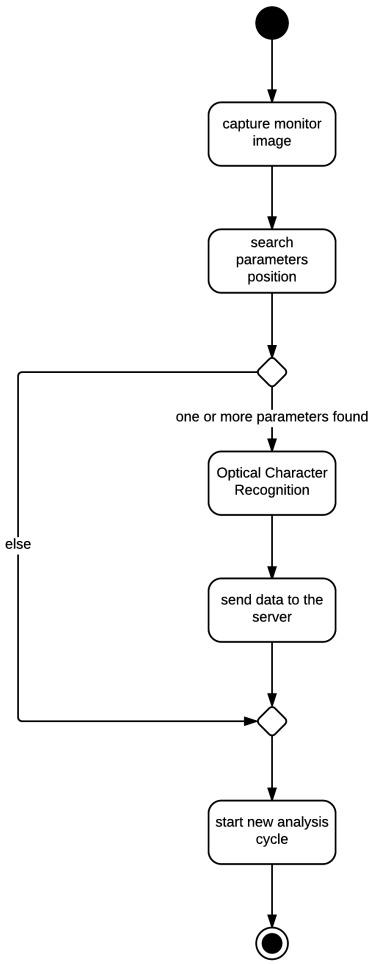

Figura 4.3: Activity Diagram singola analisi del Patient Monitor Subsystem.

L'algoritmo qui esposto, può essere facilmente parallelizzato, sovrapponendo più task di analisi tra loro, poiché comunque risulterebbero indipendenti. In questo modo possono essere sfruttate al massimo le risorse del sistema, e il tempo di campionamento dei frame del monitor può diminuire.

Il server web frapposto nello scambio di dati tra il Raspberry e lo smartphone è stato aggiunto per differenti motivazioni. Innanzitutto, bisogna precisare che, dopo aver compiuto dei piccoli test per verificare le latenze del siste-

ma dal momento in cui il Raspberry inizia l'elaborazione dell'immagine alla conclusione dell'algoritmo per il riconoscimento dei caratteri, è emerso che in questa prima implementazione prototipale del progetto, il tempo impiegato per le elaborazioni è notevole, in quanto si aggira nell'ordine di alcuni secondi. Per questo motivo è da subito parso poco conveniente un'implementazione del sistema in cui la richiesta dei parametri da parte dello smartphone al microprocessore avvenisse solamente nel momento in cui tali valori venissero ricercati dall'utente. L'intero procedimento di riconoscimento e invio impiegherebbe un lasso di tempo non accettabile per gli scopi del progetto. Per questo è stata quasi obbligata la scelta di progettare il sistema del Raspberry come un ciclo infinito, in cui esso procede ad un analisi periodica e continuativa dei parametri dello schermo. A questo punto la scelta più ovvia sarebbe stata quella di mantenere gli ultimi dati rilevati all'interno del microprocessore, ed inviarli solo all'occorrenza ad Android, nel momento in cui vengono richiesti. Questo approccio però presenta due svantaggi: il primo è ovviamente quello rappresentato dall'aggiunta di un carico di lavoro in più per il microprocessore, che dovrebbe sempre rimanere in attesa di essere contattato da Android prima di iniziare la comunicazione dei valori; in secondo luogo, dato che oltre ai singoli valori viene inviata anche l'immagine catturata, sulla quale sono stati individuati e calcolati i parametri vitali per garantire un feedback di controllo dei dati letti nel report finale, l'invio di tutto il pacchetto di informazioni mediante la tecnologia Bluetooth avrebbe aggiunto un'ulteriore latenza, seppur piccola. Per questi motivi è stata prediletta l'implementazione con il server web che fa da mediano nella comunicazione: esso mantiene lo stato aggiornato dei dati che periodicamente gli vengono inviati dal microprocessore, e solo all'occorrenza fornisce in modo veloce e rapido i parametri richiesti dallo smartphone. Il diagramma di sequenza proposto di seguito può aiutare a comprendere meglio lo scenario di applicazione di questo componente. Come si può notare, esso reagisce prontamente alle richieste di Android o Raspberry, anche se concorrenti tra loro. A seguito della chiamata asincrona del microprocessore separa il proprio flusso di controllo principale, creandone un altro dedicato al salvataggio dei dati. Similmente accade anche a seguito della chiamata sincrona da parte dello smartphone: ricerca i parametri richiesti nel database in maniera parallela rispetto al flusso di controllo principale, per poi restituire i dati

al sistema richiedente. Un vantaggio intrinseco nell'utilizzo del server, è la possibilità di scalabilità del sistema, che sommata alla programmazione orientata agli agenti utilizzata nella parte *Core* del progetto, apre ampie possibilità di realizzazione di sistemi multi-agenti in futuro, in cui diversi dispositivi di analisi dei parametri vitali comunicano con più dispositivi richiedenti valori, creando di conseguenza una rete di sistemi intelligenti di supporto al lavoro di medici ed infermieri.

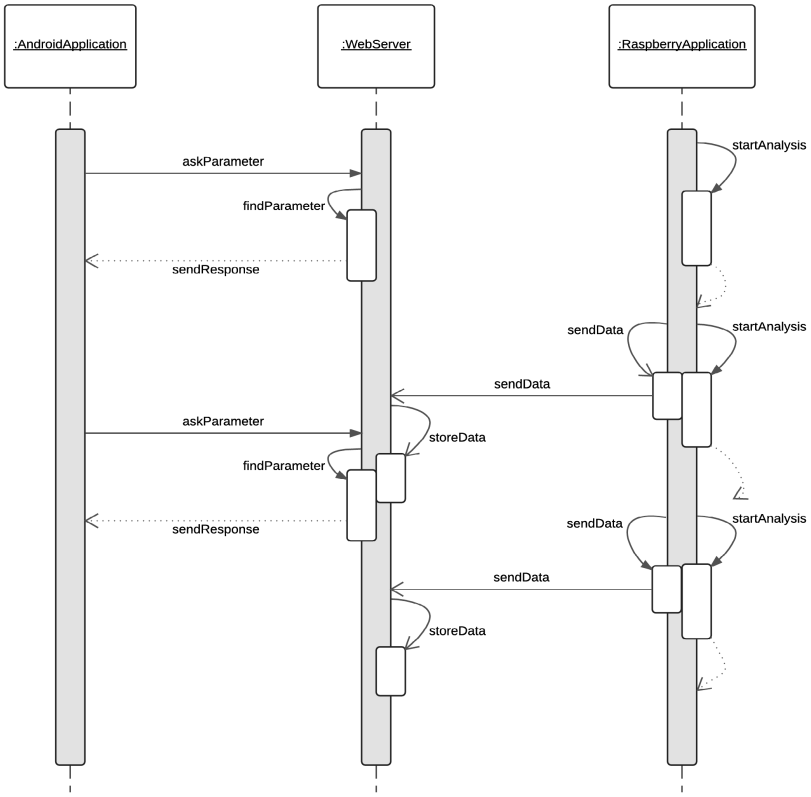

Figura 4.4: Sequence Diagram Patient Monitor Subsystem.

Capitolo 5

Implementazione Patient Monitor Subsystem

Il progetto Trauma-Tracker in questa sua prima versione è stato concepito come un *Proof of Concept*, ossia un prototipo creato per dimostrare la fattibilità del sistema nel suo complesso, e per individuare punti critici e problematiche che solo testando il sistema nella realtà possono venire alla luce. In questo frangente quindi non è stato dato troppo riguardo alle prestazioni del sistema, privilegiando una buona progettazione ed utilizzando linguaggi di alto livello come *Java* per la sua realizzazione immediata. Per lo sviluppo di versioni future sicuramente si potrà dare maggiore importanza alle prestazioni di ogni singola parte del sistema, favorendo quindi la sua implementazione in linguaggi più vicini a quello macchina, come per esempio in *C++* o in *C*, che sicuramente garantirebbero migliori performance.

In questo capitolo verranno esposti dapprima gli elementi utili a comprendere a pieno l'implementazione del progetto, in secondo luogo le parti più salienti dell'implementazione del sottosistema, ed infine le problematiche riscontrate in fase di implementazione.

5.1 Elementi utili per la comprensione dell'implementazione del sistema: introduzione alle tecniche di Computer Vision utilizzate

Prima di proceder con la seguente sezione, è necessario rendere noti al lettore gli elementi utili per la comprensione dell'implementazione del sistema. Di seguito, verranno brevemente esposte le principali tecniche di Computer Vision utilizzate nella sua implementazione.

5.1.1 Immagini Grayscale, RGB, HSV e HSL

Prima di tutto bisogna partire dalle basi, ossia dai tipi di immagini digitali utilizzati in questo progetto. La forma più basilare di rappresentazione utilizzata è quella *grayscale* o a livelli di grigio. Queste sono immagini digitali raster in bianco e nero, costituite da una gamma di sfumature di grigi che in genere vengono codificati con valori da 0 a 255. Esse tipicamente vengono rappresentate attraverso una matrice di valori, avente come dimensioni le stesse dell'immagine che si vuole raffigurare, in cui ogni elemento della matrice rappresenta un singolo pixel dell'immagine. Queste immagini *grayscale* possono poi essere utilizzate per costituire ciascuna un livello di un'immagine a più canali: è questo il caso delle immagini *RGB*. Esse sono spesso costruite sovrapponendo differenti canali di colori, ciascuno dei quali rappresenta in livelli di grigio i valori del canale specificato. Ad esempio le immagini *RGB* sono composte da tre canali indipendenti, red, green e blue, mentre le immagini *RGBA* chiaramente sono costituite da quattro canali, gli stessi delle *RGB*, più un canale chiamato alfa, utilizzato per fornire informazioni aggiuntive sul colore. Il seguente esempio mostra la suddivisione di un'immagine *RGB* nei suoi tre canali costituenti. A destra sono presenti i tre canali in livelli di grigio equivalenti.

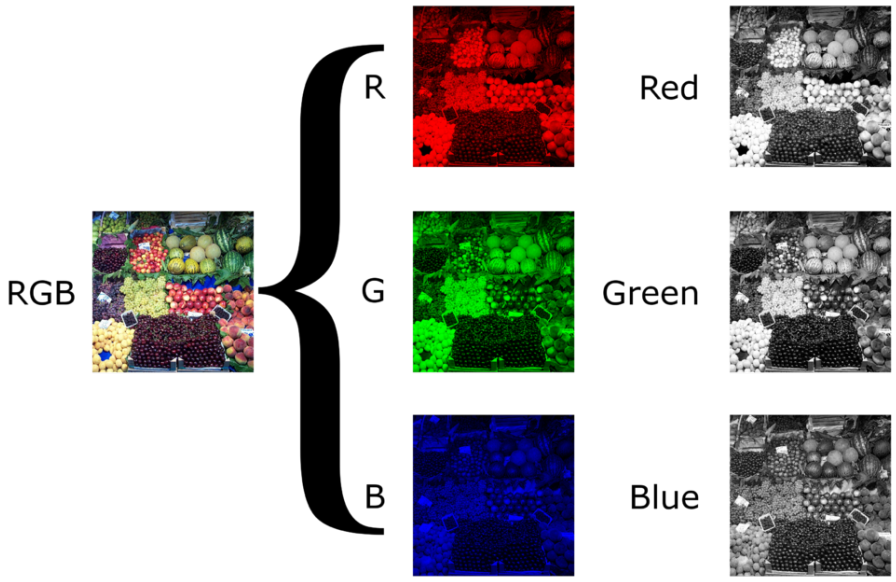

Figura 5.1: Scomposizione immagini a colori RGB in tre immagini a livello di grigio. [1]

Anche *HSV* e *HSL* sono modelli di rappresentazioni digitale dei colori molto simili tra loro, e con qualche caratteristica in comune ad *RGB*. Innanzitutto, sono composti da tre canali, che in questi ultimi rappresentano *hue* o tonalità, *saturation* o saturazione e *value/lightness*, ossia valore o luminosità. Questi modelli, ed in particolare quello *HSL*, sono particolarmente orientati alla percezione umana del colore, in quanto lo codificano in termini di tinta, sfumatura e tono. Dall'analisi di questo ultimo tipo di immagini risulta a volte più facile ricavare elementi utili all'interpretazione dell'immagine.
Normalmente i tre canali costituenti vengono informatizzati nel seguente modo: la tonalità viene misurata in un angolo intorno all'asse verticale dello spazio del colore, e quindi generalmente viene memorizzata con valori da 0 a 360, che indicano i gradi del vettore del colore attorno a questo asse; la luminosità, la saturazione e il valore vengono invece rappresentati come di consueto con valori da 0 a 255, oppure in alcuni casi con valori decimali da 0 a 1. Il seguente schema espone graficamente la struttura di questi due modelli ed il significato dei punti a cui queste tre grandezze corrispondono.

[1] Scomposizione immagini a colori RGB in tre immagini a livello di grigio: `https://it.wikipedia.org/wiki/File:Beyoglu_4671_tricolor.png`

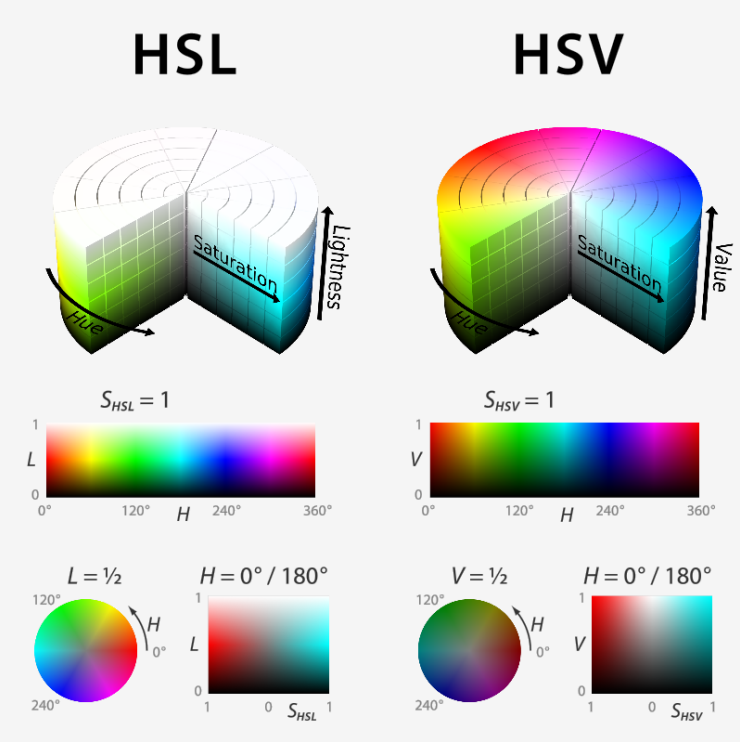

Figura 5.2: Modelli cilindrici HSL e HSV. [2]

5.1.2 Binarizzazione di immagini Grayscale

La binarizzazione di un'immagine in scala di grigi è il processo con il quale essa viene trasformata in un'immagine binaria, ovvero avente solo due livelli di grigio(bianco e nero). Esistono diverse tecniche per il raggiungimento di questo risultato. La più banale consiste nello stabilire a priori una soglia fissa con la quale l'immagine verrà trasformata: tutti i valori dei pixel dell'immagine maggiori di tale limite vengono cambiati e portati a 255(bianco), mentre i rimanenti a 0(nero). Tecniche poco più avanzate consistono nella scelta dinamica della soglia da applicare, calcolata come media dei livelli di grigio, o attraverso l'istogramma dell'immagine. L'utilizzo di soglie globali in genere non garantisce un buon risultato, soprattutto in immagini che presentano condizioni di luminosità differente nelle diverse aree. In queste circostanze esistono tecniche

[2]Modelli cilindrici HSL e HSV: `https://it.wikipedia.org/wiki/File:Hsl-hsv_models_b.svg`

per cui la soglia viene calcolata per piccole regioni dell'immagine, sulle quali poi viene applicato l'algoritmo di binarizzazione, così da ottenere risultati migliori anche in condizioni di luminosità differente.

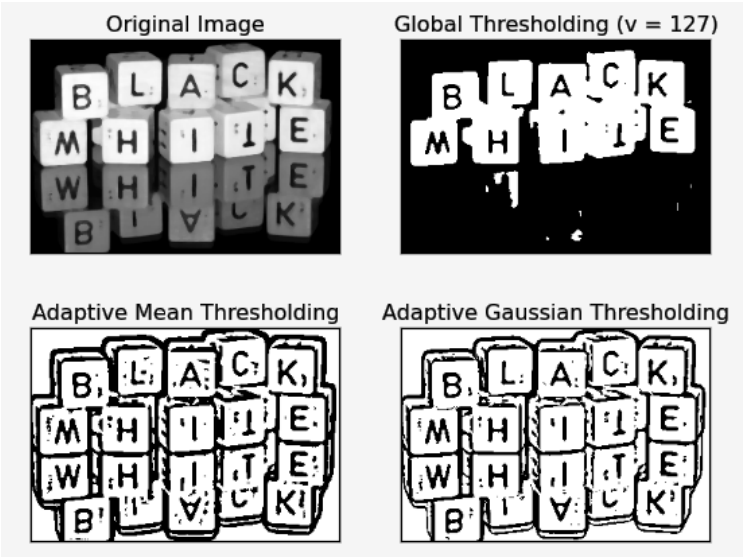

Figura 5.3: Binarizzazione immagini grayscale. [3]

Si può chiaramente notare dall'immagine di Figura 5.3 come la binarizzazione con soglia locale (le due immagini più in basso) garantiscano un risultato migliore rispetto alla binarizzazione con soglia fissa(in alto a destra).

[3]Binarizzazione immagini grayscale: `http://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Global_Thresholding_Adaptive_Thresholding_Otsus_Binarization_Segmentations.php`

5.1.3 Operazioni morfologiche

La morfologia matematica è una tecnica di analisi di forme geometriche ampiamente utilizzata nell'elaborazione di immagini digitali. In questo campo di applicazione, le tecniche morfologiche sono basate sulle due operazioni fondamentali: la dilatazione e l'erosione.
La **dilatazione morfologica** utilizza un elemento strutturante per ampliare le forme contenute nell'immagine in ingresso. Questo argomento non verrà ulteriormente approfondito, poiché non è rilevante per i fini e gli scopi di questo capitolo della tesi.

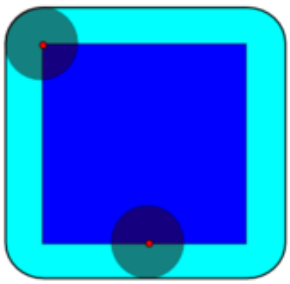

Figura 5.4: Dilatazione del quadrato blu con il disco. Il risultato è evidenziato dall'area azzurra. [4]

L'**erosione morfologica** al contrario utilizza un elemento strutturante per ridurre le forme contenute nell'immagine in ingresso.

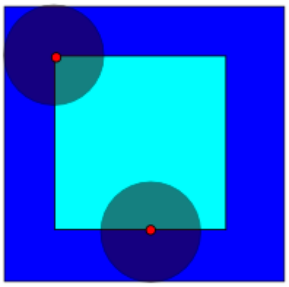

Figura 5.5: Erosione del quadrato blu con il disco. Il risultato è evidenziato dall'area azzurra. [5]

[4]Dilatazione: `https://en.wikipedia.org/wiki/File:Dilation.png`
[5]Erosione: `https://en.wikipedia.org/wiki/File:Erosion.png`

Questi due operatori fondamentali possono essere combinati per ottenere cambiamenti nell'immagine originale. Le principali operazioni morfologiche utilizzate nello sviluppo del progetto *Trauma Tracker* sono l'**apertura** (dilatazione seguita da erosione), e la **chiusura** (erosione seguita da dilatazione). Queste due tecniche sono utilizzate rispettivamente per ridurre le aree di foreground dell'immagine e rimuove piccoli oggetti, e nel caso opposto per ampliarle, chiudendo inoltre piccoli buchi all'interno delle aree di foreground e connettendo aree debolmente connesse.

5.1.4 Bordi, Contorni e Componenti Connesse

In questa sottosezione verranno solamente illustrate le definizioni di bordo, contorno e componente connessa applicati nel mondo dell'elaborazione di immagini, poiché gli algoritmi per la loro ricerca, oltre ad essere piuttosto complessi per essere illustrati in questo frangente, non servirebbero a comprendere meglio l'implementazione del sistema.

I **bordi** di un immagine a livelli di grigio sono i punti estremi del gradiente dell'immagine, nella sua direzione. Essi sono quindi un concetto che è volto a trovare discontinuità nell'immagine, ovvero dove la luminosità dell'immagine cambia bruscamente. Possono essere sfruttati per ottenere la sagoma di separazione tra un oggetto all'interno dell'immagine e lo sfondo, o tra l'oggetto ed altri oggetti. Il bordo risulta in molte situazioni indispensabile per poter interpretare forme geometriche all'interno dell'immagine.

I **contorni** sono invece applicati in immagini binarie, e sono definiti come l'insieme dei pixel con distanza unitaria dal background.

Le **componenti connesse** sono aree di foregorund, separate tra loro, e che non hanno elementi di congiunzione o intersezione. La seguente figura mostra un esempio chiarificante di cosa siano le componenti connesse: ogni chicco di riso colorato di un colore differente è una singola componente connessa, ossia un'entità a se stante, che non entra in conflitto con nessun altra.

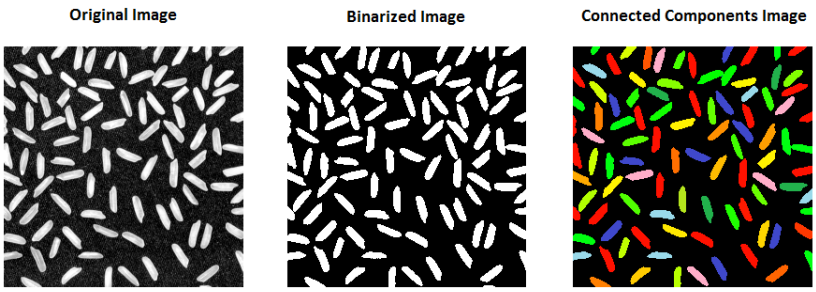

Figura 5.6: Componenti connesse di un immagine.

5.1.5 OCR - Optical Character Recognition

Gli algoritmi di *Optical Character Recognition* o riconoscimento ottico dei caratteri, sono dedicati alla conversione di un immagine contenente caratteri in testo vero e proprio.
I primi sistemi *OCR* per funzionare necessitavano di esempi di immagini contenenti i caratteri corrispondenti a quelli che devono essere riconosciuti. In questo modo essi venivano calibrati rispetto alla tipologia di caratteri da analizzare, da ideogrammi a codici alfanumerici. Questi primi algoritmi si avvalevano di tecniche di *pattern matching* o *pattern recognition*. In pratica sovrapponevano le immagini di esempio fornite in ogni punto dell'immagine, e ne calcolavano il grado di correlazione tra di esse. Se risultavano abbastanza similari, allora veniva riconosciuto il carattere corrispondente alla maschera sovrapposta. Chiaramente essi sono stati ottimizzati in vari modi, per esempio analizzando l'immagine in più stadi, partendo inizialmente da basse risoluzioni per poi, alla fine, comparare con l'immagine originale solamente i caratteri che, dalle precedenti analisi, si sono rivelati essere più promettenti. Comunque anche con tutti questi accorgimenti i tentativi per cui provare la sovrapposizione delle immagini sono davvero innumerevoli, considerando anche tutte le possibili rotazioni e distorsioni che il testo nell'immagine può avere. Per questo motivo, oggi vengono utilizzati software adoperanti algoritmi che, avvalendosi anche di apprendimenti automatici attraverso modelli di reti neurali, sono in grado di riconoscere i contorni dei caratteri e di ricostruire oltre al testo anche la formattazione della pagina.

5.2 Implementazione

La parte di software utilizzata per il riconoscimento dei parametri vitali ed il loro invio al web server è stata interamente sviluppata in Java. La struttura del programma è stata suddivisa in più *package*, contenente classi ed interfacce riguardanti ambiti distinti del progetto. Essi sono stati raggruppati nel seguente modo:

- **Camera.** Questo package contiene le classi e le interfacce che hanno il compito di dialogare con la telecamera installata nel sistema. In particolare, la camera è stata utilizzata per prelevare frame ad intervalli regolari, contenenti gli screenshot del monitor da analizzare.
- **Sender.** Qui si trovano gli elementi necessari per l'invio dei dati analizzati al web server.
- **Core.** Il package *Core* come dice la parola stessa contiene tutti i file necessari per svolgere il compito al cuore dell'intero progetto, ossia il rilevamento e l'analisi dei dati del monitor. All'interno è presente un altro package, chiamato **Analysis**, che contiene, più in specifico, le classi volte ad analizzare le immagini del monitor.
- **Test.** Le classi che hanno l'incarico di innestare e gestire tutto il sistema si trovano qui. Anch'esso è suddiviso in due sotto-package: **Data Analyzer** in cui è presente una versione del programma senza interfaccia grafica; **Camera Calibration** in cui invece i risultati dell'analisi vengono esposti in una *GUI* dedicata.

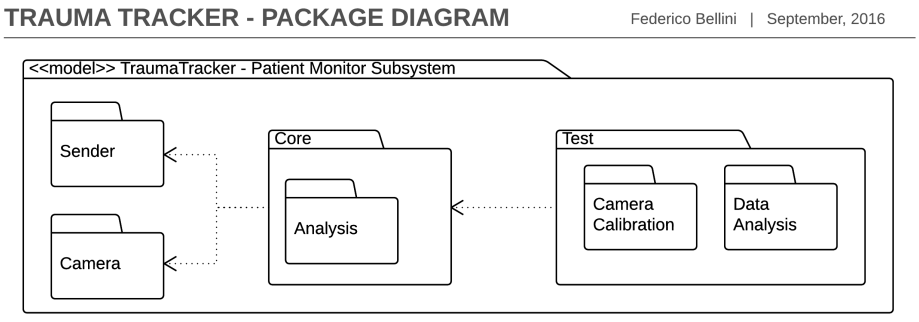

Figura 5.7: Package Diagram Patient Monitor Subsystem.

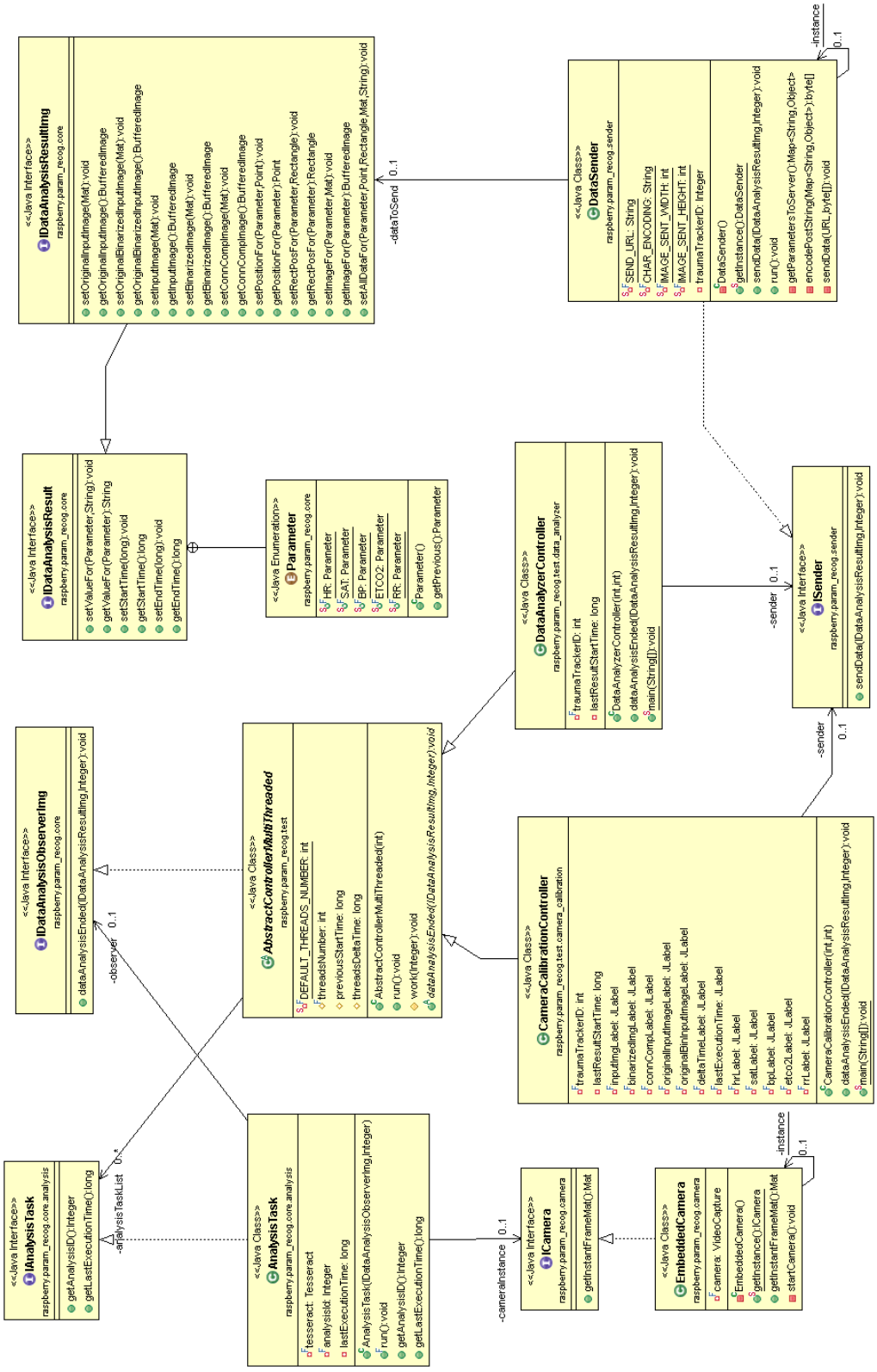

Figura 5.8: Class Diagram Patient Monitor Subsystem.

La Figura 5.8 mostra il diagramma UML delle classi. In questo diagramma sono mostrate solo le classi e le interfacce più significative. Come si può notare, esistono due classi contenenti il metodo *main*: *CameraCalibrationController* e *DataAnalyzerController*. Questa scelta è stata fatta per dare la possibilità di eseguire il programma rispettivamente con o senza interfaccia grafica. L'utilizzo di tale interfaccia è stato utile soprattutto in fase di debug e testing dell'applicazione, per calibrare la telecamera sul bersaglio da inquadrare. Si può notare dal diagramma che entrambe estendono dalla classe astratta *AbstractControllerMultiThreaded*, la quale estende a sua volta dalla classe *java.lang.Thread*. Come si può già intuire dal nome, il compito di questa classe è implementare il comportamento del *Controller* dell'applicazione, gestendo inoltre in maniera efficiente i worker threads che hanno il compito di svolgere parallelamente l'analisi dell'immagine, e lasciando il compito della visualizzazione dei risultati alle sottoclassi. Questo viene fatto implementando l'interfaccia *IDataAnalysisObserverImg*, dichiarando così di essere un *Observer* della procedura di analisi, ma poi lasciando astratto il metodo *dataAnalysisEnded*, richiamato dall'analisi al momento della conclusione della stessa, e forzando così le sue sottoclassi a prendersene carico.

```
public abstract class AbstractControllerMultiThreaded extends Thread
      implements IDataAnalysisObserverImg {

   private static final int DEFAULT_THREADS_NUMBER = 3;
   protected final int threadsNumber;
   protected long previousStartTime = System.currentTimeMillis();
   protected long threadsDeltaTime;
   private final List<IAnalysisTask> analysisTaskList;

   [ ... ]

   @Override
   public void run() {
      // Start working all threads
      for (int i = 0; i < this.threadsNumber; i++) {
         this.work(i);
      }
   }
```

```
  protected synchronized void work(final Integer analysisID) {
     final IAnalysisTask completedAnalysis = analysisTaskList
           .get(analysisID);
     final Thread worker = new Thread(completedAnalysis);

     this.threadsDeltaTime = completedAnalysis.getLastExecutionTime()
           / this.threadsNumber;
     final long currTime = System.currentTimeMillis();
     final long waitTime = this.previousStartTime + threadsDeltaTime
           - currTime;
     this.previousStartTime = currTime;
     if (waitTime > 0) {
        try {
           this.wait(waitTime);
        } catch (InterruptedException e) {
           System.out.println(e.toString());
        }
     }
     worker.start();
  }

  @Override
  public abstract void dataAnalysisEnded(final IDataAnalysisResultImg
      result,
        final Integer id);

}
```

Listato 5.1: Abstract Controller Multi Threaded

Come si può vedere dal Listato 5.1, la classe raffigurata presenta un metodo protetto e sincronizzato *work*, utilizzato per far partire i differenti threads paralleli che eseguono il compito di analizzare simultaneamente più screenshot del monitor, ottimizzando e rendendo più dinamico l'intero processo di analisi. Questo metodo è stato implementato in modo da distribuire equamente nel tempo questi threads. Per fare un esempio, se l'analisi completa di un'immagine richiedesse due secondi per l'elaborazione totale, e i threads dedicati a tale compito dovessero essere due, allora questo metodo si assicurerebbe che ogni thread sia fatto partire a distanza di un secondo dopo l'avvio del prece-

dente. Tutto questo viene calcolato dinamicamente durante l'esecuzione del programma, tenendo conto anche delle possibili variazioni nel tempo impiegato per l'analisi, adattandosi ad esse. Lo scopo di questa gestione parallela dei threads, è innanzitutto quello di sfruttare al massimo le risorse del sistema, e in secondo luogo di ridurre il più possibile il tempo di campionamento delle immagini dello schermo. Infatti se tutti i threads impegnati nel compito di analisi fossero fatti partire assieme, si ritroverebbero tutti ad analizzare frame temporalmente molto vicini tra loro, peggiorando sensibilmente la qualità dei risultati delle analisi. L'architettura utilizzata in questa parte è del tipo Master-Worker, in cui un singolo elemento master organizza e gestisce i differenti worker del sistema.

Per l'implementazione delle funzioni di analisi delle immagini sono state utilizzate le librerie Java di *OpenCV* [6] per tutte le funzioni legate al *Computer Vision*, e di *Tessseract* [7] per gli algoritmi di OCR. L'utilizzo delle classi presenti in queste librerie, è stato confinato nelle classi di analisi, garantendo in questo modo il pieno disaccoppiamento di queste librerie con il resto del programma. In questo modo, se si volesse implementare nuovamente il sistema senza fare uso di tali librerie, non sarebbe necessario riscrivere tutto il programma, ma solo le parti di analisi interessate. Questa separazione è stata possibile grazie anche alle classi contenenti i dati elaborati dall'analisi. Queste classi implementano le interfacce *IDataAnalysisResult* e *IDataAnalysisResultImg*, e vengono istanziate dalla classe *AnalysisTask*, per poi essere passate come risultato dell'analisi richiamando il metodo *dataAnalysisEnded*, notificando in questo modo a tutti gli osservatori che l'analisi si è conclusa e comunicandogli i risultati. Come si può notare dalla Figura 5.8, queste due interfacce presentano metodi come per esempio *setOriginalInputImage* utilizzato per settare l'immagine di input, che prende in ingresso un oggetto di tipo *Mat*, ossia la forma di rappresentazione delle immagini utilizzata dalle librerie *OpenCV*, e con il suo corrispondente getter, *getOriginalInputImage*, restituisce l'immagine settata con un oggetto di tipo *java.awt.image.BufferedImage*, classe presente di default nelle librerie Java. E' anche interessante notare che le classi che implementano tali interfacce sono state pensate per essere indipendenti dal

[6] http://opencv.org/
[7] http://tess4j.sourceforge.net/

numero di parametri presenti sul monitor, adattandosi dinamicamente ai loro possibili cambiamenti futuri. Il seguente listato mostra una parte della classe *DataAnalysisResult*, e come si può vedere i valori vengono salvati in una mappa chiave-valore dipendente solamente dall'enumerazione *Parameter*, con la quale vengono definiti i parametri presenti sullo schermo del monitor.

```
public class DataAnalysisResult implements IDataAnalysisResult {

   private static final String PARAM_N_D = "nd";

   private final Map<Parameter, String> valMap = new
       EnumMap<>(Parameter.class);

   @Override
   public void setValueFor(final Parameter param, final String val) {
      valMap.put(param, val);
   }

   @Override
   public String getValueFor(final Parameter param) {
      final String output = valMap.get(param);
      if (output == null || output.length() == 0) {
         return PARAM_N_D;
      }
      return output;
   }

   [ ... ]

}
```

Listato 5.2: Data Analysis Result

A causa dell'unicità hardware della telecamera, la classe Java che la descrive è implementata come *Singleton*, in quanto non avrebbe senso l'esistenza contemporanea di più istanze di questo oggetto. Le stesse considerazioni valgono anche per la classe *DataSender*, poiché la socket con cui comunica con il server è unica, e quindi non avrebbe senso creare più istanze dell'oggetto dal momento che un solo pacchetto di dati per vota può essere spedito. Per evitare

corse critiche e race-condition sulle risorse condivise da questi due oggetti, essi sono stati implementati in modo da essere thread-safe, garantendo così in ogni punto del codice la mutua esclusione dei threads.

```
public final class EmbeddedCamera implements ICamera {

   private final VideoCapture camera;
   private static EmbeddedCamera instance;

   private EmbeddedCamera() {
      if (!this.camera.isOpened()) {
         this.startCamera();
      }
   }

   public static ICamera getInstance() {
      synchronized (EmbeddedCamera.class) {
         if (instance == null) {
            instance = new EmbeddedCamera();
         }
         return instance;
      }
   }

   [ ... ]

}
```

Listato 5.3: Embedded Camera

Infine, gli ultimi elementi presenti nel diagramma delle classi di Figura 5.8, non ancora approfonditi, sono l'*AnalysisTask* e l'*IAnalysisTask* da cui estende. La classe appena indicata svolge il compito di eseguire una singola analisi sull'immagine catturata dalla telecamera, individuando la posizione dei parametri vitali e riconoscendone i valori. Di seguito verrà esposto passo per passo l'algoritmo implementato, utilizzato per raggiungere questo scopo. Per comprendere meglio i vari passaggi effettuati, di seguito viene mostrata un'immagine che raffigura l'interfaccia grafica dell'applicazione durante un test di analisi.

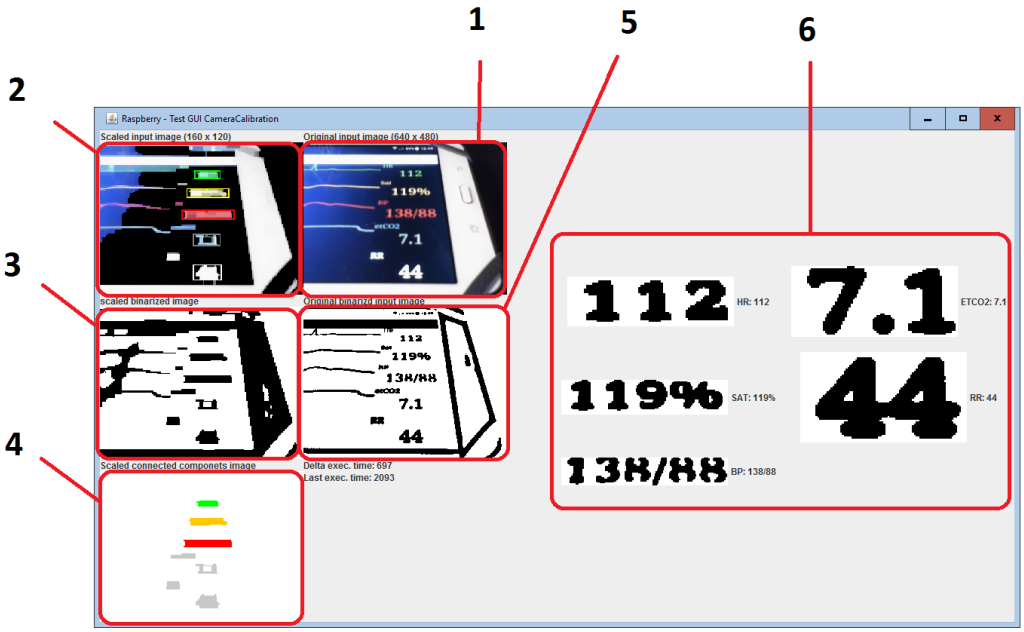

Figura 5.9: Interfaccia grafica di TraumaTracker.

Per semplicità l'algoritmo può essere schematizzato in nove passaggi:

1. **Acquisizione dell'immagine** dalla fotocamera. L'immagine acquisita è contraddistinta in Figura 5.9 dal numero uno.

2. **Riduzione della risoluzione** dell'immagine originale per rendere più veloce ed efficiente il calcolo dei prossimi passaggi.

3. **Oscuramento dei pixel** con luminosità inferiore alla luminosità media dell'immagine. Questo passaggio aiuta a rimuovere rumore e possibili elementi che potrebbero disturbare i prossimi passaggi di analisi.

4. **Chiusura morfologica** dell'immagine, utilizzando un elemento strutturante molto largo e poco alto. In questo modo l'area coperta dai valori viene ingrandita, definendone bene i contorni dell'area su cui essi risiedono. Il risultato di questi ultimi tre passaggi è mostrato in Figura 5.9, contrassegnato con il numero due. Inoltre questo passaggio vine effettuato per migliorare il riconoscimento dei colori delle aree, che verrà eseguito in seguito.

5. **Binarizzazione** con soglia locale dell'immagine appena prodotta al passo precedente. Il risultato è contraddistinto dal numero tre.

6. Calcolo delle **componenti connesse** sull'ultima immagine prodotta. Qui vengono filtrate le componenti che hanno un'area troppo grande o troppo piccola per poter rappresentare i valori dei parametri. Infine, esse vengono colorate con la tinta predominante dell'area della componente. Per il calcolo del colore predominante viene utilizzata la media degli *Hue* su cui la componente connessa giace, prendendo come riferimento la porzione di immagine originale codificata in *HSV*. Il risultato di tutte le operazioni, finora effettuate, è mostrato nel rettangolo numerato con il quattro.

7. Utilizzando l'immagine del punto precedente come maschera, e avvalendosi dell'uso delle componenti connesse calcolate in precedenza, vengono ritrovate le **aree di ricopertura** dai parametri nell'immagine originale. I risultati di questi calcoli sono mostrati nell'immagine due di Figura 5.9, contraddistinti da un rettangolo dello stesso colore di quello riconosciuto. Come si può notare, l'accuratezza del calcolo di una buona maschera nei passi precedenti porta ad ottimi risultati nel rintracciamento della posizione dei parametri.

8. Ora per ogni area rintracciata al passo precedente, viene **ritagliata la porzione di immagine corrispondente** nell'immagine originale a risoluzione maggiore, dopo che è stata binarizzata. Le porzioni di immagini ritagliate sono visibili nel riquadro sei della Figura 5.9.

9. Come ultimo passo dell'algoritmo, su questi ritagli di immagine vengono applicati gli algoritmi di **riconoscimento dei caratteri**. Il risultato prodotto da tale operazione su di ogni frammento di immagine viene visualizzato alla destra della stessa, visibile sempre nel sesto riquadro.

Durante tutta la fase implementativa è sempre stato tenuto conto dei possibili sviluppi futuri di questo progetto. La programmazione è stata eseguita cercando di garantire la maggior flessibilità possibile in termini di maggior riutilizzo di codice in caso di cambiamenti. Si è anche sempre tenuto conto della possibilità della comparsa di malfunzionamenti, dovuti per esempio

all'oscurazione della videocamera, alla lettura non riuscita dei dati, alla possibile perdita di connessione e tanti altri, cercando così di fornire la maggior robustezza possibile al software prodotto.

5.2.1 Problemi riscontrati in fase implementativa

Diversi problemi sono stati riscontrati durante la fase di implementazione del sottosistema. In particolare le maggiori complicazioni si sono verificate nella parte di individuazione dei parametri sul monitor. Infatti è stato necessario utilizzare tecniche come il calcolo della maschera mediante l'utilizzo delle componenti connesse per ottenere risultati migliori in termini del colore rilevato. Questi problemi sono sorti soprattutto a causa dell'utilizzo di schermi differenti, aventi luminosità e contrasti diversi tra loro, ed utilizzati in ambienti che talvolta presentavano punti di luce riflessa che rendevano l'acquisizione dei parametri ancora più difficile. Nonostante queste problematiche, il sistema è sufficientemente stabile in situazioni normali per garantire una corretta valutazione dei parametri. Nel seguente capitolo verrà mostrato il comportamento del sistema in diverse situazioni di luminosità e riflessi.

Un'altra complicazione sorta durante il processo implementativo è stata l'impossibilità di trovare le librerie native compilate per sistemi Linux ARM, necessarie a garantire il funzionamento delle classi Java di OpenCV e Tesseract utilizzate sul sistema Raspberry. Per risolvere questo inconveniente è stato necessario compilare direttamente sul microprocessore i sorgenti originali delle librerie, generando così i driver richiesti per l'esecuzione del programma.

Capitolo 6

Demo

In questo capitolo verranno effettuate differenti simulazioni in un ambiente controllato, per mostrare il comportamento del sistema sviluppato nelle diverse condizioni. Va sempre ricordato che nonostante gli ottimi risultati, il progetto Trauma Tracker non è ancora pronto per essere applicato in maniera definitiva all'ambiente ospedaliero reale, ma gli obiettivi posti per questa versione del progetto, ossia per il raggiungimento di una versione volta a dimostrare la fattibilità di un sistema di questo tipo, sono stati ampiamente raggiunti.

6.1 Test del Patient Monitor Subsystem

Il *Patient Monitor Subsystem* è stato collaudato in tutte le possibili condizioni in cui esso può imbattersi, cercando di non tralasciare nessun caso.
Per presentare il funzionamento del sistema nelle diverse situazioni in cui è stato esaminato, verrà mostrato lo screenshot dell'interfaccia grafica del programma, già descritta nella Sezione 5.2, dalla quale si può dedurre il suo comportamento.

La seguente figura mostra i risultati ottenuti dal sistema, applicato in condizioni ideali per il funzionamento.

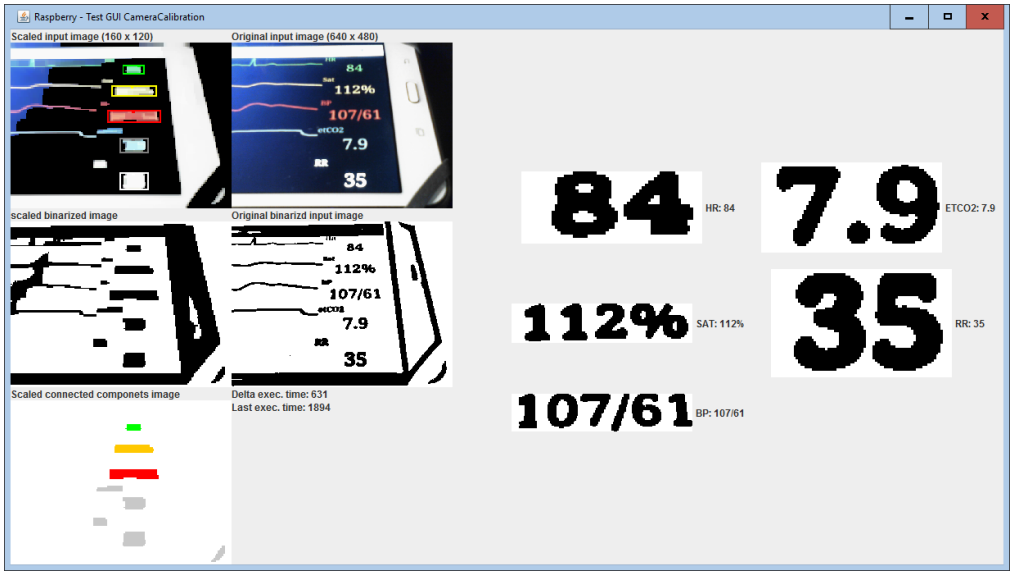

Figura 6.1: Condizioni ottimali per il funzionamento.

Al contrario il sistema si comporta correttamente anche nelle condizioni pessime di applicazione, senza generare errori o eccezioni, anche quando nessun parametro viene rilevato.

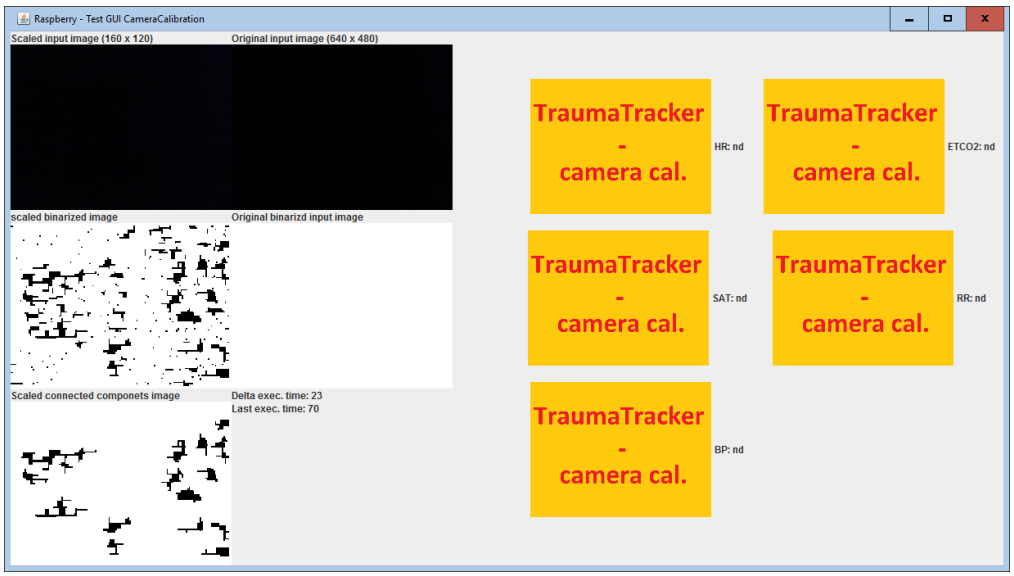

Figura 6.2: Condizioni pessime.

Il sistema reagisce correttamente anche quando solamente metà dello schermo contenente i parametri viene inquadrato. Come ci si aspetterebbe, esso riconosce solamente i parametri presenti nella metà mostrata.

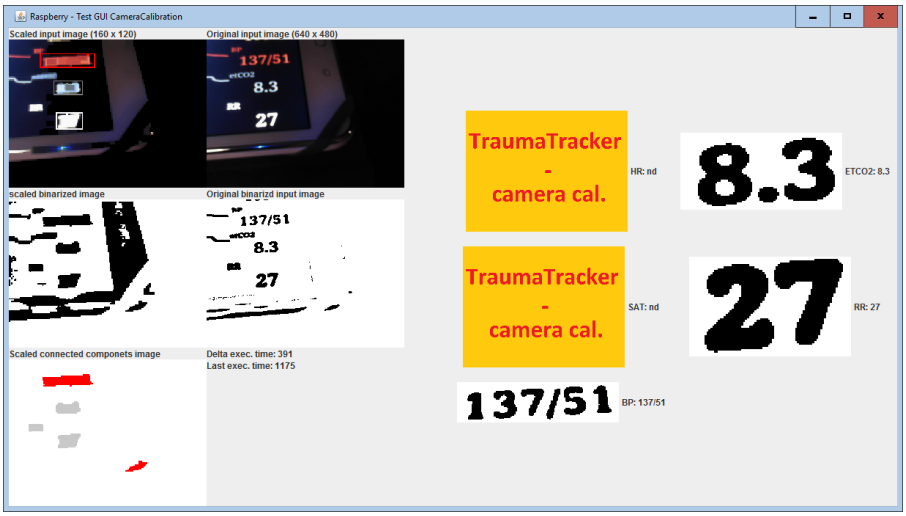

Figura 6.3: Schermo parzialmente inquadrato.

Anche nel caso in cui il valore di un parametro venisse solo parzialmente inquadrato, l'algoritmo di riconoscimento procederebbe con l'analisi. La seguente figura mostra come l'ultimo parametro contenente il valore '41' viene inquadrato solo per metà, ma comunque è stato correttamente interpretato.

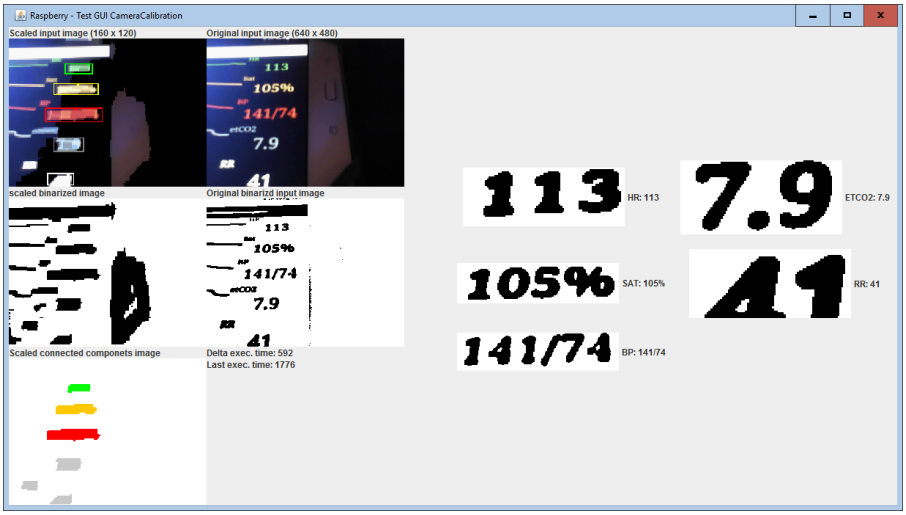

Figura 6.4: Numero tagliato.

Il sistema reagisce positivamente anche inquadrando il monitor multiparametrico da differenti angolazioni. La Figura 6.5 mostra il comportamento del sistema in questo frangente. Si può notare inoltre come la rilevazione della saturazione (in giallo) non abbia avuto successo, ma i parametri seguenti vengono comunque collocati correttamente, grazie all'analisi non solo dei colori, ma anche del layout e della formattazione dei parametri all'interno del monitor.
Inoltre dalla stessa immagine si può constatare che il frame del monitor è stato catturato nel momento in cui il valore dell'ultimo parametro stava subendo una modifica. Il sistema in questo caso procede comunque con il rilevamento dei caratteri, restituendo un valore chiaramente errato. Il compito di dare significato ai valori, rintracciando errori di questo genere dovuti a problemi fisici, è affidato al web server, il quale riceve sempre tutti i dati estratti e cerca di interpretarli, garantendo così una migliore qualità delle informazioni.

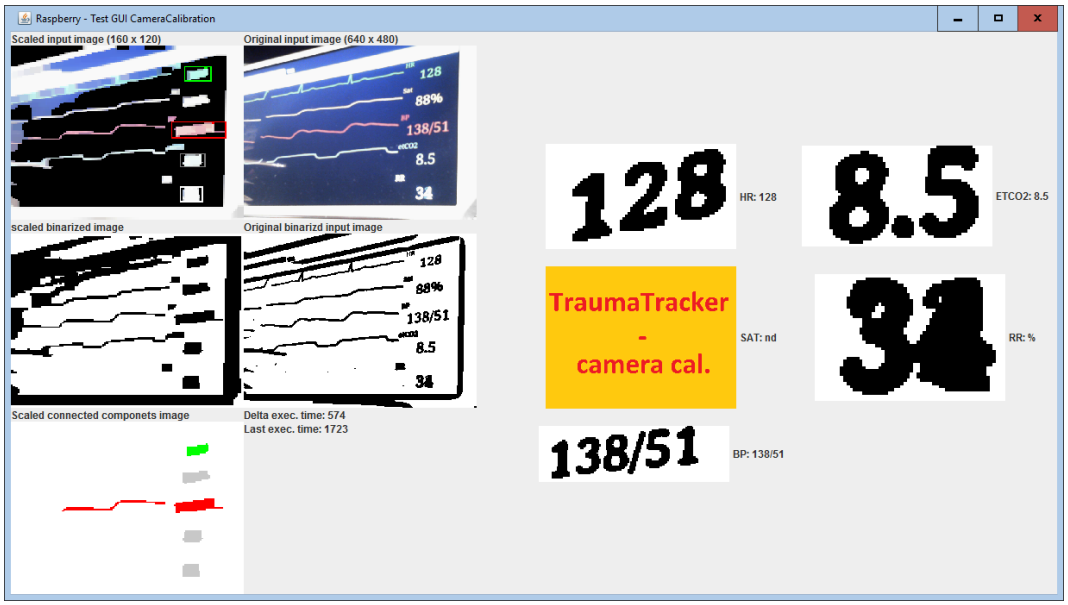

Figura 6.5: Inquadratura angolata, parametro non interpretato e valore che sta mutando.

Anche quando l'inquadratura dello schermo viene ruotata, fino ad un certo angolo di rotazione, i numeri vengono comunque correttamente interpretati. La figura riportata di seguito mostra l'angolo di rotazione massimo per cui sono sempre garantiti ottimi risultati. Per angoli superiori a questo i valori vengono più frequentemente interpretati non correttamente.

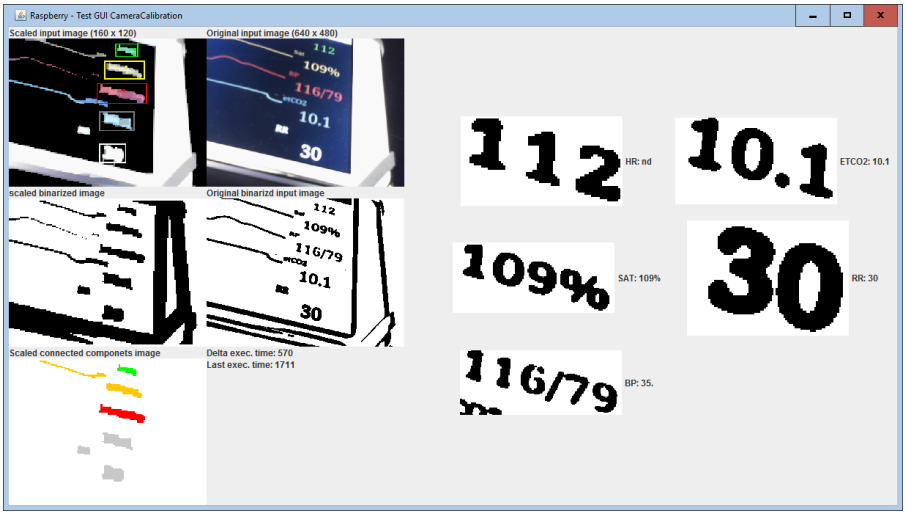

Figura 6.6: Inquadratura ruotata.

Con l'utilizzo di differenti font per la visualizzazione dei dati nel monitor multiparametrico, il sistema è comunque funzionante.

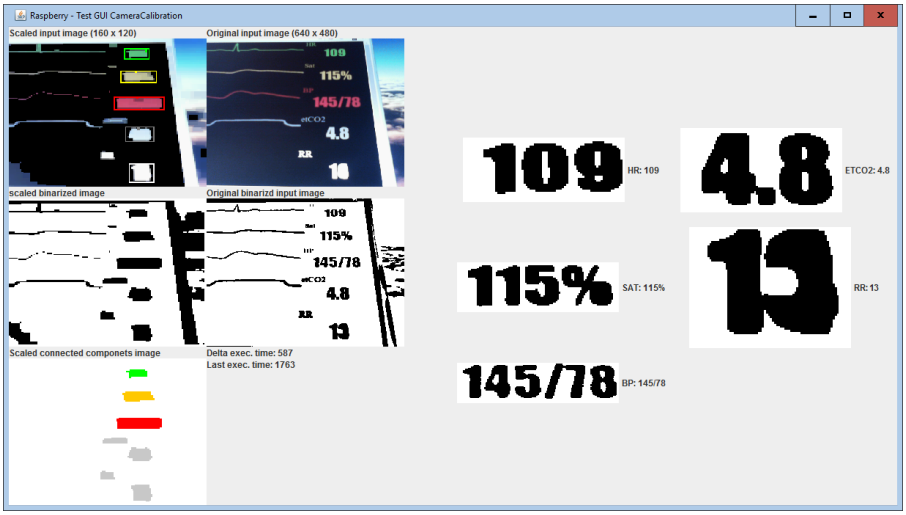

Figura 6.7: Utilizzo di font differente.

La Figura 6.8 e la Figura 6.9 mostrano il sistema testato durante condizioni di luminosità ambientale rispettivamente molto bassa e molto alta. Si può notare da entrambe le immagini che il sistema risponde correttamente al variare di tale parametro.

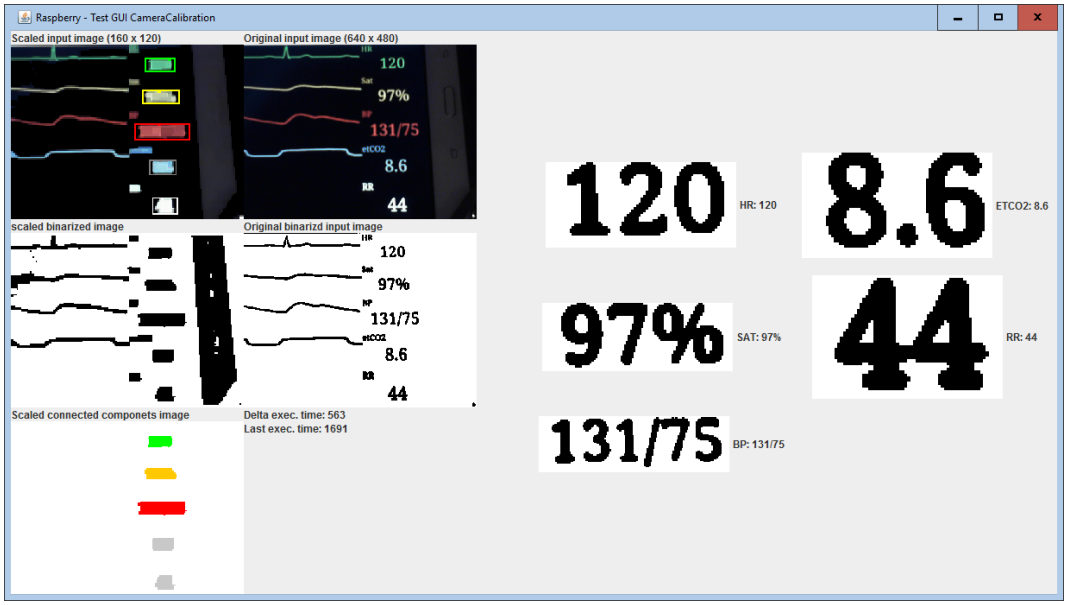

Figura 6.8: Bassa luminosità ambientale.

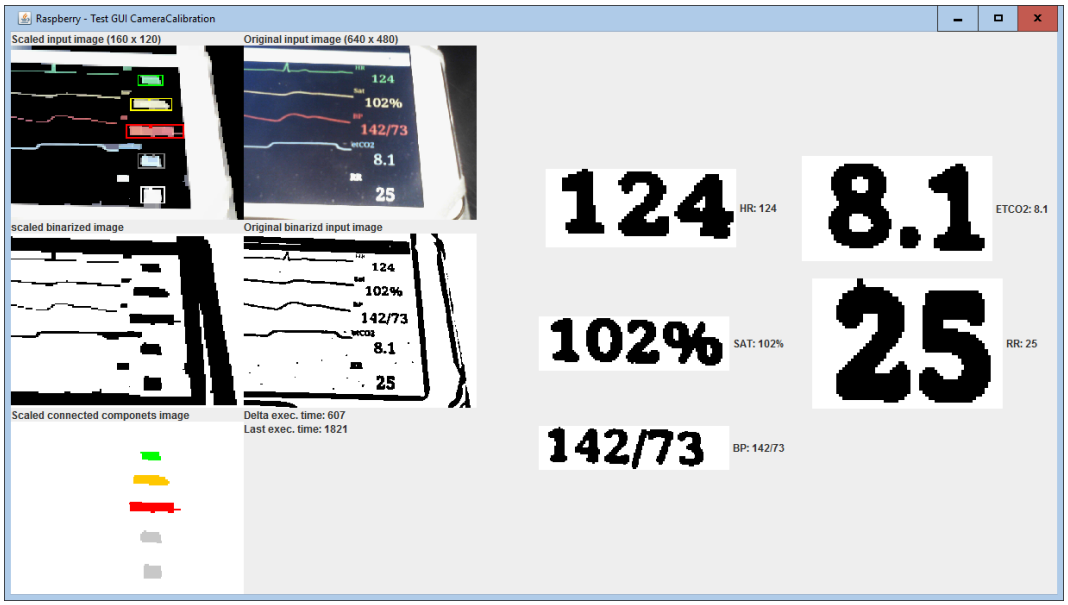

Figura 6.9: Alta luminosità ambientale.

Il sistema si comporta ottimamente anche quando le condizioni di luminosità sono tali da presentare riflessi sul monitor. Come mostra le figura riportata di seguito, i parametri che vengono nettamente cancellati dalla presenza del riflesso non vengono interpretati, ma quelli che restano parzialmente visibili vengono riconosciuti correttamente.

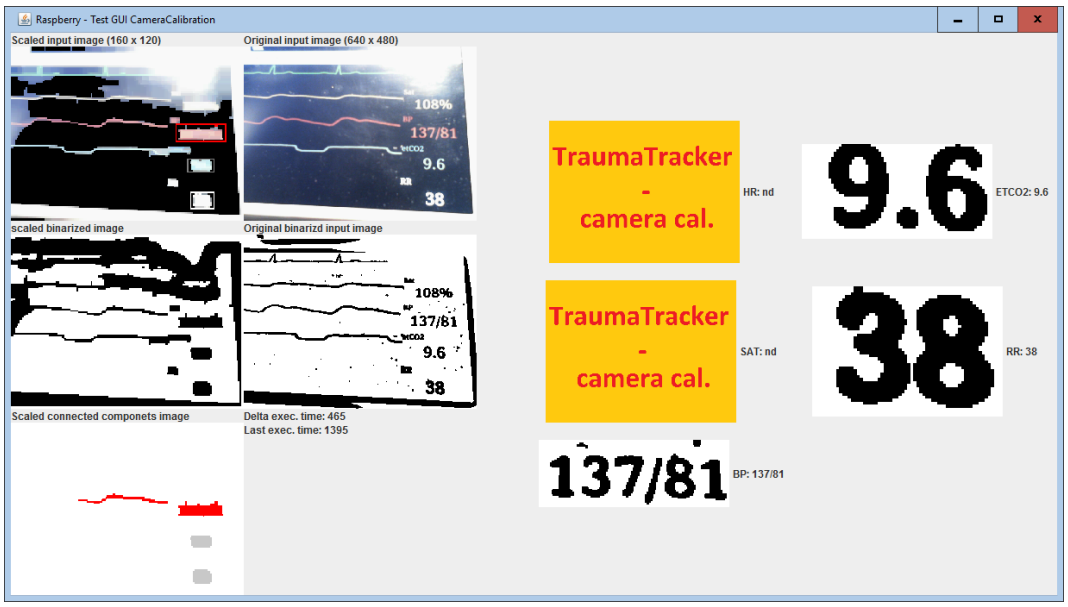

Figura 6.10: Presenza di riflessi.

I test sono stati eseguiti anche per verificare il funzionamento del sistema nel rilevamento di ogni possibile cifra e carattere presente nel monitor ('%', '.' e '/'). Inoltre il sistema è stato verificato anche in condizioni in cui i parametri presentavano più cifre, sia decimali che non, e non sono state rilevate anomalie nel suo comportamento.

E' possibile visualizzare due video dimostrativi, girati con lo scopo di presentare la costruzione di questo sottosistema e valutarne le prestazioni [1] [2]. La Figura 6.11 mostra un frame prelevato dai video, in cui viene esposta la costruzione del sistema: si può notare a sinistra uno schermo raffigurante la simulazione del monitor multi-parametrico, e una telecamera collegata tramite USB al Raspberry Pi 3, posta di fronte ad esso. Sulla destra invece sono

[1] Video test 1: https://youtu.be/znEdgY1kNqE
[2] Video test 2: https://youtu.be/FcK61gQfyus

posizionati un postatile, collegato via SSH per mezzo di un cavo Ethernet al microprocessore, che mostra l'interfaccia grafica del programma in esecuzione sul Raspberry, e un tablet contenente i dati rilevati dal microprocessore, inviati al web server ed infine giunti su di esso. Come si può notare dai video, questo programma Android è stato creato espressamente per visualizzare i dati ricevuti dal server, aiutandoci così nelle fasi di debug e testing del sistema. Nella parte centrale della schermata, rappresentata nel tablet, vengono raffigurati i valori riconosciuti e, nella parte sottostante, l'immagine sulla quale sono stati applicati gli algoritmi di localizzazione e riconoscimento dei parametri.

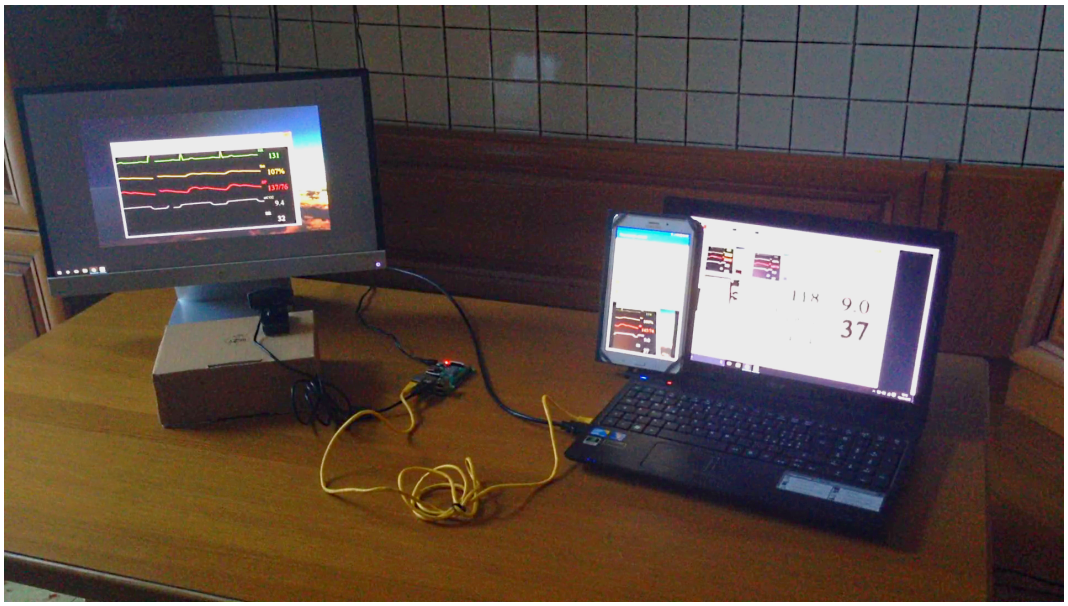

Figura 6.11: Costruzione test sottosistema.

Il primo video di test realizzato espone inizialmente la costruzione del sottosistema, in seguito l'avvio del software sul Raspberry e infine confronta i dati rilevati da Raspberry con quelli presenti sul tablet Android.
Il secondo video mostra invece il comportamento del sistema a seguito di piccoli spostamenti e rotazioni della telecamera rispetto al monitor da rilevare, e negli ultimi minuti espone l'interfaccia non grafica del software in esecuzione sul microprocessore.

6.2 Affidabilità

L'affidabilità del *Patient Monitor Subsystem* è stata verificata a fondo, come si è anche potuto vedere nella sezione precedente di testing. Il sistema si è rivelato molto affidabile in situazioni standard, e adeguatamente sicuro anche in condizioni di utilizzo diverse da quelle normali. Le problematiche maggiori sorgono nel momento in cui si presentano riflessi sul monitor da rilevare, dovuti a condizioni ambientali sfavorevoli. Il sistema garantisce anche in queste situazioni prestazioni sufficienti per la rilevazione dei parametri, anche se fortemente limitate dell'estensione e dall'intensità del riflesso sul monitor.

6.3 Valutazione prestazioni

Come si può notare anche dai video presentati nella Sezione 6.1, il tempo totale che intercorre tra l'acquisizione di un'immagine dello schermo, alla sua visualizzazione sul tablet, è mediamente compreso tra i cinque e i sei secondi. Gli algoritmi di *OCR* applicati all'immagine, incidono per quasi la totalità dell'intervallo trascorso: è stato stimato che appena cento millisecondi vengono impiegati per il trasferimento dei dati dal microprocessore al tablet e 150 millisecondi circa sono necessari per il calcolo delle varie maschere da sovrapporre all'immagine per la localizzazione dei parametri. Il tempo restante è utilizzato proprio per il riconoscimento dei caratteri.
Nonostante questo arco temporale possa essere considerato enorme dal punto di vista informatico, da quello di medici ed infermieri è stato valutato come assolutamente accettabile, e ci è stato assicurato che, in questo ambito di applicazione, latenze di questo genere sono all'ordine del giorno e possono essere considerate trascurabili. Inoltre bisogna sempre ricordare che il progetto è stato concepito come un *Proof of Concept*, e in versioni future, dove queste latenze costituiranno invece l'elemento centrale degli studi, si potrà sicuramente arrivare ad una gestione *real-time* degli eventi.

Conclusioni e Sviluppi futuri

L'obiettivo della presente tesi è stato lo sviluppo di un sistema *Hands-Free* applicato in una realtà *Healthcare*, volto ad aiutare il personale sanitario nello svolgimento delle mansioni lavorative. In particolare il sistema in questione, commissionato da medici ed infermieri dell'ospedale Maurizio Bufalini di Cesena, si prende carico della compilazione e produzione del report finale contenente tutte le operazioni svolte sui pazienti nell'ambito del Pronto Soccorso. Durante le fasi di sviluppo e progettazione sono state aggiunte ulteriori funzionalità al sistema, come per esempio la possibilità di richiamare un parametro vitale da parte dell'utilizzatore, visualizzandolo sugli smart-glass. L'introduzione di queste nuove particolarità ha reso il sistema un vero e proprio *oggetto incantato*[16], in grado di esibire proprietà finora inimmaginabili in questo campo di applicazione.

In questa tesi è stato approfondito in particolare il sottosistema utilizzato per il riconoscimento dei parametri vitali dal monitor multi-parametrico. Esso ha richiesto lunghe fasi di implementazione e collaudo per ottenere dei risultati soddisfacenti, che alla fine sono stati raggiunti. Nel capitolo finale è inoltre possibile vedere il frutto di questi studi, applicato in situazioni controllate, volte a verificare il funzionamento del sottosistema in ogni possibile condizione.

Tenendo sempre a mente che il progetto è stato concepito fin dall'inizio come un *Proof of Concept*, ossia è stato sviluppato per verificare se le idee alla sua base possano realmente essere concretizzabili ed utili una volta applicate nel mondo reale, si può concludere che l'obiettivo primario di questa tesi è stato pienamente raggiunto. Purtroppo però, a causa di rallentamenti burocratici, sommati ad altri fattori contrastanti, non è stato possibile per il momento testare il sistema nella sua interezza in un ambiente ospedaliero reale.

Gli sviluppi futuri di questo progetto mirano alla creazione di un vasto sistema multi-agente, composto da più sistemi Trauma Tracker che collaborano tra loro per assistere in maniera ancora più consistente medici ed infermieri nello svolgimento delle loro mansioni. Le potenzialità che una rete di sistemi come questo può offrire sono infinite: dalla possibilità di fruire di dati dei pazienti da ogni luogo, a quella di assistere in remoto ad operazioni chirurgiche, grazie alla telemedicina.

Recentemente è stato organizzato da parte dei medici responsabili della collaborazione con la nostra università, un incontro a cui tutti abbiamo potuto prendere parte, in cui ci è stata data la possibilità di conoscere i rappresentanti della *Drager* [3], ossia l'azienda fornitrice dei macchinari di cui l'ospedale si serve. In questa occasione è stato possibile delineare quello che si prospetta essere il prossimo passo verso una realizzazione più concreta di Trauma Tracker: grazie ad un gateway fornitoci dall'azienda, sarà possibile usufruire direttamente dei dati rilevati dai macchinari, senza la necessità di doverli interpretare dal monitor. In questo modo si potrà completamente astrarre da possibili problematiche dovute alla lettura errata dei parametri vitali, aggiungendo in questo modo maggiore sicurezza ed affidabilità al sistema.

In conclusione il sistema creato è stato molto utile per testare la fattibilità dell'idea, e per verificarne la sua applicabilità in uno scenario reale e critico come il Pronto Soccorso. Esso necessiterà ancora di molti sviluppi prima di poter essere approvato per un utilizzo permanente in ospedale, ma i primi passi per raggiungere questo ambizioso obiettivo sono stati compiuti.

[3]Drager: `http://www.draeger.com/sites/it_it/Pages/default.aspx`

Ringraziamenti

Il primo immenso ringraziamento è rivolto a tutta la mia famiglia e agli affetti a me più cari, che mi hanno sempre aiutato e sostenuto nella vita e nella carriera universitaria.

Un profondissimo grazie anche a tutti i miei amici, senza i quali non avrei mai avuto la forza e la grinta per completare questo mio grande cammino.

Un sentito ringraziamento al Professor Alessandro Ricci, che mi ha dato la possibilità di svolgere questo lavoro su un argomento così interessante ed immerso nella realtà, fornendomi indispensabili aiuti e garantendo sempre la sua presenza, anche a distanza. Ringrazio per la sua disponibilità, pazienza e professionalità il Professor Alessandro Bevilacqua, che ha seguito lo sviluppo della tesi e mi ha fornito preziosi suggerimenti. Inoltre desidero ringraziare anche l'Ingegner Angelo Croatti, per aver condiviso le sue conoscenze in materia e avermi affiancato nello svolgimento del progetto.

Infine vorrei esprimere la mia gratitudine verso i professori del Liceo Scientifico di Forlì, che mi hanno aiutato a edificare le basi della persona che sono oggi, facendomi capire davvero quale fosse la strada che dovevo percorrere nella vita.

Bibliografia

[1] Nikola Tesla, *Collier's Illustrated Weekly, Volume 77*, Crowell-Collier Publishing Company, 1926.

[2] Gerd Kortuem, Fahim Kawsar, Vasughi Sundramoorthy, *Smart objects as building blocks for the Internet of things*, IEEE Internet Computing (Volume: 14, Issue: 1), Jan.-Feb. 2010, `http://ieeexplore.ieee.org/document/5342399/`.

[3] Wikipedia, *Realtà aumentata*, 2016, `https://it.wikipedia.org/wiki/Realt%C3%A0_aumentata#cite_note-1`.

[4] Andreas Bulling, Ozan Cakmakci, Kai Kunze and James M. Rehg, *Eyewear Computing – Augmenting the Human with Head-mounted Wearable Assistants*, Schloss Dagstuhl, 24 – 29. Januar 2016, Dagstuhl Seminar 16042, `http://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=16042`.

[5] Seth Schiesel, *A Home System Leaves Hand Controls in the Dust, page C1 of the New York edition.* , New York Times, 4 November 2010, `http://www.nytimes.com/2010/11/04/arts/television/04kinect.html`.

[6] Bernadette Tansey, *Augmedix Plans To Expand Google Glass-Based System For Healthcare*, XConomy, 28 April 2016, http://www.xconomy.com/san-francisco/2016/04/28/augmedix-plans-to-expand-google-glass-based-system-for-healthcare/.

[7] Brian Dolan, *Augmedix gets 3.2 million to bring Google Glass to doctors*, Mobi Health News, 24 March 2014, http://mobihealthnews.com/31314/augmedix-gets-3-2-million-to-bring-google-glass-to-doctors/.

[8] Jonah Comstock, *Rhode Island Hospital ER begins Google Glass dermatology study*, Mobi Health News, 11 March 2014, http://mobihealthnews.com/30855/rhode-island-hospital-er-begins-google-glass-dermatology-study/.

[9] Pei Jia, *Head gesture recognition for hands free control of an intelligent wheelchair*, Emerald Group Publishing Limited, January 2007.

[10] Steve Feng, Romain Caire, Bingen Cortazar, Mehmet Turan, Andrew Wong, Aydogan Ozcan, *Immunochromatographic Diagnostic Test Analysis Using Google Glass*, American Chemical Society, February 2014, http://pubs.acs.org/doi/pdf/10.1021/nn500614k.

[11] Oliver J. Muensterer, Martin Lacher, Christoph Zoeller, Matthew Bronstein, Joachim Kübler, *Google Glass in pediatric surgery: An exploratory study*, International Journal of Surgery, February 2014, http://www.journal-surgery.net/article/S1743-9191(14)00040-5/pdf.

[12] Bob Zemke, *How Google Glass Will Transform Healthcare*, Extreme Networks, 6 March 2015, http://www.extremenetworks.com/how-google-glass-will-transform-healthcare/.

[13] Tracie White, *Medical student's startup uses Google Glass to improve patient-physician relationship*, Stanford Medicine, 9 February 2015, https://med.stanford.edu/news/all-news/2015/02/medical-students-startup-uses-google-glass.html.

[14] Philips Healthcare, *Experience the future of wearable technology*, Philips Healthcare, October 2013, http://www.usa.philips.com/healthcare/innovation/research-and-exploration/google-glass-and-intellivue.

[15] R. C. Gonzalez, R. E. Woods, *Digital image processing Third Edition*, Prentice Hall, 2008.

[16] David Rose, *Enchanted Objects: Innovation, Design, and the Future of Technology*, Scribner, 2015.