ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE DELL'INFORMAZIONE

SISTEMA DI MONITORAGGIO MODULARE ATTRAVERSO TECNOLOGIE MOBILE PER LA SICUREZZA IN AMBIENTE DOMESTICO

Relazione finale in: Sistemi Multimediali

Relatore:
Chiar.ma Prof.ssa Paola Salomoni

Presentata da:
Monti Lorenzo

Correlatore:
Dott.ssa Catia Prandi

Sessione III
Anno accademico 2012/2013

Indice

Al mio Babbo e alla mia Mamma
genitori che si meritano a pieno titolo l'iniziale maiuscola

E ad Oxana
che mi ha sempre supportato e SOPPORTATO

Per l'uomo, una vita non meditata non è davvero degna di essere vissuta .

Bertrand Russell
tratto da "La saggezza dell'Occidente"

Introduzione

Con questo progetto ci si propone di creare un'applicazione per i device mobili intelligenti, in particolare smartphone e smartwatch, con sistema operativo Android, in grado di rilevare una caduta con il conseguente invio automatico di un messaggio di allarme mirato a richiamare i soccorsi in tempo reale. Data la funzionalità, l'app è stata principalmente progettata per un utente di età avanzata. L' intento non è del tutto nuovo, dato che sul mercato sono già presenti varie tipologie di device indossabili, come quelli integrati in una cintura, posti in vita o basati su sensori posti sotto l'ascella. In particolare ci sono 3 principali categorie di tecnologie rivolte a questo proposito che utilizzano diversi device: la camera, device di tipo ambientale ed infine device indossabili. Questi dispositivi restano opinabilmente molto invasivi se teniamo conto dell'utente interessato, ovvero la persona anziana; per non parlare dei costi aggiuntivi rispetto all'applicazione qui proposta scaricabile gratuitamente tramite smartphone o smartwatch direttamente dallo store di Google Play. Basta pensare all'aspetto intrusivo ed invadente della camera, che va ad intaccare la privacy dell'utente interessato, o ancora alle soluzioni ambientali che comportano l'installazione di appositi device nel pavimento, ancora una volta limitativi nella funzione svolta, in quanto circoscritti all'area d'installazione. Da questa inquadratura, la soluzione del device mobile con l'aggiuntivo applicativo proposto in questo progetto sembra risolvere sia i problemi d'intrusività sia quelli riguardanti l'aspetto economico. A tal riguardo vengono considerati tre studi di rilievo in tale ambito, quello di Pérolle del 2002 [21], integrato con il localizzatore GPS, di Hwang del 2004 [22] basato sulla comunicazione bluetooth ed infine lo studio di Bourke del 2007 [30], il cui algoritmo ci siamo permessi di adottare per lo sviluppo del nostro progetto, in quanto il più completo e sicuramente maggiormente adatto per l'applicabilità smartphone e smartwatch. L'autore in questo studio propone lo sviluppo di un sistema di controllo della persona che vive da sola e che quindi, corre il rischio di infortunio, anche grave, attraverso un dispositivo posto alla cintura con un sistema basato su accelerometro tri-assiale. In questa direzione, al fine di perfezionare la nostra applicazione ci siamo preoccupati di analizzare alcune delle app di fall detection già presenti nello store cercando di rafforzare ed ottimizzarne le funzionalità laddove fossero risultate approssimative od addirittura assenti. In

particolare sono state analizzate le seguenti app: la T3LAB Fall Detector, la CRADAR, ovvero Crash Detection And Response o ancora la Fall Detector , la Smart Fall Detection.

Sembra opportuno a questo punto trattare della scelta dell'utente, la persona di età avanzata, che rappresenta statisticamente l'individuo maggiormente predisposto al rischio di caduta, le cadute all'interno dell'ambiente domestico sono il 43,7% delle cause di infortunio delle persone anziane, nonché come dagli ultimi dati ISTAT in aumento demografico (riportati dal sito www.istat.it), sono stimati circa 16 milioni di over 65, cioè il 26,5% della popolazione totale italiana. Risulta quindi, scontato che la soluzione proposta sia un sistema valido di soccorso capace di intervenire in caso di malori o incidenti domestici portando così un vantaggio sia per gli utenti stessi che per la società, in quanto rendono non più necessarie le strutture adeguate con assistenza continua. Si intenda bene come per vantaggio risulti implicito sia l'aspetto fisico, che riguarda la presenza materiale di una persona pronta a soccorrere l'individuo anziano in caso di necessità, sia l'aspetto economico non di minor importanza. In effetti, da alcuni studi condotti in tale ambito, sembra che nel 2020 i costi relativi alle conseguenze della cadute negli anziani tenderanno a superare i 32 miliardi di dollari. Il problema riferito all'assistenza degli anziani, che con l'avanzare dell'età diventano sempre meno autosufficienti, viene affrontato ricorrendo a soluzioni di assistenza domiciliare, come ad esempio badanti o al trasferimento in strutture di ricovero permanente. Sebbene questa spesso possa rappresentare l'unica alternativa possibile, essa non sempre può risultare efficiente in quanto può provocare disagi nella persona anziana, oltre ad avere un costo cospicuo. E' quindi necessario pensare a soluzioni alternative che permettano alle persone anziane con un grado di indipendenza ancora elevato di poter rimanere nella propria casa in sicurezza. Un altro vantaggio dell'applicazione, fin da subito evidente è la tempestività con la quale, in caso di bisogno, il sistema di soccorso si pone. Fino ad oggi, tra le tecnologie che troviamo in commercio entrano in azione solamente se la persona bisognosa di cure è in grado di agire sul dispositivo di allarme. Con questo nuovo paradigma di apparati, l'invio della richiesta di soccorso viene velocizzata grazie ad un sistema che automaticamente avvisa in caso di emergenza, allertando le persone che son state pre-memorizzate all'interno dell'apparecchio. Spostando l'attenzione sulle tecnologie utilizzate per implementare l'app mobile e

l'architettura scelta, è di dovere mostrare il sistema operativo mobile utilizzato, ovvero Android, il quale rappresenta una soluzione ottimale da un lato per l'ampia fascia di mercato su cui risulta presente, come ricavato dallo studio condotto nel 2013 dall'istituto Gartner, ne consegue infatti che nello stesso anno questo sistema operativo, abbia coperto quasi l'80% dei device venduti nel mondo, dall'altro lato per l'approccio open source. E' chiaro che prima di arrivare a tale conclusione ci ben guardati dal non sottovalutare

gli altri sistemi operativi analizzandone attentamente le caratteristiche, come dell'Apple iOS e del Windows Phone. Nello specifico del sistema operativo Android abbiamo utilizzato l'ambiente di sviluppo Eclipse con il plug-in ADT.

L'aspetto più considerevole capace di dare un'impronta del tutto innovativa all'interno della nostra area di ricerca, è dato dalla modularità dell'applicazione, che consente di ridurre i problemi di falsi negativi e di gestire qualche verifica da remoto. Ricollegandoci agli aspetti vincenti di concorrenza sul Google Play Store l'aspetto modulare non tarda sicuramente ad avvantaggiarne la competitività dell'applicazione in merito. Modularità presente, ma non obbligatoria, infatti l'applicazione opera anche nel caso in cui l'utente è sprovvisto di smartwatch che consente l'utilizzo modulare del servizio applicativo. Facendo riferimento all'aspetto implementativo, sono stati utilizzati come già in precedenza nominati, due dispositivi per la creazione dell'applicativo, uno smartphone Samsung Galaxy Note III ed uno smartwatch, il Samsung Galaxy Gear, mentre per la fase di debug ed i campionamenti sono stati utilizzati 3 dispositivi mobile sempre del marchio Samsung. Per valutare realmente la funzionalità dell'applicazione sono stati eseguiti dei test con 14 volontari di età fra i 18 ed i 60 anni per un totale di 112 campionamenti, suddivisi in 8 situazioni differenti per persona: *mentre ci si stende sul letto, mentre ci si siede in poltrona, mentre ci si siede in una sedia, mentre ci si siede al bagno, mentre ci si siede in macchina, mentre ci si siede nel letto, mentre si salgono le scale, mentre si cammina per circa 10 metri.* Questa serie di stati con i relativi esiti della reazione applicativa hanno infine dimostrato l'attendibilità dell'applicazione proposta pari al 100%, in quanto non è mai stata rilevata durante i test alcuna caduta laddove non vi fosse realmente avvenuta. Anticipando il contenuto della tesi, i punti fondamentali trattati al suo interno possono essere riassunti come segue. Nel primo capitolo vengono evidenziati i punti fondamentali del problema delle cadute negli

anziani che l'applicativo proposto cerca di risolvere, sottolineandone le cause e le conseguenze in base a studi statistici. Restando all'interno del primo capitolo, vengono spiegate in seguito le tecnologie assistenziali oggi esistenti sul mercato. In seguito, nel secondo capitolo viene affrontato il tema degli strumenti tecnologici ed informatici utilizzati alla costruzione vera e propria dell'applicazione, offrendo un quadro completo di quelli che sono i sistemi operativi mobili emergenti evidenziando e motivando la scelta fatta per poi indirizzarsi verso l'ambiente di sviluppo e la stesura dell'algoritmo utilizzato. Nel terzo capitolo invece, prende spazio la fase implementativa che mostra i device utilizzati, i campionari e soprattutto mette in evidenza ulteriormente l'aspetto modulare dell'intera applicazione proposta.

Capitolo 1

Panoramica generale

In questo capitolo verranno presentate le criticità inerenti le cadute degli anziani, sia come problematica sociale che economica. Inizialmente saranno presentate le statistiche epidemiologiche, poi verranno descritte le principali cause che portano alla caduta dell'anziano ed infine, le conseguenze che ne derivano. Verranno inoltre poi esposti i progressi raggiunti nell'attività di monitoraggio e riconoscimento della caduta, attraverso una panoramica sulle tecnologie assistenziali, e sui diversi dispositivi disponibili nel mercato che vengono utilizzati per questo scopo. In particolare, visto la grande diffusione che hanno avuto negli ultimi anni (e stanno continuano ad avere) i device mobili intelligenti (smartphone, tablet, smartwatch, ecc.), saranno presentate varie applicazioni mobile che permettono di riconoscere la caduta e i chiamare tempestivamente i soccorsi.

1.1 Epidemiologia delle cadute

Le cadute nelle persone anziane indicano una criticità rilevante di sanità pubblica in termini di costi. Nel 1994 i costi relativi superavano i 20 miliardi di dollari ed è stimato che entro il 2020 i costi supereranno i 32 miliardi [1]. Queste cadute per le persone anziane sono eventi ordinari e i rischi associati aumentano in maniera esponenziale all’avanzare dell’età in entrambi i sessi e in tutti i gruppi etnici. Si stima infatti che una persona su tre con età superiore ai 65 anni e una persona su due di oltre 80 anni cade in maniera incidentale almeno una volta l'anno [2], e in circa il 10% si presenta un danno fisico rilevante. Negli Stati Uniti tre cadute accidentali su quattro sono effettuate da ultra sessantenni (che costituiscono il 13% della popolazione): possiamo dunque affermare che si tratta di un problema sostanzialmente geriatrico [4].
Sappiamo inoltre, che circa il cinquanta percento dei casi ospedalizzati dopo una caduta

non sopravvive più di un anno. Molti studi basati sulle popolazioni descrivono l'epidemiologia delle cadute negli anziani in differenti contesti e con variazioni significative. Per gli anziani dai 65 anni ed oltre, che vivono in comunità sono molto bassi i tassi di incidenza (30-160 per 100 persone all'anno, in media 65 per 100) .

Molte di queste cadute non causano lesioni importanti, infatti solo il 5% riscontra una frattura o si deve recare in ospedale. Per gli anziani ricoverati in residenze assistite hanno tassi di incidenza verso le cadute molto più alti, anche la percentuale delle complicanze è più alta e di conseguenza più seria, il 10-25% delle cadute si trasformano in fratture, ferite, tagli o lesioni. E' dimostrato inoltre che il tasso di cadute e le complicanze associate aumentano in maniera rilevante con l'età e raddoppiano nel caso in cui l'anziano abbia più di 75 anni [4]. Le cause relative alle cadute degli anziani sono molteplici, per poterlo spiegare in maniera più agevole riportiamo una tabella riassuntiva di ben 12 studi inerenti alle cadute di persone anziane in diversi contesti [12]

Cause	**percentuale media (1)**	**range (2)**
accidentale - correlata all'ambiente	31%	1-53%
disturbo di andatura, riduzione della forza muscolare	17%	4-39%
capogiri, vertigini	13%	0-30%
Collasso	9%	0-52%
confusione	5%	0-14%
ipotensione posturale	3%	0-24%
disturbo visivo	2%	0-5%
sincope	0,3%	0-3%
altre cause specificate (3)	15%	2-39%
cause sconosciute	5%	0-21%

1 = Percentuale media calcolata da 3.628 cadute in 12 studi.

2 = Range tra il minimo e il massimo.

3 = Qui si include artrite, malattie acute, farmaci, alcol, dolore, epilessia e caduta dal letto.

La prima considerazione da fare è che un sostanzioso 30-50% delle cadute avviene in maniera accidentale in ambiente domestico. Dobbiamo precisare che le cadute accidentali sono in realtà il risultato di un interazione fra l'ostacolo presente nell'ambiente domestico e quelle che possono essere le condizioni psico-fisiche di un anziano. Infatti possiamo affermare che gli anziani hanno un'andatura più rigida, meno coordinata e più pericolosa rispetto ad una persona giovane. L'uomo è dotato della cosiddetta strategia di conservazione dell'equilibrio che in età avanzata si tramuta notevolmente. Per una persona giovane è naturale rispondere rapidamente ad una caduta spostando il peso sull'anca, mentre un anziano perde gradualmente la prontezza di riflessi esponendosi maggiormente ai rischi connessi alle cadute. Dobbiamo inoltre tener conto che riduzioni della vista, dell'udito e della memoria sicuramente giocano un ruolo non meno importante nella probabilità di inciampare. I problemi di andatura e forza muscolare presenti in una persona di età avanzata condizionano maggiormente l'abilità di quest'ultimo nel salvarsi da una caduta. Sappiamo che l'equilibrio viene assicurato da diversi fattori bio-meccanici, come la mobilità articolare, il giusto timing della contrazione muscolare, l'appropriata intensità della contrazione muscolare ed il normale input sensoriale, inclusi vista, propriocezione e sistema vestibolare. I problemi di deambulazione sono dovuti anche a determinate disfunzioni del sistema nervoso, muscolo-scheletrico, circolatorio e respiratorio come lunghi periodi di scarsa attività.Un'altra importante causa la possiamo trovare nel cosiddetto capogiro, sintomo comune negli anziani derivabile da diversi fattori, come i disturbi cardiovascolari, l'iperventilazione, ipotensione ortostatica, effetti collaterali da farmaci, ansia e depressione.

Prendiamo per esempio l'ipotensione ortostatica, dovuta ad un calo di pressione sistolica nel momento in cui la persona si alza in piedi (da posizione distesa a quella eretta). Possiamo imputarla ad una disfunzione autonomica (spesso in relazione all'età, al diabete o a danni cerebrali), ipovolemia, bassa gittata cardiaca, morbo di Parkinson, patologie metaboliche ed endocrine, uso di farmaci. Nonostante sia molto comune negli

anziani, l'ipotensione ortostatica non rappresenta un grosso rischio per le cadute, in quanto chi ne è affetto riesce ad evitare la caduta essendosi abituato a questa sindrome.

Diversamente accade per il collasso, il quale determina una caduta afflosciante e non rovinosa, senza perdita di coscienza o capogiri. Si presenta come un'improvvisa debolezza delle gambe, nella maggior parte dei casi transitoria ma che può anche persistere per diverso tempo. Data la maggior precisione diagnostica, oggi il collasso è molto meno segnalato, infatti è piuttosto raro. La sincope invece si manifesta con la perdita di coscienza dovuta solitamente alla riduzione del flusso ematico cerebrale o di fattori metabolici. Alcuni studi la considerano causa possibile tra il 2% e il 10% delle cadute, altri studi invece negano questa correlazione. Altre cause da considerare possono essere le seguenti: disturbi del sistema nervoso centrale, deficit cognitivi, deficit della vista, effetti collaterali dei farmaci, consumo di alcol, anemia, ipotiroidismo, articolazioni instabili, problemi dei piedi, osteoporosi severa con fratture spontanee e malattie acute.

Condizioni funzionali	**Significativi/ Totale (1)**	**RR - OR medi (2)**	**Range**
debolezza	11/11	4.9 (8)	1.9-10.3
deficit equilibrio	9/9	3.2 (5)	1.6-5.4
deficit andatura	8/9	3.0 (5)	1.7-4.8
deficit vista	5/9	2.8 (9)	1.1-7.4
limitazioni della mobilità	9/9	2.5 (8)	1.0-5.3
deficit cognitive	4/8	2.4 (5)	2.0-4.7
stato funzionale compromesso	5/6	2.0 (4)	1.0-3.1
ipotensione posturale	2/7	1.9 (5)	1.0-3.4

1= numero di studi in cui l'associazione è risultata significativa diviso il numero di studi che ha preso in esame il fattore

2 = Rischio Relativo (dagli studi prospettici), Odds Ratios (dagli studi retrospettivi). Tra parentesi il numero di studi che riportavano il RR oppure l'OR.

Molti ricercatori di epidemiologia considerano più efficace individuare le condizioni funzionali che favoriscono le cadute al posto che focalizzarsi semplicemente sulle cause di queste. Riportando nella pagina precedente la tabella comprensiva di 16 studi controllati, possiamo notare che le principali condizioni funzionali volte a facilitare la caduta degli anziani sono la perdita graduale della forza muscolare ed i problemi di deambulazione ed equilibrio. Studi riportano che questi fattori sono responsabili non solo di un'elevata percentuale di caduta verificata, ma anche di importanti conseguenze, come ad esempio le fratture. Caratteristiche come andatura ed equilibrio, possono essere sottoposte ad analisi che ne controlla l'efficienza, per esempio test come "timed up and go" o il "Tinetti" specifici per la verifica dell'andatura e dell'equilibrio. Secondo gli studi prospettici, i fattori di rischio relativi alle cadute negli anziani possono essere raggruppati in 5 categorie. In particolare, la prima riguarda gli aspetti socio-demografici. Infatti, il fattore più evidente, che va in ogni caso preso in considerazione è proprio quello dell'età avanzata. Questo perché il rischio di caduta negli anziani aumenta con l'avanzare dell'età. Non meno importanti, ma difficilmente intuibili sono i fattori collegati al sesso ed alla mancanza del coniuge. E' comprovato infatti, che le donne sono più a rischio degli uomini allo stesso modo in cui gli uomini risultano maggiormente esposti a tale rischio se ospitati in una casa di riposo, dove questi tendono ad essere più spesso malati o disabili. Tornando alle condizioni funzionali, sono stati individuati 6 fattori principali di caduta negli anziani, alterazioni della marcia, problemi di mobilità, alterazioni dell'equilibrio, debolezza muscolare, deficit della vista, limitazioni funzionali. Anche lo stato mentale ha un'importanza elevata nell'abilità della persona anziana di evitare le cadute o comunque di reagire in maniera positiva. Indubbiamente, da un punto di vista psichico l'anziano presenta un deterioramento cognitivo, ma si riscontra anche maggiore esposizione a casi di depressione. Bisogna però precisare che alcuni studi escludono questo fattore. Sono da considerare cause della caduta degli anziani anche le malattie, come l'osteoartrite, il Parkinson, l'ictus, il diabete, l'incontinenza urinaria, le vertigini, l'ipotensione ortostatica, come già detto in precedenza. Molte di queste sono da considerare come fattori di causalità della caduta a

prescindere dall'età, ma nel nostro caso rappresentano le malattie in maggior misura presenti nell'uomo di età avanzata che giocano un ruolo di privilegio nella caduta di questo. Indirettamente a contribuire alla probabilità di caduta di un anziano sono i farmaci di politerapia o gli psicofarmaci che possono influenzare le prestazioni tanto fisiche quanto psichiche. Un uso improprio dei farmaci, l'assunzione per esempio di una dose che supera quella consigliata può compromettere varie funzionalità dell'anziano. In particolare le reazioni avverse sono presenti negli anziani intorno al 5% quando si assume un solo farmaco e arriva a quasi 100% quando se ne assumono dieci o di più. Un altro fattore può essere quello dell'inattività fisica. Infatti, una persona attiva fisicamente aiuta a mantenere saldi elementi quali l'equilibrio, la forza muscolare, la coordinazione ed i riflessi. Analogamente una persona inattiva risulta più esposta ad un'eventuale caduta. Dopo aver valutato le diverse cause e condizioni funzionali è di maggior rilievo per la mia tesi sapere dove gli anziani cadono. Secondo i risultati conseguiti dallo studio Argento [5] in *Figura 1.1* si può afferire che gli anziani cadono il 48% delle volte fuori casa. Dentro casa, gli ambienti a maggior rischio invece sono riportati nel grafico qui di seguito.

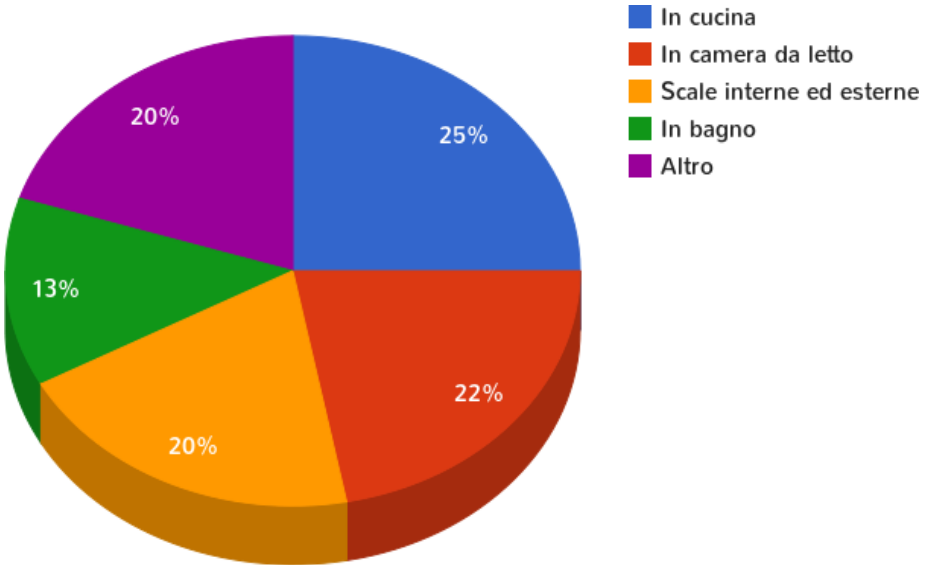


Figura 1.1: Grafico torta inerente le statistiche delle studio Argento [5]

1.2 Conseguenze delle cadute

Sappiamo che le cadute per persone ultra 65enni sono la prima causa di ricovero

ospedaliero che rappresenta il 4% di tutti i ricoveri in questa fascia di età. Sappiamo inoltre, che passata la soglia dei 40 anni, le cadute aumentano progressivamente ogni anno del 4,5% per gli uomini, mentre per le donne del 7,9%. Arrivando all'età di 85 anni, i dati statistici inerenti alle cadute ci rivelano che per gli uomini la soglia scende al 4% annuo, mentre per le donne scende al 7% anno[11]. Riportiamo qui di seguito il grafico con l'andamento (*Figura 1.2*):

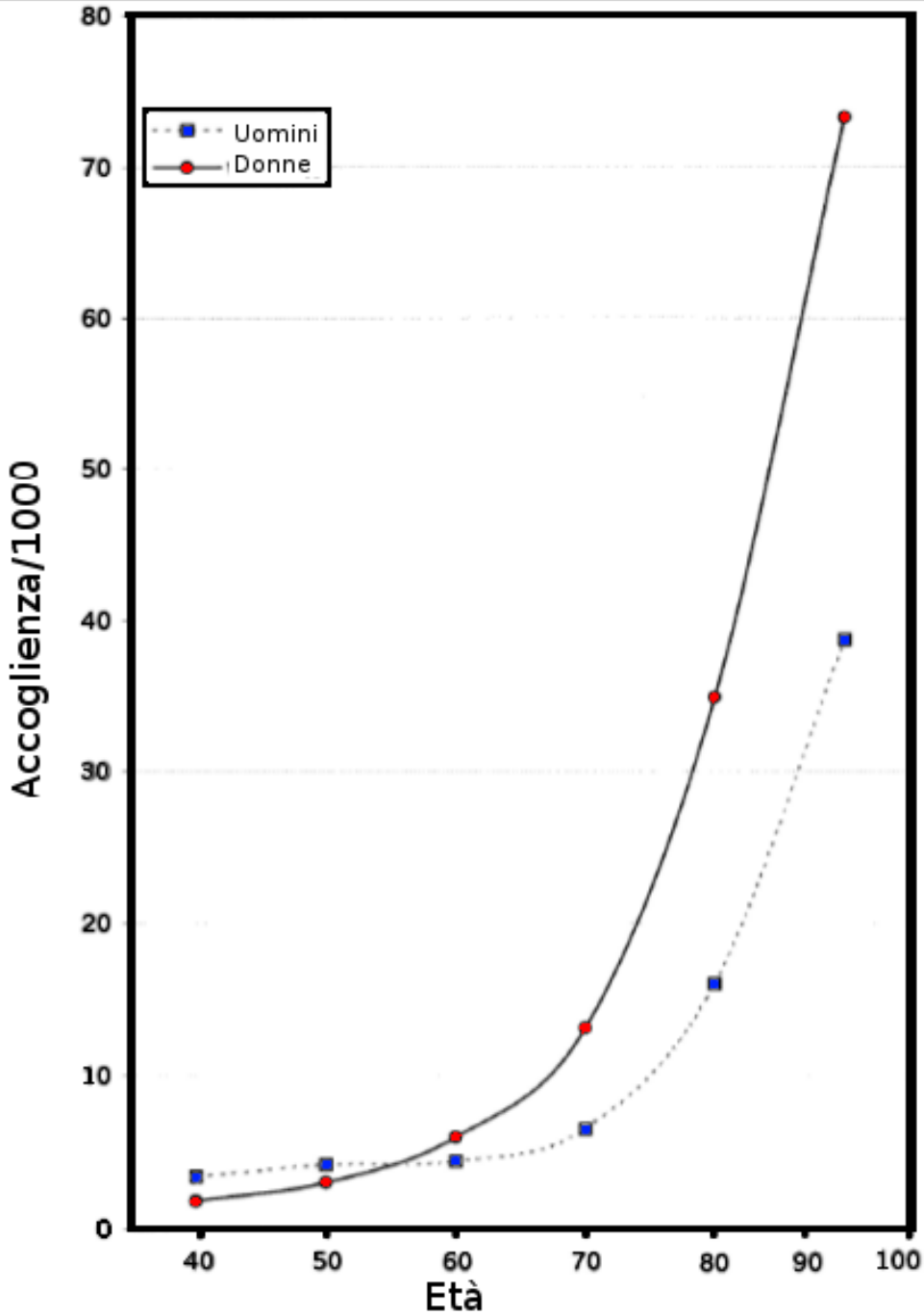


Figura 1.2: Andamento cadute in base all'età

Come Rubenstein [12] espone nei suoi studi, su quaranta anziani che cadono solamente uno viene ospedalizzato, ma di questi, soltanto la metà sopravvive più di un anno. Infatti, sappiamo che, se un anziano cade più volte in un arco temporale di sei mesi, questo deve essere attentamente valutato da esperti per capire quali siano state le cause e conseguentemente rispondere in maniera adeguata. Le conseguenze di queste cadute possono essere molteplici e diversificate anche nella gravità. Alcuni esempi possono essere: contusioni, ematomi, ferite lacero-contuse, fratture, traumi cranici, post fall syndrome, riduzione dell'attività motoria, riduzione del livello di autonomia, aumento del rischio di ospedalizzazione e istituzionalizzazione, fino alla morte. Recentemente è stato condotto, in Inghilterra, uno studio retrospettivo, il quale afferma che in 500 strutture per anziani, in una casistica pari al 64,7% dei casi, non vi è riscontro di conseguenze degne di rilievo. Infatti, in rapporto all'insorgenza le conseguenze possono essere precoci o tardive. La morte, come evento immediato è raro, ma frequente come successione ritardata, anche in relazione all'età del soggetto, alla tempestività dei soccorsi e in ultima istanza alle condizioni di salute del paziente.

L'incidente in molteplici casi è l'elemento che crea l'effetto domino (focolai bronco-pneumonici, decubiti, trombosi venosa profonda (TVP) ed embolia polmonare (EP), sindrome da immobilizzazione), il quale è responsabile dell'exitus. Per Wieman la prima causa di morte dopo i 70 anni è da attribuirsi alle cadute.

Sappiamo inoltre che la probabilità di fratture sono al 10% dei casi circa come conseguenza di una caduta. Il soggetto di età avanzata, a causa di una maggiore fragilità ossea (osteoporosi, osteomalacia, Morbo di Paget, metastasi ossee) è maggiormente a rischio di fratture, e questo costituisce un problema sanitario rilevante per i Paesi industrializzati. Come già discusso precedentemente in questa tesi, è risaputo che ogni anno si verificano 500.000 nuovi casi, questo impone un onere economico stimato sui 4 miliardi di euro solamente per le cure ospedaliere e si ritiene che questi aumenteranno a 750.000 nuovi casi/anno per i prossimi anni, fino ad arrivare a 1.000.000 nel 2050. È giusto inoltre mettere il luce che nell'anno successivo all'intervento i costi verranno raddoppiati per le spese di riabilitazione, visite specialistiche e visite inerenti l'invalidità sociale conseguente. Nell'anziano, le fratture, oltre ad un incremento della mortalità, determinano un peggioramento della qualità della vita, chiamata in gergo ADL (Activity

of Daily Living) e conseguentemente un ricovero in una residenza sanitaria assistenziale (RSA). Troviamo quasi ugual incidenza nelle fratture di femore sia all'esterno che presso il proprio domicilio. E' anche da considerare il quadro clinico caratterizzato da turbe della marcia e della postura, paura, ansia, insicurezza e turbe comportamentali definita da Murphy [19] nel 1982, la Post Fall Syndrome e poi successivamente descritta anche da autori francesi come Syndrome de désadaptation psychomotrice, Possiamo distinguerla in due forme, forma severa e quella moderata basandoci sulla modalità di esordio e alla gravità. La forma severa si manifesta con una sindrome ipocinetica: il paziente non cammina più, anche se sono assenti in lui patologie di tipo neuromuscolari evidenti.

In definitiva non possiamo indicare come unica causa riferita alle cadute nella popolazione anziana l'elevata incidenza statistica, poiché bambini o atleti hanno sicuramente tassi molto più elevati, senza che questo, si possa indicare come problema sanitario.
Piuttosto possiamo dire che si tratta di una combinazione di incidenza e facilità nel farsi male come ad esempio una lesione oppure una frattura, in relazione al fatto che negli anziani si presenta un'elevata prevalenza di malattie come l'osteoporosi, cambiamenti fisiologici inerenti l'età come ad esempio il rallentamento dei riflessi che può causare un danno anche con una caduta lieve.
In ultima istanza è importante dire anche che la guarigione per una persona anziana è sicuramente più lenta e questo può aumentare il rischio di una nuova caduta [12].

1.3 Panoramica sulle tecnologie assistenziali

La definizione di tecnologia assistenziale (in inglese Assistive Technology, AT) in letteratura non è sicuramente univoca, infatti troviamo una differenziazione sia per gli obiettivi, sia per i mezzi impiegati. E' semplice capire la motivazione che sta dietro alla non-unicità del termine: l'anziano, la sua salute e il contesto che ha attorno, dimostrando così che le variabili sono molteplici.

Come descritto in Hirsch [6], si prova a definire lo scopo delle tecnologie assistenziali asserendo che mantengono, o se possibile migliorano, la qualità della vita (QoL).
Per QoL si indica l'equilibrio tra ciò che il paziente può fare autonomamente e la loro capacità di relazionarsi ed aiutarsi reciprocamente.
Per un anziano questo equilibrio è personale, imputabile al vissuto e allo stato attuale di abilità della persona, oltre che fondamentale data la natura precaria del ultra sessantenne.
Come indicato negli studi di Mann [7] e di Giugliani [8] gli anziani, in maggior quantità quelli in età molto avanzata, sono disposti a cambiare le loro abitudini inserendo nel proprio ambiente TA se queste hanno lo scopo di migliorare in maniera esplicata la loro QoL.
Con i lavori effettuati da [9] e [10] possiamo individuare gli strumenti atti al supporto dell'anziano e conseguentemente tirare le somme per cercare di capire quale dispositivo hardware e software sia meno invasivo per i soggetti in causa.
Per questa tesi si è presa come assunto l'ipotesi semplificativa per cui l'anziano sia in sostanza autonomo e che quindi le AT abbiano il solo scopo di gestione delle emergenze all'interno di strutture domestiche.
E' stato quindi, utilizzato un algoritmo di riconoscimento della caduta (in letteratura definito Fall Detection), per fornire un controllo ove incorra un emergenza.

Nel panorama delle tecnologie hardware inerenti questo campo abbiamo 3 grandi categorie:

- **Camera,** generano immagini bi o tri dimensionali nella zona interessata.
  Risolvono le problematiche inerenti di verifica da remoto e non sono vincolanti per il soggetto una volta installate. D'altra parte sono estremamente invasive dal punto di vista della privacy, soprattutto in ambienti quali il bagno e la camera da letto. Un esempio su tutti è il prodotto di casa Microsoft, infatti la Kinect si può utilizzare anche per tale scopo. Spieghiamo ora il suo funzionamento.

- **Ambientale**, modificando superfici quali pavimenti, ma non solo, è possibile posizione rivelatori della pressione ed impatti.

Presenta il vantaggio di non essere invasivo, ha come svantaggi quelli di mancanza di controllo remoto e una limitazione assoluta alla sua area di installazione.

- **Indossabili**, piccoli dispositivi che si possono posizionare o su indumenti o a contatto con il soggetto.
  Il vantaggio più evidente è l'indipendenza dal luogo fisico in cui si trova.
  Lo svantaggio più evidente è dato dal fatto di non avere una sicura verifica da remoto (se non con applicativi modulari). Per questa categoria si utilizzano solitamente accelerometri, moduli GPS e moduli Bluetooth.

Andiamo ora ad analizzare nel dettaglio alcune possibili soluzioni per ogni categoria:

1.3.1 Soluzioni basate su videocamera

Rougier [23] ha presentato nel 2006 degli studi per ricostruire un modello tridimensionale della testa del soggetto partendo dai suoi dati biometrici e lo streaming video. Riconoscendo che la testa del soggetto si può approssimare ad un ellisse, si applica un algoritmo per trovare la posizione della testa nel frame, stimata la nuova posizione nello spazio tridimensionale dell'ellissi e conseguentemente vengono calcolate la velocità verticale e orizzontale a partire dai modelli nello spazio. Se i valori di velocità verticale ed orizzontale superano una certa soglia allora è stata trovata una caduta.
E' giusto dire che al momento della pubblicazione dell'articolo, comparivano alcune problematiche, come quella di inizializzare manualmente la posizione del testa nello streaming.
Un'altro approccio, trovato in letteratura è quello proposto da Foroughi [24], il quale vuole realizzare un sistema robusto per la rilevazione di cadute riconoscendo la postura. Tale soluzione combina dati provenienti dalla telecamera, quindi frame video, con l'uso di Support Vector Machine per classificare le posizioni.
Nella fattispecie, dopo aver trovato la sagoma umana, vengono estratti tre parametri.
Il primo dei parametri è l'ellissi che approssima la Figura umana rilevata, considerando

il centro, l'angolo di rotazione e le misure degli assi, il secondo è invece l'insieme degli istogrammi di proiezione della Figura sugli assi orizzontale e verticale normalizzati e come ultimo punto la posizione della testa, e il suo conseguente andamento temporale.
Un altro esempio è quello inerente lo studio di Mastorakis [25], il quale utilizza una soluzione visiva con l'ausilio della Kinect, prodotto di Microsoft in accoppiata con la console Xbox [27]. Tale metodo ci è sembrato ingegnoso poiché utilizza un prodotto che può essere già presente in casa. A questo punto andiamo a descrivere quello che è il funzionamento di una telecamera come la già sopracitata Kinect.

1.3.1.1 Kinect

La Kinect è un accessorio originariamente concepito per la console Xbox 360, al a differenza del Wiimote della Nintendo, consente di al giocatore di potersi muovere liberamente senza ausilio di dispositivi. Sebbene nata esclusivamente per console, da febbraio 2012, Kinect è stata resa disponibile anche per PC, quindi utilizzabile anche come telecamera. Infatti essa è dotata di una telecamera RGB, un doppio sensore di profondità IR con uno proiettore ad infrarossi ed una telecamera sensibile alla stessa banda. La telecamera RGB ha una risoluzione a 640 x 480 pixel, mentre quella ad IR utilizza una matrice 320 x 240 pixel. Al suo interno troviamo anche un array di microfoni atti alla calibrazione del ambiente in cui si trova, mediante l'analisi della riflessioni dei suoni sulle pareti. Inoltre Kinect ha anche un Tilt motor per la motorizzazione verticale di se stesso, infatti, grazie a tale metodo è possibile orientarsi nella posizione migliore per il riconoscimento dei movimenti.

1.3.2 Soluzioni basate su sistemi ambientali

Per quanto concernano le soluzioni con sistemi ambientali possiamo trovare un possibile esempio in Alwan del 2006 [25], il quale osserva come le attività umana generano una vibrazione nel pavimento misurabile. Quando le persone cadono, il loro corpo impatta con il pavimento e questo trasmette vibrazioni al pavimento. Quindi si è pensato di inserire uno speciale sensore piezo per catturare tali vibrazioni, poiché è significativa la differenza tra una caduta e un'attività quotidiana. In aggiunta si dimostra

che è diversa anche la caduta di un oggetto rispetto una persona.
Ritroviamo un approccio simile anche in Tzeng del 2010 [26], questo metodo però prevede l'utilizzo di un sistema ibrido, il quale unisce sensori di pressione posti sotto al pavimento ad immagini IR. Il prototipo prevede un controllo costante sui valori che i sensori di pressione "ascoltano", se questi valori di pressione superano una certa soglia, allora è attivata una telecamera IR per il controllo remoto.
Si vuole quindi approfondire come funziona il sensore utilizzato in questo tipo di soluzioni.

1.3.2.1 Sensore di pressione

E' giusto, per completezza, fare una panoramica su tutte le tipologie relative ai sensori di pressione, anche se, in questo specifico contesto si utilizza solitamente il sensore di pressione piezoresistivo.
Andiamo quindi ad elencarli:

- **a soffietto**
- **a tubo di Bourdon**
- **a membrana e diaframma**
- **ottici**
- **piezoresistivi**

Il primo della lista, quello a **soffietto** è un sensore che converte la pressione applicata in uno spostamento lineare. In questo caso la sensibilità dipenda dall'elasticità del materiale utilizzato. Lo spostamento lineare subisce poi una conversione in grandezza elettrica.
In ordine troviamo al secondo posto quello **a tubo di Bourdon.** Per questo sensore si utilizza un tubo di sezione ellittica chiuso ad una estremità. Il tubo è a spirare ed al variare della differenza di pressione subisce un arrotolamento e conseguente srotolamento: una lancetta indica quindi il movimento. Utilizzo per pressioni medio-basse ed è poco accurato. Successivamente, nella lista troviamo il **barometro a membrana e diaframma** . Il diaframma è soggetto a una tensione radiale λr (N/m) che

richiama la membrana nella posizione “piana”. Al quarto posto invece facciamo spazio ai **sensori ottici** . In questo caso la membrana (passiva) in silicio trasduce la pressione in una deformazione del primo specchio di un interferometro di Fabry-Perot. Il secondo invece, specchio si trova su un supporto fisso di vetro. La luce riflessa verso il rivelatore è funzione dello spessore w dell’interferometro e dunque della pressione applicata alla membrana. Infine troviamo i **sensori piezoresistivi**, quelli che sono stati utilizzati negli esperimenti fatti dagli articoli citati in precedenza. In questo caso membrana (sensore di pressione) e sensore di forza/deformazione sono ambedue realizzati in silicio. Il segnale è basso (mV) e occorre una compensazione in temperatura. La deformazione produce una variazione di resistenza $\Delta R \propto \sigma \propto P$ che solitamente è misurata da un circuito a ponte di Wheatstone.

1.3.3 Soluzioni basate su device indossabili

Un esempio di utilizzo di soluzioni indossabili è descritto in Bourke 2007 [20]. In questo articolo si può leggere di esperimenti con due accelerometri triassiali posti al torace e alla coscia per trovare quale sia il vettore di accelerazione nei due punti variando il tempo. E' stato chiesto, a persone dai 20 ai 30 anni di simulare, in ambiente sicuro, una serie di cadute, e ad un insieme di anziani, con età compresa tra i 70 e gli 80 anni, è stato chiesto di svolgere le loro attività giornalierre (ADL). Altri studi, come ad esempi quelli di Pérolle del 2002 [21] sono stati fatti integrando anche un localizzatore GPS oppure gestendo la comunicazione tramite Bluetooth come in Hwang del 2004 [22]. E' giusto quindi, descrivere i componenti inerenti le soluzioni indossabili.

1.3.3.1 Accelerometro

L'accelerometro è un componente elettromeccanico utile a misurare la forza di accelerazione. Questa accelerazione può essere statica, come la costante di gravitazione universale che grava ai nostri piedi, oppure può essere dinamica causando il movimento o la vibrazione dell'accelerometro. L'accelerometro, come rappresentato in Figura 3, ci aiuta a misurare la quantità di accelerazione dinamica, possiamo quindi analizzare come il nostro dispositivo mobile si muove. La Nintendo lo ha intuito integrando nei suoi

controller, i Nunchuck, all'uscita delle Nintendo Wii, un altro esempio lo possiamo trovare proprio a bordo dei nostri dispositivi mobili; Samsung infatti, ha utilizzato negli ultimi anni, le Gesture, il riconoscimento dei gesti attraverso algoritmi e la gestione sapiente di questi sensori. A dispetto delle applicazioni ludiche in cui possiamo trovare questo componente, l'accelerometro viene utilizzato anche in altri contesti. Il Segway, un innovativo mezzo di trasporto su due ruote, utilizzato anche dalla polizia per la normale vigilanza in città, utilizza proprio questo componente per la gestione di stabilità dello stesso. Sorge ora spontaneo chiedersi il funzionamento del sensore. Per capire ciò, possiamo schematizzarlo come in *Figura 1.3*, immaginando un cubo al cui interno si trova una massa sospesa al centro, attraversata da molle (tre nella casistica triassiale) a loro volta agganciate al centro di ogni faccia del cubo. Muovendo il cubo, la massa all'interno si muoverà comprimendo o allungando le molle che la tengono sospesa. Questo permette la misurazione dell'accelerazione dato il grado di compressione delle molle. Nel caso in cui le molle vengono allungate l'accelerazione sarà negativa, positiva altrimenti.

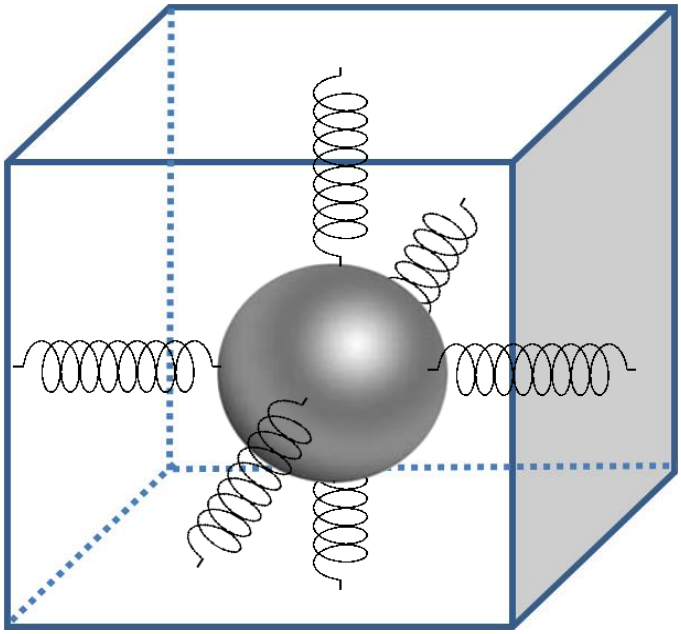


Figura 1.3: Rappresentazione di un accelerometro

1.3.3.2 Modulo GPS

GPS sta per Global Position System, e cioè sistema di posizionamento globale. Il suo

scopo è quello di trovare sempre la nostra posizione in modo preciso ed esatto all'interno del pianeta Terra. Tecnologia nata per scopi militari dal Ministero della difesa Americano, il GPS trova un'applicazione civile negli ultimi anni.
Il GPS è composto da un certo numero di satelliti, da 24 a 32, i quali trasmettono ripetutamente informazioni nella banda delle microonde. Le orbite dei satelliti sono circolari su 6 piani orbitali paralleli inclinati di 55° rispetto al piano equatoriale.
Ogni satellite si trova a circa 20 km dalla terra, il suo periodo di rivoluzione dura circa 12 ore. Le orbite son state studiate così da poter avere, in ogni momento, almeno 4 satelliti qualunque sia il posto in cui ti trovi all'interno della Terra.
Per la gestione e la verifica dei satelliti esistono 4 stazioni di controllo a terra, esse si occupano per l'appunto di correggere l'orologio atomico e l'orbita in caso di bisogno, oppure di verificare lo stato stesso dei satelliti.
Il resto del lavoro è effettuato dal modulo ricevitore del segmento utente, infatti esso localizza 4 o più satelliti, ne calcola la distanza da ogni satellite ed utilizza questi dati per calcolarne la propria posizione tramite trilaterazione, utilizzando le proprietà dei triangoli infatti, possiamo calcolare la distanza tra 2 punti.
Questo metodo quindi, per calcolare la posizione effettiva attraverso la determinazione di 3 valori (nel caso bidimensionale) fondamentali, con i quali si può costruire un unico triangolo, è analogo il caso tridimensionale, solamente che le misurazione sono quattro.
Nella pratica il ricevitore GPS comunica con i satelliti analizzando le alte frequenze (le bande usate sono 1575.42 Mhz e 1227.60 Mhz) con cui questi trasmettono i dati a terra.
Per capire quale distanza intercorre tra satellite e ricevitore viene calcolata la latenza del segnale inviato dal satellite. Una decisiva sincronizzazione permette al ricevitore di sfruttare la latenza registrata fra il tempo del segnale ricevuto (comunicato dal satellite) per calcolarne la distanza moltiplicando tale dato per la velocità della luce.
Una volta effettuati i calcoli il nostro ricevitore GPS può trovare le informazioni riguardanti longitudine, latitudine e ovviamente altitudine, e applicato ad una mappa si può sapere perfettamente dove siamo.

1.3.3.3 Modulo Bluetooth

Essendo questa una tecnologia di riferimento possiamo trovare un gran numero di

applicazioni, questo è anche un dei motivi che ci ha spinto ad utilizzarla nel nostro progetto.

La tecnologia che sta dietro al Bluetooth è nata in casa Ericsson nel 1994, stava infatti cercando un'interfaccia radio a bassa potenza per poter collegare gli accessori (auricolari/microfoni etc.).

Analizzando questa tecnologia è facile riconoscere quali siano le sue caratteristiche più evidenti; L'assenza completa di cavi, un costo basso derivato da una tecnologia semplice, il suo essere così autonomo, infatti i dispositivi si connettono assieme senza che l'utente conosca nulla del suo funzionamento.

La vera innovazione introdotta è data dal fatto che, il Bluetooth, crea una PAN(Personal Area Network), ossia una rete privata utilizzando onde radio.

Il raggio d'azione varia in base alle classi del dispositivo, partendo da un raggio di un metro circa in classe 3 a circa 100 metri di raggio per la classe 1.

Parlando di specifiche, possiamo affermare che il Bluetooth trasmette nella stessa banda delle reti wireless 802.11b/g (Wi-Fi), ma con potenza molto inferiore, circa 1 mW contro i 35-100 mW del Wi-Fi e tranfer rate pari a un Mbit al secondo.

La potenza del segnale debole non deve essere vista come un difetto ma anzi, fa si che esso non interferisca con altri apparecchi a radio frequenze, in alcun modo.

Si può a questo punto supporre però, che due dispositivi che inviano/ricevono dati ad altri dispositivi nelle vicinanze possano disturbarsi a vicenda, ma il fatto non sussiste.

La tecnologia Bluetooth utilizza un metodo nominato Spread-Spectrum Frequency Hopping, invece di utilizzare un canale di frequenza fissa, salta 1600 volte al secondo, da uno all'altro dei 79 canali a sua disposizione, questo per ridurre le probabilità di interferenza tra i diversi dispositivi disposti nella stessa area.

Supponendo una situazione del genere è improbabile che i due dispositivi riescano a trasmettere dati nella stessa frequenza allo stesso momento. Anche nel caso remoto in cui ciò possa avvenire, si tratterebbe solamente per una frazione millesimale di secondo.

La scelta è ricaduta sulla terza delle tre macro categorie, i device indossabili. Si è pensato a tale soluzione per la sua indipendenza dal luogo fisico in cui si trova e per la sua economicità rispetto le categorie concorrenti.

1.4 Stato dell'arte dei device indossabili

Negli ultimi decenni questo argomento è stato trattato in modo molto accurato in letteratura, essendo, come già esposto nei paragrafi precedenti, una criticità sociale non indifferente.

In questo senso possiamo trovare molti lavori accademici, a partire da sostenitori come Lord e Colvin [14] nel 1991 e Williams [15] nel 1998, possiamo ricavare degli studi su device indossabili completamente autonomi integrati in una cintura, i quali, rilevavano gli impatti sul suolo e, con un sensore al mercurio, rilevavano l'inclinazione di una persona per capirne la posizione. Continuando a ricercare si può trovare lo studio accademico di Mathie [16] del 2001, dimostra che attraverso un accelerometro tri-assiale posto in vita, utilizzando una serie di parametri come l'accelerazione, l'angolo di inclinazione del paziente, la durata di una postura e attività precedenti e postume ad una caduta è possibile creare un sistema per rilevare le cadute con sensibilità pari al 98% e specificità tra l'88% e il 94%. Seguendo la linea temporale incontriamo altri studi in questo ambito. Noury [17] per esempio affronta uno studio su un sensore autonomo da fissare sotto l'ascella. Il dispositivo riceve dati relativi ad un accelerometro e ove necessario rivela anomalie quando la velocità supera una certa soglia, la posizione verticale diviene orizzontale e questa posizione permane per diversi secondi. Questo studio porta ad una sensibilità e ad una specificità pari all'85%. Tutti i lavori illustrati fino ad ora si basano su dispositivi indossabili da applicare al soggetto interessato. Da un punto di vista operativo, però, nascono delle problematiche, dovute alla scarsa ergonomicità del prodotto e che quindi li rendono poco diffusi. Si dovrebbe inoltre pensare “all'intrusività tecnologica” di questi dispositivi, infatti è plausibile supporre che una persona anziana non sia avvezza alle tecnologie e nuovi dispositivi, seppur semplici possono creare imbarazzo nella persona. Una soluzione meno invasiva, ma che al contempo non privi degli standard alti di sensibilità e specificità imposti dai device indossabili va ricercata, nella praticità e nella semplicità d'uso, degli smartphone che sono ampiamente diffusi. Giunti a questo punto è doveroso prendere in considerazione lo stato dell'arte di alcune delle applicazioni per smartphone, riferite al fall detection, presenti all'interno dello Store di Google [28].

Attraverso la nostra ricerca, ci siamo imbattuti in diverse applicazioni.

1.4.1 T3LAB Fall Detector

La prima delle app presa in esame, intitolata: T3LAB Fall Detector, visibile in *Figura 1.4*. Essa è una delle creazioni del consorzio no-profit T3LAB, nato a sua volta dalla collaborazione tra l'Università di Bologna e Unindustria Bologna. L'app si presenta con un'interfaccia grafica molto pulita, gradevole e lineare. Appena attivata, si presentano a noi solamente le opzioni per l'inoltro dei dati ad un ricevente prestabilito (tramite chiamata e/o messaggio) e un pulsante per l'attivazione del servizio in background. D'altra parte testandola, nel caso di una possibile caduta, non si avvia nessun tipo di dialog per avvertire l'utente di riscontro di caduta o di un falso negativo; Inoltre, la chiamata non è funzionante, infatti si disattiva immediatamente dopo l'intento. Si evince dai fatti che l'algoritmo utilizzato si prende cura solamente delle variazioni di accelerazione e non di riconoscimento di postura, a parer nostro una scelta un poco approssimativa. In ultima istanza vorrei far notare come il service non viene mai distrutto, se non con una terminazione forzata.

Figura 1.4: Applicazione T3LAB Fall detector

1.4.2 CRADAR

Sempre attraverso la nostra ricerca ci siamo imbattuti in una seconda app, la quale si intitola: CRADAR, acronimo di Crash Detection And Response.

Si presenta con un'interfaccia grafica un po' confusionaria e poco curata *Figura 1.5*, infatti i box dove poter inserire i dati da gestire in caso di caduta, si confondono e non risaltano all'interno dell'activity, cosa sconsigliata a parer nostro essendo questi, punti focali dell'applicativo.

Confrontandola con l'app precedente, la gestione di falsi negativi, inerenti la possibile caduta attraverso un dialog, viene presa in esame; Nella fattispecie si attiva una nuova activity che ci considera lo stato fisico dell'utente, infatti ci chiede di cliccare su un pulsante se il nostro stato è nella norma, in caso contrario verrà inviato un messaggio all'utente impostato in precedenza.

Anche in questo caso possiamo desumere dai nostri test che non viene preso in considerazione il riconoscimento della posizione nell'algoritmo.

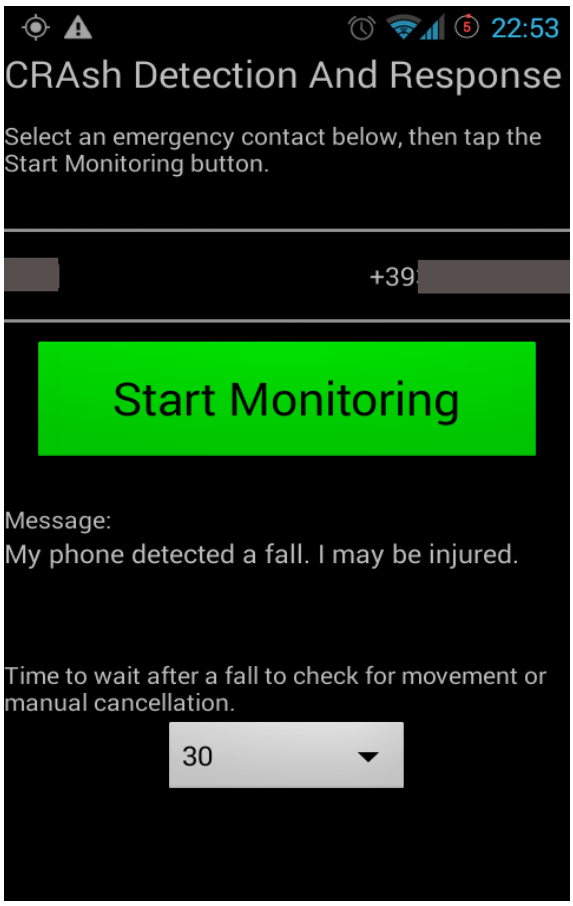


Figura 1.5: Applicazione CRADAR

1.4.3 Fall Detector di Spantec GmbH

La terza app installata nel nostro smartphone per la fase di testing ha come nome: Fall Detector, questa applicazione è stata creata dalla Spantec GmbH, azienda tedesca che opera in questo campo, anche con device indossabili. Si presenta attraverso un'activity, la quale, al centro si presentano tre anelli, come in *Figura 1.6*, i quali, virtualmente dovrebbero rappresentarci e far capire il nostro stato. L'intenzione ci è parsa simpatica, ma onestamente con scarsi risultanti per quanto riguarda l'intuitività e la rapidità d'uso, soprattutto se ipotizzata per una persona anziana.Non abbiamo trovato un modo per poter attivare un servizio in background, così che l'utente possa operare con il suo smartphone in maniera completamente libera; Questo, ci ha francamente stupiti, soprattutto per un'azienda che opera in questo settore. Premendo il tasto reporting possiamo infatti gestire tutte le caratteristiche inerenti la caduta come visibile in *Figura 1.7*, che possiamo affermare siano veramente tante, forse troppe.Infatti possiamo trovare, oltre alle peculiarità inerenti la sensibilità del sensore per carpire la caduta, anche le opzioni per coordinare vibrazione, audio e localizzazione GPS. L'opzione per inviare un messaggio su Twitter ci ha lasciato un po' perplessi perché ipoteticamente inutile nel caso di una reale caduta. Possiamo affermare che l'algoritmo funziona correttamente, perlomeno per i nostri test, inviando tutti gli avvisi correttamente, anche in questo caso non è stato gestito il riconoscimento della postura.

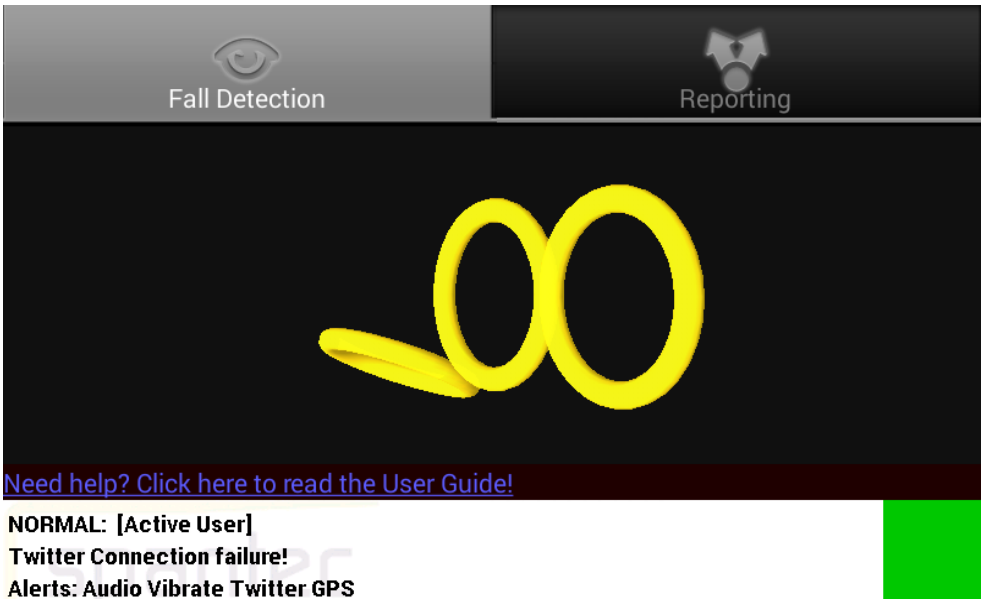


Figura 1.6:Activity principale di Fall Detector di Spantec GmbH

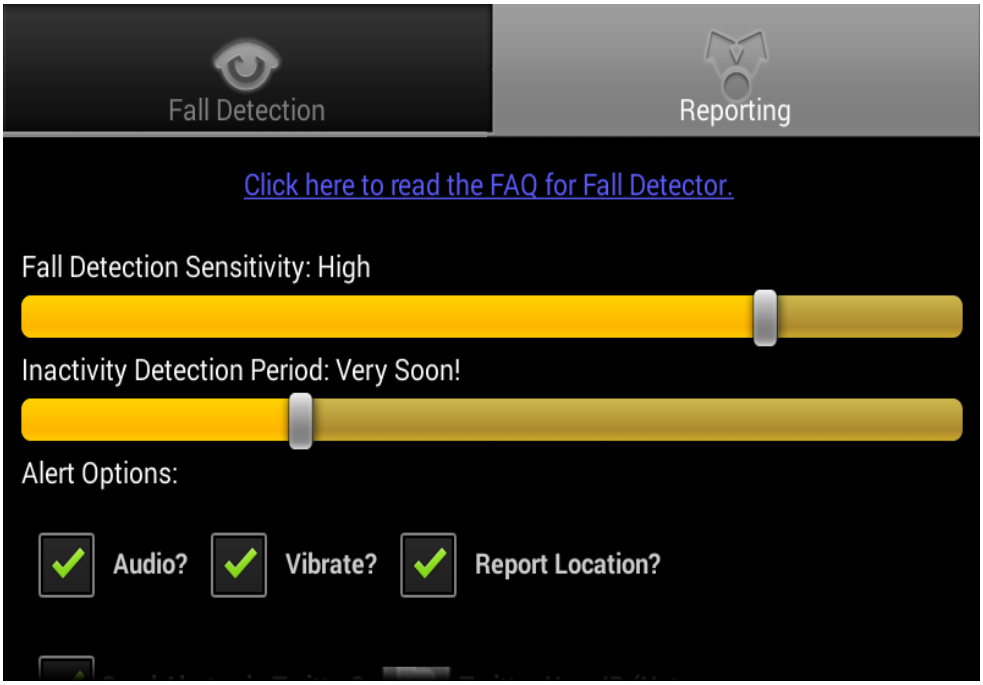


Figura 1.7: Reporting dell'applicazione Fall Detector di Spantec GmbH

Ultima pecca è stata riscontrata nell'utilizzo poco intuitivo per gestire i falsi negativi, infatti è presente solo un tasto molto piccolo per poter fermare l'allarme.

1.4.4 Smart Fall Detection

L'ultima delle app che siamo riusciti a trovare all'interno del Play Store, e conseguentemente a testare è stata l'app denominata: Smart Fall Detection, visibile in *Figura 1.8* e *1.9*.

Questa applicazione è stata il risultato del dottorato di ricerca in intelligenza artificiale presso il Department of Computer and Communication Systems Engineering, University Putra Malaysia (UPM) da parte di Hamideh Kerdegari [18].

Essa infatti utilizza un metodo completamente diverso per ricercare le cadute, utilizza una rete neurale artificiale (ANN) per processare i segnali di accelerazione. Quando ANN rileva una variazione significativa di accelerazione ricerca all'interno di un database formato da circa mille campioni di cadute e non cadute, per capire se la persona è realmente caduta oppure è un falso negativo.

L'applicazione è sicuramente ben fatta, l'unico appunto che possiamo permetterci di fare è anche qui la gestione di un servizio in background che possa funzionare senza che

l'utente debba “prendersi cura” dell'applicazione.

In questo caso l'applicazione è gestita attraverso un servizio e questo viene anche notificato, ma quando si cerca di uscire dall'applicazione anche il servizio viene distrutto.

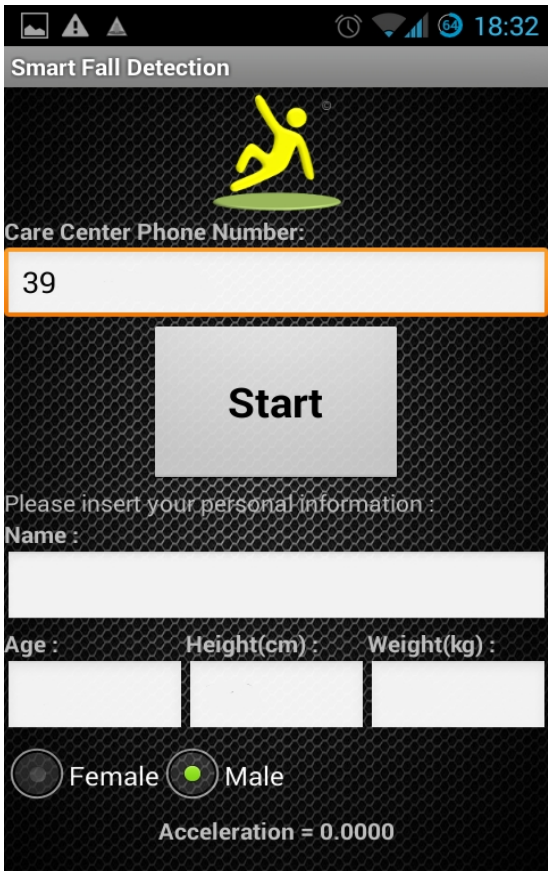


Figura 1.8: Activity S.F.D.

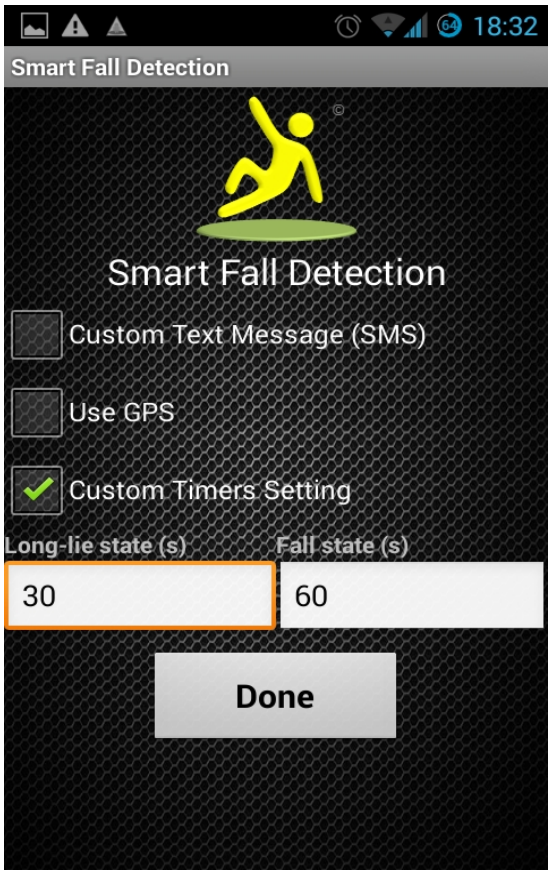


Figura 1.9 Setting S.F.D.

Capitolo 2

La fase progettuale

Nel precedente capitolo sono state descritte varie applicazioni mobili che permettono di monitorare l'anziano e di verificarne l'effettiva caduta. Il processo di analisi delle applicazioni esistenti è stato fondamentale per capire come progettare la nostra app, mantenendo le funzionalità fondamentali ed implementando quelle che invece non sono fornite ma sarebbe utile implementare.
In questo capitolo saranno quindi descritte le tecnologie che abbiamo utilizzato per implementare l'app mobile e l'architettura scelta.

2.1 Sistemi operativi mobile

I sistemi operativi mobile, permettono di controllare i dispositivi mobili allo stesso modo in cui Unix, Linux, Mac OS o Windows controllano un desktop computer o un laptop. Tuttavia alcune delle problematiche sono differenti, come la limitazione delle risorse (CPU), la ridotta dimensione del display e diverse tecnologie per l'accesso ad internet. Esso gestisce tutte le funzioni che è in grado di svolgere lo smartphone. Nel mercato odierno esistono diversi tipi di sistemi operativi mobile e conseguenti fork. I più diffusi sono cinque:

- Android
- Apple iOS
- Windows Phone
- Symbian OS
- Blackbarry OS

Da un'attenta analisi di mercato, si sono rilevate le statistiche inerenti le vendite per

sistema operativo nel 2013 da due differenti istituti di statistica; IDC [35] e Gartner [36]. Questi, affermano che più del 90% dei dispositivi mobile venduti nel mondo hanno come sistema operativo, Apple iOS [37], Windows Phone [38] e Android [39]. Per questo motivo ci è sembrato ragionevole limitarci a prendere in considerazione solamente questi tre sistemi operativi mobile. Andremo, qui di seguito a spiegare funzionamento e caratteristiche di ognuno.

2.1.1 Apple iOS

Apple iOS è il sistema operativo sviluppato da Apple per dispositivi mobile quali iPhone, iPod Touch e iPad, la prima versione è stata presentata il 9 gennaio 2007 al Macworld Conference & Expo. Questo è un sistema operativo proprietario e disponibile solo per dispositivi Apple. Tale limitazione rende possibile un'ottimizzazione delle risorse circa l'hardware posto nel dispositivo. Ad oggi, l'ultima release presentata al pubblico è la iOS 7.0.4.

Scendendo nei retroscena di questo sistema operativo possiamo mostrare una panoramica sulle caratteristiche tecniche. IOS come Mac OS X, è una derivazione di UNIX ed usa un microkernel XNU Mach basato sul sistema operativo open source Darwin OS. Questo è un kernel ibrido, creato dalla fusione del kernel FreeBSD [], variante originaria di Unix creata dall'università di Berkeley, e il microkernel Mach, uno dei primi microkernel creati e tra i più famosi. Apple, ha modificato ed esteso questo kernel per raggiungere i livelli prestazionali di Mac OS X o di iOS. Risultano i vantaggi di Mach e di BSD in questo modo. Mach, e nella fattispecie il suo microkernel si occupa della gestione della memoria, del sistema input/output e della comunicazione tra processi; Inoltre precisiamo che gestisce la protezione della memoria, preemptive multi- tasking e una gestione avanzata della memoria virtuale . La parte del kernel BSD invece, è responsabile degli utenti e dei permessi, al suo interno è presente lo stack di rete, offre un file system virtuale (VFS) ed è compatibile con le specifiche POSIX (IEEE 1003).

I livelli di astrazione di iOS sono essenzialmente quattro:

- il Core OS layer
- il Core Services layer
- il Media layer
- il Cocoa Touch layer

Possiamo affermare che il sistema operativo di casa Apple occupa meno di mezzo gigabyte della memoria interna del dispositivo.
Al livello più basso presentiamo il *Core OS*, al cui interno troviamo lo stack di BSD, quindi:

- le funzionalità di I/O
- gestione dei processi
- gestione thread
- sqlite [40]

Il secondo livello è quello definito *Core Services*, da questo livello in poi troviamo tecnologie sviluppate da Apple, e nel caso specifico da Core Foundation che sviluppa molte delle API che si utilizzano anche a livello più alto.
Assieme al Core Os, il Core Services offre la maggior parte dei servizi fondamentali al sistema operativo.
Seguendo, in maniera ascendente la pila, troviamo il livello *Media*, il quale fornisce i principali strumenti di gestione grafica, audio e video. Per la cura dei disegni 2D e 3D abbiamo i framework Open-GLES oppure Quartz Core, per progettare effetti visivi abbiamo invece Core Animation. Attraverso la libreria LibSystem possiamo avere accesso allo strato sottostante, il già sopracitato Core OS per le funzionalità di più basso livello. Elenchiamo di seguito alcune delle gunzionalità:

- Threading
- Networking
- Gestione file system
- Standard input/output

- Bonjour e sevizi DNS
- Allocazione delle memoria
- Computazione matematica

All'ultimo livello, quello più alto, troviamo lo strato chiamato *Cocoa Touch*. Esso si compone dei framework UIKit e foundation. Questi rendono possibile l'implementazione delle grafica tramite gli strumenti necessari. UIKit gestisce l'interazione con l'utente in maniera completa oppure la visualizzazione, alcuni esempi possono essere la renderizzazione dello schermo e le classi per gestire il multitouch.
Foundation, dall'altra parte gestisce le classi legate alla gestione, alla manipolazione e al salvataggio dei dati. Cocoa Touch, in particolare, è utilizzato per l'implementazione delle seguenti caratteristiche:

- Application Managment
- Supporto alla grafica
- Gestione dell'interfaccia grafica
- Supporto per il testo e i contenuti Web
- Gestione dell'accelerometro
- Gestione della fotocamera
- Supporto alla gestione degli eventi

2.1.2 Windows Phone

Windows Phone è un sistema operativo basato sulle API di Win32 di Microsoft. Nato dalle ceneri di Windows Mobile, venne presentato al Mobile World Congress nel febbraio 2010. Ad oggi, l'ultima versione ufficiale disponibile è Windows Phone 8. Passiamo ora alle caratteristiche tecniche.
Windows Phone supporta vari dispositivi e richiede delle specifiche tecniche minime per essere messo in esecuzione. Anche questo sistema è strutturato a strati come i precedenti . Windows Phone utilizza una soluzione ibrida, prendendo parte del kernel da

Windows Embedded CE 6.0 R3 e in parte da Windows Embedded Compact 7. Uno dei cambiamenti fondamentali che troviamo su Window Phone è la possibilità di sfruttare appieno l'architettura ARMv7. Windows Phone è un sistema operativo Hard real time a 32 bit e modulare pensato per dispositivi embedded. Supporta il multitasking e architetture diverse quali x86, ARM, MIPS e SH4 operando su uno spazio di indirizzamento di 4 GB. Il kernel utilizza 2 GB di memoria virtuale superiore, mentre i restanti vengono impiegati per la gestione dei processi utente attivi, esso ne supporta fino a 32768.

Tra i processi, gioca un ruolo centrale Nk.exe, il nucleo del sistema operativo. Al suo interno sono caricate le librerie dinamiche atte a varie funzionalità di sistema. Altro componente di rilevo è il file eseguibile Oal.exe detto OAL, questo ha il compito di far interagire il kernel (Kernel.dll) con i componenti hardware a bordo del device caricando il Device Manager (Device.dll) durante la fase di boot.

Per quanto riguarda il tipo di file system, Windows Phone ne utilizza uno molto simile a quelli già visti nei sistemi UNIX, infatti hanno una root unica, mount dei dispositivi con lettere, RAM come ObjectStore, crittografia dei media removibili e gestione supporti di memorizzazione e file fino a 4 GB.

Tra le caratteristiche che lo contraddistinguono dalla concorrenza possiamo trovare:

- **Silverlight [41]**, Tecnologia di Microsoft per la visualizzazione, all'interno del browser di contenuti RIA, applicazioni multimediali ed interattive.
- **Connection Manager**, Infrastruttura tecnologica per la gestione di interfacce di rete sul dispositivo.
- **Internet Explorer integrato [42]**, browser di casa Redmond, possibilità di multitouch, panning e zoom e un'interfaccia personalizzabile per una migliore esperienza dell'utente finale.
- **Microsoft Office e PDF Viewer**, applicazioni per visualizzare ed editare documenti, tra i quali, Microsoft Office Word, Excel, PowerPoint ed Adobe reader direttamente nel dispositivo.
- **Web Services on Devices API (WSDAPI)** , costruzione di servizi Web per la migliore interazione tra i dispositivi in rete. Con WSDAPI si possono scambiare messaggi e metadati, oltre ad un supporto per la gestione degli eventi e la

sicurezza.

- **Driver**, produzione di qualità dei driver per garantire la minor quantità di modifiche possibili da approntare su hardware personalizzato; Esso contribuisce quindi a garantire una migliore portabilità del sistema.

Microsoft ha deciso di restringere il campo sulle specifiche hardware da utilizzare sui dispositivi.
Questo riduce notevolmente il rischio di frammentazione hardware garantendo così che il prodotto sarà perfettamente ottimizzato, e la sua esecuzione la più veloce possibile in ogni dispositivo venduto ne mercato. Per fare un'analogia con i due sistemi operativi sopraelencati, asseriamo che Microsoft si posizione tra la vasta selezione di prodotti che Android ci offre e il controllo autoritario di Apple.
Per completezza è giusto ricordare che nel febbraio 2011 Steve Ballmer , amministratore delegato di Microsoft ha annunciato, in compresenza con Stephen Elop, amministratore delegato di Nokia, la partnership delle due. In questo modo, Nokia abbandona definitivamente il sistema operativo fino ad allora utilizzato Symbian OS.

2.1.3 ANDROID

Android è un sistema operativo per dispositivi mobili, inizializzare sviluppato da Android Inc. e poi acquistato nel 2005 da Google.
Android, non è stato sviluppato da zero, ma si basa su diverse versioni del kernel Linux. Una delle caratteristiche che lo contraddistingue dalla concorrenza è la sua natura open source e la sua versatilità, infatti, è utilizzabile in qualsiasi dispositivo mobile. Per completezza è giusto mettere in luce il fatto che il primo smartphone ad utilizzare questa piattaforma è stato l'HTC Dream presentato il 22 ottobre 2008 con la versione Android 1.0: ad oggi, l'ultima versione presentata è la 4.4.2 detta KitKat.
Nello specifico sappiamo che questo sistema operativo è costituito da uno stack software (un set di sottoinsiemi software) che include il sistema operativo di base, che si basa sul kernel Linux, ed è composto da applicazioni Java che vengono eseguite da uno speciale framework, anch'esso basato su Java e orientato agli oggetti. Vengono inoltre utilizzati degli strati middleware (applicativi intermediari tra le diverse

applicazioni e i componenti software) per la comunicazione con le applicazioni di base. Sopra il kernel troviamo le librerie fondamentali. Questa piattaforma utilizza una serie di librerie in linguaggio C e C++, riportiamo di seguito alcuni esempi.

- **Il leggerissimo SQLite [40]**, una libreria software che implementa un DBMS SQL di tipo ACID incorporabile all'interno di applicazioni. Tra le sue qualità ricordiamo la leggerezza (meno di 500k tutta libreria), la velocità, le API sono semplici da usare, è multipiattaforma e completamente libero.
- **La libreria SGL**, essa è utilizzata da Android come libreria per implementare la grafica bidimensionale.
- **Le API OpenGL ES 2.0 [43]**, sono utilizzate invece per gestire accelerazione trimensionale della grafica.
- **Media Libraries**, esso è basato sulla libreria open-source OpenCore e riguarda il supporto audio e video, avendo a disposizione una serie di codec tra cui mp3, wav, avi, mp4, flv eccetera...
- **WebKit**, il motore grafico per un browser web utilizzato per il rendering delle pagine web. WebKit è un progetto open-source che nasce dalla combinazione di componenti del sistema grafico KDE e di tecnologie Apple.
- **Surface Manager**, esso gestisce la visualizzazione grafica.
- **Un framework multimediale OpenCore e System C library**, cioè un ottimizzazione della libreria libc (libreria standard di Linux) per sistemi mobili.

Una volta che le applicazioni sono scritte con il linguaggio Java, vengono eseguite tramite una virtual machine del tutto simile a quella che, Java utilizza per eseguire i propri programmi, ma ottimizzata per smartphone, la cosiddetta Dalvik Virtual Machine (DVM) [31]. Essa è stata scritta da Dan Bornstein , un dipendente di Google. Questo è uno dei componenti fondamentali di Android, infatti grazie ad un intelligente utilizzo dei registri di sistema, permette di indurre il dispositivo a sprecare meno memoria possibile, consentendo di far girare più istanze delle macchine virtuale contemporaneamente, nascondendo al sistema operativo la gestione dei thread e in ultima istanza quella della memoria. Attraverso un pacchetto auto-installante con estensione APK, il software viene distribuito all'utente finale. Esso non è altro che un

archivio compresso, al cui interno abbiamo il software (con estensione .dex che andrà in pasto alla DVM), le risorse (immagini, suoni etc...) e alcuni file XML per la gestione dell'interfaccia grafica e non solo. L'utente finale ovviamente, non è al corrente di tali informazioni, dato che il dispositivo gestisce tutto il processo di installazione autonomamente tramite un web services come l'Android Market. All'interno del archivio troviamo anche un certificato, il quale descrive le "impronte" dell'app, in questo modo siamo sicuri di non incorrere in un applicativo compromesso o revocato.

2.1.3.1 I vantaggi di Android

Avendo ora ben presente la panoramica che ci si prospetta davanti, possiamo decidere quale sistema operativo utilizzare per la nostra applicazione. Noi abbiamo scelto di utilizzare Android per due semplici motivi che andremo ad elencare:

- **Ampia fascia di mercato.** Come già detto anche nel capitolo precedente, uno dei punti focali su cui abbiamo sempre puntato è una soluzione che permetta di rispondere ad una fascia di mercato ampia. Come si può vedere in *Tabella 2.1*, tratta dallo studio condotto nel 2013 dal istituto Gartner possiamo notare come il sistema operativo Android, nel anno 2013, abbia coperto quasi l'80% dei device venduti nel mondo. E' anche rilevante notare come questo andamento sia in ascesa, infatti nel 2012, Android era installato nel 64% dei device circa e nel 2013 ricopre il 79%.

- **E' Open Source**, Un altro punto a favore, che gli altri due sistemi operativi contendenti non posseggono è l'approccio open source. Esso indica un software i cui autori permettono la visione e lo studio del codice sorgente a chiunque voglia visionarli.

Sistema Operativo	Unità vendute 2013	% nel mercato 2013	Unità vendute 2012	% nel mercato 2012
Android	177,898.2	79,0	98,664.0	64,2
iOS	31,899.7	14,2	28,935.0	18,8
Microsoft	7,407.6	3,3	4,039.1	2,6
BlackBerry	6,180.0	2,7	7,991.2	5,2
Bada	838.2	0,4	4,208.8	2,7
Symbian	630.8	0,3	9,071.5	5,9
Altri	471.7	0,2	863.3	0,6
Totale	225,326.2	100,0	153,772.9	100,0

Tabella 2.1 Studio Gartner su vendite sistema operativo smartphone nel 2012/2013

2.2 Ambiente di sviluppo

Per poter realizzare un'applicazione con il sistema operativo Android è necessario utilizzare l'SDK (Software Development Kit). Nella pagina Web di Android si può liberamente scaricare [32]. L'SDK mette a disposizione degli sviluppatori una serie molto corposa di librerie che permettono di interagire con il device mobile con il sistema operativo Android su cui verrà installata l'applicazione attraverso diverse funzioni che permettono di gestire l'input/output, la connessione di rete e l'interfaccia utente. Dopo meno di un mese, al Open Handset Alliance sono stati rilasci i sorgenti, in maniera integrale, del sistema operativo. Elemento disponibile e fondamentale per lo sviluppo è l'emulatore che si trova integrato tra gli strumenti che lo sviluppatore ha in suo possesso nell'SDK Android. Esso simula un device mobile con schermo touch screen, chip audio, memoria a stato solido, modem gsm, accelerometro GPS, tastiera etc. L'emulatore permette all'utente di non preoccuparsi del sistema hardware che andrà ad eseguire, potendo così sviluppare qualsiasi tipo di applicazione in maniera trasparente. E' anche possibile fare il debug delle proprie applicazioni direttamente sul proprio dispositivo Android, attivando le opzioni inerenti al debug usb nell'apposita

sezione delle opzioni dello sviluppatore visibile in *Figura 2.1*.

Figura 2.1 Emulatore per il debug dell'app.

Detto ciò cerchiamo di capire, attraverso una panoramica, quali siano gli ambienti di sviluppo adeguati alla nostra casistica. La nostra scelta è ricaduta tra due differenti piattaforme, non commerciali:

- **Android Studio [44]**
- **Eclipse con il plugin ADT [45]**

2.2.1 Android Studio

Android Studio, è un ambiente di sviluppo molto giovane, infatti è stato presentato il 16 maggio 2013 da Ellie Powers, il product manager di Google all'annuale conferenza Google I/O. Esso si basa su un altro ambiente di sviluppo, IntelliJ IDEA, IDE con

licenza Apache 2, già famoso per essere considerato uno dei migliori nel campo, riconosciuto anche da Infoworld nel 2010. Android Studio, al contrario di Eclipse, si propone di fornire ai programmatori un software completamente dedicato proprio allo sviluppo di applicazioni Android, cosa che nel mercato odierno ancora non esisteva. Altro aspetto da tenere in considerazione è il fatto che Android Studio, non soltanto si possa scaricare gratuitamente ma che anche la licenza sia open source. Questo apre diverse porte ai programmatori che vogliono modificare le caratteristiche a seconda delle proprie esigenze. Per completezza andremo a vedere come è formata una classica struttura in un progetto Android Studio. Come mostrato in *Figura 2.2* possiamo innanzitutto notare come l'approccio grafico, rispetto Eclipse sia molto diverso. Scendendo nel particolare possiamo analizzare il progetto in tutte le sue caratteristiche:

- `MyApplication`, questo è il nome dell'intero progetto,la radice di una struttura ad albero che ci permetterà si scorrere tutto il progetto.

- `.idea`, contiene una serie di conFigurazioni per la creazione del nostro progetto. Le conFigurazioni possono essere molteplici, dal versioning, passando dalla dipendenze, al compilatore e ai moduli salvati fino ad arrivare alle conFigurazioni relative ai encoding supportati. Si tratta di una cartella che non va modificata se non in casi specifici.

- `Gradle`, essa contiene tutti gli strumenti per eseguire gli script di Gradle. Esso avviene attraverso quello che viene chiamato Gradle wrapper, le cui conFigurazioni e classi risiedono all'interno di tale cartella. Si tratta di uno script batch se il sistema operativo è Windows, altrimenti script di shell per ambienti UNIX, questo permette di scaricare Gradle e quindi eseguire l'eventuale script di build.

- `app`: essa è la cartella più importante del nostro progetto, essa si suddivide nuovamente in altre 3 sottocartelle e un file che andremo ad analizzare:
  - il file build.gradle: contiene le informazioni per il build delle nostra applicazione. Come già accennato, i sistemisti Android hanno messo a punto

un nuovo sistema di building, tale Gradle. Attraverso questo strumento molto flessibile, possiamo definire un nostro linguaggio dinamico (DSL), con il quale stabilire tutte le regole di build fino a realizzare diverse versioni dello stesso progetto basandoci su particolari parametri. Data la grande frammentazione dei dispositivi non è raro dover creare diverse versioni di un applicazione in base alle caratteristiche del device.

- `libs`: cartella standard per le librerie visibile anche in Eclipse
- `src`: all'interno di questa cartella, abbiamo la cartella main, la quale, a sua volta, contiene le cartelle java e res. La prima riassume al suo interno tutti i file sorgenti Java del nostro progetto, la seconda invece, tutte le risorse quali immagini, layout, elementi visualizzabili nel nostro display. Possiamo portare come esempio la cartella chiamata *drawable*, la quale conterrà tutti gli elementi che ipoteticamente potranno trovare spazio nel nostro futuro layout (come ad esempio immagini), a sua volta, al suo interno sono presenti diverse cartelle che si differenziano per un codice il cui valore semantico identifica la densità dell'elemento. Un altro tipo di risorsa essenziale sono i layout, file XML che ci permettono di definire in maniera dichiarativa le diverse sezioni della nostra applicazione. E' giusto far notare come anche la cartella *values* e la cartella *menu* hanno il loro peso, infatti grazie alla prima possiamo gestire i menu della nostra applicazione, mentre grazie alla seconda potremo gestire tutti i valori testuali di cui avremo bisogno. E' importante far notare come all'interno di *src* ci sia anche il Manifest di Android chiamato per l'appunto AndroidManifest, questo è il file di configurazione che definisce i contenuti e il comportamento della nostra app. Esse potranno essere opportunamente selezionate in base al particolare dispositivo che conterrà la nostra applicazione.
- `build`: in questa cartella sono presenti tutti gli output dei processi generati, come ad esempio i file con estensione .dex, le classi compilate, le risorse e molto altro. Uno dei file di maggior rilievo che si trovano dentro questa cartella è il file R.java, Android infatti, per gestire al meglio le risorse genera, per ognuna di esse una costante nel suddetto file, permettendo cosi il riferimento all'interno del codice Java.

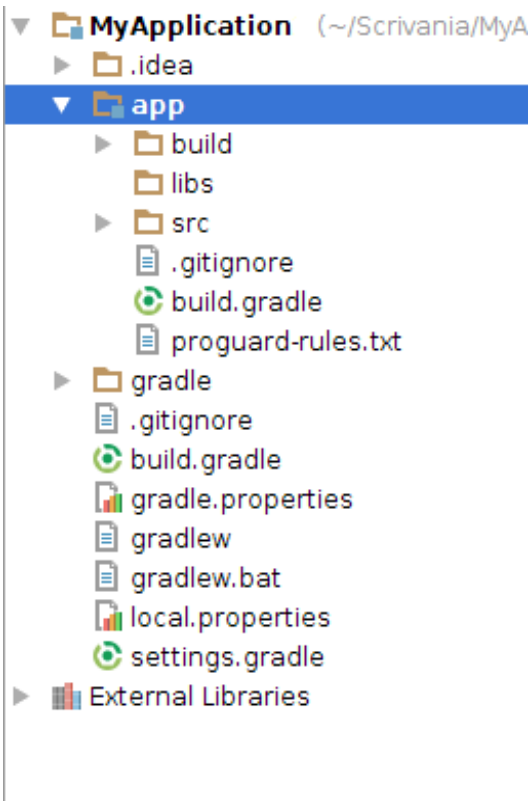


Figura 2.2 Struttura di un progetto con Android Studio

Una volta definito quale sia la struttura di un progetto con Android Studio, andremo a definire alcune delle caratteristiche che lo contraddistinguono:

- **Sistema di designing delle interfacce XML**, offre infatti una continua preview di come il layout su cui stiamo lavorando sarà effettivamente sul dispositivo sul quale lo andremo a testare, o anche su molteplici contemporaneamente. Questo ci permette una modifica on-the-fly sul codice XML se è necessaria una modica sull'interfaccia. Android Studio inoltre, evidenzia anche le eventuali differenze tra i temi, con API diverse.

- **Analisi profonda del codice**, è un sistema di auto-completamento chiamato IntelliJ, esso analizza il codice suggerendone eventuali modifiche necessarie, e letteralmente “imparando” dal codice già scritto in precedenza così da anticipare

le nostre mosse.

- **Sistema di compilazione Gradle**, esso è un sistema automatico di build flessibile ed evoluto. Esso può infatti automatizzare la fase di building, testing, pubblicazione e molto altro.

2.2.2 Eclipse con il plugin ADT

Eclipse, sin dalla sua nascita, è stato apprezzato in tutti gli ambiti della produzione software, partendo dallo studio e l'implementazione, passando per la ricerca, la progettazione e infine la commercializzazione. Questo software è rilasciato con i termini Eclipse Public Licence , i quali seguono i principi fondamentali del paradigma open source. Eclipse ha una community molto vasta, va dai singoli programmatori che per passione portano avanti questo progetto, fino alle grandi aziende informatiche quali HP, IBM ed Eriksson . Possiamo quindi asserire che questo progetto ha al suo interno un sistema ben organizzato, che integra la comunità open source e prodotti commerciali Eclipse-based . Ogni idea infatti, viene portata avanti in maniera completamente aperta e trasparente, e coordinata perfettamente dalla Eclipse Foundation. Essa ha il compito di comunicare costantemente con i membri per garantire il raggiungimento di certi obiettivi. Per far ciò, ogni anno, vengono organizzati meeting e gestiti i compiti tramite una maniacale pianificazione capillarizzata. La piattaforma fu ideata e conseguentemente sviluppata con il chiaro intento di essere estensibile in maniera profonda tramite plug-in scritti in Java. Ed è proprio la modularità uno dei punti di forza che ha portato Eclipse a ciò che è oggi in termini di diffusione. Entriamo nel particolare descrivendolo, come sia fatta una struttura di un'applicazione con Eclipse(come mostrato in *Figura 2.3*):

- `MyApplication`: la radice di una struttura ad albero, Contiene al suo interno tutti i file del nostro progetto.

- `src`: Questa directory contiene i sorgenti della nostra applicazione, organizzati

secondo i loro package di appartenenza.

- `gen:` è la cartella al cui interno vengono generati automaticamente dal sistema diversi file. Sicuramente il più importante è il file R.java, questo infatti, permette di creare un ponte tra i file Java e quelli XML. Android infatti, genera una costante per ogni risorsa creata, in questo modo è possibile un riferimento all'interno del codice Java.

- `assets:` qui, inseriamo le risorse che non vogliamo siano mappate in R.java. Questo tipo di risorse non vengono ne compilate e ne ottimizzate da Android. In questa cartella si può inserire un qualunque tipo di file organizzato in una qualsiasi struttura a directory. Per accedere a tali risorse, si utilizza un meccanismo tramite la classe `android.content.res.AssetManager` all'interno dell'Activity.

- `res`, contiene le risorse della nostra applicazione, ovvero quei file che in fase di building potranno essere ottimizzati per il dispositivo e memorizzati all'interno del file APK. Alcuni esempi sono la cartella *drawable* nella quale vengono messi tutti quegli elementi che saranno visibili nel layout della nostra applicazione, esistono diverse cartelle che si differenziano solamente per un codice il cui valore che identifica la densità dell'elemento. Altra cartelle essenziale è quella nominata *layout* infatti è grazie a questa che possiamo enunciare in maniera dichiarativa (tramite XML) le varie sezioni che andranno a comporre la nostra applicazione.

- `Il file di conFigurazione AndroidManifest.xml:` ultimo ma non ultimo per importanza l'AndroidManifest contiene al suo interno l'elenco completo delle activity e i service di un applicazione, con i permessi che servono al corretto funzionamento dell'applicazione. Possiamo quindi definirlo come il cuore pulsante di un'applicazione Android.

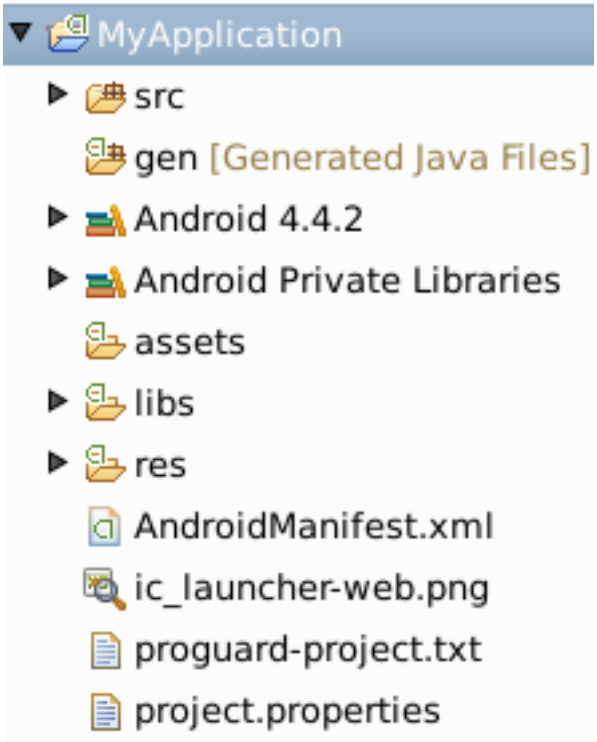


Figura 2.3 Struttura di un progetto con Eclipse

Nel nostro caso, è stato utilizzato un plug-in per la creazione di un progetto Android. Il suo acronimo è ADT e significa Android Developer Tools , questo fornisce, una volta installato, una serie di strumenti che si integrano con Eclipse. ADT fornisci una GUI a molti strumenti a riga di comando SDK. Qui di seguito presentiamo le caratteristiche di ADT:

- **Creazione, installazione e debug dei progetti creati con Android**. ADT riunisce tutto il flusso di lavoro di sviluppo in Eclipse, questo rende più rapido lo sviluppo di applicazioni Android.
- **Integrazione degli strumenti del SDK**. Molti degli strumenti del plug-in vengono integrati all'interno di Eclipse, anche questo rende lo sviluppo più rapido.
- **Documentazione integrata della documentazione Android**, E' possibile consultare la documentazione direttamente sul SDK.
- **Programmazione Java ed Editor XML**. L'editor del linguaggio di

programmazione Java IDE ha al suo interno le funzioni comuni quali: la verifica della sintassi in fase di compilazione, la documentazione integrata per il framework Android e l'auto-completamento. ADT inoltre, fornisce anche editor XML personalizzati che consentono di modificare i file XML specifici per Android. Un editor di layout grafico consente di progettare interfacce utente con una interfaccia drag and drop.

2.2.2.1 I Vantaggi di Eclipse con ADT

In definitiva, abbiamo proteso verso l'ambiente di sviluppo Eclipse per tre semplici motivi:

- **Familiarità,** Era già stato utilizzato nel mio percorso accademico per progetti Java, cosa che con IntelliJ IDEA non è avvenuta.

- **Maggior maturità del prodotto**, questo non implica che in un futuro non si possa utilizzare Android Studio, ma per ora non può essere considerato un prodotto affidabile.

- **Semplicità nella struttura dell'applicazione**, è oggettivo che, anche se ben pensata, la gestione di un progetto con Android Studio è più complesso, meno intuitivo e lineare rispetto ad un progetto Eclipse.

2.3 Principi e algoritmi per il fall detection

I metodi utilizzati per il fall detection, come descritto da Noury [29] del 2007, sono basati su metodi analitici oppure su tecniche definite di “machine learning” ovvero tramite metodi di apprendimento. Andremo ora ad analizzare questi due metodi molto diversi tra loro.

2.3.1 Metodo analitico

Quasi tutte le cadute terminano con una posizione sdraiata sul pavimento, l'approccio più semplice in questo caso, è quindi di rilevare questa posizione tramite sensori d'inclinazione. Questo metodo è indicato per il monitoraggio di “soggetti isolati” che stanno svolgendo qualsiasi tipo di attività, ma meno adatto a persone che non svolgono alcun tipo di movimento; infatti il semplice atto di dormire su un letto oppure sdraiarsi sul divano potrebbe generare un falso positivo. Una soluzione ibrida è quella di rilevare la posizione sdraiata sul pavimento, inserendo sensori di pressione sul pavimento, ma se la cadute è effettuata al di fuori delle zona monitorata oppure non termina a terra, questa non verrà mai rivelata. Quando una persona cade solitamente colpisce il pavimento oppure un ostacolo (“l'impact shock”, definito così da Noury [29]), che si traduce in un'intensa inversione del verso del vettore di accelerazione lungo la direzione della traiettoria. Questa inversione può essere rilevata tramite uno “shock detector”, un accelerometro con una soglia in precedenza prefissata. Anche se nella maggior parte dei casi le cadute si verificano sul “piano frontale” (in avanti o all'indietro), la direzione della traiettoria è evidentemente variabile da una caduta all'altra, quindi anche la posizione del sensore sul corpo relativa al punto di impatto può avere un incidenza sui parametri registrati al momento della caduta. Un altro parametro che può rilevare una caduta è la mancanza di movimento. Dopo una caduta seria infatti, se una persona ha subito gravi traumi, solitamente rimane immobile e non cambia posizione. Anche questo è possibile monitorarlo tramite sensori di movimento e di rilevazione delle vibrazioni piazzati in un'estremità del corpo (ad esempio al polso o sulla caviglia), oppure tramite semplici sensori infrarosso posti nell'ambiente domestico.

Punto cruciale per questo approccio è la scelta del tempo di latenza (il ritardo, in termini di tempo, prima di prendere la decisione), poiché dev'essere abbastanza lungo per ridurre i falsi positivi, ma d'altra parte si traduce anche in un tempo più lungo di attesa prima dell'intervento nel caso in cui la caduta sia reale. Durante la caduta si crea un periodo di cosiddetta “caduta libera”, in questa fase la velocità verticale del corpo aumenta linearmente con il passare del tempo a causa dell'accelerazione gravitazionale. Misurando la velocità verticale dei movimenti volontari di una persona, è possibile

trovare una discriminante con quelli relativi alle cadute potendo conseguentemente fissare una soglia appropriata. La difficoltà in questo caso risiede nella scelta della soglia: se è troppo bassa il dispositivo tenderà a rilevare eventi negativi (falsi positivi); se la soglia invece, è troppo alta allora sarà rilevati gli eventi desiderati solo in parte (falsi negativi). La soglia è variabile, essa dipende da soggetto a soggetto. Per superare quest'ostacolo si può ricorrere ad un periodo di apprendimento supervisionato o meno. Durante il primo caso, verrà chiesto a chi indossa il dispositivo di effettuare una serie di movimenti volontari al fine di identificare la velocità normale di esecuzione. Nel secondo caso è sufficiente la registrazione dei movimenti della persona durante lo svolgimento di normali attività, e procedere in seguito con un'analisi statistica sulla velocità misurata. Anche le immagini provenienti da una sorgente video possono essere processate ed usate al fine di ricercare una caduta, identificando la postura scorretta oppure rilevando movimenti bruschi utilizzando l'analisi vettoriale. Quest'ultimo metodo, tipicamente, tende a trovare le differenze tra due immagini consecutive ,in ordine temporale, per cercare di carpirne le variazioni, le quali sono organizzate in base alla loro direzione e/o al loro ampiezza.

Mentre queste tecniche sono molto stabili in ambienti controllati, come ad esempio nei laboratori, devono trovare un diverso approccio in ambienti non controllati ove non è possibile né modificare l'illuminazione, né la definizione del campo (ovviamente è necessario che il soggetto preso in esame sia nel campo visivo). Inoltre, poiché si soggetto vive in uno spazio a tre dimensioni, sarebbe utile impiegare tecniche più sofisticate come ad esempio la “stereo-visione”, ovvero utilizzando due videocamere prospettiche che inquadrano la stessa scena da posizioni leggermente sfalsate, ricreando in questo modo la tridimensionalità della scena, riuscendo a percepire il senso di profondità. Negli ultimi anni, queste tecniche stanno divenendo sempre più realizzabili, sia per un fattore tecnologico, che per quello economico, grazie alla creazione di telecamere a basso costo, come ad esempio le web-cam, la possibilità di poter trasmettere immagini attraverso la tecnologia wireless su distanze brevi, e la disponibilità di algoritmi ad-hoc. Tuttavia l'accettazione di tale tecnologia pone un problema maggiore, queste infatti richiedono il posizionamento di telecamere negli ambienti domestici dove i soggetti presi in esame vivono, in particolare in bagno e camera da letto questi, possono causare problemi riferiti alla privacy.

2.3.2 Il metodo Machine learning

Machine learning, occupa una delle aree fondamentali dell'intelligenza artificiale, essa si prende cura della realizzazione di algoritmi e sistemi collegati ad esso, basati su osservazioni come dati per la sintesi di nuova conoscenza. L'apprendimento avviene tramite proprietà di interesse conseguenti ad esempi, sensori o strutture dati. Il suo fine è quello di sfruttare la gigantesca potenza computazionale presente oggi nei nostri computer per esplorare tutti modi possibili di apprendimento automatico e realizzarlo con l'utilizzo di metodologie statistiche e computazionali. La qualità degli algoritmi è anche il punto di forza del Machine learning infatti, nel corso degli anni questi, sono stati migliorati e aggiornati; Questi possono essere suddivisi in basa due grandi tipologie, a seconda del tipo di apprendimento automatico per cui sono realizzati:

- **Apprendimento supervisionato**, qui, vengono forniti esempi con relative feature in input e un valore di output. Se abbiamo come valore di output un dato discreto allora possiamo classificarlo come un problema di classificazione, un esempio può essere l'appartenenza o meno ad una determinata classe.
  Se invece abbiamo un valore di output reale continuo in un determinato range allora questo sarà definito come problema di regressione. In ambedue i casi ciò che vogliamo trovare è quella funzione, che è chiamata ipotesi, la quale prevede che dato un input sconosciuto stima il valore di output. L'obiettivo dei problemi di apprendimento supervisionato è quello di riuscire a fare una predizione che si basa su proprietà conosciute, acquisite dai dati di input. Questi dati di input vengono definiti training set e si compongono tramite una serie di esempi, composti a loro volta da una serie di feature. Il valore di output invece, è chiamato target value.

- **Apprendimento non supervisionato**, in questo caso vengono forniti esempi con relative feature in input, ma nessun valore di output viene dato, quindi abbiamo a disposizione dati non etichettati. Questa famiglia di algoritmi ha come obbiettivo quello di trovare delle strutture dentro questi dati non

etichettati. Definiamo problema di Clustering se vogliamo trovare dei raggruppamenti di elementi definiti simili, altrimenti se vogliamo identificare sorgenti diverse che hanno contribuito alla creazione dei dati, allora lo definiamo problema di Blind Source Separation. Esistono altri problemi di apprendimento non supervisionato, e tutti hanno in comune il fatto di avere dati non etichettati o valori di output. Essi consistono nello scoprire le proprietà "nascoste" nei dati forniti in input.

Nel nostro caso, sempre grazie alle ricerche messe in atto da Noury [29], sappiamo che senza un modello analitico, possiamo applicare un approccio "intuitivo" per lo sviluppo di un sistema di "machine learning" basato sulla rilevazione di cadute, partendo da osservazioni (periodo di training) e successivamente la classificazione degli stessi. E' necessario decretare diversi criteri per la classificazione così che risultino sufficientemente significativi e indipendenti. Possiamo procedere attraverso un un periodo di training, nel quale si realizzerà una rete neurale, che potrà poi essere utilizzata per classificare automaticamente i futuri coinvolgimenti. Solo le situazioni che incontreremo durante la formazione potranno essere classificate, le altre saranno aggiunte in un classe definita "others". Se invece abbiamo una formazione non supervisionata, il classificatore si comporterà bene solo se il tempo di training è abbastanza lungo, idealmente dovrebbe essere inversamente proporzionale alla frequenza di caduta del soggetto stesso. E' però molto probabile che le prime cadute non saranno rilevate dal sistema in quanto la classe non sarà ancora nota.

2.3.3 L'algoritmo utilizzato

Per questa tesi abbiamo utilizzato il metodo analitico. Nel caso specifico abbiamo preso spunto dall'algoritmo proposto da Bourke [30] del 2010. In questo studio possiamo leggere di come sia stato utilizzato un device indossabile commerciale di Nokia, al cui interno sono presenti un sensore accelerometrico tri-assiale di casa Freescale (Freescale Semiconductor, Inc., 6501 William Cannon Drive West, Austin, TX 78735, USA) MMA7261QT), un micro processore di Texas Instruments (Texas Instruments, Inc., 12500 TI Boulevard, Dallas, TX 75243, USA) MSP430F1611), un modulo Bluetooth e

un alloggio per SD Card provenienti da Roving networks (Roving Networks, Inc., 809 University Avenue, Los Gatos, CA 95032, USA). Questo apparecchio è inserito nella cintura del soggetto in causa, e con esso è possibile registrare le variazioni di accelerazione e successivamente salvarle su SD Card o inviare i dati al computer tramite Bluetooth. Invece noi, per i motivi già sopraelencati nel capitolo precedente, abbiamo deciso di utilizzare questo algoritmo, che andremo ora ad esporre, all'interno della nostra applicazione per Android.
Innanzitutto Bourke, nel suo lavoro, esamina la struttura di una caduta tipica visibile in *Figura 2.4*, e ci fa notare come questa può essere divisa in quattro fasi distinte:

- **Fase di pre-caduta**, in questa parte il corpo non presenta nessuna variazione significativa, visibile in (A) di *Figura 2.4*.

- **Fase critica**, qui il corpo è in caduta libera, infatti in questa fase il corpo aumenta la sua accelerazione (B), inoltre abbiamo un intervallo di tempo in cui il corpo letteralmente vola (C), mentre la velocità verticale discendente aumenta.

- **Fase post-caduta**, quando il corpo tocca a terra l'accelerazione diminuisce fino a tornare a zero, mentre la velocità verticale tocca il suo picco negativo (D). Il soggetto ora affonda nel materasso, questo è il momento di massima decelerazione come riportato in (E),

- **Fase di recupero**, infine il corpo riprende stabilità in quest'ultima fase (F).

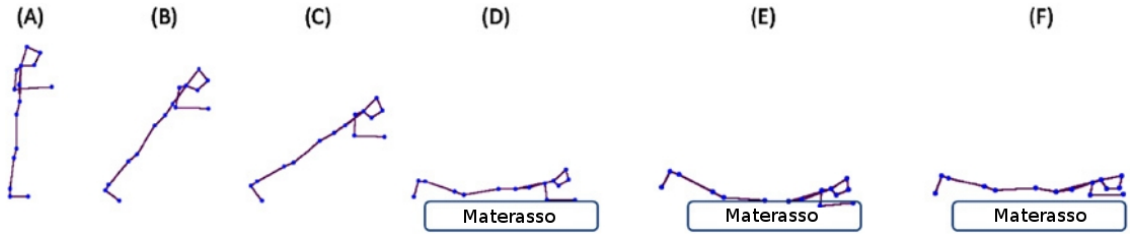


Figura 2.4, Fasi di una caduta (immagine presa dallo studio di Bourke [30]).

L'algoritmo utilizzato si struttura attraverso tre differenti parametri per esaminare le cadute:

- **Impatto**, per questo parametro utilizziamo un accelerometro tri-assiale per la rilevazione di una variazione d'accelerazione, infatti come già spiegato sopra e visto in *Figura 2.4* sappiamo che in fase critica l'accelerazione del soggetto aumenta. Utilizziamo quindi il metodo definito come Root-Sum-of-Square (o anche l'acronimo RSS), la formula prende i dati inerenti ad i tre assi dell'accelerometro, in un determinato momento, applica ad ognuno la potenza e,infine, viene applicata la radice quadrata su tutti e tre i membri. Se questo supera una soglia, in termini di accelerazione di gravità, sappiamo che è stata rilevata una decisa accelerazione del corpo, quindi un segnale importante, anche se non decisivo. Nel caso specifico 2,8 g sono la soglia che si deve superare.

$$RSS = \sqrt{x^2 + y^2 + z^2}$$

- **Velocità verticale**, questo criterio di giudizio invece, lo misuriamo nella fase post caduta, infatti è proprio nel momento in cui il soggetto tocca il suolo che la velocità verticale (Vve) tocca il suo punto negativo più alto. La Vve stimata la otteniamo tramite la seguente formula $((t2 - t1)/n) + (RSS - g)$, cioè l'integrazione numerica dei campioni forniti dall'**Impatto** sottratti dal valore di accelerazione di gravita **g**. In seguito poniamo una soglia negativa, che nella fattispecie, in base agli studi proposti da Bourke [30] è -0,7 m/s. Anch'esso si può utilizzare come termine di paragone positivo per ricercare una caduta.

- **Postura**, nella fase post-caduta, per aver un'ulteriore indizio sulla reale caduta del soggetto o meno, Bourke [30], esamina anche la postura nel caso in cui l'impatto o la velocità verticale abbiano superato la loro soglia. Attraverso il suo metodo si può determinare il soggetto è in piedi, se è seduto oppure se è inginocchiato. Tramite gli esperimenti condotti nello studio sappiamo che se un corpo, rimane per circa due secondi con un inclinazione maggiore od uguale a 60° rispetto l'asse terrestre, per almeno il 75% dei due secondi, questo significa

che esso è steso a terra. Per tale scopo si utilizza la formula abs[ori[1]] per almeno il 75% di t+1s – t+3s.

2.3.3.1 Approccio modulare

Per il nostro progetto abbiamo pensato di rendere modulare l'applicazione così da diminuire alcuni problemi inerenti i falsi positivi e falsi negativi (uno dei problemi più rilevanti per questo tipo di tecnologie) e gestire qualche verifica da remoto. Per modulare intendiamo l'utilizzo di più device che, comunicando tra loro gestiscono in maniera più precisa ed organica l'algoritmo implementato, riducendo al contempo alcuni errori. Infatti oltre allo smartphone, come in tutte le applicazioni presentate nel capitolo precedente, abbiamo deciso di utilizzare uno smartwatch, cioè un dispositivo da polso, che attraverso uno schermo Touchscreen è possibile utilizzare come estensione indossabile dello smartphone. In particolare, in base ai modelli che il mercato ci offre, possiamo utilizzare i diversi sensori a bordo come l'accelerometro, l'altimetro, il giroscopio o il contapassi. Possiamo ricevere ogni genere di notifiche che arrivano nel nostro smartphone, come chiamate o messaggi. Nella fattispecie, abbiamo pensato di gestire tramite la nostra applicazione la gestione dei falsi negativi dovuti alla caduta dello smartphone dalla tasca, e una maggior sensibilità nella rilevazione della stessa (nel caso fosse una caduta reale).
L'algoritmo di Noury [30], già sopraccitato, è inserito sia nell'applicazione per smartphone sia per smartwatch, perlomeno per quanto concerne la parte dell'algoritmo riguardante **l'impatto** e la **velocità verticale**. Infatti attraverso lo smartwatch teniamo conto solamente delle variazioni accelerometriche, e non quelle di postura.
Come visibile in *Figura 2.5* abbiamo deciso di connettere tramite Bluetooth lo smartphone e lo smartwatch. In questo modo, è possibile tenere costantemente aggiornati i due dispositivi l'un l'altro sullo stato dell'accelerometro. Abbiamo due casi:

1. Nel caso in cui il dispositivo posto al polso segnali una possibile accelerazione “anomala”, invia immediatamente allo smartphone il momento esatto in cui è avvenuta, esso, conseguentemente cercherà una corrispondenza temporale. Se questo accade allora lo smartphone avvierà le “indagini di postura”.

2. Nel caso in cui solo lo smartphone rilevi una accelerazione “anomala”, verrà presentato all'utente un Toast che indica il fatto che potrebbe essere caduto il cellulare dalla tasca.

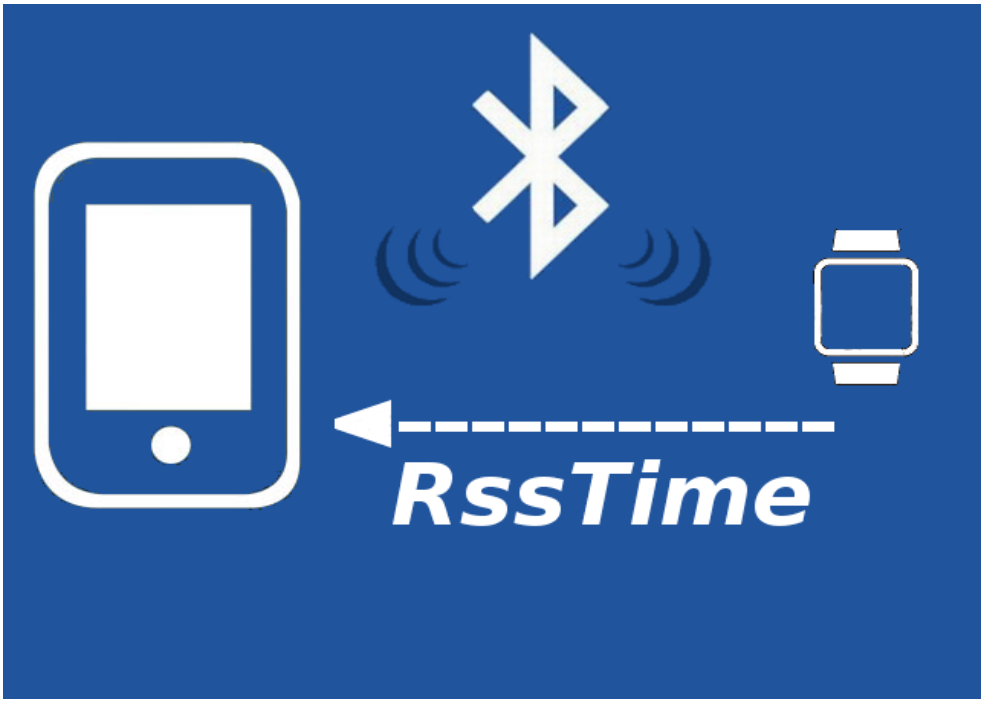


Figura 2.5 Rappresentazione dell'approccio modulare tramite Bluetooth.

Questo chiaramente è solo un esempio dimostrativo di come questo approccio possa migliorare effettivamente il metodo generale.

Capitolo 3

La fase implementativa

In questo capitolo verranno riportati, in prima istanza, l'elenco dei dispositivi utilizzati per il testing e il debug dell'applicazione e in seguito le soluzioni implementative per il fall detection, divise tra modulo Smartphone e Smartwatch, analizzando ogni funzionalità. Infine presenteremo i risultati dei campionamenti eseguiti durante la fase di testing.

3.1 Hardware utilizzato

Per la creazione dell'applicativo Android sono stati utilizzati i seguenti dispositivi messi a disposizione dal Campus di Cesena:

- Samsung Galaxy Note III con Android 4.3, schermo Full HD superAMOLED da 5.7 pollici, cpu Quad Core a 2,3 Ghz.
- Un Samsung Galaxy Gear con O.S. Android 4.3, schermo superAMOLED da 1.63 pollici e proccessore a 800 Mhz.

In aggiunta abbiamo utilizzato questi dispositivi per la fase di debug e campionamento delle cadute:

- Samsung Galaxy S II con Android CyanogenMod 4.3, schermo superAMOLED da 4,3 pollici e cpu dual core a 1,2 Ghz.
- Samsung Galaxy S III con Android 4.3, schermo superAMOLED HD da 4,8 pollici, cpu Quad Core a 1,4 Ghz.
- Samsung Galaxy S III mini con O.S Android 4.1, schermo superAMOLED da 4.0 pollici e cpu dual core ad un Ghz.

Abbiamo cercato di utilizzare prodotti delle stessa casa produttrice, vicini per quanto riguarda la versione del sistema operativo e la data di uscita commerciale nel mercato

per due semplici motivi:

- La facilità di riscontro nel mercato odierno.
- I sensori all'interno dei dispositivi sono della stessa casa produttrice e le loro funzionalità sono pressoché identiche.

3.2 Modulo Smartphone

Il modulo Smartphone è, il fulcro della nostra applicazione. All'avvio dell'applicazione si attivano i diversi sensori in ascolto. Questi hanno il compito di "udire" le variazioni in termini di accelerazione che lo smartphone subirà. Al contempo verrà richiesto di attivare sia il modulo Bluetooth, e il modulo GPS per tener traccia della posizione dell'utente in esame. Detto ciò, all'utente si presentano (come visibile in *Figura 3.1a):* 3 grafici, un menu nella parte alta dello schermo e un toggle button in fondo allo schermo. I tre grafici rispecchiano rispettivamente i tre parametri dell'algoritmo già esposti nel capitolo precedente, ovvero: (i) **impatto**, (ii) **velocità verticale** e (iii) **postura.** Nel menù in alto, invece, scorrendo da sinistra verso destra, viene visualizzato il titolo dell'applicazione e immediatamente sottostante lo stato del Bluetooth per la connessione con un altro dispositivo. Sono presenti anche tre icone, visibile in *figura 3.1b*: il primo dei quali ci permette di accedere all'activity per la selezione dei device disponibili: il secondo tasto (*figura 3.1c*) invece è stato implementato per rendere visibile il dispositivo per almeno 300 secondi, così da poter portare a termine la connessione Bluetooth. Il terzo e ultimo tasto del menu (*figura 3.1d*), apre una nuova activity per l'immissione di un numero di telefono e un eventuale messaggio. Questo servirà nel caso in cui sia rilevata una caduta e il messaggio composto verrà inviato al numero immesso in questa activity. Essendo attivo anche il modulo GPS, nel caso in cui il dispositivo sia agganciato ai satelliti, all'interno del messaggio inviato verranno messe automaticamente anche le coordinate geografiche e generato direttamente il link per Google Maps. Nel caso fosse rilevata una caduta, verrà attivato un Progress Dialog, che da la possibilità di fermare l'allarme entro dieci secondi. Questo per ridurre il rischio di un falso negativo. Se al termine dei dieci secondi non è stato cliccato il pulsante

cancella, allora verrà inviato un messaggio, e il processo di rilevazione ricomincerà nuovamente. Infine, il toggle button, attiva il servizio in background. In questo caso, dopo averlo premuto verrà segnalato, attraverso una notifica, l'attivazione del servizio. Nel caso di una caduta, verrà creata un activity ad-hoc con Progress Dialog per la gestione dei falsi negativi, come nel caso precedente. Quando si riattiva l'activity, il servizio viene automaticamente distrutto e riattivato al termine di se stesa.

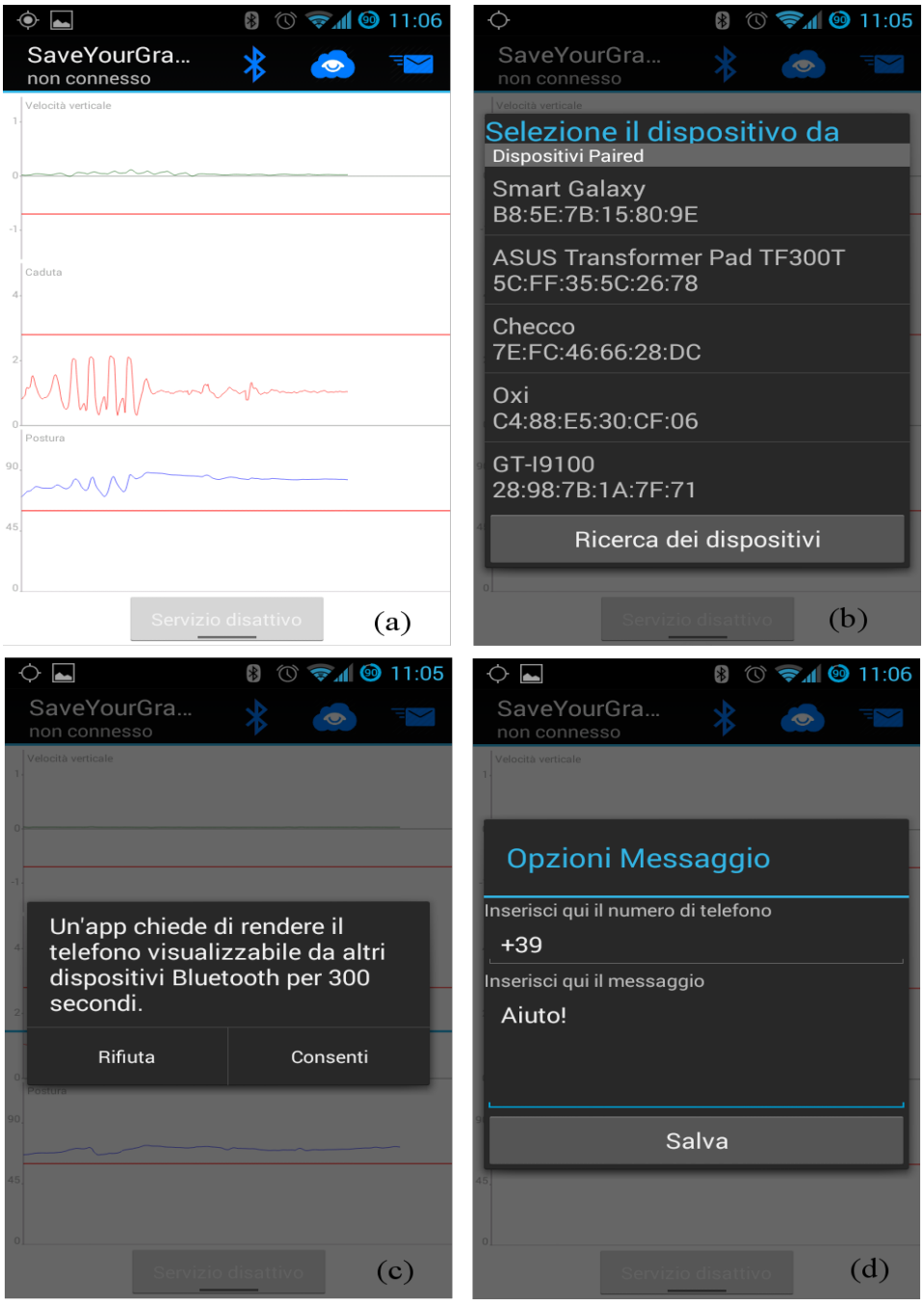


Figura 3.1: (a) SaveYourGranny activity, (b) Rilevazione device bluetooth,(c) Rendersi visibili, (d) Opzioni invio SMS.

Nel seguente paragrafo presenteremo le caratteristiche implementative di questo modulo, descrivendo inoltre nel dettaglio il package e tutta la gerarchia del nostro applicativo.

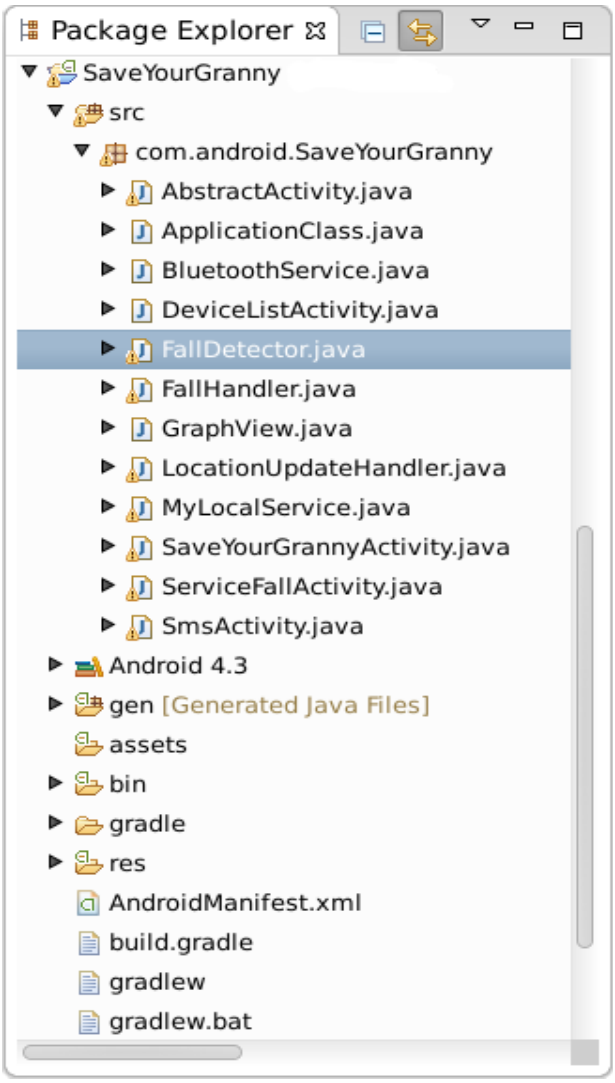


Figura 3.2, Struttura dell'app SaveYourGranny

3.2.1 Package e gerarchia

In ambiente Android, come già detto precedentemente si utilizza Java. Esso utilizza un meccanismo per organizzare le proprie classi all'interno di sottogruppi ordinati definiti Package. Android, come visibile in *Figura 3.2* presenta una cartella auto-generata al termine della creazione di un nuovo progetto ed è chiamata **src**, al suo interno sono

inseriti tutti i sorgenti Java per il funzionamento dell'applicazione in ordine alfabetico:

- *AbstactActivity:* questa è una classe marginale che serve a connettere i due device Bluetooth anche quando si attiva il servizio in background. Quello che vogliamo ottenere è una connessione automatica, nel nostro caso i due device che sono già paired e quindi l'indirizzo MAC è già conosciuto.

- *ApplicationClass:* anche questa classe è utilizzata come utility per la connessione la connessione Bluetooth attraverso un servizio in background. ApplicationClass, al suo interno ha un handler per la gestione dei callback dei dati, implementando un metodo set:

```
public Handler getHandler()
{
   return handle;
}
```

e un metodo get:

```
public void setCallBack(Handler.Callback callback)
{
       this.realCallback = callback;
}
```

- *BluetoothService:* questa è la classe adibita a tutti i lavori di settaggio e gestione delle connessioni Bluetooth con altri device. Contiene al suo interno un thread che ascolta le connessioni in entrata definito come:

```
private class AcceptThread extends Thread
```

Lo possiamo definire come la componente server nella fase in cui si costituisce una connessione. Esso è in esecuzione fino a che la connessione non è accettata (o cancellata).

Un altro thread invece è attivo nel momento in cui attendiamo di creare una connessione con un device:

```
private class ConnectThread extends Thread {
```

In questo caso la connessione può andare a buon fine oppure fallire.
Infine abbiamo un thread per eseguire la trasmissione dati quando è connesso:

```
private class ConnectedThread extends Thread {
```

Questo tratterà tutte le trasmissioni sia in entrata che in uscita.

- *DeviceListActivity:* questa activity è richiamata quando viene cliccato il primo pulsante (da sinistra) nel menu in alto della nostra applicazione. Essa appare come un Dialog, infatti nel `AndroidManifest.xml` viene richiamato il tema attraverso questo comando:

```
android:theme="@android:style/Theme.Holo.Dialog"
```

La funzione principale di questa activity è mostrare la lista completa di tutti i dispositivi già paired così da poter connetterli, oppure poter ricercarne uno tra quelli che ancora non lo sono, così da poterli connettere. Nel secondo caso possiamo avviare la ricerca tramite un pulsante, esso ci mostrerà un ulteriore lista dei dispositivi che sono all'interno del raggio d'azione del nostro dispositivo e che sono visibili. Infine, quando l'utente ha deciso con quale dispositivo collegarsi, viene preso l'indirizzo MAC di quest'ultimo e "spedita" all'activity padre. Il motivo risiede nel fatto che se l'utente, come probabile, decide di attivare il servizio in background sarà lo stesso stabilita una connessione automatica con l'ultimo dispositivo scelto.

- *Fall Detector:* questa classe genera la grafica dinamica dell'applicazione, cioè disegna le tre diverse funzioni prese direttamente dai sensori. Inoltre all'interno è presente anche l'algoritmo di rilevamento delle cadute. Andremo qui di seguito a commentarlo:
  Innanzitutto viene implementato l'interfaccia `SensorEventListener` per gestire i sensori che saranno registrati tramite il metodo `registerListeners()` così da attivare un listener per i sensori che ci servono. Uno dei metodi della nostra interfaccia `SensorEventListener` è

di fondamentale importanza per questa classe, il suo nome è `onSensorChanged(SensorEvent event)`, il quale si attiva quando il valore del sensore rileva un cambiamento. All'interno è stato inserito tutto l'algoritmo di fall detection che ora andremo a presentare, partendo dal grafico che si preoccupa dell'**impatto**:

```
if(event.sensor.getType()==Sensor.TYPE_ACCELEROMETER)
     {
     // Calcola RSS [sqrt(pow(x)2 + pow(y)2 + pow(z)2)
     float
rss=(float)Math.sqrt(Math.pow(event.values[0],2)
                              +Math.pow(event.values[1],2)
                              +Math.pow(event.values[2],2));
     // se RSS è maggiore di 2,8
     if (rss>RssTreshold*SensorManager.STANDARD_GRAVITY
                                   && granny.RssTime == 0)
          {
          if(granny.VveTime == 0)
          {
               granny.RssVal = rss;//salviamo il valore
rss
               granny.RssTime=date.getTime();//segna tempo
          }
     paint.setColor(0xFF0000FF); //colore BLU
     canvas.drawText("v",newX - 3,mGraphView.mYOffset * 2
+    4 *  SensorManager.STANDARD_GRAVITY *
     mGraphView.mScale[1], paint);
     // disegna una v per segnalare l'impatto
     }
     //ora disegniamo la curva che si formerà per cercare
     l'impatto.
     // draw_rss è la fine della linea
     float draw_rss = mGraphView.mYOffset* 2 + rss *
```

```
     mGraphView.mScale[1];
     paint.setColor(mGraphView.mColors[0]); // colore
rosso
     canvas.drawLine(mLastX, mLastValues[0], newX,
     draw_rss,paint);
     mLastValues[0] = draw_rss;
}
```

Questo codice dapprima calcola RSS dai dati presi dall'accelerometro poi, se è maggiore della soglia prefissata questo valore viene salvato assieme al momento in cui questo è successo. Nello stesso momento viene anche segnalato con una *v* di colore blu nel grafico. Viene fatta la stessa cosa con la **velocità verticale** calcolandola in primo luogo e poi controllando se supera la soglia che avevamo precedentemente deciso. Infine abbiamo la parte di codice che si occupa della **postura**:

```
else if (event.sensor.getType()==Sensor.TYPE_ORIEN..)
{
// orientation = 90° - |asse x del giroscopio|
     float ori = (90 - Math.abs(event.values[1]));
     float draw_ori=mGraphView.mYOffset * 3 + ori *
     mGraphView.mScale[2];
     paint.setColor(mGraphView.mColors[2]);
     canvas.drawLine(mLastXOri, mLastValues[2], newX,
                                    draw_ori, paint);
     mLastValues[2] = draw_ori;
     long wait_interval = (granny.RssTime != 0 ?
     date.getTime() - granny.RssTime:(granny.VveTime !=
0 ?  date.getTime() - granny.VveTime : 0));
     if (wait_interval >= OriOffset)
     {
          if (OriStartTime == 0)
               OriStartTime = date.getTime();
```

```
          else if (date.getTime()-OriStartTime< OriWindow)
          {
               if (ori_index < OriValues.length)
                    OriValues[ori_index++] = ori;
     canvas.drawLine(mLastXOri, mGraphView.mYOffset* 3 +
90   * mGraphView.mScale[2] - 2,newX, mGraphView.mYOffset
*    3 + 90 * mGraphView.mScale[2] - 2,paint);
          } else {
            int count = 0;
            for (int i = 0; i < ori_index; i++) {
               if (OriValues[i] > OriTreshold)
                    count++;
                 }
          if (count / ori_index >= OriConstraint&&
               granny.hasAcquiredGps)
          {
                         paint.setColor(0xFF0000FF);
                              canvas.drawText("v", newX -
4,                           mGraphView.mYOffset * 3 + 90  *
                              mGraphView.mScale[2] -
2,paint);
               granny.fall_detected = true;
          } else
               {
                    granny.reset_fall_values();
               }
          OriStartTime = ori_index = 0;
          }
       }
       mLastXOri = newX;
     }
```

Per la postura viene utilizzato il sensore orientation. Come nei due casi

precedenti prima viene calcolato, disegnati nel grafico i sample generati, poi nel caso in cui si sia rilevato un valore sopra la soglia per ambe i casi precedenti, si l'algoritmo aspetta un secondo e poi se per almeno il 75% dei due secondi successiva a quello appena aspettato il valori ori è maggiore di 60° viene segnalato nel grafico con una v di colore blu e poi viene cambiato un flag booleano che ci servirà a segnalare la caduta nel ciclo immediatamente successivo. Se quindi viene rilevata una possibile caduta l'algoritmo cerca se è in primis connesso allo smartwatch oppure no. Nel primo caso cerca una correlazione tra le tempistiche dei due dispositivi, se lo trova allora verrà attivato il ProgressDialog altrimenti verrà attivato solamente un messaggio. Nel secondo caso invece verrà automaticamente attivato il ProgressDialog

- *`FallHandler:`* questa classe non ha per ora un utilizzo pratico. Infatti qui risiede un solo metodo, `postDetectedFall()`, il quale ci permette di connetterci ipoteticamente ad server, inviandone tutti i dati nel caso in cui sia rilevata una possibile caduta. Sono diversi i dati inviati, abbiamo infatti: i valori dell'**impatto**, della **velocità verticale**, la data esatta e dove è successo attraverso le coordinate geografiche. In un futuro prossimo potrà sicuramente essere un valido aiuto per aumentare la modularità del nostro progetto.

- *`GraphView:`* essa contiene tutta la struttura grafica statica. Ciò significa che, attraverso oggetti Canvas vengono generati i tre grafici, come visibile in *Figura 3.1*, quello di Velocità verticale, quello di Caduta e quello di Postura. Si è deciso di estendere la classe alle View: `public class GraphView extends View` così da poter utilizzare il metodo `onDraw(Canvas canvas)` per disegnare i nostri grafici. Mostriamo ora un estratto del sorgente per il primo dei tre grafici creati quello della velocità verticale, ricordando che anche gli altri due sono del tutto simili:

```
@Override
protected void onDraw(Canvas canvas) {
    ...
    ...
```

```
cavas.drawColor(0xFFFFFFFF); //Colore dello sfondo
paint.setColor(0xFFAAAAAA);
cavas.drawText("Velocità verticale",mXOffset+4,20,
paint); //Settiamo il testo
cavas.drawLine(mXOffset, yoffset, mXOffset, 5,
paint); //disegno del asse Y
cavas.drawLine(mXOffset,yoffset/2.0f ,maxx ,
yoffset/2.0f, paint); //Linea su cui la funzione gira
float minusone=yoffset/2.0f
SensorManager.STANDARD_GRAVITY     *mScale[0];
cavas.drawText("-1", 7, minusone+3, paint); //numero
a fianco dell'asse Y (-1)
cavas.drawLine(mXOffset,minusone,mXOffset-2,minusone,
paint);
float zero = yoffset / 2.0f;
cavas.drawText("0", 10, zero + 3, paint); //numero a
fianco dell'asse Y (0)
cavas.drawLine(mXOffset, zero, mXOffset - 2, zero,
paint);
float one;
one= yoffset/2.0f+SensorManager.STANDARD_GRAVITY*
mScale[0];
cavas.drawText("1", 10, one + 3, paint); //numero a
fianco dell'asse Y (1)
cavas.drawLine(mXOffset, one, mXOffset - 2, one,
paint);
paint.setColor(0xFFFF0000);
float ytresholdVve = yoffset / 2.0f+
granny.mFallDetector.VveTreshold*SensorManager.
STANDARD_GRAVITY*mScale[0];//Linea rossa della soglia
cavas.drawLine(mXOffset,ytresholdVve, maxx,
ytresholdVve,paint);
...
```

```
...
}
```

- *LocationUpdateHandler:* classe adibita alla ricerca GPS la conseguente gestione per la geo-localizzazione del device. Il suo uso è finalizzato all'aggiunta delle coordinate geografiche (latitudine e longitudine) inviate all'interno dell'SMS inviato automaticamente nel caso di rilevamento di una caduta. La classe implementa l'interfaccia `LocationListener` per poter utilizzare le coordinate inviate dal satellite una volta attivato il modulo GPS (all'avvio dell'applicazione). Inoltre all'interno della classe è stato implementato il metodo `checkGPS()` attivato all'avvio dell'applicazione. Il suo scopo è quello di attivare il modulo GPS attraverso un dialog, e nel caso in cui non fosse disponibile o attivo, l'applicativo termina; Questo perché è essenziale sapere sempre dove l'utente è caduto.

- *MyLocalService:* è la classe adibita al compito di servizio in background. Si attiva tramite un bottone posto nel layout dell'activity principale (SaveYourGrannyActivity) attraverso un toggle botton. Attivato esegue tutte le funzioni di monitaraggio inerenti la caduta senza interfaccia grafica. Appena avviato il servizio (metoto `onCreate()`) lancia una notifica, in cui ci dice che il servizio è attivo. Nel caso in cui si rilevi una caduta, il sistema attiva un activity con un progress dialog per informarci di aver rilevato una possibile caduta, l'utente può decidere di cancellare (falso negativo) il messaggio, quindi l'activity si chiude e si riattiva il service. Altrimenti se passano i dieci secondi, viene inviato un messaggio, in seguito viene chiusa l'activity e riattivato il service tutto in automatico. Facciamo presente che la classe estende i `Service` e implementa `SensorEventListener` e `LocationListener`, questo per poter prendere i segnali dai sensori, così da poter riutilizzare l'algoritmo già implementato e, nel caso si rilevi una caduta, poter inviare un SMS di soccorso con le coordinate geografiche in maniera del tutto simile a ciò che succederebbe con l'activity. E' importante far notare che viene implementato il metodo `getSensor(event)` per il rilevamento dell'**impatto** e della **velocità**

verticale in maniera del tutto simile a quella implementata nella classe *`FallDetector`*. Questo metodo è a sua volta richiamato all'interno del metodo `onSensorChanged(SensorEvent event)` così da poterlo richiamare ogni volta che rileviamo un cambiamento di valore nei nostri sensori.

- *`SaveYourGrannyActivity:`* la Main Activity, quella principale. Essa è incaricata a far interagire le diverse componenti all'interno dell'applicazione. Esempi sono:
  - *la gestione dell'action bar*, il menu in alto con il quale possiamo vedere lo stato del Bluetooth, possiamo ricercare i device per la connessione Bluetooth cliccando sul primo tasto, renderci visibili cliccando sul secondo e infine, con l'ultimo tasto impostare i dati per l'inoltro del messaggio.
  - *Metodi per l'invio di messaggi Bluetooth*, `sendMessage(String message)` è il metodo adibito all'invio di un messaggio in una stringa, castandolo poi, all'interno del metodo in un vettore di byte (così è richiesto dallo standard). Infine chiediamo al thread `ConnectedThread` richiamato tramite il metodo `write(byte[] message)`, posto nella classe `BluetoothService` di inviarne il contenuto al device all'altro capo della connessione stabilita.
  - *Il metodo per l'inoltro dell'SMS*, utilizzando il metodo `sendSMS(String phoneNumber, String message)` riusciamo ad inviare messaggi per l'eventuale caduta che l'app potrebbe riconoscere. Questo metodo oltre ad inviare il messaggio ci segnalerà, attravero dei messaggi Toast se l'sms è stato inviato o meno, e se è stato consegnato al destinatario designato.
  - *Gestione della geo-localizzazione,* infatti è qui che viene inizializzato l'oggetto LocationManager, appena viene creata l'activity, nel metodo `onCreate()`. In seguito viene avviato il metodo protetto `checkGPS()` che ci chiede, attraverso un dialog di attivare il modulo GPS per poter così ricercare la posizione attuale.
  - *ProgressDialog per eventuali falsi negativi,* questo è molto importante all'interno della nostra applicazione. Infatti se l'algoritmo riscontra un

probabile caduta, viene immediatamente attivato questo dialog progressivo, così che l'utente possa sedare eventuali equivoci provenienti da un falso negativo attraverso un tasto “cancella” posto in fondo al box. Se questo viene premuto entro dieci secondi, il dialog viene cancellato e le attività di routine vengono ripristinate. Nel secondo caso invece viene inviato un SMS per segnalare la caduta, anche in questo caso però, tutte le funzioni di ricerca tornano a funzionare normalmente.

- *Mediaplayer,* sempre in questa classe viene creato l'oggetto alarm: `alarm=MediaPlayer.create(this, R.raw.alert_alarm);`

  il quale ci sarà utile a far squillare il nostro device, con un suono decisamente dissonante, nel caso in cui venga presentato il ProgressDialog relativo ad una possibile caduta. Esso avrà fine solo quando sarà utilizzato il pulsante “cancella” nel ProgressDialog, oppure se l'sms sarà inviato. La cartella a cui si fa riferimento per reperire il file in formato .mp3 é **res/raw**.
- *Attivazione Service,* in ultima istanza, qui risiedono anche i metodi per gestire il toggle button in fondo al layout dell'activity. Gestiamo l'avvio e l'arresto del nostro local service attraverso un booleano per sapere se il togglebutton è checcato o meno. Ricordo che nel file top.xml (file adibito alla gestione del layout del bottone), nel togglebutton abbiamo la proprietà: `android:onClick="onToggleClicked"` per poter gestire l'evento click sul bottone e "azionare" il seguente metodo atraverso un intent esplicito.

- *ServiceFallActivity:* questa classe è attivata tramite intent, da parte del service in caso di riscontro di una possibile caduta. Alla creazione attiva un Progress Dialog circa la potenziale caduta. Se l'utente non è caduto può cancellare il falso positivo tramite l'apposito tasto. Altrimenti viene attivato il metodo `sendSMS()` che gestisce l'inoltro del messaggio e i dati inerenti la posizione (vengono inviati i dati della posizione tramite un bundle dal service). Sia che il messaggio venga mandato oppure venga cliccato il bottone cancella, l'activity finisce per lasciare nuovamente spazio al service.

- *SmsActivity:* questa classe ha lo scopo di far inserire all'utente i dati che l'invio dell'sms che saranno gestiti nel caso si riscontrasse una caduta. Questa activity è attivata con la pressione dall'ultimo bottone del menu in alto (da destra verso sinistra). Anche questa activity appare a noi come un Dialog, infatti nel `AndroidManifest.xml` viene richiamato il tema attraverso questo comando:

```
android:theme="@android:style/Theme.Holo.Dialog"
```

  Attivata, ci si presentano due EditText che ci richiedono di inserire il numero di telefono e un messaggio di aiuto. In automatico, verrà aggiunto nel messaggio il link al sito Google Maps dove indica la latitudine e la latitudine in cui il soggetto si trova.

3.3 Modulo Smartwatch

Il modulo smartwatch è utilizzato come estensione di quello smartphone. Esso infatti è utilizzato, come già esposto in precedenza, per districarci da alcuni falsi negativi che con un approccio non modulare non potremmo risolvere. Vogliamo far notare che quindi parte del codice sorgente sarà in comune con quello del modulo smartphone, poiché le finalità, nella maggior parte dei casi sono all'unisono. All'avvio l'applicazione attivarò i listener sul sensore accelerometro così da poterne carpire i valori, per sottoporli all'algoritmo di **Impatto** e **Velocità verticale**. L'applicazione chiede inoltre all'utente di attivare il modulo Bluetooth poiché senza, l'applicativo non avrebbe senso di esistere. Una volta effettuato il collegamento nel menu in alto a sinistra verrà indicato lo stato della connessione. In questo menu sono presenti anche il tasto per poter connetterci ai device già paired oppure fare una ricerca di quelli ancora non conosciuti, inoltre è presente anche il tasto per renderci visibili ad un'ipotetica connessione. Come nel modulo precedente, nel caso in cui questi valori di **Impatto** o di **Velocità verticale** superino le soglie prestabilite verrà immediatamente inviato il momento in cui questi

“valori anomali” sono stati riscontrati tramite messaggi Bluetooth. Lo smartphone analizzerà i dati ricevuti e se le tempistiche coincidono verrà allora attivato un allerta tramite un ProgressDialog. Anche in questa applicazione si è tenuto conto dell'uso quotidiano che una persona potrebbe farne, quindi si è deciso di implementare un servizio in background, così che l'utente possa utilizzare l'applicativo anche senza averlo attivo continuamente. A tale scopo abbiamo pensato ad un solo tasto cliccabile esclusivamente nell'applicazione dello smartphone, che automaticamente creerà un intent al proprio service per tutte e due le applicazioni. Infatti quando si clicca il pulsante nell'applicativo smartphone, questo invierà un flag Bluetooth prima di richiamare l'intent. Questo servirà poi all'applicazione lato smartwatch come monito per creare l'intento al servizio in background. La connessione Bluetooth, quando verrà avviato il service sarà per un momento disconnessa ma subito riconessa automaticamente. Questo grazie al fatto che quando avviene la connessione tramite activity ci salviamo l'indirizzo MAC del device a cui ci stiamo connettendo, così da poter compiere successivamente tutte funzioni le in maniera autonoma.

Figura 3.3, L'app SmartGranny

Di seguito presenteremo qui di seguito le caratteristiche implementative di questo modulo. Descriveremo inoltre, il package e tutta la gerarchia del nostro applicativo, visibile in *Figura 3.4*, ed andremo ad analizzarlo nel dettaglio:

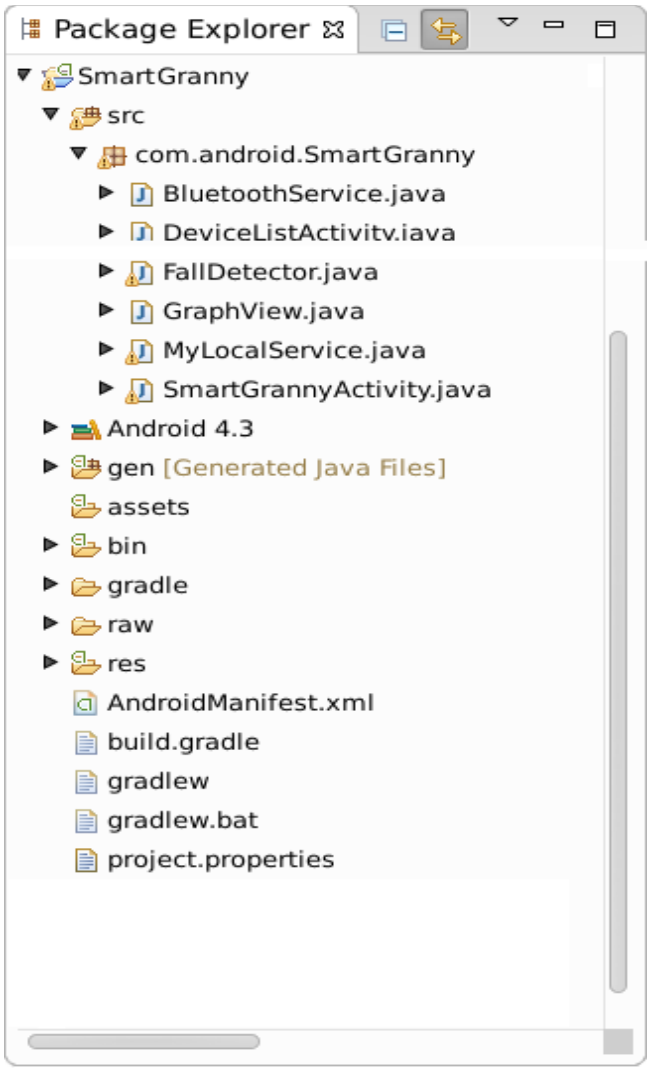


Figura 3.4, Struttura dell'app SmartGranny

3.2.1 Package e gerarchia

In questa sezione andremo ad analizzare in maniera esaustiva tutte le classi Java che troviamo all'interno della nostra cartella **src**, dentro il package *com.android.SmartGranny*. Ovviamente parte del codice del modulo Smartphone è stato riutilizzato per implementare le funzionalità anche nello smartwatch, dato che anche quello utilizzato da noi per i nostri test sfrutta Android (4.2.2) come sistema

operativo:

- *BluetoothService:* questa classe si occupa, in maniera del tutto simile a quella per il modulo smartphone, del settaggio e della gestione delle connessioni Bluetooth con altri device. Contiene al suo interno tre thread. Il primo che ascolta le connessioni in entrata definito come:

```
private class AcceptThread extends Thread
```

  Definibile come la componente lato server in fase di costituzione di una connessione. Esso è in esecuzione fino a che la connessione non è accettata (o cancellata).

  Il secondo thread invece è attivo nel momento in cui attendiamo di creare una connessione con un device e la sua classe è:

```
private class ConnectThread extends Thread {
```

  In questo caso la connessione può andare a buon fine oppure fallire. Infine, il terzo e ultimo thread ci serve per inoltrare la trasmissione dati in entrata e in uscita quando questo è connesso:

```
private class ConnectedThread extends Thread {
```

- *DeviceListActivity:* Anche questa classe è implementat nel modulo smartphone. Essa si attiva quando premiamo il primo tasto da sinistra nel menu in alto dell'applicazione. Questa serve a noi per poter decidere con quale device connettere il nostro e nella fattispecie per rendere modulare l'applicazione. Oltre ai dispositivi paired che sono già presenti lista in alto, si possono ricercare anche dispositivi che ancora non lo sono, così da poterli connettere. In questo caso possiamo avviare la ricerca tramite un pulsante che ci mostrerà un ulteriore lista dei dispositivi che sono all'interno del raggio d'azione del nostro dispositivo e che sono a noi visibili. Oltre che per un fattore estetico e anche di praticità abbiamo deciso di rendere questa activity simile ad Dialog, inserendo nel `AndroidManifest.xml` questo comando:

```
android:theme="@android:style/Theme.Holo.Dialog"
```

- *FallDetector:* questa classe, in maniera eguale al modulo smartphone genera la grafica dinamica dell'applicazione. Esso infatti disegna i due grafici visibili in *Figura 3.3* prendendo i valori direttamente dai sensori. In maniera conseguenziale sono presenti anche gli algoritmi per rilevare le cadute. Andremo qui di seguito a commentarlo:

  Innanzitutto viene implementato l'interfaccia `SensorEventListener` per gestire i sensori che saranno registrati tramite il metodo `registerListeners()` così da attivare un listener per i sensori che ci servono. Uno dei metodi della nostra interfaccia `SensorEventListener` è di fondamentale importanza per questa classe, il cosiddetto `onSensorChanged(SensorEvent event)`,esso è in funzione ogni volta che i valori del sensore rileva un cambiamento. All'interno è stato inserito tutto l'algoritmo di fall detection. Andremo ora a presentarne un frammento, partendo dal grafico che si preoccupa dell'**impatto**:

```
if(event.sensor.getType()==Sensor.TYPE_ACCELEROMETER)
{
     // Calcola RSS [sqrt(pow(x)2 + pow(y)2 +
pow(z)2)

rss=(float)Math.sqrt(Math.pow(event.values[0],2)

+Math.pow(event.values[1],2)
                          +Math.pow(event.values[2],2));
     // se RSS è maggiore di 2
if(rss>RssTreshold*SensorManager.STANDARD_GRAVITY)
{
granny.impact_detected = true; // flag per inviare i
dati bluetooth nel ciclo successivo.
          if(granny.VveTime == 0)
          {
               granny.RssVal = rss;//salviamo il
```

```
    valore rss
                    granny.RssTime=date.getTime();//segna
    tempo
              }
```

In seguito viene generata la parte grafica, cioè verrà disegnata la funzione che andrà a sovrapporsi nel grafico, già generato grazie alla classe *GraphView*.

In maniera analoga viene calcolata anche la **velocità verticale**, e se anche questa supera la soglia prefissata verrà inviato un messaggio bluetooth nel ciclo successivo. Infine andiamo a mostrare il frammento di codice che, nel caso in cui fosse rilevata valore "anomalo" (cioè che superi la soglia), questo verrebbe prontamente inviato allo smartphone:

```
if (granny.impact_detected || granny.Vve_detected)
            {
              if(granny.impact_detected)
              {
                 granny.sendMessage("I"String.valueOf
                                        (granny.RssTime));
                       //manda un messaggio bluetooth allo
smartphone
                 granny.impact_detected = false;
               }
          else if (granny.Vve_detected)
          {
                 granny.sendMessage("V"+String.valueOf
                                         (granny.VveTime));
            // manda un messaggio granny allo smartphone
                 granny.Vve_detected = false;
               }
      }
```

Se l'**impatto** oppure la **velocità verticale** superano le loro soglie nel momento in cui si ricercano, allora verranno attivati dei flag, i quali nel ciclo successivo ci faranno accedere all'interno del istruzione if sopracitata. Il frammento di codice invia un messaggio Bluetooth distinto in base a quale soglia è stata superata. All'interno della stringa inviata tramite il metodo *sendMessage(String)* abbiamo posto una lettera identificativa prima del

valore, così da poter capire quale valore sia stato inviato allo smartphone. Infine il flag viene ristabilito ad un valore falso, così da poterlo riutilizzare nuovamente.

- *`GraphView:`* questa classe gestisce tutta la grafica statica. Attraverso oggetti Canvas vengono generati i due grafici, come visibile in *Figura 3.3*, quello inerente la **Velocità verticale** e quello di **Caduta**. Estendendo alle `View` in questo modo:

```
public class GraphView extends View
```

  si può utilizzare il metodo `onDraw(Canvas canvas)` per disegnare i nostri grafici.

- *`MyLocalService:`* è la classe adibita al compito di servizio in background, in maniera identica a quella del modulo smartphone. Esso viene attivato automaticamente quando nello smartphone viene attivato il service. Infatti appena prima di attivarlo, viene inviato un messaggio Bluetooth al device posto al polso, il quale richiama nella propria activity principale (*SmartGrannyActivity*) l'intent per il service. La classe estende i *`Service`* e implementa *`SensorEventListener`* questo per poter prendere i segnali generati dall'accelerometro, così da poter riutilizzare l'algoritmo già implementato. Viene poi creato il metodo `getSensor(event)` al cui interno è implementato l'algoritmo per carpire i dati presi dal sensore. Questo metodo è a sua volta richiamato all'interno del metodo `onSensorChanged(SensorEvent event)` così da poterlo richiamare ogni volta che rileviamo un cambiamento di valore nei nostri sensori. Nel caso in cui si rilevi un campione che supera la soglia prefissata, esso invierà i dati relativi al momento in cui questo è successo. Sarà poi lo smartphone a gestire il ProgressDialog per segnalare la possibile caduto (o un falso positivo).

- *`SmartGrannyActivity:`* questa è l'activity che ci appare appena attiviamo

la nostra applicazione, anche visibile in *Figura 3.3*. Essa è incaricata a far interagire le diverse componenti all'interno dell'applicazione. Esempi sono:

- L*a gestione dell'action bar*, con il menu in alto possiamo vedere lo stato del Bluetooth, ricercare i device per la connessione Bluetooth cliccando sul primo tasto partendo da sinistra e con il secondo renderci visibili per una possibile connessione Bluetooth non paired.
- *Metodi per l'invio di messaggi Bluetooth*, `sendMessage(String message)` è il metodo adibito all'invio di un messaggio in una stringa, castandolo poi, all'interno del metodo in un vettore di byte (così è richiesto dallo standard). Infine chiediamo al thread `ConnectedThread` richiamato tramite il metodo `write(byte[] message)`, posto nella classe `BluetoothService` di inviarne il contenuto al device all'altro capo della connessione stabilita.
- *Attivazione Service,* per la gestione dei servizio in background è stato utilizzato il metodo `startService(serviceIntent)` con il quale far partire il service attraverso un intent. Questo è attivo solo nel momento in cui viene inviato uno speciale flag identificativo dallo smartphone tramite un messaggio Bluetooth da parte dello smartphone. Questo per poter creare un servizio in background a tutte e due i dispositivi in contemporanea, senza che l'utente se ne debba occupare.

3.4 Campionamenti

Durante l'implementazione, per poter dimostrare che l'applicazione funzionasse realmente abbiamo deciso di “utilizzare” alcuni volontari che simulassero diverse attività quotidiane (il cui acronimo è ADL). Sono stati effettuati 112 campionamenti prodotti da 14 persone. Essi sono stati eseguiti da 9 uomini e 5 donne in età compresa tra i 18 e i 60 anni.E' stato preso in considerazione, come già citato in precedenza lo studio di Bourke [30] come raffigurato in *Figura 3.5,*riportando dai risultati dello studio sopracitato la parte relativa all'algoritmo che integra al suo interno **l'impatto** la **velocità verticale** e la **postura.** I test eseguiti da noi sono 8 rispetto ai 12 riportati in Bourke

[30] poiché non abbiamo potuto testarlo con persone di età avanzata (sopra i 65 anni). Essi sono così suddivisi:

1. *Mentre ci si stende nel letto*, in questo caso, come previsto i 14 volontari che si sono prestati ai campionamenti, l'app non hanno dato alcun falso negativo.
2. *Mentre si ci siede in una poltrona*, lo stesso risultato lo abbiamo ricavato anche per questo tipo di test.
3. *Mentre si ci siede in una sedia*, essendo molto simile al precedente test, anche questo come previsto i risultati non hanno tardato a riconfermare i precedenti.
4. *Mentre si ci siede al bagno*, abbiamo cercato di prendere in considerazione un numero alto di condizioni in cui ci si siede
5. *Mentre si ci siede in macchina*, ugualmente al precedente test i risultati si sono rilevati di successo in maniera completa, in quanto anche in questo caso la nostra non ha percepito nessuna caduta.
6. *Mentre ci si siede nel letto*, anche in quest'ultimo caso di posizione seduta, il nostro applicativo non ha smentito i precedenti risultati.
7. *Mentre si salgono le scale*, in questo caso specifico, diverso da quelli precedenti, nei 14 campionamenti testati l'app si è riconfermata pienamente efficiente.
8. *Mentre si cammina per circa 10 metri*, questo ultima situazione ha dimostrato che l'applicazione è all'altezza dell'algoritmo che abbiamo utilizzato, poiché anche in quest'ultima casistica non si sono presentati falsi negativi.

	VELOCITY+IMPACT+POSTURE			
	UFT	UFT_D	*Profile FE*	*Profile RE*
Sensitivity (%)	100	100	100	100
Specificity (%)				
Elderly lying bed	100	100	100	100
Elderly Sitting armchair	100	100	100	100
Elderly sitting kitchen chair	100	100	100	100
Elderly sitting toilet seat	100	100	100	100
Elderly sitting car seat	100	100	100	100
Elderly Sitting bed	100	100	100	100
Elderly climbing stairs	100	100	100	100
Elderly walking 10m	100	100	100	100
Young lying bed	100	100	100	100
Young sitting kitchen chair	100	100	100	100
Young sitting bench	100	100	100	100
Young walking 10m	100	100	100	100
Total specificity (%)	100	100	100	100

Figura 3.5: Risultati degli studi di Noury [30]

Inoltre sono stati acquisiti, sempre in fase di test 70 campionamenti da 14 persone in età compresa tra 18 e 60 anni. Questo per verificare se i sensore di accelerazione all'interno dei nostri smartphone e del nostro smartwatch fossero simili, in termini di sensibilità rispetto al sensore utilizzato da Noury [30] nel suo studio (Freescale (Freescale Semiconductor, Inc., 6501 William Cannon Drive West, Austin, TX 78735, USA) MMA7261QT). Per far ciò abbiamo creato un applicazione ad-hoc che potesse in primis acquisire i dati in fase di caduta, dopodiché generare un grafico dimostrativo della stessa e infine poter salvare il tutto per analizzarlo in seguito. Nella fattispecie ogni volontario ha simulato 5 cadute di vario tipo su un materasso:

1. *Caduta da fermo:* in questo caso si vuole simulare una caduta dovuta ad un ipotetico malore da parte della persona.

2. *Caduta camminando:* in questo caso simuliamo una caduta accidentale dovuto ad un ostacolo mentre si compiono le normali attività giornaliere.

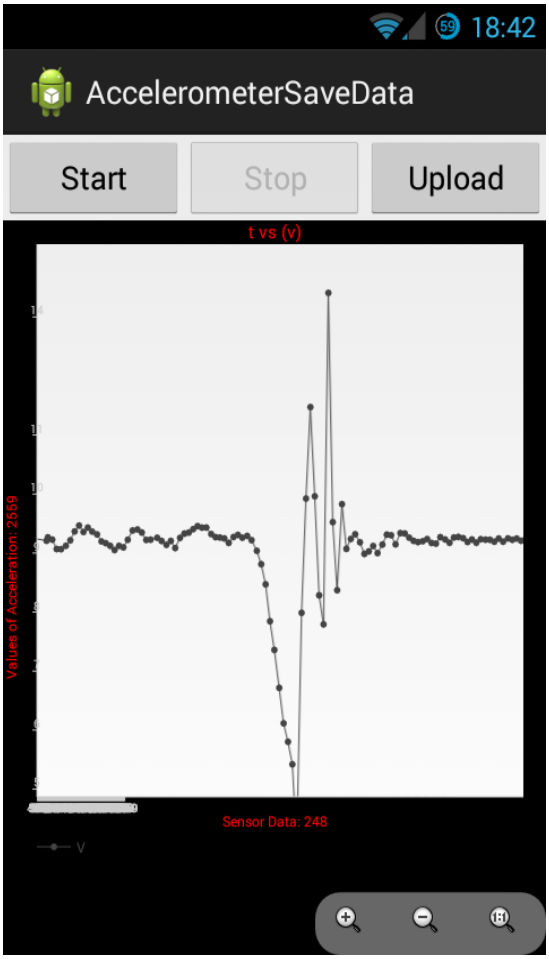
3. *Caduta correndo:* questo campionamento, essendo un caso estremo per quel che riguarda l'accelerazione dovuta ad una caduta, è stato preso come limite superiore per poter trovare la soglia giusta per la nostra applicazione.

4. *Caduta all'indietro:* è possibile che per una persona anziana, data l'avanzata età, gli arti inferiori siano instabili e quindi il corpo possa sbilanciarsi all'indietro.

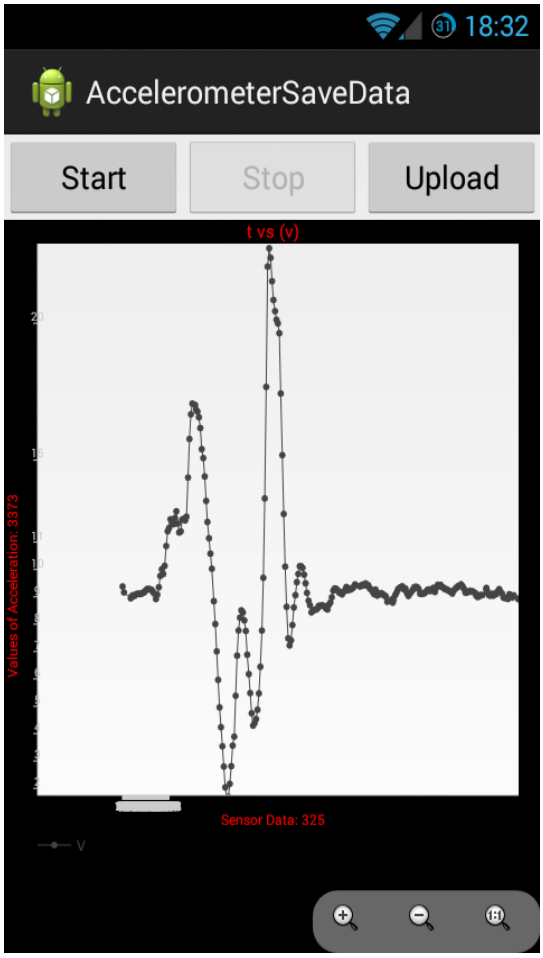
5. *Caduta ponendo le mani avanti:* anche questo caso è stato testato poiché l'accelerazione in una situazione reale è plausibile supporre che l'anziano ponga avanti le mani anche se i riflessi sono più lenti.

Per concludere diciamo che questi campionamenti sono stati analizzati attraverso il programma MATLAB [47], abbiamo utilizzato la funzione di plot del programma per generare tutti i grafici e abbiamo evidenziato come i risultati ottenuti si attengono a quelli esposti nel lavoro Noury [30]. Infatti la soglia imposta per la nostra applicazione è di 2,8g. Riportiamo qui di seguito alcuni screenshot del nostro applicativo dopo aver

catturato due tipi di cadute diverse. La prima è una caduta mentre si simula una camminata (2), mentre la seconda è invece mentre si corre (3).



3.6 Campionamento di un volontario che simula la caduta mentre sta cammina



3.7 Campionamento di un volontario che simula la caduta mentre sta correndo

Capitolo 4

Conclusioni

In questo capitolo conclusivo andremo ad esporre quelle che sono le nostre considerazioni personali sugli sviluppi futuri di questo progetto derivanti dall'analisi e dallo studio degli argomenti trattati nei precedenti capitoli. Nonostante l'applicazione da me sviluppata possa essere definita formalmente completa per rispondere agli scopi e alle esigenze per cui è nata, è indubbio che possa essere migliorata sul fronte del interfaccia, rendendola più semplice e maggiormente user-friendly, e sul fronte della stabilità dopo un'ulteriore e più diffusa fase di test. Per quanto riguarda il miglioramento dell'interfaccia grafica, sarà necessario ristrutturarla venendo incontro alle esigenze di un pubblico non specializzato, inoltre potrebbe essere opportuno ridisegnare il layout secondo gli standard indicati da Apple [37] per una futura conversione in versione iOS, conversione che ci viene suggerita dalla diffusione non trascurabile di questo sistema operativo mobile, come ci ricordano le statistiche Gartner [Tabella 2.1] . Per quanto concerne la struttura del codice il suo miglioramento non può che essere subordinato ad un periodo di prova che permetta l'emergere degli errori rimasti irrisolti. Si può guardare al futuro di questo progetto non solo focalizzandoci su quanto sia migliorabile, ma anche sulle sue possibili evoluzioni. Ciò che ho intenzione di fare nel proseguimento della mia carriera accademica è impegnarmi a portare avanti questo progetto nell'ottica di virare verso una soluzione maggiormente ibrida. La concretezza di questa prospettiva è data dal carattere modulare, che sin dalle prime fasi, ho voluto imprimere al progetto. Ritengo possibile implementare un sistema di controllo basato su telecamere ad infrarossi da affiancare all'applicazione mobile. Tale soluzione, coordinata interamente tramite server, porterebbe una maggiore efficacia nella risposta alle esigenze di sicurezza e di tutela della privacy dell'utente.

Per quanto riguarda la portata del progetto sul mercato odierno, abbiamo già accennato in precedenza come la richiesta di applicativi incentrati sulla tecnologie assistenziali stia crescendo con forza costante. Ciò che però può permettere a questo progetto di

convogliare una fetta sempre più larga di questa crescente domanda sarà mantenere il suo futuro sviluppo strettamente legato ad una logica di integrazione con progetti che possano rafforzare l'affidabilità e, al tempo stesso, ampliare le funzionalità del progetto originale. Gli strumenti che saranno necessari per il proseguimento nello sviluppo di questo progetto non si differenziano molto da quelli utilizzati fino ad ora. L'esempio più lampante di come sarà possibile il contenimento dei costi negli sviluppi futuri è sicuramente l'ambiente di sviluppo Eclipse con l'aggiunta del plug-in ADT, soluzione adatta a farci raggiungere elevati livelli di complessità al progetto senza alcun costo in virtù della sua licenza (Eclipse Public License). Per quanto riguarda invece le risorse umane, è chiaro che le promettenti possibilità di ampliare ulteriormente il progetto ci suggeriscono la necessità di affidare ciò alle competenze non di un singolo individuo, ma di un team che possa occuparsi della varietà delle problematiche che questa sfida ci pone.

Bibliografia

1. http://www.msd-italia.it/altre/geriatria/sez_2/sez2_20%20.html

2. Kannus P, Sievänen H, Palvanen M, Järvinen T, Parkkari J. Prevention of falls and consequent injuries in elderly people. Lancet 2005;366:1885-93.

3. Mancini C, Williamson D, Binkin N, Michieletto F, De Giacomi GV; Gruppo di Lavoro Studio Argento. Epidemiology of falls among the elderly Ig Sanita Pubbl. 2005 Mar-Apr;61(2):117-32.

4. Rubenstein LZ. Fall in older people: epidemiology, risk factors and strategies for prevention. Age and Ageing 2006; 35-S2: ii37–ii41doi:10.1093/ageing/afl084ii37

5. http://www.epicentro.iss.it/passi-argento/default.asp

6. Hirsch, T., Forlizzi, J., Hyder, E., Goetz, J., Stroback, J. & Kurtz, C. 2000, The ELDer project: Social, emotional, and environmental factors in the design of eldercare technologies, Proceedings ofthe Conference on Universal Usability, pp. 72.

7. Mann, W.C., Ottenbacher, K.J., Fraas, L., Tomita, M. & Granger, C.V. 1999, Effectiveness of assistive technology and environmental interventions in maintaining independence and reducing home care costs for the frail elderly a randomized controlled trial, Archives of Family Medicine, vol. 8, no. 3, pp. 210-217.

8. Giuliani, M.V., Scopelliti, M. & Fornara, F. 2005, Elderly people at home:Technological help in everyday activities, Proceedings - IEEE International Workshop on Robot and Human Interactive Communication, pp. 365.

9. Miskelly, F.G. 2001, Assistive technology in elderly care, Age and Ageing, vol. 30, no. 6, pp. 455-458.

10. O'Brien, A. & Ruairì, R.M. 2009, Survey of assistive technology devices and applications for aging in place, 2nd International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies, and Services - CENTRIC 2009, pp. 7.

11. Harvey AH Baker SP. Fall injuries in the elderly. clinics in geriatric medicine. volume 1, pages 501–512, 1985.

12. Rubenstein LZ. Falls in older people: epidemiology, risk factors and strategies for prevention. Age and Ageing 2006; 35-S2: ii37–ii41doi:10.1093/ageing/afl084ii37

13. Prof. Giuseppe Lullo, Il sistema GPS,Anno Accademico 2008/2009, per il corso di Elettronica delle comunicazioni.

14. C.J. Lord and D.P. Colvin. Falls in the elderly: Detection and assessment. pages 1938 –1939, oct. 1991.

15. G. Williams, K. Doughty, K. Cameron, and D.A. Bradley. A smart fall and activity monitor for telecare applications. volume 3, pages 1151 –1154 vol.3, oct. 1998.

16. M.J. Mathie, J. Basilakis, and B.G. Celler. A system for monitoring posture and physical activity using accelerometers. volume 4, pages 3654 – 3657 vol.4, 2001.

17. N. Noury, P. Barralon, G. Virone, P. Boissy, M. Hamel, and P. Rumeau. A smart sensor based on rules and its evaluation in daily routines. volume 4, pages 3286

– 3289 Vol.4, sept. 2003.

18. Kerdegari, H. Samsudin, K. ; Ramli, A.R. ; Mokaram, S. Evaluation of fall detection classification approaches, Intelligent and Advanced Systems (ICIAS), 2012 4th International Conference on (Volume:1) pages 131 – 136

19. Murphy J. · Isaacs B. , The Post-Fall Syndrom, Gerontology 1982;28:265–270 (DOI: 10.1159/000212543)

20. [Bourke et al, 2007] Bourke, A.K., O'Brien, J.V. & Lyons, G.M. 2007, Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm, Gait and Posture, vol. 26, no. 2, pp. 194-199. [Bourke, Lyons, 2008] Bourke, A.K. & Lyons, G.M. 2008, A .

21. G. Pérolle1, P. Fraisse2, M. Mavros3, I. Etxeberria4 , Automatic Fall Detection and Activity Monitoring for Elderly , Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier

22. Hwang, J.Y. ; Dept. of Biomedical Eng., Seoul Nat. Univ., South Korea ; Kang, J.M. ; Jang, Y.W. ; Kim, H.C. Engineering in Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE (Volume:1) pages 2204 - 2207

23. [Rougier et al, 2006] Rougier, C., Meunier, J., St-Arnaud, A. & Rousseau, J. 2006, Monocular 3D head tracking to detect falls of elderly people, Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings, pp. 6384.

24. Foroughi, H., Rezvanian, A. & Paziraee, A. 2008, Ro- bust fall detection using human shape and multi-class support vector machine, Proceedings - 6th Indian Conference on Computer Vision, Graphics and Image Processing, ICVGIP 2008, pp. 413.

25. Alwan, M. Rajendran, P.J. ; Kell, S. ; Mack, D. ; Dalal, S. ; Wolfe, M. ; Felder, R,. A Smart and Passive Floor-Vibration Based Fall Detector for Elderly, Information and Communication Technologies, 2006. ICTTA '06. 2nd (Volume:1),pages1003–1007.

26. Tzeng, H.W., Chen, M.Y. & Chen, J.Y. 2010, Design of fall detection system with floor pressure and infrared image, 2010 International Conference on System Science and Engineering, ICSSE 2010, pp. 131.

27. http://www.xbox.com/it-IT/kinect/

28. https://play.google.com/store?hl=it

29. N. Noury, A. Fleury, P. Rumeau, A.K. Bourke, G.O. Laighin, V. Rialle, and J.E. Lundy. Fall detection - principles and methods. Pages 1663–1666, 2007.

30. A.K. Bourke, P. van de Ven, M. Gamble, R. O'Connor d, K. Murphy, E. Bogan, E. McQuade, P. Finucane, G. OLaighin, J. Nelson, Evaluation of waist-mounted tri-axial accelerometer based fall-detection algorithms during scripted and continuous unscripted activities , Journal of Biomechanics, Volume 43, Issue 15,16 November 2010, Pages 3051–3057.

31. http://android-developers.blogspot.it/2010/05/dalvik-jit.html

32. http://developer.android.com/sdk/index.html

33. Android - Guida per lo sviluppatore, Massimo Carli, Apogeo, 2010

34. Java 7 - Guida Completa,Pellegrino Principe, Apogeo, settembre 2012

35. http://www.idc.com/

36. http://www.gartner.com/technology/home.jsp

37. https://www.apple.com/it/ios/

38. http://www.windowsphone.com/it-IT/phones

39. http://www.android.com/

40. http://www.sqlite.org/

41. http://www.microsoft.com/silverlight/

42. http://windows.microsoft.com/it-it/internet-explorer/download-ie

43. http://www.opengl.org/

44. http://developer.android.com/sdk/installing/studio.html

45. http://developer.android.com/sdk/index.html

46. http://www.netcarity.org/About.11.0.html

47. http://www.mathworks.it/products/matlab/