ALMA MATER STUDIORIUM · UNIVERSITA' DI BOLOGNA

Campus di Cesena - Scuola di Scienze

Corso di Laurea in Scienze e Tecnologie Informatiche

Sensing della presenza di scale tramite smartphone

Relazione finale in

Sistemi Multimediali

Relatore

Prof.ssa PAOLA SALOMONI

Correlatore

Dott.ssa CATIA PRANDI

Presentata da

ANDREA VECCHIOTTI

III sessione

Anno Accademico 2013-2014

Indice

Introduzione

Negli ultimi cinquant'anni si è assistito ad una notevole rivoluzione tecnologica avvenuta grazie a invenzioni sempre più sensazionali come il transistor, sostituto delle valvole termoioniche, che ha permesso circuiti elettronici sempre più piccoli, più complessi e con consumi energetici ridotti. A pari passo con la tecnologia segue l'informatica; con configurazioni hardware sempre più potenti è stata possibile la realizzazione di sistemi operativi più reattivi e applicazioni con costi computazionali sempre più elevati. Figlia di queste due evoluzioni è lo smartphone o telefono intelligente.

Lo smartphone è un dispositivo elettronico che incorpora sia le funzioni di un telefono cellulare, sia le funzioni di un computer. Il dispositivo è dotato di una grande quantità di memoria e da elevate capacità di calcolo in rapporto alla dimensione del dispositivo stesso, senza però trascurare il consumo di energia. Inoltre, a differenza dei vecchi telefoni cellulari, gli smartphone contengono dei veri sistemi operativi che consentono all'utente un interazione migliore e permettono l'esecuzione di applicazioni sempre più complesse. Oltre a quelle già descritte, il dispositivo contiene una buona quantità di sensori utili ad aumentarne l'efficienza o le funzionalità di alcune applicazioni.

Grazie alle sue funzionalità, lo smartphone si è guadagnato un enorme fetta di mercato e oggigiorno tutte le persone viventi in un paese tecnologicamente progredito ne possiede almeno uno, che siano per motivi personali o di lavoro. Proprio per questo,

il dispositivo verrà utilizzato come strumento per il raggiungimento degli obiettivi di tesi.

Lo scopo di questa tesi è di usare i sensori dello smartphone per cercare di rilevare alcune tra le attività umane più frequenti, come il salire una scala. Si presenta quindi lo smartphone in termini di hardware elencando i possibili sensori contenuti in esso e descrivendone le qualità. Vengono presentati i principali sistemi operativi mobili adottati per questi tipi di dispositivi mostrando un confronto tra di essi in termini di pregi e difetti e viene narrata un minimo di storia relativa ai dispositivi mobili in generale spiegando come lo smartphone abbia sostituito il classico telefono cellulare.

Nel documento di tesi verranno poi presentati i principali strumenti con cui verrà sviluppata l'applicazione di rilevamento e verranno presi in considerazione due programmi di sviluppo per due linguaggi e tipologie di smartphone differenti: Windows Phone e Android. Tuttavia per lo sviluppo del progetto di tesi verrà impiegata solo la strumentazione relativa a Windows Phone. In seguito verrà presentata l'applicazione d'ausilio alla progettazione dei metodi, Scilab.

Per il rilevamento dell'attività verranno esposti due diversi metodi e ne verranno mostrati il codice per l'implementazione e i relativi risultati. Questi verranno poi discussi e confrontati per analizzare l'affidabilità di entrambi i metodi.

La tesi è così suddivisa:

- nel primo capitolo viene introdotto lo smartphone, viene esposta la storia di tale prodotto, vengono presentati i più famosi sistemi operativi adoperati dagli smartphone e vengono descritti tutti i sensori che è possibile al loro interno;
- nel secondo capitolo vengono descritti gli strumenti utili per sviluppare un'applicazione per Windows Phone e Android, si parla di macchine virtuali utili per testare le applicazioni, si enunciano i requisiti generali affinchè è possibile sviluppare e testare un'applicazione sul dispositivo fisico e si descrive lo strumento con cui è possibile ipotizzare, progettare e testare gli algoritmi per il progetto;

- nel terzo capitolo si descrive cosa si ottiene dall'accelerometro di uno smartphone, come organizzare i dati grezzi, si mostra del codice di programmazione in Windows Phone e Android per l'ascolto del sensore di accelerazione e vengono proposti i metodi di progetto;
- nel quarto capitolo, dopo aver effettuato dei test sul campo, vengono mostrati i risultati ottenuti dai metodi di progetto e vengono illustrate possibili migliorie

1 Introduzione allo smartphone

In questo primo capitolo si inizierà descrivendo cosa è lo smartphone, da cosa è caratterizzato e quali sono i suoi componenti.

Successivamente verranno discussi il modo in cui questo si è diffuso e le ragioni, esponendo poi il motivo per cui è stato usato questo tipo di dispositivo per lo sviluppo di questo lavoro di tesi.

1.1 Cos'è uno smartphone

Uno smartphone (in italiano telefono intelligente) è un dispositivo portatile (mobile device) che abbina funzionalità di gestione dei dati personali e di telefono, arrivando fino a simulare completamente le funzionalità di un personal computer. Deriva quindi dall'evoluzione di un PDA (Personal Digital Assistent) a cui si aggiungono funzioni di un telefono cellulare.

Lo smartphone è caratterizzato da una capacità di calcolo, memoria e connessione dati molto più avanzate rispetto ai normali telefoni cellulari e ha un sistema operativo apposito per dispositivi mobili.

Per questi motivi è possibile installarvi ed eseguire applicazioni di complessità maggiore rispetto a quanto precedentemente fattibile con i vecchi dispositivi mobili.

1.2 Da cellulare a smartphone

1.2.1 Utilità di un telefono cellulare

Appena prima dell'invenzione del cellulare, la comunicazione orale tra due persone distanti tra loro, avveniva tramite la telefonia fissa attraverso i PTP (Posti di Telefonia Pubblica) o con apparecchi fissi situati in uffici e in quasi tutte le case di privati [1].

La telefonia fissa, in quanto tale, obbliga chiunque ne voglia usufruire di trovarsi in determinati punti per chiamare e attendere chiamate e non permette a chi le sta usando di spostarsi. Proprio per questo motivo era necessaria una soluzione alternativa alla telefonia fissa: la telefonia mobile o cellulare.

Il cellulare [2] è un apparecchio radio mobile terminale ricetrasmittente per la comunicazione in radiotelefonia sull'interfaccia radio di accesso di una rete cellulare; si collega alla rete telefonica e alla rete dati tramite centrali di smistamento a loro volta collegate a delle stazioni radio base capaci di tenere connessi più apparecchi mobili in una certa area di copertura e secondo le frequenze supportate.

Il telefono cellulare consente quindi di avere disponibile un collegamento telefonico quando ci si trova nel raggio di copertura di una delle stazioni radio (o "celle radio"). Grazie a questa tecnologia i dispositivi mobili permettono alle persone di essere sempre reperibili tramite chiamate.

Dopo le chiamate, si è sentita la necessità d'introdurre gli SMS (Short Message Service) che permettessero ai possessori di cellulari di avere un secondo modo per contattare gli altri che fosse più economico rispetto alle chiamate di una certa longevità e che permettesse di rendere la comunicazione asincrona.

1.2.2 Necessità di uno smartphone

La comunicazione tra due persone distanti tra loro non è l'unico problema. Si è venuta a creare la necessità di avere a portata di mano dispositivi che simulassero, almeno in parte, le funzioni di computer che allora erano molto costosi e poco pratici in termini di dimensioni o peso. Viene introdotto quindi il PDA (Personal Digital Assistant) o computer palmare.

Il PDA [3] è un computer di dimensioni molto contenute e dotato di uno schermo tattile. Inizialmente le sue funzioni erano limitate all'agenda elettronica, l'orologio, la calcolatrice, il calendario e la rubrica contatti; nel tempo il PDA si è evoluto fino a ricoprire tutte le funzionalità dei computer; tuttavia, il trasporto di un PDA e di un cellulare rimane scomodo. A questo punto sorge il bisogno di un dispositivo che incorpori sia le funzioni di un cellulare sia le funzioni di un PDA.

La IBM (International Business Machines Corporation) progettò nel 1992 il primo smartphone di nome Simon, commercializzato poi dalla BellSouth dal 1993. Esso incorporava, oltre alle funzioni di telefono, il calendario, la rubrica, l'orologio, black notes, funzioni di email e giochi e un pennino con cui scrivere sullo schermo.

Con l'avanzare della tecnologia e l'introduzione di sistemi come UMTS (Universal Mobile Telecommunications System) o LTE (Long Term Evolution), gli smartphone permettevano una capacità di connessione dati superiore rispetto a quella dei cellulari.

1.2.3 Mercato degli smartphone

Nonostante i primi smartphone siano stati progettati nel 1992, non ci fu una vera diffusione di questi prodotti nel mercato a causa del loro elevato prezzo, della loro autonomia molto limitata e delle dimensioni inappropriate. Solo nel 2006 lo smartphone passò dal mercato di nicchia al mercato di massa conquistando il 15% del mercato grazie alla vendita di più di 100 milioni di unità [4].

Secondo Philip Solis, il Senior Analyst Mobile Wireless Research di ABI, sono cinque le cause per cui il boom degli smartphone è avvenuto solo in questo periodo:

- una crescente domanda di connessioni dati, grazie alla diffusione di mobile email e instant messaging;
- calo dei prezzi dovuto dall'aumento dei volumi;
- differenziazione dei prodotti con scelta più ampia;
- fattori di forma più gradevoli e autonomia dei dispositivi più elevata;
- introduzione dei moduli Wifi nei dispositivi.

Dal 2006 le vendite degli smartphone aumentarono fino a quando, nel 2011, superano persino le vendite dei cellulari tradizionali [5].

Il mercato degli smartphone è tutt'oggi in continua crescita; i principali produttori di smartphone sono Samsung, Apple, Nokia, HTC, Sony, LG, Lenovo, Huawei e altri di non poca importanza. Ogni produttore di smartphone installa sui propri dispositivi un determinato OS.

1.3 Sistemi Operativi Mobile

Per definizione "un sistema operativo è un insieme di programmi (software) che gestisce gli elementi fisici di un calcolatore (hardware), fornisce una piattaforma ai programmi applicativi e agisce da intermediario tra l'utente e la struttura fisica del calcolatore" [6], un qualsiasi dispositivo mobile come uno smartphone ha bisogno quindi di un sistema operativo per essere funzionale.

Dal 2006 ad oggi, sono emersi molti OS tutti differenti tra loro e, sapere quanti OS ci sono nel mercato e in che modo funzionano, aiuta a capire il motivo per cui lo smartphone è stato preso in considerazione per il progetto di tesi.

1.3.1 Windows Phone

WP (Windows Phone) è una famiglia di sistemi operativi per smartphone di Microsoft, presentata nel 2010 al Mobile World Congress. WP è destinato al mercato consumer, a differenza di Windows Mobile che è dedicato al mercato enterprise; inoltre, rispetto a quest'ultimo, è incompatibile poichè si presenta radicalmente ridisegnata nella struttura e nell'interfaccia.

Il primo WP di casa Microsoft fu Windows Phone 7 e insieme ad esso fu rilasciato un pacchetto software gratuito per sviluppare applicazioni per questo OS, contenente VS2010 Express (Visual Studio 2010 Express), Silverlight, Expression Blend, XNA Game Studio e WPE (Windows phone emulator). Nel 2011 Steve Ballmer, allora amministratore delegato Microsoft, e l'amministratore delegato di Nokia, Stephen Elop, annunciarono una partnership tra le loro aziende in base alla quale WP sarebbe

diventato il principale sistema operativo dei futuri smartphone Nokia. Tuttavia sono 17 i partner che puntano a Windows Phone, tra cui Samsung, HTC, Sony, LG.

Nel 2012 fu annunciato e immesso nel mercato WP8 attraverso la vendita di Nokia Lumia 820 e 920. WP8 prevede il kernel Windows NT al posto di Windows CE presente in WP7, IE10 (Internet Explorer 10), il pacchetto Office e altre migliorie come il supporto NFC, Wi-Fi direct e il supporto ai processori multi-core fino a 64 cores.

Figura 1.1: Schermata iniziale di WP fonte [7]

L'interfaccia usata da WP si chiama Modern UI; la schermata principale, chiamata Schermata di Start, è composta da Live Tiles, che sono collegamenti ad applicazioni, funzioni o elementi individuali, come contatti o pagine internet, a forma di piastrelle. L'utente può aggiungere, ridimensionare, spostare o eliminare le tiles, che appaiono dinamiche grazie ad effetti grafici (come ribaltamenti e animazioni) e si aggiornano in tempo reale; per esempio la tile della email mostra all'utente il numero di mail non lette.

L'interfaccia utente di default ha un tema scuro, interscambiabile con un tema chiaro, così com'è modificabile il colore delle tiles che non hanno un'immagine indipendente. L'utente interagisce con il dispositivo WP attraverso tre tasti principali: Indietro, Start e Cerca.

- Indietro, rappresentato da una freccia rivolta verso sinistra, riporta alla schermata precedente e permette di chiudere un'applicazione se la si sta eseguendo; con una pressione prolungata sul tasto touch, è possibile vedere tutte le applicazioni aperte in background.
- Start, rappresentato dal logo di Windows, riporta alla schermata di Start.

- Cerca, rappresentato da una lente d'ingrandimento, permette di accedere alla ricerca integrata di Bing; con una pressione prolungata è possibile attivare i comandi vocali.

I restanti tasti previsti per un dispositivo windows phone sono per il volume, per l'accesso all'applicazione della fotocamera e per passare alla schermata di blocco. Attraverso il touchscreen è possibile interagire con le applicazioni di Windows Phone.

Per quanto riguarda la gestione delle app in background, il sistema risulta differente rispetto ad altri OS; le applicazioni in background non vengono eseguite ma ibernate nella memoria cache e rimosse dalla RAM andando quindi in pausa; appena l'utente sceglie di riaprire un'applicazione in background, l'applicazione viene ripresa dal punto in cui è stata interrotta l'esecuzione.

Questo sistema aiuta al dispositivo a risparmiare batteria aumentandone la durata e, nello stesso tempo, liberando memoria che sarà pronta per essere occupata da nuove applicazioni senza perdere di fluidità.

Il sistema di notifiche viene gestito dai server Microsoft e supporta app di sistema o di terze parti. L'utente può ricevere notifiche sotto forma di Toast, ovvero tramite un ban che appare nella parte superiore dello schermo.

Con l'aggiornamento di WP8 a WP8.1, viene introdotto l'Action Center, un menù a tendina attivabile tramite uno swype verso il basso dalla barra di stato e che mostra le notifiche non ancora visualizzate.

Tutte le applicazioni possono essere scaricate e installate dal Windows Phone Store; lo store è amministrato dalla Microsoft che deve prima approvare i software per prevenire ogni tipo di malware. Lo store microsoft conta più di 500 mila applicazioni.

1.3.2 iOS

Figura 1.2: Schermata iniziale iOS8 fonte [8]

iOS (precendemente iPhone OS) è un sistema operativo sviluppato da Apple per iPhone e altri dispositivi quali iPod e iPad.

Il sistema operativo è stato presentato nel 2007 al MacWorld Conference & Expo di San Francisco ed è entrato in commercio nello stesso anno con il primissimo iPhone con iPhone OS. Nel 2008 l'iPhone OS passa alla versione 2 insieme all'uscita dell'iPhone 3G. Tra le varie cose, l'aggiornamento consiste nell'inserimento nello store di applicazioni di terze parti. Un ulteriore aggiornamento viene pubblicato nel 2009 insieme alla vendita di iPhone 3GS; in questa versione vengono aggiunte le funzionalità di copia e incolla e di MMS. Nel 2010 l'OS passa alla versione 4, pubblicato con l'uscita dell'iPhone 4; il nome gli viene cambiato in iOS 4.2.1 e tra gli aggiornamenti rientra la funzione di multitasking per le applicazioni di terze parti.

Gli aggiornamenti continuano di anno in anno fino ad oggi con iOS 8; tra la versione 4 e la versione 8 si può notare uno stile grafico più minimale, un multitasking rivisitato e migliorato, vengono integrati i servizi di iCloud di Apple, viene aggiunta e rinnovata l'applicazione per le mappe e nuove funzioni e lingue per l'assistente vocale.

Per quanto riguarda l'interfaccia grafica, l'iOS si presenta con una schermata iniziale in cui in alto vi è una barra che mostra l'ora, il segnale, stato della batteria e stato d'attivazione dei sistemi di connessione (3G, Wifi, Bluetooth) e in basso vi sono le icone delle applicazioni che possono essere spostate e modificate a proprio piacimento.

Dalla versione 3 viene integrata la funzione Spotlight, accessibile scorrendo verso il basso dalla schermata di home, in cui è possibile ricercare nel dispositivo contatti, messaggi, email e altro.

Dalla versione 4 di iOS è possibile creare cartelle nella schermata principale semplicemente trascinando un'applicazione su di un'altra e dando un titolo ad essa. Se si ricevono delle notifiche dalle applicazini al suo interno, vengono segnalate sulla cartella. Ogni cartella può tenere dentro di se 12 applicazioni, limite esteso con l'aggiornamento a iOS7, dove le applicazioni vengono mostrate in più pagine della stessa.

Dalla versione 5.0, trascinando il dito dall'alto dello schermo verso il basso, viene mostrata una tendina con varie informazioni, quali meteo, borsa, notifiche dei social network e varie (come il promemoria).

Dalla versione 7 il centro notifiche è diviso in tre sezioni, il primo per il riepilogo degli eventi, il secondo per tutte le notifiche, il terzo per le notifiche perse; la terza sezione è stata rimossa dalla versione 8 di iOS per scambiarla con i widget.

Solo dalla versione iOS 4.0 si può iniziare a parlare di multitasking supportate da appropriate API (Application programming interface) quali:

- Audio di sottofondo: l'applicazione continua a funzionare in background fino a quando la riproduzione di contenuti audio o video non termina;
- Voice Over IP: l'applicazione viene sospesa quando una telefonata è in corso;
- Posizione di geolocalizzazione: l'applicazione informa sui cambiamenti di posizione;
- Notifiche push: ricezione di avvisi da server remoti anche quando l'applicazione alla quale sono destinati non è in esecuzione;
- Notifiche locali: l'applicazione può avvisare gli utenti di eventi in programma o di allarmi in background;
- Completamento delle attività: l'applicazione può chiedere al sistema del tempo extra per completare un determinato compito;

- Cambio rapido applicazione: l'applicazione non esegue alcun codice e può essere rimossa dalla memoria in qualsiasi momento, per poi riprendere dal punto in cui è stata interrotta.

1.3.3 Android

Figura 1.3: Schermata iniziale Android 4.4 fonte [9]

Nel 2003 fu fondata la società Android Inc. da Andy Rubin, Rich Miner, Nick Sears e Chris White; nel 2005 Google Inc. acquistò la società e in poco tempo svilupparono un sistema operativo sulla base del kernel Linux, chiamato Android. Android è stato progettato principalmente per smartphone e tablet, con interfacce utente specializzate per televisori, automobili, orologi da polso, occhiali e altri.

L'OS, a parte per i driver inclusi dai produttori di dispositivi, è Free and Open Source Software ed è distribuito sotto i termini di licenza libera Apache 2.0 [10].

La presentazione ufficiale di Android avvenne nel 2007 dalla OHA (Open Handset Alliance), un consorzio di aziende del settore Hi Tech che include, oltre a Google, produttori di smartphone come HTC e Samsung e produttori di microprocessori come Qualcomm. Il primo dispositivo Android lanciato sul mercato fu Dream di HTC nel 2008.

Dal 2008 ad oggi sono state rilasciati 49 aggiornamenti più o meno importanti passando dall'API level 1 all'API level 21. In questi aggiornamenti, oltre a risolvere numerosi bug, ottimizzazioni e migliorie della UI, vengono introdotti il supporto per i widget, la ricerca vocale e testuale per i contenuti presenti in locale e sul Web, funzionalità di fotocamera, sincronizzazione dell'account Google, supporto al multitouch, supporto alla tecnologia Adobe Flash (unico tra gli OS mobile), il

download manager, supporto per gli schermi ad alta risoluzione, supporto a Hangouts per il sistema di messaging e MMS e altre funzionalità di minore importanza.

Oggi Android ricopre la fetta più grande del mercato degli smartphone e la fetta più grande del mercato dei dispositivi di ogni genere, che sia una tv, un tablet o un microonde.

L'interfaccia utente di Android è basata sul concetto di direct manupulation per cui si utilizzano gli ingressi mono e multi-touch come striscate, tocchi e pizzichi sullo schermo per manipolare gli oggetti visibili sullo stesso. La risposta all'input dell'utente è stata progettata per essere immediata e tentare di fornire un'interfaccia fluida.

Per quanto riguarda la UI, Android si presenta con una schermata principale simile a quella del desktop di Windows; questa schermata è occupata sia dalle icone delle applicazioni che dai widget, gadget con varie funzioni (orologio, calendario..); inoltre, la homescreen può essere costituita da più pagine tra cui l'utente può scorrere avanti e indietro.

Nella parte superiore dello schermo si trova una barra di stato, che mostra le informazioni sul dispositivo e la sua connettività; trascinando la barra di stato verso il basso compare una schermata in cui si possono visualizzare le varie notifiche lanciate dalle applicazioni (telefono, messaggi, email, social network); le notifiche vengono cancellate dalla schermata una volta lette.

La gestione delle applicazioni in Android è più complessa rispetto agli altri OS più famosi; qui tutte le applicazioni sono caratterizzate da almeno un'activity che vengono gestite con uno stack. Le activity hanno diversi metodi che vengono chiamati durante tutto il ciclo di vita dell'applicazione. Quando una nuova activity parte, viene piazzata in cima allo stack e viene eseguita; le activity nello stack sotto la prima non saranno visibili in primo piano fino a quando la nuova attività non verrà eseguita.

Se un'activity perde il focus ma è ancora visibile, come due finestre con dimensione ridotta ma sovrapposte tra loro, essa verrà messa in pausa;

in questo caso, lo stato e le informazioni dell'applicazione rimangono in memoria e la finestra di quell'activity continua a rimanere visibile ma il suo processo può essere fermato nel caso in cui la memoria si esaurisca.

Se l'activity, invece, viene completamente oscurata da un'altra activity, la prima viene fermata; come nel caso precedente, rimangono lo stato e le informazioni relative ad essa ma la finestra non sarà più visibile sul display e molto probabilmente il processo verrà arrestato per rilasciare memoria. Le activity in stato di pausa o fermo possono essere riaperte e tornare in esecuzione dallo stato precedente.

1.3.4 Altri sistemi operativi

I sistemi operativi precedemente descritti sono i più famosi ma non sono gli unici; tanti altri sistemi operativi, non meno importanti dei primi, sono stati progettati durante gli anni.

Tra di essi possiamo citare Blackberry OS, Firefox OS, Ubuntu Touch, webOS.

1.3.4.1 Blackberry OS

Blackberry OS è un sistema operativo mobile proprietario, sviluppato da RIM (Research in Motion) per la sua linea di smartphone Blackberry.

Il sistema operativo fornisce multitasking e supporta dispositivi di input specializzati che sono adottati da RIM, come la trackball, il trackpad e il touchscreen. Nonostante la buona fetta di mercato iniziale nel mondo consumer, l'OS è orientato più per l'uso enterprise.

1.3.4.2 Ubuntu Touch

Ubuntu Touch è sistema operativo, presentato nel 2013 da Canonical Ltd, che riprende la versione di Ubuntu Desktop con kernel linux e ottimizzata e adattata agli smartphone. Il sistema è progettato in modo da non fare completo affidamento alla macchina virtuale Java. Dispone 40 lingue e sono preinstallate 12 applicazioni, di social network e utility e persino un terminale, presente in tutte le distribuzioni linux-based.

Le applicazioni appaiono sulla barra sinistra dello schermo, permettendo di avere sempre a disposizione le più usate. Per vedere quali sono le applicazioni aperte è

sufficiente uno swipe da sinistra a destra dello schermo; uno swipe in senso contrario serve a far riaprire l'ultima applicazione usata. Per far apparire i pulsanti di controllo è sufficiente scorrere il dito dal bordo inferiore.

1.3.4.3 Firefox OS

Firefox OS è un sistema operativo open source basato su kernel linux e in sviluppo da Mozilla. L'idea alla base dell'OS è quella di avere un sistema completo guidato da una comunità e disponibile come un'applicazione web usando gli standard web, tecniche avanzate di HTML5 e le open web API del dispositivo per accedere direttamente all'hardware dello stesso con Javascript.

Il sistema operativo in questione usa Gaia come interfaccia grafica; essa controlla tutto ciò che viene rappresentato dallo schermo, include le implementazioni per la schermata di blocco, la scrivania, un dialer, le applicazioni di messaging e fotocamera.

1.4 Sensori degli smartphone

Una caratteristica veramente importante degli smartphone è la presenza dei sensori. Il sensore è un trasduttore che si trova in diretta interazione con il sistema misurato ed è riferito solamente al componente che fisicamente effettua la trasformazione della grandezza d'ingresso in un segnale di un altra natura, nel nostro caso di natura elettrica.

Come vedremo nei seguenti paragrafi, i sensori incrementano l'usabilità dello smartphone dal punto di vista dell'utente, incrementandone le funzionalità e l'autonomia.

1.4.1 Accelerometro

L'accelerometro è un componente, fatto di silicio e costituito da un minuscolo involucro con all'interno una massa. Quest'ultima ha una certa flessibilità che consente di valutarne gli spostamenti senza dover utilizzare una molla, come succede negli accelerometri di grandi dimensioni. La massa è costituita da piccole lamelle mobili che si muovono tra una serie di lamelle fisse.

Poichè le lamelle mobili non possono toccare le lamelle fisse, si è pensato di misurare la differenza di potenziale tra la lamella mobile e le due lamelle fisse; la lamella mobile e quella fissa fungono quindi da armature del condensatore. La differenza di potenziale che si ha tra la lamella mobile e la lamella fissa varia in funzione della distanza tra le due lamelle.

L'accelerometro è il sensore più utile per lo smartphone; Apple fu la prima a scroprirne i vantaggi, usando questo sensore per orientare l'immagine dello schermo in base all'orientazione del dispositivo.

1.4.2 Giroscopio

Lo scopo principale del giroscopio è quello di mantenere l'orientamento grazie alla conservazione del momento angolare. Così come un oggetto tende a conservare il suo stato di moto rettilineo uniforme, un oggetto ruotante tenderà a mantenere il suo moto rotatorio nella stessa direzione. Grazie a questa proprietà, il giroscopio è in grado di rilevare la rotazione relativa dell'oggetto cui è solidale, descrivendone la variazione nel corso del tempo.

Il giroscopio di uno smartphone è capace pertanto di rilevare la rotazione sutti e tre gli assi dello spazio.

1.4.3 Magnetometro

Il magnetometro è lo strumento di misura del campo magnetico e simula, nello smartphone, una bussola. Le applicazioni più interessanti che usano il magnetometro sono quelle per navigazione satellitare in tempo reale.

1.4.4 Sensore di prossimità

I sensori di prossimità sono dei sensori in grado di rilevare la presenza di oggetti nelle immediate vicinanze del lato sensibile del sensore stesso, senza che vi sia un effettivo contatto fisico. La distanza entro cui questi sensori rilevano oggetti dipende esclusivamente dalla tipologia del sensore e dalla loro qualità progettuale.

Negli smartphone vengono usati i sensori di prossimità ottici; la loro tecnologia si basa sulla rilevazione della riflessione di fasci di raggi infrarossi opportunamente

scelti per non avere interferenze con la luce ambientale. Il fascio viene riflesso dalla superficie stessa dell'oggetto rilevato, per lo stesso fenomeno per cui la luce visibile può essere riflessa e percepita dai nostri occhi.

L'utilità di questo sensore sta nel fatto di disattivare lo schermo dello smartphone quando esso si avvicina all'orecchio durante una conversazione oppure quando viene messo in tasca; in questo modo non si verificheranno tocchi indesiderati sullo schermo.

1.4.5 Sensore di luminosità

Il sensore di luminosità ambientale ha il compito di preservare la durata della batteria variando il livello di retroilluminazione del display LCD in funzione della luce che viene a contatto con il dispositivo.

1.4.6 Termometro

Il termometro negli smartphone è un sensore che rileva la temperatura.

1.4.7 Barometro

Il barometro è un sensore che misura la pressione atmosferica. Nello smartphone il barometro serve a migliorare il fix GPS.

1.4.8 Pedometro

Il pedometro non rientra nella categoria dei sensori ma è un apparato utilizzato per misurare il numero di passi e pertanto può fornire una misura indiretta della distanza, velocità e il passo di una persona. Al suo interno vi è un sensore capace di registrare il movimento corporeo fatto ad ogni passo.

1.4.9 Sensore di impronte

Apple introduce per la prima volta in uno smartphone il primo sistema di riconoscimento biometrico, quale il sensore d'impronte, rivoluzionando quindi il modo di vedere la sicurezza negli smartphone.

Il sensore d'impronte permette, con il semplice appoggiare di un dito sul tasto centrale del dispositivo, l'autenticazione di una persona.

1.5 Sommario

In questo capitolo abbiamo visto come le necessità dell'uomo abbiano portato all'invenzione del telefono, del PDA e successivamente alla progettazione dello smartphone e abbiamo analizzato il come e il perchè lo smartphone si è diffuso così in ritardo rispetto alla data di uscita del suo primo prodotto in genere e il motivo per cui, nonostante il boom di vendita si avvenuto quasi 10 anni fa, i volumi di vendita di tali dispositivi sono ancora crescenti.

Inoltre si è analizzata l'enorme varietà dei sistemi operativi relativi agli smartphone, conoscendone i vari aspetti in termini di usabilità, prestazioni e funzionalità, cercando di portar alla luce le differenze tra di essi. Malgrado già grosse compagnie si siano già affermate nel marketshare, continuano a spuntano fuori nuovi sistemi operativi che, malgrado siano open source, tendono ad implementare funzionalità mai viste fin'ora.

Infine si son studiati i principali sensori montati in un uno smartphone cercando di portar a conoscenza il modo con cui essi riescano a rivoluzionare il dispositivo allargandone le prestazioni e funzionalità.

2 Ambienti di sviluppo e analisi

In questo capitolo verranno descritti gli strumenti usati per la programmazione di applicazioni per sistemi operativi mobili Windows Phone e Android. In seguito ci concentreremo sull'impiego delle macchine virtuali necessarie per emulare i dispositivi con sistemi operativi mobili e testare quindi le applicazioni per smartphone direttamente sul pc. Infine verranno descritti gli strumenti usati per l'analisi numerica dei dati raccolti utilizzando gli smartphone.

2.1 Sviluppare applicazioni per Windows Phone

Per lo sviluppo di applicazioni in Windows Phone si è vincolati all'uso di un personal computer (PC) con Windows 8 Professional (o più recente) e dell'ambiente di sviluppo (IDE) Visual Studio 2013.

Per quanto riguarda il debug dell'applicazione si può procedere in due modi:

- L'esecuzione dell'applicazione sulla macchina virtuale Windows Phone Emulator.
- L'esecuzione dell'applicazione previa installazione sullo smartphone.

Come si vedrà in seguito, la macchina virtuale ci sarà molto utile per la risoluzione di errori di codice di una certa entità e per verificare l'effettiva correttezza ed estetica della UI (User Interface o interfaccia grafica).

2.1.1 Sistema Operativo

Windows 8 è un sistema operativo appartenente alla serie Microsoft Windows, uscita nel quarto quadrimestre del 2012 (Q4). Questo OS è disponibile in quattro edizioni:

- Windows RT, per il supporto all'architettura di processori ARM
- Windows 8
- Windows 8 Pro
- Windows 8 Enterprise

Nel 2013, venne poi rilasciato l'aggiornamento da 8 a 8.1.

2.1.1.1 Interfaccia Utente

Windows 8 si differenzia dai predecessori con una nuova interfaccia utente, simile a quella di Windows Phone, progettata per adattarsi meglio all'input da touch screen e, nello stesso tempo, rimanere compatibile con l'uso di mouse e tastiera. Questa interfaccia e l'approccio a riquadri che contengono i link alle applicazioni si chiama *Modern UI*; i riquadri corrispondono a mattonelle (*tiles* in inglese) .

2.1.1.2 Shell

La barra delle applicazioni di Windows 8 è priva del pulsante di Start. Viene incluso un supporto migliorato per configurazioni a più schermi. La barra delle applicazioni infatti può essere mostrata su più schermi, oppure ogni schermo può mostrare la propria barra delle applicazioni dedicata. Gli sfondi possono essere estesi su più schermi oppure ogni schermo può avere il proprio sfondo separato.

Le immagini disco ISO e IMG e i file VHD possono essere montati come unità virtuali attraverso un doppio click, a differenza dei vecchi sistemi operativi Microsoft.

2.1.1.3 Hyper-v

Tra le nuove funzionalità implementate in Windows 8 spicca quella che ci ha vincolato al suo utilizzo, l'*Hyper-v,* precedentemente offerto solo in Windows Server.

Hyper-V permette di eseguire test, ottenere retro compatibilità e simulazioni di macchine e ambienti virtuali, garantisce un'ampia scalabilità per le macchine virtuali e il supporto per macchine virtuali x86/x64, processori virtuali multi-core, gestione dinamica della memoria, switch virtuali e snapshots.

Grazie a questa tecnologia quindi, è possibile eseguire la macchina Windows Phone Emulator tramite l'esecuzione del comando debug di Visual Studio 2013 relativo al progetto di tesi.

2.1.2 Visual Studio 2013

Visual Studio è un ambiente di sviluppo integrato sviluppato da Microsoft che supporta diversi tipi di linguaggio quali C, C++, C#, F#, Visual Basic.Net e ASP .Net che permette la realizzazione di applicazioni, siti web, applicazioni web e servizi web. Inoltre Visual Studio è multi-piattaforma e con esso è possibile realizzare programmi per server, workstation, pocket PC, smartphone e per browsers.

Visual Studio integra la tecnologia IntelliSense che permette di correggere eventuali errori sintattici senza compilare l'applicazione, possiede un debugger interno per il rilevamento e la correzione degli errori logici nel codice in runtime e fornisce diversi strumenti per l'analisi prestazionale.

Per quanto riguarda il compilatore, .Net Framework converte il codice sorgente in codice IL (Intermediate Language o linguaggio intermedio tradotto in italiano). IL è un nuovo linguaggio progettato per essere convertito in modo efficiente in codice macchina nativo su differenti tipi di dispositivi. E' un linguaggio di livello più basso rispetto a Visual Basic .NET o C#, ma è a un livello di astrazione più alto rispetto al linguaggio assembly o ai linguaggi macchina.

Visual Studio 2013 è l'ultimo IDE sviluppato da Microsoft in concomitanza con l'uscita di Windows 8 ed è disponibile nelle seguenti versioni:

- Visual Studio 2013 Ultimate
- Visual Studio 2013 Premium
- Visual Studio 2013 Professional

Tra gli aggiornamenti più importanti si può notare: un accesso più rapido alle informazioni di cui si ha bisogno in un determinato contesto; un debug cronologico; test per verificare la scalabilità e le prestazioni in fase di produzione; una visualizzazione della struttura di un'applicazione con diagrammi UML; una gestione dell'architettura e delle dipendenze tra componenti e strumenti per comprendere le relazioni nel codice esistente.

2.1.2.1 Progetto in Visual Studio 2013

Per poter sviluppare un'applicazione per Windows Phone occorre creare un nuovo progetto in Visual Studio 2013. Per fare questo, una volta aperto il programma, si dovrà cliccare in alto a sinistra su *File*, passare il mouse sopra *Nuovo* e infine cliccare su *Progetto*. Verrà poi mostrata una finestra di *Nuovo Progetto* dove è possibile scegliere il tipo di progetto da creare. Sulla sinistra, dove sono presenti tutti i modelli installati, si dovrà abbassare la gerarchia chiamata *Visual C#* e cercarvi la voce *Applicazioni Windows Store.* A questo punto, al centro finestra, cliccare su *Applicazione vuota (applicazioni universali)*, scrivere in basso il nome dell'applicazione sviluppata e confermare il tutto cliccando sul pulsante di ok in basso a destra della finestra.

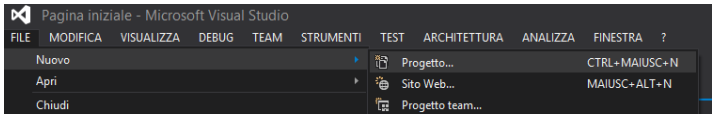


Figura 2.1: Procedimento per creare un nuovo progetto in Visual Studio 2013

Selezionato il modello, Visual Studio mostrerà il layout adatto allo sviluppo di applicazioni per smartphone. Nel caso in cui si voglia sviluppare l'interfaccia grafica dell'applicazione, si aprirà il file con estensione ".xaml". Quando questo tipo di progetto viene aperto, la finestra principale viene divisa in due mostrando sulla sinistra un'anteprima della schermata dell'applicazione e sulla destra la rappresentazione in codice simile all'xml della stessa.

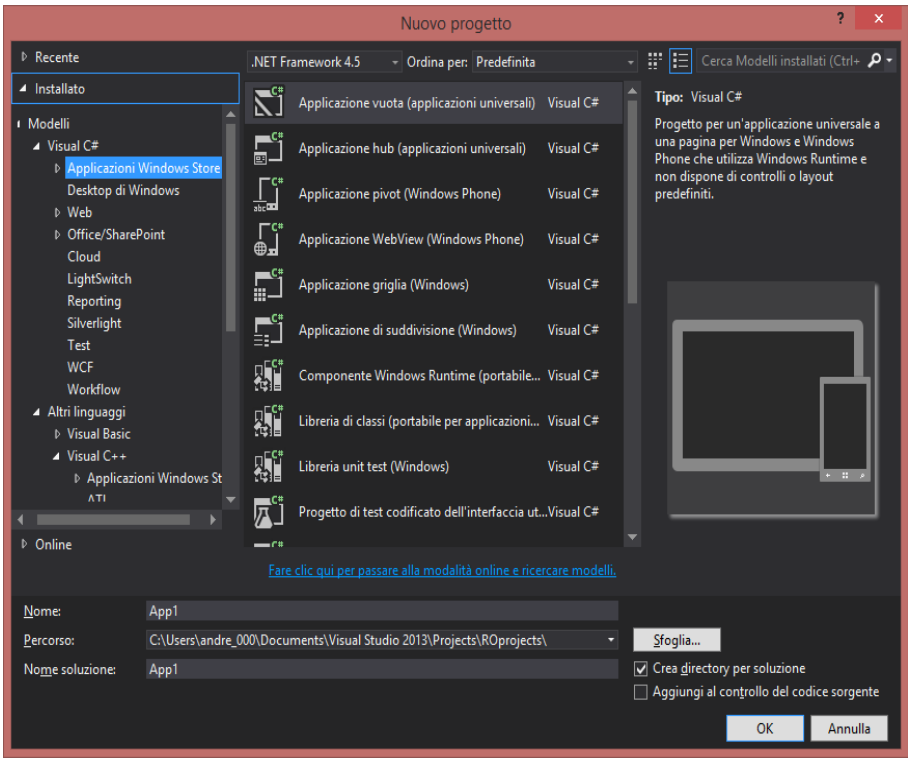


Figura 2.2: Finestra che permette di scegliere il tipo di progetto da creare

Per poter scrivere la parte del codice funzionale relativa alla pagina xaml, occorre cliccare sulla freccia al lato sinistro del nome della nostra pagina per poi cliccare sul file omonimo con estensione ".cs". A questo punto verrà visualizzata la finestra classica relativa al codice c#.

Una volta sviluppata l'applicazione, è possibile compilarla premendo sulla tastiera il tasto "F6" corrispondente al tasto di scelta rapida per la compilazione compilazione del codice. Se si volesse testare l'applicazione sulla macchina virtuale "Windows Phone Emulator" sarà necessario cliccare sul simbolo di esecuzione (triangolino verde nella barra degli strumenti) accertandosi che affianco al simbolo ci sia scritto *Emulator WVGA 512MB*. Nel caso in cui si voglia far eseguire l'applicazione sul proprio dispositivo Windows Phone, occorre collegarlo al PC tramite l'apposito cavo usb, assicurarsi che il dispositivo non sia in blocco (schermata nera o schermata di blocco) e mandare in esecuzione l'applicazione assicurandosi che affianco al triangolino verde ci sia scritto *Device*.

2.1.3 Windows Phone Emulator

Windows Phone Emulaor è un'applicazione deskop che emula un dispositivo Windows Phone. È provvisto di un ambiente virtualizzato dove è possibile eseguire il debug e il test dell'applicazione Windows Phone senza aver bisoggno del dispositivo fisico e di un ambiente isolato per le applicazioni prototipo.

Figura 2.3: Schermata di Home di Windows Phone Emulator fonte [11]

Windows Phone Emulator è progettato per fornire prestazioni paragonabili ad un dispositivo reale. È preferibile, tuttavia, testare l'applicazione su un dispositivo vero prima di pubblicare l'applicazione sullo store di Microsoft.

È possibile eseguire il testing dell'applicazione sull'immagine unica dell'emulatore per ciascuna delle versioni del sistema operativo e risoluzioni dello schermo supportate da Windows Phone. L'immagine standard è Emulator WVGA 512 MB che emula un telefono Windows Phone con memoria limitata. Lavorando con questo emulatore si è sicuri che l'applicazione sia compatibile con quasi tutti i dispositivi Windows Phone sul mercato. Affiché WPE sia disponibile, occorre installarlo attraverso il pacchetto "Windows Phone SDK" scaricabile dal sito Microsoft.

Windows Phone Emulator permette di emulare determinate funzioni e servizi dello smartphone:

- Il cambiamento dalla modalità portrait a landscape (orientamento dello schermo).
- Il supporto alla rete.
- Il ciclo di vita dell'applicazione, compreso di "tombstoning".
- Sensore della fotocamera.
- Simulazione GPS.

- Simulazione Accelerometri.
- Multi-touch.
- Microfono.
- Sensore di prossimità, scaricando un pacchetto dedicato.
- Schermata di blocco.
- Notifiche.

2.1.4 Licenza developer per app Windows Phone

Per testare un'applicazione sul dispositivo Windows Phone, occorre ottenere innanzitutto una licenza per sviluppatori per ogni computer in cui si vuole eseguire l'applicazione. Le licenze per sviluppatori sono gratuite e le si possono ottenere senza limiti; le licenze scadono dopo 30 giorni dopo la richiesta dopodiché vanno rinnovate.

Quando si esegue il debug o si avvia un'applicazione per la prima volta in un computer remoto o su un dispositivo direttamente connesso al computer di sviluppo, viene richiesto all'utente di scaricare una licenza per sviluppatori per il computer o per il dispositivo in questione. Ottenuta la licenza per sviluppatori è possibile installare e usare pacchetti non certificati ed eseguirne eventualmente il debug. L'ottenimento e il rinnovo della licenza per sviluppatori si può ottenere tramite Visual Studio.Sviluppare applicazioni Android

2.2 Sviluppare applicazioni

Per poter sviluppare sono richieste una buona conoscenza del linguaggio di programmazione Java e una conoscenza del linguaggio XML. Come IDE può essere utilizzato Eclipse con Android Development Tool (ADT), rimanendo però vincolati ai sistemi operativi con cui Eclipse è compatibile (come Windows, Linux e Mac OS). Negli inizi di dicembre 2014, con l'uscita ufficiale di Android Studio, è stata necessaria la migrazione da Eclipse a questo nuovo IDE.

2.2.1 Eclipse

Eclipse è un ambiente di sviluppo integrato multi-linguaggio e multipiattaforma ideato dalla *Eclipse Foundation.* E' un software libero distribuito sotto i termini della Eclipse Public License.

L'IDE può essere utilizzato per la produzione di software di vario genere; sono presenti infatti ambienti di sviluppo per il linguaggio Java e per il linguaggio C++ e dispone di plug-in che permettono di gestire XML, Javascript, HP e pacchetti che permettono di progettare graficamente una GUI per un'applicazione Java (Window Builder), rendendo Eclipse un ambiente RAD (Rapid Application Development).

Il software di sviluppo è scritto in Java e il toolkit grafico di Sun Microsystem si appoggia a SWT (Standard Widget Toolkit), librerie di nuova concezione che conferiscono ad Eclipse un'elevata reattività.

La piattaforma di sviluppo è incentrata sull'uso di plug-in e di componenti software ideate per uno specifico scopo, come la generazione di diagrammi UML. Nella versione base è possibile programmare in Java usufruendo di comode funzioni di aiuto quali il completamento automatico, suggerimento dei tipi di parametri dei metodi, possibilità di accesso diretto a CVS e riscrittura automatica del codice (Refactoring) in caso di cambiamenti nelle classi.

Il progetto “Eclipse Web Tools Platform” estende la piattaforma Eclipse con dei tool dedicati allo sviluppo di applicazioni Web e Java EE. Esso include editor per diffenti linguaggi di programmazione, wizard e applicazioni built-in per semplificare lo sviluppo, tool e API per il supporto alla fase di deployment, esecuzione e testing di applicazioni.

2.2.2 Android studio

Android studio è un IDE per lo sviluppo di applicazioni Android. Come per Eclipse, Android Studio è un software libero sotto licenza di Apache License 2.0. Esso è basato su IntelliJ Idea, un software sviluppato da JetBrains, progettato specificatamente per lo sviluppo su Android. L'ambiente di sviluppo è disponibile per Windows, Mac OS X, Linux.

Tra le funzionalità, si distinguono le seguenti:

- Live Layout. WYSIWYG Editor- Live Coding- Real-time App Rendering.
- Console di sviluppo: suggerimenti di ottimizzazione, assistenza per la traduzione, tracking referenziale.
- Supporto build basato su Gradle.
- Refactoring specifici per Android e fix veloci.
- Strumenti Lint per monitorare le performance, usabilità, compatibilità delle versioni e diversi altri problemi.
- ProGuard per l'ottimizzazione del codice e app-signing per la registrazione e la possibile distribuzione dell'applicazione
- Set-up degli ambienti basati su template per creare determinati componenti di design
- Un editor di layout che permettete all'utente di spostare le componenti UI, opzioni di anteprima per configurazioni a schermi multipli.

2.2.2.1 Configurazione di Android Studio

Prima di scaricare Android Studio è consigliabile, se non già presente nel pc, scaricare ed installare l'ultimo pacchetto JDK dal sito Oracle. Ora è possibile entrare nella pagina di download di Android Studio e cliccare sul pulsante *Download Android Studio*. Una volta installato il pacchetto Android studio sul sistema, è possibile avviare l'ambiente di sviluppo. Al primo avvio, l'IDE richiede se si è interessati a reimpostare le preferenze personali di una precedente installazione del programma. Cliccare sul Radio Button "*I do not have a previous version of Android Studio....*". Nella finestra di benvenuto di Android Studio si dovrà far click su *Configure* e successivamente cliccare infine sul pulsante *SDK* Manager.

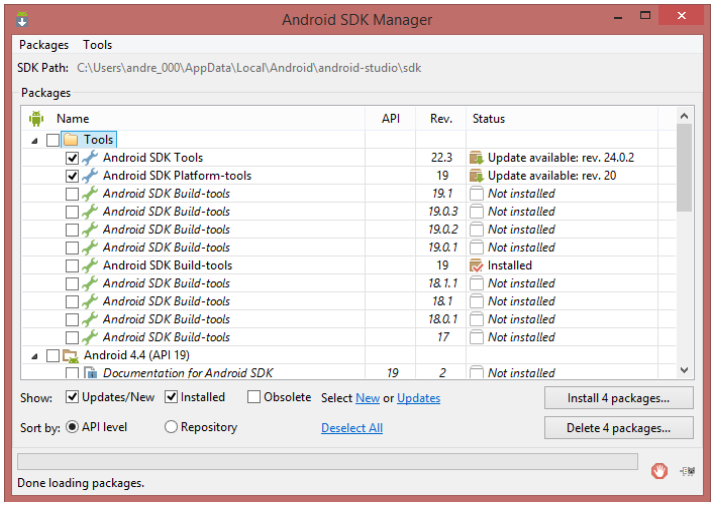


Figura 2.4: Finestra di SDK Manager

L'SDK Manager mostra tutti i componenti utilizzabili per realizzare le proprie applicazioni per la piattaforma Android. Qui si possono spuntare le caselle corrispondenti alle versioni di Android per le quali si è previsto lo sviluppo. Ogni versione di Android è contraddistinta da un numero progressivo di versione della corrispondente API. In questa finestra ci si dovrà accertare di aver spuntato le seguenti caselle:

- Android SDK Tools
- Android Plaform-Tools
- Android SDK Build-Tools

Infine sarà necessario attivare le caselle seguenti contenute nella sezione *Extras*:

- Android Support Repository
- Android Support Library
- Google USB Driver

2.2.2.2 Progetto in Android Studio

Dopo aver opportunamento configurato l'ambiente di sviluppo di Android Studio, sarà possibile creare un nuovo progetto cliccando su *New Project*. L'ambiente di sviluppo richiederà di specificare: un nome per l'app Android che si sta per sviluppare; il dominio sul quale si compiono le proprie attività; un nome da assegnare al pacchetto. Il nome del pacchetto dev'essere obbligatoriamente univoco, non usabile quindi da nessun'altra applicazione, anche di altri

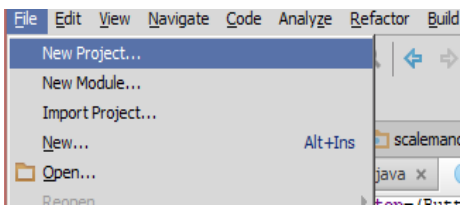
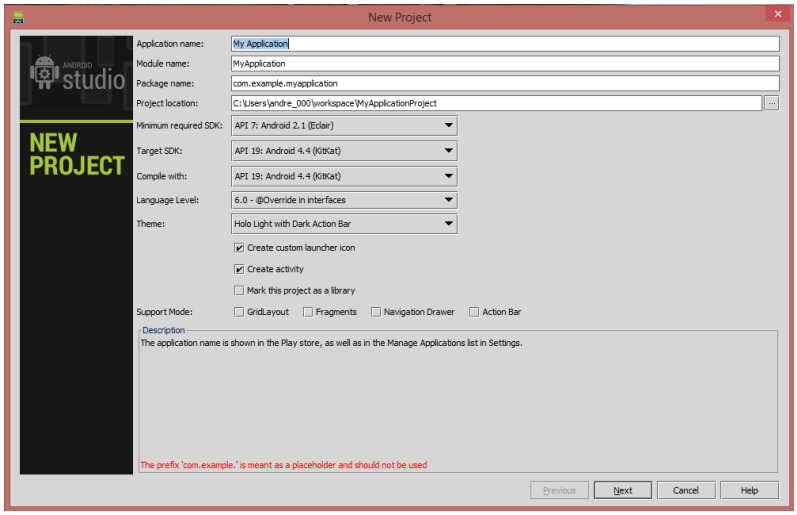


Figura 2.5: Procedura per la creazione di un nuovo progetto in Android Studio

produttori. Il percorso riportato nella casella *Project Location* indica la cartella nella quale verranno memorizzati i file dell'applicazione Android.



2.6: Compilazione dei dati per la creazione del nuovo progetto

Cliccando su *Next*, Android Studio permetterà di scegliere su quali dispositivi dovrà funzionare l'applicazione e, nello specifico, la versione di Android da considerare come requisito.

Android Studio permette di realizzare applicazioni installabili anche o esclusivamente su dispositivi indossabili come gli smartwatch. E' interessante notare che l'IDE visualizza la percentuale di dispositivi coi quali l'applicazione sarà compatibile a seconda dell'API scelta.

Cliccando ancora su *Next*, Android Studio consentirà di creare una nuova attività (*activity*) scegliendo fra diversi template. La procedura guidata si conclude facendo scegliere all'utente il nome della sua nuova attività. Cliccare su *Finish* per iniziare a sviluppare. Android Studio visualizzerà il messaggio *Building Gradle project info* e, in basso, *Scanning files to index*. A questo punto bisognerà attendere la conclusione di tutte le operazioni. L'ambiente di sviluppo verrà visualizzato in tre parti principali:

- A sinistra sarà presente l'elenco gerarchico dei file relativi all'applicazione.
- Al centro il nostro codice.

- A destra l'anteprima dell'activity.

La regione destra del nostro layout sarà visibile solo se si aprono i file xml relativi alle activity.

2.2.3 Android Virtual Device

Per testare l'applicazione sviluppata è possibile usare uno o più dispositivi virtuali AVD (Android Virtual Devices). Ciascun AVD viene avviato da parte di un emulatore che ricalca da vicino tutte le principali caratteristiche di un dispositivo mobile Android. È possibile create tanti AVD quanti ne risultino necessari; infatti ciascun dispositivo virtuale può essere sfruttato per provare le applicazioni in corso di sviluppo utilizzando varie configurazioni hardware e software.

Figura 2.7: schermata di home di Android Virtual Device fonte [12]

Android Studio consente di definire tutte le caratteristiche tecniche del dispositivo virtuale in corso di creazione, dalla tipologia di display alla versione di Android utilizzata, passando per la presenza o meno di una tastiera "fisica", di una fotocamera frontale o posteriore, dalla memoria a disposizione e della capacità della scheda SD.

Per creare un nuovo AVD bisogna cliccare sul pulsante *AVD Manager* caratterizzato dall'immagine dello smartphone con la testa del logo di Android situato sulla barra degli strumenti dell'applicazione. Fare click sul pulsante *Create* quindi scegliere le caratteristiche del dispositivo virtuale. A questo punto, l'emulatore è pronto.

Con un click sul pulsante di Run ci si troverà di fronte alla schermata *Choose Device*; in questa schermata è possibile scegliere il dispositivo virtuale che si desidera utilizzare per eseguire l'applicazione Android. Nel caso in cui l'AVD sia già attivo sarà sufficiente mettere la spunta in *Choose a running device* per poi cliccare sul nostro AVD, altrimenti si dovrà selezionare l'AVD da attivare e premere quindi *Ok*.

Dopo il caricamento del sistema operativo nel dispositivo virtuale, l'applicazione di test appena creata in Android Studio verrà automaticamente eseguita. Il dispositivo virtuale si comporterà esattamente con un normale device Android; nello specifico, nella colonna di destra, si troveranno i pulsanti hardware che tipicamente sono presenti in ogni smartphone Android. Saranno presenti il pulsante *Home* per tornare alla schermata principale, il pulsante *Menu* e di *Return* oltre ai controlli per aumentare e diminuire il volume e tenendo premuto il pulsante di Power Off, si potrà richiedere lo spegnimento del dispositivo virtuale.

2.3 Applicazioni per il calcolo numerico

Lo scopo principale del progetto di tesi qui descritto riguarda la realizzazione di algoritmi che lavorano sui dati di tipo numerico ottenuti dai sensori degli smartphone. Eseguire dei test parziali sugli algoritmi direttamente sullo smartphone può risultare particolarmente oneroso in termini di tempo per lo sviluppatore, come è oneroso lo sviluppo di applicazioni ad hoc per monitorare i dati con linguaggi di sviluppo come c# o Java.

È stato quindi importante trovare dei programmi che aiutino ad analizzare più velocemente i dati salvati, a scrivere gli algoritmi e a testarli direttamente, monitorando i risultati. In questo modo, ingegnarsi nel trovare algoritmi adatti alle nostre esigenze è risultato molto più semplice ed efficacie. Trovati gli algoritmi, si è poi passati all'implementazione ed esecuzione nei dispositivi mobili per valutarli nel contesto di nostro interesse.

2.3.1 Scilab

Scilab [13] è un pacchetto di programmi gratuiti per la computazione numerica, sviluppato da Scilab Enterprises. Utilizza la licenza CeCILL [14] v2 che è compatibile con la licenza GNU GPL v2 [15] della Free Software Foundation.

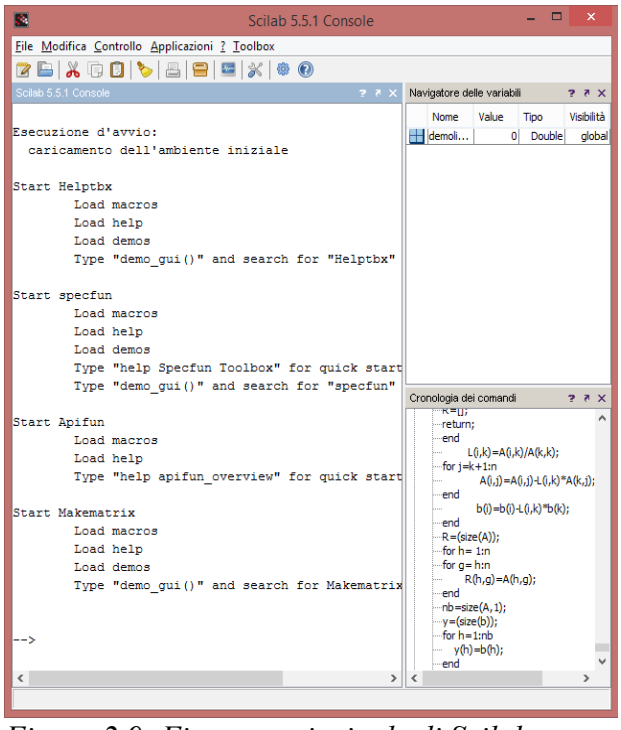


Figura 2.9: Finestra principale di Scilab

Scilab è stato impiegato in alcuni progetti industriali e di ricerca ed è stato possibile migliorarlo grazie a dei contributi, molti dei quali fatti dagli utenti. La sintassi è simile a quella di MATLAB [16] ma i due programmi, i loro applicativi ed i plug-in non sono completamente

compatibili (anche se esiste un convertitore nel pacchetto di Scilab che opera le conversioni da MATLAB).

Il programma adotta un linguaggio di programmazione di alto livello, basato sul linguaggio di programmazione C e consente di elaborare un'ampia gamma di operazioni matematiche: da operazioni relativamente semplici come le moltiplicazioni a quelle di livello più alto, come correlazioni ed aritmetica dei numeri complessi. Il software risulta quindi adatto per l'elaborazione dei segnali e analisi statistica.

Il Programma è compatibile con quasi tutti i sistemi operativi in circolazione e richiede Java 6 o più recenti per il suo funzionamento; i requisiti minimi in termini di Hardware sono bassi, consentendone l'uso su tutti i pc non troppo datati.

A differenza di Matlab, è possibile scaricare gratuitamente Scilab dal suo sito ufficiale, motivo per cui si è scelto questo programma per lo sviluppo del progetto di tesi.

2.4 Sommario

In questo capitolo abbiamo visto in che modo vanno impostati gli ambienti di sviluppo sia che si voglia sviluppare un'applicazione per Windows Phone, sia che si voglia sviluppare un'applicazione per Android. Per entrambi i casi vengono messi alla luce i requisiti affinché gli ambienti di sviluppo possano essere installati correttamente e viene fornita una linea generale di come vadano installati e impostati.

Si è inoltre parlato di macchine virtuali ì, sia per Android che per Windows Phone e anche per essi vengono spiegati i requisiti hardware e software, come vanno installati e come vanno configurati. Nel caso di Windows Phone si parla anche del sistema operativo Windows 8, per via delle sue importanti funzionalità che permettono di far girare il suo relativo emulatore.

Per finire si è parlato di applicazioni per il calcolo numerico; in particolare si descrive la loro funzione e il motivo per cui sono stati impiegati per la realizzazione del progetto di tesi. Per finire è stato descritto in linea generale cos'è Scilab e il perché del suo utilizzo.

3 Analisi dei dati

In questo capitolo verranno illustrate le procedure necessarie per catturare i segnali provenienti dai sensori degli smartphone sia per i sistemi Android che per quelli Windows Phone. Chiariremo con quale tipo di dati ci ritroveremo a lavorare e illustreremo i principali obiettivi di questo progetto di tesi.

Si studieranno due diversi metodi di elaborazione dei dati raccolti dai sensori per ottenere dei risultati coerenti al nostro obiettivo; per ognuno di essi verrà descritto il principio di funzionamento seguito da del codice esempio nel linguaggio di programmazione c# e dalla matematica di base del progetto.

Infine verranno mostrati dei test in cui si metteranno alla luce le qualità e i difetti dei rispettivi metodi usati nel progetto. Per ognuno di essi verranno ipotizzate possibili migliorie.

3.1 Raccolta dei dati

Il primo passo da compiere è quello di recuperare il segnale dai sensori degli smartphone. In questo progetto l'unico sensore interessato è l'accelerometro poiché verrà studiato il riconoscimento di alcuni pattern di movimento svolti dall'uomo registrati dallo smartphone. Successivamente analizzeremo come prelevare i dati dall'accelerometro e in quale forma ci verranno mostrati per capire come dovranno essere trattati durante lo sviluppo dell'applicazione.

Nell'ultima sezione verrà mostrato del codice c# e Java per la lettura dell'accelerometro sia per i dispositivi WP che per i dispositivi Android.

3.1.1 Cosa si ottiene dall'accelerometro

Come già descritto, l'accelerometro si basa sulla rilevazione dell'inerzia di una massa quando viene sottoposta ad una accelerazione; nel nostro caso la massa è rappresentata da delle minuscole lamelle mobili che si muovono tra una serie di lamelle fisse. In uno smartphone sono presenti tre sensori per l'accelerometro, ognuno per ogni coordinata, come mostrato in Figura 3.1; ogni sensore può restituire un segnale che rappresenta l'estremo di un vettore di accelerazione "subita" dal cellulare, dove il centro di massa del telefono rappresenta il punto di origine del vettore. Se per esempio teniamo lo smartphone in mano con il display rivolto esattamente verso l'alto, il sensore dell'accelerometro relativo all'asse Z restituirà un valore negativo corrispondente alla forza di gravità applicata al dispositivo. Muovendo la mano in avanti o indietro si può facilmente notare come il sensore dell'accelerometro relativo all'asse Y restituisca un valore di accelerazione che controbilancia l'accelerazione applicata alla mano per spostarsi. Interrogando l'accelerometro in un determinato istante si otterrà quindi un insieme di tre vettori che rappresenta l'attuale stato di accelerazione che controbilancia l'effettiva accelerazione applicata al dispositivo.

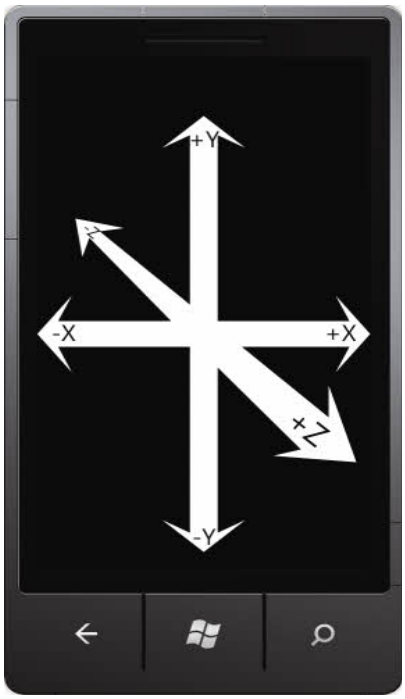


Figura 3.1 : Coordinate accelerometro, fonte [17]

3.1.2 Cosa analizzare e cosa si vuole ottenere

Interrogando l'accelerometro del dispositivo in più momenti equidistanti tra loro è possibile ottenere un insieme di tre elementi che rappresentano, in forma discreta, l'andamento dell'accelerazione percepita dal dispositivo nel tempo. Nella pratica, grazie a questo insieme di valori, si possono osservare le forze emanate dalla parte del nostro corpo che interagiva con il cellulare.

Tuttavia i soli dati grezzi non sono sufficienti a far capire né a noi né al nostro smartphone quale attività stava svolgendo chi lo stava utilizzando; serve infatti un algoritmo capace di prendere l'insieme di elementi popolato dall'accelerometro e trasformarlo in un qualcosa di più interpretabile. L'algoritmo dovrà essere in grado di valutare se una certa attività umana è stata svolta durante la lettura dell'accelerometro.

Inoltre, l'agortitmo deve essere progettato in modo tale da valutare in maniera univoca un determinato tipo di attività fisica. Per esempio, se si vuole progettare un algoritmo per rilevare i salti in alto stando fermi, l'algoritmo potrebbe valutare in maniera erronea quel tipo di salto mentre in realtà si sta semplicemente spostando il telefono dal basso verso l'altro per poi spostarlo nuovamente al suo punto di partenza. In questo caso si parla di "falso positivo".

Usando solamente un sensore del dispositivo i falsi positivi sono inevitabili. Buona parte dei rilevamenti che si potrebbero avere svolgendo determinate attività fisiche potrebbero essere simulati con semplicità con particolari movimenti della mano. Perciò possiamo dire che, per quanto riguarda gli algoritmi che usano solo l'accelerometro, l'affidabilità dipende da quanti falsi positivi vengono rilevati. Meno ne vengono rilevati, più l'algoritmo è affidabile.

Inoltre bisogna prendere in considerazione l'esistenza dei "falsi negativi", ovvero quando l'algoritmo fallisce nel rilevare determinate attività fisiche per cui sono stati progettati. Bisogna stare attenti quindi a non rendere gli algoritmi troppo restrittivi ad un determinato caso. Per esempio, se si volesse attaccare lo smartphone ad un guanto da boxe e si volesse rilevare quanti pugni vengono tirati dal pugile in un determinato momento, bisogna tener conto che ci son diversi tipi di pugni come il dritto, il rovescio o il gancio; scrivere un algoritmo che rileva solo uno di quei pugni può determinare solo la sua inaffidabilità. Bisogna anche considerare che un algoritmo può essere efficiente se provato sul progettista ma meno efficiente se provato con altri soggetti; infatti i "fattori di forma" del soggetto possono influenzare notevolmente sulla precisione dell'algoritmo.

3.1.3 Prelevare i dati dai sensori

Definite le caratteristiche che un algoritmo deve avere, occorre spiegare in che modo è possibile prelevare i dati dai sensori.

In linea generale, affinché la nostra applicazione possa usare l'accelerometro, bisogna effettuare una chiamata alle API native dei sensori; attraverso di esse la classe relativa ai sensori diventa "visibile" ed è possibile interagirci. Si potranno quindi recuperare tutte le informazioni relative all'accelerometro come la portata massima o il consumo in un determinato istante e impostarne di conseguenza le caratteristiche come la frequenza di recupero dei dati.

Di seguito verranno riportati i modi in cui è possibile recuperare i dati dall'accelerometro, sia per il caso di Android che per il caso di Windows Phone.

3.1.3.1 Prelevare i dati dall'accelerometro in Windows Phone

In primo luogo, aprire Visual Studio 2013 e creare un nuovo progetto scegliendo un'*Applicazione vuota* dai tipi di progetto *App di Visual c#/Windows Store*; aprire quindi il file "MainPage.xaml.cs" del progetto.

Al file vanno aggiunte due direttive alla lista già presente:

```
using Windows.UI.Core;
using WIndows.Devices.Sensors;
```

Queste due direttive sono richieste per il supporto del dispatcher e dell'accelerometro.

Aggiungere un'istanza globale privata della classe *Accelerometer*.

```
private Accelerometer accelerometer;
```

Aggiungere un gestore di eventi che viene chiamato ad ogni lettura dell'accelerometro:

```
        private void accelerometer_valueChanged(object sender,
SensorReadingEventArgs<AccelerometerReading> e)
    {
        Dispatcher.BeginInvoke(()=>
                    updateValues(e.SensorReading);
    }
```

Codice 3.1: Codice C# per l'ascolto dell'accelerometro

Il metodo `BeginInvoke` del dispatcher mostrato in Codice 3.1 esegue in maniera

asincrona un *delegate* al metodo `updateValues` passandogli come parametro l'oggetto d'istanza *AccelerometerReading*. Nel metodo `updateValues`, tramite il parametro passatogli dal precedente *delegate* è possibile prelevare la struttura *Vector3* che contiene i valori recuperati dall'accelerometro per l'asse x, y e z.

Nel costruttore della nostra *MainPage* vanno aggiunte le seguenti righe di codice:

```
            accelerometer=new Accelerometer();
            accelerometer.TimeBetweenUpdates=
                                        TimeSpan.FromMilliseconds(25);
            accelerometer.CurrentValueChanged+=new
EventHandler<SensorReadingEventArgs<AccelerometerReading>>(accelerometer_va
lueChanged);
            accelerometer.Start();
```

Codice 3.2: Codice c# per impostare l'ascolto dell'accelerometro

Nelle righe di codice illustrate in Codice 3.2 viene inizializzato l'oggetto *`accelerometer`* e viene impostata la frequenza di lettura a 25 millisecondi; successivamente viene registrato il gestore di eventi all'oggetto d'istanza *`Accelerometer`* specificando nei tipi (all'interno di < >) il sensore d'interesse; infine, tramite il metodo `accelerometer.Start()` viene fatta partire la lettura.

La lettura dell'accelerometro continuerà fino a quando non viene richiamato il metodo pubblico di stop dell'oggetto d'istanza `Accelerometer`.

3.1.3.2 Prelevare i dati dall'accelerometro in Android

Aprire Android Studio, creare un nuovo progetto e aprire la `MainActivity`. In Android l'activity deve svolgere il ruolo di listener per eventi dell'accelerometro o sensore in genere. Per far ciò occorre aggiungere del codice alla definizione di classe:

```
public class MainActivity extends Activity implements
SensorEventListener.
```

A questo punto vanno aggiunti i membri privati globali che contengono riferimenti al *`SensorManager`* e al *`Sensor`*, che in questo caso è l'accelerometro:

```
private SensorManager mSensorManager;
```

```
private Sensor mAccelerometer;
```

Va poi aggiunta alla classe un *Override* al metodo `onCreate` chiamato quando la *`MainActivity`* viene caricata. All'interno del metodo in *Override* inizializziamo i membri privati precedentemente definiti:

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
       mSensorManager=
       (SensorManager)getSystemService(SENSOR_SERVICE);
       mAccelerometer=
       mSensorManager.
       getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
}
```

Codice 3.3: Metodo in Java per la chiamata al SensorManager e l'inizializzazione all'accelerometro

```
@Override
protected void onResume()
{
       super.OnResume();
       mSensorManager.registerListener(this,
              mAccelerometer,
              SensorManager.SENSOR_DELAY_NORMAL);
}
@Override
protected void OnPause()
{
       super.OnPause();
              mSensorManager.unregisterListener(this);
}
```

Codice 3.4: Metodi in Java per l'interruzione e la ripresa dell'ascolto dell'accelerometro

Come mostrato in Codice 3.3, in *`mSensorManager`* viene dato un riferimento al gestore dei sensori, grazie ad esso possiamo recuperare i riferimenti dell'accelerometro

da dare a `mAccelerometer`.

In Codice 3.4 vengono mostrati altri due *Override* che si occupano di attivare e disattivare l'ascolto dell'accelerometro:

In questo modo l'ascolto del sensore viene attivato quando l'applicazione è in primo piano mentre viene disattivato quando l'applicazione viene sospesa da una seconda applicazione, da un'activity o viene chiusa.

Infine, per recuperare i dati dall'accelerometro, occorre scrivere l'ultimo metodo in *Override*:

```
@Override
public void onSensorChanged(SensorEvent event)
{
	//algoritmo voluto
}
```

Codice 3.5: Metodo chiamato ad ogni ascolto dell'accelerometro

Il parametro passato al metodo in Codice 3.5 contiene un vettore pubblico `values` composto da tre elementi, ognuno per ogni coordinata x, y, z.

3.2 Studio dei dati raccolti

Dopo aver spiegato come ricavare i dati dai sensori, occorre illustrare come vengono elaborati tali dati affinché lo smartphone valuti correttamente l'azione fisica che stiamo svolgendo.

Per il progetto di tesi si è puntato sul rilevamento di una scala e verranno proposti due metodi di elaborazione dei dati differenti tra loro in termini di principi di funzionamento:

- Metodo dei canestri
- Metodo della correlazione incrociata

3.2.1 Metodo dei canestri

Il metodo dei canestri è un metodo di elaborazione dei dati provenienti dall'accelerometro che permette di capire quando l'utente sta salendo una scala. Questo metodo da buoni risultati se l'utente esegue l'azione tenendo lo smartphone in mano.

3.2.1.1 Principio di funzionamento

Il metodo consiste nel partizionare l'intervallo dei valori ottenibili dal sensore dell'accelerometro, di ogni asse di coordinate, in dieci sottoinsiemi uguali tra loro in dimensione fungendo quindi da canestri. Ogni volta che viene letto un nuovo segnale dall'accelerometro viene dapprima valutato il suo valore e poi spedito nell'apposito canestro. Si supponga che il range dei valori ottenibili dall'accelerometro sia [-2 , 2], posizionando lo smartphone su un tavolo perfettamente livellato si potrà notare che tutti i valori recuperati dallo smartphone sono uguali a 0 per l'asse x e per l'asse y, pertanto tutti i valori letti finiranno nel sesto canestro rappresentante il sottoinsieme [0 , 0.4).

Tuttavia, raccogliere in un tempo indefinito i dati potrà solo portare all'inaffidabilità del metodo poiché uno stato molto precedente a quello attuale andrà ad influire sul risultato appena calcolato. Tra una lettura ed un'altra del sensore è presente una latenza di tempo dell'ordine di 10-100 ms (dipendente dalla frequenza di rilevamento) e per studiare il segnale in un intervallo di tempo non troppo breve è sufficiente togliere dal canestro gli elementi più vecchi adoperando una tecnica FIFO (First In First Out).

Per valutare il tipo di attività svolta è sufficiente contare quanti elementi sono presenti in ogni canestro per ogni asse. Si può dimostrare, in maniera pratica, che salendo più volte le scale con lo smartphone in mano si avranno più o meno gli stessi risultati.

3.2.1.2 Programmazione in Windows Phone

Premettendo che tutto il codice relativo alla gestione dell'accelerometro sia stato già scritto e impostato ad una frequenza di lettura di 25 millisecondi, si può iniziare definendo e inizializzando le seguenti variabili:

```
static int numberOfBuckets=10;
int[] xBuckets = new int[numberOfBuckets];
int[] yBuckets = new int[numberOfBuckets];
int[] zBuckets = new int[numberOfBuckets];
List<int> xIndexList = new List<int>();
List<int> yIndexList = new List<int>();
List<int> zIndexList = new List<int>();
```

Codice 3.6: Codice C# per la definizione e inizializzazione delle variabili

In Codice 3.7 l'intero statico serve a definire il numero di sottoinsiemi con cui definire il range di valori ottenibili dall'accelerometro. Si definiscono quindi tre array di interi, ognuno per ogni coordinata, che rappresentano il numero di elementi presenti per ogni canestro. Infine vengono definite delle liste che serviranno a decidere, per ogni elemento che entra, quale elemento dovrà essere rimosso dai canestri (ovvero quello con timestamp più meno recente).

Per capire in quale canestro un valore verrà inserito, ci si aiuta con il metodo in Codice 3.7:

```
private int retrieveBucketPosition(float value)
{
        if (value>= -2.1 && value < -1.6) return 0;
        if (value>= -1.6 && value < -1.2) return 1;
        if (value>= -1.2 && value < -0.8) return 2;
        if (value>= -0.8 && value < -0.4) return 3;
        if (value>= -0.4 && value < 0) return 4;
        if (value>= 0 && value < 0.4) return 5;
        if (value>= 0.4 && value < 0.8) return 6;
        if (value>= 0.8 && value < 1.2) return 7;
        if (value>= 1.2 && value < 1.6) return 8;
        return 9;
}
```

Codice 3.7: Codice C# per il recupero dell'indice di canestro interessato dal valore di accelerazione

Nel metodo mostrato in Codice 3.8, il parametro passato viene confrontato per ogni range di valori e si arresta quando trova il range a cui ne fa parte.

Per ogni lettura invece deve essere eseguito questo metodo:

```
private void updateBuckets(Vector3 acc)
{
            int xIndexB,yIndexB,zIndexB;
            xIndexB = retrieveBucketPosition(acc.X);
            yIndexB = retrieveBucketPosition(acc.Y);
            zIndexB = retrieveBucketPosition(acc.Z);
            xIndexList.Add(xIndexB);
            yIndexList.Add(yIndexB);
            zIndexList.Add(zIndexB);
            xBuckets[xIndexB]++;
            yBuckets[yIndexB]++;
            zBuckets[zIndexB]++;
            if (xIndexList.Count > 40)
            {
                  xBuckets[xIndexList[0]]--;
                  yBuckets[yIndexList[0]]--;
                  zBuckets[zIndexList[0]]--;
                  xIndexList.RemoveAt(0);
                  yIndexList.RemoveAt(0);
                  zIndexList.RemoveAt(0);
            }
}
```

Codice 3.8: Codice C# per l'aggiornamento dei canestri

Come è possibile notare in Codice 3.8, al metodo privato soprastante viene passato come parametro la struttura recuperata dalla lettura dell'accelerometro. Per ogni coordinata vengono recuperati i numeri dei canestri (indice dell'array *xBuckets*) in cui gli elementi dovranno finire, che verranno poi aggiunti alla lista degli indici. Nel caso in cui la lista superi i 40 elementi, utilizzando le liste degli indici e prendendo il valore in cima, è possibile capire quale elemento tra i canestri, per ogni coordinata, dovrà essere eliminato e verrà rimosso l'indice dalla relativa lista. Ai fini del metodo non è necessario inserire nel canestro il valore ottenuto dall'accelerometro; infatti, una volta riconosciuto il canestro a cui l'elemento deve far parte, questo viene solo incrementato di uno.

Per ogni lettura, superati i 40 elementi nella lista degli indici, deve essere controllato il numero di elementi presenti in ogni canestro per il vettore di asse z.

3.2.2 Metodo della cross correlazione

Il metodo della cross correlazione (o correlazione incrociata) è un metodo particolarmente più complesso rispetto al metodo dei canestri che si basa appunto sulla cross-correlazione studiata in *Teoria dei segnali*.

3.2.2.1 Cross-correlazione e linea di pensiero

In *Teoria dei segnali* la cross-correlazione [18] rappresenta la misura di similitudine di due segnali come funzione di uno spostamento o traslazione temporale applicata ad uno di essi.

Considerando due segnali a valori reali x e y che differiscono solamente per uno spostamento sull'asse t (del tempo), si può calcolare la cross-correlazione per mostrare di quanto y deve essere anticipato per renderlo identico ad x. In pratica, la formula anticipa il segnale y lungo l'asse t, calcolando l'integrale del prodotto per ogni possibile valore dello spostamento. Quando i due segnali coincidono il risultato $(x \star y)$,dove la stella indica il tipo di funzione, è massimizzato poiché, quando le forme d'onda sono allineate, esse contribuiscono solo positivamente al computo dell'area.

Per due segnali di energia finita x ed y, la correlazione incrociata è definita come:

$$R_{xy}(t) = (x \star y)(t) \overset{def}{=} \int_{-\infty}^{\infty} x^*(\tau) y(t+\tau) d\tau$$

in cui x^* denota il complesso coniugato di x.

Per due sequenze tempo-discreto, la correlazione incrociata è definita come:

$$R_{xy}(n) = (x \star y)(n) \overset{def}{=} \sum_{m=-\infty}^{\infty} x^*[m]\, y[n+m]$$

Prendendo spunto dalle nozioni appena apprese è possibile modificare la formula matematica della correlazione incrociata per due sequenze tempo-discrete in modo tale da calcolare la differenza dei due segnali per ogni valore di spostamento di uno dei due segnali. Se la differenza tra i due segnali tende ad annullarsi in un determinato istante di tempo, si può dedurre che questi siano uguali o simili.

Le due sequenze tempo-discrete sono la sequenza da studiare rappresentata dal sottoinsieme dell'insieme dei valori recuperati dall'accelerometro e la sequenza campione.

Si nota già in linea teorica un primo difetto; l'algoritmo per funzionare ha bisogno di un campione su cui calcolare le differenze. Occorre quindi "registrare" un campione di movimento svolto dall'essere umano prima di poter eseguire l'algoritmo sui valori ricevuti dall'accelerometro.

Inoltre la posizione dello smartphone può influire sull'affidabilità del metodo; una sequenza rilevata mentre si sale una rampa con il dispositivo in mano è differente dalla sequenza rilevata con il dispositivo in tasca o in un qualsiasi altro punto del corpo.

3.2.2.2 Elaborazione dei dati

In un contesto più pratico, poiché l'accelerometro restituisce un valore per ogni asse cartesiano, è ipotizzabile attuare il calcolo ad ogni sequenza per ogni asse cartesiano; come è possibile intuire, i costi computazionali sono notevoli. Occorre trovare un metodo per calcolare le differenze delle sequenze di dati per ogni coordinata cercando di ridurre al minimo i tempi computazionali.

Si può pensare di fondere i valori di accelerazione per le tre coordinate in un unico valore successivamente memorizzato; in questo modo si terrà in memoria un sola lista di valori e la differenza tra la sequenza catturata dall'accelerometro durante l'esecuzione dell'algoritmo con la sequenza campione viene calcolata una sola volta.

Per non modificare il range di valori ottenibili con la fusione dei dati, è possibile calcolare il nuovo valore come la media dei valori di ogni coordinata.

3.2.2.3 Campionamento

Il campionamento dei dati rappresenta la prima fase del nostro metodo poiché senza di essa non ci sarebbe una sequenza con cui studiare i dati raccolti dall'accelerometro. Questa fase consiste nel registrare una sequenza di dati che descrive l'andamento dell'accelerazione nel tempo catturata dal nostro dispositivo mentre si svolge una particolare attività fisica ciclica come il passo di camminata o come la spinta di corsa, nel nostro caso il salire un gradino di scala.

La difficoltà di questa fase consiste nel catturare una sequenza di dati, di poter visionare la sequenza registrata e sceglierne un sottoinsieme che rappresenti l'azione ciclica svolta.

3.2.2.4 Scansione dei dati riceventi dal sensore

Registrato un campione si può passare alla seconda fase: il rilevamento dell'attività svolta dall'utente. Durante questa fase l'applicazione si mette in ascolto con l'accelerometro e genera una sequenza di dati che descrive l'andamento nel tempo del dispositivo in termini di accelerazione. Questa fase rappresenta il cuore dall'algoritmo in quanto cerca, nella sequenza dei dati recuperata durante il proprio ciclo di vita, la funzione descritta nel campione.

Per ogni punto di dimensione tre (poiché l'accelerazione ha tre coordinate), devono essere eseguiti i seguenti passaggi:

1. Fondere i dati descrittivi dell'accelerazione per due o tre coordinate in un unico valore;
2. Aggiungere il valore calcolato ad una lista;
3. Se la lista ha un numero di elementi minore rispetto al numero di elementi del campione interrompere la procedura;
4. Considerare il primo sottoinsieme, di dimensione pari alla dimensione del campione, più recente nella lista passo passo popolata;

5. Calcolare la sequenza *differenza*, ovvero calcolare la differenza di ogni valore del sottoinsieme con il suo corrispettivo valore del campione
6. Calcolare la media della sequenza differenza;
7. Se la media è al di sotto di un determinato valore l'attività campionata è stata rilevata.

Osservando la procedura appena descritta si può facilmente notare che la valutazione della sequenza *differenza* non è stato eseguita con un calcolo di un integrale ma di una semplice media tra i punti. In questo modo viene ridotto il tempo di calcolo.

3.2.2.5 Codice esempio in Windows Phone

Per la realizzazione di questo metodo si è scelto di creare una classe dedicata d'ausilio ai fini del nostro progetto di nome *CrossCorrelationCore.*

Vengono definite in Codice 3.9 le variabili globali private utili al metodo quindi:

```
public Queue<Vector3> accPointList;
private List<float> unionList;
private float[] sample;
private float[] differenceList;
private int sampleSize;
```

Codice 3.9: Codice C# per la definizione delle variabili per il metodo della cross-correlazione

dove la prima variabile rappresenta la coda dei punti recuperati dall'accelerometro e che usa la tecnica FIFO per la gestione dell'ingresso e uscita dei suoi elementi; la seconda variabile è una collezione di elementi di “fusione” dei dati ricevuti dal sensore; il terzo e il quarto *array* invece rappresentano il campione registrato nelle fasi iniziali dello svolgimento dell'iter e la differenza calcolata in un determinato istante i valori di unione e il campione; l'ultimo intero, invece, ci da l'informazione sulla dimensione del campione e ci è d'ausilio per le verifiche.

Nel costruttore di classe si andranno ad inizializzare la coda e la lista privata, quindi:

```
public ConvolutionCore()
{
        accPointList = new Queue<Vector3>();
        unionList = new List<float>();
}
```

Codice 3.10: Codice C# relativo al costruttore di classe in cui si inizializzano le variabili

Come descritto in Codice 3.10 nelle prime fasi del metodo si dovrà registrare un campione; per non ripetere ad ogni avvio dell'applicazione questa prima fase è possibile salvare il campione su un file e perciò servirà un metodo che carichi in memoria il campione registrato. Il metodo utilizzato per caricare il campione in memoria è scritto a discrezione del programmatore. In questo metodo verrà quindi inizializzato il vettore `sample` e modificato il valore di `sampleSize`.

Per aggiungere i punti rilevati dall'accelerometro, occorre usare un metodo pubblico:

```
public void addPointToList(Vector3 point)
{
        accPointList.Enqueue(point);
}
```

Essendo una tecnica FIFO il nuovo elemento `point` viene aggiunto alla fine della coda.

Nel progetto di tesi, si è scelto di fondere i dati di solo due coordinate y e z corrispondenti alle coordinate più sollecitate dall'accelerazione quando lo smartphone è in tasca. Il metodo che segue mostra in che modo è possibile fondere i dati:

```
public void unifyElementsOfPoints()
{
       while (accPointList.Count != 0)
       {
              Vector3 point = accPointList.Dequeue();
              float unifyValue = (point.Y + point.Z)/2;
              unionList.Add( unifyValue );
       }
}
```

Codice 3.11: Metodo in C# che ha il compito di fondere i valori

```
public void calculateDifference()
{
       differenceList = new float[samplesize];
       if (unionList.Count >= sampleSize)
              for (int i = 0; i < sampleSize; i++)
              {
                     float differenceValue =
       unionList[unionList.Count – (sampleSize – i)] -
       sample[i];
                     if (differenceValue != 0)
                     {
                     float diffPowered =
       (float)Math.Pow(differenceValue , 2);
                     differenceValue =
       diffPowered / (float)Math.Sqrt(diffPowered);
                     }
                     else
                            differenceValue = 0;
                     differenceList[i] = differenceValue;
              }
}
```

Codice 3.12: Metodo in C# che ha il compito di calcolare il vettore differenza

Il metodo presente in Codice 3.11 serve a svuotare la coda popolata dal *retrieving* dei dati dal sensore; per ogni punto in coda viene estratto l'elemento in cima rappresentante il punto di accelerazione che ha atteso di più in coda; i valori di accelerazione dell'asse

y e z vengono fusi insieme attraverso il calcolo della media; infine il valore "fuso" viene aggiunto alla `unionList`.

In seguito viene mostrato il metodo pubblico per il calcolo della differenza tra il segnale rilevato e il campione:

Nel metodo in Codice 3.12 viene inizializzato il vettore `differenceList` con la dimensione del vettore campione e poi viene verificato che la lista dei valori fusi abbia un numero di elementi uguale o maggiore del numero di elementi che formano il vettore di campione. Se la condizione è verificata, si prosegue con l'algoritmo per il calcolo del vettore differenza. All'interno del ciclo i valori che popolano il vettore differenza sono tutti positivi o nulli; ciò faciliterà le cose quando si andrà poi a valutare l'intera sequenza.

L'ultimo metodo riguarda la valutazione del vettore *differenza*:

```
public float calculateAvarage()
{
        float result=0;
        for (int i = 0; i < sampleSize; i++)
                result += differenceList[i];
        result = result / sampleSize;
        return result;
}
```

Codice 3.13: Metodo in C# che ha il compito di calcolare la media dei punti presenti nel vettore differenza

Il metodo in Codice 3.13 non fa altro che calcolare la media degli elementi presenti all'interno del vettore `differenceList`. Se la media calcolata è al di sotto di una certa soglia si può affermare che è stata rilevata l'attività campionata.

Si può quindi passare ad eseguire dei test; nella Figura 3.2 è possibile osservare un grafico in cui viene mostrato l'andamento nel tempo del valore ricavato dall'accelerometro e il valore di media ricavato dall'algoritmo. La polilinea rossa del grafico rappresenta l'insieme dei valori presenti nel vettore unione, in giallo la media calcolata durante l'algoritmo; più i punti di media si avvicinano all'origine (linea grigia

continua) più ci si avvicina a rilevare la scala.

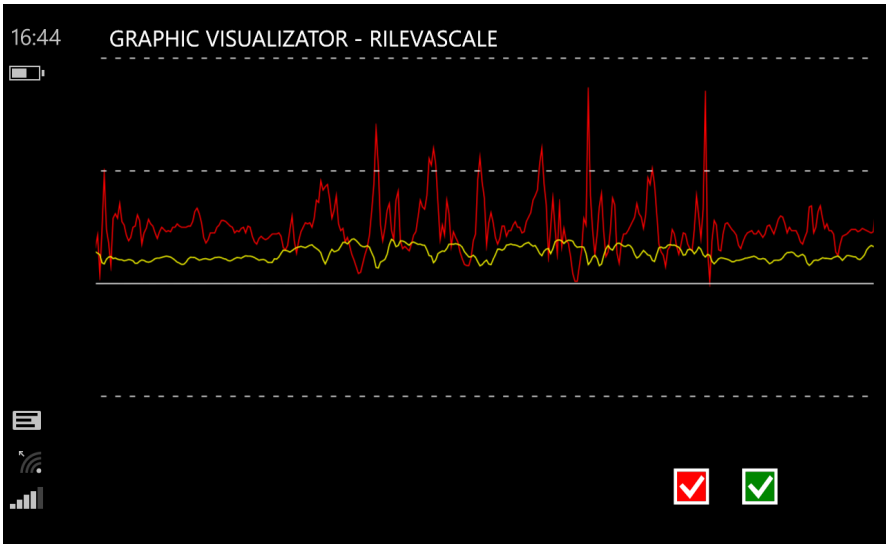


Figura 3.2: Grafico dei risultati

3.3 Sommario

In questo capitolo si è spiegato come estrapolare i dati dall'accelerometro e in che forma ci vengono dati; si è spiegato, poi, cosa si vuole ottenere dal progetto di tesi. Si è poi illustrato il codice di programmazione per ottenere da questo particolare sensore.

Sono stati poi presentati due metodi di manipolazione dei dati progettati per il rilevamento di una scala; sono state descritte le linee di pensiero che sono alla base dei due metodi anticipando, dove intuibile, possibili pregi e difetti.

Infine si sono illustrate linee di codice di programmazione in C# per l'implementazione dei due metodi spiegando, segmento per segmento, le azioni da loro svolte.

4 TEST E ANALISI DEI RISULTATI

In questo capitolo valuteremo l'affidabilità dei metodi precedentemente descritti a seguito di test su campo e relative analisi dei risultati ottenuti. Verranno quindi mostrati i pregi e i difetti di ciascuno dei due metodi motivandone le cause.

Infine verranno descritte possibili modifiche ai due metodi affinché venga ridotto al minimo il numero di falsi negativi e positivi.

4.1 METODO DEI CANESTRI

Per eseguire dei test sul riconoscimento di una scala da parte di uno smartphone con il metodo dei canestri, si sono scelte le seguenti attività:

- Cammino con andatura lenta, normale e veloce
- Corsa con andatura lenta, normale e veloce
- Salto sul posto (saltello, salto normale, salto in alto)
- Salita di una scala con andatura lenta, normale e veloce

Per semplicità, durante i test, si controllano solo i canestri relativi all'asse z, poiché si presuppone che siano i canestri più sollecitati in quanto lo smartphone è in mano; nel particolare, si controlla che il numero di valori all'interno del terzo canestro dell'asse z scenda al di sotto di una certa soglia. Si determina inoltre una dimensione

massima di canestri di 40; tenendo presente che i dati dall'accelerometro vengono recuperati con una latenza di 25 ms da un punto all'altro, si può dire che si studia lo stato dei canestri in un intervallo di tempo di 1 secondo.

4.1.1 Risultato dei test

Nei risultati che vengono mostrati in seguito si userà il valore “1” nel caso in cui il metodo rileva una scala, “0” in caso contrario.

4.1.1.1 Rilevamenti che si ottengono camminando

Ecco i risultati per la prima attività:

CAMMINO			
sensibilità scelta	PASSO LENTO	PASSO NORMALE	PASSO VELOCE
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0
16	0	0	0
17	0	0	0
18	0	0	0
19	0	0	0
20	0	0	0
21	0	0	0
22	0	0	0
23	0	0	0
24	0	0	1
25	0	0	1

Tabella 4.1: verifica dell'algoritmo di rilevamento delle scale su una camminata in piano

Nella tabella 4.1 si può notare come il metodo rilevi dei falsi positivi solo con andamenti molto veloci e sensibilità molto elevate. Infatti per un passo lento o un passo normale il metodo dei canestri non rileva nessuna scala.

4.1.1.2 Rilevamenti che si ottengono correndo

I risultati che si ottengono per l'attività di corsa sono i seguenti:

CORSA			
sensibilità scelta	CORSA LENTA	CORSA NORMALE	CORSA VELOCE
7	0	1	0
8	0	1	0
9	0	1	0
10	0	1	0
11	1	1	1
12	1	1	1
13	1	1	1
14	1	1	1
15	1	1	1
16	1	1	1
17	1	1	1
18	1	1	1
19	1	1	1
20	1	1	1
21	1	1	1
22	1	1	1
23	1	1	1
24	1	1	1
25	1	1	1
26	1	1	1

Tabella 4.2: verifica dell'algoritmo di rilevamento delle scale su una corsa in piano

Nella tabella 4.2 è possibile notare invece che l'algoritmo rileva scale mentre si corre su un terreno non provvisto di scale anche da sensibilità molto basse. Questo dimostra che correndo idealmente con uno smartphone in mano tenendo la faccia dello schermo rivolta verso il cielo, il dispositivo subisce notevoli sollecitazioni dell'accelerazione sull'asse z, soprattutto per quanto riguarda la corsa con andatura normale.

4.1.1.3 Rilevamenti che si ottengono saltando sul posto

I risultati che si ottengono saltando sul posto sono i seguenti:

SALTO DA FERMO			
sensibilità scelta	SALTO BREVE	SALTO NORMALE	SALTO IN ALTO
7	0	1	1
8	1	1	1
9	1	1	1
10	1	1	1
11	1	1	1
12	1	1	1
13	1	1	1
14	1	1	1
15	1	1	1
16	1	1	1
17	1	1	1
18	1	1	1
19	1	1	1
20	1	1	1
21	1	1	1
22	1	1	1
23	1	1	1
24	1	1	1
25	1	1	1
26	1	1	1

Tabella 4.3: verifica dell'algoritmo di rilevamento delle scale sul salto sul posto

Dalla tabella 4.3 si può arrivare alla stessa conclusione che si ha con l'attività fisica precedente; l'algoritmo rileva le scale anche durante le attività di salto sul posto di qualsiasi tipo sia, già dalle sensibilità più basse. Questa è un'ulteriore dimostrazione del fatto che il salto, di qualsiasi entità, va a sollecitare eccessivamente il sensore di accelerometro relativo all'asse z.

4.1.1.4 Rilevamenti che si ottengono salendo una scala

I risultati che si ottengono salendo una scala sono i seguenti:

SCALA			
sensibilità scelta	PASSO LENTO	PASSO NORMALE	PASSO VELOCE
7	0	0	0
8	0	0	0
9	0	0	1
10	0	0	0
11	0	0	1
12	0	0	1
13	0	0	1
14	0	0	1
15	0	1	1
16	0	1	1
17	0	1	1
18	0	1	1
19	0	1	1
20	0	1	1
21	1	1	1
22	1	1	1
23	1	1	1
24	1	1	1
25	1	1	1
26	1	1	1

Tabella 4.4: verifica dell'algoritmo di rilevamento delle scale sul salto sul posto

Dalla tabella 4.4 si può notare che la scala, in base all'andatura scelta, viene rilevata solo da determinate sensibilità in poi. Questo dimostra che, sempre in base all'andatura scelta, il sensore dell'accelerometro relativo all'asse z può essere più o meno sollecitato e di conseguenza la scala, sotto determinate sensibilità potrebbe non essere rilevata.

4.1.2 Qualità del metodo dei canestri

Come si è potuto notare nelle tabelle dei risultati, il metodo dei canestri sembrerebbe inaffidabile; infatti, con il criterio di valutazione scelto e in base a molti dei valori di soglia impostati, vengono registrati molti falsi positivi e molti falsi negativi; occorre quindi trovare delle soluzioni affinché la qualità generale del metodo sia elevata.

4.1.2.1 Falsi negativi

La grande quantità di falsi negativi registrati si può notare nella Tabella 4.4. Per evitare che una scala non venga rilevata, occorre prendere il primo valore di sensibilità, partendo dalle sensibilità basse, affinché la scala venga rilevata con qualsiasi tipo di andamento.

Nello stesso tempo si deve scegliere un valore di sensibilità affinché non si abbiano falsi positivi con altre attività con qualsiasi andatura. Poiché le attività di corsa e salto registrano falsi positivi fin da sensibilità molto basse è necessario trascurarle per la risoluzione di questo particolare problema. In questo caso viene considerato solo l'attività di camminata.

Confrontando la Tabella 4.1 e la Tabella 4.4 si può notare che il giusto range di sensibilità va da 21 a 23.

4.1.2.2 Falsi positivi

Come già osservato dalla Tabella 4.2 e dalla Tabella 4.3, il metodo registra un quantitativo troppo elevato di falsi positivi in casi in cui l'utente corre o salta. Se si dovessero valutare tutti i canestri di tutti gli assi mentre si corre o mentre si salta si può fare un'osservazione molto importante: la corsa e il salto per il cellulare sono due attività completamente differenti.

Infatti, l'attività di corsa sollecita l'accelerometro su tutti gli assi e distribuisce in maniera più uniforme i punti per tutti i canestri per ogni asse; invece l'attività di salto sul posto ideale va interagire unicamente l'accelerazione sull'asse z spostando i punti dal terzo canestro al primo e all'ultimo, dovuti dal raggiungimento del segnale alla portata massima mentre si effettua la spinta e mentre si tocca il terreno dopo il salto.

Per evitare l'enormità di falsi positivi registrati nei precedenti test, quindi, si potrebbero controllare anche i canestri dell'asse z popolati mentre si salta e i canestri dell'asse y mentre si corre. Così facendo la qualità del metodo aumenta nettamente rendendolo affidabile al nostro caso.

Il limite di questo metodo però è l'algoritmo è pensato per funzionare solo quando lo smartphone è in mano.

4.2 Metodo della cross-correlazione

Per eseguire dei test sul riconoscimento di una scala da parte di uno smartphone con il metodo della cross-correlazione, si sono scelte le seguenti attività come con il metodo precedente:

- Cammino con andatura lenta, normale e veloce
- Corsa con andatura lenta, normale e veloce
- Salto sul posto (saltello, salto nornale, salto in alto)
- Salita di una scala con andatura lenta, normale e veloce

Il metodo della cross-correlazione è pensato per funzionare con lo smartphone in tasca; per questi test si è usato lo stesso vestito e si è scelto di ripetere 5 volte lo stessa tipologia di test per valore di soglia; inoltre si son scelti 4 valori di soglia con cui valutare il rilevamento della scala.

Per quanto riguarda il campionamento invece, si è scelto di registrare un solo gradino di scala e nei casi in cui il test consiste nel salire una scala, si è scelto di salire la scala precedentemente campionata.

4.2.1 Risultato dei test

Come con il metodo dei canestri, si mostrano delle tabelle dove viene riportato il valore "1" se la scala è stata rilevata in base alla soglia e all'andatura scelta, "0" in caso contrario.

4.2.1.1 Risultati che si ottengono camminando

Ecco i risultati per la prima attività:

CAMMINO			
sensibilità scelta	PASSO LENTO	PASSO NORMALE	PASSO VELOCE
0,05	0	0	0
0,05	0	0	0
0,05	0	0	0
0,05	0	0	0
0,05	0	0	0
0,1	0	0	0
0,1	0	0	0
0,1	0	0	0
0,1	0	0	0
0,1	0	0	0
0,15	0	0	0
0,15	0	0	0
0,15	0	0	0
0,15	0	0	0
0,15	0	0	0
0,2	1	0	0
0,2	1	0	0
0,2	1	0	0
0,2	1	1	0
0,2	1	0	0

Tabella 4.5: verifica dell'algoritmo di rilevamento delle scale camminando in piano

Nella Tabella 4.5 è possibile notare che si hanno falsi positivi solo quando si cammina con andatura lenta e con sensibilità relativamente elevata.

4.2.1.2 Rilevamento mentre si corre

I seguenti sono i risultati per l'attività di corsa:

CORSA			
sensibilità scelta	CORSA LENTA	CORSA NORMALE	CORSA VELOCE
0,05	0	0	0
0,05	0	0	0
0,05	0	0	0
0,05	0	0	0
0,05	0	0	0
0,1	0	0	0
0,1	0	0	0
0,1	0	0	0
0,1	0	0	0
0,1	0	0	0
0,15	0	0	0
0,15	0	0	0
0,15	0	0	0
0,15	0	0	0
0,15	0	0	0
0,2	0	0	0
0,2	0	0	0
0,2	0	0	0
0,2	0	0	0
0,2	0	0	0

Tabella 4.6: verifica dell'algoritmo di rilevamento delle scale mentre si corre

Come si può notare dalla Tabella 4.6, il metodo della cross-correlazione per l'attività di corsa è esente da falsi positivi. Il risultato in questo caso è ottimo.

4.2.1.3 Rilevamento mentre si salta sul posto

I seguenti sono i risultati ottenuti mentre si salta sul posto:

SALTO			
sensibilità scelta	SALTO BREVE	SALTO NORMALE	SALTO IN ALTO
0,05	0	0	0
0,05	0	0	0
0,05	0	0	0
0,05	0	0	0
0,05	0	0	0
0,1	0	0	0
0,1	0	0	0
0,1	0	0	0
0,1	0	0	0
0,1	0	0	0
0,15	0	0	0
0,15	0	0	0
0,15	0	0	0
0,15	0	0	0
0,15	0	0	0
0,2	0	0	0
0,2	0	0	0
0,2	1	0	0
0,2	0	0	0
0,2	0	0	0

Tabella 4.7: verifica dell'algoritmo di rilevamento delle scale mentre si salta sul posto

Come per l'attività di corsa, il metodo non rileva, a parte un caso isolato, scale durante l'attività di salto. Il risultato è soddisfacente.

4.2.1.4 Rilamenti mentre si sale una scala

In seguito vengono mostrati i risultati ottenuti mentre si sale una scala:

SCALA			
sensibilità scelta	PASSO LENTO	PASSO NORMALE	PASSO VELOCE
0,05	0	0	0
0,05	0	0	0
0,05	0	0	0
0,05	0	0	0
0,05	0	0	0
0,1	0	0	0
0,1	0	0	0
0,1	0	0	0
0,1	0	0	0
0,1	0	0	0
0,15	1	1	0
0,15	1	1	0
0,15	1	1	0
0,15	1	1	0
0,15	0	1	0
0,2	1	1	0
0,2	1	1	0
0,2	1	1	0
0,2	1	1	0
0,2	1	1	0

Tabella 4.8: verifica dell'algoritmo di rilevamento delle scale mentre si sale una scala

La Tabella 4.8 ci espone dei dati molto interessanti dal punto di vista qualitativo del metodo; per quanto riguarda il salire una scala con andatura lenta e normale, la scala viene rilevata solo da una determinata soglia minima, mentre se si sale la scala con passo veloce, la scala non viene rilevata neanche con valori di sensibilità relativamente alti.

Questo ci fa capire che, per quanto riguarda l'andatura lenta e normale, è molto difficile eguagliare un segnale campionato con un secondo segnale registrato in tempo reale, motivo per cui si sceglie una certa soglia di “assomiglianza”. Mentre, il movimento che si ottiene con andature veloci è troppo differente rispetto ai precedenti; il segnale catturato non sarà neanche somigliante al segnale campione.

4.2.2 Qualità del metodo della cross-correlazione

Alla luce dei dati mostrati, il metodo della cross-correlazione al contrario del metodo dei canestri risulta essere descritto da un numero molto basso di falsi positivi ma, al contempo, da un numero molto alto di falsi negativi. Come con il metodo dei canestri, si deve trovare una soluzione affinché la qualità generale del metodo migliori.

4.2.2.1 Falsi positivi

Per ridurre al minimo la quantità di falsi positivi è necessario scegliere un valore di sensibilità abbastanza elevato da poter rilevare una scala mentre la si sale con tutte le andature. Nello stesso tempo, bisogna trovare un valore di sensibilità abbastanza basso da poter ridurre o annullare il numero di falsi positivi ottenuti svolgendo un'attività fisica diversa dal salire una scala, con qualsiasi andatura.

Nella Tabella 4.6 e nella tabella 4.7 si può notare che durante la corsa e durante il salto non si verificano mai, o quasi mai, falsi positivi; le attività possono essere trascurate per questo tipo di valutazione. Inoltre si trascura anche l'andatura veloce mentre si sale una scala poiché non viene rilevata neanche con sensibilità molto elevate per le motivazioni precedentemente esplicitate.

Incrociando la Tabella 4.5 e la tabella 4.8 si può facilmente dedurre che la sensibilità giusta da impostare al metodo è di 0.15 . Sempre incrociando le due tabelle, si può notare quanto il movimento di una camminata lenta sia sostanzialmente simile al movimento che si fa salendo una scala.

4.2.2.2 Falsi negativi

Dal punto di vista di falsi negativi il discorso è diverso e l'implementazione così com'è stata descritta nel precedente capitolo risulta essere limitante. Infatti i dati mostrano quanto sia restrittivo questo tipo di algoritmo in quanto la scala viene rilevata solo quando vengono registrati precisi tipi di sequenze.

Per poter ridurre i falsi negativi è possibile scansionare il segnale rilevato dall'accelerometro con più campioni; nel nostro caso, bastava prendere un campione dell'utente che sale un gradino con andatura veloce e la scala in quel caso sarebbe stata poi rilevata.

Tuttavia sono stati volutamente dimenticati alcuni dettagli:

- I test sono stati eseguiti da una sola persona
- I test sono stati eseguiti con un abito e quindi un solo tipo di tasca
- I test sono stati eseguiti con un solo tipo di scala

A questo punto bisogna necessariamente porsi una domanda: quanti campioni bisogna registrare affinché non si verifichino più falsi negativi?

Per semplicità si può comunque impostare l'applicazione per funzionare su una sola persona in quanto uno smartphone può essere posseduto e utilizzato in genere da una persona sola. Tuttavia la risposta rimane incompleta poiché durante il ciclo di vita dello smartphone e si indossano più abiti (sollecitando in maniera differente lo smartphone in tasca) e si possono percorrere molte scale differenti che portano inevitabilmente a dei falsi negativi.

4.3 Sommario

In questo capitolo vengono svolti numerosi test con delle implementazioni dei metodi descritti nel progetto di tesi. Per ogni test viene poi riportato il risultato argomentando le motivazioni di tale risultato.

In base ai risultati ottenuti si è valutata la qualità dei metodi, principalmente in base al numero di falsi positivi e falsi negativi riscontrati e se possibile vengono proposti possibili migliorie ai metodi.

Per concludere questo capitolo si può dire, dati alla mano, che il metodo dei canestri risulta essere molto affidabile ma risulta particolarmente restrittivo il suo metodo d'uso, ovvero, tenere lo smartphone in mano. Invece il metodo della cross-correlazione, al contrario del metodo precedente, risulta essere molto più versatile come metodo d'uso (infatti il metodo può essere mirato al rilevamento di una determinata azione tenendo lo smartphone in un determinato modo) ma ha il difetto che deve essere istruito, raccogliendo più campioni possibile per poter essere definito affidabile.

Conclusioni

In questo volume di tesi è stato introdotto lo smartphone, descrivendone i vantaggi rispetto ad altri dispositivi mobili come il telefono cellulare e il PDA. Si è raccontato brevemente come lo smartphone ha preso tendenza sorpassando, di fatto, i cellulari in termini di vendita. Si è parlato poi di sistemi operativi con cui gli smartphone lavorano descrivendone le loro caratteristiche in termini di funzionalità e interfaccia grafica e sono stati illustrati tutti i possibili sensori che si possono trovare all'interno di questo particolare dispositivo.

In seguito sono state introdotte le applicazioni necessarie per sviluppare un'applicazione Windows Phone e Android descrivendo brevemente le relative macchine virtuali con cui eseguire il debug delle applicazioni sviluppate. Nel caso di Windows Phone si è parlata anche della licenza necessaria affinché sia possibile installare le applicazioni su un dispositivo fisico. Si è poi parlato dell'applicazione utile alla realizzazione e al test degli algoritmi.

Si è spiegato poi in che modo, a livello di software, funziona l'accelerometro dello smartphone e sono stati descritti due metodi progettati per il rilevamento di una particolare attività fisica umana: il salire una scala.

Infine, dopo aver eseguito dei test e riportati i risultati in tabelle, sono stati valutati e confrontati i due metodi in base al numero di falsi positivi e falsi negativi riscontrati esponendo, possibili migliorie al fine di aumentare l'affidabilità per ciascuno dei due metodi.

Alla luce dei fatti è possibile concludere che, tramite uno smartphone, è possibile rilevare le attività fisiche svolte dall'uomo. Questo progetto di tesi è alla base di un ragionamento in cui intende dimostrare che uno smartphone potrebbe interpretare tutte le azioni svolte dall'uomo e fornire, in tempo reale, informazioni sull'utente, per esempio il suo stato di salute motoria o i suoi tempi di reazione ad un determinato stimolo, di qualunque natura esso sia.

L'uso di semplici metodi di scansione o verifica del segnale in particolari momenti di esecuzione porta però alla loro inaffidabilità in quanto, in qualche modo, sono limitati a pochi casi d'uso. Un esempio lo si ha avuto con il metodo della cross-correlazione dove la scala non viene rilevata se percorsa in una determinata maniera.

Per risolvere questo problema è possibile far in modo che più metodi entrino in funzione in base all'attività svolta e in base al modo con cui l'attività viene svolta. Però bisogna far capire al nostro smartphone quando attivare un metodo piuttosto che un altro. Si può pensare quindi ad una sorta di apprendimento del programma che nelle prime fasi impara tutte le attività per cui è stato progettato.

Per il futuro sarebbe molto interessante apprendere nozioni di IA (Intelligenza artificiale) relative all'apprendimento automatico e capire in che modo è possibile applicarle a questo progetto di tesi in modo tale da rendere lo smartphone ancora più intelligente nel campo del rilevamento delle attività umane.

Bibliografia

[1] Museo Nazionale Scienza e Tecnologia Leonardo Da Vinci, "Breve storia del telefono", 2015, http://www.museoscienza.org/approfondimenti/documenti/telefono/

[2] L. Savioli, Grande, piccolo e di nuovo grande: la grande storia del cellulare in 16 modelli, 2015, http://www.ilsole24ore.com/art/tecnologie/2014-10-14/la-storia-cellulare-16-modelli-motorola-startac-161517.shtml?uuid=ABQ8E82B&nmll=2707#navigation

[3] Computer conoscenza, Storia dei computer palmari, 2015, http://it.wingwit.com/hardware/computer-drives-storage/52163.html

[4] CellulareMagazine.it, Smartphone: raddoppiano le vendite, 2006, http://www.cellulare-magazine.it/sito/news/5224/Smartphone_raddoppiano_le_vendite.php

[5] F. Giambarresi, In Europa gli smartphone superano i cellulari, 2011, http://www.webnews.it/2011/09/12/in-europa-gli-smartphone-superano-i-cellulari/

[6] Wikipedia, Sistema Operativo, 2015, http://it.wikipedia.org/wiki/Sistema_operativo

[7] ITPRO IT ANALYSIS. BUSINESS INSIGHT, , 2015

[8] iSpazio, , 2015, http://www.ispazio.net/wp-content/uploads/2014/06/IMG_0009.png

[9] Android World, , 2015, http://www.androidworld.it/wp-content/uploads/2013/11/Android-4.4-app-drawer-2.png

[10] The Apache Software Foundation, Apache License, Version 2.0, 2004, http://www.apache.org/licenses/LICENSE-2.0

[11] G. Mahajan, , 2015, https://gauravmahajan1.files.wordpress.com/2013/02/wp8-0.png

[12] Android-er, , 2013, http://2.bp.blogspot.com/-

9pj3qrItWcE/UORqv5VFJHI/AAAAAAAAGxQ/EL7HEupTwkA/s1600/avd_02.png

[13] Scilab, 2015, www.scilab.org

[14] CeCILL, CeCILL and Free Software, 2015, http://www.cecill.info/index.en.html

[15] GNU Operating System, Gnu General Public License, version 2, 2014, http://www.gnu.org/licenses/gpl-2.0.ht

[16] MathWorks, MATLAB, Il linguaggio del calcolo tecnico, 2015, http://it.mathworks.com/products/matlab/

[17] msdn, , 2015, https://msdn.microsoft.com/jj133827.Petzold_Figure1_hires%28en-us,MSDN.10%29.jpg

[18] Wikipedia, Correlazione incrociata, 2015, http://it.wikipedia.org/wiki/Correlazione_incrociata