Alma Mater Studiorum
Università di Bologna
Campus di Cesena

SCUOLA DI INGEGNERIA E ARCHITETTURA
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

GlovePi: un device wearable a supporto della comunicazione many-to-many tra utenti sordo-ciechi

Relatore:
Dott.ssa
Silvia Mirri

Presentata da:
Lorenzo Monti

Sessione III
Anno Accademico 2015-2016

a voi, genitori speciali.

DON'T PANIC

Douglas Adams

tratto da Guida galattica per gli autostoppisti

Introduzione

È riconosciuto che più della metà delle informazioni sensoriali che analizziamo con il nostro cervello provengono da ciò che vediamo, rendendo la vista il senso dominante [1]. In accordo con il paper "Vision trumps all the other senses" del biologo molecolare John Medina, la "battaglia" tra i nostri cinque sensi è vinta da quello visivo. Inoltre è stato osservato come gli stimoli visivi possano alterare la percezione del gusto e dell'olfatto [2]- [4], motivando il ben noto motto "mangiare prima con gli occhi", come già espresso in [5]- [6]. È importante menzionare, quindi, come i problemi che riguardano la vista possono fortemente influenzare il modo di comunicare ed interagire con contenuti e dispositivi digitali che richiedono tecnologie assistive [7]- [8] e che possono divenire essenziali nello loro attività quotidiane, migliorando la loro indipendenza [9]- [10].

In questo contesto, la combinazione di cecità e sordità è causa di una importante barriera di comunicazione [11] e bisogni educazionali [12] che richiedono un adattamento ad-hoc negli strumenti che ogni persona sordo-cieca dovrà adottare [13]. Per poter comunicare ed interagire con gli altri, molte persone sordo-cieche necessitano di una costante presenza da parte di un operatore il quale sarà il suo interprete. Infatti, le persone sordo-cieche possono comunicare in diversi modi, sulla base delle loro condizioni e sulle risorse effettivamente disponibili a loro. Alcuni dei metodi più utilizzati sono sicuramente il LIS, LIS tattile e l'alfabeto manuale (alfabeto ad una mano oppure alfabeto a due mani) [14].

Negli alfabeti a due mani le lettere vengono digitate sul palmo della mano

oppure sulle dita di chi riceve la comunicazione, utilizzando la mano dominante come strumento di comunicazione. Vi sono diversi alfabeti nati con questo intento, tra cui: il British manual alphabet (utilizzato in Inghilterra), l'alfabeto Lorm (utilizzato in Austria, Germania, Polonia), l'alfabeto Malossi (utilizzato in Italia). Ognuno di questi alfabeti differiscono sulla base di come le mani interagiscono oppure come le dita/palmo sono toccati. Nel British manual alphabet ad esempio, l'interazione avviene tramite gesture che si basano sul tocco delle dita oppure del palmo della mano e dal movimento su di essi. Nell'alfabeto Lorm invece, vengono tracciate linee, cerchi e altre figure sul palmo e sulle dita della mano. L'alfabeto Malossi permette l'interazione e la comunicazione attraverso il tocco ed il pizzico su determinati punti delle dita e sul palmo della mano.

Quest'ultimo alfabeto fu inventato da Eugenio Malossi, un insegnante sordo-cieco italiano. Esso è considerato estremamente intuitivo e si basa su simboli tattili di base, nella fattispecie le lettere sono posizionate sopra ad ogni falange in senso orario partendo dal pollice. Poichè il numero di lettere è maggiore del numero di falangi in una mano, le lettere dalla A alla O vegono toccate nella parte frontale della mano, mentre dalla lettera P alla lettera Z saranno pizzicate lateralmente con due dita. Grazie alla sua semplicità è utilizzato anche da bambini e persone affette da deficit cognitivi i quali non possono imparare metodi più complessi (come l'alfabeto Lorm e il Braille).

Questo è lo scenario in cui il dispositivo digitale più idoneo che potrebbe essere sfruttato è un device wearable, e un guanto rappresenta il mezzo naturale per la comunicazione e l'interazione tra persone sordo-cieche. Alcuni esperimenti sono stati fatti in tal senso come descritto in [15] e [16], altri prodotti sono stati commercializzati e sono in grado di gestire la comunicazione proveniente da diversi alfabeti come in [17], ovviamente ciò implica un costo rilevante che gli utenti sordo-ciechi devono affrontare.

In tale contesto si inserisce questo lavoro di tesi, il quale si pone come obiettivo innanzitutto quello di progettare ed implementare una tecnologia assistiva a basso costo ed open-source, in modo da supportare la comunicazio-

ne sociale delle persone affette da sordocecità utilizzando l'alfabeto Malossi. Questo documento di tesi è organizzato nel seguente modo. Nel primo capitolo verrà esposta una panoramica sul mondo della sordo-cecità e saranno presentate le cause e l'impatto che si riferiscono ad esso, inoltre il lettore verrà introdotto ai concetti base e alle metodologie di comunicazione che verranno analizzate. Nel secondo capitolo verranno presentate invece le diverse fasi di progettazione del guanto, ponendo inizialmente uno sguardo propedeutico alle tecnologie consone che ci serviranno a svillupare il sistema, verrà presentata l'architettura di sistema e analizzato poi, in maniera specifica, ogni componente della stessa. Sempre in questo capitolo verrà esposta un'architettura di sistema per la comunicazione many-to-many in ambiente distribuito appoggiandoci a TuCSoN, tecnologia già matura in tale ambito. Il terzo capitolo riguarda l'implementazione vera e propria di GlovePi e i test sulle performance dello stesso. Il documento di tesi tratterà infine le conclusioni e gli sviluppi futuri.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione alla comunicazione tra persone sordo-cieche

In questo capitolo viene introdotta la condizione di sordocecità, dapprima con una definizione formale e in seguito illustrando quelle che sono le cause e l'impatto riferite alla perdita di udito e vista. In seguito il lettore verrà introdotto al concetto di comunicazione e come questo sia stato studiato ed espanso, in diverse forme, al fine di superare la barriera di isolamento in cui le persone affette da sordocecità si trovano. Nell'ultimo paragrafo di questo capitolo andremo ad analizzare alcune delle tecnologie di riferimento che sono allo stato dell'arte in questo settore.

1.1 Definizione di sordocecità

La definizione di *sordocecità* è la più recente fra quelle che riguardano le minorazioni civili, riconoscendo nella particolarità della doppia grave limitazione sensoriale una "disabilità specifica unica" come descritto in [18]. La disposizione di riferimento è la Legge 24 giugno 2010, n. 107. a Legge 107/2010, all'articolo 2, definisce "sordocieche le persone cui siano distin-

tamente riconosciute entrambe le minorazioni, sulla base della legislazione vigente, in materia di sordità civile e di cecità civile." L'articolo 3 precisa che "la condizione di sordocieco viene riconosciuta al soggetto che dall'accertamento risulti in possesso dei requisiti già previsti dalla legislazione vigente rispettivamente in materia di sordità civile e di cecità civile ai fini dell'ottenimento delle indennità già definite in base alle vigenti normative relative alle due distinte minorazioni."

Un'altra definizione la possiamo trovare nel dizionario di medicina Treccani, di seguito la citazione testuale: "*Stato patologico costituito dalla duplice mancanza, congenita o acquisita, della facoltà uditiva e della vista. Si tratta di una grave disabilità che comporta importanti difficoltà o impedimento nei comuni atti della vita quotidiana (per es., limiti alla comunicazione e alla mobilità). I problemi della s. hanno fatto nascere varie associazioni di supporto che si occupano sia di s. sia di pluriminorazioni dell'ambito psicosensoriale. Aspetti legislativi sulla s. sono in corso di definizione per migliorare la qualità di vita di questi pazienti che necessitano di particolari metodiche di comunicazione e apprendimento*".

Infine, possiamo trovare una definizione formale anche dalla NCDB (National Center on Deaf-Blindness),l'organizzazione che si occupa di persone affette da sordocecità. La definizione arriva direttametne da Barbara Miles, consulente educativo che lavora presso l'Università del Vermont e per l'associazione stessa. Ecco la citazione testuale: *"It may seem that deaf-blindness refers to a total inability to see or hear. However, in reality deaf-blindness is a condition in which the combination of hearing and visual losses in children cause "such severe communication and other developmental and educational needs that they cannot be accommodated in special education programs solely for children with deafness or children with blindness" (34 CFR 300.7 (c) (2), 1999) or multiple disabilities. Children who are called deaf-blind are singled out educationally because impairments of sight and hearing require thoughtful and unique educational approaches in order to ensure that children with this disability have the opportunity to reach their full potential. A*

person who is deaf-blind has a unique experience of the world. For people who can see and hear, the world extends outward as far as his or her eyes and ears can reach. For the young child who is deaf-blind, the world is initially much narrower. If the child is profoundly deaf and totally blind, his or her experience of the world extends only as far as the fingertips can reach. Such children are effectively alone if no one is touching them. Their concepts of the world depend upon what or whom they have had the opportunity to physically contact. If a child who is deaf-blind has some usable vision and/or hearing, as many do, her or his world will be enlarged. Many children called deaf-blind have enough vision to be able to move about in their environments, recognize familiar people, see sign language at close distances, and perhaps read large print. Others have sufficient hearing to recognize familiar sounds, understand some speech, or develop speech themselves. The range of sensory impairments included in the term "deaf-blindness" is great."

1.2 Cause e impatto riferite alla perdita di udito e vista

1.2.1 Maggiori cause

Vi sono molteplici cause che possono portare alla sordità e/o alla cecità, alcune comuni altre più singolari che possono mostrarsi, sia in maniera *congenita* che *acquisita*, per maggior chiarezza è giusto fare un distinguo. La prima riguarda persone che nascono con questa patologia mentre la seconda si riferisce ad una problematica acquisita dopo la nascita, la sordocecità *acquisita* proprio quest'ultima ha 3 accezioni come descritto in [20]:

1. **Una persona nata sorda o ipoudente e solamente dopo si va a deteriorare la vista:** In questo caso la persona può imparare forme di comunicazione visuale (vedi LIS, Auslan, labbiale), per persone con disabilità cognitive si può usare il Makaton. Vi sono impianti possibili (non sempre) per cercare di risolvere il problema, come impianti co-

cleari o hearing aid ma dipende dal livello e dalla tipologia di perdita dell'udito.

2. **Una persona nata cieca o ipovedente e solamente dopo si va a deteriorare l'udito:** In questo caso invece si possono sviluppare skills, cercando di compensare con l'udito. Per migliorare l'orientamento e la mobilità, queste persone possono usare un cane guida. Se hanno un residuo visivo possono acquisire skills per usare al meglio la loro vista per essere indipendenti nella vita di tutti i giorni.

3. **Una persona vedente e udente che in seguito ha un deterioramento incrementale dei due sensi (non necessariamente allo stesso momento):** in quest'ultimo caso queste persone possono imparare normalmente le varie forme di comunicazione. La modalità sarà ovviamente soggettiva sulla base del residio visivo/uditivo della persona sottoposta.

Per maggior chiarezza andremo qui di seguito ad elencare alcune delle maggiori cause:

Congenito dalla nascita :

- **Sindrome da rosolia congenita:**come descritto in [22] e in [21], questa sindrome può colpire il feto in sviluppo nel caso si abbia contratto la rosolia durante il primo trimestre gravidanza. Il neonato non è generalmente colpito dalla malattia se la rosolia è contratta durante il terzo trimestre, o tra le 26-40 settimane dopo il concepimento.
- **Prematurità:** un parto il cui travaglio ha luogo tra la 22^{a} settimana e la 37^{a} settimana completa di gestazione è definito parto prematuro o parto pretermine come descritto in [23] e in [24].
- **Sindrome di CHARGE:**come descritto in [25], la sindrome di CHARGE è una patologia rara che può colpire diverse parti del

corpo e viene riconosciuta come una delle maggiori cause di cecità e sordità. La parola "CHARGE" corrisponde all'acronimo delle più comuni caratteristiche di questa malattia. **C = Coloboma** E' un'anomalia congenita consistente in un difetto di una struttura oculare. Può dare origine a una serie di difetti della visione, come il campo visivo ristretto, intolleranza per la luce viva e, a volte, un rischio di ulteriori complicanze oculari come il distacco della retina. **H = Heart defects (difetti cardiaci)** La sindrome CHARGE è correlata a una molteplicità di difetti cardiaci diversi, alcuni dei quali richiedono interventi chirurgici. **A = Atresia of the Choanae (atresia delle coane)** Uno o entrambe le cavità nasali possono essere ostruite oppure presentarsi insolitamente strette. Il difetto è correggibile chirurgicamente, ma spesso sono necessari più interventi. **R = Retardation of growth and developmental delay (ritardo della crescita e dello sviluppo)** Nella Sindrome di CHARGE i problemi legati all'ormone della crescita sono piuttosto rari. Sembra infatti che il ritardo nella crescita sia dovuto essenzialmente alle difficoltà di alimentazione e, in seguito, all'assenza della pubertà. Le persone con CHARGE possono presentare vari livelli di disabilità e il ritardo dello sviluppo sembra derivare dai deficit sensoriali e dai problemi cronici di salute e di equilibrio. **G = Genital anomalies (malformazioni dei genitali)** Le malformazioni colpiscono i genitali esterni. I problemi più comuni consistono nella mancata discesa dei testicoli (testicoli ritenuti) o nelle dimensioni ridotte del pene. Alcune bambine hanno le labbra della vulva piccole. **E = Ear anomalies (malformazioni dell'orecchio)** L'orecchio esterno, medio e interno può essere oggetto di alcune problematiche; tra le più comuni si segnalano la fusione degli ossicini nell'orecchio medio, la formazione cronica di liquido nell'orecchio medio, il canale auricolare stretto o assente e malformazioni dell'orecchio esterno.

- **Cytomegalovirus (CMV):** Questo genere di virus colpisce diverse specie di primati tra cui l'uomo, essa è la più grande delle herpesvirus, provoca una vasta gamma di sindromi cliniche, da infezione asintomatica a grave malattia in pazienti immunocompromessi come descritto in [26].

Acquisita durante il tempo :

- **Sindrome di Husher:** Essa è la maggiore causa di sordocecità dopo l'infanzia. E' autosomica recessiva ed è costituita da una perdita uditiva congenita combinato e una lenta ma progressiva perdita della retina pigmentosa ossia una perdita progressiva della vista dovuta al deterioramento della retina. Ad oggi questa sindrome non è curabile.
- **Malattie e infezione:** Sia malattie che infezioni possono portare a menomazioni uditive e visuali.
- **Invecchiamento:** Anche l'invecchiamento può portare ad un deterioramento degenerativo elevato.

1.2.2 Tipologie

Date le cause appena descritte vi possono essere diverse tipologie di perdita d'udito e di perdita della vista come descritto in [27]. Andremo di seguito ad analizzarle:

Tipologie di perdita d'udito

- **Ipoacusia trasmissiva:** Definita come ostruzione, infezione, abnormalità strutturale oppure altre condizioni che prendono di mira la parte esterna dell'orecchio e che come risultato provocano la perdita d'udito. Nella maggior parte dei casi questa tipologia è riconducibile all'età della persona.

- **Ipoacusia neurosensoriale:** In questo caso invece è dovuto ad un danneggiamento del nervo acustico oppure una parte interna dell'orecchio. Può avvenire tramite la prolungata esposizione a forti rumori che danneggiano le cellule sensoriali della coclea (in questo caso è possibile recuperare parte del udito tramite un impianto), oppure tramite la presbiacusia, dovuta principalmente all'età della persona.
- **Perdita dell'udito mista:** Una combinazione delle tipologie appena descritte (ipocusia trasmissiva e neurosensoriale).
- **perdita dell'udito centrale:** In questo caso la perdita di udito è derivante da un deficit nelle aree del cervello (corteccia uditiva) che ricevono e processano l'input uditivo, o le vie che vanno dal tronco cerebrale alla corteccia uditiva.

Tipologie di perdita di vista

- **Perdita di visione centrale:** Detta anche *degenerazione maculare*, essa è una una patologia multifattoriale che colpisce la zona centrale della retina, detta per l'appunto macula. Ha un andamento progressivo e può portare alla completa cecità. Si presenta solitamente nelle persona con età superiore ai 50 anni (una persona su sette ne è affetta, è questo aumenta con l'avanzare dell'età). Questa degenerazione, tra le altre cose, affligge l'abilità di vedere i dettagli, di leggere, riconoscere volti e vedere espressioni facciali o leggere le labbra. Le cause sembrano essere legate alla predisposizione genetica, l'età, l'ambiente ed avere un fototipo "chiaro".
- **Perdita di visione periferica:** Detto anche *glaucoma* che definisce un insieme di disturbi dell'occhio che distruggono lentamente ma in maniera progessiva il nervo ottico. Può provocare gravi lesioni irreversibili che possono arrivare all'ipovisione se non addirittura alla cecità. Anche in questo caso non esistono cure ma ci

sono trattamenti per poter tenere sotto controllo o rallentare un ulteriore degrado della vista.

- **Visione sfuocata:** La visione sfuocata è molto spesso causata dalla cataratta in un processo di progressiva perdita di trasparenza del cristallino dell'occhio. Essa è una delle cause principali di disabilità visiva. Anche in questo caso è causata comunemente dall'invecchiamento della persona. Quando compaiono i sintomi, la visione può essere migliorata attraverso l'uso di occhiali idonei e illuminazione adeguata. In molti casi vi è la possibilità di ripristinare la visione attraverso la chirurgia.
- **Visione frammentaria:** Definita anche con il termine *retinopatia diabetica* ed è riscontrata nei soggetti affetti da diabete mellito nella grande maggioranza dei casi. Il diabete può influenzare il sistema circolatorio della retina e provocare micro emorragie che, con il passare del tempo, possono danneggiare il tessuto retineo e portare alla cecità. Con un adeguato controllo continuativo è possibile mitigare il problema mantenendo il livello glicemico (causa della patologia) entro una soglia adeguata.

1.2.3 Impatto nella qualità della vita

Immaginando di avere un impedimento sia dal punto di vista uditivo che da quello visivo, sicuramente la quantità di energie per carpire informazione per dare un senso, una semantica all'ambiente che ci circonda, sono elevatissime. Anche le attività di vita quotidiana (ADL o ADLs) come ad esempio usare il telefono oppure leggere l'orario diventano molto difficoltose se non impossibili. L'impatto è a tutto tondo, infatti comprende la *comunicazione*: ogni persona reagisce diversamente tuttavia ad ogni piccolo cambiamento di percezione può influire la nostra capacità di comunicare. Ad esempio, con una piccola perdita di sensibilità d'udito alcuni suoni potrebbero sembrare simili e quindi rendere confusa la persona, similmente la perdita di sensibilità

nell'apparato visivo invece potrebbe impedire la capacità di percepire gli elementi non verbali, parte essenziale dell'interazione sociale (circa il 65% [28]). La *mobilità*, altro punto essenziale è toccato in maniera lapalissiana, basti pensare a quanto una visione ridotta possa portare a sensirsi meno sicuri nel muoversi in maniera indipendente e conseguentemente rendere più difficoltoso completare le proprie ADLs. La problematica di comunicazione e mobilità come descritto in [29] e in [30] è un probabile veicolo di isolamento sociale ed emotivo. Una ridotta capacità di accedere al mondo e una conseguente riduzione di indipendenza, potrà portare all'*isolamento*, alla *depressione* e *riduzione di autostima* come descritto in [31] e in [32].

1.3 Comunicazione

1.3.1 Cos'è la comunicazione

La comunicazione (dal latino cum = con, e munire = legare, costruire e dal latino communico = mettere in comune, far partecipe) nella sua prima definizione è l'insieme dei fenomeni che comportano la distribuzione di informazioni. Il termine ha una moltitudine di accezioni in base al contesto, ma grazie al linguista Roman Jakobson possiamo schematizzare sei aspetti fondamentali della comunicazione verbale come descritto in [35], che sono riconducibili anche ad altre forme di comunicazione come ad esempio quella di suoni o di gesti. Egli individuò:

1. Un **mittente** che è colui che invia il messaggio.
2. Un **messaggio** che è l'oggetto dell'invio.
3. Un **destinatario** che è colui che riceve il messaggio.
4. Un **contesto** che è l'insieme della situazione generale e delle circostanze particolari in cui ogni evento comunicativo è inserito.
5. Un **codice comune** a mittente e destinatario.

6. Un **canale** di comunicazione (connessione fisica o psicologica) tra mittente e destinatario.

1.3.2 Comunicare nella sordocecità

Immaginiamo di essere in una stanza buia, con le orecchie tappate e gli occhi coperti, senza alcuna possibilità di recepire suoni, di vedere dove mettiamo i piedi o cosa intralcia il nostro percorso. E' questa la condizione in cui si trova perennemente chi è affetto da gravi menomazioni della vista e dell'udito. Nonostante la grave patologia, esistono vari metodi di comunicazione per le persone affettate da tale disturbo.

La comunicazione è un aspetto di vitale importanza per una persona sordocieca, ma anche la più complessa. Comunicare per un sordocieco non significa solo parlare, ma vivere, entrare in contatto con gli altri, conoscere il mondo e superare la barriera di isolamento in cui si trovano. Per comunicare vengono utilizzati sistemi di comunicazione che privilegiano come principale canale sensoriale il tatto: si tratta di codici che vanno dal più semplice ai più complesso, il cui uso varia a seconda dell'età e delle capacità residue possedute dalla singola persona [34]. Esistono diversi metodi di comunicazione non verbali, applicati alternativamente a seconda che vi sia una perdita totale di vista e di udito o che sussistano residui nei canali sensoriali. Molto spesso, però, non è possibile ricorrere ad un vero e proprio codice linguistico; basti pensare al caso di persone sordocieche o pluriminorati psicosensoriali con deficit a livello cognitivo. In questi casi possiamo ricorrere ad una moltitudine di forme di comunicazione mediante l'utilizzo di gesti o movimenti del corpo (forma comunicativa comportamentale), di oggetti, immagini che rappresentino azioni, situazioni o bisogni. Andremo ad elencare qui di seguito le principali.

Lingua Italiana dei Segni (LIS)

La Lingua Dei Segni viene utilizzata in tutto il mondo da persone sorde con ancora un buon residuo visivo. Questo linguaggio visivo-gestuale impe-

gna principalmente le dita, le mani, i polsi e le braccia come visibile in figura 1.1. Con "gestuale" si intende che le unità del linguaggio sono costituite da movimenti. Con "visivo" si intende dire che il linguaggio è ricevuto dal canale visivo. La LIS è differente rispetto ad ogni paese del mondo, in Italia inoltre ci sono, come per il linguaggio vocale, molte variazioni dialettali che soddisfano le esigenze linguistiche di ogni piccola comunità, questo crea non poche difficoltà ai sordi che vogliono comunicare anche a di fuori del loro territorio [33].

Figura 1.1: Lingua Italiana dei Segni

Lingua Italiana dei Segni Tattile

Quando anche il residuo visibile non è disponibile, la LIS viene impiegata a livello tattile. La mano della persona sordocieca viene posta su quella dell'interlocutore per poter ricevere la comunicazione, in questo modo essa può percepire tattilmente il segno grazie ad opportuni accorgimenti [33].

Comunicazione gestuale

Numerosi sono i sistemi di comunicazione di tipo gestuale, nei quali i gesti vengono utilizzati per esprimere parole o concetti. In alcuni casi si tratta di sistemi semplici, fatti di gesti anche personali e in genere compiuti spontaneamente; in altri di sistemi codificati, ossia veri e propri linguaggi strutturati convenzionalmente, con una propria grammatica e sintassi. Tra i quali il principale è la Lingua Italiana dei Segni (LIS). Questo metodo di comunicazione è possibile solamente con persone sorde con ancora un buon residuo visivo [33].

Figura 1.2: Lis Tattile

Comunicazione comportamentale

Si cerca di mandare messaggi e farsi capire attraverso movimenti del corpo sin dall'infanzia, gesti spontanei ed espressioni del viso. Si tratta di una forma personale di espressione, per lo più compresa da poche persone che conoscono bene il bambino. Viene utilizzata per comunicare un numero ristretto di bisogni e pertanto viene impiegata come punto di partenza per l'apprendimento successivo di altri codici di comunicazione [33].

Figura 1.3: Comunicazione comportamentale

Comunicazione pittografica

Utilizzato per lo più da persone con un residuo visivo, questa sistema di comunicazione si basa sul riconoscimento di immagini. Lo scambio di messaggi avviene attraverso cartellini disegnati, che rappresentano azioni, oggetti oppure situazioni rilevanti [33].

Figura 1.4: Comunicazione pittografica

Comunicazione oggettuale

Questo sistema di comunicazione è utilizzato da persone con seri problemi di vista e difficoltà di apprendimento. Esso si basa sulla rappresentazione di situazioni o azioni attraverso degli oggetti. Questi oggetti devono essere riconoscibili e significativi, mantenere una buona somiglianza tattile con ciò che rappresentano. Ad esempio, un piccolo bicchiere ed un cucchiaio possono essere utilizzati dal bambino per esprimere il bisogno di mangiare o di bere [33].

Figura 1.5: Comunicazione oggettuale

Dattilologia

Sistema utilizzato normalmente in concomitanza con altri sistemi tattili o visivi, come ad esempio la Lingua dei Segni; Esso è composto da una serie di movimenti effettuati con le mani con i quali è possibile rappresentare le singole lettere dell'alfabeto [33].

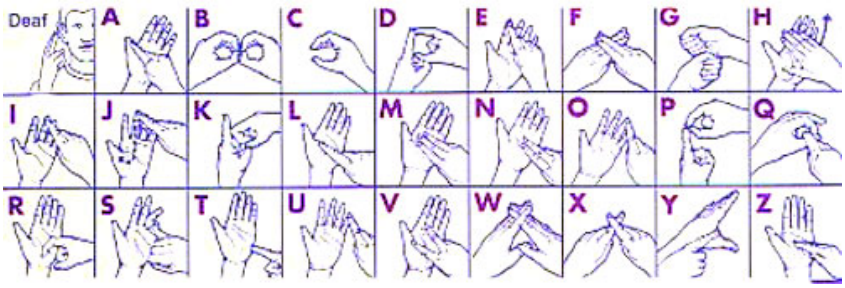

Figura 1.6: Dattilologia

Braille

Forse il più "famoso" tra i metodi di comunicazione per non vedenti, il braille è un sistema di scrittura e lettura a rilievo per non vedenti ed ipovedenti messo a punto dal francese Louis Braille nella prima metà del XIX secolo. Esso è costituito da punti in rilievo a cui corrispondono le lettere dell'alfabeto. La lettura Braille viene effettuata di solito dall'indice della mano

destra, seguito da quello della mano sinistra che ha il compito di orientare nella individuazione delle righe [33].

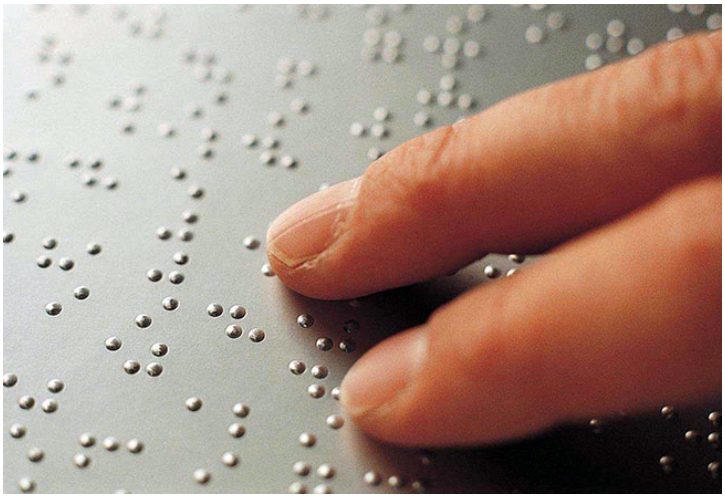

Figura 1.7: Braille

Stampatello sulla mano

Sistema utilizzabile solamente se la persona è riuscita ad imparare e scrivere prima di diventar sordocieco, esso permette di utilizzare il palmo della propria mano (o anche altre parti del corpo) come foglio dove poter ricevere la comunicazione scritta da un interlocutore. Si utilizza lo stampatello per poter chiarificare le lettere nel miglior modo possibile. Molte persone anziane, grazie a questo metodo, possono mantere il contatto con il mondo esterno anche dopo aver perso vista ed udito [33].

Metodo Tadoma

Questo metodo comunicativo è attuabile attraverso il riconoscimento dei suoni vocali. La persona affetta da sordocecità appoggia il pollice sulle labbra e il palmo della mano sulle guancie di chi parla,le diverse forme che acquisirà la bocca e le labbra per ogni suono emesso, attraverso il tatto, ne permetterà la comprensione [33].

Figura 1.8: metodo Tadoma

Metodo Malossi

Metodo ideato da Eugenio Malossi lui stesso affetto da sordocecità e da cui prende il nome. Si tratta di un metodo per l'utilizzo del quale si presuppone la conoscenza della lingua italiana scritta: ad ogni falange della mano viene associata una lettera dell'alfabeto, diventando una sorta di macchina da scrivere. La parola viene, così, composta toccando o pizzicando le falangi o il palmo. Questo metodo è utilizzato generalmente dalle persone che hanno appreso la lettura e la scrittura prima di diventare sordocieche [33].

Figura 1.9: metodo Malossi

1.4 Tecnologie e comunicazione: stato dell'arte

Questo progetto di tesi prende in considerazione una specifica fase del percorso riabilitativo, quello del reinserimento della persona con disabilità nel proprio normale ambiente di vita e della consegna degli strumenti per diventare egli stesso attivo protagonista della propria integrazione e partecipazione alla vita sociale. In questa fase giocano un ruolo di primo piano gli ausili tecnici, oggi detti anche tecnologie assistive (assistive technologies) ossia quei prodotti, dispositivi o sistemi tecnologici atti a superare barriere esistenti nell'ambiente o a compensare specifiche limitazioni funzionali così da facilitare o rendere possibili le attività della vita quotidiana. Le tecnologie prese in considerazione spaziano dai dispositivi di ausilio elettronici, informatici e telematici più sofisticati per la comunicazione e la mobilità, ai dispositivi per l'automazione dell'ambiente (domotica), agli adattamenti strutturali dell'ambiente domestico o di vita sociale, ai semplici ausili per le attività basilari della vita quotidiana, a strumenti e materiali con funzione preventiva di disabilità secondarie. Rientra nell'ambito di interesse di questo tema anche l'ergonomia generale dell'ambiente, dei prodotti e dei servizi di uso quotidiano secondo quell'approccio oggi noto come design for all (progettazione che risponda esigenze di tutta l'utenza, sia normodotata che disabile). La ricerca in questo settore richiede l'apporto di una vasta gamma di competenze interdisciplinari di tipo tecnologico, clinico, sociale, economico ed educativo, orientate alle problematiche dell'applicazione "sul campo" della tecnologia al servizio dell'autonomia personale e familiare della persona disabile, nonchè della sua integrazione nella scuola, nel lavoro e nella società.

Dopo l'analisi dei precedenti capitoli in cui vengono descritte le cause, le problematiche e l'importanza di poter comunicare con il mondo esterno, in questo capitolo daremo un sguardo a tali tecnologie. In letteratura possiamo trovare tecnologie per:

1. *l'apprendimento* fulcro centrale e base nativa per la comunicazione so-

prattutto per chi ne soffre dalla nascita (vedesi paragrafo 1.2.1) ma non solo;

2. *il movimento* tema importantissimo per l'indipendenza della persona e per quanto riguarda il portare a termine anche le più semplici attività giornaliere (dette ADLs);

3. *la comunicazione* come descritto nel paragrafo 1.3.2, parte essenziale per una persona sordocieca per potersi aprire al mondo ed avere una socialità.

Nella fattispecie, in questa relazione, analizzeremo quest'ultima. Alcune di queste tecnologie si ispirano a metodi di comunicazione già esistenti come quelli visti nel paragrafo 1.3.2, altre invece hanno un'origine e un respiro più vicino alla ricerca pura.

1.4.1 Device Braille

Il Braille, come descritto in 1.3.2, è un metodo complesso e che ha bisogno di un dispositivo di input, uno di output e una fase di elaborazione per la traduzione da testo a Braille e viceversa. Per tale ragione esistono in commercio diverse tipologie di dispositivi:

1. **Display Braille**, è un dispositivo tattile usato per leggere testo da un computer o l'input da un notetaker Braille come visibile in Figura 1.10. Esso è composto da una riga di "celle" che possiede sei o otto puntine di metallo oppure nylon (la versione ad otto puntine è maggiormente utilizzata per i computer). Queste puntine sono controllate elettronicamente per muoversi su e gù in base alla lettura del carattere del sistema sorgente. Vi sono display Braille da 40, 65 e 80 celle. Diversi studi dimostrano come integrare la rappresentazione aptica del Braille utilizzando la vibrazione del touch-screen di uno smartphone come dimostrato in [36].

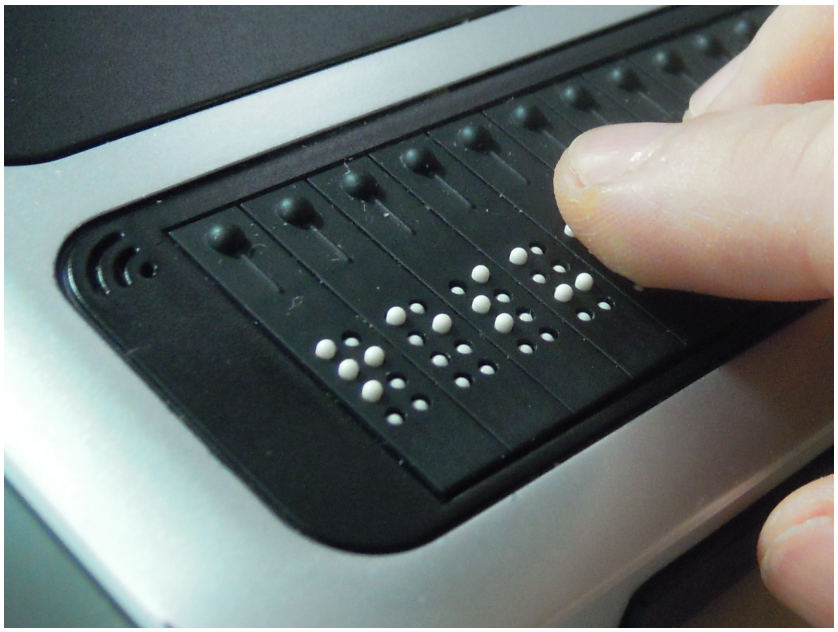

Figura 1.10: Display braille

2. **Notetakers Braille**, generalmente multifunzionale, esso è simile ad un palmare ma più grande e molto più costoso. Oltre ad essere utilizzato per l'elaborazione di testi quest'ultimo supporta funzioni di rubrica, note, invio e-mail e inoltre facilita l'accesso al Web. Il notetaker inoltre include il correttore ortografico, è in grado di tradurre un testo per il braille e di creare un output per il Braille Embosser. I servizi di posta elettronica possono essere ottenuti dal notetaker attraverso la maggior parte dei fornitori di servizi Internet. Software aggiuntivi possono essere utilizzati per espandere le funzionalità, per includere la gestione del denaro, foglio di calcolo, analisi scientifica e matematica. Oltre che in commercio possiamo trovare diversi esempi anche in letteratura come in [37].

3. **Braille Embosser**, spesso i computer sono utilizzati per la traduzione del testo in Braille, esso è stampato attraverso uno speciale embosser. L'algoritmo per la traduzione non è semplice poichè le regole dipendono dalla pronuncia e dal significato. Metodi più vicini per ora alla ricerca,

ma che supponiamo un giorno potranno essere di uso commerciale, ambiscono ad esempio alla traduzione two-way in tempo reale tra l'inglese e il Braille utilizzando un guanto ad-hoc come interfaccia tattile [38].

1.4.2 Device per dattilologia

La dattilologia è un importantissimo mezzo di comunicazione per molte persone sordocieche, pertanto lo sviluppo di approcci automatizzati di questo tipo sono particolarmente importanti, anche per facilitare la comunicazione con persone non affette da questa patologia. La dattilologia è utilizzata sia per la comunicazione faccia a faccia sia per la comunicazione a lunga distanza. In questo modo è possibile migliorare l'indipendenza di queste persone e opportunità di lavoro. Vi sono diverse accezioni di utilizzo di questa tecnica in base al paese, e le soluzioni tecniche proposte variano in base ai dettagli delle diverse lingue. Sono state sviluppate diverse tecnologie in merito a partire dagli anni '80, andremo di seguito ad esaminarle:

1. **Dexter I [40]** consiste in quattro dita di alluminio lavorato e un pollice uniti al palmo della mano. Questa protesi robotica è proiettata in verticale. Le dita, indipendenti l'una dall'altra sono state costruite per poter eseguire movimenti del tutto simili alle mani umane. Vi è un microprocessore per controllare la mano posto nel vano al dì sotto del dispositivo. Nella progettazione originale hannno inserito il microprocessore a 8 bit 8085 prodotto da Intel, hanno utilizzato *Forth* come linguaggio di programmazione e vi è stato un supporto ad una memoria e a timer per misurare il movimento della mano e la durata di ogni posizione della mano. E' possibile comunicare con il device attraverso la comunicazione seriale tra un laptop Epson HX-20 e il sistema stesso.

2. **Dexter II e III**, il progetto Dexter venne portato avanti negli anni a seguire, un secondo prototipo (Dexter II) fu progettato nel 1988 al fine di sopperire ad alcune mancanze della prima mano robotica inserendo servo motori per muovere le dita rendendo così la mano molto più

minuta. E' stato inserito un processore diverso, lo Zilog Z80 e lo stesso laptop è stato utilizzato. Infine il progetto Dexter II evolse in Dexter III per portare il guanto in produzione, anche se con scarsi risultati commerciali.

3. **Ralph (robotic alpahbet)** [**41**], progetto più recente ma con le medesime prospettive di Dexter. Anche in questo caso abbiamo una mano robotica gestita da servo motori e un microprocessore per gestire la traduzione (ASCII/lingua dei segni e viceversa) e il movimento delle dita. La miglior progettazione meccanica rese Ralph più veloce rispetto i predecessori nel movimento delle dita. Questo progetto utilizzò sia lo stesso linguaggio che lo stesso laptop dei precedenti progetti.

4. **The Handtapper - uno sviluppo inglese** [**42**], la progettazione di Handtapper è significativamente diversa rispetto ai progetti precedenti (tutti Americani) dovuta al differente approccio di dattilologia tra i due paesi. In questo caso la comunicazione è precisata lettera per lettera sulla mano sinistra di chi "ascolta". La mano del ascoltatore rimane fissa mentre il messaggio viene "scritto" in maniera aptica lettera per lettera.

5. **AUTOSEM** [**43**], progetto più recente che cerca di abbassare i costi di produzione utilizzando l'hardware presente nei controller delle console per videogiochi e negli smartphone. Esso si ispira alle bandiere semaforiche presenti nei contesti navali per comunicare tra utenti sordo-ciechi usando le loro stesse mani. La combinazione di diversi orientamenti delle mani corrispondono alle diverse lettere dell'alfabeto. Come input sono utilizzati device a basso costo con un acceleremetro integrato al loro interno, mentre per l'output sono utilizzati feedback vibrotattili, comuni in questi device.

1.4.3 Device Malossi

L'alfabeto Malossi, come descritto in 1.3.2, è un metodo semplice e per questo utilizzato anche per persone con gravi problemi cognitivi. Esso è prettamente utilizzato in Italia, in altre parti dell'Europa come l'Austria, la Germania e la Polonia è infatti utilizzato principalmente l'alfabeto Lorm (data la similitudine tra i due alfabeti, progetti riferiti a quest'ultimo verranno trattati in questa sezione).

1. **DbGlove** [**45**], l'articolo mostra un sistema harware e software che supporta persone sordo-cieche al fine di farle interagire con l'ambiente attraverso l'alfabeto Malossi. Consiste in una periferica wearable di input/output equipaggiata di sensori e attuatori che agisce come interfaccia natuale abilitando la comunicazione sociale attraverso l'alfabeto sopracitato. L'interazione è gestita da un ambiente software il quale traduce testo in sitoli tattili e viceversa, per eseguire comandi e inviare messaggi ad altri utenti. Fornisce inoltre un feedback multimodale su diversi dispositivi di standard output per supportare l'interazione tra utenti sordo-ciechi e utenti normodotati.

2. **Mobile Lorm Glove** [**46**], si basa sull'alfabeto Lorm, alfabeto molto simile al Malossi, ma in questo caso per ogni lettera dell'alfabeto romano è associato un movimento sul palmo della mano, quindi anche in questo caso la persona sordo-cieca mostra il palmo della mano (solitamente la destra a differenza del Malossi) all'interlocutore, quest'ultimo andrà poi a digitare lettera per lettera sul palmo apposto. Questo progetto introduce un prototipo che traduce questo alfabeto in testo e viceversa. E' presentato un guanto con il quale una persona sordocieca può comporre messaggi attraverso sensori di pressione posti nel palmo del guanto. Sulla parte posteriore del guanto sono stati inseriti ulteriori sensori per poter avere feedback tattili alla ricezione del messaggio.

1.4.4 Altre soluzioni

Alcune soluzioni in letteratura non aderiscono precisamente ai metodi nati per la comunicazione nella sordocecità ma abbracciano un contesto più ampio pur attendosi sempre allo stesso fine.

1. **Scrittura basata sulla pressione attraverso i device mobili** [**47**], in questo paper vengono analizzati i device mobili per sfruttarne la pressione come nuovo metodo di scrittura per sordociechi. Il presupposto di questo articolo indiga sulle potenzialità di questa tecnologia e come questa sia stata poco sfruttata nell'ambito mobile. Ad esempio è possibile distinguere lettere maiuscole con lettere minuscole semplicemente decidendo la pressione del tocco.

2. **Silooets** [**48**], questo progetto cerca di convertire immagini in un esperienza audio-tattile attraverso l'utilizzo di un software. Il sistema può codificare immagini in audio o gli equivalenti tattili (sfruttando il Braille).

3. **Cintura aptica per la comunicazione non verbale in constesti sociali** [**49**], l'interazione sociale è composta da spunti non verbali per circa il 65% [28], le persone sordocieche non possono però accedere a queste informazioni visuali ponendole di fatto in svantaggio rispetto a persone normodotate. Questo articolo, dopo un attento focus group che aveva come obiettivo quello di carpire le informazioni che una persona cieca vorrebbe ottenere durante una sessione sociale, ha partorito un prototipo di cintura vibrotattile atta al veicolare tutti i segnali non verbali durante una sessione sociale. Una microcamera posta nella montatura di un paio di occhiali è servita come base di input per un sistema di face detection, ciò ha fornito tutti gli spunti non verbali richiesti dal focus group tra le quali il numero e la posizione delle persone nel campo visivo dell'utente, la posizione degli occhi dell'interlocutore. L'algoritmo divide ogni frame catturato in 7 sezioni (come i contattori nella cintura aptica). Quindi una volta riconosciuto il volto viene

attivato il contattore rispetto alla regione del frame. La durata della vibrazione identifica la distanza con le persona tra l'utente e la persona. L'identificazione passa attraveso un db, se questo non è presente l'utente può inserirlo.

Capitolo 2

Progettazione

Ora che abbiamo introdotto i vari metodi di comunicazione nel primo capitolo della relazione possiamo decidere quale impiegare per questo progetto, nella fattispecie è stato scelto l'alfabeto Malossi come base di partenza. I motivi alla base della scelta sono molteplici:

1. **L'intuitività:** essendo stato creato da una persona sordo cieca, tale metodo ha sicuramente messo in primo piano questa caratteristica.

2. **Semplicità d'uso:** Il metodo Malossi è sicuramente uno dei più semplici, infatti è impiegato ampiamente con bambini e persone con deficit cognitivi, che non riescono ad imparare metodi più complessi e sofisticati.

3. **Supporto multilingua e multiconcetto:** La disposizione delle aree può essere applicata a diverse lingue e fonemi. Oltre a rappresentare lettere, segnali singoli o multipli, i segnali tattili possono esprimere concetti più articolati.

4. **L'analogia informatica:** Ultimo ma non ultimo per importanza, l'analogia informatica è fortissima in questo caso, infatti la mano può essere vista benissimo come una tastiera nella quale possiamo digitare caratteri e creare frasi al fine di poter comunicare con l'ambiente esterno.

Il processo di analisi dello stato dell'arte è fondamentale, e per capire come progettare il nostro dispositivo per poter carpire le informazioni chiave di cui una persona affetta da sordo-cecità avrebbe bisogno da una tecnologia di questo genere. Saranno quindi essenziali: (i) un guanto fisico per poter apporrre la sensoristica, (ii) un'unità di input e di elaborazione dei dati (single-board computer) e infine (iii) un'unità di output (smartphone). Nella prima parte di questo capitolo saranno quindi descritte e motivate propedeuticamente le tecnologie utili per la progettazione di questo progetto, in seguito verranno descritti i crismi di progettazione partendo dall'architettura per poi esaminare con perizia ogni componente del sistema.

2.1 Single-board computer

Il single-board computer è un calcolatore completo costruito su una singola scheda elettronica, scelta imprescindibile per questo progetto al fine di minimizzare dimensione e costi. Queste schede ricalcano le architetture hardware dei personal computer, al loro interno infatti troviamo in generale un SoC (System-on-a-Chip) per computare e periferiche di I/O per comunicare con il mondo esterno.

Questo, ad oggi, è un mercato molto fervido ed esistono diversi modelli per tutte le esigenze e per tutte le "tasche". Alcuni dei modelli più diffusi sono:

- *Arduino.*
- *Banana Pi.*
- *BeagleBone Black.*
- *DragonBoard 410c.*
- *Firefly.*
- *Intel Galileo.*

- *ODROID-C2 e ODROID-XU4.*
- *Orange Pi.*
- *PINE64.*
- *Raspberry Pi.*
- *The HummingBoard-Gate.*

Ma al fine di mantenere bassi i costi, che ricordiamo essere uno degli obiettivi di questo progetto, andremo ad analizzare solamente i dispositivi più economici: Arduino e Raspberry Pi.

2.1.1 Arduino

Arduino è una piattaforma hardware composta da una serie di schede elettroniche dotate di un microcontrollore. è stata ideata e sviluppata da alcuni membri dell'Interaction Design Institute di Ivrea come strumento per la prototipazione rapida e per scopi hobbistici, didattici e professionali. Il nome della scheda deriva da quello del bar di Ivrea frequentato dai fondatori del progetto, nome che richiama a sua volta quello di Arduino d'Ivrea, Re d'Italia nel 1002. L'hardware che è completamente realizzato in Italia e consiste tipicamente in un microcontrollore a 8-bit AVR prodotto dalla Atmel, inoltre vengono inseriti componenti complementari al fine di facilitare l'incorporazione con altri circuiti. Nelle schede vengono utilizzati chip della serie megaAVR (ATmega8, ATmega168, ATmega328, ATmega1280 e ATmega2560). Molte delle schede arduino includono un regolatore lineare di tensione a 5 volt e un oscillatore a cristallo a 16 Mhz (LilyPad ha un clock a 8 Mhz e non è dotata di un regolatore di tensione). Tutti i modelli sono programmabili attraverso un IDE chiamato ARDUINO ad oggi alla versione 1.8.1, esso è disponibile per Windows, Mac OS X (dalla versione 10.7), e Linux (su architetture x86, x64 e ARM).

Modelli

Ad oggi sono state realizzate differenti versioni dell'hardware Arduino suddivise per caratteristiche e obiettivi come visibile in Figura 2.1. Andremo di seguito ad analizzare i modelli più caratterizzanti:

- *Arduino Uno*, evoluzione della Duemilanove con un differente chip, programmabile e più economico, dedicato alla conversione USB-seriale. La board è basata su un microcontrollore della Atmel, ATmega328, che incorpora una memoria ed uno storage necessario a lanciare gli *sketch* (programmi). In particolare vi sono a disposizione:
  - **32 KB di memoria flash** (di cui 0,5 KB sono occupati dal bootloader), questa memoria nasce con l'obiettivo di salvare i nostri sketch che saranno lanciati all'avvio della board;
  - **2 KB di SRAM** (RAM statica), questa memoria è utilizzata a runtime (ad esempio per tener traccia delle variabili);
  - **1 KB di EEPROM** (Electrically Erasable Programmable ROM), per poter salvare dati e parametri di configurazione utili e/o necessari al programma.

  è giusto sottolineare come l'architettura del microcontrollore sia **RISC** ad 8-bit con una frequenza di clock di 16Mhz. I principali pin del microcontrollore sono posti ai lati della scheda, in particolare vi sono 14 pin digitali e 6 pin analogici, in quanto l'Atmega328 è dotato anche di un AD Converter (Analog to Digital) a 6 canali e con 10 bit di risoluzione. Ciascuno dei pin digitali può essere programmato in maniera indipendente tramite software, ed utilizzato con pin di GPIO, alcuni di questi hanno inoltre funzionalità aggiuntive:
  - **Una porta seriale** di tipo TTL caratterizzata da segnali RX (in ricezione) e TX (in trasmissione).
  - Due pin per poter triggerare **un interrupt** verso il microcontrollore, a cui sarà possibile associargli un specifica azione.

  - **PWM**, segnale di output per ottenere tensione media variabile (legata alla durata di un impulso nel tempo).
  - **Una porta SPI** utilizzabile per interconnettersi con device esterni con l'apposito bus sincrono.
  - **Un aporta I2C** utilizzabile come per SPI con un bus sincrono.
  - Il pin che stabilisce **la tensione di riferimento (AREF, Analog REFerence)** per tutti gli input analogici. Essa è necessario per stabilire il range di valori che l'AD Converter produce sulla base dei 10 bit a disposizione.

- *Arduino Mega2560*, è un'evoluzione dell'Arduino Mega, questa scheda monta un chip Mega 2560 che può essere programmato e attraverso un bootloader (residente nella memoria flash) è possibile uploadare il codice senza un programmatore esterno. Comunica attraverso il protocollo STK500 e possiede:

  - **256KB KB di memoria flash** (di cui 8 KB sono occupati dal bootloader), memoria per poter salvare i nostri sketch,
  - **8 KB di SRAM**;
  - **4 KB di EEPROM**.

  Sottolineiamo che la scheda può operare con un supporto esterno dai 6 a 20 volts, si raccomanda che il range però stia nell'intorno tra i 7 e i 12 volts.

- *Arduino Yùn*, che fa uso di un ATmega32u4 e del processore Atheros AR9331, una delle novità di questa scheda è la compatibilità con una distribuzione Linux basata su OpenWrt e chiamata Linino OS, poichè non è mai stata dotata prima di un sistema operativo su cui poter andare ad operare ma l'esecuzione avveniva attraverso il salvataggio del codice sulla memoria flash. Altre novità riguardano il supporto built-in sulla board di un modulo Wi-fi e Ethernet, una porta USB

ed un alloggio per la micro-SD (dove effettivamente vi risiede sistema operativo e dati).

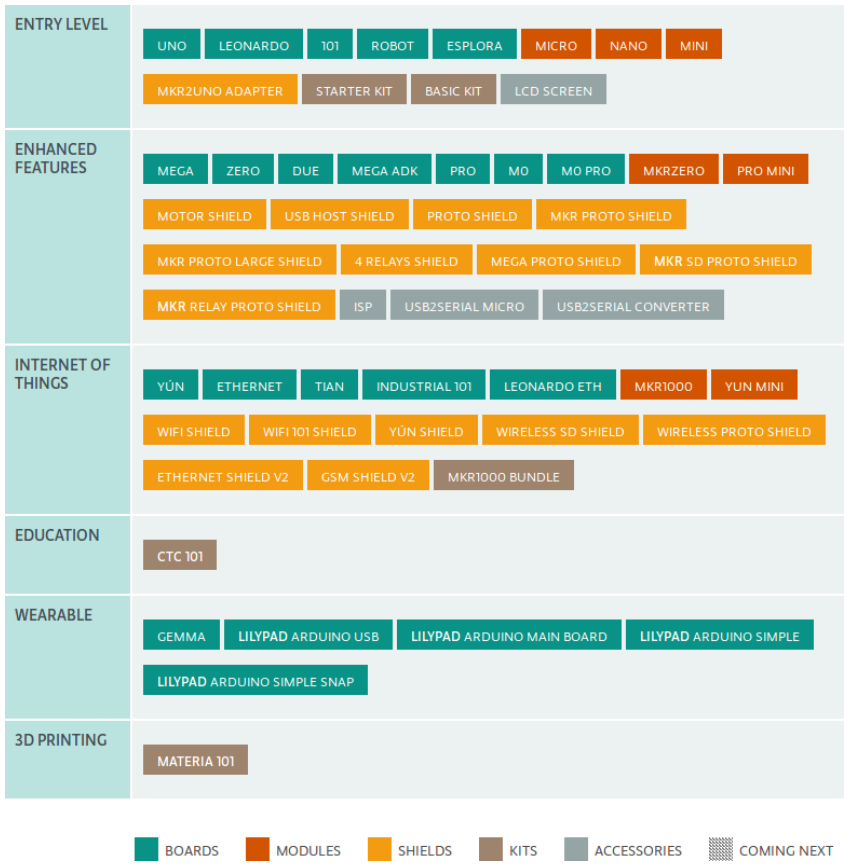

Figura 2.1: Modelli Arduino

2.1.2 Raspberry Pi

Sviluppata in Gran Bretagna, l'idea di base di questa scheda è la realizzazione di un dispositivo economico, concepito per stimolare l'insegnamento di base dell'informatica e della programmazione nelle scuole. Il progetto gira attorno ad un un System-on-a-chip (SoC) Broadcom (BCM2835 fino a Raspberry B+, BCM2836 per il Raspberry Pi 2, o BCM2837 per Raspberry Pi 3) che incorpora un processore ARM, una GPU VideoCore IV e 256 Megabyte o 512 Megabyte oppure 1 Gigabyte di memoria come visibile in Figura 2.2.

All'interno della scheda non prevede nè un hard disk nè tantomeno un'unità a stato solido, affiancando invece la possibilità di inserire una scheda SD per il boot e la memoria non volatile. La scheda è stata progettata per ospitare sistemi operativi basati sul kernel Linux o RISC OS. è assemblata fisicamente in Galles, nel SoNy UK Technology Centre.

Modelli

Durante gli anni (dal 2006) sono stati sviluppati diversi modelli di questo computer in miniatura. La prima generazione del Raspberry Pi (Raspberry Pi 1 Model A+), messa in commercio nel novembre 2014. Essa soppianta il primo prototipo (Raspberry Pi 1 Model A). Questa prima versione del prototipo ha le seguenti caratteristiche:

- basata su un'**architettura ARM** era montata su una scheda grande circa come una chiavetta USB,
- **una porta USB** su un lato;
- **porta HDMI** sull'altro;
- **pin GPIO**, 26 per la precione;
- **inferfaccia CSI**, per la videocamera;
- **alloggio per microSD**.

Nel luglio del 2014 la fondazione annuncia un nuovo modello: Raspberry Pi 1 Model B+, le caratteristiche rispetto il precedente modello sono molto migliorate

- **la ram** all'interno del SoC è stata aumentata fino a 512MB,
- è stata aggiunta un ulteriore **porta usb**;
- sono stati aggiunti altri **pin GPIO** crescendo fino a 40.

La fondazione vuole qualcosa di ancora più piccolo ed economico, punto essenziale per la loro filosofia e ad aprile 2014 viene rilasciata il modello Raspberry Pi zero. Minimizzata in termini di superficie rispetto alle precedenti versioni e con un costo pari a 5 US$. Di seguito inseriamo le caratteristiche della scheda:

- **1GHz, Single-core CPU**;
- **512MB RAM**;
- **Mini-HDMI port**;
- **porta Micro-USB OTG**;
- **40-pin GPIO HAT-compatibili**;
- **inferfaccia CSI**, per la videocamera.

Gli ultimi due modelli (Rasberry Pi 2 model B e Raspberry Pi 3 model B) sono stati rilasciati rispettivamente a febbraio 2015 e febbraio 2016, la prima propone:

- miglioramento decisivo del processore inserendo un **ARM Cortex-A7 quad-core a 900 Mhz**,
- un miglioramento nella **RAM**, portando ad 1GB;
- l'aggiunta di 2 **porte USB** arrivando a 4;
- inserimento una **porta Ethernet**;
- aggiunta di **una VideoCore IV 3D graphics core** nel SoC.

Per l'ultima versione di questa scheda invece, si è puntato non tanto sul miglioramento dal punto di vista delle performance quanto sul miglioramento delle periferiche pre installate a bordo del chip, hanno infatti pensato di aggiungere un modulo per il Wi-Fi 802.11n e uno per il Bluetooth 4.1 Low Energy (BLE). Nel febbraio 2017 esce Raspberry Pi Zero Wireless(uscita nei

giorni in cui sto scrivendo questa tesi) un'estensione per la famiglia Raspberry Pi Zero. Tale scheda ha tutte le funzionalità dell'originale ma in più sono stati aggiunte features orientate alla connettività:

- **802.11 b/g/n wireless LAN**;
- **Bluetooth 4.1**;
- **Bluetooth Low Energy (BLE)**.

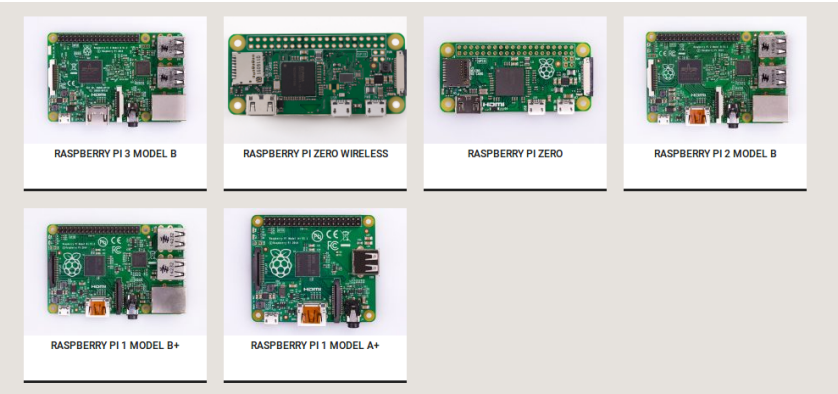

Figura 2.2: Modelli Raspberry

2.1.3 La scelta del single-board computer

Dopo aver analizzato minuziosamente le diverse offerte che il mercato ci offre, abbiamo optato per Raspberry Pi. Di seguito andremo a motivare la nostra scelta:

- **Economicità del prodotto,** questo fa parte della filosofia della fondazione Raspberry Pi. Per questo progetto è essenziale pensare anche a questo al fine di creare un prototipo ad un prezzo contenuto.
- **è linux-based,** nel nostro caso è di primaria importanza avere un sistema operativo stabile ed open source al quale affidarsi e per poter elaborare e mantener traccia dei dati.

- **Comunità attiva,** ultimo ma non ultimo per importanza, tra le caratteristiche che ci hanno portato a decidere di utilizzare questo dispositivo è la comunità, Raspberry Pi vanta infatti la possibilità di trovare molti progetti open source e molti siti in cui viene offerto l'aiuto da persone esperte.

2.2 I sistemi operativi mobile

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

- Apple iOS.
- Windows Phone.
- Android.

Andremo, qui di seguito a spiegare funzionamento e caratteristiche di ognuno.

2.2.1 Apple iOS

Apple iOS è il sistema operativo sviluppato da Apple per dispositivi mobile quali iPhone, iPod Touch e iPad, la prima versione è stata presentata il 9 gennaio 2007 al Macworld Conference and Expo. Questo è un sistema operativo proprietario e disponibile solo per dispositivi Apple. Tale limitazione rende possibile un'ottimizzazione delle risorse circa l'hardware posto nel dispositivo. Ad oggi, l'ultima release presentata al pubblico è la iOS 10.2.1 (23

gennaio 2017). Scendendo nei retroscena di questo sistema operativo possiamo mostrare una panoramica sulle caratteristiche tecniche. iOS come Mac OS X, è una derivazione di UNIX ed usa un microkernel XNU Mach basato sul sistema operativo open source Darwin OS. Questo è un kernel ibrido, creato dalla fusione del kernel FreeBSD, variante originaria di Unix creata dall'università di Berkeley, e il microkernel Mach, uno dei primi microkernel creati e tra i più famosi. Apple ha modificato ed esteso questo kernel per raggiungere i livelli prestazionali di Mac OS X o di iOS. Risultano i vantaggi di Mach e di BSD in questo modo. Mach, e nella fattispecie il suo microkernel si occupa della gestione della memoria, del sistema input/output e della comunicazione tra processi. Inoltre, precisiamo che gestisce la protezione della memoria, preemptive multi-tasking e una gestione avanzata della memoria virtuale. La parte del kernel BSD invece, è responsabile degli utenti e dei permessi, al suo interno è presente lo stack di rete, offre un file system virtuale (VFS) ed è compatibile con le specifiche POSIX (IEEE 1003). I livelli di astrazione di iOS sono essenzialmente quattro:

1. **Il Core OS layer:** il quale fornisce i servizi a basso livello connessi con l'hardware e la rete. Questi servizi si basano fortemente sul layer del Kernel e dei Driver, inoltre esso si occupa anche di aspetti legati alla sicurezza delle app.

2. **Il Core Services layer:** si occupa di fornire i servizi essenziali alle applicazioni senza però averne un rapporto diretto.

3. **Il Media layer** L'accuratezza rispetto alla grafica e l'alta fedeltà rispetto la user experience sono un simbolo distintivo dei prodotti targati Apple, questo è il layer che si occupa di animazione, grafica 2D e 3D, effetti audio/video all'interno della applicazioni.

4. **Il Cocoa Touch layer:** il primo responsabile dell'aspetto delle applicazioni e della responsiveness dell'intera user experience.

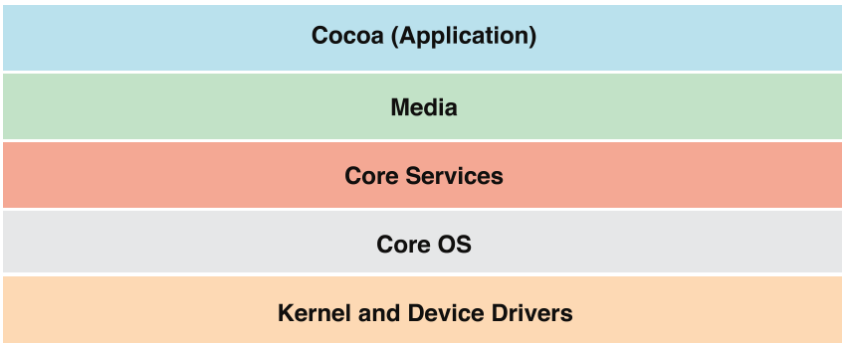

Figura 2.3: Architettura OS X

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

- le funzionalità di I/O;
- gestione dei processi;
- gestione thread;
- sqlite.

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

- Threading.

- Networking.
- Gestione file system.
- Standard input/output.
- Bonjour e sevizi DNS.
- Allocazione delle memoria.
- Computazione matematica.

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

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

2.2.2 Windows phone

Windows Phone è un sistema operativo basato sulle API di Win32 di Microsoft. Nato dalle ceneri di Windows Mobile, venne presentato al Mobile World Congress nel febbraio 2010. Ad oggi, l'ultima versione ufficiale disponibile è Windows Phone 8.1 Update (GDR1) (8.10.12293.216) rilasciata in aprile 2015. Passiamo ora alle caratteristiche tecniche. Windows Phone supporta vari dispositivi e richiede delle specifiche tecniche minime per essere messo in esecuzione. Anche questo sistema è strutturato a strati come il precedente. Windows Phone utilizza una soluzione ibrida, prendendo parte del kernel da Windows Embedded CE 6.0 R3 e in parte da Windows Embedded Compact 7. Uno dei cambiamenti fondamentali che troviamo su Window Phone è la possibilità di sfruttare appieno l'architettura ARMv7. Windows Phone è un sistema operativo Hard real time a 32 bit e modulare pensato per dispositivi embedded. Supporta il multitasking e architetture diverse quali x86, ARM, MIPS e SH4 operando su uno spazio di indirizzamento di 4 GB. Il kernel utilizza 2 GB di memoria virtuale superiore, mentre i restanti vengono impiegati per la gestione dei processi utente attivi, esso ne supporta fino a 32768. Tra i processi, gioca un ruolo centrale Nk.exe, il nucleo del sistema operativo. Al suo interno sono caricate le librerie dinamiche atte a varie funzionalità di sistema. Altro componente di rilevo è il file eseguibile Oal.exe detto OAL, questo ha il compito di far interagire il kernel (Kernel.dll) con i componenti hardware a bordo del device caricando il Device Manager (Device.dll) durante la fase di boot. Per quanto riguarda il tipo di file system, Windows Phone ne utilizza uno molto simile a quelli già visti nei sistemi UNIX, infatti hanno una root unica, mount dei dispositivi con lettere, RAM come ObjectStore, crittografia dei media removibili e gestione supporti di memorizzazione e file fino a 4 GB. Tra le caratteristiche che lo contraddistinguono dalla concorrenza possiamo trovare:

- *Silverlight*, Tecnologia di Microsoft per la visualizzazione, all'interno del browser di contenuti RIA, applicazioni multimediali ed interattive.

- *Connection Manager*, Infrastruttura tecnologica per la gestione di interfacce di rete sul dispositivo.
- *Internet Explorer integrato*, browser di casa Redmond, possibilità di multitouch, panning e zoom e un'interfaccia personalizzabile per una migliore esperienza dell'utente finale.
- *Microsoft Office e PDF Viewer*, applicazioni per visualizzare ed editare documenti, tra i quali, Microsoft Office Word, Excel, PowerPoint ed Adobe reader direttamente nel dispositivo.
- *Web Services on Devices API (WSDAPI)* , costruzione di servizi Web per la migliore interazione tra i dispositivi in rete. Con WSDAPI si possono scambiare messaggi e metadati, oltre ad un supporto per la gestione degli eventi e la sicurezza.
- *Driver*, produzione di qualità dei driver per garantire la minor quantità di modifiche possibili da approntare su hardware personalizzato; Esso contribuisce quindi a garantire una migliore portabilità del sistema.

Microsoft ha deciso di restringere il campo sulle specifiche hardware da utilizzare sui dispositivi. Questo riduce notevolmente il rischio di frammentazione hardware garantendo così che il prodotto sia perfettamente ottimizzato, e la sua esecuzione la più veloce possibile in ogni dispositivo venduto sul mercato. Per fare un'analogia con i due sistemi operativi sopraelencati, asseriamo che Microsoft si posiziona tra la vasta selezione di prodotti che Android ci offre e il controllo autoritario di Apple. Per completezza è giusto ricordare che nel febbraio 2011 Steve Ballmer, amministratore delegato di Microsoft ha annunciato, in compresenza con Stephen Elop, amministratore delegato di Nokia, la partnership delle due, in questo modo, Nokia abbandona definitivamente il sistema operativo fino ad allora utilizzato Symbian OS. La partnership tra le due grandi aziende ebbe il suo apice quando la prima acquisì la seconda per 7,1 miliardi di dollari nel 2014 inserendo il sistema operativo di casa Redmond in tutti i dispositivi mobile di Nokia. Il connubio delle due compagnie

non ebbe vita lunga, infatti nel maggio 2016 Microsoft vendette Nokia a FIH Mobile (sussidiaria della celebre Foxconn, che produce anche gli iPhone) e HMD Global (fondata da alcuni ex Nokia) ponendo fine al monopolio del sistema operativo sui prodotti della compagnia svedese.

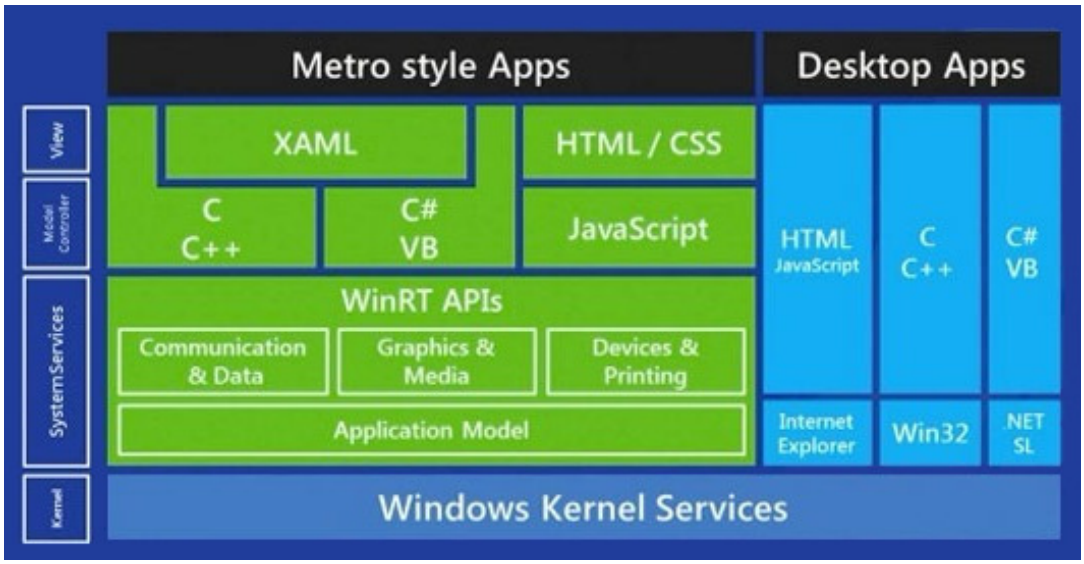

Figura 2.4: Architettura Windows 8

2.2.3 Android

Android è un sistema operativo per dispositivi mobili, inizialmente sviluppato da Android Inc. e poi acquistato nel 2005 da Google. Android, non è stato sviluppato da zero, ma si basa su diverse versioni del kernel Linux. Una delle caratteristiche che lo contraddistingue dalla concorrenza è la sua natura open source e la sua versatilità, infatti, è utilizzabile in qualsiasi dispositivo mobile. Per completezza è giusto mettere in luce il fatto che il primo smartphone ad utilizzare questa piattaforma è stato l'HTC Dream presentato il 22 ottobre 2008 con la versione Android 1.0: ad oggi, l'ultima versione presentata è la 7.1.1 (4 ottobre 2016) detta Nougat. Nello specifico sappiamo che questo sistema operativo è costituito da uno stack software (un set di sottoinsiemi software) che include il sistema operativo basilare, che si basa

sul kernel Linux, è composto da applicazioni Java che vengono eseguite da uno speciale framework, anch'esso basato sulla stessa tecnologia e orientato agli oggetti. Vengono inoltre utilizzati degli strati middleware (applicativi intermediari tra le diverse applicazioni e i componenti software) per la comunicazione con le applicazioni di base. Sopra il kernel troviamo le librerie fondamentali. Questa piattaforma utilizza una serie di librerie in linguaggio C e C++, riportiamo di seguito alcuni esempi.

- *Il leggerissimo SQLite*, una libreria software che implementa un DBMS SQL di tipo ACID incorporabile all'interno di applicazioni. Tra le sue qualità ricordiamo la leggerezza (meno di 500k tutta la libreria), la velocità, le API sono semplici da usare, è multipiattaforma e completamente libero.

- La libreria SGL, essa è utilizzata da Android come libreria per implementare la grafica bidimensionale.

- Le API OpenGL ES 2.0, sono utilizzate invece per gestire accelerazione tridimensionale della grafica.

- Media Libraries, esso è basato sulla libreria open-source OpenCore e riguarda il supporto audio e video, avendo a disposizione una serie di codec tra cui mp3, wav, avi, mp4, flv eccetera...

- WebKit, il motore grafico per un browser web utilizzato per il rendering delle pagine web. WebKit è un progetto open-source che nasce dalla combinazione di componenti del sistema grafico KDE e di tecnologie Apple.

- Surface Manager, esso gestisce la visualizzazione grafica.

- Un framework multimediale OpenCore e System C library, cioè un ottimizzazione della libreria libc (libreria standard di Linux) per sistemi mobili.

Una volta che le applicazioni sono scritte con il linguaggio Java, vengono eseguite tramite una virtual machine del tutto simile a quella che, Java utilizza per eseguire i propri programmi, ma ottimizzata per smartphone. Dapprima veniva utilizzata la cosiddetta Dalvik Virtual Machine (DVM) scritta da Dan Bornstein , un dipendente di Google. Dalla versione 4.4 KitKat è stata soppiantata da Android RunTime (ART), basato sulla tecnologia AOT(ahead-of-time) che esegue l'intera compilazione del codice durante l'installazione dell'app e non durante l'esecuzione stessa del software, vantaggioso sia dal punto di vista delle prestazioni che da quello della gestione delle risorse. Questo è uno dei componenti fondamentali di Android, infatti grazie ad un intelligente utilizzo dei registri di sistema, permette di indurre il dispositivo a minimizzare il consumo di memoria, consentendo di eseguire più istanze delle macchine virtuali contemporaneamente, nascondendo al sistema operativo la gestione dei thread e in ultima istanza quella della memoria. Attraverso un pacchetto auto-installante con estensione APK, il software viene distribuito all'utente finale. Esso non è altro che un archivio compresso, al cui interno abbiamo il software (con estensione .dex che andrà in pasto alla DVM oppure alla ART), le risorse (immagini, suoni etc...) e alcuni file XML per la gestione dell'interfaccia grafica e non solo. L'utente finale ovviamente, non è al corrente di tali informazioni, dato che il dispositivo gestisce tutto il processo di installazione autonomamente tramite un web services come l'Android Market. All'interno dell'archivio troviamo anche un certificato, il quale descrive le "impronte" dell'app, in questo modo siamo sicuri di non incorrere in un applicativo compromesso o revocato. Tuttavia questo incremento di prestazioni e gestione delle risorse in fase di esecuzione incide con un maggior tempo per l'installazione di un app. Si tratta comunque di tempi quasi impercettibili, specie in vista di hardware sempre più potenti.

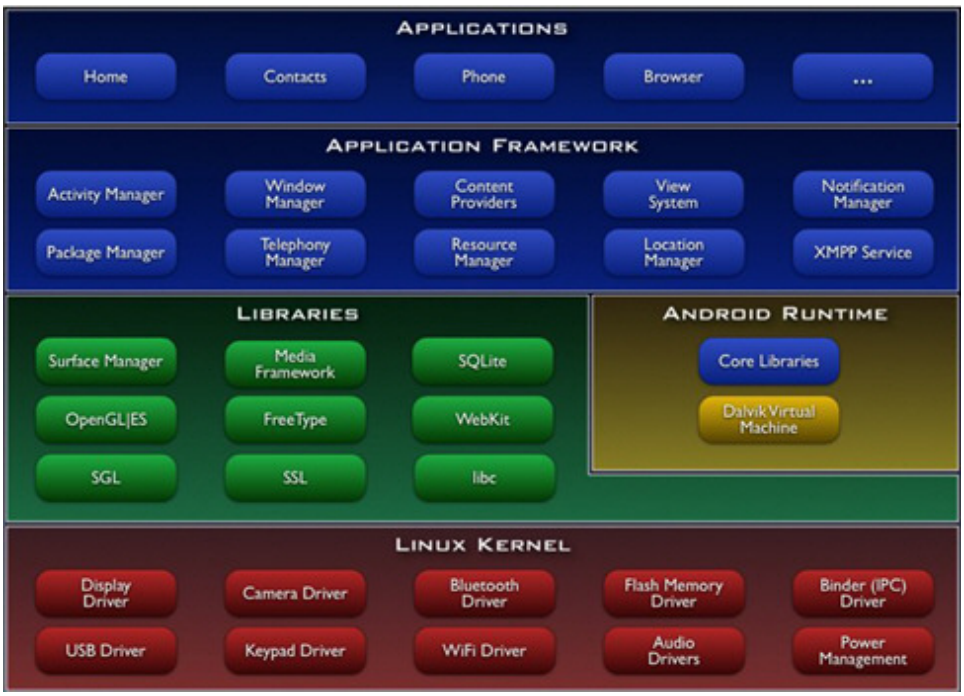

Figura 2.5: Architettura Android

2.2.4 La scelta del sistema operativo

Ora, avendo ben presente la panoramica che ci si prospetta davanti, possiamo decidere a quale sistema operativo mobile affiliarci. Per questo progetto la scelta è ricaduta sul OS Android per diversi motivi:

- **Ampia fascia di mercato,** Uno dei punti focali su cui si basa questo progetto è sicuramente la possibilità di arrivare al maggior numero di persone, quindi questa è la miglior soluzione che permetta di rispondere ad una fascia di mercato ampia. Come visibile in Tabella 2.1, tratta dallo studio condotto nel 2016 dal istituto Gartner possiamo notare come il sistema operativo Android, abbia coperto più dell'86% dei device venduti nel mondo. E' anche rilevante notare come questo andamento sia in ascesa, infatti nel 2012, Android era installato nel 64% dei device circa, nel 2013 ricopre il 79% e nel 2015 nell'80% circa.

- **è open Source,** Un altro punto a favore, che gli altri due sistemi

operativi contendenti non posseggono è l'approccio open source. Esso indica un software i cui autori permettono la visione e lo studio del codice sorgente a chiunque voglia visionarli.

Tabella 2.1: Studio Gartner su vendite O.S per smartphone nel 2015/2016

O.S.	**2Q16 Units**	**2Q16 M.S.(%)**	**2Q15 Units**	**2Q15 M.S.(%)**
Android	296,912.8	86.2	271,647.0	82.2
iOS	44,395.0	12.9	48,085.5	14.6
Windows	1,971.0	0.6	8,198.2	2.5
Blackberry	400.4	0.1	1,153.2,	0.3
Others	680.6	0.2	1,229.0	0.4
Total	**3440,359.7**	**100.0**	**330,312.9**	**100**

2.3 GlovePi: Architettura

Ora che abbiamo introdotto i vari metodi di comunicazione nel primo capitolo della relazione e analizzato propedeuticamente le nostre scelte, ponendo come modalità di comunicazione l'alfabeto Malossi (scelto precedentemente) possiamo predisporre un'architettura di sistema costruendola partendo dai suoi componenti (ampiamente discussi in fase progettuale propedeutica).

Si è quindi pensato, come visibile in Figura 2.6, di utilizzare (i) un guanto come base per poter apporre la sensoristica necessaria in modo che (ii) un'unità computazionale possa gestire i dati in arrivo ed infine inviare il risultato ad (iii) un'unità di output. Nei prossimi paragrafi presenteremo in dettaglio i diversi componenti di questo progetto.

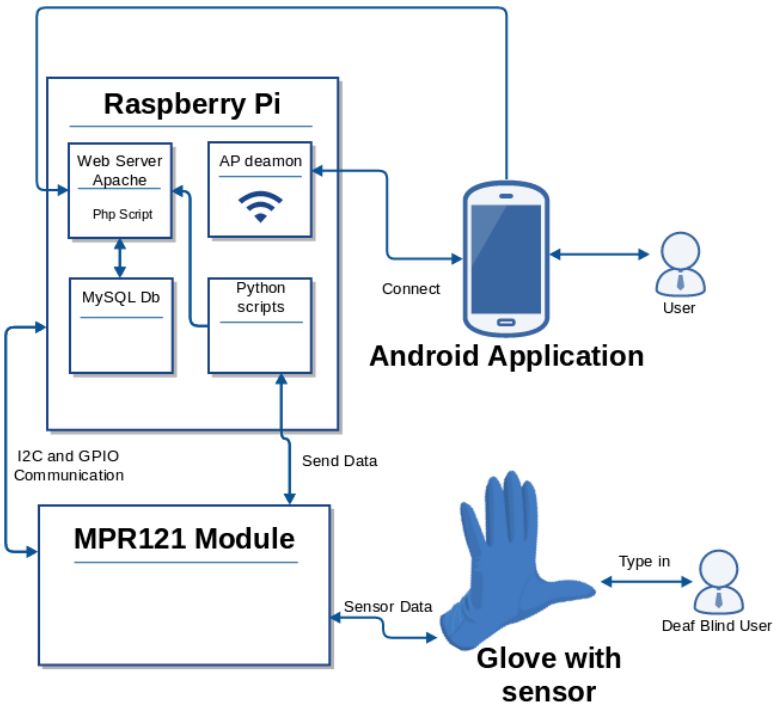

Figura 2.6: Architettura Software

2.4 Il guanto

Per la prototipazione è stato acquistato un guanto da giardiniere, perchè lo si voleva con nessuna cucitura tra le dita e il palmo. Come visibile in Figura 2.7, avendo utilizzato il metodo Malossi in questo progetto, si è deciso di mappare il guanto in due parti distinte: la parte frontale (lettere dalla A alla O) e la parte posteriore (lettere dalla P alla Z e lo spazio), poichè ognuna di esse si riferisce a una parte ben precisa che ora andremo ad affrontare:

1. **La parte frontale:** In questa sezione del guanto le lettere, nell'alfabeto preso in esame, sono premute, quindi si è pensato bene di utilizzare dei tasti, tre per ogni dito, come impone il Malossi e inoltre è stato cablato un cavo per poter distribuire la tensione ad ogni tasto da una sorgente unica (5V presi direttamente dalla Raspberry Pi, di cui parleremo più

avanti) come visibile in Figura 2.7 (in alto).

2. **La parte posteriore:** Per questa porzione invece, è stato ritagliato e cucito sul guanto la parte di velcro ad uncino, la parte ad asola è invece stata utilizzata per incollarvi del materiale conduttore (carta stagnola) così da poter cambiare rapidamente la sezione in caso di usura. Questo lavoro è stato fatto per la prima e la terza falange di ogni dito, come visibile in Figura 2.7 (in basso). Per non creare conflitti di segnali è stato poi aggiunto uno spessore tra le parti di carta stagnola avvolta in nastro isolante. I segnali vengono poi inviati alla scheda di espansione MPR121, (verrà analizzata con maggior perizia nei paragrafi successivi) posta come sovrastruttura della Raspberry Pi, carpendoli come tocco di tipo capacitivo.

In definitiva, per una miglior comprensione si vuole puntualizzare la modalità di cablaggio, stagnando nella scheda di espansione MPR121 in totale:

1. *15 tasti* per la parte frontale del guanto.
2. *12 cavetti* per touch sensor capacitivo a bordo della scheda per la parte posteriore del guanto.
3. *1 cavo di alimentazione* utile per ogni componente ambedue i lati del guanto.

Figura 2.7: Il prototipo GlovePi

2.5 Modulo MPR121

Per il naturale prosieguo del progetto è stato sicuramente utile se non indispensabile, l'utilizzo del modulo acquistato presso il sito www.adafruit.com. Attraverso ciò è possibile aggiungere alla nostra Raspberry Pi 12 sensori touch capacitivi attraverso i cosiddetti "buchi a forma di 8". È infatti possibile attanagliare ogni sezione con degli alligator, unendo una parte del morsetto al modulo e l'altra parte a qualsiasi materiale elettricamente conduttivo. Nel progetto, tale board è stata utilizzata come bridge tra la Raspberry Pi e il guanto, perchè la totalità delle connessioni del guanto sono state stagnate a questo modulo e vengono veicolate tramite lo stesso alla Raspberry Pi (anche i segnali GPIO della parte frontale del guanto vengono reindirizzati tramite il modulo essendo questo un HAT).

2.6 Raspberry Pi

Nel progetto in questione, sicuramente questo dispositivo single-board ha un ruolo di indiscussa centralità infatti, possiamo ben capire come questo sia la connessione centrale di tutti i componenti di questa architettura. È stata utilizzata come: (i) *access point* così da poter accedervi direttamente senza utilizzare il web ed aver un'utilizzabilità anche senza l'accesso ad internet; (ii) *scripting manager*, in questo modo ogni script creato viene lanciato all'avvio del sistema per poter acquisire i dati dal guanto; (iii) *server* per poter, una volta connesso al dispositivo, inviare tutti i dati all' applicativo che si occupa di generare l'output giusto (l'applicazione Android di cui ne parleremo in seguito). Per maggior portabilità del guanto, la Raspberry Pi (assieme ad un alimentatore) è stata cucita su un polsino, come visibile in Figura 2.8.

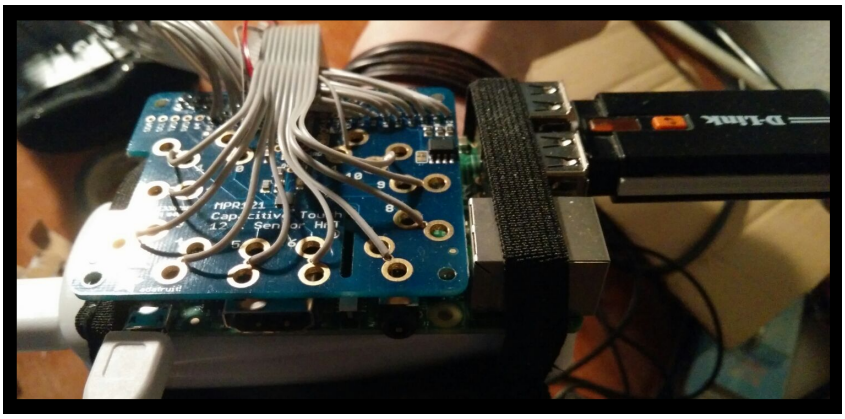

Figura 2.8: Raspberry Pi

2.6.1 RPi as an access point

Per questo progetto si è deciso di configurare la Raspberry Pi come fosse un access point per poter accedervi direttamente senza doversi connettere al Web, ma poterlo utilizzare come fosse un punto wi-fi. Per far ciò è stata scaricata una normale distruzione Raspbian (versione debian ottimizzata per Raspberry Pi) dal sito https://www.raspberrypi.org/ ed è stata configurata

ad-hoc per l'obiettivo postoci. Sono stati installati diversi pacchetti che vengono attivati come demoni all'avvio del dispositivo. I principali pacchetti sono i seguenti:

1. **hostapd:** Questo è un demone che, dato il modulo wireless utilizzato sulla Raspberry Pi, permette di creare un wireless access point e genera anche un Server di autenticazione nel caso servisse, infatti implementa il protocollo IEEE 802.11 per la gestione del access point, autenticazione IEEE 802.1X/WPA/WPA2/EAP IEEE 802.11 client RADIUS, EAP server e autenticazione RADIUS server.

2. **udhcpd:** Questo demone, invece essendo un server DHCP, si occupa di assegnare in automaticamente l'indirizzo IP ai client che accedono alla rete. Si è scelto di utilizzare questa soluzione poichè più leggera rispetto ad altre (le altre soluzioni testate sono visionabili attraverso le seguenti reference [50, 51] anche se meno agevole dal punto di vista della configurazione completa, poichè sicuramente una delle criticità maggiori è quella delle performance date le prestazioni dell'hardware del device su cui ci si basa.

2.6.2 RPi as a scripting manager

In questa porzione di progetto ci occupiamo di acquisizione dei dati dal guanto per poterli salvare in locale così ed utilizzarli alla richiesta del client. Per far ciò sono stati creati diversi script, alcuni in Python ed altri in Bash. Uno di questi (start.sh) è attivato, dopo l'avvio di tutti i demoni che servono al sistema operativo per funzionare, ed è adibito all'avvio di tutti gli altri demoni che si occuperanno in primis di eseguire un controllo (initGlovePi.sh è lo script candidato) rispetto tutti i servizi in background che servono al nostro progetto, in caso di successo, che verrà notificato attraverso un segnale audio, verrà attivato lo script (glovePi.py) per "mettersi in ascolto" rispetto ai cambiamenti di tensione sul guanto. Una volta carpito un segnale, immediatamente verrà salvato in locale.

2.6.3 RPi as a server

Altra sezione di rilievo per questo progetto è sicuramente la parte server. È stato creato per poter inviare i dati all'applicativo Android, infatti quando stimolato (in polling da parte del client) il server controlla se ci sono nuove lettere e se l'esito è positivo le invia al client che lo ha richiesto.

2.7 Applicazione Android

Dopo aver generato, computato e immagazzinato i dati con la Raspberry Pi abbiamo la necessità di ricevere la totalità di essi in maniera semplice ed intuitiva. Per questo motivo è stata creata una applicazione Android. Una volta connessi alla Raspberry Pi (come già spiegato nel paragrafo RPi as an access point) possiamo accedere alla nostra applicazione e dopo qualche secondo di presentazione (splashActivity) come visibile in Figura 2.9 (sinistra), ci appare un Activity, visibile in Figura 2.9 (destra) che ci permette di decidere come impostare le richieste di invio delle lettere dalla Raspberry Pi (Manuale oppure Polling) e se resettare tutto ciò che è stato fatto con il guanto fino ad ora. Inoltre è presente una textArea non editabile che avrà la funzione di visualizzazione delle lettere "scritte" con il guanto. Infine attraverso il tasto PARLA potremmo ascoltare ciò che è arrivato tramite il server e palesato nella textArea immediatamente sopra ad esso.

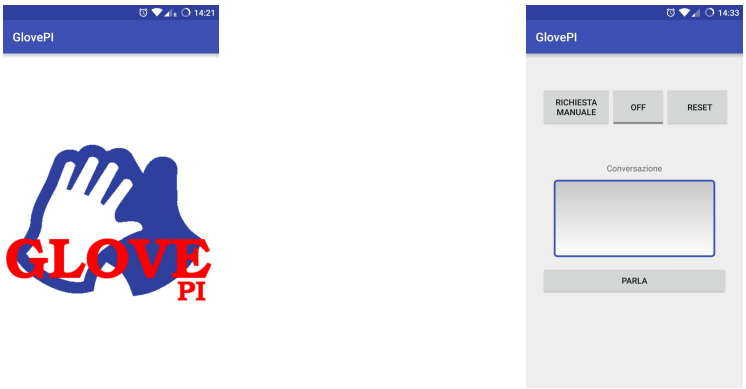

Figura 2.9: Applicazione Android

2.8 Comunicazione many-to-many

Nel capitolo precedente sono state analizzate le tecnologie di riferimento che ad oggi sono state pubblicate e/o messe in commercio. Le tecnologie sono molteplici e coprono molte sfaccettature del mondo della comunicazione tra persone sordocieche, ma non in maniera completamente esaustiva. In ambito distribuito ad esempio vi è una mancanza che questa relazione cercherà di sopperire almeno in parte. Di seguito verrà presentato uno scenario di riferimento e le problematiche relative all'identità in ambito distribuito per le persone sordo-cieche.

Sappiamo infatti come anche il movimento sia una problematica di rilievo ed allo stesso tempo come la comunicazione sociale sia un desiderio forte e marcato. Per questo nasce spontanea l'esigenza di estendere le idee che costituiscono i progetti dello stato dell'arte e spostare il contesto in ambito distribuito per gestirne nativamente le comunicazioni one-to-many e many-to-many. Potremmo prendere spunto dall'ipotesi di scenario visibile in Figura 2.10 che sarà sorretta da un'infrastruttura client-server per poter comunicare e coordinare le interazioni tra più persone. Nella fattispecie si assumerà concettualmente un tipico esempio di comunicazione many-to-many, *la conferenza* che nella fattispecie sarà una *conferenza aptica*. Peraltro vi è un corrispettivo reale poichè le conferenze tra/per persone sordocieche esistono anche se non in maniera massiccia. Esse sono gestite in maniera diversa sulla base della tipologia e gravità della patologia, alcuni utilizzano barre braille, altre persone formano un gruppo con un operatore, e alcuni, con patologie più gravi, sono aiutati e seguiti individualmente da un operatore.

Si vuole inoltre mettere in evidenza la problematica dell'identità in contesto distribuito, infatti concetti come:

- il **numero delle persone** attorno all'utente;
- la **posizione delle persone** e dove è posta l'attenzione in un dato momento;
- **l'identità e l'aspetto delle persone** davanti all'utente;

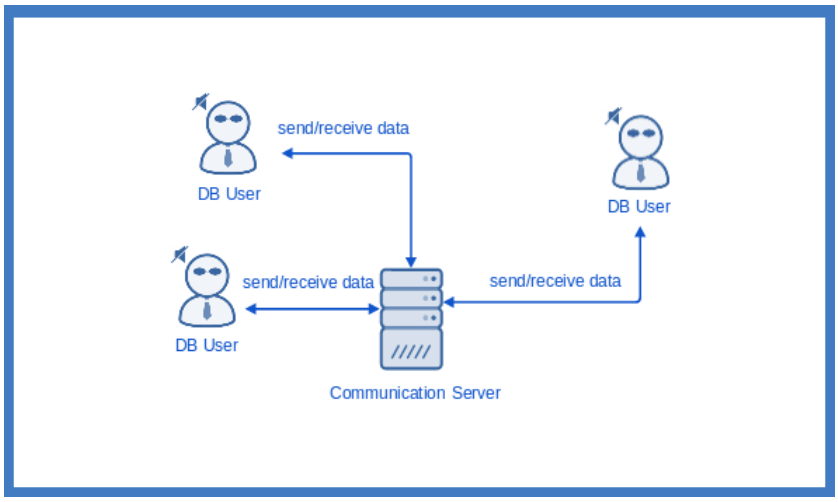

Figura 2.10: Architettura dello scenario

- **l'espressione facciale** e gli spunti non verbali;
- capire se la persona davanti è **conosciuta**.

Questi sono concetti facilmente astraibili per persone senza alcuna disabilità, ma difficilmente acquisibili per una persona sordo-cieca (se non attraverso un operatore). In molti casi la possibilità di interazione è dettata dalla stretta relazione che lega i due individui, come ad esempio tra madre e figlio o comunque parenti stretti oppure persone a stretto contatto. Di conseguenza lo scenario di riferimento rende ancor più ardua la soluzione. Il contesto infatti, assume implicitamente l'interazione sociale tra persone che non hanno un rapporto stretto.

2.9 Architettura del sistema many-to-many

Dato lo scenario di riferimento della sezione precedente possiamo ora definire un'architettura per la comunucazione aptica tra persone sordo-cieche in ambito distribuito ponendo innanzitutto sotto esame le componenti chiave di quest'ultima. Appoggiandoci al guanto progetto per questa tesi abbiamo quindi un prototipo per la comunicazione one-way tra persone sordo-cieche

e normodotate, in questo caso non vi è un feedback tattile di ritorno per le persone sorco-cieche anche se presupposto negli sviluppi futuri. Un ulteriore paper ci viene in aiuto, infatti come descritto in [46] possiamo leggere di come l'aggiunta di sensori aptici non sia uno scoglio nè progettuale nè implementativo. Essendo in un costesto distribuito sicuramente avremo bisogno anche di un *client* per poter comunicare con il mondo esterno, la Raspberry Pi utilizzata nel progetto di riferimento fà al caso nostro. In questo modo potremo comunicare inviando e ricevendo i messaggi interessanti. Come visibile in Figura 2.11 si è pensato ad una soluzione ibrida tra il paradigma *client-server* e paradigma *peer-to-peer*. La descrizione dell'architettura di sistema presa in esame avverrà focalizzando l'attenzione su tre punti di vista:

- **Struttura:** l'organizzazione del sistema in parti;
- **Comportamento:** il funzionamento del sistema e di ogni singola parte;
- **Interazione:** il modo in cui le diverse parti scambiano informazione implicita o esplicita tra loro.

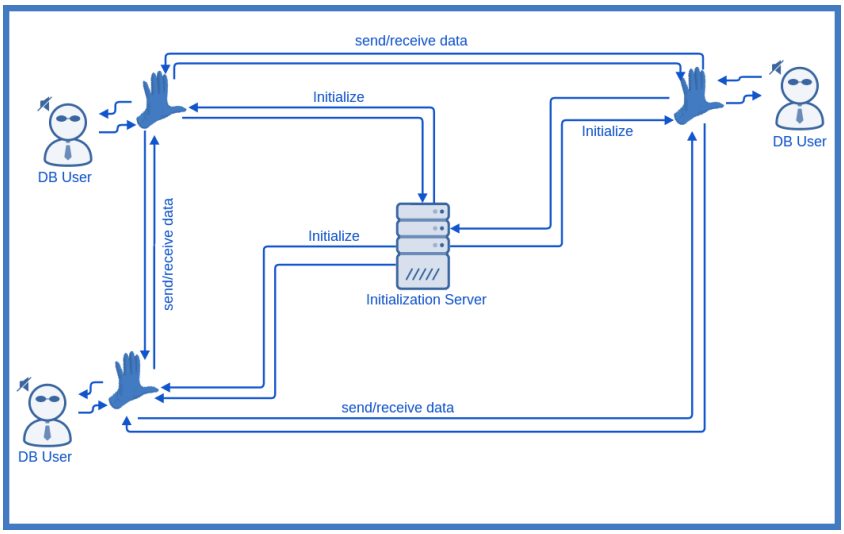

Figura 2.11: Architettura del sistema

2.9.1 Struttura

All'interno del modello, a livello tassonomico, avremo ai lati gli utenti sordo-ciechi che indossano il guanto, il quale, attraverso l'utilizzo di una Raspberry Pi, tradurrà i segnali tattili in stringhe codificate e fungerà da *client* per poter comunicare inizialmente con il *server*. Avremo un server per una prima fase di inizializzazione, con il quale ogni *client* si potrà registrare ed avere una lista completa degli altri client per connettersi reciprocamente in maniera *peer-to-peer*. Infine, sarà la Raspberry Pi che si occuperà di ritradurre le stringhe acquisite in segnali aptici.

2.9.2 Comportamento

Anche attraverso questa dimensione analizzeremo le componenti del sistema definendo i comportamenti del *server* e i comportamenti dei *client* come segue:

1. **Comportamento Server:** Il server è l'entità cardine per le primissimefasi di configurazione della rete e per l'aggiornamento della stessa. Esso è stato modellato come segue:

    - *Registrazione client:* questa è la primissima fase, nella quale il client,incapsulato all'interno di uno script Bash nella Raspberry Pi, esegue una richiesta HTTP al server per entrare a far parte della rete e rendersi noto agli altri client interessati al topic generando l'effettiva registrazione di quest'ultimo. Se andrà a buon fine il server ritornerà una notifica HTTP al client con un messaggio di successo.
    - *Inoltro della lista dei nodi:* uno dei compiti importanti che il server possiede è quello di tener traccia dei nodi della rete rispetto ad un determinato topic. Infatti il secondo passo, dopo quello della registrazione è appunto l'invio della lista completa dei client interessati al topic a cui esso si è iscritto.

- *Aggiornamento client:* il server, in maniera pro-attiva invierà, tramite protocollo HTTP, aggiornamenti delle lista completa agli utenti registrati per mantener loro aggiornati sullo stato e la topologia dellarete.

2. **Comportamento Client:** il client è una componente omnipresente in questa architettura ibrida, poichè interagisce sia con il server che con gli altri client della rete. Il comportamento è modellato come segue:

   - **Registrazione al server:** come già descritto nella sezione precendente, questa è l'operazione duale a quella del server. In questa fase, attraverso la Raspberry Pi, il client notifica al server la sua volontà di seguire un determinato topic notificando così la propria presenza a tutto lo sciame. Inoltre sempre in questa fase viene fornita una lista completa dei nodi della rete.
   - **Comunicazione peer-to-peer:** a questo punto la comunicazione avviene tra pari, quindi ogni client diverrà un nodo e quest'ultimo potrà richiedere o fornire uno oppure un insieme di messaggi d'interesse.
   - **Aggiornamento lista nodi:** Ogniqualvolta si registri un nuovo client al server, esso invierà tramite il protocollo HTTP la parte di lista mancante ai nodi già registrati per mantenerli aggiornati sullo stato della rete.

2.9.3 Interazione

In questa sezione analizzeremo l'interazione delle singole entità di questa architettura. Vi sono sostanzialmente due entità principali i client e il server, che si distinguono sia per ruolo che per paradigma utilizzato durante l'interazione con le altre entità.

1. **Interazione client-server:** permette di avere una lista completa dei nodi interessati ad un determinato topic, questi comporranno la rete

peer-to-peer. Essi si divideranno in "Nodi Fornitori", nel caso inviassero i messaggi, e "Nodi Riceventi" nel caso volessero leggere il messaggio. Inoltre, in questo modo il client potrà render nota la propria presenza attraverso una fase di inizializzazione dove effettivamente avverrà la registrazione nella lista dei nodi sul server (dichiarando la tipologia di nodo tra le possibili). Di conseguenza, il server invierà ad ogni client questa lista di nodi (nel caso questa venga aggiornata) perchè ogni nodo possa rimanere aggiornato sulla configurazione della rete senza minare la sua consistenza.

2. **Interazione tra client (P2P):** Una volta ultimata la fase di inizializzazione e reperita la lista dei nodi interessati, è possibile comunicare tra pari mettendo concettualmente in qualche modo da parte il server (anche se in realtà esso contatterà regolarmente i nodi registrati, come descritto sopra). Ogni client quindi sarà un nodo della rete che potrà comunicare con gli altri al fine di veicolare i messaggi interessanti.

2.10 Architettura del sistema con TuCSoN

Si è deciso di estendere l'architettura di sistema (pienamente funzionante in termini di progettazione) attraverso l'utilizzo di un middleware per migliorare la coordinazione tra i nodi e rendere più agevole la comunicazione. TuCSoN è stata la risposta, esso è un modello di coordinazione di processi distribuiti per agenti autonomi. Permette quindi agli agenti di interagire con il centro di tuple eseguendo le operazioni di coordinazione tramite primitive di coordinazione, come leggere, scrivere oppure consumare tuple. è stata quindi ripensata l'architettura lasciando praticamente inalterato il *server* ma modificando i *client* in favore di entità attive chiamate agenti e aggiungendo ad ognuno di essi un proprio *centro di tuple* come visibile in Figura 2.12. Come nel paragrafo precendente l'architettura verrà studiata attraverso le consuete tre dimensioni:

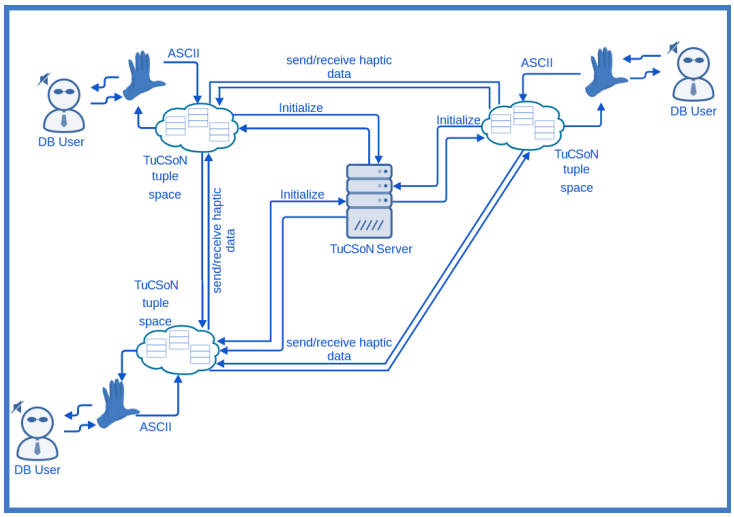

Figura 2.12: Architettura TuCSoN

2.10.1 Struttura

La struttura sarà composta da utenti affetti da sordocecità che vestiranno un guanto, il quale verrà gestito interamente tramite una Raspberry Pi traducendo i segnali aptici in stringhe codificate pronte per l'invio nella rete. In questo caso però, avremo un *agente* che si occuperà in primis di comunicare al *server* la sua presenza e richiedere la lista completa ed aggiornata dei nodi della rete per poter comunicare con essi. La comunicazione rimarrà *peer-to-peer* ed avverrà tramite l'interazione di *spazi di tuple* attraverso ad alcune primitive che il linguaggio TuCSoN ci offre. L'agente infine farà da ponte tra lo spazio di tuple, dove i messaggi sono situati e la Raspberry Pi che si occuperà di tradurre le stringhe in segnali aptici per l'utente.

2.10.2 Comportamento

1. **Comportamento Server:** in questa architettura il comportamento saràdel tutto simile a quello proposto nel paragrafo precedente. Gestirà infattila fase registrazione e aggiornamento dei nodi della rete. Ecco come è stato modellato:

- *Registrazione agente:* fase iniziale nella quale, in maniera identica alla precedente architettura, viene registrato un agente per rendere la sua identità pubblica. Un agente posto come demone all'interno della Raspberry Pi che monitora il guanto invierà una richiesta tramite le primitive TuCSoN al server per registrarsi, quest'ultimo poi salverà nella lista dei nodi nel proprio centro di tuple di un determinato topic. Se andrà a buon fine il server interagirà con il client tramite lo spazio di tuple con un messaggio di successo altrimenti verrà notificato l'errore.

- *Inoltro della lista dei nodi:* compito essenziale al fine di permettere la comunicazione tra nodi per poter veicolare i messaggi tramite gli spazi di tuple. Il server invierà quindi la lista completa dei nodi ad ogni agente una volta che avverrà la registrazione. In seguito ognuno di essi potrà connettersi ad altri agenti interessati ad un determinato topic.

- *Aggiornamento agente:* il server notificherà le novità agli agenti già registrati se vi è un aggiornamento di stato nello spazio di tuple di un determinato topic per mantener loro aggiornati sullo stato e la topologia della rete.

2. **Comportamento Agente:** l'agente interagisce sia con il server che con gli altri agenti della rete oltre che con il proprio centro di tuple. Il comportamento è modellato come segue:

   - *Registrazione al server:* come già descritto nella sezione precendente, questa è l'operazione duale a quella del server. In questa fase verrà posto un agente come demone attivo all'interno della Raspberry Pi che gestisce il guanto, il quale notificherà al server la sua volontà di seguire un determinato topic e registrandosi nella lista dei nodi per tutto lo sciame.

- *Comunicazione peer-to-peer:* una volta che l'agente è registrato e possiede la lista dei nodi interessati può collegarsi utilizzando la comunicazione tra pari.
- *Coordinamento tra spazi di tuple:* una volta stabilità la connessione peer-to-peer, potrà avvenire l'effettivo inoltro dei messaggi interessanti attraverso il coordinamento degli spazi di tuple associati ad ogni nodo di questo sciame. Ciò è possibile grazie a TuCSoN, infatti abbiamo primitive native per la comunicazione diretta tra spazi di tuple. In questo modo l'agente dovrà solamente leggere dal proprio spazio di tuple, attraverso una primitiva del linguaggio, nel caso fosse un *"Nodo Ricevente"*, oppure potrà, sempre attraverso primitive messe a disposizione nativamente, scrivere sullo spazio di tuple, se quest'ultimo è un *"Nodo Fornitore"*.
- *Aggiornamento lista nodi:* Il TuCSoN server si occuperà di notificare ad ogni nodo l'avvenuto aggiornamento della lista tramite l'interazione con gli spazi di tupla. Ogni nodo, in questo modo sarà sempre aggiornato sullo stato della rete.

3. **Comportamento Spazio di tuple:** in questa architettura lo spazio di tuple è la vera novità, essa si pone come punto di congiunzione per lo scambio dei messaggi e per il coordinamento della lettura e la scrittura da parte dell'agente. In questa fase di progettazione tale entità è stata modellata come segue:

   - *Coordinamento con agente:* al di sotto del layer Raspberry Pi vi è posto un agente che comunica con questo spazio di tuple. Sulla base della tipologia di agente, questo, inserirà tramite una primitiva, il messaggio oppure potrà leggerlo.
   - *Interazione con spazi di tuple:* i vari spazi di tuple potranno quindi comunicare tra loro al fine di inoltrare i messaggi nella rete peer-to-peer creata. L'interazione è completamente gestita da primitive del linguaggio che abbiamo scelto come ipotesi tecnologica.

2.10.3 Interazione

L'interazione verrà analizzata rispetto le entità chiave dell'architettura presa in esame. Vi sono sostanzialmente tre entità principali: il *server*, *gli agenti* e *lo spazio di tuple*. Essi si distinguono per ruolo assunto.

1. **Interazione agente-server:** questa interazione permette di ricevere una lista completa di nodi interessati ad un determinato topic, in definitiva questi diverrano poi lo sciame della rete peer-to-peer. Come nell'architettura vista in precedenza anche quì verranno suddivisi i nodi in: *"Nodi Fornitori"*, nel caso inviassero i messaggi, e *"Nodi Riceventi"* nel caso volessero leggere il messaggio. Inoltre, l'agente potrà pubblicizzare la propria identità attraverso una fase di inizializzazione dove effettivamente avverrà la registrazione nella lista dei nodi sul server (dichiarando la tipologia di nodo tra le possibili). Di conseguenza, ogni nodo verrà informato dell'aggiornamento della lista dei nodi da parte del server (in maniera pro-attiva) attraverso le primitive TuCSoN. Tutto questo fa sì che ogni nodo rimanga aggiornato sulla configurazione della rete senza minare la consistenza.

2. **Interazione tra agenti (P2P):** in questo caso, a dispetto della precedente architettura, l'interazione tra agenti collegherà i vari nodi per formare una rete peer-to-peer, ma non per inoltrare i messaggi tra loro. Il collegamente avverrà grazie al possesso della lista aggiornata dei nodi interessati avuto dall'interazione tra *agente* e *server*.

3. **Interazione tra spazi di tuple (P2P):** è tramite questa interazione che avviene realmente lo scambio di messaggi, infatti sono gli spazi di tuple che comunicano tra loro, attraverso costrutti nativi del linguaggio preso come riferimento. Nella fattispecie un nodo che è definito *"Nodo Ricevente"* comunicherà la sua volontà di possedere quel determinato messaggio e qualsiasi *"Nodo Fornitore"* che ne è in possesso cercherà di fornirglielo.

Capitolo 3

Implementazione

Con questo capitolo possiamo finalmente descrivere in maniera più specifica i dettagli di questo progetto, analizzando con perizia le diverse parti: presentiamo qui di seguito l'implementazione dei principali componenti che costituiscono questo progetto.

3.1 Modulo MPR121

Come già espresso nel capitolo precedente, questo è il modulo che fa da ponte tra il guanto e la Raspberry Pi, infatti, da una parte abbiamo tutti i cavi (è stato utilizzato un bus dati cannibalizzato da un vecchio pc fisso) che arrivano dal guanto e che sono stati saldati alla scheda in questione, e dall'altra parte, per connettersi alla Raspberry Pi abbiamo un cosiddetto pin header. Qui di seguito riportiamo la configurazione impostata per poter utilizzare la scheda di espansione con la Raspberry Pi. Il modulo sfrutta il sistema di comunicazione seriale I2C, che nella distribuzione in uso in questo progetto (Raspbian) non è attiva e configurata di default. La configurazione segue i seguenti passi: Innanzitutto connettiamo i pin:

- *Raspberry Pi 3.3V - MPR121 VIN.*
- *Raspberry Pi GND - MPR121 GND.*

- *Raspberry Pi SCL - MPR121 SCL.*
- *Raspberry Pi SDA - MPR121 SDA.*

Ora dobbiamo installare il supporto I2C per i processori ARM e i kernel linux. Questo lo facciamo tramite la seguente procedura da terminale:

Codice 3.1: Configurazione iniziale Raspberry Pi

```
sudo raspi-config
```

Abilitiamo l'avvio automatico del modulo I2C nel kernel all'avvio della Raspberry Pi e seguendo la procedura step-by-step concluderemo la nostra configurazione. Una volta riavviato il dispositivo potremo testare se è funzionante tramite il comando bash:

Codice 3.2: Modulo I2C

```
sudo i2cdetect -y 1
```

In questo modo, se tutto è configurato in maniera esatta riusciremo a visualizzare, tramite il comando gli indirizzi I2C utilizzati. Una volta configurata la scheda di espansione possiamo utilizzare le librerie python MPR121 offerte da Adafruit.

3.2 Raspberry

3.2.1 RPi as an access point

Spiegheremo qui di seguito i vari step per rendere la nostra Raspberry Pi un access point. É stato utilizzato come modulo wi-fi un D-Link DWA-140 Rangebooster N Adapter, il quale contiene un chipset Ralink RT2870.

1. Abbiamo installato i pacchetti *hostapd* e *udhcpd*, come già descritti nel capitolo precedente, essi sono atti a creare un sistema access point completo (connessione, autenticazione e assegnazione IP).

2. E' stato editato il file /etc/udhcpd.conf per configurare il pacchetto udhcpd:

Codice 3.3: Configurazione udhcpd

```
start 192.168.42.2
end 192.168.42.20
interface wlan0
opt dns 8.8.8.8 4.2.2.2
opt subnet 255.255.255.0
opt router 192.168.42.1
opt lease 864000
```

3. Assegniamo un IP statico su cui possiamo accedere (ci servirà per poter avere accesso al server interno alla Raspberry Pi con l'applicazione Android):

Codice 3.4: Assegnamento IP statico

```
sudo ifconfig wlan0 192.168.42.1
```

4. Configuriamo ora /etc/hostapd/hostapd.conf, il file di configurazione di pacchetto hostapd:

Codice 3.5: Configurazione hostapd

```
driver=n180211
ssid=Rpi_1
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=guantoraspberry2015
```

```
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

5. Il prossimo passo è quello di configurare il NAT così da poter editare il file /etc/sysctl.conf inserendo in fondo a quest'ultimo la seguente riga di codice:

Codice 3.6: Configurazione NAT: ip forward

```
net.ipv4.ip_forward=1
```

Oltre all'ip forward modifichiamo le *iptable* in questo modo:

Codice 3.7: Configurazione NAT: iptable

```
sudo iptables -t nat -A POSTROUTING -o
eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m
state --state
RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o
eth0 -j ACCEPT
```

Infine facciamo partire i service per poterli utilizzare:

Codice 3.8: Lancio dei servizi in background

```
sudo service hostapd start
sudo service udhcpd start
```

3.2.2 RPi as a scripting manager

In questo paragrafo spiegheremo in dettaglio gli script che sono stati creati per la gestione dei dati in arrivo dal guanto e il salvataggio degli stessi in attesa di essere inviati al client richiedente (vedesi RPi as a server).

1. *start.sh*: Quando viene avviata la Raspberry Pi, dopo la fase di boot dove tutti i processi demone vengono avviati, viene fatto partire in maniera automatica l'X Server e di conseguenza la X session (utilizzata per far coesistere gli script in foreground e i demoni che ci servono per la sezione server e quella router), ed è qui che facciamo partire in maniera automatica il nostro script di avvio, aggiungendo il medesimo in coda al file /etc/xdg/lxsession/LXDE/autostart. Andremo ora ad analizzarlo:

   Codice 3.9: Script autostart.sh

   ```
   #!/bin/bash
   sudo ifconfig wlan0 192.168.42.1
   echo 'Your static IP: 192.168.42.1'
   sudo lxterminal -e bash
   /home/pi/Desktop/glovePi/initGlovePi.sh
   ```

   Questo semplice script in primo luogo assegna un IP statico alla nostra Raspberry Pi, così da avere la certezza di accedervi quando si utilizzerà l'applicazione Android, e in seguito viene aperto un terminale e avviato uno script Bash per l'inizializzazione delle procedure di gestione del guanto.

2. *initGlovePi.sh*: Questo script guarda, in primo luogo, se i demoni che servono al funzionamento del progetto sono attivi (apache2, hostapd, udhcpd) e in caso negativo li riavvia, poi una volta effettuato il check viene fatto partire un audio (glovePi.wav) che ci assicura che possiamo iniziare ad utilizzare il guanto ed infine viene fatto partire lo script Python che finalmente gestirà il guanto.

3. *glovePi.sh*: Una volta effettuata tutta la procedura di init possiamo controllare (in polling) realmente il guanto; Questo è ciò di cui si occupa lo script Python glovePi.sh. Una volta importate tutte le librerie che servono, inizializziamo i pin I2C e GPIO per essere recettivi in questo modo:

Codice 3.10: INIT GPIO e I2C

```
#INIT GPIO (Fronte del guanto)
GPIO.setmode(GPIO.BCM)
mode = GPIO.getmode()
GPIO.setwarnings(False)
channel_list = [17,18,27,22,23,24,
25,5,6,12,13,16,19,20,21]
keyboardGPIO = {17:'o',18:'e',27:'j',22:'d',23:'i',
24:'n',25:'h',5:'c',6:'m,12:'l',13:'g',
16:'b',19:'k',20:'a',21:'f'}

#INIT I2C (Retro del guanto)
keyboardI2C ={0:'w',11:' ',1:'p',2:'u',3:'q',
4:6:'r',7:'x',8:'s',9:'y5:'t',10:'z'}
GPIO.setup(channel_list,GPIO.OUT,initial=GPIO.LOW)
cap = MPR121.MPR121()
if not cap.begin(): print 'Error on MPR121!'
```

Una volta conclusa la fase di inizializzazione in polling analizziamo se ci sono cambiamenti in atto ed eventualmente salvarli in un file così da poterlo utilizzare nel caso venga fatta una richiesta al server:

Codice 3.11: Script glovePi.py

```
while True:
    current_touched = cap.touched()
    for i in range(12):
        pin_bit = 1 << i
        if current_touched & pin_bit and
        not last_touched & pin_bit:
            print '{0}
            touched!'.format(keyboardI2C[i])
```

```
        pinI2C = str(keyboardI2C[i])
        myfile = open('/var/www/Server/phrase.txt','a')
        myfile.write(pinI2C)
        myfile.close()

        if not current_touched & pin_bit and
        last_touched & pin_bit:
          print '{0} released!'.format(keyboardI2C[i])
        last_touched = current_touched

        for i in range(len(channel_list)):
            if GPIO.input(channel_list[i]):
              print(keyboardGPIO[channel_list[i]])
              pinGPIO =
              str(keyboardGPIO[channel_list[i]])
              myfile =
              open('/var/www/Server/phrase.txt','a')
            myfile.write(pinGPIO)
            myfile.close()
    time.sleep(0.2)
```

Per ogni tipologia di pin è stato creato un ciclo a parte, inoltre al termine dei due cicli viene fatta una pausa di 0,2 secondi per poter dare una cadenza regolare al polling.

3.2.3 RPi as a server

Qui invece è stata creata la sezione Server per poter inviare le frasi salvate all'applicativo mobile Android qualora vengano richieste. Per far ciò si è deciso di utilizzare Apache2 come HTTP Server e PHP come linguaggio lato server, e di non usare un database Mysql ma bensì un più semplice file di

testo per non appesantire i già tanti processi sfruttati dalla Raspberry Pi. Seguirà un estratto del codice:

Codice 3.12: inserimento dati nel file phrase.txt

```
$readFile = fopen($timestampFile, "rb")
or die("Unable to open file!");
$timestamp = fread($readFile,filesize($timestampFile));
fclose($readFile);
```

Viene in queste righe letto un timestamp da un file, questo ci servirà per capire se sono state aggiunte altre parole dall'ultima volta che è stata fatta una richiesta al server da parte dell'applicativo Android, come vedremo nei prossimi estratti. Innanzitutto controlliamo che ci sia una variabile tag nel momento in cui il client invia una richiesta POST al nostro server, e che questa non sia vuota.

Codice 3.13: richieste dal client

```
if (isset($_POST['tag']) && $_POST['tag'] != '')
```

una volta assicuratoci che esiste il tag, andiamo a verificare che sia uno dei consentiti:

Codice 3.14: identificativo della richiesta JSON

```
if($tag == 'phrase')
```

Se il tag è 'phrase' allora possiamo leggere il timestamp e se questo è stato modificato leggiamo il file e inviamo al richiedente un JSON file, altrimenti l'invio sarà fatto con una serie speciale di caratteri all'interno del file JSON del tipo: ##NoModified## che dovranno essere gestiti dal client.

Codice 3.15: messaggio JSON

```
//Messaggio JSON
$response["success"] = 1
```

```
$response["error"] = 0
$response["sygnal"] = $jsonFile;
echo json_encode($response);
```

Se invece il messaggio è del tipo:

Codice 3.16: messaggio exit

```
if($tag == 'exit')
```

Allora verranno cancellate tutte le lettere nel file phrase.txt e re-inizializzato a 0 il timestamp. Questo tag è utile quando viene chiusa l'applicazione Android o quando viene espressamente richiesto.

3.3 L'applicazione Android

L'applicazione Android è la parte Client di questo progetto, infatti in questo caso l'app richiede, dopo essersi connessa alla Raspberry Pi, se ci sono nuove lettere per poterle visualizzare oppure no. Questo può essere fatto manualmente (richiesta effettuata ad ogni click del bottone) oppure in polling (attraverso un toggle button che attiva/disattiva il polling). Inoltre si può richiedere un reset così da poter ripartire dall'inizio. Il codice si divide in due Activity principali e classi ausiliarie di utilità per fare il parsing JSON piuttosto che controllare se si è connessi alla rete wi-fi e se questa è quella giusta. Andremo di seguito ad analizzare l'applicativo incominciando dalle due Activity:

1. *SplashActivity.java*: questa è una semplice Activity informativa che si occupa di mostrare il logo del progetto per circa un secondo prima di attivare l'Activity DashBoardActivity.java attraverso un intent. E' giusto precisare che essa ha uno scopo puramente estetico.

2. *DashBoardActivity.java*: In questa Activity possiamo interagire realmente con la Raspberry Pi, infatti una volta connessi, abbiamo la

possibilità di richiedere le lettere immagazzinate. Possiamo, come già introdotto,farlo in due modi:

- In manuale: è stato istanziato un listener per il bottone, cosicché quando viene premuto invio un richiesta con tag='phrase':

Codice 3.17: invio richiesta per phrase

```
json_send1=userFunctions.sendSygnal("phrase");
```

e aspetto la risposta da parte del server:

Codice 3.18: risposta del server per la richiesta phrase

```
try {
  if(json_send1.getString(KEY_SUCCESS)!=null){
      ErrorMsg.setText("");
      String res = json_send1
      .getString(KEY_SUCCESS);
      phrase = json_send1.getString(SYGNAL);

      if(phrase == "false")
        phraseArea.setText("");
      else
        phraseArea.setText(phrase);

      if (Integer.parseInt(res) != 1)
        ErrorMsg.setText("Error on sending");
   ...
```

- In polling: In questo caso, quando viene attivato il toggle button, è richiamato un metodo: callAsynchronousTask(), il quale attiva un oggetto asyncTask così da poter continuare ad avere un'applicazione reattiva anche se impegnato con il polling, infatti

ad esempio è possibile in questo modo ascoltare le frasi che sono arrivate anche se è attivo l'asyncTask. Ma vediamo in dettaglio:

Codice 3.19: AsynchTask: doInBackground

```
JSONObject doInBackground(Void... Params) {
 jsOb = userFunctions.sendSygnal("phrase");
 return jsOb;
}
```

Questa è la porzione di codice che invia una richiesta al server e deve essere rigorosamente fatta in background. Mentre la parte di interfaccia grafica viene aggiornata tramite il metodo onPostExecute, in questo modo:

Codice 3.20: AsynchTask: onPostExecute

```
void onPostExecute(JSONObject result){
  try {
    if (jsOb.getString(KEY_SUCCESS) != null) {
      ErrorMsg.setText("");
      String res = jsOb.getString(KEY_SUCCESS);
      phrase = jsOb.getString(SYGNAL);
        if(phrase == "false")
           phraseArea.setText("");
        else
           phraseArea.setText(phrase);
        if (Integer.parseInt(res) != 1) {
           ErrorMsg.setText("Invio errato");
        else
           ErrorMsg.setText("Errore!");
        } catch (JSONException e) {
                e.printStackTrace();
          }
```

```
        }
    }
}
```

Una volta che le lettere sono aggiunte nell'apposita textArea chiamata phraseArea è possibile ascoltare il contenuto attraverso il bottone nominato tts (textToSpeech) opportunamente settato. Per far ciò è stato creato un listener sul bottone tts che quando viene cliccato viene chiamato il metodo speak della classe pubblica TextToSpeech:

Codice 3.21: Text-to-speech

```
tts.setOnClickListener(
        new View.OnClickListener(){

    @Override
    public void onClick(View v) {
        textToSpeech.speak(phrase,
        TextToSpeech.QUEUE_FLUSH, null);
    }
});
```

Inoltre all'interno della directory library abbiamo due classi di utility che andremo ad analizzare:

- JsonParser.java: Essa ha un unico metodo pubblico al suo interno chiamato getJSONFromUrl il quale si occupa di creare una richiesta HTTP, fare il parsing JSON della stringa da inviare e riceve le risposte dal server.

- UserFunction.java: Questa classe invece ha diversi metodi, sicuramente meritevole di nota è sendSygnal, esso infatti "prepara" la stringa da

inviare al server prima che il metodo getJSONFromUrl della classe JsonParser.java la invii realmente.

Codice 3.22: invio richiesta JSON al server

```
JSONObject json;

public JSONObject sendSygnal(String parameter){
List<NameValuePair> params =
new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("tag",param));
params.add(new BasicNameValuePair("sygnal",param));
json =jsonParser.getJSONFromUrl(sygnalURL,params);
return json;
}
```

3.4 Test e risultati

È stato effettuato un test sulle performance, usabilità del guanto e del progetto in generale. Il test è stato sottoposto a diversi campioni nella totalità dei casi normodotati che non conoscevano precedentemente l'alfabeto Malossi. Questo non dovrebbe essere visto come un'invalidazione dei test poichè si basa sulle performance e sull'usabilità del guanto stesso, è comunque nostra intenzione proporre, in un prossimo futuro, gli stessi test agli utenti target di questo progetto, le persone affette da sordo-cecità. A tal proposito sono già state contattate diverse associazioni che si occupano di questo, una su tutte: "La Lega del Filo d'Oro".

Sono state scelte 15 citazioni famose ed abbiamo provato a "scrivere" con il guanto in questione. Attraverso il test teniamo conto dei falsi positivi e dei falsi negativi e li mettiamo in relazione le parole inserite correttamente, così da avere la percentuale di precisione. Di seguito mostriamo le frasi proposte:

1. *carpe diem - Orazio*
2. *hello world - programmatore X*
3. *i have a dream - Martin Luther King*
4. *un passo alla volta mi basta - Mahatma Gandhi*
5. *nel mezzo del cammin di nostra vita - Dante Alighieri*
6. *dio non gioca a dadi con l universo - Albert Einstein*
7. *cio che non mi distrugge mi rende più forte - Nietzsche*
8. *dio è morto marx è morto e io mi sento poco bene - Woody Allen*
9. *in questo mondo pieno di responsabilità e di fuochi fatui - Italo Calvino*
10. *se non fai parte della soluzione allora fai parte del problema - Anonimo*
11. *se ho visto più lontano, e perche stavo sulle spalle di giganti - Isaac Newton*
12. *la fortuna non esiste esiste il momento in cui il talento incontra l'opportunità - Seneca*
13. *la scienza non ci ha ancora insegnato se la pazzia sia o no più sublime dell'intelligenza - Edgar allan Poe*
14. *con l'insonnia nulla è reale. Tutto è lontano. Tutto è una copia di una copia di una copia - Fight Club (Chuch Palahniuk)*
15. *il problema dell'umanità è che gli stupidi sono sempre sicurissimi mentre gli intelligenti sono pieni di dubbi - Bertrand Russell*

Tabella 3.1: Campione 1

frase	**num parole**	**fasi positivi**	**falsi negativi**	**% precisione**
1	10	1	1	80
2	11	1	0	90,9090909091
3	15	1	2	80
4	28	3	2	82,1428571429
5	35	1	3	88,5714285714
6	35	1	1	94,2857142857
7	43	4	2	86,0465116279
8	48	3	2	89,5833333333
9	57	4	2	89,4736842105
10	62	4	7	82,2580645161
11	63	4	9	79,3650793651
12	80	8	8	80
13	89	8	10	79,7752808989
14	90	7	13	77,7777777778
15	110	15	14	73,6363636364
Totale	**776**	**65**	**76**	**83,5883457517**

Tabella 3.2: Campione 2

frase	**num parole**	**fasi positivi**	**falsi negativi**	**% precisione**
1	10	3	1	60
2	11	0	4	63,6363636364
3	15	2	1	80
4	28	3	3	78,5714285714
5	35	4	3	80
6	35	0	2	94,2857142857
7	43	1	0	97,6744186047
8	48	2	1	93,75
9	57	2	2	92,9824561404
10	62	6	7	79,0322580645
11	63	8	8	74,6031746032
12	80	10	7	78,75
13	89	12	7	78,6516853933
14	90	13	7	77,7777777778
15	110	15	9	78,1818181818
Totale	**776**	**81**	**62**	**80,5264730173**

Tabella 3.3: Campione 3

frase	**num parole**	**fasi positivi**	**falsi negativi**	**% precisione**
1	10	2	1	70
2	11	1	3	63,6363636364
3	15	0	2	86,6666666667
4	28	2	3	82,1428571429
5	35	1	2	91,4285714286
6	35	0	1	97,1428571429
7	43	1	0	97,6744186047
8	48	2	2	91,6666666667
9	57	2	2	92,9824561404
10	62	5	4	85,4838709677
11	63	6	7	79,3650793651
12	80	9	4	83,75
13	89	7	9	82,0224719101
14	90	9	12	76,6666666667
15	110	10	13	79,0909090909
Totale	**776**	**57**	**65**	**83,9813236953**

Tabella 3.4: Campione 4

frase	**num parole**	**fasi positivi**	**falsi negativi**	**% precisione**
1	10	1	1	80
2	11	1	2	72,7272727273
3	15	1	3	73,3333333333
4	28	1	2	89,2857142857
5	35	4	2	82,8571428571
6	35	0	0	100
7	43	0	0	100
8	48	1	3	91,6666666667
9	57	3	2	91,2280701754
10	62	4	4	87,0967741935
11	63	5	7	80,9523809524
12	80	7	10	78,75
13	89	7	9	82,0224719101
14	90	9	13	75,5555555556
15	110	11	14	77,2727272727
Totale	**776**	**55**	**72**	**84,1832073287**

Tabella 3.5: Campione 5

frase	**num parole**	**fasi positivi**	**falsi negativi**	**% precisione**
1	10	2	2	60
2	11	1	3	63,6363636364
3	15	3	3	60
4	28	1	4	82,1428571429
5	35	4	0	88,5714285714
6	35	3	3	82,8571428571
7	43	2	3	88,3720930233
8	48	5	1	87,5
9	57	4	3	87,7192982456
10	62	5	6	82,2580645161
11	63	4	9	79,3650793651
12	80	10	8	77,5
13	89	9	11	77,5280898876
14	90	9	13	75,5555555556
15	110	17	14	71,8181818182
Totale	**776**	**79**	**83**	**77,6549436413**

Si evince quindi, una buona precisione in tutti i test effettuati, soprattutto tenendo conto che questo è un primo prototipo, inoltre è stato notato che la totalità dei falsi negativi sono dati dall'utilizzo di materiali non propriamente idonei al fine postoci.

Sono stati effettuati cinque test in accordo con il paper di [52], il quale afferma che questo è il numero giusto per aver il miglior risultato dai test di usabilità. Possiamo notare come tutte le prove da noi effettuate abbiano un andamento simile, come visibile anche in Figura 3.1. La figura pone sull'asse delle ascisse il numero delle frasi proposte ai campioni, mentre sull'asse delle ordinate la percentuale di precisione del guanto rispetto ai campioni. Dividendo il test in tre fasi da cinque frasi l'uno possiamo notare come nella prima fase ci siano più errori rispetto alla seconda (anche se le frasi hanno meno lettere) e come nell'ultima fase riaumentino gli errori. La nostra ipotesi è che in prima istanza l'utente deve prendere confidenza con il guanto e che quindi sbaglia di più rispetto alla seconda fase dove in effetti pare più a suo agio e digita in maniera più spedita, infine nella terza fase gli errori aumentano nuovamente poiché le frasi sono più lunghe e complicate ed inoltre la stanchezza inizia a farsi sentire (il guanto non è propriamente comodo essendo particolarmente rigido).

Inoltre, vogliamo evidenziare come anche la grandezza della mano incida in questo test. Durante le prove con i campioni sono state appuntate alcune frasi chiave al fine di generare anche una valutazione qualitativa sull'usabilità del guanto. Ad esempio per alcuni il guanto risultava troppo grande e questo rendeva difficile il digitare dei tasti posti nella parti alte del guanto: "Non riesco a cliccare sulle lettere B e C", afferma uno degli utenti presi in esame. Inoltre, lo spessore delle dita ha un'importanza fondamentale in quanto ad uno spessore maggiore corrisponde un'indossabilità migliore, e dall'altra parte chi deve digitare ha meno difficoltà. In accordo con [53], durante questa fase non abbiamo interrotto le performance degli utenti se non per mostrare le frasi da digitare. Come descritto in [54], alcune metriche standard di valutazione, come nel nostro caso la libertà di movimento, il movimento stesso

e la precisione, influiscono sui risultati finali. Ad esempio, nelle ultime frasi, che ricordiamo essere quelle con più lettere, quasi tutti i tester si sono dimostrati visibilmente affaticati nel mantenere l'attenzione e soprattutto la precisione nello svolgimento della prova. É quindi evidente, poiché è stato creato un solo guanto per il prototipo, che con una mano con una superficie più ampia, è più semplice premere i tasti e di conseguenza in media fare meno errori.

In definitiva per l'obiettivo che ci eravamo prefissati, dal punto di vista delle performance e dell'usabilità, possiamo ritenerci soddisfatti.

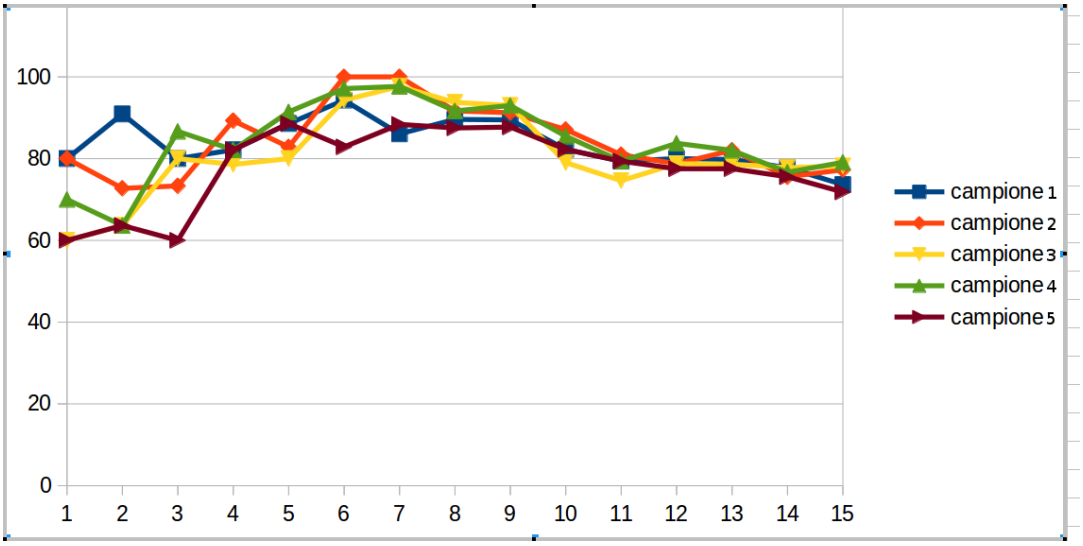

Figura 3.1: Campioni test performance

Conclusioni

GlovePi è un guanto per la comunicazione sociale aptica tra persone affette da sordo-cecità attraverso l'alfabeto Malossi.

Il sistema permette di comunicare con persone sordo-cieche anche se non si è a conoscenza di tale codifica, infatti la persona normodotata che indossa il guanto appone la mano alla persona affetta da sordocecità la quale "digitando" il suo messaggio, quindi senza dover conoscere la codifica Malossi attraverso un app verrà palesato il suddetto messaggio.

Tra i metodi di comunicazione più diffusi ed utilizzati da persone affette da sordocecità vi è l'alfabeto Malossi, il quale risulta il più affine alle esigenze di questo progetto: (i) l'intuitività, (ii) la semplicità d'uso, (iii) il supporto multilingua e multiconcetto, (iv) l'analogia informatica.

A rendere possibile la comunicazione vi è un guanto corredato di sensori divisi per tipologia tra parte frontale e parte posteriore collegato (tramite pin GPIO) ad una Raspberry Pi 1 Model B+ che si occupa del salvataggio ed elaborazione dei dati in arrivo oltre che destinare in maniera idonea i dati in uscita. A concludere, un applicativo Android richiede i dati (in manuale oppure in polling) al single-board computer utilizzato al fine di visualizzarli sotto forma di testo in un'apposita textview, e nel caso poterli ascoltare tramite un apposito tasto.

Dopo un'attenta analisi dello stato del'arte nella quale si è evinta una necessità ancora da esplorare, il progetto è stato espanso per favorire la comunicazione sociale aptica many-to-many in ambito distribuito. Appoggiandosi ad uno scenario di riferimento, ovvero la *conferenza*, è stata progettata

un'architettura in tal senso facendo riferimento a TuCSoN, tecnologia già consolidata.

Tra gli obiettivi posti in questo progetto vi è la rimozioni di barriere comunicative tra persone sordo-cieche e persone che non necessitano di tecnologie assistive così vincolanti, avendo come aspetto principe il fatto che sia una tecnologia a basso costo ed open-source. Questo offre la possibilità di ulteriori miglioramenti da parte di una comunità sensibile al problema.

Le innovazioni introdotte da GlovePi sono state presentate al workshop "Accessible Devices and Services", che si è tenuto a gennaio 2017, nell'ambito della conferenza internazionale IEEE Consumer Communications and Networking Conference 2017 (CCNC 2017), conferenza che si tiene tutti gli anni a Las Vegas, come evento satellite dell'Annual and Global Consumer Electronic Show (CES).

Gli sviluppi futuri prevedono la completa implementazione dell'architettura distribuita, la quale renderebbe possibile la comunicazione many-to-many utilizzando come base il guanto sopraccitato ma inserendo ulteriori sensori aptici di output che reagiranno vibrando nelle zone della mano corrispondenti all'alfabeto utilizzato, dando la possibilità anche al sordo-cieco di percepire il contenuto inviatogli.

Inoltre, si potrebbe ampliare il sistema dando la possibilità di configurazione del guanto sulla base dei fonemi corrispondenti ad una determinata lingua, coprendo in tal modo un'area geografia più sostanziosa e conseguentemente apportando delle migliorie in ambito comunicativo tra i soggetti interessati.

Infine, resta non meno importante lo sviluppo fisico del prodotto, il quale essendo al momento in fase di prototipaggio, richiede sicuramente migliore vestibilità.

Bibliografia

[1] J. Medina, *Brain Rules: 12 Principles for Surviving and Thriving at Work, Home, and School (Large Print 16pt).* ReadHowYouWant. com, 2011.

[2] J. Delwiche, *"The impact of perceptual interactions on perceived flavor,"* Food Quality and preference, vol. 15, no. 2, pp. 137–146, 2004.

[3] S. Mirri, C. Prandi, M. Roccetti, and P. Salomoni, *"Food and gastronomic heritage: Telling a story of eyes and hands,"* in Computers and Communication (ISCC), 2016 IEEE Symposium on. IEEE, 2016, pp. 6–9.

[4] S. Mirri, P. Salomoni, A. Pizzinelli, M. Roccetti, and C. E. Palazzi, *"di piazza in piazza": Reimagining cultural specific interactions for people-centered exhibitions,"* in 2016 International Conference on Computing, Networking and Communications (ICNC). IEEE, 2016, pp. 1–5.

[5] J. F. Delwiche, *"You eat with your eyes first,"* Physiology and behavior,vol. 107, no. 4, pp. 502–504, 2012.

[6] C. Spence, K. Okajima, A. D. Cheok, O. Petit, and C. Michel, *"Eating with our eyes: from visual hunger to digital satiation," Brain and cognition, 2015.*

[7] S. Ferretti, S. Mirri, C. Prandi, and P. Salomoni, "Automatic web content personalization through reinforcement learning," Journal of Systems and Software, 2016.

[8] "User centered and context dependent personalization through experiential transcoding," in 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC). IEEE, 2014, pp. 486–491.

[9] S. Mirri, C. Prandi, and P. Salomoni, "A context-aware system for personalized and accessible pedestrian paths," in High Performance Computing and Simulation (HPCS), 2014 International Conference on. IEEE, 2014, pp. 833–840.

[10] P. Salomoni, C. Prandi, M. Roccetti, V. Nisi, and N. J. Nunes, "Crowdsourcing urban accessibility:: Some preliminary experiences with results," in Proceedings of the 11th Biannual Conference on Italian SIGCHI Chapter. ACM, 2015, pp. 130–133.

[11] M. Roccetti, S. Ferretti, C. E. Palazzi, P. Salomoni, and M. Furini, "Riding the web evolution: from egoism to altruism," in 2008 5th IEEE Consumer Communications and Networking Conference. IEEE, 2008, pp. 1123–1127.

[12] S. Ferretti, S. Mirri, M. Roccetti, and P. Salomoni, "Notes for a collaboration: On the design of a wiki-type educational video lecture annotation system," in International Conference on Semantic Computing (ICSC 2007). IEEE, 2007, pp. 651–656.

[13] C. Andersson, D. Campbell, A. Farquharson, S. Furner, J. Gill, A. Jackson, J. Lucker, K. Nolde, E. Werner, M. Whybray et al., Assistive technology for the hearing-impaired, deaf and deafblind. Springer Science and Business Media, 2006.

[14] V. Khambadkar, E. Folmer, *"A tactile-proprioceptive communication aid for users who are deafblind,"* in 2014 IEEE Haptics Symposium (HAPTICS). IEEE, 2014, pp. 239–245.

[15] U. Gollner, T. Bieling, G. Joost, *"Mobile lorm glove: introducing a communication device for deaf-blind people,"* in Proceedings of the

Sixth International Conference on Tangible, Embedded and Embodied Interaction. ACM, 2012, pp. 127–130.

[16] N. Caporusso, "A wearable malossi alphabet interface for deafblind people," in Proceedings of the working conference on Advanced visual interfaces. ACM, 2008, pp. 445–448.

[17] *"db glove,"* 2016, retrieved: September, 2016. [Online]. Available: http://www.dbglove.com/ .

[18] Carlo Giacobini, Direttore responsabile di HandyLex.org *http://www.handylex.org/schede/sordocecita_definizioni.shtml.*

[19] Kathryn Wolff Heller, Cheryl Kennedy *Etiologies and Characteristics of Deaf-Blindness*,pp. 36-51.

[20] *http://www.deafblindinformation.org.au/acquired-deafblindness/?contrast=reg.*

[21] Shirley M. Gumpel, Kathleen Hayes, J. A. Dudgeon *Congenital Perceptive Deafness: Role of Intrauterine Rubella.*

[22] CatherineS Peckham a, J.A.M Martin b, W.C Marshall c, J.A Dudgeon c *CONGENITAL RUBELLA DEAFNESS: A PREVENTABLE DISEASE*, Volume 313, Issue 8110, 3 February 1979, Pages 258-261.

[23] Charlene M. T. Robertson, Tanis M. Howarth, Dietlind L. R. Bork, Irina A. Dinu *Permanent Bilateral Sensory and Neural Hearing Loss of Children After Neonatal Intensive Care Because of Extreme Prematurity: A Thirty-Year Study*,Pediatrics May 2009, VOLUME 123 / ISSUE 5.

[24] Elaine S Marlow, Linda P Hunt, Neil Marlow *Sensorineural hearing loss and prematurity*, IArch Dis Child Fetal Neonatal Ed 2000.

[25] Kim D BlakeEmail and Chitra Prasad *CHARGE syndrome.* Orphanet Journal of Rare Diseases2006, licensee BioMed Central Ltd. 2006.

[26] Karen B. Fowler, DrPH, Faye P. McCollister, EdD, Arthur J. Dahle, PhD, Suresh Boppana, MD, William J. Britt, MD, Robert F. Pass, MD *Progressive and fluctuating sensorineural hearing loss in children with asymptomatic congenital cytomegalovirus infection* The Journal of Pediatrics Volume 130, Issue 4, April 1997, Pages 624–630.

[27] Kathryn Wolff Heller, R.N., Ph.D.,Georgia State University Cheryl Kennedy *University of Pittsburgh Etiologies and Characteristics of Deaf-Blindness.*

[28] M.L. Knapp, *Nonverbal communication in human interaction*, New York: Holt, Rinehart and Winston, 1978.

[29] Wahlqvist, Moa; Möller, Claes; Möller, Kerstin *Physical and Psychological Health in Persons with Deafblindness That Is Due to Usher Syndrome Type II*, Journal of Visual Impairment & Blindness (May-Jun 2013): 207.

[30] Jesper Dammeyer *Mental and behavioral disorders among people with congenital deafblindness*, Research in Developmental Disabilities, Volume 32, Issue 2, March–April 2011, Pages 571–575.

[31] Sarah M. Bodsworth Isabel C.H. Clare Sara K., *Psychological distress and unmet need among adults with dual sensory impairment*, British Journal of Visual Impairment, Vol 29, Issue 1, 2011.

[32] Brennan, M. (2001) *'Psychosocial Issues of Deafblindness'* , Deafblind American 40(4): 16-24.

[33] *https://www.legadelfilodoro.it/chi-sosteniamo/se-conosci-un-sordocieco/la-comunicazione.*

[34] *http://www.legadelfilodoro.it/chi-sosteniamo/se-conosci-un-sordocieco/la-comunicazione.*

[35] R Jakobson *Linguistics and communication theory*,Proceedings of symposia in applied mathematics, 1961.

[36] Chandrika Jayant, Christine Acuario, William A. Johnson, Janet Hollier, Richard E. Ladner, *VBraille: Haptic Braille Perception using a Touch-screen and Vibration on Mobile Phones*,Computer Science and Engineering Department University of Washington.

[37] Michitaka Hirose,Tomohiro Amemiya; *Wearable Finger-Braille Interface for Navigation of Deaf-Blind in Ubiquitous Barrier-Free Space*, HCI International 2003.

[38] Tanay Choudhary, Saurabh Kulkarni, Pradyumna Reddy, *A Braille-Based Mobile Communication and Translation Glove for Deaf-blind People*, Birla Institute of Technology & Science, Pilani - K.K. Birla Goa Campus Goa, India.

[39] Telebraille, *http://www.deafblind.com/telebrl.html*, access date: 2-2-2012.

[40] Gilden, Deborah; Jaffe, David L. , *Dexter, a robotic hand communication aid for the deaf-blind.*, International Journal of Rehabilitation Research: June 1988 - Volume 11 - Issue 2 - ppg 198;

[41] Jaffe David L., *Ralph, a fourth generation finger-spelling hand.*, Rehabilitation Research and Development Center, 1994.

[42] Peter Grigson,Nils Lofmark, Roger Giblin, *Hand-tapper III: a prototype communication device using finger-spelling*, British Journal of Visual Impairment March 1991 vol. 9 no. 1 13-15.

[43] Vinitha Khambadkar, Eelke Folmer, *A Tactile-Proprioceptive Communication Aid for Users who are Deafblind.*

[44] Silvia Mirri, Catia Prandi, Paola Salomoni, Lorenzo Monti, *Fitting like a GlovePi: a wearable device for deaf-blind people*, 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC).

[45] Nicholas Caporusso,*wearable Malossi alphabet interface for deafblind people*,Proceeding AVI '08 Proceedings of the working conference on Advanced visual interfaces,Pages 445-448.

[46] Ulrike Gollner, Tom Bieling, Gesche Joost, *Mobile Lorm Glove – Introducing a Communication Device for Deaf-Blind People*, TEI '12 Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction Pages 127-130.

[47] Stephen Brewster, Michael Hughes, *Pressure-Based Input for Mobile Devices*.

[48] David Dewhurst, *"Silooets" - Audiotactile Vision-Substitution Software*.

[49] Troy McDaniel, Sreekar Krishna, Vineeth Balasubramanian, Dirk Colbry and Sethuraman Panchanathan,*Using a Haptic Belt to Convey non-verbal communication cues during social interactions to Individuals who are blind*.

[50] *http://www.pi-point.co.uk/*.

[51] *http://blog.davidsingleton.org/introducing-piui/*.

[52] Nielsen, Jakob, and Landauer, Thomas K., *"A mathematical model of the finding of usability problems,"* Proceedings of ACM INTERCHI'93 Conference (Amsterdam, The Netherlands, 24-29 April 1993), pp. 206-213.

[53] Brian P. Bailey, Joseph A. Konstan, and John V. Carlis, *Effects of Interruptions on Task Performance, Annoyance, and Anxiety in the User Interface,*, Interact. Vol. 1. 2001.

[54] Kirkpatrick, Arthur E., and Sarah A. Douglas., *"Application-based evaluation of haptic interfaces."* Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002. HAPTICS 2002. Proceedings. 10th Symposium on. IEEE, 2002.

Ringraziamenti

Inizio ringraziando chi mi ha permesso di realizzare questo lavoro di tesi. Ringrazio di cuore la Dott.essa Silvia Mirri che si è sempre dimostrata molto disponibile, il Prof. Dario Maio a cui devo l'inizio di questo progetto che ha preso forma proprio durante il suo corso, fonte di grandissima ispirazione e senza il quale questo lavoro non si sarebbe mai realizzato.

Ora proseguo con i ringraziamenti alle persone a me più vicine, prima di tutto i miei genitori che mi hanno dato effettivamente i mezzi per poter vivere questi intensi anni di università e che mi hanno sempre supportato e SOPPORTATO nelle mie scelte senza mai obbiettare. Risulta doveroso ringraziare coloro che mi hanno appoggiato nella realizzazione di questo progetto, Diego che come sempre è stato un supporto indispensabile oltre che un Amico con la lettera maiuscola, Sax per aver contribuito con i cavetti, i tastini e la sua indiscussa abilità nel saldare a stagno, Norma per aver accettato una delle richieste più improbabili nella sua carriera da sarta, ovvero cucire una Raspberry Pi su una sezione di guanto e a tutti gli amici che si sono sempre prestati come cavie ai miei deliri. Grazie.

Ultimo ma non ultimo per importanza, vorrei ringraziare Oxana, per il suo incondizionato sostegno.

Addio e grazie per tutto il pesce.