Università degli Studi di Napoli Parthenope

Dipartimento di Scienze e Tecnologie

Corso di laurea in informatica

Tesi di laurea

RoboBVN: Assistive Robot for people's language predictive of the later emergence of ASD

Relatore

Prof.ssa Mariacarla Staffa

Candidato

Gianmarco Donnesi

Matr. 0124002147

Anno Accademico 2022/2023

A Totò e Mimma, i miei nonni,

coloro che mi hanno insegnato

ad essere la persona che sono.

Indice

Elenco delle figure

Capitolo 1

Introduzione

I disturbi dello spettro autistico (*DSA*) si riferiscono a disturbi dello sviluppo neurologico con esordio nei primi tre anni di vita e sono caratterizzati da difficoltà di comunicazione, di interazione sociale e da modelli limitati e ripetitivi nei comportamenti, negli interessi e nelle attività di un individuo. L'autismo rappresenta una sindrome molto complessa, nella maggior parte dei casi di natura genetica e poiché le sue manifestazioni sono molto varie e possono differire notevolmente da soggetto a soggetto, è più appropriato parlare di "spettro autistico". Per diversi anni i *DSA* sono stati considerati essenzialmente rari con la prevalenza minore di un bambino affetto su mille. Ad oggi, il *World Health Organization* stima che la prevalenza sia di un bambino su cento e sostiene che sembra essere in aumento negli anni futuri. Essi vengono diagnosticati quando il soggetto, a seguito di test standardizzati somministrati individualmente, su lettura, calcolo, o espressione scritta, ottenga risultati significativamente al di sotto delle soglie predeterminate che tengono conto dell'età, dell'istruzione e del livello di intelligenza.

Come confermato a più riprese dagli esperti, la diagnosi e l'avvio conseguente di un intervento precoce può migliorare notevolmente la prognosi dei bambini con

DSA e la qualità di vita delle loro famiglie. Purtroppo, la varietà e la diversità delle manifestazioni legate ai DSA, la limitata conoscenza degli stessi da parte delle famiglie, il numero insufficiente di centri specializzati e di professionisti esperti disponibili, concorrono a causare ritardi nella diagnosi e quindi nella cura.

Le nuove tecnologie informatiche rispondono alla crescente esigenza di individuare le migliori pratiche per lo screening e la diagnosi dei DSA già nelle prime fasi dello sviluppo dell'individuo. Esse, infatti, sono da tempo all' attenzione del mondo medico, oltre che educativo e scientifico, per la valenza facilitante che possono avere nella progettazione di programmi abilitativi personalizzati/individualizzati [1]. In particolare, per i pazienti con DSA le tecnologie informatiche adeguatamente impiegate, costituiscono un valido supporto nei percorsi di intervento e di trattamento finalizzati al superamento dei deficit comunicativi e relazionali potenziando l'apprendimento, la comunicazione e la socializzazione.

Esse consentono di usare un canale comunicativo prevalentemente visuo-spaziale, particolarmente adatto alle caratteristiche dei pazienti con disturbi dello spettro autistico. Inoltre, il linguaggio informatico, in quanto chiaro, strutturato e prevedibile, ha il grande pregio di non comportare inferenze emotive, incomprensibili per una persona con autismo. Tra gli approcci più innovativi vi è sicuramente quello relativo alla possibilità di utilizzare tecnologie assistive durante le sessioni di terapia. I rapidi progressi nell'area della robotica offrono oggi enormi possibilità di innovazione e di trattamento o addirittura di educazione per i pazienti affetti da DSA. Fu lo scrittore ceco *Karel Čapek* che nel suo dramma fantascientifico teatrale *R.U.R.* (*I robot universali di Rossum*) nel 1920, in sostituzione del termine "automa", ideò il termine "robot" che stava ad indicare una macchina meccanica

con funzioni di lavoro. Successivamente, lo scrittore russo *Isaac Asimov* nel 1942 pubblicò tre leggi ancor oggi geniali nella loro intuizione. Si trattava delle tre leggi della robotica, nate per regolamentare e garantire il comportamento etico di una macchina dotata di intelligenza artificiale.

All'epoca, tali leggi si limitavano ad apparire come l'acuta invenzione di uno scrittore di fantascienza. Oggi, invece, le ricerche scientifiche e tecnologiche hanno compiuto progressi tali da renderle quanto mai attuali; sono diventate un punto fermo della fantascienza e un fondamento dell'etica robotica.

Da allora la robotica ha fatto dei progressi straordinari fino a trasformare gli attuali robot in entità autonome con capacità di movimento, apprendimento, comunicazione e adeguamento all' ambiente circostante, fino a prospettare la possibilità di robot dotati di intelligenza artificiale (IA).

La Robotica di Assistenza Sociale (RAS) è quel campo della ricerca riguardante il modo in cui i robot assistono le persone attraverso l'interazione sociale. Essa costituisce un sottocampo dell'interazione uomo-robot (HRI), volto allo sviluppo di interazioni efficienti con l'utente in contesti terapeutici ed educativi per aiutarlo a costruire abilità di comportamento sociale [2].

L'utilizzo di interventi basati sulla RAS ha dimostrato in primo luogo il notevole potenziale nel trattamento dei DSA con l'aumento del coinvolgimento del paziente con difficoltà nella comunicazione. Inoltre, l'impiego di tali tecnologie ha messo a disposizione dei terapisti la possibilità di organizzare sessioni di trattamento più interattive [3][4] con supporto negli aspetti valutativi (oggettivazione, raccolta dati), nonché la facoltà di delineare percorsi di cura riabilitativi innovativi e altamente personalizzati. Il miglioramento ed il potenziamento dello sviluppo sociale, emotivo,

cognitivo e sensoriale, sono i principali obiettivi terapeutici ed educativi per il trattamento e la cura dei DSA. Naturalmente, ognuna di queste aree richiede un approccio e tipologie di dispositivi specifici, per questo sono stati realizzati modelli di intervento distinti a seconda degli scopi della determinata competenza su cui lavorare.

Nel contesto riabilitativo, le due categorie di robot che vengono maggiormente utilizzati sono quelli a scopo terapeutico e quelli a scopo assistivo. Nel presente lavoro di tesi, si è deciso di impiegare robot antropomorfi di tipo assistivo per favorire lo sviluppo di competenze emotive e sociali al fine di migliorare il benessere psicologico del paziente.

Numerosi studi condotti su bambini affetti da disturbi dello spettro autistico hanno evidenziato risultati positivi in termini di interazione sociale. I bambini hanno mostrato una maggiore interazione diadica e triadica, uno sviluppo migliore delle capacità di comunicazione sociale e un'acquisizione di regole comportamentali più efficace.

Un robot umanoide è in grado di fornire gli stimoli sociali adeguati in modo molto simile a quello degli operatori umani, può essere riconosciuto facilmente come "compagno" e può essere programmato con applicazioni che sollecitano lo sviluppo di abilità interpersonali di cui i pazienti hanno bisogno: imitazione, contatto oculare e fisico, comunicazione verbale e non verbale. I robot possono attivare luci o musiche e compiono movimenti lenti e prevedibili al fine di fornire i rinforzi positivi di cui necessitano i pazienti.

Allo stato attuale non esiste ancora un metodo esclusivo per confermare una diagnosi di autismo attraverso test di laboratorio, di neuroimaging o genetici; anche gli

strumenti psicodiagnostici sono spesso difficili da impiegare per le difficoltà tipiche della sindrome. Rilevato che la diagnosi di autismo si basa in gran parte sulla valutazione clinica, l'avvalimento in fase diagnostica delle capacità del robot di rilevazione e registrazione delle risposte dell'individuo, consente di determinare con una certa accuratezza l'eventuale presenza di un paziente affetto da DSA.

Le valutazioni sono favorite fornendo stimoli standardizzati per valutare i comportamenti caratteristici dell'autismo così da giungere a diagnosi integrate sempre più affidabili. La risposta del paziente agli stimoli del robot umanoide, al tono della sua voce, ai movimenti degli occhi, ai cambiamenti di espressione e al tipo di conversazione forma dati che vengono raccolti ed elaborati per una successiva analisi. Quest'ultima viene svolta da un lato in maniera totalmente automatizzata e dall'altro tramite una valutazione clinica di un esperto.

Tutte le considerazioni e le ricerche sopra esposte hanno posto le basi per il presente lavoro di tesi, il quale ha avuto come obiettivo lo sviluppo di una applicazione RAS per la somministrazione di test per l'analisi delle funzioni cognitivo-linguistiche in individui affetti da disturbi dello spettro autistico [5].

Capitolo 2

Tecnologie impiegate

In questo capitolo sono descritte in dettaglio le tecnologie utilizzate per la realizzazione dell'applicazione. Lo studio di esse ha occupato una parte rilevante del lavoro di tesi in quanto si tratta di tecnologie di nuova generazione, come di seguito elencate:

1. **Pepper**: robot umanoide progettato e sviluppato dall'azienda *SoftBank Robotics*™, specializzata nella progettazione, sviluppo e commercializzazione di robot umanoidi e servizi robotici intelligenti.

2. **Choregraphe**: software sviluppato dall'azienda *SoftBank Robotics*™, progettato specificamente per la programmazione e la gestione dei robot umanoidi della famiglia *NAO*, compresi *NAO* e *Pepper*.

3. **NAOqi OS**: un sistema operativo sviluppato da *SoftBank Robotics*™, che fornisce un'interfaccia di programmazione completa e potente, per accedere e controllare le funzionalità dei robot della famiglia *NAO*.

Figura 2.1: Logo dell'azienda SoftBank Robotics

2.1 Pepper

Pepper è un robot umanoide progettato e sviluppato dall'azienda SoftBank Robotics™, un'azienda specializzata nella creazione di robot sociali [6]. Ha un'altezza di circa 1 metro e 20 cm, un peso di circa 30 kg e presenta un design in grado di conferirgli sembianze umane. Esso è alimentato da un sistema operativo basato su *Linux* e utilizza una combinazione di software proprietari e applicazioni sviluppate da terze parti per fornire una vasta gamma di funzionalità. È stato progettato per svolgere compiti come l'assistenza all'utente, la presentazione di informazioni, l'intrattenimento e l'interazione sociale.

L'obiettivo principale del suo utilizzo è quello di creare connessione emotiva con l'utente, offrendo un'esperienza coinvolgente e personalizzata.

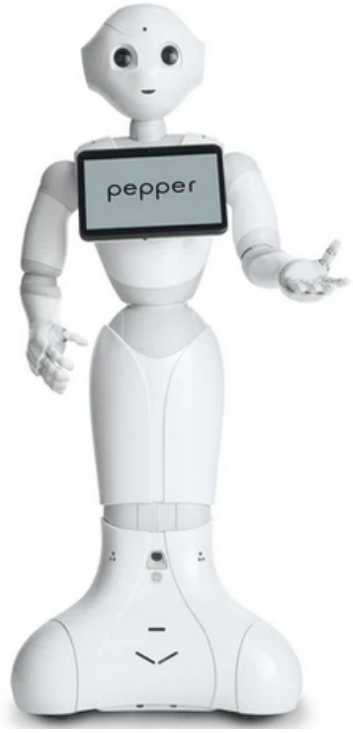


Figura 2.2: Robot umanoide Pepper

Pepper è costituito da un sistema a tre strati.

La testa contiene tutti i sensori e i componenti di elaborazione dei dati, compresi microfoni, telecamere, sensori di prossimità e un processore *Intel ® Atom™*.

Il torace ospita il computer principale, che include una *CPU*, una *GPU*, memoria *RAM* e *flash*, nonché connettività *Wi-Fi* e *Ethernet* per consentire il collegamento a Internet e l'accesso a informazioni e servizi basati sul cloud.

La base contiene le ruote per la mobilità del robot, la batteria per l'alimentazione e sonar per l'individuazione di ostacoli durante lo spostamento del robot.

L'androide è equipaggiato con un vasto assortimento di sensori, tra cui telecamere, microfoni direzionali, sensori di prossimità e sensori tattili, che gli consentono di percepire l'ambiente circostante e interagire in modo intelligente con le persone.

Questi vengono elencati ed analizzati di seguito:

- **Sensori di contatto**: situati nelle mani e nelle dita, consentono a Pepper di rilevare il tocco e le interazioni fisiche.

- **Sensori di prossimità**: posizionati nel petto, rilevano la presenza di oggetti o persone nelle vicinanze.

- **Sensori a infrarossi**: consentono a Pepper di misurare la distanza dagli oggetti e di eluderli per evitare collisioni.

- **Microfoni**: permettono al robot di rilevare e registrare i suoni ambientali.

- **Telecamere**: Pepper ha telecamere ad alta definizione posizionate nella testa e nel petto per riconoscere i volti, rilevare le espressioni facciali e percepire l'ambiente visivo.

Il robot utilizza una combinazione di ruote e giunti per il movimento. Ha ruote omnidirezionali nella base che gli consentono di ruotare in ogni direzione in modo fluido. Inoltre l'automa è equipaggiato con un sistema di riconoscimento vocale avanzato che gli consente di comprendere e rispondere a comandi vocali in diverse lingue. Ancora, è in grado di analizzare il linguaggio naturale utilizzando algoritmi di elaborazione del linguaggio e di generare risposte coerenti.
Pepper utilizza il framework NAOqi, basato su Linux, che fornisce un'ampia gamma di API e strumenti per gli sviluppatori per creare applicazioni personalizzate e ampliare le capacità del robot.

Gli sviluppatori possono creare moduli comportamentali, implementare algoritmi di visione artificiale, elaborare dati sensoriali e personalizzare l'interazione con l'utente. Infine, Pepper presenta un display touch da 10,1 pollici montato sul petto. Questo schermo viene utilizzato per visualizzare informazioni e comunicare con l'utente tramite interfacce personalizzate.

In conclusione, la sua versatilità, la sua tecnologia innovativa e la capacità di interazione *user-friendly* lo hanno reso sicuramente un prezioso e interessante strumento, offrendo opportunità di apprendimento e di sperimentazione nelle nuove frontiere della robotica e dell'intelligenza artificiale.

2.2 Choregraphe

Aldebaran Choregraphe [7] è un ambiente di programmazione visuale utilizzato per creare e gestire comportamenti di robot umanoidi della famiglia NAO, messo a disposizione dalla SoftBank Robotics.

Figura 2.3: Logo del software Aldebaran Choregraphe

Esso offre un ambiente di sviluppo basato su un'interfaccia grafica intuitiva, che permette agli sviluppatori di programmare i robot utilizzando il concetto di "grafi di comportamento". Questi grafi rappresentano una serie di azioni, movimenti

e interazioni che il robot può eseguire. I comportamenti possono essere creati collegando blocchi funzionali predefiniti, chiamati *"box"*, tramite linee di connessione che rappresentano il flusso dell'esecuzione delle azioni del robot.

Ogni box rappresenta una specifica azione o una sequenza di azioni che il robot può compiere, come ad esempio camminare, riconoscere il volto umano o riprodurre un suono.

Choregraphe fornisce anche una vasta gamma di strumenti per la personalizzazione dei comportamenti dei robot. Infatti tramite l'utilizzo di script Python all'interno del software, è possibile estendere le funzionalità di base dei box predefiniti o creare nuovi box personalizzati.

Sono messi a disposizione dello sviluppatore anche strumenti di simulazione (*virtual robot*) che consentono di testare i comportamenti dei robot senza doverli eseguire fisicamente. Ciò facilita il processo di debugging e di ottimizzazione delle sequenze di azioni sviluppate.

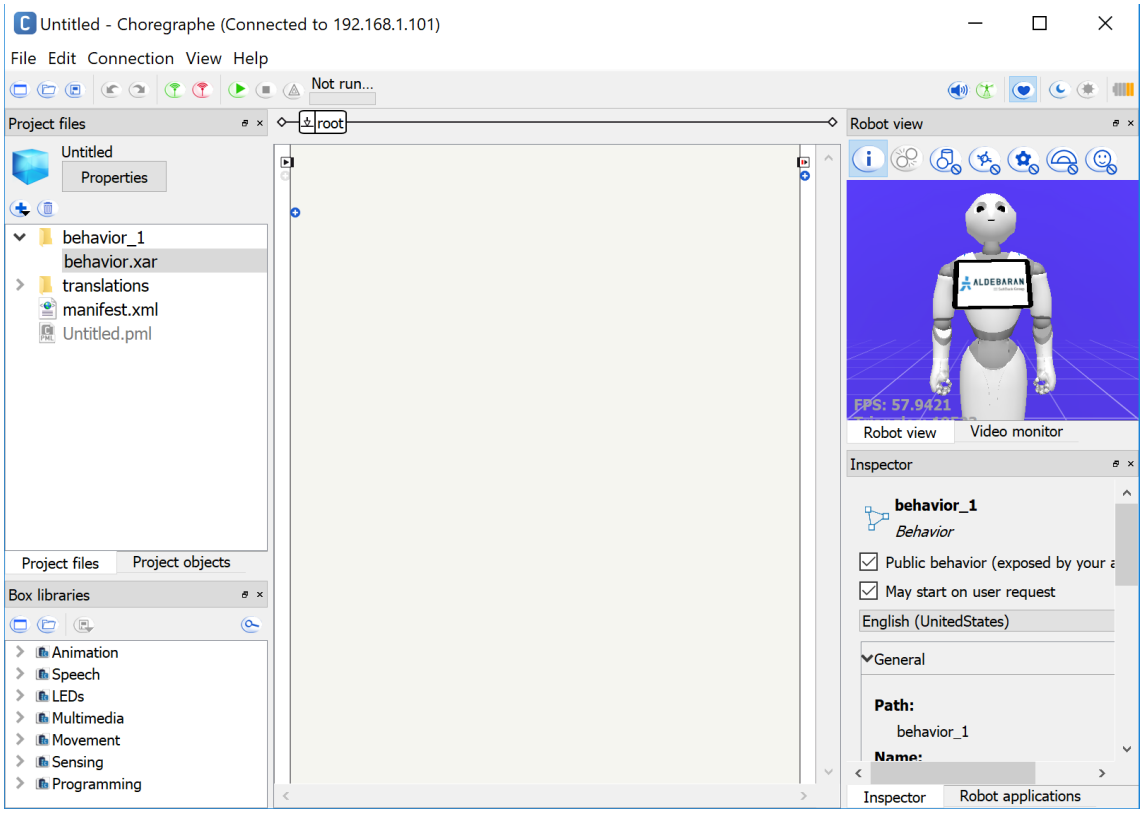


Figura 2.4: Interfaccia del software Aldebaran Choregraphe

Il software si basa su un'architettura di tipo *client-server*, in cui l'interfaccia utente e l'elaborazione dei comportamenti avvengono su un computer (*client*) e le istruzioni vengono inviate al robot (*server*) per l'esecuzione.

Per comunicare con il robot, Choregraphe utilizza un protocollo di comunicazione che consente di accedere alle funzionalità hardware dell'automa, come i sensori, i motori e i moduli di intelligenza artificiale, attraverso un'API completa e ben documentata. Questo permette agli sviluppatori di integrare facilmente i comportamenti creati con Choregraphe con altre applicazioni e sistemi come ad esempio *ROS* (Robot Operating Sistem), framework *open-source* utilizzato per lo sviluppo di applicazioni

robotiche.

In conclusione, l'intero ambiente di sviluppo si è rivelato essere un potente ed intuitivo strumento di programmazione visuale facilmente adattabile anche all' implementazione di comportamenti più complessi, creando esperienze interattive e coinvolgenti per gli utenti.

2.3 NAOqi OS

NAOqi OS è un sistema operativo sviluppato da *SoftBank Robotics* per i loro robot umanoidi NAO e Pepper. Si tratta di una distribuzione GNU/Linux basata su Gentoo[1] e sviluppata appositamente per le esigenze degli automi dell'azienda *Aldebaran*.

Figura 2.5: Logo di NAOqi OS

La particolarità di questo sistema operativo si ritrova nella sua architettura a componenti modulari. Ogni componente, chiamato "*module*", svolge un compito specifico, come il controllo del movimento, la percezione dei sensori o l'elaborazione del linguaggio naturale.

Questi moduli interagiscono tra loro attraverso un sistema di messaggistica per consentire la comunicazione e la cooperazione.

NAOqi OS utilizza un middleware chiamato "*ALBroker*" per gestire la comunicazione tra i moduli. L'*ALBroker* funge da intermediario tra i moduli, consentendo loro di pubblicare e sottoscriversi a *topic*, inviare e ricevere messaggi, e richiedere e fornire servizi.

[1] **Gentoo**: distribuzione Linux basata su sorgenti, che si distingue per il suo approccio altamente personalizzabile e orientato alle prestazioni.

Topic

Più nello specifico, con *topic* si intende un canale di comunicazione asincrona utilizzato per scambiare messaggi tra i diversi componenti di un sistema robotico. Un topic può essere considerato come un argomento di discussione su cui i nodi del sistema possono pubblicare e sottoscriversi per inviare e ricevere messaggi relativi a quell'argomento specifico.

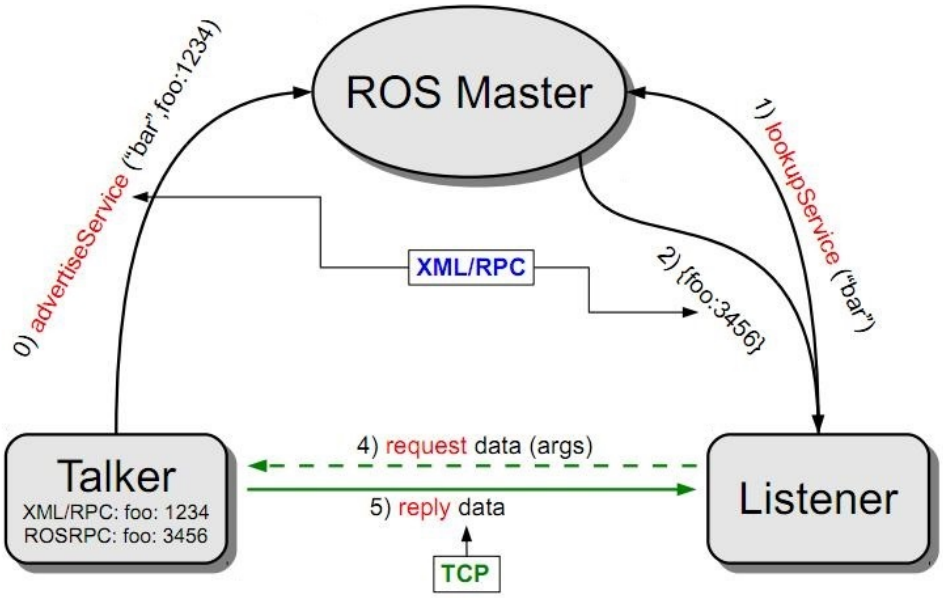


Figura 2.6: Esempio di comunicazione tra i nodi in ROS

I messaggi possono contenere dati di vario tipo, come informazioni provenienti dai sensori del robot, comandi di controllo, dati di stato, immagini, odometria (consente di stimare e tenere traccia della posizione e dell'orientamento di un robot in base alle informazioni dei suoi motori di movimento) e altro ancora.

Quando un nodo pubblica un messaggio su un topic, tutti i nodi che si sono sottoscritti a quel topic riceveranno il messaggio e potranno elaborarlo di

conseguenza. Ciò consente una comunicazione efficiente e disaccoppiata tra i diversi componenti del sistema robotico, in quanto i nodi possono inviare e ricevere dati senza la necessità di un collegamento diretto uno-a-uno.

Tornando a NAOqi OS, questo offre una serie di strumenti di sviluppo per facilitare la creazione di applicazioni robotiche. Questi strumenti includono l'ambiente di programmazione visuale Choregraphe, che consente di creare comportamenti per il robot trascinando e rilasciando blocchi di azioni, e l'API di sviluppo NAOqi SDK, che fornisce una vasta gamma di funzioni per interagire con i moduli e controllare il robot.

Queste funzioni vengono descritte di seguito:

- **Accesso ai sensori**: fornisce un'interfaccia per accedere ai sensori del robot, come telecamere, microfoni, sensori tattili e giroscopi. Si possono utilizzare queste informazioni sensoriali per creare comportamenti basati sulla percezione dell'ambiente circostante.

- **Controllo dei motori**: permette di controllare i motori del robot, consentendo di gestire il movimento e la postura degli attuatori. Ciò consente di creare movimenti fluidi e naturali per i robot umanoidi.

- **Gestione del suono**: offre funzionalità per la registrazione e la riproduzione dei suoni, consentendo ai robot di comunicare e interagire tramite l'emissione di suoni o la riproduzione di file audio predefiniti.

- **Riconoscimento vocale e linguaggio naturale**: sono inclusi moduli per il riconoscimento e la sintesi vocale, consentendo ai robot di comprendere e rispondere ai comandi vocali generando feedback verbali.

- **Intelligenza artificiale**: sono integrati moduli di intelligenza artificiale, come il riconoscimento facciale e il riconoscimento degli oggetti, che consentono ai robot di interagire con gli esseri umani e di analizzare l'ambiente circostante.

- **Comunicazione**: è supportata la comunicazione tra i robot stessi o con dispositivi esterni attraverso protocolli come TCP/IP, permettendo la collaborazione tra più robot o l'integrazione con altri sistemi.

Per concludere, questo sistema operativo embedded[2] è progettato per essere multi-piattaforma, per semplificare lo sviluppo di applicazioni robotiche e gestire comportamenti personalizzati, consentendo agli sviluppatori di concentrarsi sulla logica dell'applicazione senza doversi preoccupare dei dettagli implementativi e tecnici di basso livello.

[2]**Sistema operativo embedded**: sistema operativo ottimizzato per funzionare su hardware a risorse limitate, con restrizioni di memoria, potenza di calcolo e spazio di archiviazione. Caratterizzato da un basso consumo energetico, affidabilità e una richiesta minima di risorse di sistema rispetto alle distribuzioni *desktop* o *server*.

Capitolo 3

Applicazione Realizzata

Questo capitolo ha lo scopo di descrivere ed analizzare l'applicazione sviluppata per questo lavoro di tesi per il training di pazienti affetti da disturbi dello spettro autistico (DSA) e caratterizzata dalle potenzialità e risorse offerte dalle tecnologie introdotte nel capitolo precedente.

In particolare, per *training* si intende un insieme di attività volte al potenziamento delle abilità cognitive di un individuo[8]. Esse richiedono un serio e attento processo multidisciplinare che coinvolge diversi professionisti del settore medico e terapeutico, il cui obiettivo principale è valutare le abilità cognitive, comportamentali e sociali dei pazienti al fine di fornire una diagnosi accurata e un piano di trattamento personalizzato.

Inizialmente viene effettuata una valutazione da un team di specialisti che includono psicologi, psichiatri, neuropsichiatri infantili e terapisti occupazionali. Questa fase permette di raccogliere informazioni dettagliate sullo sviluppo del paziente, le sue abilità cognitive, il linguaggio, il comportamento sociale e le competenze motorie.

Sulla base delle informazioni raccolte nella valutazione iniziale, vengono selezionati i test più appropriati da sottoporre al paziente.

Questi test possono includere valutazioni standardizzate che misurano le abilità cognitive, il linguaggio, le abilità sociali e il comportamento adattivo dell'individuo.

Nello specifico sono stati realizzati ed implementati due test molto diffusi per il training di pazienti affetti da DSA:

- **Test sulla Discriminazione uditiva**

  Il test sulla discriminazione uditiva è un test per l'analisi delle funzioni linguistiche del paziente;

- **Test della Torre di Londra (TOL)**

  Il Test della Torre di Londra (TOL) è un test per l'analisi delle funzioni esecutive del paziente.

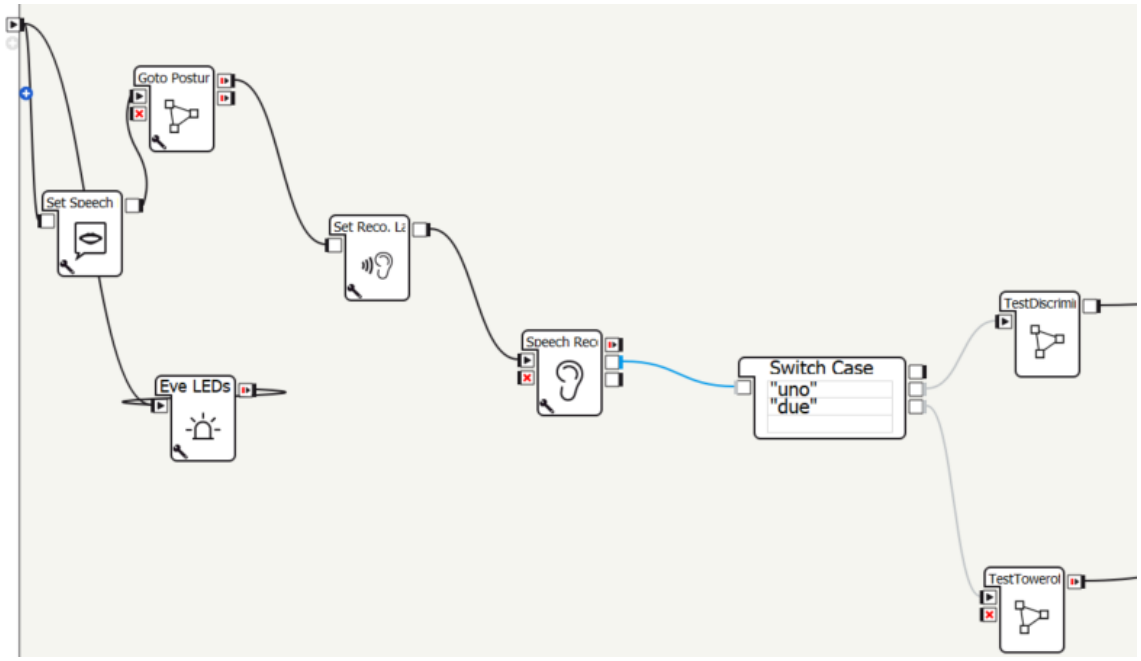


Figura 3.1: Panoramica dell' applicazione realizzata tramite il software Choregraphe

3.1 Discriminazione uditiva

La discriminazione uditiva è la capacità di percepire e riconoscere, attraverso l'udito, le caratteristiche che rendono diversi fra loro i suoni della propria lingua, e di attribuire ad essi un significato. Infatti, ogni fonema (ovvero ogni suono della lingua) si differenzia dagli altri per una o più caratteristiche, definite "tratti".

Molte condizioni, tra cui l'autismo, possono influenzare l'abilità dell'individuo di ascoltare e capire ciò che sente, ossia di comprendere, confrontare e distinguere i "suoni" del linguaggio parlato.

Il Test sulla Discriminazione uditiva di *Pinton* e *Zanettin* (1998) è utilizzato per valutare le abilità uditive e di elaborazione del suono in individui con disturbi dello spettro autistico. Nel caso particolare questo test è stato adoperato per valutare la capacità uditiva dell'individuo di distinguere tra coppie di *non parole*.

La prova è composta da 37 item, ciascuno dei quali rappresenta una coppia di non parole da discriminare. Per ogni coppia vengono comunicate le non parole al paziente, il quale ha il compito di riconoscere se queste risultano essere uguali o diverse tra loro.

Nel caso in cui la risposta fosse corretta, questo comporterebbe un aumento del punteggio del paziente (definito *score* e registrato per ogni paziente). Lo score di ogni paziente rappresenta una metrica fondamentale per il terapista, il quale dopo averlo analizzato può verificare la presenza di eventuali disturbi nelle funzionalità uditive del paziente.

3.1.1 Implementazione

Per l'implementazione di questo test sono stati utilizzati diversi box script Python in Choregraphe, i quali verranno analizzati di seguito:

- **Animated Say text**: box che consente a Pepper di emettere messaggi vocali animati. Questo box integra le funzionalità di sintesi vocale del robot, consentendogli di comunicare verbalmente con gli utenti in modo coinvolgente e realistico. Riceve in input un box testuale e lo trasforma in un messaggio vocale animato.

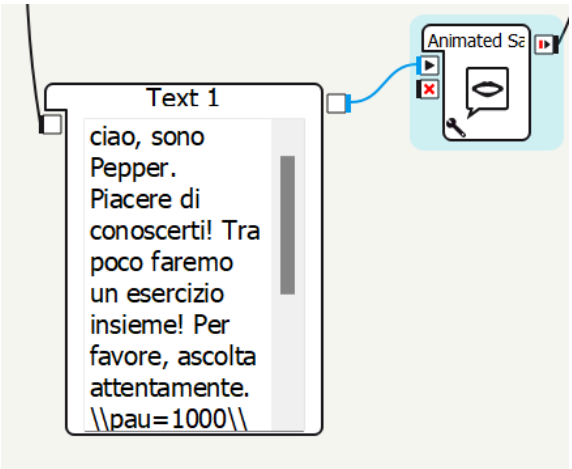


Figura 3.2: Box per la lettura di testo animato

- **Record Sound**: box che permette di registrare i suoni catturati dai microfoni direzionali di Pepper e salvarli sulla memoria temporanea di quest'ultimo.

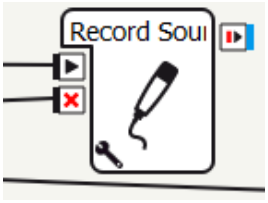


Figura 3.3: Box per la registrazione audio

- **Record Video**: box che permette di registrare video catturati dalle telecamere frontali posizionate negli occhi di Pepper e salvarlo sulla memoria temporanea di quest'ultimo.

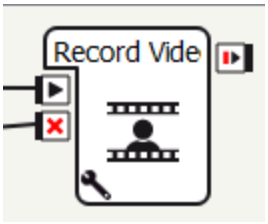


Figura 3.4: Box per la registrazione video

- **Face Tracker**: box progettato per rilevare e tracciare i volti umani nell'ambiente circostante il robot. Questo modulo consente a Pepper di identificare e seguire i volti umani, facilitando l'interazione e la comunicazione con gli utenti. Questo modulo utilizza una combinazione di algoritmi di visione artificiale e sensori per individuare i volti presenti nell'ambiente circostante. L'immagine acquisita viene quindi elaborata utilizzando algoritmi di elaborazione dell'immagine per individuare le caratteristiche facciali comuni associate ai volti umani, come gli occhi, il naso e la bocca. Questa elaborazione può includere tecniche come la segmentazione dell'immagine, la rilevazione dei contorni e il riconoscimento delle caratteristiche.

Una volta individuato il volto, il modulo Face Tracker traccia e segue i movimenti di quest' ultimo nel tempo. Questo consente a Pepper di mantenere una connessione visiva con le persone nell'ambiente e di rilevare eventuali cambiamenti di posizione o espressione facciale. Il robot quindi può mantenere il contatto visivo con una persona mentre si sposta o può indirizzare i suoi movimenti e le sue espressioni facciali verso un individuo specifico per creare una connessione personale.

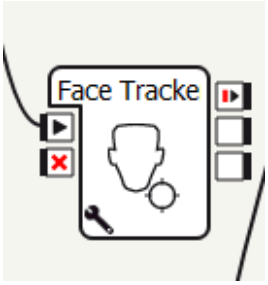


Figura 3.5: Box per il tracciamento del volto umano

- **Set Language**: box che permette di settare la lingua riconosciuta e parlata del robot.

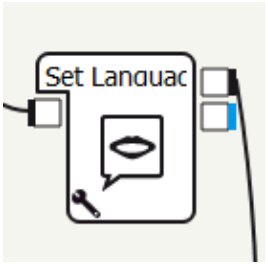


Figura 3.6: Box per settare la lingua riconosciuta dal robot

- **LRS** (lettura,riconoscimento vocale e incremento dello score): box che racchiude uno script python personalizzato e che permette al robot di leggere

le coppie di non parole all'utente, aspettare la risposta di quest' ultimo e verificarne la correttezza. Nel caso in cui la risposta fosse corretta, verrebbe incrementato lo score dell'utente, salvando il valore di quest' ultimo in un file esterno.

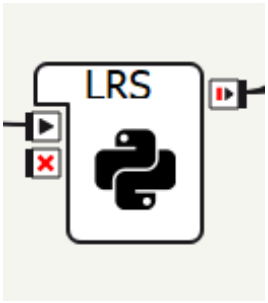


Figura 3.7: Box Python personalizzato

Di seguito viene riportato e analizzato il codice dello script Python personalizzato denominato LRS:

```
import time
import os

class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)
        self.memory = ALProxy('ALMemory')
        self.speech = ALProxy('ALSpeechRecognition')
        self.tts = ALProxy('ALTextToSpeech')
        self.score = 0
        self.parole =
["faratu","loli","nape","rapi","difi","fefe","nami","mero","rimo","nago",
"bimota","lebeto","falofa","bareda","ganeli","peze","numu","bica","cilo"]
```

Alla creazione della classe vengono inizializzati gli oggetti che verranno utilizzati per accedere alla memoria tramite modulo *ALMemory*, al riconoscimento della voce umana tramite modulo *ALSpeechRecognition* e per la lettura vocale di un testo tramite modulo *ALTextToSpeech*. Fatto ciò viene inizializzata la lista delle non parole che faranno parte del test.

```
def onLoad(self):
    self.speech.setLanguage("Italian")
    self.tts.setParameter("speed",70)
    self.speech.pause(True)
```

La funzione *onLoad* viene invocata al caricamento del box e permette di inizializzare alcuni parametri del robot. Viene impostata la lingua del componente di sintesi vocale (*self.speech*) su italiano e viene settato il parametro di velocità (*speed*) del componente di sintesi vocale (*self.tts*) che rappresenta la velocità di lettura di un testo di Pepper.

```
def onInput_onStart(self):
    self.score = 0
    self.speech.pause(False)
    i = 0
    j = 0
    for i in range(0,18):
        for j in range(0,18):
            self.askWordComparison(self.parole[i],self.parole[j])
            i += 1

            if i % 2 == 0:
                j += 1
            time.sleep(3)
```

La funzione *onInput onStart* viene invocata quando il box riceve un input esterno. In particolare il funzionamento di questa porzione di codice viene descritto di seguito. Per prima cosa viene inizializzata a 0 una variabile *score* che terrà traccia del punteggio del paziente durante il test. Successivamente viene richiamata l'esecuzione del componente di sintesi vocale e vengono inizializzati due indici, i e j, per riferirsi di volta in volta a due non parole dalla lista di non parole inizializzata in precedenza. Il ciclo *for* successivo permette di iterare, scorrendo la lista di non parole e creando delle coppie da comunicare al paziente per lo svolgimento del test.

A questo punto viene richiamata la funzione *askWordComparison* il cui funzionamento verrà analizzato successivamente. Al termine della funzione viene utilizzato il metodo *time.sleep()* per introdurre un ritardo di 3 secondi nel programma, così da creare una pausa tra le iterazioni del ciclo.

```
def askWordComparison(self,word1,word2):
    self.tts.say(word1 + " <break time='2000ms'/> "+ word2)
    time.sleep(3)
    self.tts.resetSpeed()
    self.tts.say("Le due parole sono uguali?")
    risposta = self.speech.recognize(["sì","no"],20000)

    if risposta == "sì" and word1 == word2:
        self.tts.say("Perfetto, andiamo avanti.")
        self.score += 1
    elif risposta == "no" and word1 != word2:
        self.tts.say("Perfetto, andiamo avanti.")
        self.score += 1
    elif risposta != "no" and risposta != "sì":
        self.tts.say("Ti ripeto le parole")
        self.askWordComparison(word1,word2)
    elif risposta == "no" and word1 == word2:
        self.tts.say("Perfetto, andiamo avanti.")
    elif risposta == "sì" and word1 != word2:
        self.tts.say("Perfetto, andiamo avanti.")
```

La funzione *askWordComparison* prende in input due parole (*word1* e *word2*) e svolge una serie di operazioni per confrontare la risposta del paziente, dopo averle comunicate a quest'ultimo.

Per prima cosa viene utilizzo il componente di sintesi vocale (*self.tts*) per riprodurre l'audio delle due non parole concatenate insieme, separandole da una pausa di 2000 millisecondi (2 secondi). Viene utilizzato il metodo *say* facendo si che Pepper pronunci le due non parole, separate da una pausa per scandirle correttamente, al paziente.

Fatto ciò Pepper porrà una domanda al paziente, chiedendo se le due non parole

siano uguali e resterà in attesa di una sua risposta.

Una volta ottenuta verrà analizzata per 5 casistiche:

1. Se la risposta dell'utente è "**sì**" e le due parole sono **uguali**, viene incrementato il punteggio del paziente in quanto la risposta risulta corretta e viene pronunciato il messaggio "*Perfetto, andiamo avanti.*"

2. Se la risposta dell'utente è "**no**" e le due parole sono **diverse**, viene incrementato il punteggio del paziente in quanto la risposta risulta corretta e viene pronunciato il messaggio "*Perfetto, andiamo avanti.*"

3. Se la risposta dell'utente non è né "**no**" né "**sì**", viene pronunciato il messaggio "*Ti ripeto le parole*" e la funzione *askWordComparison* viene richiamata nuovamente per ottenere una nuova risposta dall'utente.

4. Se la risposta dell'utente è "**no**" e le due parole sono **uguali**, non viene incrementato il punteggio del paziente in quanto la risposta non è corretta e viene pronunciato il messaggio "*Perfetto, andiamo avanti.*"

5. Se la risposta dell'utente è "**sì**" e le due parole sono **diverse**, non viene incrementato il punteggio del paziente in quanto la risposta non è corretta e viene pronunciato il messaggio "*Perfetto, andiamo avanti.*"

```
def saveScore(score,filepath):
    with open(filepath,'w') as file:
        s = "Punteggio test: {0}".format(score)
        file.write(s)
```

La funzione *saveScore* accetta due argomenti: **score**, che rappresenta il punteggio del paziente da salvare e **filepath**, che indica il percorso del file in cui il punteggio verrà salvato.

Più nello specifico, la funzione accede al file specificato dal **filepath** passato come argomento, crea una stringa formattata che include il punteggio e scrive questa stringa all'interno del file.

```
def onInput_onStop(self):
    self.speech.pause(True)
    self.saveScore(self.score,"/home/nao/Gianmarco/TestDU.txt")
    self.score = 0
```

Al termine dello script viene invocata la procedura *onInput onStop* che si occupa di attivare l'output del box, di richiamare la funzione *saveScore* per salvare il punteggio del paziente su un file esterno e di inizializzare il valore della variabile *score* a 0.

3.2 Torre di Londra (ToL)

Il Test della Torre di Londra (ToL) è un test cognitivo sviluppato negli anni ottanta del secolo scorso da *Tim Shallice* e *Rosaleen A. McCarthy*. Esso è utilizzato con bambini e adulti per misurare eventuali deficit cognitivi, che riducono la capacità del soggetto di pianificare, mantenere l'attenzione, decidere e risolvere problemi. Non è specificamente rivolto ai disturbi dello spettro autistico (DSA), ma può essere utilizzato per valutare le abilità cognitive di individui con DSA e altre condizioni. In particolare il test consiste in un supporto con tre paletti verticali di diverse altezze e un insieme di palline colorate (verde, rosso e blu).

Figura 3.8: Supporto fisico utilizzato durante la somministrazione del test

L'obiettivo del test è spostare le palline da una configurazione iniziale a una configurazione obiettivo, seguendo quattro regole:

1. **Prima regola**: è possibile può spostare una sola pallina alla volta.

2. **Seconda regola**: si può muovere una sola pallina da un paletto ad un altro, in modo tale da non consentire all'utente di posizionare le palline sul tavolo o averne in mano più di una alla volta.

3. **Terza regola**: è possibile inserire una sola pallina sul paletto più piccolo, due su quello intermedio e tre su quello più grande.

4. **Quarta regola**: bisogna attenersi al numero di spostamenti massimi consentiti per risolvere il problema.

Per poter calcolare il punteggio finale raggiunto dal paziente al termine del test, è necessario tenere in considerazione il numero di problemi risolti correttamente, il numero di mosse utilizzate per risolverli e la violazione delle regole esplicitate precedentemente.
Nel caso specifico dell'applicazione sviluppata, il test viene somministrato dal robot Pepper attraverso i seguenti passi:

- prima di iniziare il robot effettua un'introduzione al test e spiega le regole fornendo un esempio pratico visualizzato sul suo tablet, per assicurarsi che il paziente abbia compreso le istruzioni;

- dopo aver esplicitato le regole del test, il robot presenta la configurazione iniziale del problema e la configurazione obiettivo;

- il paziente viene invitato da Pepper a risolvere il problema nel minor numero possibile di mosse, senza limiti di tempo. Tuttavia, viene registrato il tempo impiegato per completare il test a scopo diagnostico;

- l'esaminatore registra il numero di mosse utilizzate dal paziente, l'eventuale violazione di regole e se il problema è stato risolto correttamente o meno;

- il test viene ripetuto per diverse configurazioni e nel caso specifico viene ripetuto per 12 configurazioni con difficoltà crescente.

3.2.1 Implementazione

Per l'implementazione di questo test sono stati utilizzati diversi box script Python in Choregraphe, i quali verranno analizzati di seguito:

1. **Animated Say text, Record Sound, Record Video, Set language e Face tracker**: già analizzati in precedenza.

2. **Show image**: box che permette di mostrare sul tablet di Pepper un'immagine specifica, recuperata dalla memoria interna del robot.

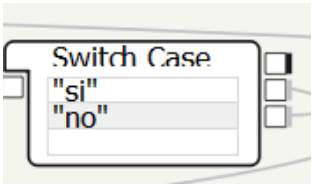
3. **Set Recognition Language**: box che permette di settare la lingua riconosciuta e compresa dal robot.

4. **Speech Recognition**: box di riconoscimento vocale che utilizza algoritmi e modelli di riconoscimento vocale avanzati per interpretare il parlato degli utenti.

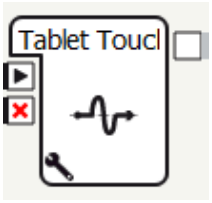
Per prima cosa il box cattura l'input audio proveniente dal microfono del robot o di un dispositivo esterno, come ad esempio un microfono collegato al computer in cui è in esecuzione Choregraphe.

L'audio viene successivamente pre-elaborato per migliorare la qualità del segnale e rimuovere eventuali rumori di fondo indesiderati. Infine l'audio pre-elaborato viene inviato all'algoritmo di riconoscimento vocale, che confronta il parlato con un modello linguistico e/o una serie di modelli acustici per determinare il testo corrispondente.

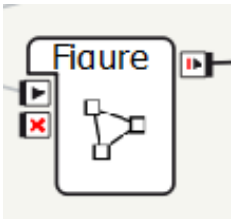
5. **Switch Case**: box che, dopo aver ricevuto la risposta dell'utente, stimola un solo output tra quelli disponibili. Utilizzato in questo caso per proseguire con la somministrazione del test, o ripetere le regole iniziali.



6. **Tablet touch**: box che invia un segnale nel momento in cui il tablet di Pepper viene toccato da un utente.



7. **Figure**: diagramma di box composto da molteplici box utilizzati per somministrare il test ToL.



Entrando nello specifico del box Tablet Touch questo presenta delle funzioni python personalizzate:

- **startTimer**: funzione che salva l'istante di inizio del test, per ogni figura, in una variabile timer.

```
def startTimer(self):
    self.timer = time.time()
```

- **stopTimer**: funzione che interrompe il timer e calcola la durata del tempo trascorso dal momento in cui la figura è stata mostrata al paziente, al momento in cui viene toccato il tablet del robot.

```
def stopTimer(self):
    if self.timer is not None:
        self.timer = time.time() - self.timer
```

- **saveTimerValue**: funzione che, dopo aver recuperato il valore della variabile *timer* (*self.timer*), lo formatta in minuti e secondi.

  Successivamente utilizza la funzione *open* per inserire i dati calcolati all'interno di un file, il cui filepath è passato in input alla funzione.

  Il file in questione presenta un'estensione **.csv**[1] e può essere scaricato dalla memoria del robot per poter essere analizzato successivamente.

```
    def saveTimerValue(self):
        if self.timer is not None:
            minutes = int(self.timer // 60)
            seconds = int(self.timer % 60)
            with open(self.path, "w") as csvfile:
                fieldnames =["Figure","Tempi"]
                writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
                writer.writeheader()
                writer.writerow({"Figure":"Figura 1", "Tempi": "{0} min e {1}
s".format(minutes,seconds)})
```

[1]**Estensione CSV (Comma-Separated Values)**: viene utilizzata per rappresentare dati in un formato tabellare, dove le colonne sono separate da virgole e le righe sono separate da nuove linee. È un formato molto comune per l'archiviazione e lo scambio di dati tra diverse applicazioni.

3.3 Risultati e obiettivi raggiunti

Questa sezione offre l'opportunità di riepilogare ed analizzare i principali risultati ottenuti e di sottolineare l'allineamento tra gli obiettivi prefissati e quelli raggiunti.

Considerando i due test analizzati in precedenza, nel caso del **test sulla Discriminazione uditiva** si è sviluppata una applicazione con comportamenti personalizzati del robot in grado di simulare la somministrazione della prova da parte di un terapista.

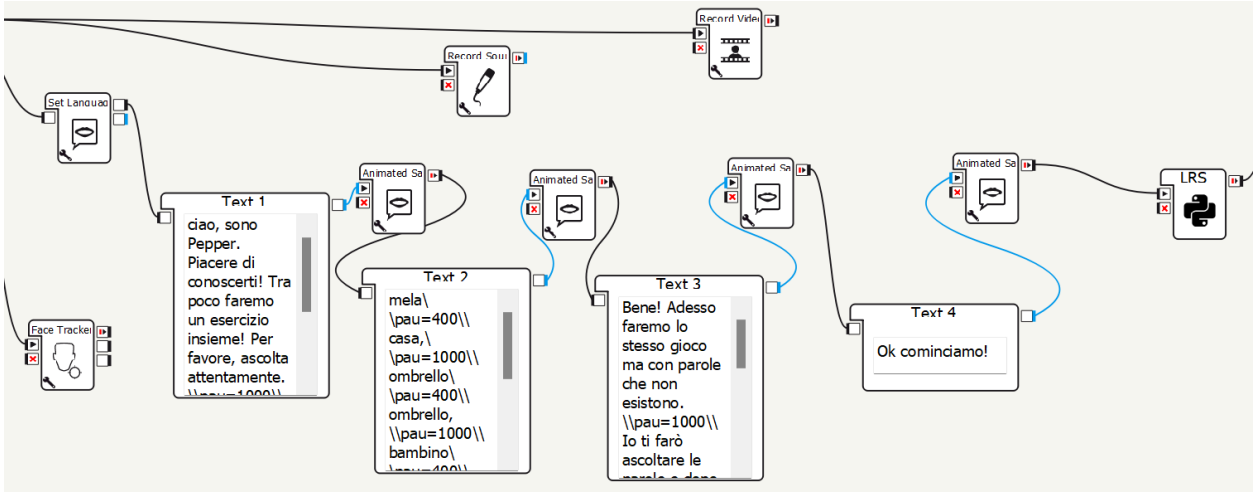


Figura 3.9: Panoramica sul test della Discriminazione uditiva

Più nello specifico, inizialmente Pepper effettua un' introduzione al test, spiegando dettagliatamente le regole di quest'ultimo, per poi comunicare al paziente le 37 coppie di *non parole*.

Il robot attende la risposta del paziente e aggiorna il suo score prima di aver comunicato la successiva coppia di non parole. Al termine del test si è in

grado di visionare un file di testo contente lo *score* del paziente, da analizzare successivamente.

Analizzando invece il **test della Torre di Londra (ToL)**, l'applicazione sviluppata permette anche in questo caso di simulare la somministrazione della prova da parte di un terapista, pur non sostituendo la sua valutazione.
Il robot dopo una prima fase nella quale fornisce al paziente una spiegazione delle regole, mostra una configurazione di prova da utilizzare per verificare la comprensione della struttura del test somministrandolo successivamente.

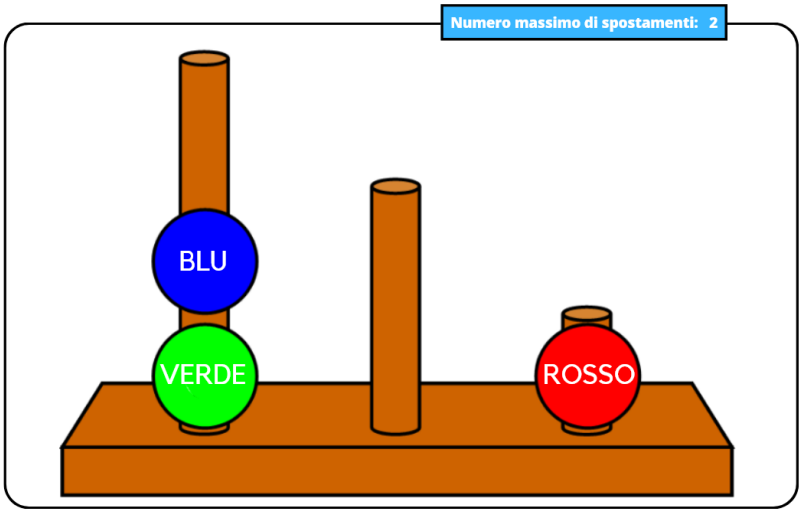


Figura 3.10: Esempio di configurazione che il paziente deve replicare (ToL)

A questo punto il paziente visualizza sul tablet di Pepper la configurazione da replicare, riproduce quella compresa e prosegue in questo modo fino ad esaurire tutte le configurazioni fornite dall'umanoide.

Al termine della prova, verranno analizzati dal terapista tutti i dati raccolti, salvati nel file cvs realizzato da Pepper e contenente i tempi di esecuzione del paziente per ogni singola configurazione.

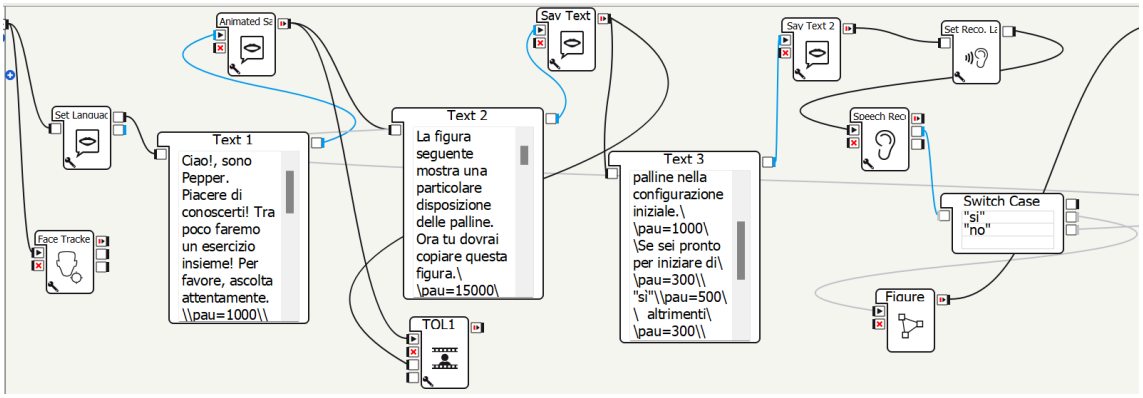


Figura 3.11: Panoramica sul test della Torre di Londra (ToL)

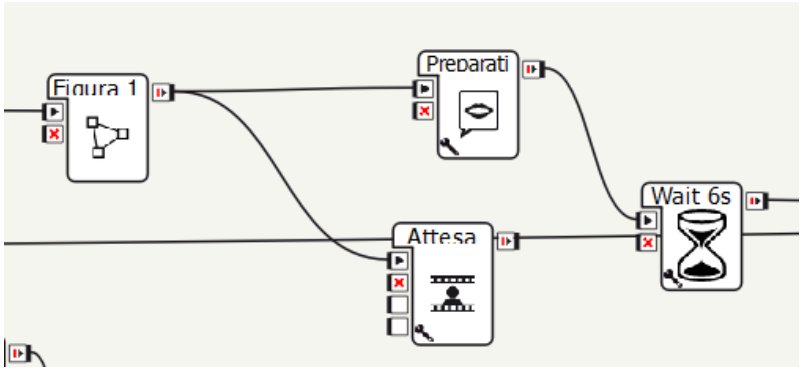


Figura 3.12: Implementazione della visualizzazione delle figure sul tablet di Pepper

Capitolo 4

Sviluppi Futuri

4.1 Introduzione

Nel corso dell'ultimo decennio sono stati condotti numerosi studi sull'applicazione della robotica nel campo della disabilità. La terapia assistita da robot è un campo di ricerca ed applicazione promettente [9], un'altra delle sfide che tecnologia e clinica affrontano insieme allo scopo di supportare i pazienti. Una parte rilevante delle attuali ricerche si è focalizzata sull'utilità delle tecnologie robotiche nello stimolare le abilità deficitarie nei Disturbi dello Spettro Autistico, dando conferma di come i robot umanoidi possano fornire un valido aiuto nel creare una comunicazione con il paziente, favorendone l'attenzione e generando nuovi comportamenti sociali.
Dagli studi che si sono succeduti negli anni, è emerso che i robot umanoidi, senza mai sostituirsi all'essere umano, migliorano le competenze sociali dei pazienti, per i quali l'interazione con le altre persone è un elemento che disorienta, anche a causa della variegata espressività del volto umano; interagendo con una persona, i pazienti affetti da DSA vengono a contatto con numerosi segnali sociali (le espressioni facciali, i gesti, la tonalità della voce) per loro difficili da interpretare.

Il robot diventa quindi una sorta di intermediario, affidabile e prevedibile, uno strumento conoscitivo per apprendere maggiori informazioni sul funzionamento della mente dei pazienti e di supporto per un percorso diagnostico e terapeutico completo.

Figura 4.1: Robot assistenziali utilizzati nel reparto di pediatria dell'ospedale di Padova

Nel prossimo futuro, si prevede che l'Intelligenza Artificiale parteciperà in modo progressivo ai processi decisionali in ambito sanitario e i professionisti del settore utilizzeranno sempre più dispositivi di intelligenza artificiale e apprendimento automatico per migliorare la precisione nella diagnosi e identificare i regimi terapeutici.

Tuttavia, pur offrendo numerose possibilità anche molto promettenti, l'adozione dei robot di assistenza rimane ancora inferiore al previsto, per un gap tra tecnologia e assistenza sanitaria. Gap che non deriva esclusivamente dalle attuali strategie per l'implementazione dei robot, ma è piuttosto un gap informativo nelle sezioni trasversali della progettazione tecnologica, della sanità e della società.

E' necessaria una diffusione delle conoscenze che favorisca l'interazione e la condivisione delle informazioni tra tutte le parti interessate coinvolte nella gestione dei robot per la diagnosi e cura dei DSA. La progettazione e l'implementazione di un robot richiedono competenze multidisciplinari provenienti da settori quali l'ingegneria, l'informatica, la psicologia e la riabilitazione clinica e la collaborazione tra i professionisti delle diverse discipline può portare ad un numero maggiore di contributi in questa area di ricerca con conseguenti ricadute positive sui pazienti.

Rimangono importanti i punti a favore della terapia assistita da robot. Essendo trattamenti informatizzati, è possibile tracciare e monitorare i progressi e gli step del trattamento, senza la necessità di affidarsi alle tecniche di osservazione tradizionali, personalizzando e adattando i percorsi alle specifiche esigenze del singolo paziente.

I disturbi dello spettro autistico si caratterizzano per la grande variabilità che c'è tra un soggetto e l'altro e la possibilità di avere strumenti condivisi ma personalizzabili è sicuramente di grande vantaggio per i clinici. Una delle sfide più ardue è quella di

rendere le soluzioni tecnologiche offerte dalla robotica assistiva usabili ed accettabili per i soggetti che dovrebbero trarne beneficio. Questo passaggio è tutt'altro che scontato. In questo senso, un approccio che nasce nel mondo dell'architettura e del design, ma che tende progressivamente ad estendersi alla progettazione di soluzioni tecnologiche ovvero di servizi integrati con soluzioni tecnologiche, è il cosiddetto *Human Centered Approach*. Il punto cardine di questo modello, come evidenziato dal nome, consiste nel mettere al centro della progettazione la persona e i suoi bisogni, per far sì che i desideri e le aspettative da questa espressi siano la linea guida per lo sviluppo del prodotto o servizio. Infine, esso mira ad includere la prospettiva di tutti coloro che possono avere un interesse nella realizzazione e ad assumere quindi non solo il punto di vista dell'utente finale ma anche quello di eventuali operatori, manutentori, specialisti, etc.

4.2 Funzionalità future dell' applicazione

Attraverso l'impiego di tecnologie avanzate e l'applicazione di principi psicologici specifici, definiti anche grazie alla collaborazione con professionisti del settore, è stato possibile realizzare uno strumento accessibile, intuitivo e personalizzabile.
Al momento sono in fase di studio e realizzazione nuove possibili funzionalità dell'applicazione, in particolare una dashboard e un database dedicato, come di seguito descritte:

1. **Dashboard**: interfaccia *user-friendly* di supporto ai clinici per avere accesso immediato alla raccolta di tutti i test da somministrare ai pazienti e per monitorarne i progressi durante la terapia nel corso del tempo.

2. **Database dedicato**: organizzazione strutturata delle informazioni raccolte da Pepper, per consentirne il salvataggio, la visualizzazione e un facile reperimento da parte degli utenti.
   Il database da utilizzare memorizzerà le seguenti informazioni:

   - Parametri facciali univoci associati ad un dato paziente e restituiti dal modulo di riconoscimento facciale del robot.

   - Informazioni anagrafiche e sanitarie, contenenti la storia clinica di ogni paziente.

   - Registrazione video delle sedute di somministrazione dei test, identificate dal codice univoco di ogni paziente e dalla data di acquisizione.

- Registrazione audio delle sedute di somministrazione dei test, identificate dal codice univoco di ogni paziente e dalla data di acquisizione.

Capitolo 5

Conclusioni

Negli ultimi anni la robotica sociale assistita (RSA) è stata oggetto di studio scientifico e di applicazione clinica nel trattamento e nella diagnosi del Disturbo dello Spettro Autistico (DSA). Gli studi in questo settore hanno dimostrato che l'uso della RSA può migliorare l'attenzione e il coinvolgimento dei pazienti affetti da disturbi dello spettro autistico, promuovere l'interazione sociale e comunicativa e supportare i professionisti del settore sanitario.

In questo lavoro di tesi è stata presentata e descritta un'applicazione sviluppata per la somministrazione attraverso il robot umanoide *Pepper - SoftBank Robotics*™ di test per l'analisi delle funzioni cognitivo-linguistiche in individui affetti da disturbi dello spettro autistico.

L'applicazione sviluppata si è dimostrata efficace nel facilitare la somministrazione dei test e la raccolta dei risultati ottenuti.

Essa consente un certo grado di standardizzazione delle procedure e un'analisi più veloce e dettagliata dei dati, generalmente collezionati solo in forma cartacea.

Ci si augura che gli sviluppi futuri previsti siano in grado nel breve tempo di rendere l'applicazione ancora più efficiente al fine di una valutazione sempre più

accurata delle funzioni cognitive-linguistiche dei pazienti, facilitando la gestione e l'elaborazione di informazioni utili alla diagnosi, alla pianificazione delle terapie e al monitoraggio dei loro progressi nel tempo.

Ringraziamenti

Al termine del lavoro di tesi, desidero dedicare questo spazio per esprimere la mia gratitudine a tutte quelle persone che hanno contribuito al raggiungimento di questo mio primo traguardo.

In primo luogo desidero ringraziare di cuore il mio relatore, la Professoressa Mariacarla Staffa, il suo supporto costante, la sua competenza e la sua guida sono stati fondamentali per la realizzazione di questa tesi. Desidero ringraziarla soprattutto per avermi trasmesso la sua passione per gli argomenti trattati e per essere stata una presenza stimolante durante il mio percorso accademico. Auguro a tutti i miei colleghi di incontrare un docente come lei, in grado di tenere viva e alimentare la "fiamma" della conoscenza.

Un ringraziamento speciale va ai miei genitori, per avermi sempre sostenuto in ogni fase della mia vita. Grazie per tutti i vostri sacrifici, per aver sempre creduto in me e per avermi incoraggiato a perseguire i miei obiettivi.

Ringrazio mio fratello Stefano, per il suo sostegno incondizionato durante tutto il mio percorso accademico. La sua voglia di fare e la sua saggezza sono per me una guida preziosa e un esempio da seguire. Grazie per essere sempre al mio fianco.

Non posso non ringraziare la mia ragazza, Emilia, per il suo amore incondizionato

e per avermi sempre spronato a dare il meglio di me. La sua forza, la sua determinazione e il suo modo di affrontare la vita, sempre con il sorriso, sono per me grande fonte di ispirazione. Il raggiungimento di qualsiasi obiettivo non sarebbe lo stesso senza la sua presenza al mio fianco.

Infine, desidero esprimere profonda gratitudine al mio migliore amico, Davide, un compagno fedele e un vero fratello per me. Le risate, le avventure e i ricordi che abbiamo condiviso rappresentano un bagaglio personale di inestimabile valore. La sua amicizia è stata un supporto fondamentale durante il mio percorso personale e accademico. Mi auguro di continuare a sostenere reciprocamente i nostri sogni e le nostre ambizioni future.

Concludo ringraziando tutti gli amici e i compagni di corso, per gli scambi di idee e il sostegno reciproco, che hanno reso questi anni universitari più ricchi e significativi.

A tutti voi, rinnovo il mio più sincero grazie.

Bibliografia

[1] Amirova A. Rakhymbayeva N. Zhanatkyzy A. Telisheva Z. Sandygulova A. *Effects of Parental Involvement in Robot-Assisted Autism Therapy. Journal of Autism and Developmental Disorders*, 53(1):438–455, 2023 Jan. doi:10.1007/s10803-022-05429-x.

[2] Mataric M. Clabaugh C. *Escaping Oz: Autonomy in Socially Assistive Robotics. Annu. Rev. Control Robot. Auton. Syst.* 2:33–61, 2019. doi:10.1146.

[3] van den Berk-Smeekens. et al. *Adherence and acceptability of a robot-assisted Pivotal Response Treatment protocol for children with autism spectrum disorder.* 10:11, 2020. doi:10.1038.

[4] Marino F. et al. *Outcomes of a Robot-Assisted Social-Emotional Understanding Intervention for Young Children with Autism Spectrum Disorders. Journal of Autism and Developmental Disorders*, 50:1973–1987, 2020. doi:10.1007.

[5] Szymona B. *Robot-Assisted Autism Therapy (RAAT). Criteria and Types of Experiments Using Anthropomorphic and Zoomorphic Robots. Review of the Research. Sensors*, 21(11):3720, 2021 May 27. doi:10.3390.

[6] Somchaya Liemhetcharat Mike Beiter, Brian Coltin. *An introduction to robotics with NAO.* URL: http://www.kramirez.net/Robotica/Material/Nao/AnIntroductionToRoboticsWithNao_TextBook_2012_US.pdf.

[7] Aldebaran. *SoftBank Robotics documentation*. URL: http://doc.aldebaran.com/2-5/index.html.

[8] Ghiglino D. Chevalier P. Floris F. Priolo T. Wykowska A. *Follow the white robot: Efficacy of robot-assistive training for children with autism spectrum condition. Research in Autism Spectrum Disorders*, 86, 2021 Aug. doi:10.1822.

[9] Alabdulkareem A. et al. *A Systematic Review of Research on Robot-Assisted Therapy for Children with Autism. Sensors*, 22(3):944, 2022 Jan 26. doi:10.3390.