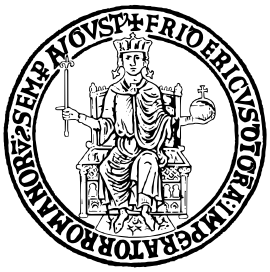
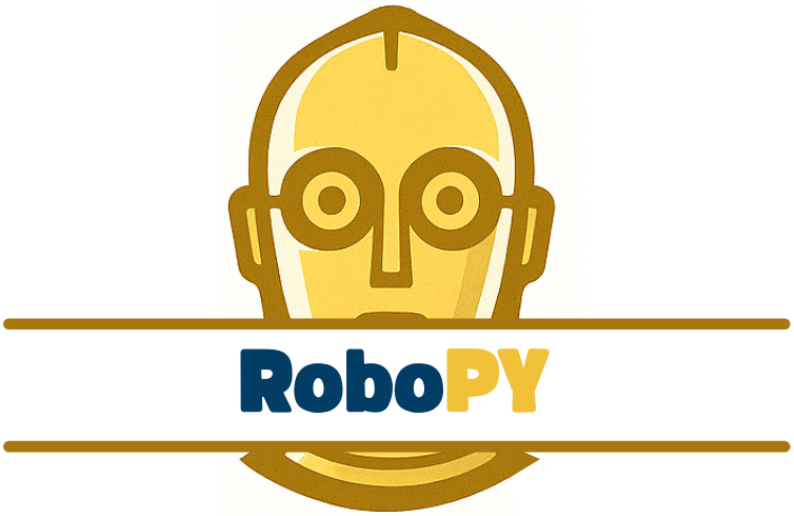
Università degli Studi di Napoli Federico II



Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione

Corso di Laurea Triennale in Informatica



Autori
Andrea Campanile Sidero *N86004091*
Giovanni del Gaudio *N86004054*

Anno Accademico 2024–2025

Indice

Introduzione

Analisi

Il progetto RoboPY ha l'obiettivo di sviluppare un sistema distribuito per l'interazione uomo-robot, basato su architettura client-server. Il sistema è composto da un server scritto in linguaggio C e da un client sviluppato in Python, che comunicano tra loro tramite socket TCP. Il client funge da intermediario tra una persona e un robot sociale (Furhat), raccogliendo input attraverso un'interfaccia dialogica e trasmettendo le informazioni al server, il quale elabora il profilo della personalità dell'interlocutore sulla base del questionario **TIPI** (Ten-Item Personality Inventory).

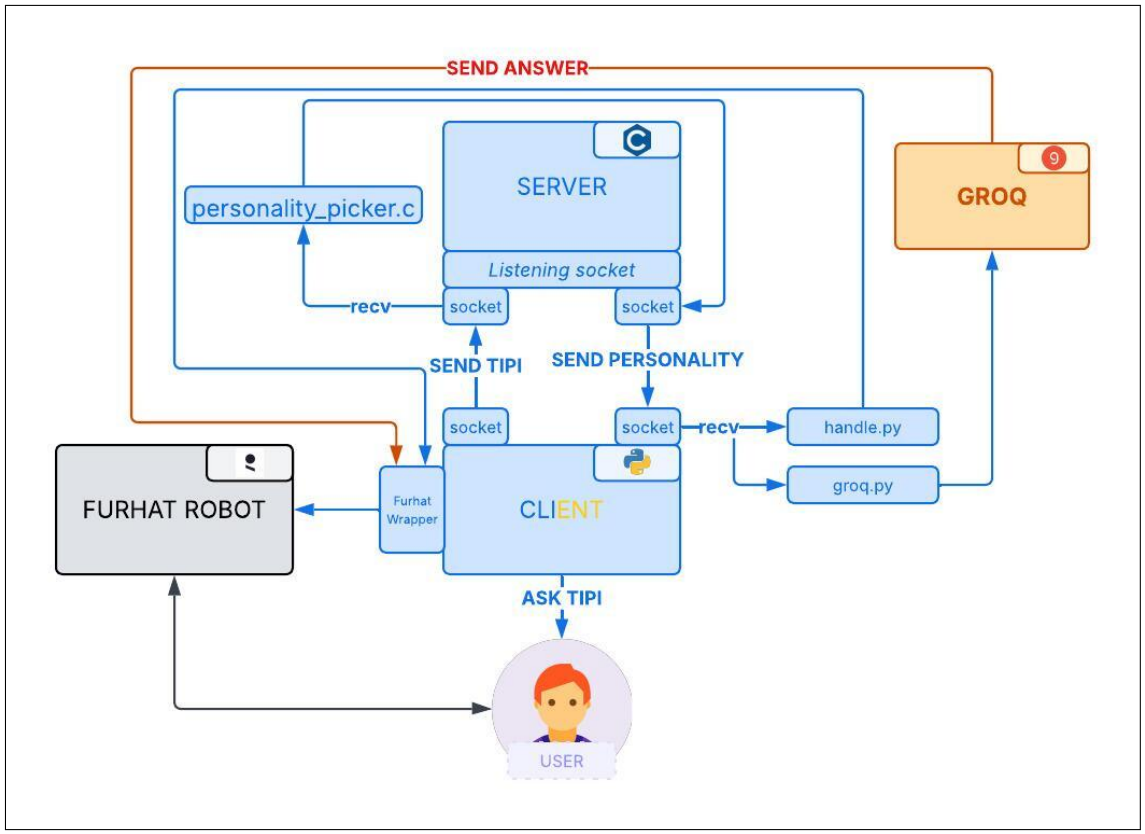


Figura 1: Schema concettuale che ha guidato lo sviluppo del software

–1–
Client

CONTENTS: **1.1 Scelte progettuali. 1.2 Fasi del Robot.** 1.2.1 Prima fase: *greeting* – 1.2.2 Seconda fase: *conversation* – 1.2.3 Terza fase: *conclusion* – 1.2.4 *Idle*. **1.3 TIPI. 1.4 Comunicazione con il Server. 1.5 Interazione con Groq. 1.6 Whisper. 1.7 Gesture.**

1.1 Scelte progettuali

Il Client del sistema RoboPY è sviluppato in linguaggio **Python** e rappresenta la componente dell'architettura responsabile della comunicazione diretta con l'utente e con il robot Furhat. Le sue funzionalità principali includono:

- la gestione del dialogo uomo-macchina tramite input vocali;
- la somministrazione del questionario **TIPI** per l'analisi della personalità;
- l'invio dei dati raccolti al server tramite **socket TCP**;
- la gestione della conversazione tramite **Groq**;
- la ricezione delle istruzioni dal server e la loro conversione in comandi REST verso **Furhat**.

Il Client utilizza l'**SDK Python** di Furhat e si occupa dell'interazione ad alto livello con il robot, lasciando al server la logica decisionale basata sull'analisi della personalità.

1.2 Fasi del Robot

Nel corso dello sviluppo sono state individuate **quattro** fasi fondamentali che descrivono in modo strutturato l'interazione tra l'utente e il robot. Ciascuna fase rappresenta un momento chiave. Questa suddivisione in fasi ha guidato la progettazione modulare dell'interazione, permettendo di dividere correttamente i compiti e garantendo che questi vengano svolti nei momenti opportuni.

1.2.1 Prima fase: *greeting*

In questa fase, il Furhat Robot non ha ancora nessuna conoscenza dell'utente, e per tale motivo mantiene un comportamento neutro fino al passaggio della fase successiva. Il principale compito che deve essere svolto è quello di sottoporre l'utente al questionario TIPI. Una volta completata tale task, si passa direttamente alla fase successiva.

1.2.2 Seconda fase: *conversation*

La seconda fase inizia nel momento esatto in cui arriva dal server la personalità calcolata in base alle risposte fornite al TIPI. Il compito del Robot è adesso quello di intrattenere una conversazione con l'utente, rispettando il carattere. Il funzionamento del processo è spiegato nel dettaglio nel paragrafo «**Interazione con Groq**».

1.2.3 Terza fase: *conclusion*

La terza fase, decisamente più piccola rispetto alle precedenti, gestisce la fine della conversazione, laddove l'utente abbia deciso di terminarla. Anche in questo caso il Robot deve gestire la situazione concordamente alla personalità rilevata.

1.2.4 *Idle*

Infine, la fase *idle* rappresenta i momenti in cui il robot non è impegnato attivamente nel dialogo. In questi intervalli, è comunque importante che il robot mantenga dei comportamenti di attesa (idle) coerenti con il profilo di personalità dell'utente, al fine di preservare la continuità comunicativa e l'effetto di presenza.

1.3 TIPI

Per la rilevazione della personalità dell'utente è stato utilizzato il questionario **TIPI** (**Ten Item Personality Inventory**), uno strumento psicometrico breve e validato per la valutazione dei cinque tratti principali del modello dei **Big Five**:

1. Estroversione;
2. Amicalità;
3. Coscienziosità;
4. Stabilità;
5. Emotività;
6. Apertura mentale.

L'utente è invitato a rispondere a *dieci* affermazioni, valutando il proprio grado di accordo su una scala Likert da 1 (fortemente in disaccordo) a 7 (fortemente d'accordo). Le risposte

vengono poi elaborate secondo il metodo standardizzato del TIPI, in cui ogni tratto è calcolato come somma di una domanda diretta e dell'inverso di una domanda opposta, al fine di ottenere un profilo sintetico ma rappresentativo della personalità.

1.4 Comunicazione con il Server

La comunicazione con il server avviene tramite una connessione socket **TCP**, che consente di stabilire un canale diretto e persistente per lo scambio di dati. Il client si connette all'indirizzo IP e alla porta del server, quindi riceve una risposta strutturata in formato **JSON** contenente informazioni rilevanti come una base di prompt e un set di tratti di personalità (traits). Questi dati vengono decodificati e trasformati in strutture dati interne, che il client utilizza per guidare il proprio comportamento. In particolare, in base ai tratti ricevuti, il client richiama specifiche funzioni associate a ciascuna caratteristica, permettendo così una personalizzazione dinamica e contestuale della risposta o dell'interazione. Questo approccio assicura che il client possa adattarsi in modo flessibile alle informazioni fornite dal server, mantenendo una comunicazione efficiente e orientata all'esperienza utente.

1.5 Interazione con Groq

L'integrazione con il servizio di intelligenza artificiale fornito da **Groq** è stata implementata utilizzando l'endpoint dedicato alla generazione di risposte testuali. Il modulo incaricato della conversazione adotta lo stesso approccio chat-based, costruendo dinamicamente il prompt iniziale in base ai tratti della personalità dell'utente, successivamente incorporati nel messaggio di sistema. La struttura dei messaggi segue il formato JSON. L'utilizzo delle variabili d'ambiente garantisce una gestione sicura delle credenziali di accesso.

1.6 Whisper

Il progetto utilizza *Whisper*, il modello open-source di riconoscimento vocale sviluppato da OpenAI, per trascrivere l'audio in tempo reale nel container client. Whisper è stato scelto per la sua elevata accuratezza, anche in presenza di accenti, rumore di fondo e frasi brevi. Il modello viene inizialmente caricato in memoria una sola volta per migliorare le prestazioni e viene utilizzato per trascrivere segmenti audio registrati tramite microfono. L'integrazione di Whisper consente all'utente di interagire vocalmente con il sistema, fornendo risposte al questionario TIPI, che il client interpreta e invia al server per l'analisi. L'intero processo è gestito in locale, **garantendo privacy e bassa latenza**.

1.7 Gesture

La comunicazione non verbale del robot è gestita attraverso un set di gesture definite in formato JSON e associate a tratti della personalità. Ogni tratto, come ad esempio *Estroverso*, *Introverso*, *Amichevole*, o *Impulsivo*, corrisponde a una particolare configurazione parametrica dei movimenti facciali del robot Furhat. Le gesture vengono selezionate dinamicamente da una mappa di associazione tra tratti e funzioni (`traits_function_map`), e successivamente eseguite in modo asincrono tramite un ciclo che ne randomizza l'ordine di attivazione. Il tratto *Instabile* rappresenta un caso particolare, poiché comporta la selezione casuale di una gesture tra quelle disponibili, simulando un comportamento più irregolare e imprevedibile. L'approccio consente di espandere facilmente il repertorio gestuale del robot, favorendo l'adattabilità del sistema a diversi profili psicologici e migliorando l'efficacia dell'interazione umana.

–2–
Server

2.1 Scelte implementative

Il Server è stato implementato in **C**. È il *core* elaborativo del sistema, che si occupa di calcolare il punteggio in base alle risposte del TIPI, e restituire la personalità al Client.

2.2 Architettura generale

L'infrastruttura si basa sull'utilizzo della libreria `<sys/socket.h>` e sulla famiglia di indirizzi IPv4 (`AF_INET`) con protocollo `SOCK_STREAM`, che garantisce una comunicazione affidabile di tipo stream (TCP). Il socket principale viene inizializzato e associato alla porta 8080 tramite il costrutto `bind()`, e successivamente messo in ascolto con `listen()`. Il server entra quindi in un ciclo infinito in cui accetta connessioni in ingresso tramite la funzione `accept()`. Una volta stabilita la connessione con un client, il server riceve un messaggio formattato in **JSON**, che contiene i risultati del questionario TIPI. Questo viene memorizzato in un buffer, terminato con `'\0'`, e successivamente analizzato. Il contenuto viene passato al modulo `personality_picker.c` per l'analisi della personalità.

2.3 Calcolo della personalità

I dati vengono passati tramite stringhe JSON, contenente i punteggi ottenuti dall'utente per ciascuna delle cinque dimensioni previste dal questionario TIPI: estroversione, amicalità, coscienziosità, stabilità emotiva e apertura mentale. Il modulo `personality_picker.c` si occupa del parsing del messaggio JSON in ingresso tramite la libreria `cJSON`. Ogni valore numerico viene validato e successivamente interpretato in base a una soglia predefinita: se il punteggio è maggiore o uguale a **8**, il tratto viene considerato "positivo" e viene descritta la corrispondente caratteristica psicologica. In caso contrario, si assegna l'etichetta opposta.

2.4 Comunicazione con il client

Il server si occupa di generare una risposta in formato JSON che include un prompt base e un array di tratti della personalità dell'utente. Dopo aver analizzato i dati ricevuti e deciso la classificazione dei tratti, viene creato un oggetto JSON principale con `cJSON_CreateObject()`. A questo oggetto viene aggiunta una stringa fissa che rappresenta il prompt base, quindi, se la determinazione della personalità ha successo, l'array di tratti viene inserito come elemento con chiave "personalità". In caso contrario, viene aggiunto un campo "error" contenente un messaggio descrittivo. Infine, l'oggetto JSON completo viene convertito in una stringa non formattata tramite `cJSON_PrintUnformatted()`, pronta per essere inviata al client, garantendo così una comunicazione strutturata e facilmente interpretabile.

2.4.1 Parallelismo di client

Poiché il compito del server è limitato a ricevere un JSON, elaborarlo rapidamente e inviare una risposta, si è optato per una gestione sequenziale delle connessioni senza l'utilizzo di threading. Questo approccio è stato ritenuto adeguato in virtù della semplicità del task. L'eliminazione della complessità legata ai thread rende il server più semplice da mantenere e meno soggetto a errori di concorrenza, pur garantendo prestazioni adeguate per il contesto applicativo.

–3–
Docker

3.1 Containerizzazione

Nel progetto sia il client che il server sono stati dockerizzati, al fine di garantire portabilità, isolamento e semplicità di esecuzione su qualsiasi sistema. Attraverso la creazione di specifici Dockerfile, entrambi i componenti vengono eseguiti in ambienti controllati, contenenti tutte le dipendenze necessarie. Questa scelta ha permesso di evitare problemi legati alla configurazione del sistema host, rendendo lo sviluppo più coerente tra diversi dispositivi e facilitando la fase di testing. L'utilizzo di **Docker** ha inoltre favorito una maggiore automazione nell'esecuzione dell'intero sistema, consentendo di avviare client e server in modo rapido e affidabile, migliorando la manutenibilità complessiva dell'architettura.

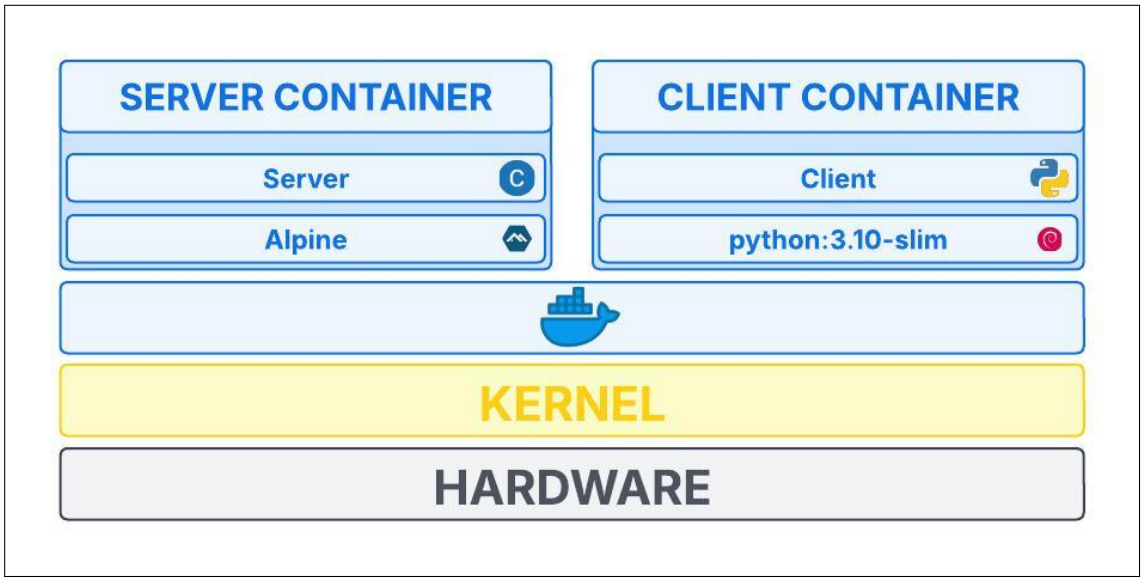


Figura 3.1: Mai più: "It works on my machine"

3.2 Docker-compose

Il progetto utilizza **Docker Compose** per semplificare l'orchestrazione dei servizi coinvolti nell'architettura client-server. Attraverso il file `docker-compose.yml`, vengono definiti e avviati in modo coordinato:

- un **container client**, responsabile dell'interfaccia vocale, del collegamento con il robot Furhat tramite SDK e della trascrizione vocale tramite *Whisper*;
- un **container server**, che riceve le risposte del questionario, elabora la stima dei tratti di personalità secondo il modello TIPI e la invia al client.

Docker Compose consente di definire facilmente le dipendenze, le variabili d'ambiente, le porte esposte e la rete condivisa tra i container, garantendo un ambiente riproducibile e coerente.