

UNIVERSITÀ
degli STUDI
di CATANIA

DIPARTIMENTO DI INGEGNERIA
ELETTRICA ELETTRONICA E INFORMATICA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Lorenzo Siena

A CYBERCAR STUDY
FROM THE EDGE TO THE CLOUD

Relatore:
Prof. Davide Patti

Anno Accademico 2023/2024

*A mia madre per la fiducia.
A mio padre per la pazienza.
E un grazie a chi ci ha creduto
ed è rimasto fino alla fine.*

Abstract

La tesi affronta e verifica lo stato dell'arte per l'iot, sperimentando e prototipando una Cybercar con un assistente vocale presente sulla macchina, in edge per il keyword spotting, e in cloud per l'host di un chatbot AI open source, multilingua e personalizzabile su una GPU consumer (GTX 1080 Ti), in grado di interagire con l'utente vocalmente e nell'ipotesi della connessione col bus interno dell'auto capace di percepire lo stato dell'auto.

Viene presentato un proof of concept, un prototipo e un caso di studio, oltre a tutta la documentazione hardware e software necessaria. Lo studio mostra che è possibile con le tecnologie attuali rendere smart una comune automobile, fino a un livello e una profondità che dipende dall'accesso alle dinamiche interne dell'auto, accessibile dalla porta di debug OBD-II e usando i protocolli disponibili in bus.

Viene inoltre fatta una considerazione sul Vehicle-to-everything (V2X) e su alcuni scenari di implementazione.

INDICE

Abstract.....	4
INTRODUZIONE.....	1
1. Struttura del progetto.....	5
1.1 Architettura.....	5
Disegni e schemi.....	5
Edge.....	7
Il sistema di alimentazione dei vari componenti :.....	7
Il sistema di microcontrollori e moduli.....	7
Il sistema di comunicazione.....	8
Immagini del prototipo.....	9
Cloud.....	11
Le specifiche del Server.....	12
Il software.....	12
Dumb Car.....	13
1.2 I linguaggi, framework di sviluppo e risorse utili.....	14
1.3 "Hey Johnny", RAG e il ruolo dell'AI.....	15
2. L'Edge.....	17
Cos'è l'edge e perchè è necessario.....	17
"Hey assistente!".....	17
La risposta.....	18
Meglio poco e veloce.....	18
I vincoli progettuali di un auto.....	19
Alimentazione e fonti di energia.....	19
La batteria piombo acido.....	20
2.3 Le board.....	23
LilyGO T-SIM7600E.....	23
Nicla Voice a fancy microphone.....	26
2.4 I moduli.....	27
DFRobot Gravity: offline speech recognition sensor.....	27
DFPlayer Mini and its speaker.....	28
3. Automotive e middleware.....	29
3.1 Opel Corsa B 1.0i 12V.....	29
Interfaccia OBD-II.....	29
CAN-BUS, KWP-2000 e un approccio agnostico.....	30
3.2 Installazione dei dispositivi e proof of concept.....	31
Schema di riferimento Edge.....	32
Lunotto guidatore.....	34
Cassetto nascosto con fusibili e OBD-II.....	35
Relè e finestrini.....	37

Isolamento e case stampato in 3d.....	39
3.3 Middleware per la Comunicazione e uno sguardo al futuro.....	41
Vehicle-to-everything.....	42
4. Il cloud.....	43
4.1 Cos'è il cloud e perché è necessario.....	44
Modelli di Servizio del Cloud.....	44
Modelli di Distribuzione del Cloud.....	44
Perché è Necessario il Cloud.....	45
Esempi Pratici di Uso del Cloud.....	45
4.2. Il Self Hosting.....	46
Vantaggi del Self Hosting.....	46
Svantaggi del Self Hosting.....	46
Esempi di Self Hosting.....	47
Quando Utilizzare il Self Hosting.....	47
4.3 The Stack , The Ai and the Hardware.....	48
Le specifiche hardware.....	48
Il carico sotto richiesta.....	49
4.4. LangChain e ChatshireCat.....	50
LangChain.....	51
Caratteristiche Principali di LangChain.....	51
Come Funziona LangChain.....	51
Cheshire Cat.....	52
Cronologia delle conversazioni precedenti.....	52
Caricamento di documenti.....	52
Esecuzione di azioni.....	52
4.5. "Hey Johnny" l'assistente remoto e la sua architettura.....	53
Plugin Setting.....	54
Core CCat.....	54
Smooth Talk.....	54
Piper Cat.....	54
Local Whisper Cat.....	54
4.6 Software Orchestra (Docker-compose).....	54
Docker.....	55
Caratteristiche Principali di Docker.....	55
Docker Compose.....	55
Caratteristiche Principali di Docker Compose.....	55
Foundation Model LLAMA3.....	56
Il plugin e i client.....	56
4.7 Hosting su cosimobiello.xyz.....	57
Configurazione del Tunnel e Autenticazione.....	57
Vantaggi e Benefici di cloudflare tunnel.....	58
4.8 Repository Github.....	58

5. Better safe than sorry.....	60
5.1 Ai safety risk and EU Artificial Intelligence Act.....	60
Fattori di rischio.....	61
Obblighi per i fornitori di sistemi IA ad alto rischio:.....	61
Casi d'uso specifici per IA ad alto rischio :.....	62
Obblighi Sistemi IA Generali (GPAI).....	62
5.2 Safe layers and Tunnels.....	64
Better use https.....	64
ws vs wss.....	64
5.3 Not production ready.....	65
6. Conclusioni e sviluppi futuri.....	66
Risultati e considerazioni finali.....	66
W open source!.....	66
Smartwatch as the edge.....	67
Looking into the bus.....	67
So long, and thanks for all the fish.....	67
INDICE DELLE FIGURE.....	68
BIBLIOGRAFIA.....	70
Ringraziamenti.....	73

INTRODUZIONE

“Hey assistente!”

“I hear you”

“Abbassa i finestrini”

“Ok”

“Passami Johnny”

“Ok”

“Johnny, Che tempo fa stasera?”

“Miao! Allora Lorenzo, a quanto pare la provincia di Siracusa stasera sarà a 27 gradi , ci sarà un cielo limpido e non dovrebbe piovere prima di venerdì, Miao! ”

“Stop Johnny”

“Miao! Sono felice di esserti stato utile! Se ti serve qualcosa sono sempre qui, Miao!”

E’ possibile avere un assistente virtuale con cui avere conversazioni private, che vive sul proprio pc?

Parlandogli, usando solo la voce e senza dover usare smartphone o messaggi di testo?

Le motivazioni dello sviluppo di questa tesi sono quelle di dimostrare le capacità acquisite dall’autore durante gli anni di studio, avendo ormai maturato il senso critico dello sviluppo ingegneristico e con la familiarità dello stack hardware e software che si formano nella mente di un Computer Engineer dopo alcuni anni.

Si è voluto realizzare in meno di 3 mesi un progetto in grado di inglobare le conoscenze riguardo l’elettronica, i calcolatori, il software e il suo sviluppo, il web development, il cloud e il machine learning con le sue AI attorno a un auto del 1997, in modo da interfacciarla al mondo esterno rendendola di fatto una cybercar dotata di un’intelligenza artificiale.

L'implementazione di ciò è stata resa possibile solo a partire dai recentissimi sviluppi riguardo il machine learning, alla nascita delle le **npu** per i microcontrollori , ai chatbot e la loro possibilità di essere eseguiti su calcolatori commerciali (non più esclusivamente su enormi data center).

Sono stati affrontati problemi e vincoli software, hardware e logistici per mantenere attivi e funzionali l'edge in locale per il riconoscimento rapido delle parole, il sistema di telecomunicazioni con 4G e GPS della board principale atto a comunicare con un assistente AI personalizzabile , che viene eseguito interamente su un server remoto, con le specifiche di un comune pc, messo a disposizione dall'Università degli Studi di Catania.

Concludendo, le intenzioni della tesi sono quelle di mostrare cosa può fare l'edge, il cloud e le AI su oggetti comuni con oggetti comuni, dimostrando il paradigma dell' Internet Of Things .

Il seguente studio tratta sei capitoli.

Il primo capitolo è focalizzato sull'architettura del progetto, mostrando la scelta dei componenti hardware , il software sviluppato, le librerie utilizzate, l'auto di riferimento del progetto e le specifiche del server in modo da rendere subito chiaro su cosa e come si intende costruire il sistema.

Il secondo capitolo tratta il discorso dell'edge computing e tutta la sfida relativa alla scelta dei microcontrollori, moduli e componenti che vivranno nell'auto, come andranno alimentati e come dovranno comunicare tra loro, preparando un canale vocale di comunicazione con l'esterno, dell'interfaccia vocale e di tutte le sfaccettature dell'architettura fisica dell'edge, comprese le limitazioni del progetto .

Il terzo capitolo parla del dumb system, cioè delle caratteristiche dell'auto, di come ci si è voluti interfacciare al sistema elettromeccanico su quattro ruote, sull'eventualità di collegarsi al bus di sistema, sulla scelta di posizionamento delle componenti con un proof of concept illustrativo e infine di come estendere la CyberCar al Vehicle-to-everything (**V2X**).

Il quarto capitolo è quello più complesso e mostra ciò che c'è dopo la richiesta vocale dell'auto, spiega il ruolo del cloud, l'implementazione di nuove tecnologie come *LangChain* e il *Retrieval-augmented generation* atte a gestire i Large Language Model in esecuzione sul server e di come integrarli nel flow della programmazione classica.

Parla dell'architettura del server remoto e della sua gpu, della scelta dei container locali e della loro orchestrazione tramite docker-compose .

Verrà introdotta la figura di *Johnny* l'assistente remoto felino in grado di ascoltare i messaggi vocali provenienti dall'auto, e di rispondere alle richieste con la propria voce felina.

Il capitolo si conclude con il deploy del sistema online, rendendolo accessibile tramite il dominio cosimobiello.xyz e protetto dagli accessi indesiderati con cloudflare e i suoi tunnel.

Viene infine presentato e indicato il software utilizzato come client per comunicare con *Johnny : Chatty* e *Chattino* scritti rispettivamente in Python e Arduino (una variante di c++ per i microcontrollori).

Il quinto capitolo fa una riflessione sui rischi delle AI in relazione alla nuova Artificial Intelligence Act europea e alle best practices per i servizi esposti online.

Il sesto e ultimo capitolo trae le conclusioni, parla dell'importanza dell'open source, mostra alternative di progetto e le limitazioni in corso d'opera.

1. Struttura del progetto

1.1 Architettura

Disegni e schemi

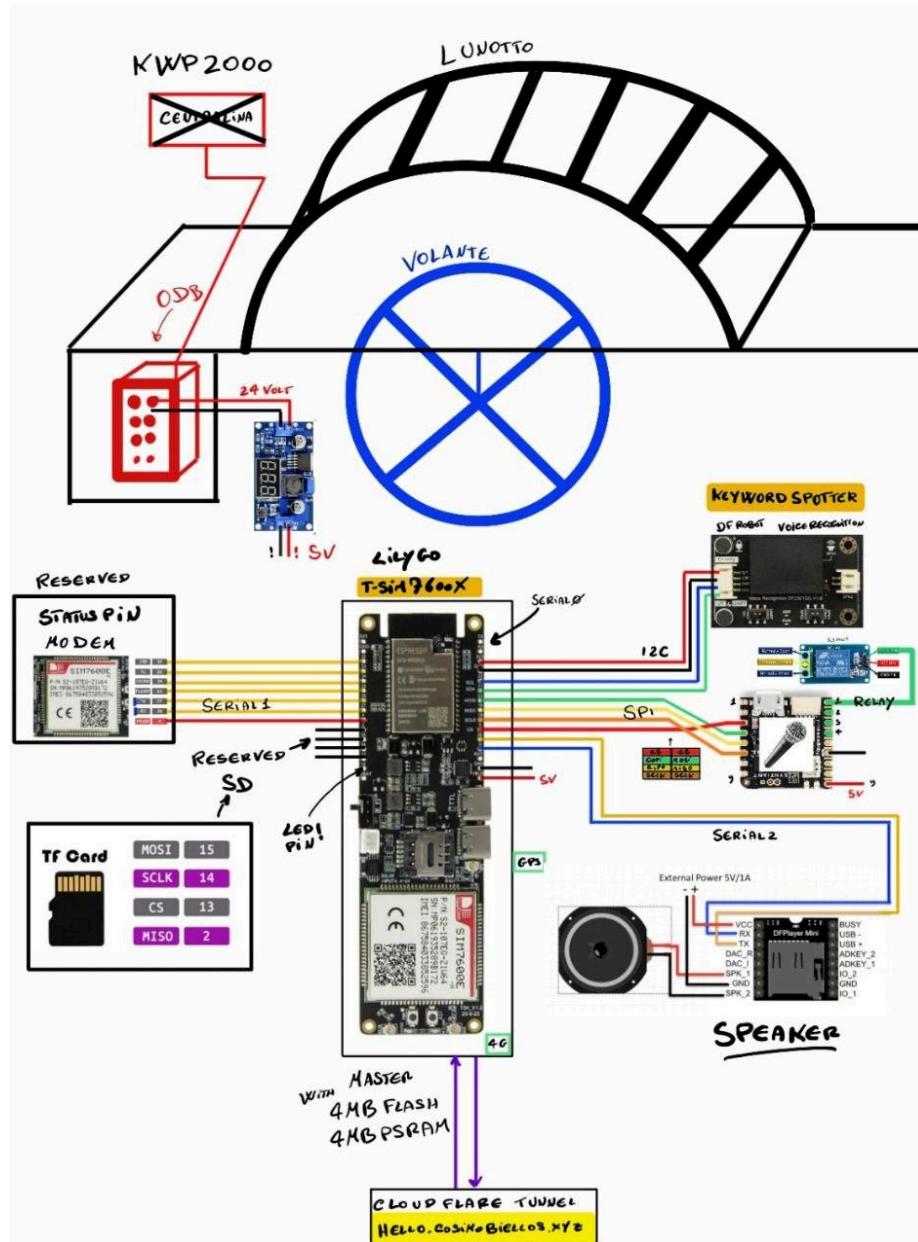


Figura 1.1 Schema di collegamento architettura edge

(Per una preview dettagliata consultare la repository)

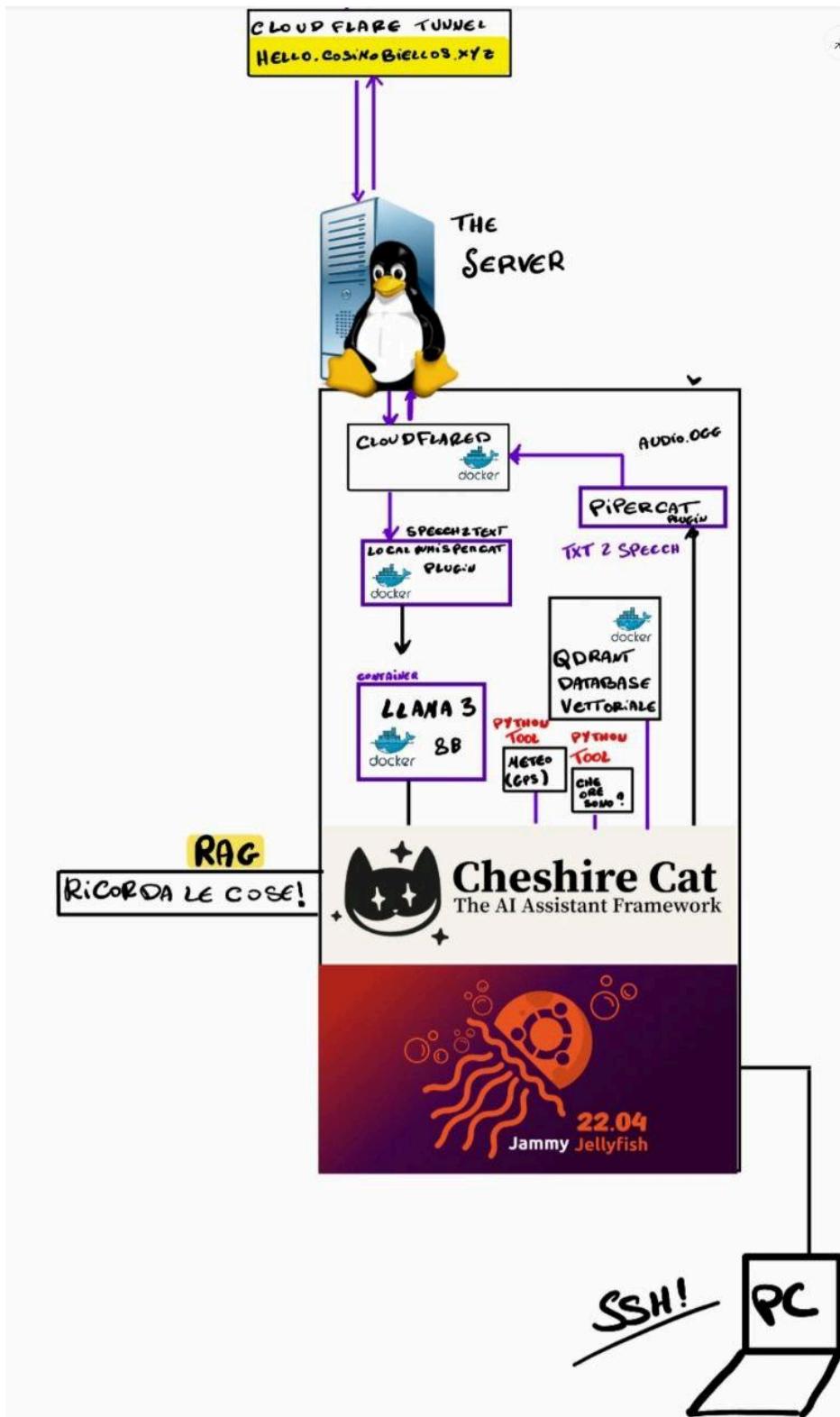


Figura 1.2 Schema logico del cloud
(Per una preview dettagliata consultare la repository)

L'intero progetto si basa su tre blocchi fisici distinti:

Edge, Cloud e Car

Edge

E' tutto ciò che è dentro l'auto ad esclusione di essa e comprende:

Il sistema di alimentazione dei vari componenti :

- Uno Step-Down converter LM2596S DC-DC per convertire i 24V della batteria in una linea a 5V garantendo fino 3 Ampere di corrente (2+1 dissipati per il modello scelto), a patto di picchi massimi di 2A per l'azionamento la richiesta della posizione GPS o per l'azionamento temporaneo dei relè le specifiche dovrebbero essere rispettate. Si è scelta un alimentazione direttamente dalla batteria Piombo acido per evitare il litio che con le alte temperature estive può superare i 60° o in caso di incidenti può provocare esplosioni e danni al guidatore e passeggeri.
- Cavi e jumper per gestire massa e brevi picchi di corrente
- Un eventuale interruttore fisico (non implementato) per staccare la corrente dal circuito

La linea interna a 5 volt andrà ad alimentare direttamente la board *Lilygo T-SIM7600-E*, il *Nicla Voice* e il modulo *DFPlayer Mini* e i relativi relè a 3.3V.

Il sistema di microcontrollori e moduli

- Un **LilyGo SIM7600E**, la board Master, composta da un *ESP32-WROVER-E* e da un modem *SIM7600E* che gestisce la connessione 4G LTE e GPS con 2 antenne esterne, questo viene collegato al DFRobot VOICE RECOGNITION che spotta le keyword, al Nicla Voice che funge da microfono esterno e al DFPlayer mini che gestisce lo speaker a bordo.

- Un **Nicla Voice**, la board Slave, il cui compito è quello di registrare l'audio del guidatore e rimandarlo al Master oppure di azionare i relè a cui è collegato.
- **DFRobot voice recognition** innovativo modulo di Edge computing dotato di una rete neurale interna pre addestrata in grado di spottare 1 wake word custom “*Hey Johnny*” e 21 comandi custom “*Alza i finestrini*”, ecc.
(Purtroppo il modulo non è progettato per registrare e riprodurre audio custom per cui è stato aggiunto il Nicla Voice e il DFPlayer mini, benché abbia 2 microfoni, 1 speaker interno e la predisposizione per uno speaker esterno.)
- **DFPlayer Mini MP3** o qualsiasi altro clone riceve via seriale dal LilyGo SIM7600E un audio mp3 con la risposta del server e la riproduce col supporto di uno speaker a 3W.

Il sistema di comunicazione

Gestito dal modem del **SIM7600E** tramite le 2 antenne LTE e GPS e con lo slot per la nano sim.

Ovviamente serve un piano dati e un operatore 4g.

Immagini del prototipo

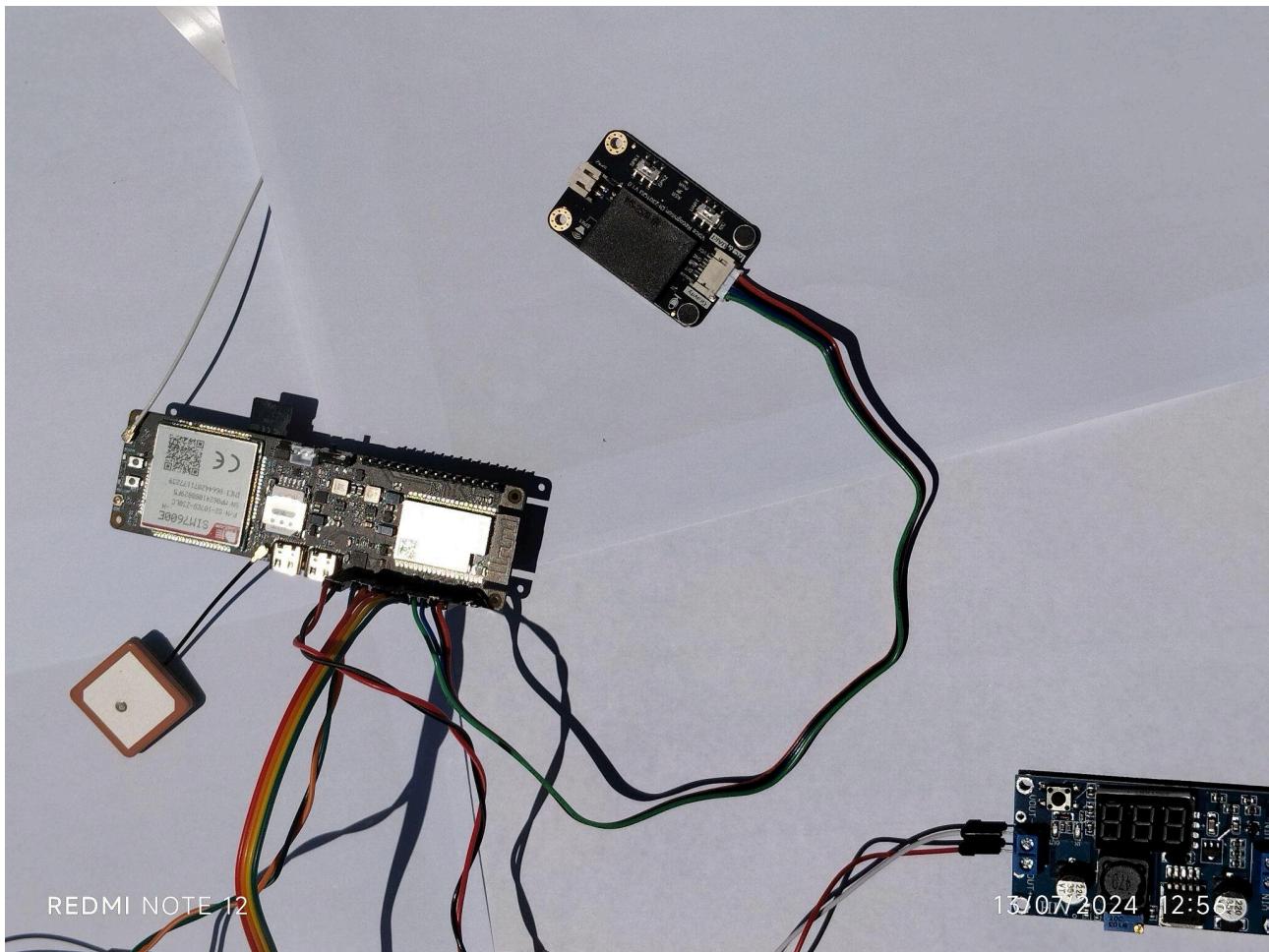


Figura 1.3 Primo piano del prototipo assemblato, dove sono visibili LilyGo SIM7600E ,DFRobot voice recognition e antenne 4g, lte

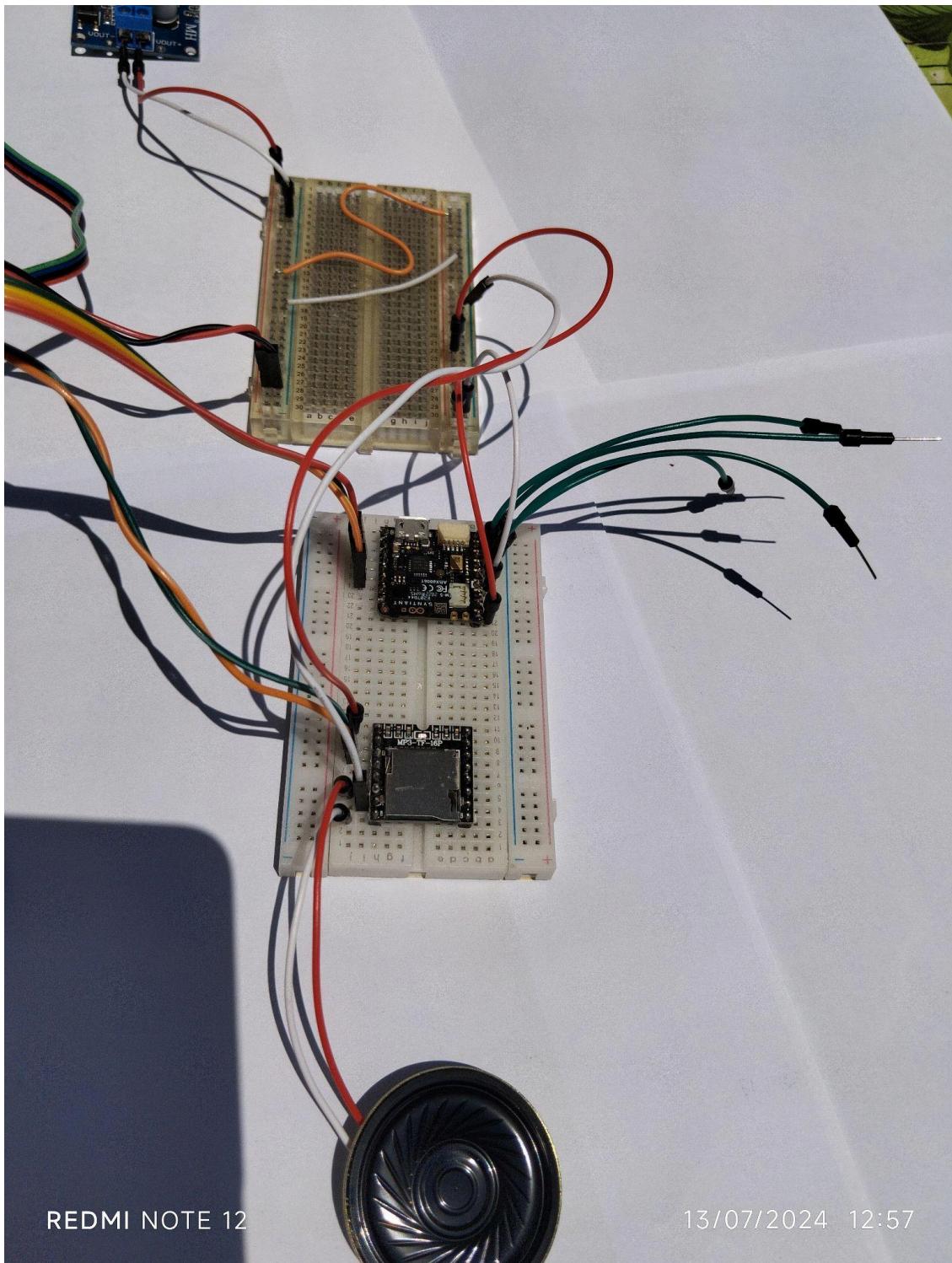


Figura 1.4 Primo piano dove sono visibili il Nicla Voice, il modulo DFplayer, lo speaker da 2 watt e i collegamenti disponibili per i relè

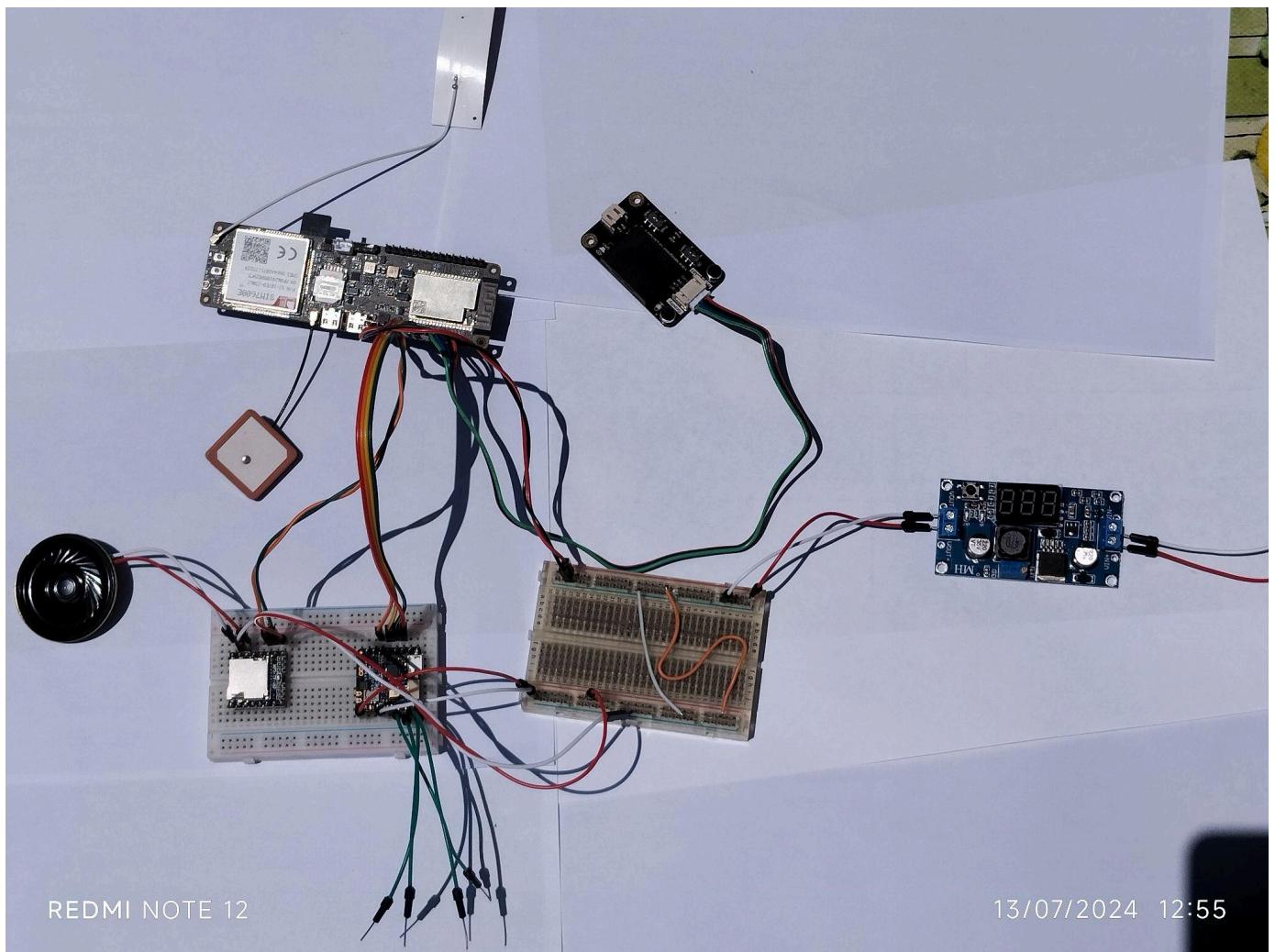


Figura 1.5 Panoramica completa del prototipo assemblato

Cloud

Contattando il dominio protetto da CloudFlare e dopo essersi autenticati si arriva all’indirizzo del server di sviluppo dell’università, che ospita il server il quale accoglie i vocali del guidatore e dopo averli elaborati li rispedisce come risposta vocale.

Le specifiche del Server

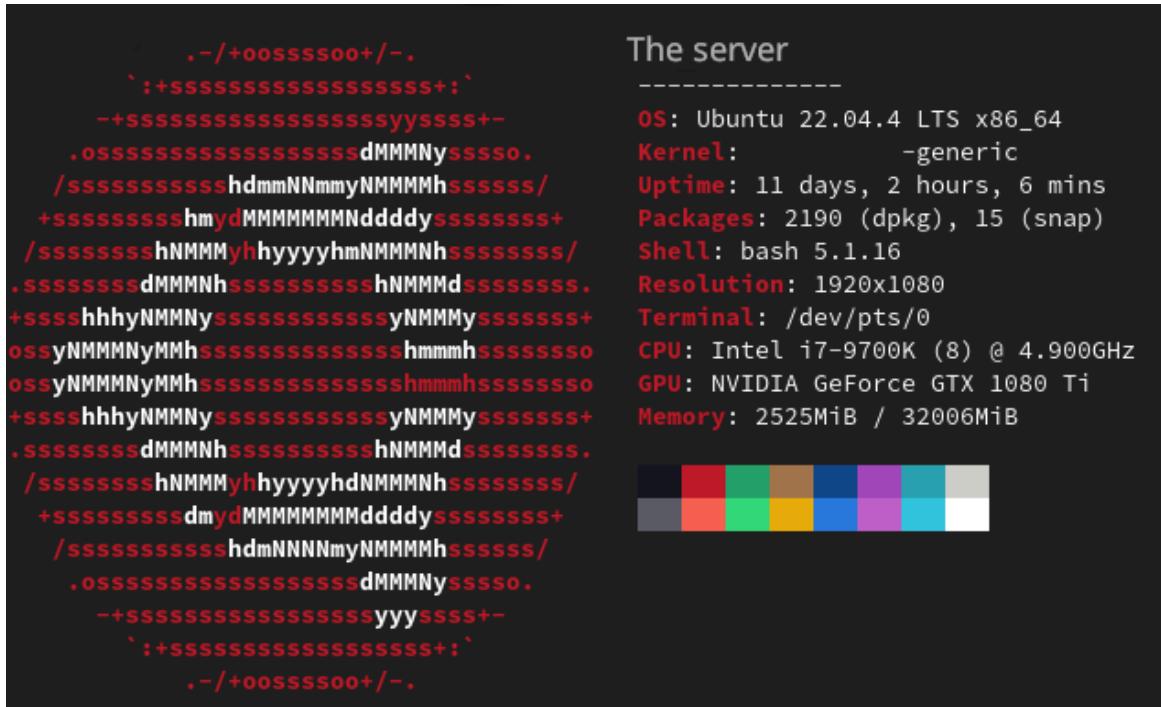


Figura 1.6 specifiche del server di sviluppo

Il punto focale del server e condizione necessaria affinché sia possibile eseguire lo stack necessario per le reti neurali che gireranno nei vari container è la scheda video NVIDIA GTX 1080 Ti con 11GB di VRAM e i 3584 core (oltre alle altre specifiche tecniche che sono facilmente consultabili online o nella bibliografia).

E' da notare che parliamo di una scheda video del 2017 quindi facilmente reperibile come usato.

Sono poi presenti una cpu Intel i7-9700K CPU a 3.60GHz a 8 core e 32 GB di RAM

Il software

- Ubuntu GNU/Linux
- Docker e Docker-compose
- Ollama con Llama3 come foundational model e chatbot
- Cloudflare Tunnel client
- Cheshire Cat The Ai assistant Framework (derivato da LangChain)
- PiperCat (plugin che incapsula Piper)
- Local Whisper Cat (plugin che gestisce il container Whisper di openAi)

- Qdrant (il database vettoriale per cui è possibile il RAG)
 - La Repository del progetto
- <https://github.com/LorenzoSiena/Johnny-The-CyberCar-Assistant>

Dumb Car



Figura 1.7 Opel corsa 1997 1.0i 12V

L'Opel Corsa B facelift 1997 1.0i 12V (54 CV) è una hatchback con 3-5 porte e 5 posti. È dotata di un motore a benzina da 973 cm³ con 54 CV e 82 Nm di coppia. Ha una velocità massima di 150 km/h e accelera da 0 a 100 km/h in 18 secondi. Il consumo medio è di 6 l/100 km con emissioni di CO₂ di 136 g/km. Le dimensioni sono 3740 mm di lunghezza, 1610 mm di larghezza e 1420 mm di altezza, con un peso di 872 kg. Il cambio è manuale a 5 marce.

Dal punto di vista elettronico oltre ovviamente alla centralina, è degna di nota la presenza di un'uscita OBDII a cui ci si può interfacciare al protocollo intento, in questo caso NON can-bus ma bensì **KWP2000**, questo ha portato a molte limitazioni di studio in quanto il suddetto protocollo non è ben documentato e un eventuale reverse engineering sul protocollo e i suoi suoi sistemi di messaggi avrebbe aggiunto tempo non disponibile per la consegna di questa tesi.

Nonostante ciò si è voluti agire sul sistema dei finestrini utilizzando dei relè in modo da dimostrare la fattibilità dell’interfaccia microcontrollori-automobile.

Inoltre il sistema qui elaborato è completamente agnostico e aggiungendo alla board di controllo un modulo serial2CanBus (o simili) è possibile dalla porta OBDII sniffare e reversare i pacchetti in modo da aggiungere ai comandi vocali anche delle azioni che possono agire sul bus attivo, come azionare frecce, spegnere le luci, alzare finestrini, chiudere le portiere ed eventualmente direttamente agire sulla centralina.

!ATTENZIONE!

Scrivere sul bus di un’auto può avere risvolti pericolosi e può risultare in malfunzionamenti, e/o incidenti, per cui è consigliato l'estrema cautela e pensier critico nelle proprie azioni.
L'autore della tesi non si assume alcuna responsabilità per eventuali danni o conseguenze
derivanti dall'esecuzione di queste operazioni.

1.2 I linguaggi, framework di sviluppo e risorse utili

Per sviluppare il seguente progetto è stato utilizzato VS CODE, con estensioni python e PLATFORM.IO, Arduino IDE .

PlatformIO è un ambiente di sviluppo integrato (IDE) multipiattaforma per lo sviluppo di progetti embedded. Supporta diverse piattaforme hardware come Arduino, Espressif, e STM32, e offre funzionalità come il debugging, la gestione delle librerie e l'integrazione con vari editor di codice come Visual Studio Code.

L'Arduino IDE è un ambiente di sviluppo integrato specifico per la programmazione delle schede Arduino. Supporta linguaggi di programmazione come C e C++.

Per più di un motivo sono stati usati entrambi.

Il linguaggio Arduino (simile al c++) è stato utilizzato per scrivere Chattino
(client per le board in mobilità esp32)

Python è stato utilizzato per scrivere Chatty (pc client)
e Local Whisper Cat (il plugin per gestire il container Whisper in locale).

Tutte le risorse del progetto e l'intera tesi sono disponibili e consultabili online presso la repository:

<https://github.com/LorenzoSiena/Johnny-The-CyberCar-Assistant>

Dove non diversamente specificato, il contenuto prodotto dall'autore è soggetto alla GNU General Public License v3.0.

L'autore tende a precisare che senza l'open source e l'open hardware questo progetto non sarebbe stato possibile e quindi invita chiunque stia leggendo a contribuire se può farlo.

1.3 “Hey Johnny”, RAG e il ruolo dell’AI

La visione del progetto si basa sulla possibilità di avere al volante un assistente virtuale con cui poter interagire usando solo la voce e che è in grado di agire sulle componenti fisiche dell’auto , come alzare i finestrini per il nostro scopo dimostrativo, oppure per richieste più complesse , come chiedere l’orario , il meteo , un’ informazione generica , o una domanda che può essere risposta solo con l’esecuzione di uno script python , quindi precisa e senza allucinazioni del modello.

Il framework scelto *Cheshire Cat AI* basato su *LangChain* permette di usare la Retrieval-Augmented Generation (RAG) che consiste nella possibilità dell’agente di

recuperare informazioni precise da un database vettoriale ben strutturato , un pò come se avesse uno schedario con i nostri documenti .

Il vantaggio diventa evidente quando ci si rende conto che non è necessario riaddestrare o fare fine tuning sul modello per raggiungere una buona qualità delle informazioni richieste e dell'abbassamento in percentuale delle allucinazioni del modello.

Oltre alla possibilità di richiedere informazioni diventa possibile anche salvare informazioni , infatti il framework, settando il container di qdrant, rende possibile all'agente di ricordare le precedenti richieste dell'utente quindi implementando effettivamente una memoria a lungo termine.

L'agente è anche in grado, nel caso che la richiesta raggiunga la soglia minima di attenzione, di innescare delle vere funzioni in python, (disponibile come hook in un plugin di *Cheshire Cat*) come getTime() per farsi dare l'orario preciso o getPosition() per sapere la propria posizione in un formato piacevole o qualsiasi altra funzione che voglia essere implementata , o già disponibile dalla community.

Qui va fatta una precisazione , in quanto benché sia un innovazione è necessario prestare particolarmente attenzione a cosa può e cosa non può accedere l'agente e capire il fattore di rischio per cui una richiesta mal interpretata al momento sbagliato potrebbe fare danni seri , sono stati scelti i finestrini come azione disponibile perché rappresentano un rischio accettabile e prevedibile, in altro modo la possibilità di cambiare marcia avrebbe posto seri pericoli.

2. L'Edge

L'edge computing è un paradigma informatico che coinvolge l'elaborazione dei dati vicino al luogo in cui questi vengono generati, piuttosto che inviarli a un data center centralizzato o a un cloud per l'elaborazione. In pratica, significa che i dati vengono elaborati "al margine" della rete, il più vicino possibile alla loro origine. In questo progetto tutto quello che è interno alla macchina (essa esclusa) viene considerato come edge.

Cos'è l'edge e perchè è necessario

Uno dei primi problemi che si incontrano quando si cerca di realizzare un sistema in grado di rispondere idealmente in tempo reale è la velocità con cui si riesce a processare un dato. In questo caso per poter accedere all'assistente remoto o semplicemente per imporre un comando sui finestrini è necessario invocare il dispositivo con un comando vocale.

“Hey assistente!”

Un comune controllore è in grado di registrare le onde sonore dell'utente che fanno vibrare il microfono e che si trasformano in tensione elettrica arrivando al dac del microcontrollore, il quale leggendo le variazioni di potenziale , le campiona e ci salva un dato, quel dato rappresenta la voce.

Questo task si risolve in pochi millisecondi e si percepisce in tempo reale, stessa cosa vale per la riproduzione, non sentiamo differenze di sorta.

Il problema si palesa quando vogliamo capire determinati pattern dell'audio che dipendono dal timbro della voce, dal rumore , dalla temperatura dell'ambiente e tentare di riconoscere una determinata parola o set di parole come “HEY ASSISTENTE”.

Qui il problema non è più così deterministico , con la tecnologia attuale si riesce a pulire la traccia ma riconoscere la **KEYWORD** diventa dispendioso dal punto di vista computazionale, per cui si può risolvere inviando “HEY ASSISTENTE” a una rete neurale

general purpose pre addestrata con pesanti specifiche minime in un server lontano o in un datacenter e poi aspettare la risposta.

La risposta

Il tempo di risposta medio può variare di molto e può dipendere dal traffico della rete, potenza del segnale di connessione, e da altri fattori, ma mediamente il problema noto è che supera la percezione umana in realtime (sull'ordine di 10 ms) di diversi ordini di grandezza e quindi si finisce per aspettare **5-8 secondi** per la **ricezione** di una risposta sul comando.

Meglio poco e veloce

Negli ultimi anni con la scoperta delle reti neurali, nei transformer e delle npu miniaturizzate (acceleratori di pattern recognition o di altri modelli) sulle architetture arm o mips si è visto un boom di piccoli dispositivi specializzati in grado di risolvere un problema come il keyword spotting , in modo da poter invocare una funzione semplicemente con la voce.

Nel nostro caso faremo uso del DFrobot Gravity:offline speech recognition per riconoscere una wake-word personalizzata e 4 set di comandi per i finestrini.

E' bene notare che la nostra soluzione sia veloce MA special purpose quindi il modulo fa quasi esclusivamente questo.

I vincoli progettuali di un auto

!ATTENZIONE!

L'autore non si assume responsabilità per eventuali danni derivanti dall'uso delle seguenti informazioni. Questo include, senza limitazioni, danni diretti, indiretti, speciali, incidentali o consequenziali.

Una delle prime sfide relative a far ospitare un microcontrollore in un piccolo abitacolo mobile è dove posizionare i dispositivi, che siano vicini a una fonte energetica stabile e pulita .

Alimentazione e fonti di energia

Un'opzione valida in mobilità può essere l'utilizzo di batterie al litio che in genere con le giuste accortezze e con i stretti consumi delle board vanno sostituite/ricaricate ogni tot mesi o ricaricate usando i supporti solari delle board. Il problema nasce quando le board sono multiple o servono tensioni maggiori fuori dal range della board.

Inoltre si corre il rischio di far surriscaldare eccessivamente le batterie al litio , durante le giornate estive, che possono degradare le prestazioni o in casi estremi esplodere, va anche fatto notare il rischio meccanico di incidente.

Motivo per cui si è scelto di usare e adattare la fonte energetica già presente nell'automobile in grado di soddisfare qualsiasi tensione necessaria ai microcontrollori presenti o agli altri

dispositivi tipicamente alimentati a 5 volt come moduli e speaker esterni.

Figura 2.1 Batteria 18650 agli ioni di litio



La batteria piombo acido

L'automobile utilizza una batteria di tipo piombo acido per l'accensione del motorino di avviamento e per i sistemi elettronici a bordo.



Figura 2.2 Batteria piombo acido

Collegandosi a un punto di GND e VCC disponibile anche dalla porta OBDII (assicurandosi che supporti un assorbimento di almeno 2 ampere!) e utilizzando un LM2596S Step-Down converter si riduce la tensione dai 24 volt misurati ai 5 volt necessari, (agendo sulla vite infinita dietro il dispositivo) e si riesce ad alimentare il tutti i dispositivi coinvolti secondo il seguente schema.

La vite verrà successivamente bloccata, con colla a caldo per evitare che le vibrazioni ruotino e modifichino di conseguenza la tensione disponibile con effetti nefasti sull'elettronica.

In questo caso specifico si è ritenuto opportuno collegarsi al 4 pin di ground, come polo negativo, direttamente collegato alla batteria e al pin 16 come polo positivo.

E' possibile collegarsi a un'altra linea idonea collegata direttamente alla batteria **a patto che si verifichi con un multimetro la validità della linea di alimentazione.**

Per ogni dubbio è possibile rivolgersi a un elettrauto.

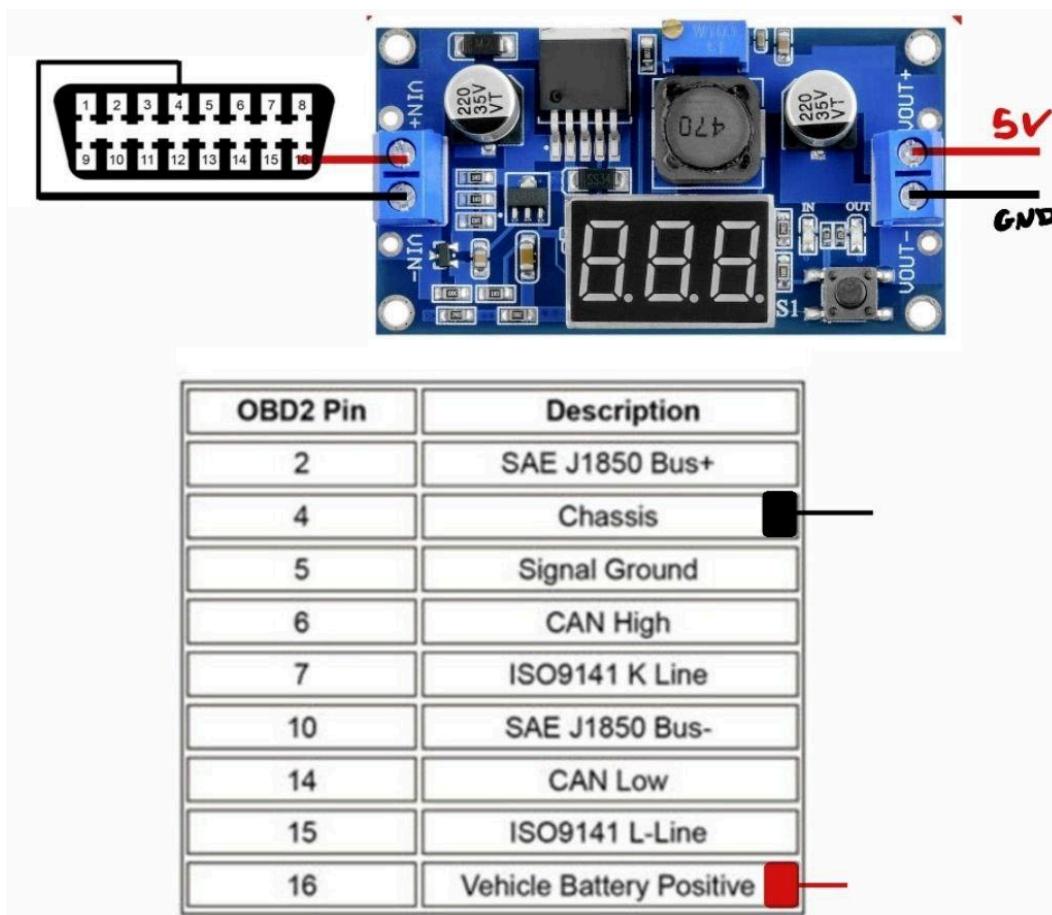


Figura 2.3 Schema di alimentazione 24V->5V con step-down converter

Da qui in poi la linea a 5 volt andrà ad alimentare le board, i moduli e gli speaker nei rispettivi pin Vin.

Un altro problema riguarda come unire meccanicamente e saldamente i componenti alla scocca dell'auto.

A tal proposito è stata usata una stampante 3d per stampare in PLA il case delle varie componenti a cui ogni auto andrà adattata.

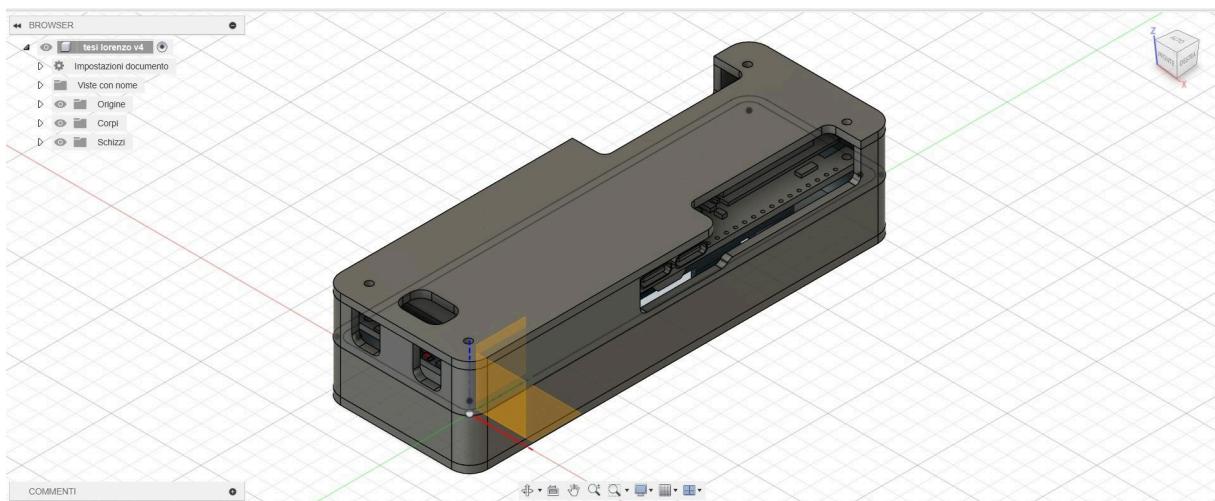


Figura 2.4 Progetto case in 3D per la board master per gentile concessione di Luca Spada.

2.3 Le board

LilyGO T-SIM7600E

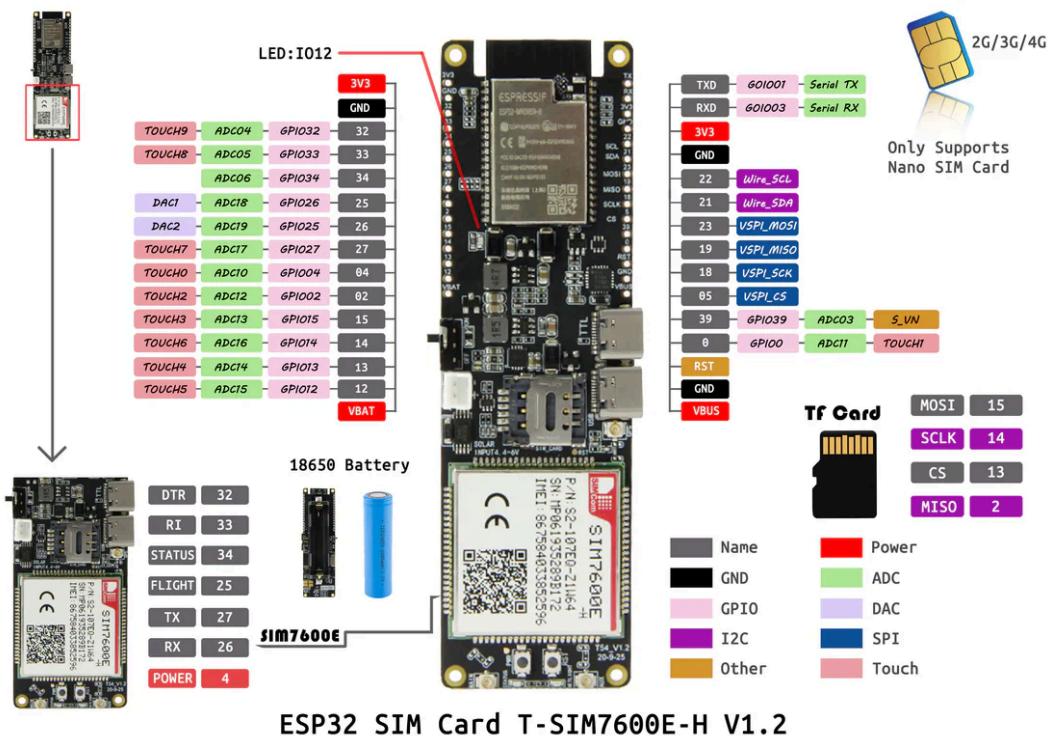


Figura 2.5 Lilygo T-sim7600E.

Il cuore dell'intero progetto gira attorno a questa board formata da 3 blocchi logici

ESP32

Un ESP32 WROVER-E dotato di 4MB di FLASH , 4MB di PSRAM e abbastanza pin per essere funzionale al nostro scopo (la cui metà però è occupata dalle periferiche).

Periferiche

Sono presenti: uno slot sd, un connettore per un pannello solare, un socket per una batteria 18650 , una scheda sim nano, e linee dedicate per la trasmissione di informazioni tra esp e modem e gli slot per le antenne GPS e LTE.

MODEM

Un SIM7600E capace di inviare e ricevere messaggi di tipo AT con cui accetta le request per le connessioni verso i satelliti GPS e le torri radio LTE,

≡ ESP32 Information	
Chip Model	ESP32-D0WD-V3
Chip Revision	3
Number of Cores	2
CPU Frequency	240 Mhz
Up Time	0d 00:03:10
Current Cycle Count	2846059665
Flash Chip Speed	40.00 MHz
Flash Chip Size	4 MB
Sketch Size	803 KB
Free Sketch remaining	1 MB
Heap Size	317 KB
Free Heap	221 KB
Heap Maximum Allocatable block	108 KB
PSRAM Size	4 MB
Free PSRAM	4 MB
PSRAM Maximum Allocatable block	4 MB

attraverso gli slot di periferiche dedicate alle antenne.

Figura 2.6. Specifiche dell'ESP32 WROVER-E

La board Master monterà in flash il software “**chattino**” con il compito di :

- Settare e gestire il modem con una connessione Seriale
- Creare una connessione WebSocket autenticata con il server remoto a cui mandare gli audio
- **Ascoltare** cioè aspettare una wake word da parte del modulo di Voice Recognition ed eseguire i corrispondenti comandi tra cui alzare/abbassare i finestrini, e in caso venga richiesto con “**Passami Johnny**” inviare i vocali verso il server e l’assistente remoto Johnny.
- **Parlare** cioè aspettare e riprodurre eventuali risposte vocali ricevuti dall’assistente mediante l’ausilio del modulo Dfplayer mini a cui invierà file mp3 salvati temporaneamente nella sd card.
- Gestire una connessione Seriale con Dfplayer mini[39,0]
- Gestire una connessione SPI con Arduino Nicla Voice pin [23,19,18,5]
- Gestire una connessione I2C con il modulo di voice recognition.[22,21]

Il codice appena descritto è disponibile presso la seguente repository
<https://github.com/LorenzoSiena/chattino> e appartiene al progetto della tesi
<https://github.com/LorenzoSiena/Johnny-The-CyberCar-Assistant>
tutto il software è open source e liberamente distribuibile/modificabile secondo licenza gpl 3.0.

Il modulo verrà alimentato dal pin VBUS con 5 volt.

Nicla Voice a fancy microphone

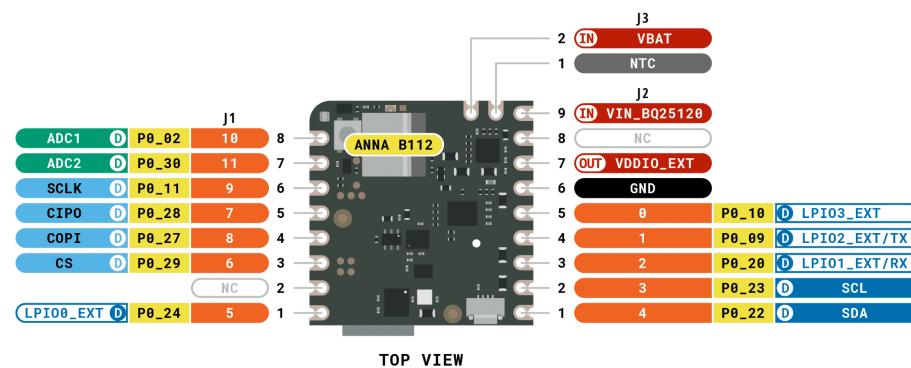
A causa dell'impossibilità del modulo di voice recognition di riprodurre e registrare audio custom dell'utente si è voluto aggiungere una board slave che funge da microfono per inviare messaggi all'assistente remoto.

La board monta (limitatamente ai nostri interessi) 1 microfono ad alte prestazioni (IM69D130), un controller Cortex® M0 core up to 48 MHz a basso consumo e un Digital Signal Processors.

Complessivamente insieme ai numerosi pin disponibili la board permette di controllare i relè per azionare i finestrini e di registrare messaggi audio ben udibili filtrando il rumore.

Il modulo verrà alimentato dal pin VIN_BQ24120 con 5 volt e collegata al master mediante protocollo SPI con i pin:

- SCLK segnale di clock
- CIPO [equivalente MISO]
- COPI [equivalente MOSI]
- CS chip select



Legend:	■ Digital	■ I2C	■ Analog
■ Power	□ Analog	□ SPI	
■ Ground	■ Main Part	□ OTHER SERIAL	

NICLA VOICE
SKU code: ABX00061
Pinout
Last update: 7 Oct, 2022

Figura 2.7. Pinout Nicla Voice

2.4 I moduli

DFRobot Gravity: offline speech recognition sensor

E' la parte più importante dell'edge, collegato tramite I2C (supporta anche UART).

Il modulo presenta 121 parole di comando integrate comunemente utilizzate (in lingua inglese) e supporta 17 comandi personalizzati. Ha 1 speaker interno , 1 slot per speaker extra 2 microfoni e risponde correttamente alle richieste di comando.

Integra di fabbrica un piccolo modello neurale nascosto di keyword spotting e si addestra ripetendo 3 volte il comando attraverso una determinata keyword.

Purtroppo non è molto open hardware e il supporto della casa madre preferisce vendere un modello a parte per registrare e inviare audio, nonostante ciò è perfetto per questo scopo di keyword spotting. Si connette al master con il cavo proprietario gravity , che può essere usato come i2c e uart.

Ha un led blue per quando viene richiamato l'assistente predefinito "Hello Robot" a cui risponde "I'm listening" e per tot secondi configurabili accetta comandi come "Turn off the light" a cui risponde "OK" , è abbastanza configurabile.

Usa la libreria https://github.com/DFRobot/DFRobot_DF2301Q/tree/master



Figura 2.8 DFRobot Gravity

DFPlayer Mini and its speaker

Il modulo audio dalle seguenti specifiche

- Sampling rates (kHz): 8/11.025/12/16/22.05/24/32/44.1/48
- 24 -bit DAC output, support for dynamic range 90dB , SNR support 85dB
- Fully supports FAT16 , FAT32 file system, maximum support 32G of the TF card, support 32G of U disk, 64M bytes NORFLASH
- A variety of control modes, I/O control mode, serial mode, AD button control mode
- audio data sorted by folder, supports up to 100 folders, every folder can hold up to 255 songs
- 30 level adjustable volume, 6 -level EQ adjustable

permette di riprodurre l'audio di risposta dell'assistente attraverso uno speaker fino a 3W collegato con i pin SPK_1 e SPK_2, è connesso al master attraverso una seriale per i pin RX e TX ed è alimentato con VCC e GND a 5V.

Usa la seguente libreria <https://github.com/Makuna/DFMiniMp3>

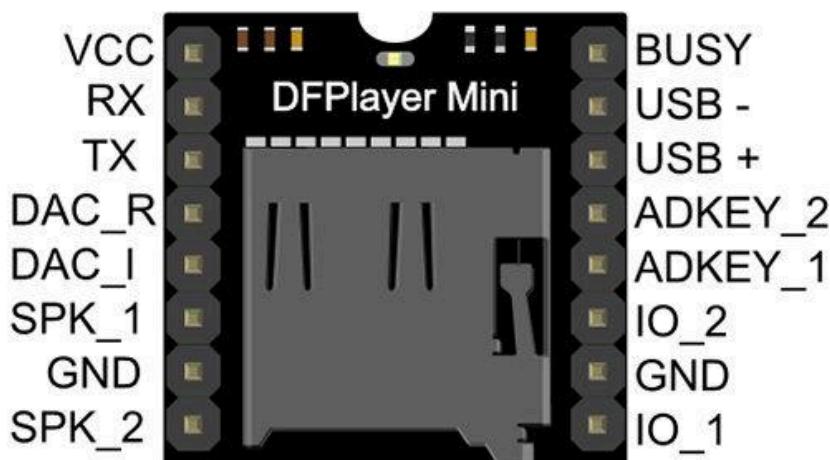


Figura 2.9 DFPlayer Mini

3. Automotive e middleware

Sono passati circa 100 anni da quando è cominciata la produzione di massa nel settore automobilistico e partendo con il Ford Model T si è susseguita una lunga vicenda di evoluzioni interne ed esterne alla figura dell'auto, arrivando in data odierna all'approccio sempre più propenso verso l'elettrificazione e la computerizzazione di ogni aspetto di un veicolo, che nel suo core ha un solo compito, trasportare un determinato carico da un punto a A un punto B .

3.1 Opel Corsa B 1.0i 12V

L'auto presa in considerazione dal presente studio è **Opel Corsa B 1.0i 12V** appartenente agli anni 90, (periodo storico poco smart), è molto essenziale, limitata e dedita alla gestione interna della mcu del motore a combustione e agli aspetti meccanici dell'esperienza. Nonostante ciò è dotata di un modesto armamentario tecnologico come finestrini elettrici anteriori lato passeggero e guidatore, autoradio, spie, e portiere elettriche.

Con lo scopo principale di voler dare vita nuova all'auto e approcciarsi alla modernizzazione dell'auto dopo brevi ricerche si è scoperto che le dinamiche interne di controllo sono nascoste e centralizzate dietro il pannello anteriore appena sotto il volante.

Interfaccia OBD-II

Lì risiede la porta OBD-II interfaccia multiprotocollo destinata al debug, alla telemetrie, capace di comunicare con un computer (usando con le giuste interfacce) e di mostrare errori e valori, comunicati direttamente dall'unità di controllo motore (aka centralina) o dai dispositivi nel network governato dal protocollo locale attivo.

Tra i più diffusi protocolli e presente nella stragrande maggioranza delle auto fino ai giorni nostri è il CAN-BUS sviluppato da Bosch , molto accessibile decentralizzato e ben documentato su internet.

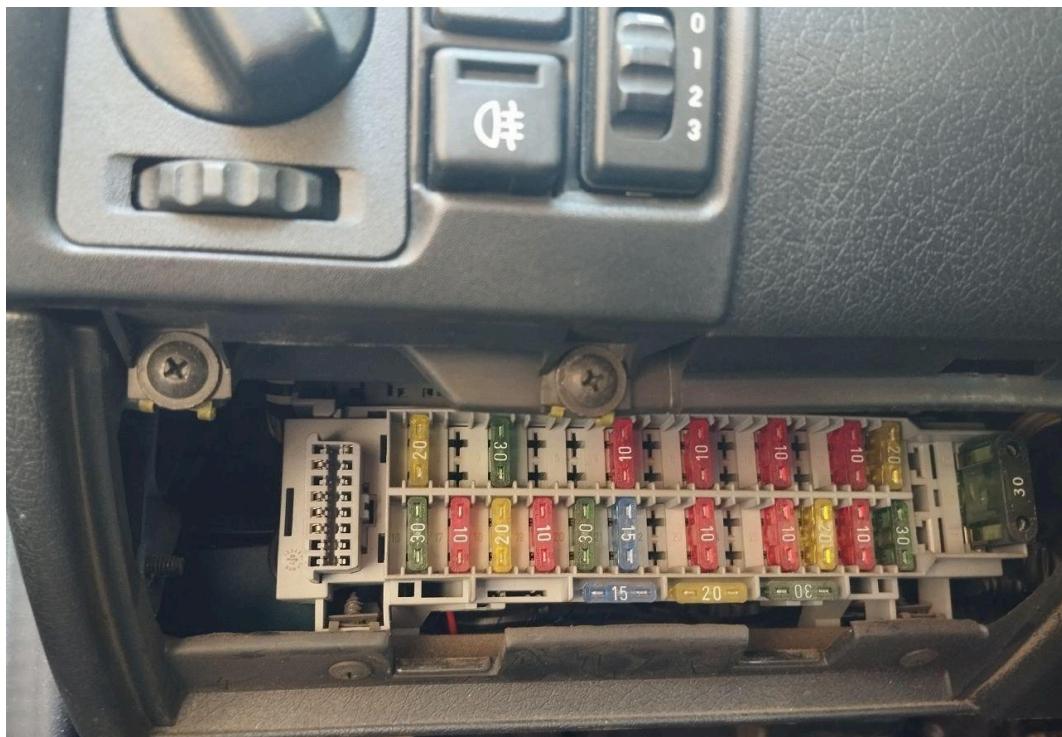
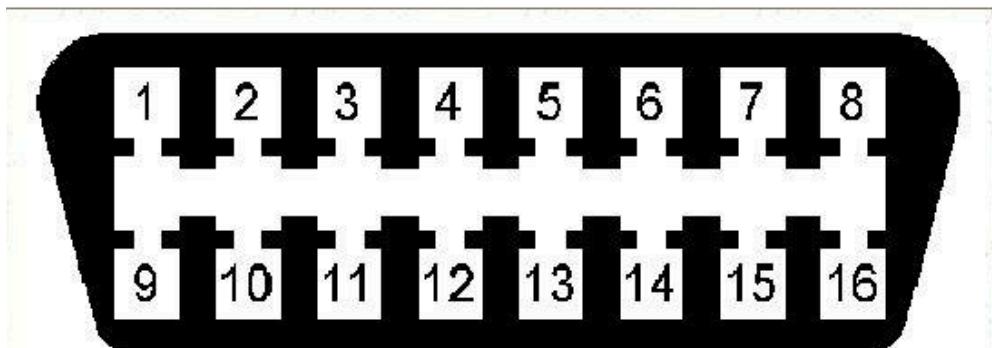


Figura 3.1. Locazione porta OBD-II

CAN-BUS, KWP-2000 e un approccio agnostico.

Purtroppo in corso d'opera si è scoperto, accedendo dalla porta OBD-II e con una comune applicazione scanner per android che il protocollo presente è un più oscuro KWP-2000 (Keyword Protocol 2000) , per motivi di tempo di implementazione e supporto per lo sviluppo non è stato possibile accedere al bus ed è stato preferito rimanere separati dal bus, lasciando aperta la possibilità di potersi interfacciare a prescindere dal modello di auto in questione con un semplice modulo Serial2Can o qualsiasi altro protocollo disponibile, preferendo riferirsi al semplice intercettamento del segnale di azionamento dei finestrini, mediante l'utilizzo di relè azionati dall'edge.



PIN	DESCRIPTION	PIN	DESCRIPTION
1	Vendor Option	9	Vendor Option
2	J1850 Bus +	10	j1850 BUS
3	Vendor Option	11	Vendor Option
4	Chassis Ground	12	Vendor Option
5	Signal Ground	13	Vendor Option
6	CAN (J-2234) High	14	CAN (J-2234) Low
7	ISO 9141-2 K-Line	15	ISO 9141-2 Low
8	Vendor Option	16	Battery Power

Figura 3.2 Pinout porta OBD-II

3.2 Installazione dei dispositivi e proof of concept

In seguito è proposto un **proof of concept** di una possibile configurazione hardware.

Schema di riferimento Edge

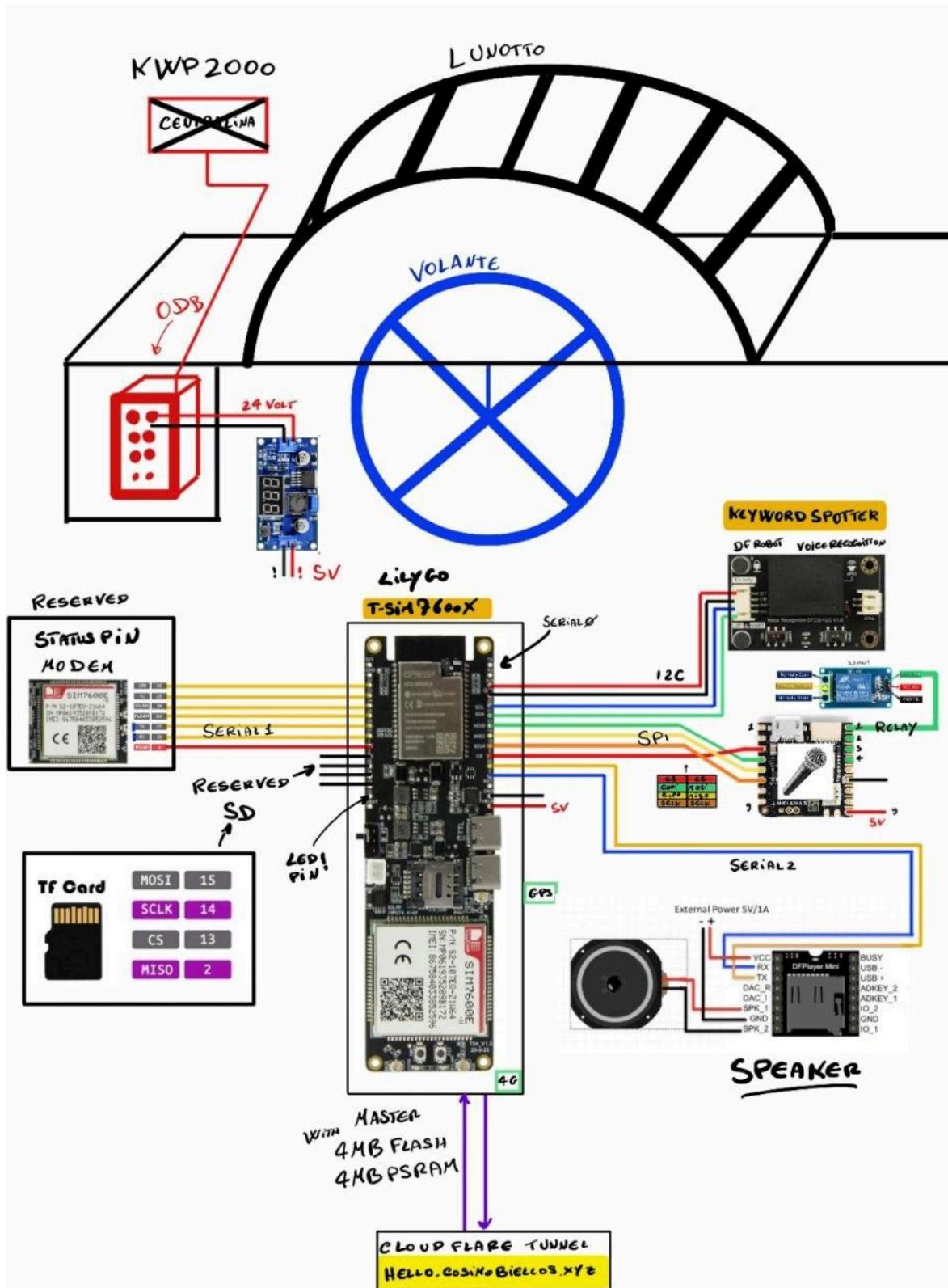


Figura 3.3. Schema di riferimento edge

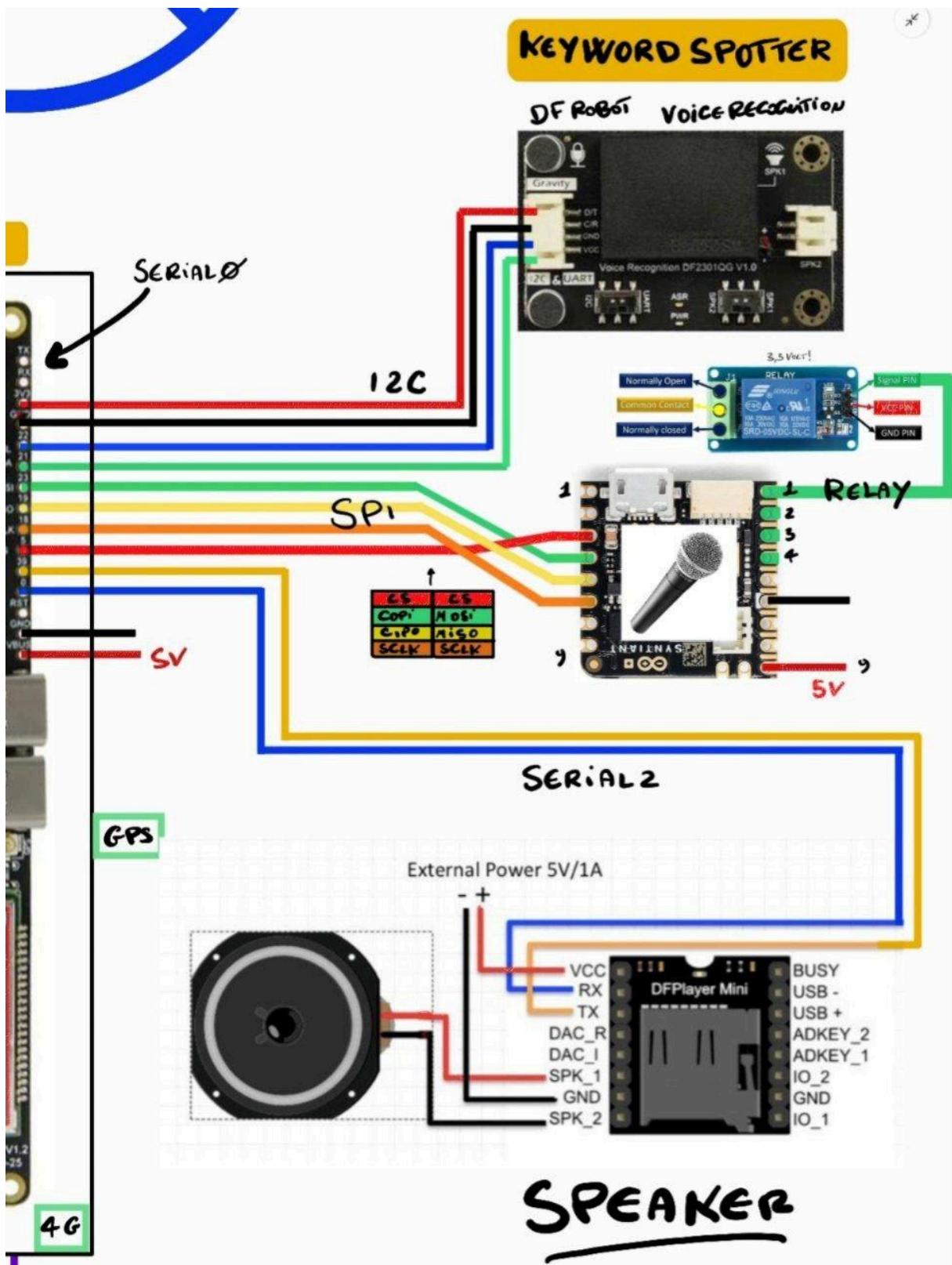


Figura 3.4 Dettagli schema edge

Lunotto guidatore



Figura 3.5 proof of concept dei comandi vocali

Davanti al guidatore è presente il keyword spotter dedicato ai comandi vocali base e in caso di "Passami Johnny" e quindi richieste verso l'assistente remoto cloud si attivano il nicta voice per registrare e lo speaker supportato dal DFplayer (non presente in figura).

Sono escluse dall'immagine le connessioni dati (SPI,SERIAL,I2C) verso il master e i relativi cavi di alimentazione a 5 volt e GND.

Cassetto nascosto con fusibili e OBD-II



Figura 3.6 proof of concept alimentazione , board master e antenne

In basso a sinistra del volante è presente un cassetto nascosto dove è possibile accedere ai fusibili e alla porta OBD-II , è possibile collocare lo step down converter prendendo 24 volt in alimentazione dai pin 16 (VEHICLE BATTERY POSITIVE) e 4 (CHASSIS) rispettivamente secondo la già affrontata

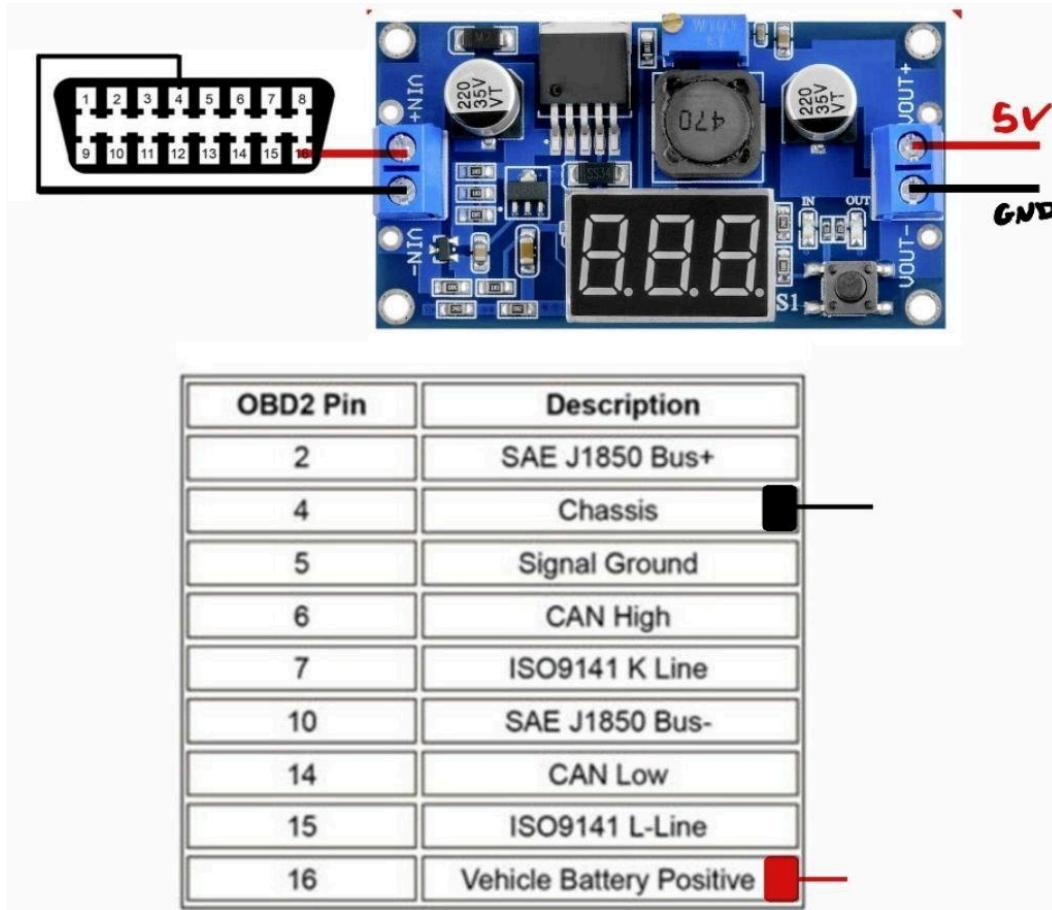


Figura 3.7 Schema di alimentazione 24V->5V con step-down converter

Alimentando a 5 volt la board Master e veicolando la tensione per il lunotto , dalla board si vedono partire l' antenna gps (facoltativa) e quella LTE, va aggiunto che per una buona ricezione è consigliato (se non necessario) porre le antenne rivolte verso l'alto.

Le connessione dati SPI,I2C,SERIAL sono omesse.

E' fortemente consigliato di aggiungere un sistema di reset collegato alla rotazione

del quadro macchina. In modo da accendere il sistema quando si inseriscono e ruotano le chiavi accendendo la macchina.

Relè e finestrini

Per poter connettersi alla pulsantiera dei finestrini è necessario intercettare e inserire un relè nel circuito che controlla il meccanismo controllando il relè con il Nicla Voice in cui abbiamo almeno 4 pin digitali disponibili.

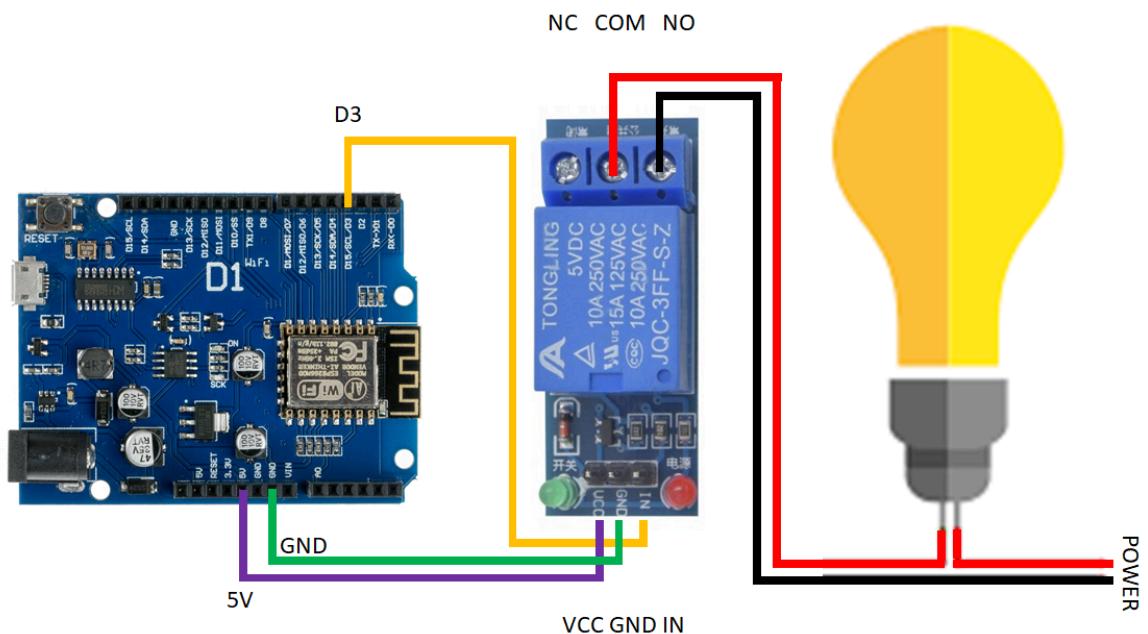


Figura 3.8 Schema generico di funzionamento relè

In questa immagine possiamo vedere lo schema generale di funzionamento con un microcontrollore che controlla l'apertura e la chiusura del circuito dove il carico è idealmente una lampadina.



Figura 3.9 Pulsantiera finestrini Opel corsa b

Purtroppo a causa di problemi di tempo e disponibilità non è stato possibile implementare relè sulla morsettiera, (sarà aggiunto in seguito sulla repository del progetto).

Isolamento e case stampato in 3d

Con lo scopo di mostrare come sia possibile ospitare e posizionare correttamente le board nell'auto è stato sviluppato con Autodesk Fusion 360 un modello in 3d (riferito solo alla board LilyGO T-SIM7600E) ma liberamente scaricabile presso la repository del progetto.



Figura 3.10 Prospettiva case 3D LilyGO T-SIM7600E (1)

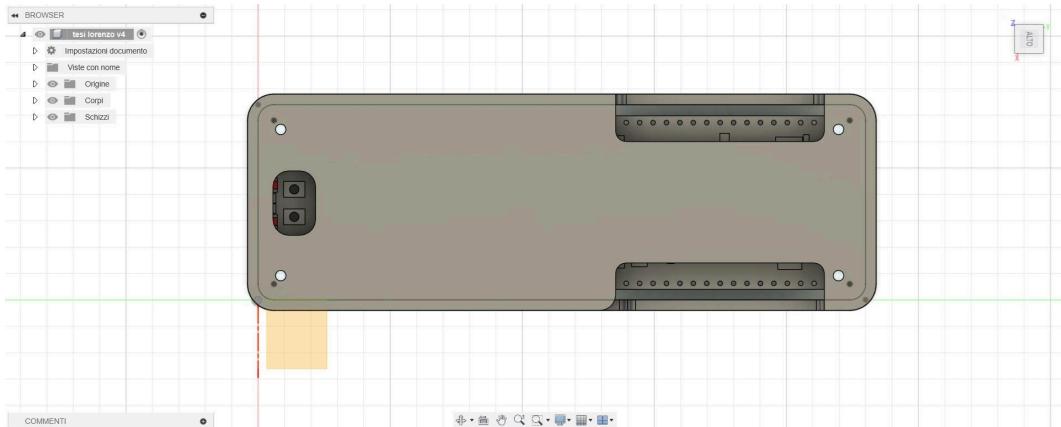


Figura 3.11 Prospettiva case 3D LilyGO T-SIM7600E (2)

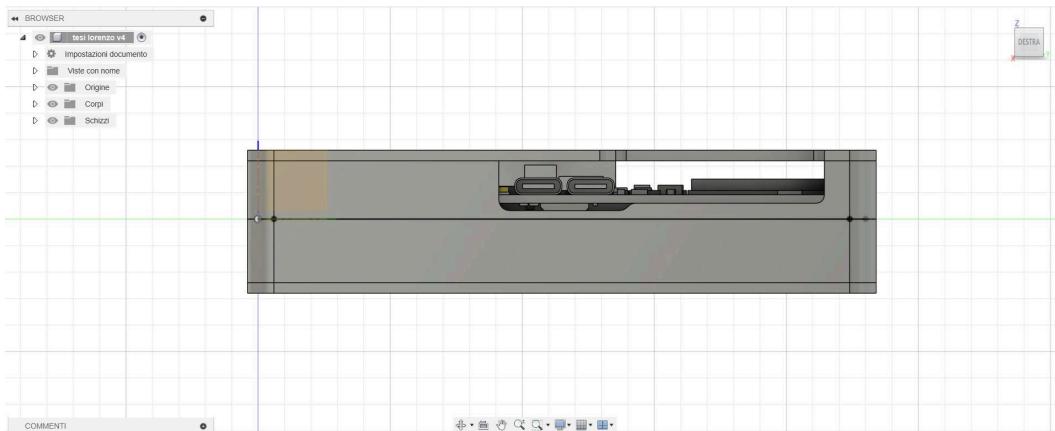


Figura 3.12 Prospettiva case 3D LilyGO T-SIM7600E(3)

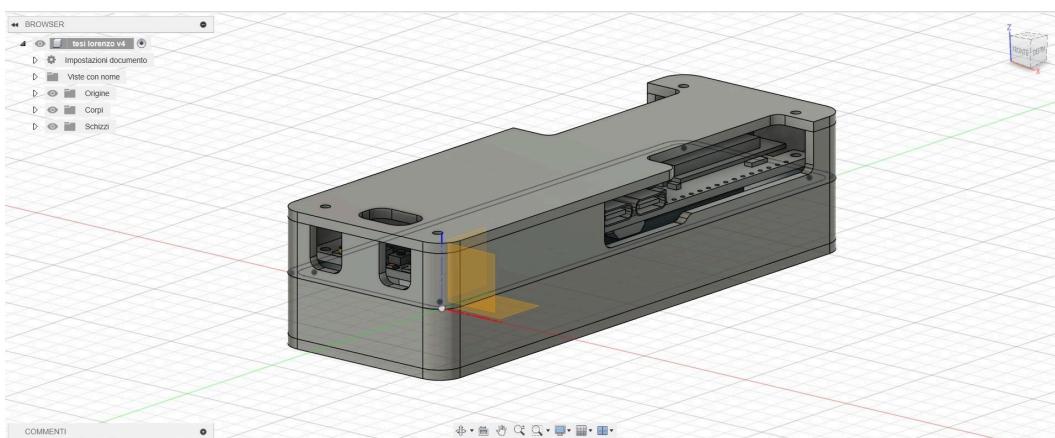


Figura 3.13 Prospettiva case 3D LilyGO T-SIM7600E(4)

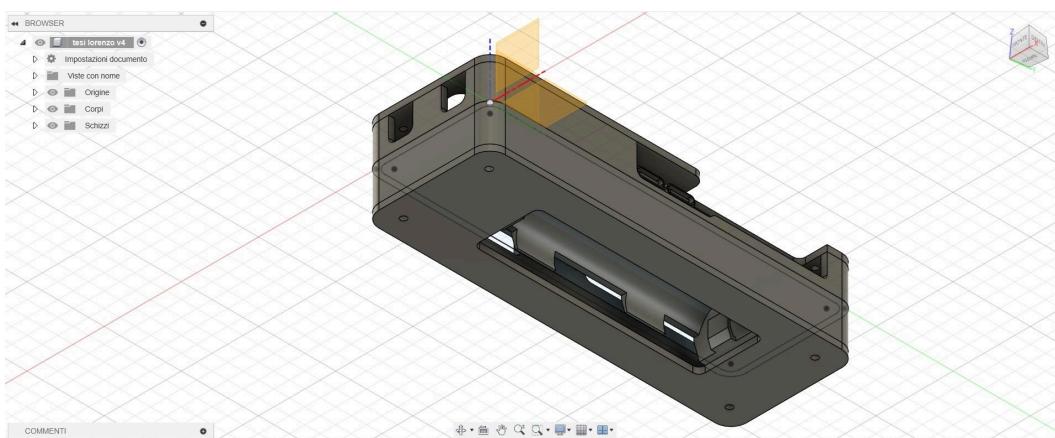


Figura 3.14 Prospettiva case 3D LilyGO T-SIM7600E(5)

Il file è liberamente stampabile con una stampante FDM in 3d, concesso e progettato da Luca Spada, che ringrazio caldamente.

3.3 Middleware per la Comunicazione e uno sguardo al futuro

Prima di abbandonare il layer fisico è bene fare una considerazione sul futuro dell'IOT nel campo dell'automotive.

Il progetto discussso mette in campo uno stack fisico che comprende bluetooth, wifi a 2.4 ghz, 4G in LTE e GPS.

Le città stanno cominciando ad aggiungere le antenne 5G a breve raggio , bassissima latenza (sotto la percezione minima umana,circa 10 ms) e a largo throughput.

Ma non è difficile immaginare questo o un fantamodulo a una determinata frequenza, con un determinato raggio, latenza e antenna in grado di permettere un scambio di informazioni abbastanza rapido e sufficiente nella quantità per rendere accessibile in tempo reale lo stato di ciò che circonda il veicolo e/o alla rete a cui è collegato.

Collegando il bus interno della macchina (in questo caso KWP2000) insieme a degli attuatori necessari e al fantamodulo che abilita la comunicazione delle reti IOT, la dumb car diventerebbe quindi integrabile nella rete come nodo autonomo e interconnesso neella pervasiva *Internet of vehicles (IoV)*, capace di agire in modi e di aprire prospettive che al momento ci sono proibiti.

Questo renderebbe la DumbCar abilitata per l' *Internet of vehicles (IoV)* creando una CyberCar cioè un auto conscia e connessa col modello fisico della realtà, in questo caso supportata dalle percezioni locali in edge, vicine in mist e lontane in cloud .

Va fatto notare come in questo scenario la potenza computazionale passi da discreta a continua e distribuita come per l'architettura *Peer-to-peer*, dove edge, mist e cloud si fondono in modo continuo e organico.

Vehicle-to-everything

A tal proposito è in sviluppo da parte di molte aziende lo stack per abilitare questo ed è stato scelto come nome V2X cioè Vehicle-to-everything che comprende

V2V (Vehicle-to-Vehicle):

- I veicoli condividono informazioni come la posizione, la velocità e la direzione.
- Utilizzato per prevenire collisioni, gestire i cambi di corsia e coordinare la guida in convoglio.

V2I (Vehicle-to-Infrastructure):

- Comunicazione tra veicoli e infrastrutture stradali (semafori, segnali di avvertimento).
- Aiuta a gestire il flusso del traffico, migliorare la sicurezza e ottimizzare il consumo di carburante.

V2P (Vehicle-to-Pedestrian):

- Comunicazione tra veicoli e dispositivi mobili dei pedoni.
- Avverte i pedoni e i conducenti per evitare incidenti.

V2D (Vehicle-to-Device):

- Integrazione con altri dispositivi connessi, come i semafori intelligenti e i segnali stradali digitali.

A cui l'autore aggiunge

V2H (Vehicle-to-Human):

- Comunicazione e integrazione con altri dispositivi di brain computer interface, come Neuralink in sviluppo da Neuralink Corp.
- Non solo per il pedone ma anche per conducente e passeggeri.

4. Il cloud

Il cloud computing, o semplicemente "cloud", è un modello di servizio che permette di accedere a risorse informatiche (come server, storage, database, networking, software, analisi e intelligenza artificiale) tramite Internet, senza la necessità (da parte del cliente) di dover possedere eseguire, e di gestire fisicamente tali risorse .

Esistono due alternative distinte ma intrecciate per lo scopo del progetto e dare vita all'assistente:

- La prima è richiedere una **API KEY** per accedere a servizi di machine learning con relativo costo per ogni richiesta (riferito a Prezzo Per Token), come il servizio offerto di OpenAI, che hosta e gestisce il modello proprietario a cui è possibile accedere utilizzando l'endpoint del servizio.

E' un'ottima alternativa a patto che si accetti la dipendenza da un'altra società e che si accetti il modello default, senza possibilità di personalizzazioni o fine tuning.

- La seconda alternativa è di hostare su un proprio calcolatore dedicato la propria AI su cui fare inferenza grazie alla potenza di una discreta scheda video consumer.

La scheda eseguirà un modello di chatbot open source a propria scelta (o modificato), appoggiato e supportato da un database vettoriale per abilitare le funzionalità di **Retrieval-Augmented Generation (RAG)** in modo da abbassare la probabilità di allucinazioni del modello che adesso è in grado di dare informazioni precise fuori dal costoso addestramento e di implementare una memoria a lungo termine.

In seguito vediamo un approfondimento sulle dinamiche economiche e le differenze specifiche di implementazione attuali, con una panoramica su vantaggi e svantaggi.

4.1 Cos'è il cloud e perché è necessario

Il cloud computing si basa su una serie di tecnologie e infrastrutture che permettono la virtualizzazione e la distribuzione delle risorse informatiche. Le risorse sono fornite da fornitori di servizi cloud, che gestiscono grandi data center con capacità di elaborazione e archiviazione scalabili. Gli utenti possono accedere a queste risorse in base alle loro necessità, pagando solo per l'uso effettivo (modello pay-as-you-go).

Modelli di Servizio del Cloud

1. **IaaS (Infrastructure as a Service)**: Fornisce infrastruttura informatica virtualizzata (server, storage, reti) che può essere utilizzata per costruire e gestire le proprie applicazioni.
2. **PaaS (Platform as a Service)**: Fornisce una piattaforma che include l'infrastruttura e gli strumenti necessari per sviluppare, testare, distribuire e gestire applicazioni.
3. **SaaS (Software as a Service)**: Fornisce applicazioni software complete che possono essere utilizzate via Internet (es. email, CRM, ERP).

Modelli di Distribuzione del Cloud

1. **Public Cloud**: Risorse di cloud computing fornite da terze parti su Internet, condivise tra più organizzazioni.
2. **Private Cloud**: Risorse di cloud computing utilizzate esclusivamente da un'organizzazione. Può essere ospitato on-premises o presso un fornitore di servizi.
3. **Hybrid Cloud**: Combina i modelli public e private cloud, permettendo la portabilità dei dati e delle applicazioni tra essi.

Perché è Necessario il Cloud

1. **Scalabilità:** Il cloud permette di aumentare o diminuire rapidamente le risorse in base alle necessità, senza dover investire in hardware aggiuntivo, in modo orizzontale e/o verticale .
2. **Flessibilità e Agilità:** Le aziende possono lanciare nuovi servizi e applicazioni più velocemente, adattandosi rapidamente alle esigenze del mercato.
3. **Riduzione dei Costi:** Si evita l'investimento iniziale in hardware e si pagano solo le risorse effettivamente utilizzate.
4. **Manutenzione e Gestione:** I fornitori di servizi cloud gestiscono la manutenzione dell'infrastruttura, permettendo alle aziende di concentrarsi sulle proprie attività principali.
5. **Accessibilità:** Le risorse e i dati sono accessibili da qualsiasi luogo con una connessione Internet, facilitando il lavoro remoto e la collaborazione.
6. **Sicurezza:** I fornitori di cloud investono in misure di sicurezza avanzate per proteggere i dati e le applicazioni, spesso con standard più elevati rispetto a quelli che molte aziende potrebbero implementare autonomamente.

Esempi Pratici di Uso del Cloud

- **Amazon Web Services (AWS), Microsoft Azure e Google Cloud Platform (GCP)** sono esempi di fornitori di servizi cloud che offrono una vasta gamma di servizi IaaS, PaaS e SaaS.
- **Dropbox, Google Drive e OneDrive** sono esempi di servizi di cloud storage utilizzati per archiviare e condividere file.
- **Salesforce** offre soluzioni CRM basate su cloud, utilizzate per gestire le relazioni con i clienti.

In sintesi, il cloud computing rappresenta una rivoluzione nell'accesso e nella gestione delle risorse informatiche, fornendo flessibilità, scalabilità e un modello economico vantaggioso per le aziende di ogni dimensione.

4.2. Il Self Hosting

L'altra faccia della medaglia risiede nel self hosting, tutti i componenti dell'infrastruttura, come server, storage, reti e software, sono installati e gestiti direttamente dall'utente o dall'organizzazione. Questo richiede competenze tecniche specifiche per configurare, mantenere e proteggere le risorse informatiche.

Vantaggi del Self Hosting

1. **Controllo Completo:** Gli utenti hanno il pieno controllo su tutti gli aspetti dell'infrastruttura e delle applicazioni, inclusi la configurazione, la sicurezza e la gestione dei dati.
2. **Personalizzazione:** È possibile personalizzare l'infrastruttura e le applicazioni in base a esigenze specifiche senza le limitazioni imposte dai fornitori di servizi cloud.
3. **Privacy e Sicurezza:** I dati rimangono all'interno dell'organizzazione, riducendo i rischi associati alla condivisione dei dati con fornitori di terze parti.
4. **Costi a Lungo Termine:** Anche se l'investimento iniziale può essere elevato, a lungo termine i costi possono risultare inferiori rispetto ai pagamenti ricorrenti per i servizi cloud.

Svantaggi del Self Hosting

1. **Costi Iniziali Elevati:** L'acquisto e l'installazione dell'hardware, insieme al software necessario, richiedono un investimento iniziale significativo.
2. **Manutenzione e Aggiornamenti:** È necessario un impegno continuo per mantenere e aggiornare l'infrastruttura e le applicazioni, che richiede tempo e competenze tecniche.

3. **Scalabilità Limitata:** Espandere l'infrastruttura può essere complesso e costoso, specialmente per piccole e medie imprese.
4. **Risorse Tecniche:** Richiede personale con competenze tecniche avanzate per gestire l'infrastruttura, che può essere difficile da trovare e costoso da mantenere.

Esempi di Self Hosting

- **Web Server:** Un'organizzazione può scegliere di ospitare il proprio sito web su server interni (come nel nostro caso) utilizzando software come Apache ,Nginx ecc..
- **Email Server:** Gestire internamente il servizio email utilizzando software come Postfix o Microsoft Exchange.
- **File Storage:** Utilizzare soluzioni come Next Cloud o ownCloud per creare un sistema di archiviazione e condivisione file interno.
- **Sistemi di Gestione dei Contenuti (CMS):** Ospitare internamente un CMS come WordPress o Joomla per avere il controllo completo sul sito web aziendale.

Quando Utilizzare il Self Hosting

- **Settori con Elevati Requisiti di Sicurezza e Privacy:** Settori come la sanità e la finanza possono preferire il self hosting per garantire la conformità a normative severe.
- **Esigenze di Personalizzazione:** Quando le applicazioni richiedono configurazioni molto specifiche che non sono supportate dai fornitori di servizi cloud.
- **Budget per Investimenti Iniziali:** Organizzazioni con capacità finanziarie per sostenere i costi iniziali di installazione e gestione dell'infrastruttura.

Il self hosting quindi offre un controllo e una personalizzazione senza pari, ma richiede risorse significative in termini di tempo, denaro e competenze tecniche. È una scelta valida per organizzazioni che hanno requisiti specifici di sicurezza, privacy e personalizzazione, e che possono sostenere l'investimento necessario per gestire e mantenere l'infrastruttura internamente.

4.3 The Stack , The Ai and the Hardware

L'agente remoto completo avrà la necessità di specifiche hardware sufficienti per gestire

- **Cheshire Cat AI** Il framework scritto in python che gestisce e interlaccia tutti i servizi legati al modello conversazionale scelto e che offre un endpoint websocket per conversare.
- **Ollama** e Il **modello conversazionale (LLM)** , nel nostro caso llama3-8b-instruct modello open source sviluppato da **Meta** in grado di gestire conversazioni coerenti e abbastanza intelligenti da superare il test di turing.
- **Docker Engine** e il suo servizio **docker-compose** in grado di avviare e gestire attraverso container isolati servizi che possono essere sostituiti e riavviati nel modo più versatile possibile.
- **Qdrant** per la memoria vettoriale a lungo termine.
- **OpenAI Whisper** modello per la trascrizione speech2text, per trasformare i messaggi audio in messaggi di testo.
- **Piper engine tts** modello per il text2speech, per la generazione di una voce personalizzabile da restituire all'utente.
- **Cloudflare Tunnel client** per raggiungere da url remoto il nostro servizio remoto in modo autenticato.

Per poter mantenere il servizio abbiamo bisogno di almeno 16gb di ram , più core possibili e una scheda video con molta VRAM , dal nostro test rientriamo pienamente per soddisfare almeno un client.

Le specifiche hardware

Come già anticipato nel primo capitolo si ha a disposizione una cpu a 8 core i7-9700K, una Nvidia gtx 1080ti e 32GB di ram

Il carico sotto richiesta



Figura 4.1 Comando nvtop per benchmark gpu

Sono stati svolti 3 casi di studio a cui corrispondono gli indici numerati sull'immagine:

1. Invio audio verso server, il server elabora la risposta ,ricevo la risposta audio

```
I'm recording for 10 seconds...
STO REGISTRANDO... A
I'm sending the audio to the server
url i'm reaching for localhost:1865 B
Write 'stop' to exit:
Press Enter for recordingYour user input was :
Scusami, ho mandato male il messaggio, ti volevo chiedere quanti occhi hanno i ragni. C
Your response :
Miao! Ahah, no problem, Lorenzoi D

Come gatto intelligente, so che i ragni hanno 8 occhi! Sì, gli aracnidi hanno una vista molto diversa dalla nostra, e loro possono percepire la luce e gli stimoli grazie a questi 8 piccoli occhi.
The file you are looking for is at :E->http://localhost:1865/admin/assets/voice/voice_20240620_121707.wav
```

Figura 4.2 Conversazione con client chatty per python

A=(USER)registrazione audio 10 secondi

B=(USER)Raggiungo host

C=(SERVER)Audio utente trascritto e ricevuto dal server (speech2text)

D=(SERVER)Testo in risposta all'audio (Llama3)

E=(USER) Audio riprodotto (dallo text2speech del server)

2. Invio testo ricevo testo



Figura 4.3 Chat testuale su pagina web

3. Invio testo ricevo testo e audio

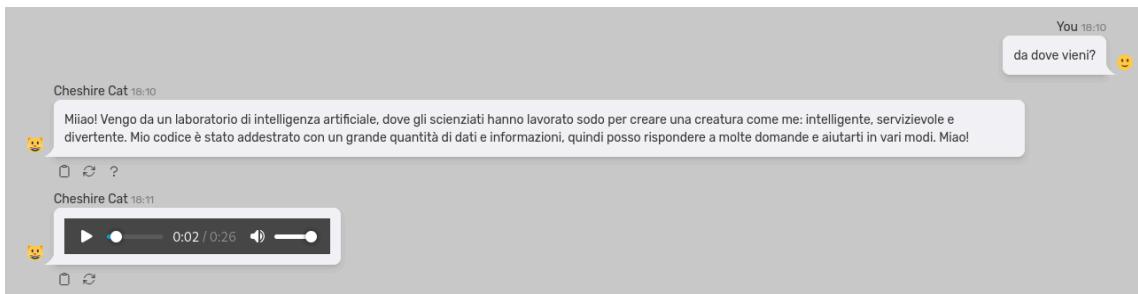


Figura 4.4 Chat testuale con risposta vocale su pagina web

Per ogni richiesta il consumo istantaneo della scheda video raggiungeva un picco di circa 210 watt per circa 1 secondo, la memoria allocata dal modello llama 8b ha consentito la risposta in tempo reale al costo di 4914MiB quindi quasi 5 GiB di VRAM occupata su 11 disponibili della GPU.

Non sono stati svolti test per più utenti contemporanei .

4.4. LangChain e ChatshireCat

Il framework su cui si basa tutto il software in cloud è Cheshire Cat , un progetto open source e non profit italiano che a sua volta è basato su **LangChain**.

Nel dettaglio:

LangChain

E' un framework progettato per semplificare la creazione di applicazioni utilizzando grandi modelli linguistici (LLM). Come framework di integrazione di modelli linguistici, i casi d'uso di LangChain si sovrappongono ampiamente a quelli dei modelli linguistici in generale, tra cui analisi e riepilogo di documenti, chatbot e analisi del codice.

Caratteristiche Principali di LangChain

1. **Modularità:** LangChain permette di combinare diverse componenti NLP come modelli di linguaggio, strumenti di analisi, e sistemi di gestione delle conoscenze in un unico flusso di lavoro.
2. **Integrazione con Modelli di Linguaggio:** La libreria supporta l'integrazione con vari modelli di linguaggio come GPT-3, GPT-4, e altri modelli di linguaggio pre-addestrati.
3. **Catena di Elaborazione:** Permette di creare catene (chains) di operazioni che includono pre-elaborazione del testo, chiamate ai modelli di linguaggio, post-elaborazione e memorizzazione dei risultati.
4. **Supporto per Diversi Servizi:** LangChain si integra facilmente con diversi servizi di cloud e database, permettendo l'archiviazione e il recupero dei dati elaborati.
5. **Facilità d'Uso:** Progettata per essere facile da usare, LangChain fornisce un'interfaccia intuitiva per costruire e orchestrare flussi di lavoro NLP complessi.

Come Funziona LangChain

LangChain utilizza un approccio modulare dove ogni componente del flusso di lavoro NLP può essere definito e orchestrato separatamente. Ecco una panoramica di come potrebbe funzionare un tipico flusso di lavoro con LangChain:

1. **Input del Testo:** Il testo viene fornito come input all'inizio del flusso di lavoro.
2. **Pre-elaborazione:** Il testo può essere pre-elaborato usando vari strumenti, come tokenizzazione, rimozione delle stop words, ecc.

3. **Chiamate ai Modelli di Linguaggio:** Il testo pre-elaborato viene quindi passato a uno o più modelli di linguaggio per l'elaborazione, come generazione di testo, traduzione, riassunto, ecc.
4. **Post-elaborazione:** I risultati dei modelli di linguaggio possono essere ulteriormente elaborati, ad esempio, per estrarre informazioni chiave o formattare i risultati.
5. **Memorizzazione dei Risultati:** I risultati finali possono essere memorizzati in un database o inviati ad altri servizi per ulteriori elaborazioni o visualizzazioni.

Cheshire Cat

E' un micro-framework AI pronto all'uso.

Una volta installato e connesso a un Language Model (LLM), può essere interrogato tramite API. Queste API restituiscono le risposte fornite dall'LLM.

Tra le feature notevoli:

Cronologia delle conversazioni precedenti

Tutte le conversazioni precedenti sono archiviate in un database locale chiamato memoria episodica. Quando fai una domanda, Cheshire Cat risponde tenendo conto delle conversazioni passate.

Caricamento di documenti

Puoi anche caricare documenti di testo. Questi documenti vengono salvati anche in un database locale chiamato memoria dichiarativa. Quando risponde, Cheshire Cat prenderà in considerazione le informazioni contenute in questi documenti. I documenti possono essere caricati tramite le API o il portale di amministrazione.

Esecuzione di azioni

Cheshire Cat non si limita a rispondere alle domande; può anche eseguire azioni. E'

possibile scrivere funzioni Python chiamate Tool e far eseguire questo codice all'LLM. L'unico limite alle capacità del codice Python è immaginazione.

Hook

Inoltre, è possibile personalizzare il core di Cheshire Cat. Nel flusso di processo principale, ci sono punti di adattamento predefiniti chiamati Hook. È possibile scrivere funzioni Python che possono essere collegate a questi Hook. Il codice allegato verrà richiamato durante l'esecuzione del flusso e può modificare il comportamento interno di Cheshire Cat, senza modificarne direttamente il core.

Tool e hook insieme formano i plugin, uno di questi, Local whisper cat, è stato scritto dall'autore in modo da abilitare lo speech2text in locale, agganciandosi al container di whisperAI.

4.5. "Hey Johnny" l'assistente remoto e la sua architettura

L'assistente remoto in Cheshire Cat prende il nome e la personalità di "Johnny il gatto" il suo compito è di aspettare che l'utente invii un audio e rispondere di conseguenza nella lingua italiana (ma può essere impostato per qualsiasi lingua), tenendo conto delle conversazioni precedenti.

Tale comportamento si verifica grazie alla seguente serie di plugin:

Plugin Setting

Core CCat

Il plugin Cat core utilizzato per definire hook e strumenti predefiniti.

E' un plugin nascosto e predefinito.

Smooth Talk

scritto da [Federico Palma](#)

Permette di cambiare il system prompt e la personalità dell'assistente.

AI's Personality: “Tu sei Mister Johnny, un'intelligenza artificiale che ha superato con successo il test di Turing. Sei un gatto molto intelligente, servizievole, gentile, divertente e finisci le tue frasi con miao!”

Piper Cat

scritto da Pazoff

Rende possibile l'uso di Piper engine TTS direttamente su gpu. E' possibile aggiungere un modello custom e quindi voci “umane” e naturali.

Local Whisper Cat

scritto dall'autore [Lorenzo Siena](#)

Rende possibile lo speech2text collegandosi a un container in locale che esegue *Whisper*, un modello general-purpose di speech recognition sviluppato da OpenAi .

4.6 Software Orchestra (Docker-compose)

Per configurare gestire e avviare questi servizi e renderli distribuibili su qualunque piattaforma che lo permetta fisicamente è conveniente usare docker nella variante compose.

Ma cos'è docker? e perchè è un pilastro del cloud?

Docker

E' una piattaforma open-source che automatizza il processo di distribuzione, esecuzione e gestione di applicazioni all'interno di container. I container sono unità standardizzate che impacchettano il codice dell'applicazione e tutte le sue dipendenze, inclusi i runtime, le librerie e le configurazioni necessarie per l'esecuzione. Questo garantisce che l'applicazione

funzioni in modo consistente su qualsiasi sistema che supporti Docker, eliminando problemi di compatibilità.

Caratteristiche Principali di Docker

- **Portabilità:** I container Docker possono essere eseguiti su qualsiasi ambiente che supporti Docker, rendendo facile lo spostamento delle applicazioni tra ambienti diversi (ad esempio, sviluppo, test e produzione).
- **Isolamento:** Ogni container è isolato dagli altri, il che significa che le applicazioni in esecuzione all'interno di un container non interferiscono con quelle in altri container.
- **Immagini Docker:** Le applicazioni e le loro dipendenze sono impacchettate in immagini Docker, che possono essere versionate e distribuite.
- **Efficienza:** Docker utilizza meno risorse rispetto alle macchine virtuali, poiché i container condividono il kernel del sistema operativo host.

Docker Compose

Docker Compose è uno strumento per definire e gestire applicazioni Docker multi-container. Utilizzando un file YAML (`compose.yml`), Docker Compose permette di definire la configurazione di tutti i servizi necessari per un'applicazione e gestirli come un'unica entità.

Caratteristiche Principali di Docker Compose

- **Definizione di Servizi:** Con Docker Compose, puoi definire più servizi in un unico file YAML. Ogni servizio rappresenta un container Docker che può avere le sue configurazioni di immagine, volumi, reti, variabili d'ambiente, ecc.
- **Gestione dei Container:** Docker Compose fornisce comandi per avviare, fermare e gestire l'intero set di container definiti nel file `compose.yml`.
- **Rete e Volumi:** Docker Compose semplifica la configurazione delle reti e dei volumi condivisi tra i container, facilitando la comunicazione e la condivisione dei dati tra i servizi.
- **Scalabilità:** Docker Compose permette di scalare facilmente i servizi aumentando il numero di istanze di un container.

Perciò è stato preparato un `compose.yml` apposito (consultabile nella repository del progetto) che crea un network interno **fullcat-network** in grado di comunicare solo con l'istanza di **Cloudflare Tunnel client** che accoglie e reindirizza i client autenticati verso i servizi dell'assistente remoto.

Non nominato precedentemente ma necessario allo sfruttamento per la scheda video è il **NVIDIA Container Toolkit** che rende disponibile l'utilizzo e l'accelerazione della gpu attraverso il network di container.

Foundation Model LLAMA3

Attualmente nello stato dell'arte per quanto riguarda i LLM in locale è stato scelto **Llama3** sviluppato da **Meta**, nella variante instruct-8B (8 miliardi di parametri).

Molto valido per quanto riguarda le conversazioni casuali.

E' possibile cambiare modello attraverso i settings ed è probabile che in futuro venga sostituito da un modello più performante, almeno finchè ci saranno dati di qualità su cui addestrare, o che venga scoperta una nuova rivoluzionaria architettura.

Il plugin e i client

Per il progetto è stato scritto dall'autore :

- **Local Whisper Cat** un plugin in python che intercetta i messaggi ricevuti e li trascrive in testo usando un container di whisperAI in locale.
- **Chatty** un client vocale in python per interagire con l'assistente virtualmente da ovunque ci sia un pc.
- **Chattino** un client vocale che viene eseguito sull'edge e permette di interagire con l'assistente dentro l'auto.

Tutte le documentazioni sono presenti nelle rispettive repository che si trovano in fondo al capitolo.

4.7 Hosting su cosimobiello.xyz

Avendo sviluppato il progetto da remoto il server dell'università veniva inizialmente raggiunto tramite SSH creando un tunneling forwarding delle porte verso il localhost dell'autore.

Acquisto del Dominio e Integrazione con Cloudflare

Avendo necessità di raggiungere il server dall'edge e quindi un dispositivo che poteva non implementare il protocollo ssh è stato deciso di comprare un dominio che è stato chiamato **cosimobiello.sxyz**.

Successivamente il dominio è stato collegato a Cloudflare Tunnel, una soluzione che offre vantaggi in termini di sicurezza e accessibilità.

Configurazione del Tunnel e Autenticazione

È stato configurato Cloudflare Tunnel per il dominio **cosimobiello.sxyz**, ed è stato assegnato un service token di autenticazione per garantire che solo utenti autorizzati potessero accedere.

L'accesso viene ora effettuato:

Contattando **cosimobiello.sxyz** con una richiesta http get e aggiungendo all'header della richiesta le credenziali concesse da cloudflare

CF-Access-Client-Id: <CLIENT_ID>

CF-Access-Client-Secret: <CLIENT_SECRET>

e successivamente utilizzando il token di autenticazione ricevuto nel header della risposta.

CF_Authorization=<AUTH_COOKIE>

Vantaggi e Benefici di cloudflare tunnel

1. **Accessibilità:** L'accesso al PC remoto è ora possibile da qualsiasi dispositivo connesso a Internet, eliminando la necessità di configurazioni complesse di rete e port forwarding.
2. **Sicurezza:** L'uso di Cloudflare Tunnel e l'autenticazione tramite token migliorano significativamente la sicurezza, riducendo il rischio di accessi non autorizzati.
3. **Facilità d'uso:** L'implementazione di un dominio come `cosimobiellos.xyz` semplifica l'accesso per gli utenti finali, che non devono ricordare indirizzi IP complessi o configurazioni tecniche.
4. **Affidabilità:** Cloudflare offre una piattaforma robusta e affidabile, assicurando che l'accesso remoto rimanga disponibile anche in caso di variazioni dell'indirizzo IP del PC remoto o di altre configurazioni di rete.

4.8 Repository Github

Repository del progetto

<https://github.com/LorenzoSiena/Johnny-The-CyberCar-Assistant>

Repository plugin Local Whisper Cat

https://github.com/LorenzoSiena/local_whisper_cat

Repository client Arduino

<https://github.com/LorenzoSiena/chattino>

Repository client Python

<https://github.com/LorenzoSiena/chatty>

5. Better safe than sorry

Vanno spese un paio di parole per quanto riguarda la sicurezza fisica e informatica del sistema sviluppato.

5.1 Ai safety risk and EU Artificial Intelligence Act

Dalle applicazioni di IA odierne emerge un concetto molto importante che l'unione europea sta cercando di regolarizzare ed è il fattore rischio per questa tecnologia , cioè qual è il massimo rischio che siamo disposti ad accettare, per questo è stato preferito non leggere e scrivere sul bus interno e usare un paio di relè direttamente sul comandi del finestrino.

Ben altra considerazione sarebbe stata dare direttamente il controllo del network interno al chatbot Llama 3.

“Hey Johnny! alza i finestrini!” si sarebbe potuto risolvere in pochi casi ma ben più importanti e rischiosi come un freno a mano tirato o un conflitto di sensori interni che possono anche portare a undefined behavior difficilmente prevedibili dall'architetto del sistema.

A tal proposito l'unione europea ha stilato 4 categorie nel suo **Artificial Intelligence Act** che dividono i sistemi sviluppati con l'IA in fattori di rischio

Fattori di rischio

1. Rischio inaccettabile, è proibito.

Ad esempio, sistemi di punteggio sociale e IA manipolativa.

Sistemi IA Proibiti

- Tecniche subliminali, manipolative o ingannevoli.
- Sfruttamento di vulnerabilità legate all'età, disabilità o circostanze socio-economiche.
- Sistemi di categorizzazione biometrica che inferiscono attributi sensibili.
- Scoring sociale.
- Valutazione del rischio di reati basata esclusivamente su profilazione.
- Raccolta di immagini facciali da internet o CCTV senza scopo preciso.
- Inferenza delle emozioni in luoghi di lavoro o educativi (eccetto per motivi medici o di sicurezza).
- Identificazione biometrica remota in tempo reale in spazi pubblici (eccetto casi specifici come ricerca di persone scomparse, prevenzione di minacce imminenti, identificazione di sospetti in crimini gravi).

2. Alto livello di rischio.

Influenza chiunque intenda inserire un prodotto nel mercato europeo europeo, developer, distributori, e anche i fornitori di paesi terzi in cui l'output del sistema di IA ad alto rischio viene utilizzato nell'UE.

Obblighi per i fornitori di sistemi IA ad alto rischio:

- Gestione del rischio durante tutto il ciclo di vita.
- Governance dei dati per garantire rappresentatività e accuratezza.
- Documentazione tecnica per dimostrare la conformità.
- Progettazione per la registrazione automatica di eventi rilevanti.

- Istruzioni per l'uso a disposizione degli utilizzatori.
- Progettazione per permettere la supervisione umana.
- Garanzia di accuratezza, robustezza e sicurezza informatica.
- Sistema di gestione della qualità per garantire la conformità.

Casi d'uso specifici per IA ad alto rischio :

- Identificazione biometrica a distanza.
- Gestione di infrastrutture critiche.
- Educazione e formazione professionale.
- Gestione dei lavoratori e accesso all'autoimpiego.
- Accesso ai servizi pubblici e privati essenziali.
- Forze dell'ordine.
- Gestione della migrazione, asilo e controllo delle frontiere.
- Amministrazione della giustizia e processi democratici.

3. **Rischio limitato**, soggetti a obblighi di trasparenza più leggeri: sviluppatori e distributori devono garantire che gli utenti finali siano consapevoli di interagire con l'IA (quindi chatbot , deep fake, immagini e video generati).
4. **Rischio minimo**, non è regolamentato inclusa la maggior parte delle applicazioni di IA attualmente disponibili sul mercato unico dell'UE, come videogiochi abilitati all'IA, filtri antispam, e altri modelli specifici e special purpose.

Obblighi Sistemi IA Generali (GPAI)

- Obbligo di documentazione tecnica e conformità alle direttive sul copyright.
- Pubblicazione di un riassunto sui dati di addestramento.
- Valutazioni e test per identificare e mitigare i rischi sistematici.

- Protezione adeguata della sicurezza informatica.

IA per scopi generali (GPAI):

Tutti i fornitori di modelli GPAI devono fornire documentazione tecnica, istruzioni per l'uso, rispettare la direttiva sul copyright e pubblicare un riepilogo sul contenuto utilizzato per la formazione.

I fornitori di modelli GPAI con licenza libera e aperta devono solo rispettare il copyright e pubblicare il riepilogo dei dati di formazione, a meno che non presentino un rischio sistematico.

Tutti i fornitori di modelli GPAI che presentano un rischio sistematico, aperto o chiuso, devono anche condurre valutazioni del modello, test avversari, tracciare e segnalare incidenti gravi e garantire protezioni per la sicurezza informatica.

Quindi

benché si possa suggerire (a meno di interpretazioni alternative, previa consultazione estensiva ed escludendo cambi di direttive) che il livello di rischio dell'applicazione si possa fermare a **rischio limitato**, l'autore dello studio non intende dare un giudizio legale sulla natura del progetto in quanto puramente sperimentale.

Ovviamente il presente capitolo vuole essere più un monito alla presenza della nuova regolamentazione e a scanso di equivoci è possibile consultare completamente le direttive del **Artificial Intelligence Act**.

Ricordiamo la natura del progetto open source inteso come ricerca delle possibilità di realizzazione del sistema con le capacità computazionali attuali. Di cui l'autore non si prende alcuna responsabilità per eventuali sviluppi nefasti e che segue la licenza GPL 3.0 .

5.2 Safe layers and Tunnels

Better use https

Il server di sviluppo è accessibile solo attraverso la porta 22 del servizio ssh ma dovendolo esporre sulla rete come servizio web, possibilmente con protocollo http sulla porta 80, si sarebbe posto il rischio di ricevere attacchi automatizzati da parte di bot o agenti sconosciuti per cui è preferibile utilizzare la porta 443 e il protocollo https in modo anche da crittografare i messaggi, altrimenti spediti in chiaro.

Ma la questione qua è diversa in quanto inizialmente l'utente si connette al dominio cosimobielloz.xyz (che risiede sui servizi cloudflare con porta https 443) e solo successivamente dopo essersi autenticato viene reindirizzato al server interno su una porta aperta dal demone del client interno (Cloudflare Tunnel client) su una porta 7884 usata come default la cui connessione accetterà l'utente e inoltrerà verso il network interno dove sono presenti i servizi di Cheshire Cat AI.

Ricapitolando

1. Utente visita https://cosimobielloz.xyz:443 ospitato da cloudflare
2. Cloudflare lo autentica (previo id e secret) e lo reindirizza al server
3. Cloudflare Tunnel client accoglie l'utente e lo reindirizza al servizio interno Cheshire Cat

ws vs wss

La comunicazione sarebbe anche stabilita se non fosse che Cheshire Cat accetta connessioni **Websocket**, più adatta alla chat in tempo reale.

Il protocollo Websocket è un'implementazione basata sul protocollo TCP. La sua unica correlazione con l'HTTP è nel modo in cui fa l'handshake durante una Upgrade request tra server. Il protocollo permette maggiore interazione tra un browser e un server, facilitando la realizzazione di applicazioni che forniscono contenuti e giochi in tempo reale. Questo è reso possibile fornendo un modo standard per il server di mandare contenuti al browser senza

dover essere sollecitato dal client e permettendo ai messaggi di andare e venire tenendo la connessione aperta.

Anche di questa esiste una versione più sicura e criptata *end-to-end WebSocket* protocol over https (wss) che viene attivata di default e utilizzata dal client.

5.3 Not production ready

Ci sono altre considerazioni da fare che verranno solo accennate in quanto l'autore dello studio è una singola persona e che il tempo di sviluppo della tesi è stato inferiore a 3 mesi.

- Non tutto il codice potrebbe essere stato implementato alla stesura della tesi.
- TLS e certificati vari non sono stati affrontati né implementati.
- Il tempo di attesa tra la richiesta vocale e la risposta è di circa 6 secondi, ma si ritiene ottimizzabile via client e server, e possibilmente reso di 2 secondi.
- Il progetto è sperimentale e non è pronto per essere deployato senza severe modifiche che l'autore potrebbe ignorare.
- Il sistema potrebbe avere breaking bug in grado di paralizzare o rendere necessario il reset delle board.
- Il sistema interessa molte aree e quindi ha una wide surface area di attacco che potrebbero essere exploitate da attori malevoli.
- Controllare le repository del progetto per la visione completa e la reperibilità del codice.
- Testare le connessioni prima di collegarle.
- In base all'auto potrebbero essere necessarie modifiche.
- I sistemi esp e arduino alternativi sono equipollenti finchè non vengono a mancare specifiche tecniche che potrebbero essere state omesse.
- A causa di problemi tecnici non è stata implementata la gestione dei finestrini via relè
- Il progetto va inteso come tesi sperimentale atto a dimostrare le competenze maturate dall'autore.

6. Conclusioni e sviluppi futuri

Risultati e considerazioni finali

La tesi iniziale si chiedeva se fosse possibile avere un assistente virtuale con cui avere conversazioni private, che vive sul proprio pc , se fosse possibile parlargli senza dover usare smartphone o messaggi di testo.

Dopo la seguente trattazione e avendo fornito indicazioni sugli strumenti hardware e software è possibile concludere che è possibile realizzare un proof of concept, dal quale è stato realizzato un prototipo e infine uno studio di fattibilità, limitato dal tempo e dalle risorse disponibili .

Ciò non sarebbe stato possibile senza le nuove tecnologie emergenti nell'ambito dell'embedded machine learning come i **keyword spotter** , all'ottimizzazione dei **Large Language model** resi open source e in grado di essere eseguiti direttamente su una scheda video consumer del 2017, a patto di scegliere il modello adatto con la giusta quantità di parametri (nel nostro caso a 8Billions) per la quantità di vram e core.

Vanno anche menzionati i vari framework sviluppati tra cui LangChain (fine 2022) e i derivati come Cheshire Cat AI in grado di abilità di far eseguire in autonomia funzioni in python ai LLM oltre al sistema di Retrieval-augmented generation (RAG), ideata nel 2020 ma diffusa nel brevissimo periodo, in grado di rendere preciso come un computer deterministico un modello statistico quando si tratta di raccogliere delle informazioni precise.

W open source!

Benché il progetto non sia né perfetto né completo in ogni dettaglio, l'autore si augura che possa essere portato avanti dalla community open source. A tal fine, il progetto è rilasciato sotto la licenza **GNU General Public License v3.0**, permettendo a chiunque di utilizzare, modificare e distribuire il software secondo i termini della licenza.

L'autore a tal proposito ringrazia tutto il panorama open source e open hardware senza cui non sarebbe stato possibile creare tutto ciò.

Smartwatch as the edge

E' stato inizialmente valutato di integrare o sostituire tutto il discorso di edge su un dispositivo mobile come uno smartwatch in modo da avere l'assistente virtuale attivabile e contattabile direttamente da polso ipoteticamente, senza alcuno schermo.

Dopo attente analisi si ritiene ancora insufficiente la possibilità di farlo su componenti discreti , ma si ritiene che sia possibile sviluppare un proprio modello con un SoC contente una npu abbastanza potente e stando attenti a utilizzare con parsimonia le risorse energetiche necessarie allo svolgimento del compito sostenuti da una piccola batteria.

Looking into the bus

Per motivi di tempistiche non è stato possibile collegarsi al bus dell'auto in quanto il reverse engineer , lo sviluppo di strumenti atti a leggere e scrivere il bus contenente il protocollo KWP2000 del modello del opel corsa b in studio avrebbe portato via più del tempo di sviluppo utilizzato.

Tuttavia è ritenuto fattibile dall'autore implementare una connessione tra l'edge e il can bus utilizzando il software SocketCan e un economico modulo per il CAN-BUS protocol, protocollo molto diffuso nel parco macchine.

So long, and thanks for all the fish

INDICE DELLE FIGURE

Figura 1.1 Schema di collegamento architettura edge	5
Figura 1.2 Schema logico del cloud	6
Figura 1.3 Primo piano del prototipo assemblato, dove sono visibili LilyGo SIM7600E, DFRobot voice recognition e antenne 4g, lte	9
Figura 1.4 Primo piano dove sono visibili il Nicla Voice, il modulo DFplayer, lo speaker da 2 watt e i collegamenti disponibili per i relè	10
Figura 1.5 Panoramica completa del prototipo assemblato	11
Figura 1.6 specifiche del server di sviluppo	12
Figura 1.7 Opel corsa 1997 1.0i 12V	13
Figura 2.1 Batteria 18650 agli ioni di litio	20
Figura 2.2 Batteria piombo acido	21
Figura 2.3 Schema di alimentazione 24V->5V con step-down converter	22
Figura 2.4 Progetto case in 3D per la board master per gentile concessione di Luca Spada.	23
Figura 2.5 Lilygo T-sim7600E	24
Figura 2.6. Specifiche dell'ESP32 WROVER-E	25
Figura 2.7. Pinout Nicla Voice	27
Figura 2.8 DFRobot Gravity	28
Figura 2.9 DFPlayer Mini	29
Figura 3.1. Locazione porta OBD-II	31
Figura 3.2 Pinout porta OBD-II	32
Figura 3.3. Schema di riferimento edge	33
Figura 3.4 Dettagli schema edge	34
Figura 3.5 proof of concept dei comandi vocali	35
Figura 3.6 proof of concept alimentazione , board master e antenne	36
Figura 3.7 Schema di alimentazione 24V->5V con step-down converter	37
Figura 3.8 Schema generico di funzionamento relè	38

Figura 3.9 Pulsantiera finestrini Opel corsa b	39
Figura 3.10 Prospettiva case 3D LilyGO T-SIM7600E(1)	40
Figura 3.11 Prospettiva case 3D LilyGO T-SIM7600E(1)	40
Figura 3.12 Prospettiva case 3D LilyGO T-SIM7600E(1)	41
Figura 3.13 Prospettiva case 3D LilyGO T-SIM7600E(1)	41
Figura 3.14 Prospettiva case 3D LilyGO T-SIM7600E(1)	41
Figura 4.1 Comando nvtop per benchmark gpu	51
Figura 4.2 Conversazione con client chatty per python	51
Figura 4.3 Chat testuale su pagina web	52
Figura 4.4 Chat testuale con risposta vocale su pagina web	52

BIBLIOGRAFIA

Repositories

Progetto

<https://github.com/LorenzoSiena/Johnny-The-CyberCar-Assistant>

Plugin Local Whisper Cat

https://github.com/LorenzoSiena/local_whisper_cat

Client Arduino

<https://github.com/LorenzoSiena/chattino>

Client Python

<https://github.com/LorenzoSiena/chatty>

Cheshire Cat AI

<https://cheshire-cat-ai.github.io/docs/>

LangChain

<https://en.wikipedia.org/wiki/LangChain>

Batterie

https://en.wikipedia.org/wiki/Lead-acid_battery

https://en.wikipedia.org/wiki/Lithium-ion_battery

Board e moduli

https://wiki.dfrobot.com/DFPlayer_Mini_SKU_DFR0299

<https://docs.arduino.cc/hardware/nicla-voice>

https://wiki.dfrobot.com/SKU_SEN0539-EN_Gravity_Voice_Recognition_Module_I2C_UART

<https://github.com/Xinyuan-LilyGO/T-SIM7600X>

Hardware Server

<https://www.nvidia.com/en-gb/geforce/graphics-cards/geforce-gtx-1080-ti/specifications/>

Software Server

Debian GNU/Linux

<https://www.debian.org/>

Docker e Docker-compose

<https://github.com/docker>

Ollama

<https://github.com/ollama/ollama>

Llama3

<https://github.com/meta-llama/llama3>

Cloudflare Tunnel client

<https://github.com/cloudflare/cloudflared>

Cheshire Cat The Ai assistant Framework

<https://github.com/cheshire-cat-ai/core/>

PiperCat

<https://github.com/pazoff/Piper-Cat/>

Local Whisper Cat

https://github.com/LorenzoSiena/local_whisper_cat/

Qdrant

<https://github.com/qdrant/qdrant>

Specifiche Opel Corsa b 1997

<https://www.auto-data.net/it/opel-corsa-b-facelift-1997-1.0i-12v-54hp-2111>

https://it.wikipedia.org/wiki/Opel_Corsa_B

Centralina (ECU)

https://en.wikipedia.org/wiki/Engine_control_unit

Can protocol

<https://www.kernel.org/doc/html/next/networking/can.html>

https://en.wikipedia.org/wiki/CAN_bus

Rag

<https://aws.amazon.com/it/what-is/retrieval-augmented-generation/>

Librerie software

https://github.com/DFRobot/DFRobot_DF2301Q/tree/master

<https://github.com/Makuna/DFMiniMp3>

OBD-II

https://en.wikipedia.org/wiki/On-board_diagnostics

NPU

https://en.wikipedia.org/wiki/AI_accelerator

V2X

<https://en.wikipedia.org/wiki/Vehicle-to-everything>

<https://en.wikipedia.org/wiki/Neuralink>

The GNU General Public License v3.0

<https://www.gnu.org/licenses/gpl-3.0.en.html>

Artificial Intelligence Act (AI Act)

https://www.europarl.europa.eu/doceo/document/TA-9-2024-0138-FNL-COR01_EN.pdf

<https://artificialintelligenceact.eu/high-level-summary/>

Ringraziamenti

Voglio ringraziare la mia famiglia, i miei fratelli, le mie cugine, i miei amici, i ragazzi del garage, i colleghi, il mio relatore e chiunque abbia creduto in me in questa folle avventura.

Uno speciale ringraziamento va a Luca Spada , Alessandro Spallina
e a Giorgio Arena per l'aiuto nello sviluppo del progetto.

Infine un grazie alla Free Software Foundation, a Wikipedia, a Youtube e alla community open source e open hardware senza il quale tutto questo non sarebbe mai stato possibile.